ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Кучеров Николай Николаевич

РАЗРАБОТКА МАТЕМАТИЧЕСКИХ МЕТОДОВ МОДЕЛИРОВАНИЯ, ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ БОЛЬШОЙ РАЗРЯДНОСТИ С ВЫСОКОЙ НАДЕЖНОСТЬЮ В ОБЛАЧНОЙ СРЕДЕ НА ОСНОВЕ СИСТЕМЫ ОСТАТОЧНЫХ КЛАССОВ

Специальность 05.13.18 —
Математическое моделирование, численные методы и комплексы программ

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
Заслуженный деятель науки и техники РФ,
доктор технических наук, профессор
Червяков Николай Иванович

Ставрополь — 2018

Оглавление

Введение

В основе облачных вычислений лежат принцип коммунального хозяйства предложенный в 1970 году J.C.R. Licklider и J. McCarthy. Данный принцип позволяет рассматривать IT-инфраструктуру в качестве услуги (сервиса) предоставляемого по требованию пользователя.

В настоящее время облачные вычисления широко используются при развертывании вычислений с интенсивной обработкой данных, к примеру, научные вычисления. Облачные сервисы позволяют хранить, удалять и гибко обрабатывать большие объемы данных, а так же производить генерацию наборов файлов большой размерности.

Научные приложения, как правило, используют ресурсы не только для высокопроизводительных вычислений, но и для хранения больших объемов данных [46]. При проведении исследований и экспериментов в астрономии [47], физике высоких энергий [81] и биоинформатики [118], требуется хранение и обработка больших массивов данных. В результате обработки и анализа информации, большое количество новых данных получается в качестве промежуточных или конечных результатов [46]. Научные приложения требуют обработки большого количества информации [48, 68], где размер отдельных получаемых файлов может составлять нескольких гигабайт или даже десятков гигабайт. Согласно исследованию из работы [101] объем научных данных будет удваиваться ежегодно.

В связи с экспоненциальным увеличением обрабатываемых данных часто применяются распределенные системы для обработки и хранения больших объемов данных [90, 103]. Алгоритмы обработки больших массивов данных в облаках схожи с алгоритмами используемыми в распределенных вычислениях [109, 111]. Данные подходы являются неоднородными в применении в вычислительных структурах. Технология управления данными в сети [40], может стать ориентиром управления данными в облаке.

Грид-технология была очень популярна в конце 1990-х и начале 2000-х годов, так как она подходит для масштабных вычислений и ресурсоемких приложений. Были разработаны и с успехом применялись многие системы управления данными, и некоторые из них используются в настоящее время в научных приложениях [23, 31, 40, 68, 80, 99, 102, 107].

Современные астрономические телескопы выдают большие объемы цифровых данных. Для исследования небесных объектов на телескопах различных обсерваторий, используются различные наблюдательные приборы, каждый из которых связан с определенным компьютерно-аппаратным комплексом. Для разных систем сбора информации, форматы цифровых данных имеют свой набор параметров для описания наблюдений и отличаются друг от друга.

Основными требованиями для построения систем хранения и обработки научной информации являются:

1. Обработка и хранения пакетов данных большого объема [116].
2. Высокая скорость передачи и обработки информации.
3. Высокая надёжность хранения.
4. Оплата за использованные ресурсы.

При работе таких установок как Европейский рентгеновский лазер на свободных электронах XFEL (X-ray Free Electron Laser), Большой адронный коллайдер (CERN), коллайдер НИКА (Дубна) и других научных систем, уже получены сотни петабайт экспериментальных данных в области физики элементарных частиц, биоинформатики, геофизики и др., причем объемы получаемых данных будут расти и скоро достигнут экзабайтной отметки. Вся полученная в ходе экспериментов информация должна быть доступна всем членам научных коллабораций и коллективов компаний, где участники коллективов почти всегда географически распределены. В этой ситуации задача надежного хранения и управления метаданными становится фундаментальной и отсутствие ее адекватного решения приводит к экономическим и функциональным потерям

При использовании облачными услугами основным риском является потеря данных без возможности восстановления в следствии сбоя сервера, ошибки администратора или вмешательства извне [32, 33]. Недоступность к хранимым данным в течении непродолжительного времени или получение недостоверных данных могут привести к большим финансовым потерям.

Для решения проблемы надежности хранимых и обрабатываемых данных российскими и зарубежными учеными предлагается использование системы остаточных классов. Система остаточных классов (СОК) за счет природной параллельности операции и работы с более меньшими числами позволяет строить надежные схемы хранения и обработки информации в облаках. Введение избыточности СОК позволяет производить поиск, локализацию и исправление возникающих при передачи ошибок, повышая тем самым достоверность информации.

Распределенное хранение данных на облачных серверах, когда для хранения используется n серверов из которых рабочие k серверов, позволяет сохранять доступность данных при выходе из строя $n-k$ серверов системы.

Для повышения надежности хранения данных используют репликацию данных [15,56]. Недостатками большинства используемых систем являются возможная временная недоступность к хранимой и обрабатываемой информации, отсутствие аппарата оценивания достоверности информации и исправления возникающих ошибок. Поэтому при использовании данными системами пользователи могут сталкиваться с перечисленными проблеммами. Таким образом, является актуальным использование алгоритмов, не требующих значительных вычислительных ресурсов осуществляющих хранение и обработку данных большой разрядности с высокой надежностью, достоверностью и постоянной доступностью, в таких случаях можно прибегнуть к схемам распределения данных основанных на принципах модулярной арифметики.

В целях повышения надежности и достоверности обработки и хранения данных целесообразно применять схемы распределенного хранения данных основанные на принципах модулярной арифметики. При организации хранения и обработки данных в облачной среде используются схемы распределенного хранения данных основанные на одноуровневой или многоуровневой СОК. Основными вопросами при реализации систем хранения и обработки данных с использованием СОК являются: определение переполнения диапазона, определение знака числа, повышение надежности, достоверности и целостности данных. Если все данные хранятся в облаке, и находятся в закодированном виде, для эффективной обработки таких данных необходимо наличия решения таких вопросов, как доступность данных и управление третьей стороной. Для эффективной обработки закодированных данных необходимо исключить доступ третьих лиц к данным. Поэтому возникает необходимость разработки эффективных методов и алгоритмов повышающих надежность обрабатываемых и хранимых данных.

Одним из перспективных направлений модулярной арифметики является разработка математических методов для хранения и обработки данных большой разрядности с высокой надежностью в облачной среде.

Основным инструментом повышения показателей надежности является введение регулируемой избыточности в систему.

Известно три основных подхода к повышению надежности хранения и обрабатываемых данных: репликация, применение сложных позиционных кодов и

применение корректирующих кодов в модулярной арифметике [12]. Применение репликации данных приводит к большой избыточности, а применение позиционных кодов к отсутствию возможности контроля выполнения арифметических операций и достоверности данных. Использование СОК за счет введения регулируемой избыточности позволяет производить поиск, локализацию и исправление ошибок, а введение совместного использования модулярной арифметики и схем распределенного хранения данных позволяет использовать СОК для повышения надежности и достоверности хранимых и обрабатываемых данных. Эта особенность модулярной арифметики широко применяется для решения проблемы повышения отказоустойчивости вычислительных структур и является мощным инструментом для автоматического обнаружения, локализации и коррекции ошибок.

Признавая важность исследований в рассматриваемой области, отметим, что научных работ посвященных сложным и многообразным проблемам теории и практики модулярной арифметики, реализуемой в области облачных вычислений, явно недостаточно. Кроме того, недостаточно рассмотрены вопросы построения надежных схем хранения и обработки данных большой размерности с достоверностью и постоянной доступностью.

Значительный научный вклад в теорию и практику облачных и параллельных вычислений внесли отечественные и зарубежные исследователи: И.Я. Акушский, Д.И. Юдицкий, В.М. Амербаев, Н.И. Червяков, И.А. Калмыков, О.А. Финько, G. Alonso, R. Buyya, Ji. Chen, A. Chervenak, C. Gentry, M. Gomathisankaran, A. Omondi, A. Premkumar, P. Paillier, A. Tchernykh, L. Yang, D. Zhang и другие.

Таким образом, как с теоретической, так и с практической точки зрения следует признать необходимость в исследовании вышеназванных проблем, носящих актуальный характер.

Существующие потребности обработки данных большой размерности обуславливают противоречие в практике, состоящее в том, что с одной стороны существует объективная необходимость в обработке данных большой размерности, с другой стороны существующие разрядные сетки современных ЭВМ не позволяют представить такие данные. Одним из путей преодоления вышеприведенного противоречия в практике является применение модулярной арифметики.

С целью соблюдения требований надежности предъявленных к хранимых и обрабатываемых данных, в работе разработана новая архитектура для перевода чисел и СОК в ПСС.

Исходя из вышеизложенного, разработка методов и алгоритмов надежного хранения и обработки данных в облачной среде с регулируемой избыточностью и высокой достоверностью, является актуальной научно-исследовательской задачей.

Цель диссертационного исследования – повышение отказоустойчивости и надежности схем обработки и хранения данных в облачных сервисах с регулируемой избыточностью.

Объект исследования – облачные инфраструктуры хранения данных.

Предмет исследования – математические модели, методы и алгоритмы хранения и обработки данных в облачных хранилищах.

Научная задача – разработка новых математических методов и моделей хранения и обработки данных большой разрядности с использованием модулярной арифметики в облачных хранилищах с регулируемой избыточностью.

Для решения поставленной общей научной задачи была произведена ее декомпозиция на ряд частных задач:

1. Аналитический обзор современных облачных хранилищ данных большой разрядности.
2. Разработка математической модели, методов и алгоритмов системы надежного, длительного хранения данных большой разрядности в мультиоблачной среде на базе системы остаточных классов с регулируемой избыточностью.
3. Разработка математической модели синтеза и анализа многоуровневых облачных систем хранения данных следующего поколения.
4. Модификация численного метода Червякова для перевода чисел из системы остаточных классов в позиционную систему счисления за счет использования ранга числа Акушского.
5. Разработка среды моделирования распределенного, длительного хранения данных большой разрядности в облачной среде.

Методы исследования базируются на использовании математического аппарата высшей алгебры, теории чисел, теории алгоритмов, численных методов, теории вероятности, теории надежности, математическом моделировании, системном анализе и теории приближенных вычислений.

На защиту выносятся следующие научные результаты:

1. Математическая модель системы надежного и длительного хранения данных в мультиоблачной среде.
2. Методы и алгоритмы надежного и длительного хранения данных в мультиоблачной среде на базе системы остаточных классов с регулируемой избыточностью.
3. Математическая модель синтеза и анализа многоуровневых облачных систем хранения данных следующего поколения.
4. Численный метод вычисления ранга числа для эффективного перевода чисел из системы остаточных классов в позиционную систему счисления.
5. Комплекс программ моделирования системы обработки и хранения данных большой разрядности с высокой надежностью и регулируемой избыточностью.

Научная новизна:

1. Разработана математическая модель и метод надежного хранения данных в мультиоблачной среде на базе системы остаточных классов с регулируемой избыточностью.
2. Разработаны методы и алгоритмы надежного и длительного хранения данных в мультиоблачной среде на базе системы остаточных классов с регулируемой избыточностью.
3. Разработана математическая модель синтеза и анализа многоуровневых облачных систем хранения данных следующего поколения
4. На основе разработанных моделей предложен новый численный метод вычисления ранга числа для эффективного перевода чисел из системы остаточных классов в позиционную систему счисления.
5. Комплекс программ, моделирования системы обработки и хранения данных большой разрядности с высокой надежностью и регулируемой избыточностью.

Достоверность результатов обеспечивается корректным и обоснованным применением методов математического моделирования и строгостью проводимых математических доказательств. Справедливость выводов относительно эффективности предложенных моделей и методов подтверждена математическим моделированием на базе разработанной модели надежного хранения данных в облачной среде.

Практическая ценность результатов состоит в возможности реализации системы надежного распределенного хранения данных, основанных на СОК, на базе разработанных методов, что способствует снижению избыточности и уменьшению затрат на содержание. Разработан универсальный метод выполнения операций в СОК, позволяющий расширить возможности использования системы остаточных классов при проектировании систем хранения данных. Разработанные программные продукты и аппаратные решения, зарегистрированные в соответствующем порядке, способствуют оптимизации эксплуатационных возможностей облачных вычислений.

Внедрение. Результаты диссертационного исследования используются в учебном процессе в СКФУ на кафедре прикладной математики и математического моделирования в дисциплинах «Приложения системы остаточных классов в информационных технологиях» и «Основы модулярной арифметики», что подтверждено Актом об использовании результатов работы в учебном процессе от 17.05.2018. Основные научные результаты использованы в опытно-конструкторских ООО «Инфоком-С» при выполнении договора на выполнение прикладных научных исследований и экспериментальных разработок №1909/16 по теме «Разработка средств высокоскоростной обработки данных информационных сенсоров в системах ситуационного управления» (Акт №101 от 04.10.2018). Кроме того, ряд результатов работы был использован при выполнении научно-исследовательских работ в базовой части государственного задания СКФУ №2.6035.2017/БЧ «Разработка математических моделей и методов снижения энергопотребления в системах мобильной связи на основе системы остаточных классов».

Апробация работы. Основные результаты работы были представлены на «International Conference on High Performance Computing & Simulation (HPCS 2018)» (г. Орлеан, Франция), «1st International Workshop on Uncertainty in Cloud Computing, in conjunction with 28th International Conference on Database and Expert Systems Applications (DEXA’17)IEEE» (г. Лион, Франция, 2017 г.), Tomsk IEEE Chapter & Student Branch of The Institute of Electrical and Electronics Engineers «International Siberian Conference on Control and Communications SIBCON-2017» (г. Астана, Казахстан, 2017 г.), «ISUM 2017 – 8th International Supercomputing Conference in Mexico» (г. Гвадалахара, Мексика, 2017 г.), «ISUM 2016 — 7th International Supercomputing Conference in Mexico» (г. Пуэбло, Мексика, 2016 г.), «II International Conference Engineering & Telecommunication –

En&T 2015» (г. Москва, Россия, 2015 г.), Proceedings of the First International Scientific Conference «Intelligent Information Technologies for Industry (IITI'16)» (г. Ростов-на-Дону – г. Сочи, Россия, 2016 г.), «6th International Conference, ICSI 2015 held in conjunction with the Second BRICS Congress (CCI 2015)» (г. Пекин, Китай, 2015 г.), XVII International School-Seminar «Optimization Methods and their Applications», (г. Иркутск, Россия, 2017 г.) «Information Technology and Systems 2015 (ITIS 2015)» (г. Сочи, Россия, 2015 г.).

Публикации по теме диссертации. Содержание диссертации опубликовано в 24 работах, среди которых имеются статьи в рецензируемых научных изданиях, входящих в перечень ВАК при Министерстве образования и науки России, а также входящих в системы индексирования научных работ Scopus и Web Of Science.

Личный вклад соискателя. Все изложенные в работе результаты исследований получены при непосредственном участии автора. Авторским вкладом являются разработка методов, моделей и алгоритмов хранения данных в облачной среде, разработка программного комплекса моделирования надежного хранения данных в облачной среде основанной на системе остаточных классов.

Структура и объем диссертации. Работа состоит из введения, четырёх глав, заключения, приложений, списка сокращений и обозначений, а также списка использованной литературы, содержащего 118 наименований наименований. Основная часть работы содержит 237 страниц машинописного текста, включая 38 рисунков и 26 таблиц.

Краткое содержание работы.

Во введении обоснована актуальность темы диссертации, сформулированы цель и задачи работы, выбраны объект и предмет исследования, показана научная новизна, практическая и теоретическая ценность полученных результатов, приведены основные положения, выносимые на защиту.

Первая глава посвящена анализу организации архитектуры облачных систем хранения данных. методов и моделей хранения и обработки данных в облачных системах. Рассмотрены наиболее распространенные решения применяемые для хранения больших данных в облачной среде. Модель хранения и обработки больших данных в облачной среде MapReduce разработана компанией Google, и используется для параллельных вычислений. Обработка данных в модели MapReduce состоит из двух этапов. Вначале производится предварительная обработка, разделение данных на части и распределение между рабочими

узлами, а после производится свертка обработанных данных на основе предоставленных отчетов. Преимущество MapReduce заключается в возможности распределенно производить операции предварительной обработки и свертки. Операции предварительной обработки могут производиться параллельно так, как работают независимо друг от друга. Основными недостатками методов является низкая отказоустойчивость и надежность. Для повышения отказоустойчивости и надежности хранения и обработки данных в облачной среде применяют различные подходы: резервирование (холодное, горячее), архивирование и распределение нагрузки. Применение данных подходов позволяет повысить отказоустойчивость и надежность облачной системы. Основными недостатками применяемых подходов является увеличение избыточности данных и повышение нагрузки на систему обработки данных. Для повышение отказоустойчивости и надежности при хранении и обработки данных применяется подход основанный на схеме распределения данных (СРД). Для увеличения быстродействия, надежности и отказоустойчивости применим подход основанный на СРД и системе остаточных классов (СОК). Введение не большой избыточности СОК позволяет производить поиск, локализацию и исправление возникающих ошибок

Во **второй главе** приведены и разработаны методы надежного и отказоустойчивого хранения данных в облачной среде на основе СОК совместно со СРД. Совместное использование избыточной СОК и СРД позволяет производить разделение и распределение данных между облачными серверами. Применение данного подхода позволяет хранить и обрабатывать данные на различных серверах у различных облачных провайдеров. Это дает преимущество в отказоустойчивости и надежности хранимых и обрабатываемых данных. На основе одноуровневой СОК была разработана модель надежного и отказоустойчивого хранения данных в облачной среде. Проведенно моделирование и сравнительный анализ разработанной системы, который показал преимущество системы построенной на основе СОК над системами основанными на позиционной системе счисления (ПСС), вероятность отказа и безвозвратной потери данных уменьшилась в 1,6 раз, избыточности данных в 2,2 раза.

Для повышения скорости обработки, надежности и отказоустойчивости разработанной модели применим двухуровневую СОК. Использование двухуровневой СОК при проектировании мультиоблачной системы позволяет разделить задачи каждого из уровней. Основные вычисления производятся на втором уровне, следовательно, необходимо подбирать для него систему оснований так,

чтобы арифметические операции на нем были максимально эффективны. Этого можно добиться, используя специальные наборы оснований.

В **третьей главе** разработан численный метод перевода чисел из СОК в ПСС. Приведен подход повышения отказоустойчивости облачного сервера путем перераспределения обрабатываемых данных между рабочими и контрольными серверами. Особенность построения модели хранения и обработки данных позволяет строить облачную систему с возникающими отказами рабочих и контрольных серверов функционирующих в СОК. При отказе серверов производится реконфигурация облачной системы, с целью исключения всех отказавших серверов системы.

Предложена эффективная реализация алгоритма обнаружения ошибок с использованием приближенного метода. В результате моделирования методов обнаружения ошибок получили, что при сравнительно одинаковой занимаемой площади временная задержка приближенного метода при использовании модулей размерности до 64 бит в 1,2 раза меньше, а при использовании модулей размерности больше 64 бит задержка меньше в 1,3 раза в сравнении с методом, основанным на КТО.

Разработана отказоустойчивая модель обработки и хранения данных в облачной среде на основе двухуровневой СОК. Проведен сравнительный анализ разработанных систем на основе СОК, который показал преимущество систем построенных базе двухуровневой СОК и повышение надежности хранимых данных в 1,2 раза.

В **четвертой главе** разработана среда моделирования распределенного хранения больших данных в облачной среде «ClouStorageSim», которая позволяет производить моделирование хранения данных на реальных вычислительных инфраструктурах. При помощи разработанной среды моделирования можно оценивать такие важные показатели облачных систем как: вероятность потери данных, избыточность, скорость выгрузки/загрузки, скорость перевода данных из СОК в ПСС, производить моделирование возникающих сбоев и ошибок. Разработанная модель является настраиваемой, для определения и конфигурации параметров необходимо учитывать следующие критерии: точность, масштабируемость, надежность и производительность.

Анализ полученных зависимостей свидетельствует о преимуществе в надежности алгоритмов хранения и обработки данных на основе СОК перед существующими системами при существенном выигрыше в избыточности. Таким

образом, при использовании алгоритмов обработки и хранения данных на основе СОК можно получить значительное увеличение надежности системы без дополнительных аппаратурных затрат при снижении избыточности.

Оценка надежности алгоритмов хранения и обработки данных в облачной среде построенных на базе СОК и сравнение их с аналогичными по качественным функциональным характеристикам существующих позиционными системами свидетельствует об их существенном преимуществе, что объясняется наличием эффекта поэлементно скользящего резервирования. Соотношение избыточного оборудования позиционных и модулярных схем обработки и хранения данных для различных параметров парируемых отказов.

В **заключении** подведены подведены итоги и обобщены результаты проведенных исследований.

В **приложениях** приведен листинг разработанных программ на языке C#/C++, позволяющих моделировать системы надежного хранения данных в облачной среде на основе системы остаточных классов.

Автор выражает искреннюю благодарность научному руководителю – доктору технических наук, заслуженному деятелю науки и техники РФ, академику МАИ, создателю и руководителю научной школы «Нейроматематика, модулярные нейрокомпьютеры и высокопроизводительные вычисления», почётному профессору Северо-Кавказского федерального университета Николаю Ивановичу Червякову, а также сотрудникам кафедры высшей математики и кафедры прикладной математики и математического моделирования Северо-Кавказского федерального университета за помощь, оказанную при написании диссертации, и критические замечания, высказанные при ее обсуждении.

Глава 1. АНАЛИТИЧЕСКИЙ ОБЗОР МЕТОДОВ И МОДЕЛЕЙ ХРАНЕНИЯ ДАННЫХ В ОБЛАЧНЫХ СИСТЕМАХ

1.1 Анализ организации архитектуры облачных систем для хранения данных

Идея облачных вычислений появилась еще в 1960 году, когда Джон Маккарти высказал предположение, что когда-нибудь компьютерные вычисления будут производиться с помощью «общенародных утилит». Считается, что идеология облачных вычислений начала бурный рост с 2007 года благодаря быстрому развитию каналов связи и стремительно растущим потребностям пользователей.

Облачными вычислениями принято считать способ обработки данных с использованием аппаратных и программных ресурсов, которые представлены в виртуализованном виде, загружаются динамически и не имеют ограничений по масштабированию, а так же предоставление пользователю компьютерных ресурсов и мощностей в виде интернет-сервиса.

Облачные вычисления и облачные системы хранения данных завоевали популярность как наиболее удобный способ передачи информации и предоставления функциональных средств в Интернете. Одни облачные службы предлагают широкий спектр услуг и функций индивидуальным потребителям (интернет-магазины и мультимедийные онлайн-технологии, социальные сети, среды для интернет-коммерции), другие обеспечивают работу коммерческих структур – предприятий малого и среднего бизнеса, крупных корпораций, государственных и прочих учреждений.

Некоторые облачные службы предоставляют потребителям пространство для хранения и использования данных бесплатно, другие взимают ту или иную оплату за услуги, предоставляемые по подписке. Существуют также частные облака, которыми владеют и управляют организации. По сути, это защищенная сеть для хранения и совместного использования критически важных данных и программ.

В основе всех облачных служб, продуктов и решений лежат программные средства, которые по функциональности можно разделить на три типа – средства

для обработки данных и выполнения приложений (серверы вычислений), для перемещения данных (сети) и для их хранения (серверы хранения данных).

Облачные решения – это средства создания, хранения и обработки содержимого, а также стратегии, определяющие, где и как это содержимое использовать. Такие решения используются при создании виртуальных инфраструктур, в которых организация любого масштаба может разместить необходимые приложения и средства, а также сред разработки и тестирования новых возможностей. Обычно они включают службы или продукты (оборудование, программное обеспечение и сети), оплачиваемые по факту использования, а также решения, которые можно купить и установить в собственной среде.

В настоящее время большую популярность набирают облачные сервисы, Google, Amazon, Dropbox, Microsoft OneDrive, предоставляющие облачные услуги по хранению и обработке данных. Основной причиной использования облачных продуктов является удобность и доступность предоставляемых услуг. Благодаря использованию облачных технологий возможна экономия финансовых затрат на содержание и обслуживание серверов по хранению и обслуживанию информации. Все проблемы возникающие при хранении и обработке информации перекладываются на облачного провайдера [4].

Облачными вычислениями принято считать способ обработки данных с использованием программных ресурсов, которые представлены в виртуализованном виде, загружаются динамически и не имеют ограничений по масштабированию, а также предоставление пользователю компьютерных ресурсов и мощностей в виде интернет-сервиса [30].

В основе всех облачных служб, продуктов и решений лежат программные средства, которые по функциональности можно разделить на три типа – средства для обработки данных и выполнения приложений, для перемещения данных и для их хранения.

Существенное увеличение объемов информации является отличительной чертой современного мира. При работе таких установок как Европейский рентгеновский лазер на свободных электронах XFEL (X-ray Free Electron Laser), Большой адронный коллайдер (CERN), коллайдер НИКА (Дубна) и других научных систем, уже получены сотни петабайт экспериментальных данных в области физики элементарных частиц, биоинформатики, геофизики и др., причем объемы получаемых данных будут расти и скоро достигнут экзабайтной отметки. Ведущими компаниями рынка информационных технологий обрабатывают-

ся огромные массивы данных: Google (поисковый индекс 100 Пбайт), Facebook (180 Пбайт), YouTube (15 Пбайт в год). Вся полученная в ходе экспериментов информация должна быть доступна всем членам научных коллабораций и коллективов компаний, где участники коллективов почти всегда географически распределены. В этой ситуации задача надежного хранения и управления метаданными становится фундаментальной и отсутствие ее адекватного решения приводит к экономическим и функциональным потерям.

Обработка и хранение больших данных являются важными задачами. Необходимо обрабатывать данные в режиме реального времени [62]. Большинство систем хранения больших данных построено на дисках, а обработка ведется на нескольких компьютерах или серверах, собранных в кластер. Технология «in-memory computing» позволяет в рамках такого же кластера использовать оперативную память компьютеров. С точки зрения доступа и обработки, оперативная память становится в тысячи, а то и в миллионы раз быстрее, чем диски. Архитектура современных компьютеров не предусматривает других возможностей для этой цели [28].

Распределенная инфраструктура представляет условия, в которых конкуренция за ресурсы между высокоприоритетными вычислительными задачами анализа данных происходит регулярно [9]. Неизбежно, перегруженность вычислительных ресурсов вызывает ухудшение функционирования обслуживающих сервисов, а порой и длительные перерывы в их работе. По этой и другим причинам в распределенной обработке данных неизбежно происходит непрерывный поток отказов, ошибок и неисправностей [64].

Анализ проблем организации хранения и обработки данных показал, что главной задачей системами хранения, является задача обработки данных в режиме реального времени, однако, возникает противоречие между практической потребностью обработки данных большой разрядности в режиме реального времени, ограниченностью аппаратных ресурсов современных систем, стоимостью, надежностью и производительностью. Таким образом, необходимо обеспечить повышение скорости выполнения арифметических вычислений, надежности и доступности за счет разработки новых математических моделей обработки данных в облачной среде, алгоритмы которых используют методы, сокращающие время их работы, а также разработать новые алгоритмы обработки и хранения данных позволяющие уменьшить финансовые издержки.

В целях повышения надежности и достоверности обработки и хранения данных целесообразно применять схемы распределенного хранения данных основанные на принципах модулярной арифметики. Одним из перспективных направлений модулярной арифметики является разработка математических методов для хранения и обработки данных большой разрядности с высокой надежностью в облачной среде. Основным инструментом повышения показателей надежности является введение регулируемой избыточности в систему.

Признавая важность исследований в рассматриваемой области, отметим, что научных работ, посвященных сложным и многообразным проблемам теории и практики модулярной арифметики, реализуемой в области облачных вычислений, явно недостаточно. Кроме того, недостаточно рассмотрены вопросы построения надежных схем хранения и обработки данных большой размерности с достоверностью и постоянной доступностью.

Когда поставщик облачных услуг хранит данные пользователя, он должен иметь возможность возвратить эти данные пользователю по требованию. С учетом простоев сети, ошибок пользователей и других обстоятельств выполнение этого условия надежным и детерминированным способом может оказаться затруднительным [63].

Например компания «Cleversafe», поставщик услуги хранения данных в частном облаке. Данная компания использует подход отличный от остальных компаний, она продает свои мощности в отличии от сдачи в аренду. Они используют СРД для повышения доступности данных перед лицом физических отказов и простоев сети они используют географическое распределение данных.

Алгоритм СРД [91], первоначально разработал Michael Rabin для телекоммуникационных систем [91], СРД позволяет разделить данные таким образом, что в случае потери или недоступности части данных возможно восстановить исходные данные.

Основная идея помехозащитного кодирования Рида-Соломона заключается в умножении информационного слова, представленного в виде полинома D, на неприводимый многочлен G, известный обеим сторонам, в результате которого получается кодовое слово $C = D \cdot G$ представленное в виде полинома. Декодирование осуществляется по алгоритму обратному приведенному выше: производится деление кодового слова C на полином G. Если в результате деления декодер получает не нулевой остаток, то он дает сигнал об произошедшей ошибки [69].

Если степень полинома G превосходит степень кодового слова на две степени, то может производится не только обнаружение, но и исправление одиночных ошибок. В случае если степень полинома G превосходит на k степень кодового слова, то в этом случае количество исправляемых ошибок равняется $t = \left\lfloor \frac{k}{2} \right\rfloor$.

Кроме того, согласно принципам СРД алгоритмов, для устранения одной ошибки, требуется увеличение объема данных в два раза, для исправления 8 ошибок увеличение объема в 4 раза. Как и RAID, СРД позволяет восстанавливать данные из подмножества исходных данных при некоторых накладных расходах на коды ошибок (рисунок 1.3).

Модель организации облачного хранилища данных предполагает, что она будет создаваться на основе модели частного облака. Особенно это утверждение справедливо в отношении крупных компаний, поскольку мало кто из них может доверить размещение корпоративной информации в публичном облаке.

В основе концепции облачного хранилища данных лежат две основные идеи:

1. Интеграция разъединенных детализированных данных (детализированных в том смысле, что они описывают некоторые конкретные факты, свойства, события) в едином хранилище.
2. Разделение наборов данных и приложений, используемых для оперативной обработки и применяемых для решения задач анализа.

Определение понятия «хранилище данных» первым дал Inmon H. William в своей монографии [67]. В ней он определил хранилище данных как «предметно-ориентированную, интегрированную, содержащую исторические данные, не разрушаемую совокупность данных, предназначенную для поддержки принятия управленческих решений».

Концептуально модель облачного хранилища данных можно представить в виде схемы, показанной на рисунке 1.1 [67,71]. Данные из различных источников помещаются в хранилище данных, а описания этих данных в репозиторий метаданных. Конечный пользователь, используя различные инструменты (средства визуализации, построения отчетов, статистической обработки) и содержимое репозитория, анализирует данные в хранилище. Результатом его деятельности является информация в виде готовых отчетов, найденных скрытых закономерностей, каких-либо прогнозов. Так как средства работы конечного пользователя с хранилищем данных могут быть самыми разнообразными, то теоретически их

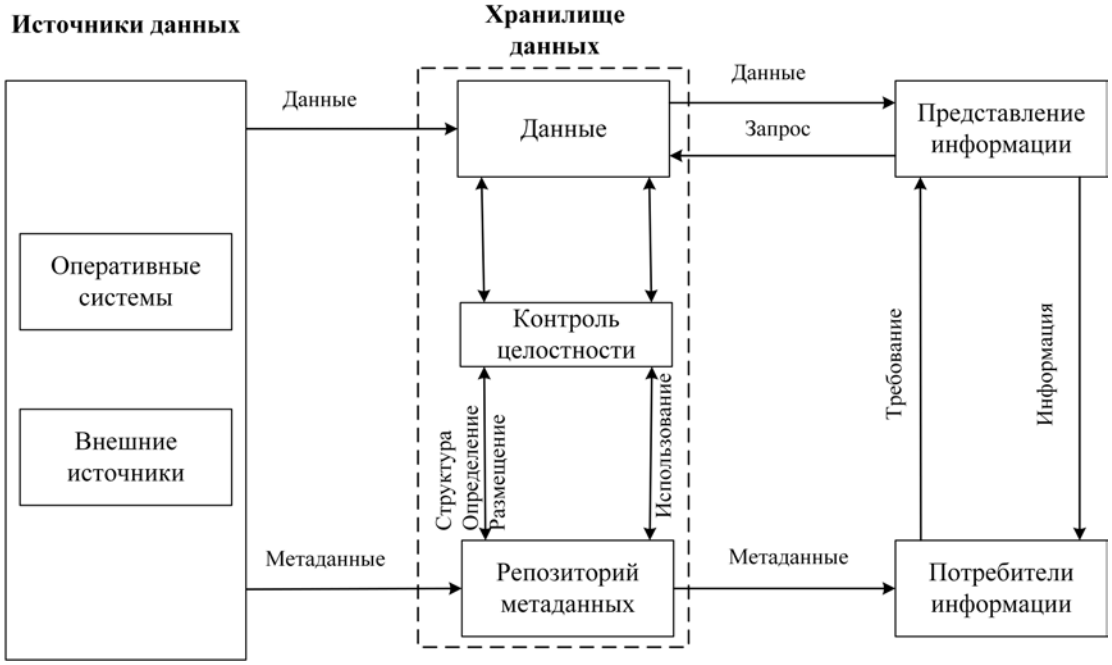

Рисунок 1.1 — Концептуальная модель архитектуры облачного хранилища данных

выбор не должен влиять на структуру хранилища и функции его поддержания в актуальном состоянии. Физическая реализация приведенной концептуальной схемы (рисунок 1.1) может быть самой разнообразной.

Построение полноценного хранилища данных обычно выполняется в трехуровневой архитектуре. На первом уровне расположены разнообразные источники данных – внутренние регистрирующие системы, справочные системы, внешние источники (данные информационных агентств, макроэкономические показатели). Второй уровень содержит центральное хранилище данных, куда стекается информация от всех источников с первого уровня, и, возможно, оперативный склад данных. Оперативный склад не содержит исторических данных и выполняет две основные функции. Во-первых, он является источником аналитической информации для оперативного управления и, во-вторых, здесь подготавливаются данные для последующей загрузки в центральное хранилище. Под подготовкой данных понимают их преобразование и осуществление определенных проверок. Наличие оперативного склада данных просто необходимо при различном регламенте поступления информации из источников. Третий уровень в описываемой архитектуре представляет собой набор предметно-ориентированных витрин данных, источником информации для которых является центральное хранилище данных. Именно с витринами данных и работает большинство конечных пользователей.

В числе явных преимуществ облачных хранилищ данных можно отметить то, что пользователь оплачивает фактически занятое место в них, не тратясь на аренду всего сервера. Кроме того, заказчик услуги не несет издержек, связанных с хранением данных и поддержкой соответствующей инфраструктуры. Немаловажно также, что ответственность за сохранность данных клиента несет облачный провайдер [68].

При построении большого облака, на котором работают все бизнес-системы компании, необходимо обеспечивать высокую отказоустойчивость, постоянно резервировать данные, чтобы при падении сервера облака не отключались одновременно, а моментально переключались на другой сервер, либо (в разумные сроки) восстанавливались из резервных копий, Все это ведет к росту затрат на создание отказоустойчивой архитектуры. Увеличиваются вложения в поддержание и создание резервных копий облаков.

Модель архитектуры облачных вычислений представлена на рисунке 1.2.

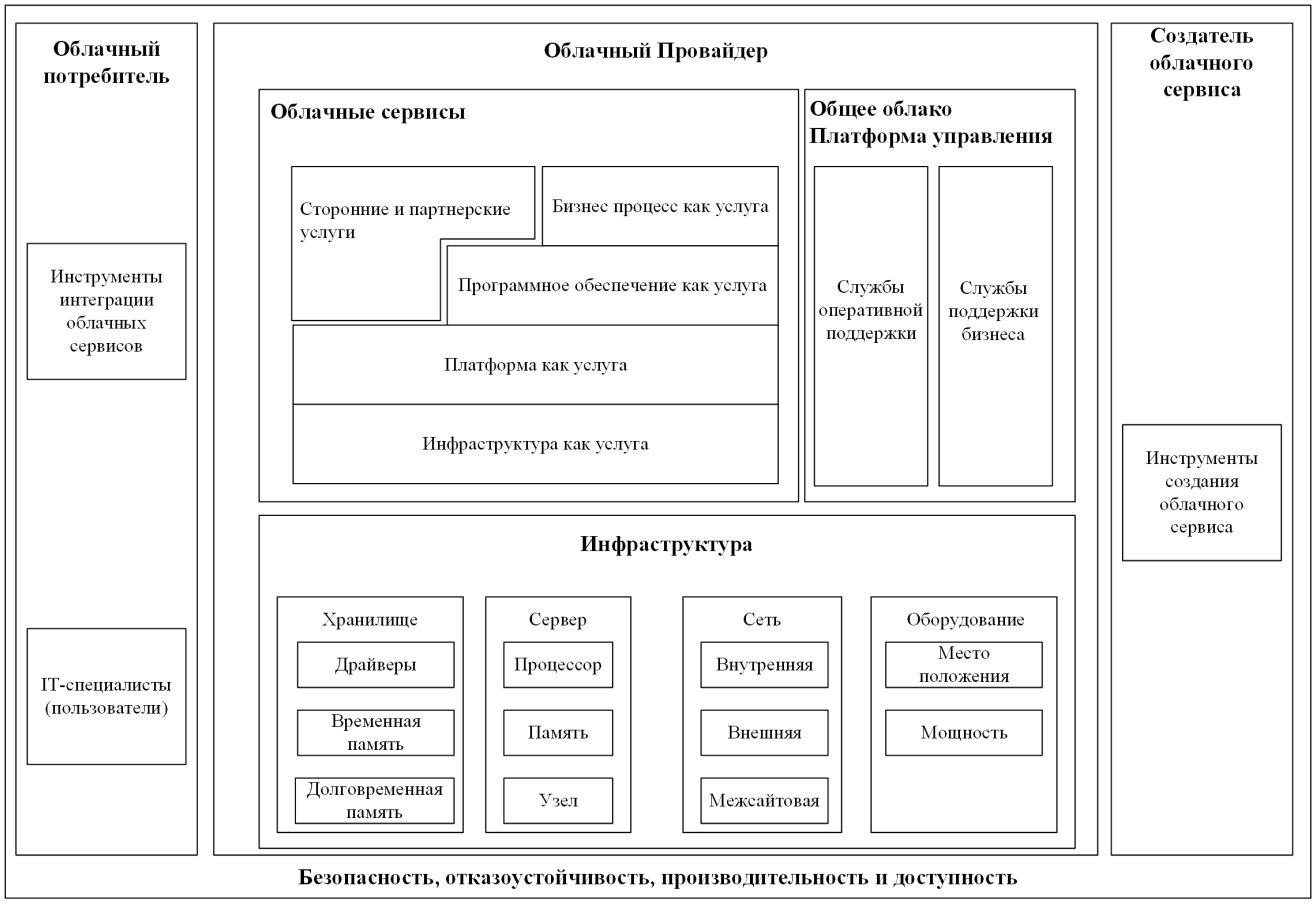

Рисунок 1.2 — Комбинированная концептуальная диаграмма референтной архитектуры облачных вычислений

Облачная архитектура хранения данных – это прежде всего доставка ресурсов хранения данных по требованию в высокомасштабируемой и мультите-

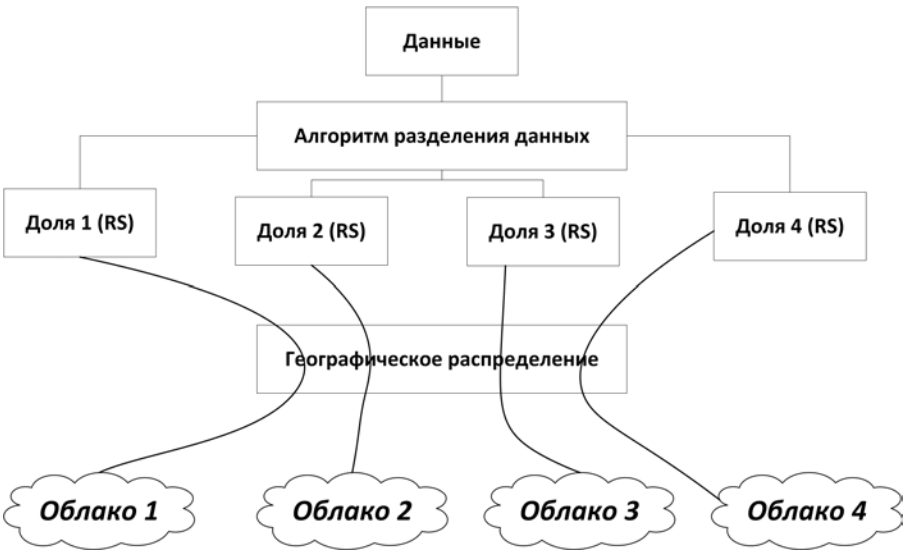

Рисунок 1.3 — Подход «Cleversafe» к обеспечению высокой готовности данных

нантной среде. Обобщенно облачная архитектура хранения данных представляет собой внешний интерфейс, который предоставляет API для доступа к накопителям 1.1. В традиционных системах хранения данных это протокол SCSI, но в облаке появляются новые протоколы. Среди них можно найти внешние протоколы Web-сервисов, файловые протоколы и даже более традиционные внешние интерфейсы (Internet SCSI, iSCSI).

На рисунке 1.3 представлена схема распределения данных используемая компанией «Cleversafe». Перед записью данных на диск, для хранения, происходит разделение данных по алгоритму СРД [91], затем происходит их распределение распределенными между серверами и запись данных в хранилища.

Оборотная сторона СРД – интенсивная обработка без аппаратного ускорения. Репликация [56] – метод который используют многие поставщики облачных услуг, за счет простоты его реализации. Основным недостатком репликации является увеличение издержек на электроэнергию, аренду помещений, хранение данных и управление этими ресурсами, а так же косвенных издержек на обеспечение сетевой инфраструктуры, инфраструктуры хранения данных и управление общей инфраструктурой, и накладных расходов при эксплуатации серверов более чем на 100%, так как необходимо производить управление, обеспечение и обслуживание большего количества жестких дисков.

Согластно теории, коммерческие облака предоставляют пользователям неограниченные ресурсы для вычислений и хранения данных [112], однако они несут ответственность за стоимость ресурсов, в связи с использованием модели оплаты по мере использования. Следовательно, пользователям нужна новая и

подходящая модель затрат, которая может представлять необходимую стоимость, соразмерную с использованием ресурсов своими приложениями в облаке [66].

При наличии крупных генерируемых наборов данных приложения в облаке, пользователям может быть предоставлен выбор, сохранить их для будущего использования или удалить их, чтобы уменьшить стоимость хранения. Различные стратегии хранения приводят к разным затратам на хранение и вычислительные ресурсы и наконец, к различной общей стоимости приложений. Кроме того, поскольку существуют зависимости между наборами данных приложений (например, задача вычисления может работать на одном или нескольких наборах данных и генерировать один или более новых наборов), затраты на хранение набора данных зависят не только от его собственной генерации и хранения, но и от состояния хранения предыдущих и последующих данных. Модель затрат должна быть в состоянии представить общую стоимость приложений, основываясь на компромиссе между обработкой и хранением данных в облаке, и при этом учитывать зависимости данных.

В связи с использованием модели затрат по мере использования в облаке, стоимость является одним из наиболее важных факторов, волнующих пользователей. Поскольку число генерируемых и хранимых наборов в облаке быстро растет, пользователи должны оценить экономическую эффективность стратегии их хранения [106]. Следовательно, поставщики услуг должны иметь возможность и необходимость, обеспечить бенчмаркинг-услуги (услуги по определению контрольных показателей), которые могут предоставить информацию о возможных минимальных затратах на хранение наборов данных приложения в облаке.

Расчет базовых минимальных затрат – это NP-трудная проблема, потому что существуют сложные зависимости между наборами данных в облаке. Кроме того, это значение затрат в облаке имеет динамическое значение. Это следствие динамического характера системы облачных вычислений: новые наборы данных могут быть сгенерированы в облаке в любое время, и частота использования наборов данных может также меняться с течением времени. Следовательно, значение минимальной стоимости может время от времени меняться. Для того чтобы гарантировать качество обслуживания (QoS) в облаке, должны быть организованы разные бенчмаркинг-подходы для разных ситуаций. Например, в некоторых случаях пользователям только нужно знать значение затрат перед выполнением приложения или периодически во время его выполнения. В этой ситуации, бенчмаркинг должен быть представлен в виде статических сервисов, которые

должны отвечать на запросы пользователей по требованию. Однако в некоторых случаях пользователям приходится чаще посылать бенчмаркинг-запросы во время выполнения. В этой ситуации, бенчмаркинг должен быть организован как динамический сервис, который сможет обрабатывать запросы пользователей в режиме реального времени [57].

Не смотря на очевидные преимущества облачных сервисов, у них есть недостатки. Основными недостатками облачных технологий являются отсутствие технологий контроля целостности, достоверности, исправления ошибок и большая избыточность хранимой и обрабатываемой информации. Для повышения надежности хранимых и обрабатываемых данных облачными провайдерами производится репликация [56, 99] которая приводит к большой избыточности данных, что приводит к дополнительной финансовой нагрузке на пользователей.

Репликация данных в системе «Google File System» (GFS), производится по умолчанию на три реплики, рисунок 1.4. Каждой реплики присваивается приоритет, сервер с исходными данными называется «Мастер» сервер. После выхода из строя реплики удаляются сборщиком мусора. После уменьшения количества доступных реплик ниже установленного клиентом порога, производится репликация данных до указанного количества. Диски являются относительно дешевыми и выполнение репликации проще, чем выполнение и обслуживание более сложных RAID подходов [89], GFS в настоящее время использует только репликацию для обеспечения избыточности и поэтому использует больше памяти, чем сырой XFS или Swift [56].

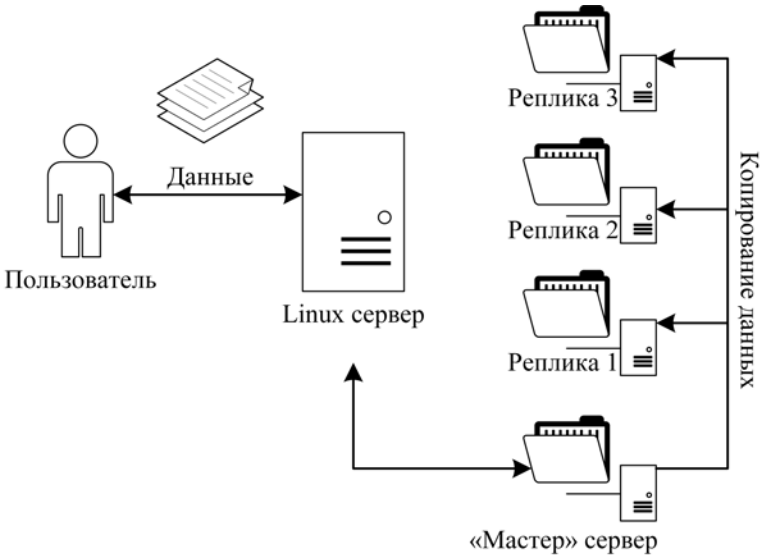

Рисунок 1.4 — Схема репликации данных «Google File System»

Некоторые распределенные файловые системы, такие как Frangipani, XFS, Minnesota's, GFS [98] и GPFS [95] удаляют централизованный сервер и основной упор делают на алгоритмы распределения данных (СРД) для согласованности и управления данными [38]. В системе GFS выбрали централизованный подход, для упрощения конструкции и повышения надежности и гибкость. В частности, централизованный мастер позволяет гораздо легче реализовать сложные политики размещения реплик и произведения репликации, так как мастер уже имеет большую часть необходимой информации и контролирует, как она меняется. В GFS отказоустойчивость обеспечивается путем поддержания мастера состояния малого и полностью реплицированного на других машинах. Масштабируемость и высокая доступность (для чтения) в настоящее время предоставляются механизмом теневого мастера. Мастер обновления состояния сделан стойким путем добавления журнала событий записи. В работе [98] была поставлена проблема GFS с точки зрения предоставления совокупной производительности с большим числом клиентов. Кроме того, GFS предполагает большое количество ненадежных компонентов [56, 70].

Помимо предоставления услуг хранения и обработки данных, облачные решения могут применяться для повышения производительности предприятий различного масштаба. Они обеспечивают совместную работу с документами (Google Docs), управление взаимодействием с клиентами (Salesforce.com), создание отчетов о расходах (Concur), обработку платежных ведомостей (ADP), услуги электронной почты, а также совместное использование, резервное копирование и архивирование файлов. Некоторые поставщики облачных решений располагают средой и средствами для предоставления платформы как услуги (PaaS), на основе которых могут создаваться новые службы типа SaaS.

Облачные ресурсы хранения, предоставляемые как услуга, могут предназначаться для совместной работы с файлами и документами, обмена аудио- и видеозаписями, публикации фотографий, резервного копирования, архивации и восстановления данных, обеспечения непрерывности бизнеса и восстановления после сбоев. Другие варианты облачных систем хранения данных включают базы данных, анализ больших данных (Hadoop и услуги на основе MapReduce), облачные накопители и другие службы на основе облачных систем хранения. Сюда же относятся продукты и решения, используемые для развертывания общедоступных, частных и гибридных облаков.

Продукты и решения являются основными элементами, используемыми в физических системах хранения данных для создания облачных хранилищ. Частные облака и общедоступные службы создаются на основе многоуровневых систем хранения данных, для которых используются как твердотельные накопители, так и жесткие диски. Как и в традиционных корпоративных средах хранения, для облачных услуг и решений зачастую создается несколько уровней с разными технологиями хранения для разных задач и требований к качеству обслуживания. Например, используя твердотельные накопители для улучшения консолидации операций ввода-вывода (с поддержкой журналов и индексов баз данных, метаданных для быстрого поиска и других транзакционных данных), можно достичь большего, затратив меньше усилий и меньше денежных средств, без необходимости расширять занимаемую площадь.

Комбинация твердотельных накопителей и жестких дисков обеспечивает хороший баланс производительности и емкости для разработки разных по стоимости вариантов обслуживания и соблюдения необходимых требований к его уровню.

Разные уровни хранения данных обеспечиваются сверхпроизводительными твердотельными накопителями и производительными жесткими дисками средней и большой емкости. Функции управления хранением включают защиту данных (обеспечение высокой доступности, резервное копирование и средства восстановления после аварий), а также сокращение занимаемых площадей с оптимизацией использования пространства (сжатие, дедупликация и динамическое распределение), что позволяет хранить больше информации в течение более длительного времени при меньших затратах.

К программным средствам, также играющим важную роль в создании служб и решений, относятся интерфейсы API, программное обеспечение (ПО) промежуточного уровня, базы данных, приложения, гипервизоры для создания виртуальных машин и инфраструктур виртуальных ПК, а также облачные программные средства формирования гипертекста (например, OpenStack) и связанные инструменты управления.

Во всех трех случаях ресурсы хранения интегрируются в системе хранения данных, программно-аппаратные комплексы хранения данных и сервера вычислений. Общедоступные облака и службы бывают платными и бесплатными и предоставляют различные наборы функций. В качестве примеров можно

привести Amazon Web Services, Google Docs и ПО для резервного копирования Seagate EVault.

Альтернативным подходом к созданию надежной системы хранения является использование кодов исправления ошибок на основе избыточной СОК, EC-кодов [50] и RC-кодов [79].

Схемы распределения данных, предложенные в работах [19,82], обеспечивают доступность и распределение данных. избыточная СОК имеет аналогичные свойства для схемы распределения данных Mignotte [82], ее арифметические свойства позволяют производить контроль результата обработки данных. Избыточная СОК представляет исходные числа как остатки по отношению к набору модулей. Таким образом, число разбивается на меньшие числа, которые являются независимыми.

Пусть $p_1, p_2, \ldots, p_n$ попарно взаимно простые числа, используемые в качестве набора модулей избыточной СОК, и $n = k + r$. Тогда диапазон избыточной СОК будет равняться $P = \prod_{i=1}^{n} p_i$. Данные представляют собой целое число X, где $X \in [0,\, P-1)$. X определен в избыточной СОК как $X \to (x_1, x_2, \ldots, x_n)$, где $x_i = |X|_{p_i}$ представляет собой остаток от деления X на p_i.

Параметры $(k,\, r)$ избыточной СОК, могут выбираться различными, в зависимости от необходимости получения определенных характеристик. Используя данные из любых k остатков от n мы можем восстановить данные. Согласно свойству избыточной СОК, если количество контрольных модулей равно r, система может обнаружить r и исправить $r-1$ ошибки. Для локализации и коррекции ошибок используются методы проекций, при которых количество вычисленных проекций растет экспоненциально в зависимости от значения r.

В работе [34] авторы предложили использовать избыточный СОК для надежных и масштабируемых систем облачного хранения. Операции могут выполняться параллельно, что упрощает и ускоряет вычисления. Избыточность системы позволяет построить систему с множественным обнаружением ошибок и их коррекцией.

Поскольку представление чисел в избыточной СОК можно рассматривать как схему разделения данных, мы можем ее использовать для надежного хранения данных большой размерности. В случае рассмотрения системы для одного ПК, достаточно обнаруживать и исправлять одну ошибку, большинство систем справляется с данной задачей. Однако, когда мы рассматриваем данные большой разрядности, необходимо иметь эффективные алгоритмы для обнаружения и ис-

правления нескольких ошибок. Схема избыточной СОК для хранения данных обеспечивает надежное и масштабируемое хранение. Она имеет свойства кодов коррекции ошибок и возможнось распределенного хранения данных большой разрядности.

Для создания надежной, отказоустойчивой и безопасной модели хранения данных в распределенной облачной структуре нами будут использоваться избыточная СОК и коды исправления ошибок, хранилище данных будет обладать свойствами: надежности, отказоустойчивости, целостности, распределения данных, контролем данных и исправлением ошибок.

1.2 Анализ методов хранения и обработки больших данных в облаках

Большими данными называются методы обработки данных основанные на нетрадиционных технологиях из-за их размера, таких как сбор, хранение, поиск, распространение, анализ и визуализация больших объемов данных. Стандартные базы данных и инструменты уже не справляются с растущим потоком данных: базы данных уже не в состоянии обрабатывать существующие объемы, ETL-процессы слишком медленны и имеют трудности с разнообразием форматов данных, поэтому традиционные BI-системы слишком медленные и не могут эффективно обрабатывать большие массы неструктурированных данных.

Обработка больших объемов данных зачастую становится самой трудной и самой проблемной областью в создании крупных агрегационных сервисов. Это привело к созданию довольно эффективных способов решения проблемы.

Подробнее остановимся на двух наиболее распространенных решениях: модель распределенных вычислений MapReduce и Percona server – сборкой MySQL изначально предназначенной и оптимизированной для работы с большими данными.

MapReduce – модель распределённых вычислений, представленная компанией Google, используется компанией в компьютерных кластерах для параллельных вычислений над очень большими, даже несколько петабайт, наборами данных.

MapReduce – это фреймворк для работы с набором распределенных задач, использующий большое количество компьютеров (называемых «нодами»), образующих структуру кластера [39].

Работа фреймворка состоит из двух шагов:

1. Map-шаг. Происходит предварительная обработка входных данных. Компьютер - главный узел (master node) получает входные данные. Происходит разделение исходного набора данных на части и передача другим компьютерам - рабочим узлам (worker node) для предварительной обработки. Название шаг берет от одноименной функции высшего порядка.

2. Reduce-шаг. Шаг заключается в свертке обработанных данных. От рабочих узлов в главный узел поступают отчеты и на их основе формируется результат – решение изначальной задачи.

Преимущество MapReduce заключается в возможности распределенно производить операции предварительной обработки и свертки. Операции предварительной обработки могут производиться параллельно так, как работают независимо друг от друга. Однако, процесс может быть менее эффективным по сравнению с более последовательными алгоритмами, так как целью алгоритма MapReduce является применение его к большим объемам данных, которые могут обрабатываться большим количеством серверов. Тем не менее, MapReduce может быть использован для сортировки огромного количества данных, и требует лишь несколько часов даже на объемы порядка петебайта данных. Также параллелизм предоставляет возможности восстановления после частичных сбоев: если возникает сбой в рабочем узле, то его работа может быть передана другому рабочему узлу. Таким образом, хотя фактически семантика отличается от прототипа, в основе фреймворка лежат функции map и reduce, широко применяемые в функциональном программировании.

Percona server – это сборка MySQL. Этой сборке по умолчанию включен XtraDB storage engine, который отличается от MySQL+InnoDB plugin. Ключевыми показателями является лучшая производительность/масштабируемость, особенно на современных многоядерных серверах. Хранилище XtraDB основано на InnoDB-plugin и полностью совместимо с ним, однако, отличается более высокой производительностью, благодаря интеграции патчей от компаний Google и Percona, а именно.

1. Улучшена работа с памятью. 2. Изменена подсистема ввода/вывода. 3. Расширены возможности по масштабированию для больших проектов. 4. Систе-

ма организации блокировок адаптирована для работы на системах с большим числом CPU. 5. Реализованы дополнительные возможности для накопления и анализа статистик. 6. Добавлен ряд новых возможностей: поддержка нескольких потоков чтения и записи, управление пропускной способностью, упреждающая выборка данных (read-ahead), адаптивная установка контрольных точек (adaptive checkpointing).

В области интеграции данных основной проблемой является скорость и управляемость структурированными данными. Для хранения и последующей обработки больших данных доступны специальные файловые системы, такие как HDFS от Hadoop, но также и так называемые базы данных NoSQL. Важно, чтобы эти методы согласовывались с классическими аналитическими базами данных, которые продолжают применяться. Только таким образом можно поддерживать согласованность данных, а типичные реляционные операции могут выполняться без проблем.

Быстрая обработка больших данных фокусируется на подходе MapReduce, разработанном Google. За этим стоит следующий алгоритм: задача разбивается на наименьшие возможные части, затем распределяется для параллельной обработки на столько же компьютеров, а затем снова объединяется. Таким образом, возможна высокая параллельная обработка данных полиструктуры. Еще один инструмент, который позволяет обрабатывать большие данные за считанные секунды – это вычисления в оперативной памяти, такие как SAP HANA, предлагаемые SAP. Здесь память компьютера используется как хранилище данных. В отличие от данных, хранящихся на жестком диске, это позволяет значительно повысить скорость доступа к данным. Существуют также решения, которые полагаются на аналитические базы данных. Это в основном базы данных, ориентированные на столбцы, которые ломаются с общей концепцией классических баз данных, ориентированных на строки. Они отфильтровывают ненужные области и, таким образом, обеспечивают гибкость и, прежде всего, быстрый доступ.

Интернет вещей (Internet of Things, IoT) предоставляет технологии, которые обеспечивают механизмы подключения к устройствам IoT, мобильным и облачным приложениям и генерируют данные, которые будут храниться, анализироваться и действовать на основе этого. При разработке распределенных больших систем хранения данных необходимо учитывать, что данные многочисленны, не могут быть классифицированы в регулярные реляционные базы данных и должны быть быстро захвачены и обработаны.

Традиционные системы хранят данные в структурированных системах управления реляционными базами данных, файловых системах и с использованием репликации. Интенсивные и обширные исследования изучают различные аспекты хранения облачных данных. Тем не менее, смягчение рисков целостности, доступности и надежности, не было должным образом рассмотрено в научной литературе.

Производительность, отказоустойчивость, надежность и масштабируемость важные факторы при обработке больших данных. Инфраструктура хранения должна обеспечивать надежное пространство для хранения с мощным интерфейсом доступа для запроса и анализа. Распределенное хранилище может быть основано на нескольких облаках. Обычно данные делятся на несколько частей, которые должны храниться на разных облаках для обеспечения доступности в случае отказа. Однако отказы распределенного хранилища могут вызывать несогласованность между разными копиями одних и тех же данных [56]. Можно использовать большие базы данных. В этом случае для высокой производительности обработка и анализ данных должны выполняться параллельно.

1.3 Методы повышения надежности и отказоустойчивости хранения и обработки больших данных

Система облако, представляет собой программно-аппаратный комплекс, предназначенный для хранения, организации доступа, управления и восстановления данных. Соответственно, потенциальные угрозы потери, искажения или недоступности информации могут иметь физическую или программную причину.

Обычно в процессе эксплуатации облачной системы предусматривается возможность возникновения аварийных ситуаций, которые должны учитываться на уровне технологий облачной системы, и эти решения должны обеспечивать системе требуемый уровень безопасности и надежности. Рассмотрим самые распространенные ситуации.

Вышел из строя один или несколько жестких дисков в хранилище данных – полная или частичная потеря данных. Необходимо иметь возможность произвести восстановление данных, хранившихся на этом жестком диске.

Вышла из строя материнская плата на сервере – сервер, в котором она установлена, станет недоступным, и все недавние изменения или результаты текущей работы сервера будут утрачены. Необходимо иметь возможность обслужить поступающую нагрузку с заданным качеством, которая изначально предназначалась для данного сервера. При этом нужно восстановить данные, измененные в результате завершенных на данном сервере транзакций.

Обрыв связи между сервером и хранилищем – потеря всех текущих изменений, обрабатываемых на данном сервере.

Вирус на сервере – могут возникнуть сбои доступа к серверу, что приведет к потере недавних изменений и остановит работу. Под угрозой также находятся целостность данных в хранилище, архивные логи, потеря которых исключит возможность восстановления утраченной части данных. Поскольку поведение вирусов непредсказуемо, последствия тоже могут иметь случайный характер.

Инструменты по обеспечению отказоустойчивости облачной системы и защите данных от утери должны обладать следующими свойствами [104]:

- надежность (reliability) – вся информация, необходимая для восстановления данных, должна храниться в надежном месте на надежных носителях и быть зарезервирована;
- гибкость (flexibility) – резервирование должно быть произведено так, чтобы при необходимости можно было восстановить все данные в целом и конкретные файлы данных;
- управляемость (manageability) – резервные файлы должны быть легко и удобно управляемы для того, чтобы восстановление можно было произвести в кратчайшие сроки;
- готовность (availability) – работа по резервированию ни при каких условиях не должна мешать работе. Кроме того, работа по восстановлению должна выполняться незаметно для пользователя.

Для обеспечения отказоустойчивости используют следующие нейтральные подходы.

Холодное (offline) резервирование. Наиболее простой способ резервирования данных: всю поступающую нагрузку обслуживает основной экземпляр, с которого с определенной периодичностью снимаются точки восстановления путем полного копирования данных на носитель.

При возникновении сбоя последняя (или более поздняя) точка восстановления загружается на сервер данных (это может быть основная или резервная копия данных), и на нее переключается вся поступающая нагрузка.

В данном случае возможны три схемы:

- холодное резервирование производится на отдельный носитель этого же сервера;
- холодное резервирование производится на носитель отдельного сервера, предназначенного для резервирования информации компании по сети (скорее всего, в качестве сервера хранения и обработки данных он не может быть использован из-за ограниченных ресурсов);
- холодное резервирование производится на отдельный сервер, который выделен в качестве запасного/резервного сервера.

Снятие точек восстановления при холодном резервировании можно осуществлять лишь периодически, поскольку при этом подразумевается остановка основного сервера хранения. Периодическое снятие точек восстановления выполняется средствами базы данных или внешними утилитами, копирующими файлы данных на уровне операционной системы. В случае с внешними утилитами временно блокируется доступ к базам данных и происходит копирование управляющих файлов (control file), имеющихся логов (log files) и архивных логов (archive log files) на носитель информации (жесткий диск). Таким образом, по завершении описанной операции получается точная копия имеющихся данных на момент остановки работы с ними.

Неудобство применения данного подхода заключается в том, что частота копирования данных ограничена и, как правило, совпадает с часами наименьшей нагрузки на сервер облака. Очевидно, что работы по резервированию приходится производить либо в ночное время, либо в выходные дни, что, в свою очередь, позволяет производить откаты только к этим моментам времени с возможной потерей информации за весь последний день или неделю. Холодное резервирование находит свое применение с данными небольшой емкости. Процесс восстановления данных занимает длительное время и является обратным копированием имеющейся информации с резервного носителя на основной. А если учесть, что продолжительность процесса восстановления – как работоспособности, так и работы в нормальном режиме с использованием основных ресурсов – жестко нормируется с заказчиком, то могут возникнуть ситуации, когда такой подход неприемлем.

Непрерывное снятие точек восстановления. Подход непрерывного снятия точек восстановления заключается в том, чтобы отслеживать изменения, происходящие с данными, в режиме реального времени и сохранять историю изменений на выделенном накопителе. Предполагается, что таким образом любые данные можно восстановить на любую, заданную администратором дату. Однако такой подход порождает немало проблем, которые приходится решать производителям. К примеру, при частом обновлении данных объем знаний об изменениях в данных может превышать объем исходных данных в несколько раз. Положительным моментом является то, что при сбое теряются только данные незавершенных транзакций, существовавших в момент сбоя. В качестве компромисса может использоваться вариант с периодическим снятием образов данных по расписанию и сохранение непрерывной истории изменений за какой-либо ограниченный промежуток времени.

Горячее (online) резервирование. Это целый класс способов проведения резервирования данных, отличительной чертой которых является то, что для выполнения резервирования нет необходимости останавливать работающую часть данных. В нормальном режиме все запросы обслуживаются основной частью, а изменения в ней синхронизируются с резервной. Резервная база чаще всего недоступна для работы пользователей или же доступна только для чтения [18].

При возникновении сбоя в основной части данных вся нагрузка переключается на резервный экземпляр. Можно устанавливать любое значение периода снятия точек восстановления, однако, поскольку снятие точки восстановления требует копирования лога изменений, архивации, передачи его в резервную базу и прочих действий, производительность основного сервера во время этого заметно падает. Следовательно, необходимо подбирать период снятия точек так, чтобы достигнуть оптимума между падением производительности и уменьшением потери данных при сбое. Часто период снятия лога регулируется через размер лог файла, например, при достижении им 100 Мб происходит переключение на следующий лог, а завершенный отправляется в резервную базу. Физическое разнесение экземпляров копий данных дает возможность избежать аппаратного сбоя и повышает устойчивость системы к авариям. Такой подход позволяет существенно повысить надежность хранения данных при возникновении сбоев в одном из экземпляров данных. В простейшем способе организации данного подхода требуется 100%-ная избыточность хранения данных, а значит, существенно увеличивается стоимость данного решения.

В современных системах нередко используется подход с распределением нагрузки: создаются несколько экземпляров данных, каждая из которых обрабатывает часть запросов. При выходе одного экземпляра из строя нагрузка распределяется между остальными. Распределение нагрузки позволяет снизить избыточность хранения. Одним из вариантов распределения нагрузки является архитектура с полной репликацией, типичным примером является архитектура «Google File System» приведенная на рисунке 1.4.

На практике используется способ, для обеспечения надежности данных суть которого заключается в хранении на нескольких серверах системы полные копии хранимой информации. Этот метод гарантирует надежность, так как информация может быть восстановлена, даже если один из серверов работоспособен. Однако данный метод является затратным, так как z реплики приводят к экспансии z времени. Один из способов уменьшить скорость расширения является использование кодов стирания для кодирования сообщений. Сообщение кодируется как кодовое слово, которое представляет собой вектор символов, и каждый сервер хранит символ кодового слова. Отказ сервера хранения моделируется как ошибка с удалением хранящихся символов кодового слова. Случайные линейные коды поддерживают распределенные кодировки, то есть, каждый символ кодового слова вычисляется независимо. Чтобы сохранить сообщение размеров в k блоков, каждый сервер хранения линейно сочетает в себе блоки со случайно выбранными коэффициентами и хранит символ кодового слова и коэффициентов.

Для того, чтобы получить сообщение для хранимых символов кодового слова и коэффициентов, пользователь запрашивает серверы хранения k и решает линейные системы.

Для повышения надежности при хранении и обработки информации в облачной среде применяют подход, основанный на СРД. СРД позволяют разделять данные между несколькими серверами (участниками), участвующими в обработке или хранении информации. Любое множество серверов при хранении информации, содержит достаточный объем информации необходимый для ее восстановления.

Одним из путей по увеличению быстродействия, надежности и отказоустойчивости вычислительных средств обусловило создание вычислительных систем основанных на модулярной арифметике, то есть коды, в которых числа представляются в СОК.

Основная идея теории представлении чисел в СОК, состоит в замене операций над целыми большими числами, операциями над остатками от деления этих чисел на заранее выбранные взаимно простые числа – модули $p_1, p_2, \ldots, p_n$. Чаще всего, числа $p_1, p_2, \ldots, p_n$ выбираются из множества простых чисел. Преимуществом модулярного представления числа, заключается в том, что операции сложения, вычитания и умножения выполняются очень просто и параллельно.Стремление к уменьшению величины оснований привело к построению многоступенчатой системы остаточных классов.

Пусть главная система оснований задана модулями $p_1, p_2, \ldots, p_n$, и эта система позволяет производить операции в диапазоне $P = p_1 \cdot p_2 \cdot \ldots \cdot p_n$. Максимальное число, которое может быть получено в этой системе при умножении отдельных цифр, будет равняться $(p_n - 1)^2$.

Будем теперь представлять все цифры главной системы в новой системе с основаниями $\pi_1, \pi_2, \ldots, \pi_n$ таким, что

$$\pi = \pi_1 \cdot \pi_2 \cdot \ldots \cdot \geqslant (p_n - 1)^2 .$$

В этой системе максимальным числом, которое может быть получено при умножении ее цифр, будет число $(\pi_r - 1)^2$. Такой процесс перехода к меньшим основаниям заметно сокращает время выполнения арифметической операции.

СОК позволяет производить обнаружение, локализацию и исправление ошибок. Это позволяет эффективно применять ее для борьбы с ошибками, возникающими в процессе передачи данных и при ее обработке в цифровых информационных системах. Однако большинство алгоритмов имеют выраженный немодульный характер и, следовательно, высокую вычислительную сложность. Для повышения эффективности коррекционных алгоритмов нужно стремиться максимально исключить наиболее часто используемую и сложную операцию перевода числа из СОК в позиционную систему счисления (ПСС), заменяя ее, например, на операцию расширения диапазона представления чисел в СОК.

Рассмотрим более подробно метод проекций для исправления ошибок. Пусть задана упорядоченная система взаимно простых оснований $p_1, p_2, \ldots, p_k, p_{k+1}, \ldots, p_n$, в которой k первых оснований являются информационными, а основания $p_{k+1}, p_{k+2}, \ldots, p_n$ контрольными. Информационные основания образуют рабочий диапазон представления СОК: $P_k = p_1 \cdot p_2 \cdot \ldots \cdot p_k$. Для обеспечения возможности локализации и исправления ошибок в коде СОК на выбор рабочих и контрольных оснований накладывается ряд ограничений, опи-

санных в [14]. Пусть в данной системе представлено число $A = (\alpha_1, \alpha_2, \ldots, \alpha_n)$. Требуется определить, ошибочно оно или нет, и по возможности исправить ошибку.

Обнаружить ошибку можно, сравнив число A со значением P_k. Если выполняется неравенство $A < P_k$, то ошибки нет, иначе – в одном или нескольких каналах произошло искажение данных [7, 14]. Сравнение в СОК является немодульной операцией. Один из самых простых путей решения проблемы сравнения – это преобразовать код СОК в позиционный и сравнить его с диапазоном в традиционной системе счисления. Переход из СОК в ПСС можно осуществить, например, с помощью ортогональных базисов $B_i = \frac{m_i P_n}{p_i}$, где $\frac{m_i P_n}{p_i} \equiv 1 \bmod p_i$, $P_n = p_1 \cdot p_2 \cdot \ldots \cdot p_n$, по формуле $A = \left| \sum_{i=1}^{n} \alpha_i B_i \right|_{P_n}$. Такой подход влечет большие вычислительные затраты. Один из путей выхода из сложившегося положения предложен в [14], где для обнаружения ошибки применяется приближенный метод.

Метод проекций относится к методам определения местоположения ошибки. Под i-ой проекцией числа $A = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ понимается представление числа A в сокращенной системе остаточных классов, представляющей собой исходную систему, из которой исключено i-е основание. Представление получается простым вычеркиванием i-ого остатка в исходной записи числа. Следовательно, число разрядности n имеет n проекций.

Непосредственно метод локализации состоит в следующем. Находятся все проекции числа $A = (\alpha_1, \alpha_2, \ldots, \alpha_n)$: $A_1 = (\alpha_2, \alpha_3, \ldots, \alpha_n)$, $A_2 = (\alpha_1, \alpha_3, \ldots, \alpha_n)$, …, $A_n = (\alpha_1, \alpha_2, \ldots, \alpha_{n-1})$. Далее с помощью рассмотренных ранее методов сравнения проверяется условие для каждой полученной проекции: $A_i < P_k$, $i = \overline{1, n}$. Если условие не выполняется, то остаток по данному модулю считается правильным, в случае выполнения условия считается, что ошибка произошла по данному основанию [14].

Пример. Пусть в системе $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, $p_5 = 11$ с контрольными основаниями p_4 и p_5 передавалось правильное число $A = 19 = (1, 1, 4, 5, 8)$, а на выход пришло искаженное число $\tilde{A} = (1, 0, 4, 5, 8)$. Требуется определить наличие ошибки, локализовать ее и исправить.

Система оснований $(2, 5, 7, 11)$ имеет ортогональные базисы $B_1 = 1155$, $B_2 = 1540$, $B_3 = 1386$, $B_4 = 330$, $B_5 = 210$. Рабочий диапазон для выбранной системы $P_3 = 30$. Найдем позиционное представление искаженного числа:

$\tilde{A} = |1155 \cdot 1 + 1386 \cdot 4 + 330 \cdot 5 + 210 \cdot 8|_2 310 = 789 > 30$, следовательно данное число содержит ошибку. Для ее локализации найдем все проекции числа $\tilde{A}$, переведем их в ПСС и сравним со значением рабочего диапазона.

По модулю $p_1 = 2$: $\tilde{A}_1 = (0, 4, 5, 8)$. Для системы $(3, 5, 7, 11)$, $B_1 = 385$, $B_2 = 231$, $B_3 = 330$, $B_4 = 210$, $\tilde{A}_1 = |231 \cdot 4 + 330 \cdot 5 + 210 \cdot 8|_1 155 = 789 > 30$.

По модулю $p_2 = 3$: $\tilde{A}_2 = (1, 4, 5, 8)$. Для системы $(2, 5, 7, 11)$, $B_1 = 385$, $B_2 = 616$, $B_3 = 330$, $B_4 = 210$, $\tilde{A}_2 = |385 \cdot 1 + 616 \cdot 4 + 330 \cdot 5 + 210 \cdot 8|_7 70 = = 19 < 30$.

По модулю $p_3 = 5$: $\tilde{A}_3 = (1, 0, 5, 8)$. Для системы $(2, 3, 7, 11)$, $B_1 = 231$, $B_2 = 154$, $B_3 = 330$, $B_4 = 210$, $\tilde{A}_3 = |231 \cdot 1 + 330 \cdot 5 + 210 \cdot 8|_4 62 = 327 > 30$.

По модулю $p_4 = 7$: $\tilde{A}_4 = (1, 0, 4, 8)$. Для системы $(2, 3, 5, 11)$, $B_1 = 165$, $B_2 = 220$, $B_3 = 66$, $B_4 = 210$, $\tilde{A}_4 = |165 \cdot 1 + 66 \cdot 4 + 210 \cdot 8|_3 30 = 129 > 30$.

По модулю $p_5 = 1$: $\tilde{A}_5 = (1, 0, 4, 5)$. Для системы $(2, 3, 5, 7)$, $B_1 = 105$, $B_2 = 70$, $B_3 = 126$, $B_4 = 120$, $\tilde{A}_5 = |105 \cdot 1 + 126 \cdot 4 + 120 \cdot 5|_2 10 = 159 > 30$.

Таким образом установлено, что ошибка произошла по модулю p_2. Вычислим верную цифру по этому основанию, применяя метод расширения системы оснований. Вычислим необходимые для метода расширения константы относительно системы $(2, 5, 7, 11)$ с новым модулем $p_{m+1} = 3$: $P_m = 770$, $R' = \frac{770}{11} = 70$, $\xi = |70|_3 = 1$, $\pi = |770|_3 = 2$, $\theta = 2$, $S_1 = 35$, $S_2 = 56$, $S_3 = 30$, $S_4 = 19$. Далее, $R_{\tilde{A}} = |35 \cdot 1 + 56 \cdot 4 + 30 \cdot 5 + 19 \cdot 8|_7 0 = 1$. Тогда, согласно формуле (3.23): $\alpha_2 = |8 + 2 \cdot 1|_3 = 1$. Значит, исправленное число для $\tilde{A} = (1, 0, 4, 5, 8)$ будет иметь вид $A = (1, 1, 4, 5, 8)$.

Рассмотрим более подробно метод синдрома ошибок. Рассмотрим систему информационных оснований $p_1, p_2, \ldots, p_k$. Число A в ней можно представить остатками $A = (\alpha_1, \alpha_2, \ldots, \alpha_k)$. Расширяя данную систему на модули $p_{k+1}, p_{k+2}, \ldots, p_n$, получим числа $\alpha'_{k+1}, \alpha'_{k+2}, \ldots, \alpha'_n$. Если данные числа совпадают с остатками числа A по контрольным основаниям в исходном представлении, то ошибки в нем нет. Вычислим синдромы ошибок:

$$\varphi_i = \alpha_{k+i} - \alpha \bmod p_{k+i}, \tag{1.1}$$

где, $i = \overline{1, n-k}$. На основании полученных синдромов из специальной корректционной таблицы выбирается константа ошибки. Эта константа показывает, по какому основанию произошла ошибка и вычисляется таким образом, что бы при ее сложении с контролируемым числом, имевшая место ошибка устранялась.

Таблица 1 — Коррекционная таблица для системы оснований $(2, 5, 23)$

φ_1	Ошибка	φ_1	Ошибка	φ_1	Ошибка
0	$(0,0,0)$	$3,16$	$(1,3,0)$	$6,19$	$(0,1,0)$
$1,14$	$(1,1,0)$	$4,17$	$(0,4,0)$	$7,20$	$(1,2,0)$
$2,15$	$(0,2,0)$	$5,18$	$(1,0,0)$	$8,21$	$(0,3,0)$
				$9,22$	$(1,4,0)$

При применении описанного табличного метода коррекции ошибки на контрольные основания системы накладывается ряд ограничений: предполагается, что контрольные остатки абсолютно надежны, то есть числа $\alpha_{k+1}, \alpha_{k+2}, \ldots, \alpha_n$ не могут быть ошибочными; величина рабочего диапазона должна быть более чем вдвое меньше, чем произведение контрольных модулей: $2 \cdot P_k < p_{k+1} \cdot p_{k+2} \cdot \ldots \cdot p_n$.

Пример. Пусть дана система оснований: $p_1 = 2$, $p_3 = 5$, $p_3 = 23$, в которой основание p_3 примем как контрольное. Для данной системы оснований коррекционная таблица представлена в таблице 1. Предположим, что задано ошибочное число $A = (0, 1, 3)$. Произведем расширение системы информационных оснований. Информационная часть числа равна $(0, 1)$, что соответствует позиционному числу 6.

Остаток от деления 6 на 23 равен 6, то есть $\alpha'_3 = 6$. Вычислим синдром по формуле (1.1): $\varphi_1 = 3 - 6 \bmod 23 = 20$. В соответствии с таблицей 1, корректирующей константой для данного числа A будет равна $(1, 2, 0)$. Иными словами, настоящее значение числа A имеет вид:

$$\tilde{A} = (0, 1, 3) + (1, 2, 0) = (1, 3, 3).$$

Рассмотрим способ построения коррекционной таблицы. Пусть задано некоторое правильное число $\tilde{A} = (\beta_1, \beta_2, \ldots, \beta_n)$. Подействуем на него некоторой ошибкой: $B = (\gamma_1, \gamma_2, \ldots, \gamma_k, 0, \ldots, 0)$. Искаженное число будет иметь вид:

$$A = (\alpha_1, \alpha_2, \ldots, \alpha_n) = \tilde{A} - B = (\beta_1 - \gamma_1, \beta_2 - \gamma_2, \ldots, \beta_k - \gamma_k, \beta_{k+1}, \ldots, \beta_n)$$

Каждой ошибке B соответствует 2 синдрома – числа, представленных в СОК с системой модулей, состоящей только из контрольных оснований исходной системы.

Число B в системе информационных оснований $p_1, p_2, \ldots, p_k$ имеет представление $B = (\gamma_1, \gamma_2, \ldots, \gamma_k)$. Первый синдром соответствует числу B_k, представленному в СОК с основаниями $p_{k+1}, p_{k+2}, \ldots, p_n$. Расширим данное число на

каждый из модулей контрольной системы $p_{k+1}, p_{k+2}, \ldots, p_n$. Полученное число $\left(\varphi_1^{(1)}, \varphi_2^{(2)}, \ldots, \varphi_{n-k}^{(n-k)}\right)$ является первым синдромом данной ошибки B.

Второй синдром соответствует числу $\left|-\left|B_k\right|_{P_k}\right|_{P'}$, где $P' = p_{k+1} \cdot p_{k+2} \cdot \ldots \cdot p_n$. Найдем в системе $p_1, p_2, \ldots, p_k$ число $-B_k$:

$$-B_k = \left(\left|-\gamma_1\right|_{p_1}, \left|-\gamma_2\right|_{p_2}, \ldots, \left|-\gamma_k\right|_{p_k}\right) = \left(\gamma_1', \gamma_2', \ldots, \gamma_k'\right).$$

Найдем представление данного числа в системе контрольных оснований, используя операцию расширения системы оснований СОК: $-B_k = \left(\varphi_1', \varphi_2', \ldots, \varphi_{n-k}'\right)$. Теперь вновь вычислим противоположное число для $-B$, действуя в СОК с модулями из системы контрольных оснований.

$$B_k' = \left(\left|-\varphi_1'\right|_{p_{k+1}}, \left|-\varphi_2'\right|_{p_{k+2}}, \ldots, \left|-\varphi_{n-k}'\right|_{p_n}\right) = \left(\varphi_1^{(2)}, \varphi_2^{(2)}, \ldots, \varphi_{n-k}^{(2)}\right).$$

Набор $\left(\varphi_1^{(2)}, \varphi_2^{(2)}, \ldots, \varphi_{n-k}^{(2)}\right)$ есть второй синдром данной ошибки.

Таким образом, для того, что бы составить коррекционную таблицу, необходимо перебрать все возможные ошибки и вычислить для них первый и второй синдромы.

1.4 Методы распределенной обработки данных с регулируемой избыточностью

Построение распределенных систем хранения и обработки можно производить различными способами, некоторые из них основаны на парадигмах облачных и Grid-вычислений [109]. Эти инфраструктуры имеют общие характеристики, но также и принципиальные различия.Использование облаков для хранения данных имеет ряд ограничений связанных с надежностью и масштабируемостью при ограниченной пропускной способности интернет-соединения. Для обеспечения быстрого и легкого доступа к распределенным данным, и обеспечения высокой степени надежности, доступности и масштабируемости Chang и др., в работе [35] предложили систему хранения «Bigtable», которая основывается на репликации данных.

Альтернативным механизмом являются подходы Hadoop и MapReduce, основанные на разделении данных на независимые части, которые обрабатываются

параллельно и уменьшают их размер [45]. Однако, как показано в работе [65], основным недостатком данного подхода является низкая эффективность.

Существуют различные способы которые используются для распределенных данных [108]. Распределенная база данных (БД) хранит данные на разных серверах компьютерной сети и использует логику для организации набора данных [17, 27, 84]. Существует два способа создания распределенной БД. Низходящий подход использует БД и распределяет ее по различным сайтам, в то время как восходящий подход объединяет несколько различных БД с одним интерфейсом. Основной областью применения распределенных БД, является структурированное хранение данных, поэтому оно не применимо к произвольным наборам данных, таким как большие данные. Сеть доставки контента (CDN) представленная в работе [49], состоит из набора серверов, которые кэшируют данные, выполняют клиентские запросы в базе данных и уменьшают нагрузку на серверы происхождения. Основные принципы CDN состоят в следующем:

– балансировка нагрузки;
– сохранение скорости пропускания данных;
– эффективность времени.

Однако CDN не используются широко на практике из-за того, что они не являются гибкими. Основными принципами P2P сетей [83] являются масштабируемость и надежность, достигаемые за счет децентрализованной структуры и избыточности, распределения ресурсов и анонимности. Сети P2P эффективны в обеспечении быстрого доступа к файлам для группы пользователей. Тем не менее, большинство сетей P2P не допускают интегрированных вычислений и служат средой распространения данных.

При хранении данных в облака, следует учитывать надежность, масштабируемость, безопасность, конфиденциальность. Эти характеристики также имеют решающее значение для мобильных устройств, где технические характеристики и потребление энергии ограничены. Черных и др., в работе [105] показали, что распределенное хранение данных в условиях неопределенности в облачных вычислениях может использовать репликацию данных, избыточную СОК, коды стирания (EC) и регенерационные коды (RC).

В таблице 2 представлены основные известные методы организации распределенного хранения данных в облаках, проведено сравнение по следующим свойствам: надежность, масштабируемость, доступность, конфиденциальность, целостность. Наиболее эффективным с точки зрения сложности является метод

представленный в работе [56]. Однако его основным недостатком является то, что данные хранятся в открытом виде с использованием репликации, что приводит к ограниченной применимости.

Альтернативным подходом к созданию надежной системы хранения является использование кодов исправления ошибок на основе избыточной СОК, EC-кодов [50] и RC-кодов [79]. В таблице 2 особое внимание заслуживает схема распределенного хранения данных [59], которое обеспечивает безопасность, целостность, надежность и масштабируемость данных. Авторы предложили два подхода к построению систем, основанных на схемах распределения данных в избыточной СОК. Модули избыточной СОК хранятся у пользователя. Обработка данных приводит к экспоненциальному увеличению нагрузки сети и памяти и делает ее неприменимой на практике.

Избыточная СОК представляет исходные числа как остатки по отношению к набору модулей. Таким образом, число разбивается на меньшие числа, которые являются независимыми.

Пусть $p_1, p_2, \ldots, p_n$ попарно взаимно простые числа, используемые в качестве набора модулей избыточной СОК. Тогда диапазон избыточной СОК будет равняться $P = \prod_{i=1}^{n} p - i$. Данные представляют собой целое число X, где $X \in [0,\, P-1)$. X определен в избыточной СОК как $X \to (x_1, x_2, \ldots, x_n)$, где $x_i = |X|_{p_i}$ представляет собой остаток от деления X на p_i.

В работе [34] авторы предложили использовать избыточную СОК для надежных и масштабируемых систем облачного хранения. Операции могут выполняться параллельно, что упрощает и ускоряет вычисления. Избыточность системы позволяет построить систему с множественным обнаружением ошибок и их коррекцией.

Общей проблемой для большинства систем является обнаружение и исправление одной ошибки. Когда надежность обеспечивается для одного компьютера, достаточно обнаружения и исправления одной ошибки. Однако, когда мы рассматриваем большие данные, необходимо иметь эффективные алгоритмы для обнаружения и исправления нескольких ошибок [80]. Схема избыточной СОК для хранения данных обеспечивает надежное и масштабируемое хранение. Она имеет свойства кодов коррекции ошибок и возможнось распределенного хранения данных.

Таблица 2 — Обзор методов распределенного хранения данных в облаках

Метод	Характеристика						
	Доступность	Конфидециальность	Гомоморфность	Целостность	Приватность	Надежность	Масштабируемость
Abu-Libdeh [15]	+					+	
Adya [16]	+	+		+	+	+	+
Ateniese [20]		+		+	+	+	
Bessani [25]	+	+		+	+	+	
Bowers [26]	+			+	+	+	+
Celesti [34]	+			+	+	+	
Dimakis [50]		+		+	+	+	+
Erkin [52]			+		+		+
Gentry [55]			+		+		
Ghemawat [56]	+	+				+	+
Gomathisankaran [59]		+	+	+	+	+	+
Kong [72]		+		+	+	+	+
Li [76]	+	+		+	+	+	+
Lin [78]	+	+		+	+	+	+
Pang [85]	+	+		+	+	+	+
Parakh 2011 [87]	+	+		+	+	+	+
Parakh 2009 [86]		+		+	+	+	+
Ruj [92]	+	+		+	+		
Samanthula [93]			+		+		
Sathiamoorthy [94]		+				+	+
Shah [97]		+				+	+
Wang [110]	+	+		+	+	+	+
Wylie [113]	+	+		+			+
Yang [114]	+	+		+			+

1.5 Постановка задачи исследования

Существенное увеличение объемов информации является отличительной чертой современного мира. При работе таких установок как Европейский рентгеновский лазер на свободных электронах XFEL (X-ray Free Electron Laser), большой адронный коллайдер (CERN), коллайдер НИКА (Дубна) и других научных систем, уже получены сотни петабайт экспериментальных данных в области физики элементарных частиц, биоинформатики, геофизики и др., причем объемы получаемых данных будут расти и скоро достигнут экзабайтной отметки. Ведущими компаниями рынка информационных технологий обрабатываются огромные массивы данных: Google (поисковый индекс 100 Пбайт), Facebook (180 Пбайт) , YouTube (15 Пбайт в год). Вся полученная в ходе экспериментов информация должна быть доступна всем членам научных коллабораций и коллективов компаний, где участники коллективов почти всегда географически распределены. В этой ситуации задача надежного хранения и управления метаданными становится фундаментальной и отсутствие ее адекватного решения приводит к экономическим и функциональным потерям [53].

Обработка больших данных является важной задачей и стоит на ровне с хранением. Необходимо обрабатывать данные в режиме реального времени. Большинство систем хранения больших данных построено на дисках, а обработка ведется на нескольких компьютерах или серверах, собранных в кластер. Технология in-memory computing позволяет в рамках такого же кластера использовать оперативную память компьютеров. С точки зрения доступа и обработки, оперативная память становится в тысячи, а то и в миллионы раз быстрее, чем диски. Архитектура современных компьютеров не предусматривает других возможностей для этой цели.

У технологии in-memory computing есть свои ограничения. Прежде всего это цена – стоимость такого решения всегда дороже, чем обработка данных на диске. Кроме того, добиться петабайтного объема сложно и не всегда экономично. Поэтому технология используется в основном для операционных, а не исторических данных. Кроме того, стабильность работы решения на in-memory computing во многом зависит от сетевого решения, на котором построен кластер. Также критически важным становится качество разработки.

Распределенная инфраструктура представляет условия, в которых конкуренция за ресурсы между высоко-приоритетными вычислительными задачами анализа данных происходит регулярно. Неизбежно, перегруженность вычислительных ресурсов вызывает ухудшение функционирования обслуживающих сервисов, а порой и длительные перерывы в их работе. По этой и другим причинам в распределенной обработке данных неизбежно происходит непрерывный поток отказов, ошибок и неисправностей. Безвозвратная потеря важных данных может привести к фатальным последствиям. В связи с этим большинство пользователей стремятся снизить риск безвозвратной потери хранимых данных. Встает вопрос о необходимости использования алгоритмов с высокой отказоустойчивостью, обеспечивающих повышенную надежность хранимых данных, способных осуществлять быстрое и стойкое преобразование и не требовать от пользователя какой-либо специальной подготовки.

Анализ проблем организации хранения и обработки данных в области «Индустрии 4.0» показал, что основной задачей, стоящей перед такими системами, является задача обработки, хранения и анализа данных в режиме реального времени. Однако, возникает противоречие между практической потребностью обработки данных большой разрядности в режиме реального времени, ограниченностью аппаратных ресурсов современных систем, стоимостью, надежностью и производительностью. Таким образом, необходимо обеспечить повышение скорости выполнения арифметических вычислений, надежности и доступности за счет разработки новых математических моделей обработки данных в облачной среде, алгоритмы которых используют методы, сокращающие время их работы, а также разработать новые алгоритмы обработки и хранения данных позволяющие уменьшить финансовые издержки.

Пространство состояний динамической системы, адекватно реальной загрузки облачного сервера. Каждый облачный сервер задается характеристикой x_1 – производительность, $\bigcup_{i=1}^{n} x_i = X$. Каждый пользователь задается параметрами y_i: y_1 – стоимость за единицу времени, y_2 – время обнаружения ошибки, y_3 $(P_{\text{П}})$ время исправления ошибки ($P_{\text{П}}$ – вероятность потери данных), $\bigcup_{i=1}^{n} y_i = Y$. Отношение x_i/y_i определяет относительную эффективность работы i-го облачного сервера. Если k_i – эффективность работы системы $k_i > 0$, и α_i – вероятность отказа работы облачной системы $0 < \alpha_i < 1$, то вероятность того, что облачный сервер выполнит задачу будет $(1 - \alpha_i e^{-k_i(x_1/y1)})^{y_i}$. Если $v_i > 0$ – ценность i-ой части данных, то пользователь хочет минимизировать вероятность потери

данных, финансовые издержки, время обработки данных и получить максимальную производительность, надежность и отказоустойчивость. Для решения задачи построения оптимальной модели обработки и хранения данных необходимо провести исследование функции

$$P_{\text{П}} = P_{\text{ПО}} + P_O + P_{DDoS} \to \min$$

где P_{DDoS} – вероятность проведения DDoS атаки, $P_{\text{ПО}}$ – вероятность отказа программного обеспечения, P_O – вероятность отказа облачной системы.

Решение поставленной задачи позволит получить преимущества связанные с повышением производительности, надежности, отказоустойчивости, производительности при минимальных затратах на аренду вычислительных мощностей.

В соответствии с темой и целью диссертационной работы, а также основываясь на результатах исследований, проведенных в первой главе, сформулируем научную задачу проведения исследований.

Цель диссертационного исследования – повышение отказоустойчивости и надежности схем обработки и хранения данных в облачных сервисах с регулируемой избыточностью.

Объект исследования – облачные инфраструктуры хранения данных.

Предмет исследования – математические модели, методы и алгоритмы хранения и обработки данных в облачных хранилищах.

Научная задача – разработка новых математических методов и моделей хранения и обработки данных большой разрядности с использованием модулярной арифметики в облачных хранилищах с регулируемой избыточностью.

Для решения поставленной общей научной задачи была произведена ее декомпозиция на ряд частных задач:

1. Аналитический обзор современных облачных хранилищ данных большой разрядности.
2. Разработка математической модели, методов и алгоритмов системы надежного, длительного хранения данных большой разрядности в мультиоблачной среде на базе системы остаточных классов с регулируемой избыточностью.
3. Разработка математической модели синтеза и анализа многоуровневых облачных систем хранения данных следующего поколения.

4. Модификация численного метода Червякова для перевода чисел из системы остаточных классов в позиционную систему счисления за счет использования ранга числа Акушского.
5. Разработка среды моделирования распределенного, длительного хранения данных большой разрядности в облачной среде.

1.6 Выводы по первой главе

1. В ходе анализа математических методов хранения и обработки больших данных, применяемых в облачных системах, установлена возможность реализации данных операций в модулярном базисе, что обеспечивает повышенную производительность и отказоустойчивость разрабатываемых на их основе алгоритмов.
2. Особенность СОК заключается в реализации высокопроизводительных алгоритмов в сочетании с возможностью контроля и исправления ошибок. Многие ученые отмечают высокую эффективность СОК при реализации процедур, применяемых в облачных системах хранения и обработки данных. В первую очередь это связано с представлением данных в непозиционной форме, что позволяет обрабатывать части параллельно и независимо. Такое представление дает преимущество в отказоустойчивости и при правильной реализации ведет к уменьшению нагрузки на серверы.
3. Показано, что примение модулярной арифметики, широко развиваемой и применяемой в настоящее время, позволяет производить построение надежных и отказоустойчивых структур хранения и обработки больших данных. Рассмотрены основные методы распределенного хранения и обработки данных и показаны преимущества совместного использования СОК и СРД для повышения отказоустойчивости и надежности. Разработка методов надежного и отказоустойчивого хранения больших данных в облачной среде на основе СОК и СРД является основной задачей исследования.

Глава 2. РАЗРАБОТКА МЕТОДОВ НАДЕЖНОГО И ДЛИТЕЛЬНОГО ХРАНЕНИЯ ДАННЫХ В ОБЛАКАХ С ИСПОЛЬЗОВАНИЕМ СИСТЕМЫ ОСТАТОЧНЫХ КЛАССОВ

2.1 Выбор критериев надежности

Надежностью будем считать способность объекта выполнять требуемую работу в указанных условиях в течении установленного времени. Основными критериями надежности являются: работоспособность, безотказность, долговечность, ремонтопригодность, сохраняемость. [60].

Работоспособность – состояние устройства, при котором оно способно выполнять заданные функции, сохраняя заданные значения параметров в пределах установленных научно-технической документации.

Безотказность – свойство изделия непрерывно сохранять работоспособность в течение заданного времени. Безотказность характеризуется – вероятностью безотказной работы $P(t)$ и интенсивностью отказов $\lambda(t)$. Под вероятностью безотказной работы $P(t)$ понимают вероятность того, что в заданном интервале времени не произойдет отказ изделия. Вероятность безотказной работы определяют по формуле: $P(t) = \frac{N-n}{N} = \frac{1-n}{N}$, где N – первоначальное число изделий; n – число отказавших изделий за время t. Вероятность безотказной работы сложного изделия равна произведению вероятностей безотказной работы отдельных его элементов: $P(t) = P_1(t) \cdot P_2(t) \cdot \ldots \cdot P_n(t)$. Интенсивность отказов $\lambda(t)$ – отношение числа n отказавших в единицу времени t изделий к числу изделий $N-n$, исправно работающих в данный момент: $\lambda(t) = \frac{n}{(N-n)\cdot t}$. Вероятность безотказной работы можно оценить по интенсивности отказов: $P(t) = 1-\lambda(t)\cdot t$.

Долговечность – свойство изделия длительно сохранять работоспособность до наступления предельного состояния при соблюдении норм эксплуатации. Под предельным, понимают такое состояние изделия, при котором его дальнейшая эксплуатация недопустима или нецелесообразна. Долговечность характеризуется – техническим и гамма-процентным ресурсами. Технический ресурс – суммарная наработка изделия от начала эксплуатации до перехода в предельное состояние. Гамма-процентный ресурс – суммарная наработка, в течение которой изделие не достигает предельного состояния с вероятностью , выражен-

ной в процентах (часто $\gamma = 90\%$). Наработка – продолжительность или объем работы изделия. Назначенный ресурс – суммарная наработка, при которой прекращается эксплуатация изделия независимого от его состояния.

Ремонтопригодность – это приспособленность изделия к предупреждению, обнаружению и устранению отказов.

Отказ – событие, заключающееся в нарушении работоспособности устройства. Постепенные (износные) отказы характеризуются возникновением в результате постепенного протекания того или иного процесса повреждения, прогрессивно ухудшающего выходные параметры объекта. Внезапные отказы возникают в результате сочетания неблагоприятных факторов и случайных внешних воздействий, превышающих возможности объекта к их восприятию. Внезапные отказы характеризуются скачкообразным характером перехода объекта из работоспособно в неработоспособное состояние. Сложный отказ включает особенности двух предыдущих отказов. К полным отказам относятся отказы, после которых использование объекта по назначению невозможно. Частичные отказы - отказы, после возникновения которых объект может быть использован по назначению, но с меньшей эффективностью или когда вне допустимых пределов находятся значения не всех, а одного или нескольких выходных параметров.

Под показателями надежности следует понимать количественные характеристики одного или нескольких критериев надежности, при этом те показатели, которое относится к одному из критериев – единичные, те которые к нескольким – комплексные. Показатели для восстанавливаемых и невосстановимых устройств различны.

Облачные системы относятся к восстанавливаемым системам. Для облачных систем хранения и обработки данных основным показателем надежности является: доступность услуг и эксплуатационная надежность.

Основные риски облачных вычислений связаны с эксплуатационной надежностью и доступностью услуг.

После удаления программ или в следствии некоторых аппаратных дефектов могут возникать ошибки, которые могут повлиять на работу системы и тем самым вызвать критические сбои в работе целой системы. Даже если сервер будет недоступен в течении нескольких секунд, то все запросы котрые были получены будут потеряны и многие пользователи не смогут получить ответ на их запрос. Для борьбы с подобными ошибками можно использовать повторную отправку запросов или пакетов, но использование данного подхода влечет за собой

большие задержки в обработки запросов. Это может привести к резкому скачку нагрузки на обслуживающие серверы, так как пользователи не дождавшись ответа на запрос будут отправлять повторные запросы.

Доступность услуг может быть вычислена по формуле (2.1).

$$\text{Д} = \frac{T_{\text{О}}}{T_{\text{О}} + T_{\text{П}}}, \tag{2.1}$$

где $T_{\text{О}}$ – время наработки на отказ, $T_{\text{П}}$ – время простоя системы.

Значения $T_{\text{О}}$ и $T_{\text{П}}$ могут быть вычислены как теоретически так и практически, либо на основе предыдущих данных о работе системы в целом. Но не в каждом случае имеется возможность хранить данные всех мониторингов системы. При помощи уравнения (2.2) можно производить расчеты доступности системы на основе времени простоя $T_{\text{П}}$ и запланированного времени работы системы $T_{\text{ЗП}}$.

$$\text{Д} = \frac{T_{\text{ЗП}} - T_{\text{П}}}{T_{\text{ЗП}}}. \tag{2.2}$$

Эксплуатационная надежность может быть найдена по формуле (2.3)

$$H_{\text{Э}} = \frac{\mu_{\text{У}}}{\mu_O} \times 100\%, \tag{2.3}$$

где $\mu_{\text{У}}$ – количества успешных запросов, μ_O – общее количество запросов. Так как в настоящее время надежность очень высока, то удобнее для вычисления надежности использовать формулу (2.4), в которой вычисляется количество неудачных запросов на 1000000 запросов.

$$DPM = \frac{\mu_O - \mu_{\text{У}}}{\mu_O} \times 1000000 = \frac{\mu_{\text{НУ}}}{\mu_O} \times 1000000. \tag{2.4}$$

Уравнение (2.5) преобразует значение неудачных попыток на миллион в значение надежности сервиса, а уравнение (2.6) преобразовывает значение надежности сервиса в эксплуатационную надежность.

$$H = \frac{1000000 - DPM}{1000000} \times 100\% \tag{2.5}$$

$$DPM = (100\% - H) \times 100000. \tag{2.6}$$

Реальное время или скорость выполнения запросов облачным сервисом оценивается временным показателем T_i. К ним можно отнести показатели производительности, пропускной способности, среднего времени обработки запроса, среднего времени ожидания запроса до начала его обработки. Использование СОК при облачных вычислениях позволяет производить реконфигурацию

вычислительной структуры, при отказе одного или нескольких облачных серверов. Перераспределение данных решаемых задач между облачными серверами позволяет сохранить работоспособность всей вычислительной модели, за счет снижения показателей, в том числе и T_i.

Для повышения надежности хранимых данных используют методы:

– резервирования файлов;
– архивного копирования файлов;
– ограничения доступа к информации;
– введение избыточности;
– применение системы остаточных классов.

При резервировании файлов происходит создание их копий на машинных носителях информации и систематическое их обновление в случае изменения резервируемых файлов. В этом способе получается простая копия одного или нескольких файлов или файловой структуры. Основным недостатком данного метода является то, что для хранения резервных копий требуется объем дискового пространства равный объему занимаемому исходными файлами. При хранении больших объемов информации будет заметно увеличивать затраты на содержание и обслуживание оборудования.

Метод архивного хранения использует такую же схему хранения, за исключением того, что производится сжатие резервной копии перед записью на носитель информации с целью уменьшения занимаемого дискового пространства. При таком резервировании создается один архивный файл, представляющий собой набор из одного или нескольких сжатых файлов, откуда их можно извлечь в первоначальном виде. Размер сжатого файла может быть до десяти раз меньше размера файла-оригинала. Степень сжатия зависит, во-первых, от типа файла, а во-вторых, от программы-архиватора. Больше всех сжимаются файлы баз данных и текстовые файлы, а меньше всех – двоичные программные файлы (типа EXE и COM) и картинки. Архивный файл содержит оглавление, позволяющее узнать, какие файлы содержатся в архиве. Данный способ позволяет сократить затраты по сравнению со способом рассмотренным выше, но он мало подходит для облачных структур.

Вычислительная модель обработки данных в облачных системах с использованием СОК, состоит из k рабочих серверов и r контрольных серверов. При отказе рабочих серверов они могут быть заменены контрольными и наоборот. Предъявляемое требование к вычислительной схеме по обеспечению гаранти-

Таблица 3 — Статистика сбоев жестких дисков

Производитель	Размер диска	Кол-во дисков (шт.)	Кол-во диско-часов	Годовой процент отказов	Вероятность отказа облака
HGST	2 TB	4 264	399 203	1.74%	0.000005
HGST	3 TB	998	89 923	0.81%	0.0000005
HGST	4 TB	2 706	243 312	1.05%	0.000001
Seagate	4 TB	34 729	2 849 179	2.54%	0.000016
Toshiba	3 TB	47	423	8.63%	0.000642
WDC	2 TB	133	11 617	12.57%	0.002
WDC	3 TB	1 054	94 384	3.09%	0.000029
WDC	6 TB	458	41 220	5.31%	0.00015

рованной защиты от выдачи недостоверного результата обуславливает необходимость сохранения в работоспособном состоянии k_{min} рабочих и хотябы одного контрольного серверов. Таким образом, надежная структура вычислительной модели будет соответствовать известному способу скользящего резервирования с нагруженным состоянием резервных элементов.

При проведении вычислений на облачных серверах каждая из перечисленных выше возможных отказов системы может произойти как в отдельности, так и совместно. Для вычисления общей надежности облачного сервера воспользуемся формулой вычисления вероятности полного отказа (2.7).

$$P_{\mathbf{П}} = P_{\text{ПО}} + P_T + P_{DDoS}, \tag{2.7}$$

где $P_{\text{ПО}}$ – вероятность отказа программного обеспечения, P_T – вероятность отказа жестких дисков, P_{DDoS} – вероятность проведения DDoS атаки на облачный сервер.

Как было сказано в разделе 2.1, что вероятность отказа программного обеспечения очень низкая и не зависит от конфигурации системы обработки, и равняется $P_{\text{ПО}} = 0.3 \cdot 10^{-3}$. Данное значение вероятности отказа программного обеспечения будем использовать при расчетах надежности различных систем обработки данных.

Согласно отчету об DDoS атаках в первом квартале 2016 года самая длинная атака длилась 197 часов, или (8,2 дней). Используя определение геометрической вероятности, вероятность отказа в доступе службы составляет

Таблица 4 — Вероятность полного отказа мультиоблачной системы

Производитель	HGST	HGST	HGST	Seagate
Объем жесткого диска	2 TB	3 TB	4 TB	4 TB
Вероятность отказа	$0.535 \cdot 10^{-3}$	$0.5305 \cdot 10^{-3}$	$0.531 \cdot 10^{-3}$	$0.546 \cdot 10^{-3}$
Производитель	WDC	WDC	WDC	Toshiba
Объем жесткого диска	6 TB	2 TB	3 TB	3 TB
Вероятность отказа	$0.68 \cdot 10^{-3}$	$0.253 \cdot 10^{-2}$	$0.559 \cdot 10^{-3}$	$0.1172 \cdot 10^{-2}$

$8{,}2/90 \approx 0.09$. Принимая среднее между самой длинной DDoS атакой на веб-сервис, а также события, когда не было DDoS атаки, мы получаем вероятность отказа доступа к данным в результате DDoS атаки равна $P_{DDoS} = 0{,}05$ [41].

Из результатов расчета полного отказа облачной системы при использовании различных конфигураций (таблица 4) можно сделать вывод о том, что вероятность отказа существующих систем не удовлетворяте требованиям предъявляемым к системам хранения и обработки больших данных. Нами в работе предлагается использование СРД основанных на СОК, для улучшения характеристик надежности, отказоустойчивости облачных систем.

2.2 Модификация облачных вычислений на основе системы остаточных классов

Применение СРД для облачных систем хранения и обработки данных позволяет производить разделение данных между облачными хранилищами, и в этом случае разделенные данные (части) хранятся в различных облачных хранилищах, количество частей может изменяться в зависимости от предъявляемым требованиям. При наличии любых k частей может быть произведено восстановление хранимых данных.

Каждая часть по отдельности ничего не значит. Данные восстанавливаются только в том случае если собрать все части вместе. Основным недостатком

Рисунок 2.1 — Примеры модификации облачных схем хранения и обработки данных на основе системы остаточных классов

данного метода является то, что при утере или ошибке в одной из частей, нет возможности восстановить хранимые данные.

Для устранения этого недостатка в СРД применяются (k, n) пороговые схемы. Пороговые схемы – это схемы, в которых достаточно знать k из n частей. К таким схемам относятся схема Блэкли, схема Шамира, схемы, основанные на Китайской теореме об остатках или схемы, основанные на решении систем уравнений.

Применение СОК позволяет производить вычисление и хранение в облаках различным образом. На рисунке 2.1 представлена схема при которой информация разделяется между выбранными пользователем облачными провайдерами [2].

Простой способ распределения модулей между облакам является создание непересекающихся подмножеств. При использовании данного подхода вероятность успеха сговора между двумя облаками будет возрастать по экспоненте

$$f\left(n, k_1 + k_2\right) = \frac{1}{S^{n-(k_1+k_2)}}.$$

Наилучшим вариантом является распределение модулей, при котором вероятность успеха сговора y облаков была меньше суммы их индивидуальных вероятностей успеха, т.е.

$$\sum_{i=1}^{y} f_i\left(n, k\right) < \frac{1}{S_P^{n-yk}}.$$

Это может быть достигнуто с помощью следующей схемы с резервированием, в которой каждому из облаков будет передано $k = q + l$ модулей, где q – различные не пересекающиеся модули, и l – модули являющиеся избыточными и пересекаются с модулями переданными различным облакам.

Таким образом, если два облака сговорились, они могут собрать не $2k$ модулей, а меньше $2k$. Конечно, если все облака сговорились, то они будут иметь

все модули, необходимые для восстановления данных представленных в СОК. Для полного восстановления данных представленных в СОК, необходим сговор, по крайней мере $\frac{n}{q} + l$ облаков. Пока этого не произойдет, восстановить данные представленные в СОК не представляется возможным.

2.3 Подходы к повышению надежности хранения данных большой разрядности в облачной среде на основе системы остаточных классов

Распределенные системы хранения и обработки информации сложны в организации. При этом одной из целей создания ряда таких систем является именно повышение надежности хранения данных. Сюда в первую очередь относится обеспечение доступности данных при хранении. Однако стоит отметить, что характер угроз, с которыми может столкнуться распределенная система в процессе функционирования, довольно разнообразен [41]. Следует учитывать высокую нагрузку таких систем, возможно гетерогенную структуру узлов, сложность решаемых задач, большие объемы данных. Все эти факторы влияют на хранимые и обрабатываемые данные, внося неопределенность в работу системы [1, 105], что отражается на возможных проблемах с целостностью и достоверностью получаемой в процессе обработки и хранения информации.

При решении проблемы повышения надежности хранения и обработки информации можно выделить три подхода [88]: репликация данных, применение помехоустойчивых кодов и применение арифметических корректирующих кодов.

При использовании репликации чаще всего применяются схемы однократного и двукратного резервирования [105]. Однократное резервирование позволяет заменять вышедшие из строя хранилища данных и не обеспечивает обнаружение ошибки в данных пока узел не выйдет из строя. Двукратное резервирование обычно использует мажоритарную схему выбора 2-х из 3-х. Этот способ требует больших аппаратных затрат так как приводит к высокой избыточности.

Информационные последовательности, представляющие с точки зрения математической логики двоичные числа A в СОК представляются неотрицательными остатками от деления на модули p_i, $i = 1,2,\ldots,n$. Единственность представления числа в СОК гарантируется КТО.

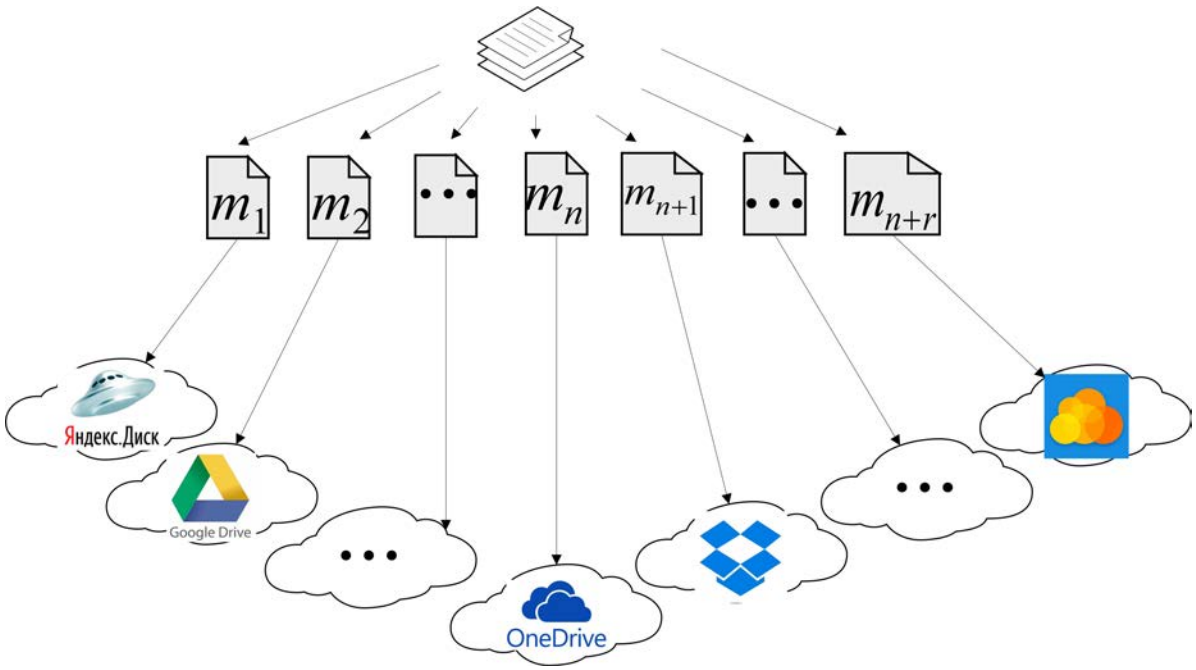

Рисунок 2.2 — Представление данных, передаваемых в облачные хранилища, в СОК

Возможности обнаружения и исправления ошибок в числах СОК проявляются при введении в нее контрольных (избыточных) модулей [100]. В качестве избыточных модулей берутся числа p_j, $j = n+1,n+2,\ldots,n+r$, удовлетворяющие условиям взаимной простоты между собой и по всей совокупности исходных неизбыточных (рабочих) модулей. На рисунке 2.2 представлена схема представления данных передаваемых в облачное хранилище.

Искажения в данных возникают при сбоях и неисправностях в процессе передачи, хранения и обработки их в облачных хранилищах [37]. Так, по сведениям только из открытых источников, доступ к информации Amazon был ограничен в течение длительного времени из-за DDoS-атак в 2009 году [115]. В 2013 году Amazon, Microsoft и Google объявили о серии сбоев в облаках. Технические сбои и потери данных из-за сбоев питания регулярно возникают у Amazon, Dropbox, Microsoft, Google и Yandex Disk. В первом квартале 2014 года Dropbox испытал перебои в обслуживании дважды. В 2013 году облачный провайдер Nirvanix был объявлен банкротом, что привело к прекращению работы всех его сервисов [54].

При хранении данных по схеме представленной на рисунке 2.2 можно восстанавливать искаженные данные отдельных хранилищ за счет избыточных данных по модулям $p_{n+1},\ldots,p_{n+r}$ Согласно [117] для гарантированного исправления ошибки по одному модулю необходимо наличие 2-х контрольных модулей

$p_{n+1}, \ldots, p_{n+2}$ по абсолютной величине превышающих любой из рабочих модулей p_i. Для исправления двукратной ошибки – 4-х и т.п.

Пример. Пусть СОК имеет систему оснований $p_1 = 3$, $p_2 = 5$, $p_3 = 7$, $p_4 = 15$, $P = 1155$. P' в два раза больше P_k, то есть условия работы алгоритма соблюдены. Найдем синдромы для ошибки $B = (1, 3, 0, 0)$. По двум первым основаниям, число $B = 13$. Первый синдром $\Omega_1 = (6, 2)$. Число, обратное 13 в диапазоне P_k есть 2. Число, обратное 2 в диапазоне P' есть 75, $\Omega_2 = 75 = (5, 9)$. Пусть теперь задано правильное число $A = (2, 3, 1, 8)$. Изменим число A, придав ему ошибку B: $\tilde{A} = (1, 0, 1, 8)$.

Рассмотрим ошибочное число $\tilde{A}$. Что бы определить ошибку, расширим число $\tilde{A}$ из диапазона P_k на основания p_3 и p_4, $(1, 0)_{P_k} = 10$. Получим число $\tilde{A}' = (1, 0, 3, 10)$. Вычислим синдром для числа $\tilde{A}$: $\tilde{\Omega} = (5, 9)$, что соответствует второму синдрому ошибки B. Таким образом, сделанная нами ошибка обнаружена и устранена.

Рассмотренный алгоритм позволяет находить и устранять любые ошибки в рабочем диапазоне. Применение ускоренного способа расширения системы оснований и табличного метода коррекции чисел делает данный метод быстрым и эффективным. К недостаткам данного метода следует отнести невозможность контроля ошибок по контрольным основаниям и значительную величину диапазона контрольных оснований по сравнению с рабочим диапазоном.

2.4 Разработка метода надежного хранения данных в облачных хранилищах на основе системы остаточных классов

Для построения надежных систем хранения данных в облачных хранилищах будем использовать СОК совместно с многоуровневыми СРД. Для хранения на сторонних серверах информация, в начале подвергается распределению между облачными серверами. Алгоритм схемы надежного хранения данных приведен на рисунке 2.3. В случае выхода из строя или недоступности одного или нескольких облачных хранилищ, информация может быть восстановлена из имеющихся частей [41].

Рисунок 2.3 — Модель многомерного хранения данных с использованием СОК и СРД

Покажем возможность восстановления данных при недоступности некоторых облачных серверов. Рассмотрим пример когда доступны два облака №1 и №3.

Таким образом известно S_{13}, S_{12}, S_{22}, S_{32}. Используя значения S_{13} и S_{12} мы восстанавливаем значение S_1. Таким образом значения S можно представить в виде $S = z\left(2^b - 1\right) + S_1$, где $z < 2^b$. Перепившим S в другом виде

$$S = z\left(2^b - 1\right) + S_1 = z2^b + S_1 - z = \frac{z - z_0}{2}\left(2^{b+1} - 1\right) + S_1 - \frac{z - z0}{2} + z_0 2^b \quad (2.8)$$

Из формулы (2.8) следует, что

$$S_{22} = S_2 \bmod 2^{m_{12}} = S_1 - z \bmod 2^{m_{12}} \quad (2.9)$$

$$S_{32} = S_3 \bmod 2^{m3} = \begin{bmatrix} \left(S_1 - \frac{z - z_0}{2}\right) \bmod 2^{m_3} \\ \left(S_1 - \frac{z - z_0}{2} + 1\right) \bmod 2^{m_3} \end{bmatrix}, \quad (2.10)$$

где z_0 – младший бит слова z. Из формулы (2.9) вычислим последние m_{12} бит z равные

$$z \bmod 2^{m_{12}} = (S_1 - S_{22}) \bmod 2^{m_{12}}. \quad (2.11)$$

Используя формулу (2.10) и (2.11) вычислим значения бит z_{m_3} если $m_3 = m_{12}$ и значения бит m_3 и m_{12} если $m_3 = m_{12} + 1$. Таким образом зная четыре части мы можем восстановить, только $b + m_3 + 1$ бит из $2b - 1$ бит информации. Следовательно мы можем вычислить, только $\frac{3}{4}$ шифр текста а $\frac{1}{4}$ остается неизвестной.

Восстановление информации при доступности Облако 3 и Облако 5 доказывается аналогично приведенному выше доказательству.

Рассмотрим случай доступности облаков №2 и №4, следовательно S_{11}, S_{21}, S_{23}, S_{33}. Используя S_{21}, S_{23} по теореме **??** восстановим значение S_2. Следовательно, S можем представить в виде $S = z2^b + S_2$, где $z < 2^{b-1}$. Перепишем S следующим образом

$$S = z2^b + S_2 = z\left(2^b - 1\right) + S_2 + z = \frac{z - z_0}{2}\left(2^{b+1} - 1\right) + S_2 + \frac{z - z_0}{2} + z_0 2^b \quad (2.12)$$

Таким образом из формулы (2.12) следует, что

$$S_1 = (S_2 + z) \bmod \left(2^b - 1\right) \quad (2.13)$$

$$S_3 = \left(S_2 + \frac{z - z_0}{2} + z_0 2^b\right) \bmod \left(2^{b+1} - 1\right) \quad (2.14)$$

Исследуем формулу (2.14). Пусть z_0, тогда формула (2.14) примет вид $S_3 = \left(S_2 + \frac{z}{2}\right) \bmod \left(2^{m+1} - 1\right)$, так как $S_2 < 2^b$ и $\frac{z}{2} < 2^{b-2}$, то $S_2 + \frac{z}{2} < 2^b + 2^{b-2} < 2^{b+1} - 1$, следовательно

$$S_3 = S_2 + \frac{z}{2} \text{ и } z = 2 \cdot (S_3 - S_2) \quad (2.15)$$

Пусть $z_0 = 1$, тогда формула (2.14) примет вид $S_3 = \left(S_2 + \frac{z-1}{2} + 2^b\right) \bmod \left(2^{m+1} - 1\right)$, следовательно

$$S_3 = \begin{cases} S_2 + \frac{z-1}{2} + 2^b, & \textit{если } \Delta < 2^{b+1} - 1 \\ S_2 + \frac{z-1}{2} - 2^b, & \textit{иначе} \end{cases},$$

где $\Delta = S_2 + \frac{z-1}{2} + 2^b$.

Вычислим значение

$$z = \begin{cases} 2S_3 - 2S_2 - 2^{b+1} + 1, & \widetilde{\Delta} \geq 0 \\ 2S_3 - 2S_2 + 2^{b+1} - 1, & \widetilde{\Delta} < 0 \end{cases} \quad (2.16)$$

где $\widetilde{\Delta} = 2S_3 - 2S_2 - 2^{b+1} + 1$.

При b – четном $b = 2m_{12}$, следовательно $\left(2^{m_{12}}\right) \mid \left(2^b - 1\right)$ и выражение $S_{11} = S_1 \bmod \left(2^{m_{12}} - 1\right) = \left((S_2 + z) \bmod \left(2^b - 1\right)\right) \bmod \left(2^{m_{12}} - 1\right) = (S_2 + z) \bmod \left(2^{m_{12}} - 1\right)$, значит

$$z \bmod \left(2^{m_{12}} - 1\right) = |S_{11} - S_2| \bmod \left(2^{m_{12}} - 1\right) \quad (2.17)$$

Если m_{12} – четно, то $3|\gcd\left(2^{m_{12}}-1, 2^{m_3+1}-1\right)$. Пусть

$$(2S_{33} - 2S_2) \bmod 3 = x, \tag{2.18}$$

тогда используя формулу (2.16) получим, что

$$\begin{aligned}\left(2S_3 - 2S_2 - 2^{b+1} + 1\right) \bmod 3 = \\ = \left(x - 2 \cdot \left(2^b - 1\right) - 1\right) \bmod 3 = \\ = (x+2) \bmod 3,\end{aligned} \tag{2.19}$$

$$\begin{aligned}\left(2S_3 - 2S_2 + 2^{b+1} - 1\right) \bmod 3 = \\ = \left(tx + 2 \cdot \left(2^b - 1\right) + 1\right) \bmod 3 = \\ = (x+1) \bmod 3.\end{aligned} \tag{2.20}$$

Проверяя условия $z \bmod 3 = (S_2 + z) \bmod 3$, где $z \bmod 3$ определяется из формулы (2.17) и из формул (2.18)-(2.20) определим значение $z \bmod 3 = \left(2^{m_3+1}-1\right)$. Зная значения $z \bmod \left(2^{m_3+1}-1\right)$ и $z \bmod \left(2^{m_{12}}-1\right)$ используя теорему **??** восстановим значение z и найдем значение S.

Если m_{12} – нечетно, то $\gcd\left(2^{m_{12}}-1\ ,2^{m_3+1}-1\right) = 1$. Используя формулу (2.17) вычислим значение $z \bmod \left(2^{m_{12}}-1\right)$. Пусть $m_{12} = m$, тогда $m_3 = m+1$. Определим параметры системы остаточных классов:

$$\begin{aligned}P &= \left(2^m - 1\right)\left(2^{m+2} - 1\right), \\ P_1 &= 2^{m+2} - 1, \\ P_2 &= 2^m - 1,\end{aligned}$$

$$B_1 = \frac{1}{P_1} \bmod p_1 \cdot P_1 = \frac{1}{2^{m+2}-1} \cdot \left(2^{m+2} - 1\right) = \left(\frac{2^{m+1}-1}{3}\right) \bmod \left(2^m - 1\right) \cdot \left(2^{m+2} - 1\right) \tag{2.21}$$

Так как $3|\left(2^{m+1}-1\right)$, то формулу (2.21) можно переписать в виде:

$$B_1 = \frac{2^{m+1}-1}{3} \cdot \left(2^{m+2} - 1\right) \tag{2.22}$$

$$B_2 = \left(\frac{4 \cdot \left(2^{m+3} - 1\right)}{-3} \right) \bmod \left(2^{m+2} - 1\right) \cdot \left(2^m - 1\right) \tag{2.23}$$

Так как $3 | \left(2^{m+3} - 1\right)$, то формула (2.23) примет вид

$$B_2 = -\frac{4}{3} \left(2^{m+3} - 1\right) \cdot \left(2^m - 1\right) \tag{2.24}$$

Учитывая, что в формуле при вычисление значения z по теореме **??** все вычисления выполняются по модулю P, получим

$$B_2 = 3 \cdot P - \frac{4}{3} \left(2^{m+3} - 1\right) \cdot \left(2^m - 1\right) = \frac{1}{3} \left(2^m - 1\right) \cdot \left(2^{m+2} - 5\right) \tag{2.25}$$

Пусть $\left(S_{11} - S_2\right) \bmod \left(2^m - 1\right) = a_1$ и $z \bmod \left(2^{m+2} - 1\right) = \left(2S_{33} - 2S_2\right) \bmod \left(2^{m+2} - 1\right) = a_2$, тогда

$$\left(2S_{33} - 2S_2 - 2^{n+1} + 1\right) \bmod \left(2^{m+2} - 1\right) = \begin{cases} a_2 - 2^{m-1} + 1, & \textit{если } a_2 \geq 2^{m-1} - 1, \\ a_2 + 7 \cdot 2^{m-1}, & \textit{иначе.} \end{cases}$$

$$\left(2S_{33} - 2S_2 + 2^{b+1} - 1\right) \bmod \left(2^{m+2} - 1\right) = \begin{cases} a_2 + 2^{m-1} - 1, & \textit{если } a_2 \geq 7 \cdot 2^{m-1}, \\ a_2 - 7 \cdot 2^{m-1}, & \textit{иначе.} \end{cases}$$

1. Если $a_2 < 2^{m-1} - 1$, тогда:

Пусть $\left(a_1 B_1 + a_2 B_2\right) \bmod P = y$, тогда

$$z = \left[\begin{array}{l} y, \\ \left(y + \frac{4}{3} \left(2^{m-1} - 1\right) \cdot \left(2^m - 1\right)\right) \bmod P \\ \left(y + \frac{1}{3} \left(2^m - 1\right) \cdot \left(2^{m+1} - 1\right)\right) \bmod P \end{array} \right. .$$

Так как $3 | \left(2^{m-1} - 1\right)$, то $2^{2m-1} < \frac{4}{3} \left(2^{m-1} - 1\right) \left(2^m - 1\right) < 2^{2m}$. Учитывая, что $3 | \left(2^{m-1} - 1\right)$, получим $2^{2m-1} < \frac{1}{3} \left(2^m - 1\right) \left(2^{m+1} - 1\right) < 2^{2m}$. Если $0 < y < 2^{2m-2}$, то

$$2^{n-1} < 2^{2m-1} < y + \tfrac{4}{3} \left(2^{m-1} - 1\right) \left(2^m - 1\right) < 2^{2m} + 2^{2m-2} < P,$$
$$2^{n-1} < 2^{2m-1} < y + \tfrac{1}{3} \left(2^m - 1\right) \left(2^{m+1} - 1\right) < 2^{2m} + 2^{2m-2} < P.$$

Так как $\left(y + \frac{1}{3} \left(2^m - 1\right) \left(2^{m+1} - 1\right)\right) - \left(y + \frac{4}{3} \left(2^{m-1} - 1\right) \left(2^m - 1\right)\right) = 2^m - 1$, то $0 \geq y + \frac{4}{3} \left(2^{m-1} - 1\right) \left(2^m - 1\right) < 2^{2m-2} - 2^m + 1$, то возможны два значения z.

2. Если $2^{m-1}-1 \geq a_2 \geq 7 \cdot 2^{m-1}$, тогда

$$z = \left[\begin{array}{l} y, \\ \left(y - \frac{4}{3}\left(2^{m-1}-1\right) \cdot \left(2^m - 1\right)\right) \bmod P \\ \left(y + \frac{1}{3}\left(2^m - 1\right) \cdot \left(2^{m+1}-1\right)\right) \bmod P \end{array} \right. .$$

Если $0 < y < 2^{2m-2}$, то

$$2^{b-1} < 2^{2m+1} < P + y - \tfrac{1}{3}\left(2^{m-1}-1\right)\left(2^m - 1\right) < P,$$
$$2^{b-1} < 2^{2m-1} < y + \tfrac{1}{3}\left(2^m - 1\right)\left(2^{m+1}-1\right) < 2^{2m} + 2^{2m-2} < P.$$

Так как

$$c\left(y + \frac{1}{3}\left(2^m - 1\right)\left(2^{m+1}-1\right)\right) - \left(y - \frac{4}{3}\left(2^{m-1}-1\right)\left(2^m - 1\right)\right) = \\ = \frac{2\left(2^m - 1\right)\left(2^{m+1}-1\right)}{3} > 2^{2m-2},$$

то значение z мы можем восстановить однозначно.

3. Если $a_2 > 7 \cdot 2^{m-1}$, тогда

$$z = \left[\begin{array}{l} y, \\ \left(y - \frac{4}{3}\left(2^{m-1}-1\right) \cdot \left(2^m - 1\right)\right) \bmod P \\ \left(y + \frac{4}{3}\left(2^{m-1}-1\right) \cdot \left(2^m - 1\right)\right) \bmod P \end{array} \right. .$$

Если $0 < y < 2^{2m-2}$, то

$$2^{b-1} < 2^{2m+1} < P + y - \tfrac{4}{3}\left(2^{m-1}-1\right)\left(2^m - 1\right) < P,$$
$$2^{b-1} < 2^{2m-1} < y + \tfrac{4}{3}\left(2^{m-1}-1\right)\left(2^m - 1\right) < 2^{2m} + 2^{2m-2} < P.$$

Так как

$$l\left(y + \frac{4}{3}\left(2^{m-1}-1\right)\left(2^m - 1\right)\right) - \left(y - \frac{4}{3}\left(2^{m-1}-1\right)\left(2^m - 1\right)\right) = \\ = \frac{8}{3}\left(2^{m-1}-1\right)\left(2^m - 1\right) > 2^{2m-2},$$

то значение z мы можем восстановить однозначно.

Вычислим три возможных значения z по формулам (2.15) и (2.16) восстановим z с использованием китайской теоремы об остатках проверяем на выполнение условия $z < 2^{b-1}$ определим искомое значение z.

При b–нечетном $b+1 = 2m$, следовательно $(2^{m_3}-1) \,|\, \left(2^{b+1}-1\right)$ и выражение $S_{33} = S_3 \bmod (2^{m_3}-1) = \left(\left(S_2 + \frac{z-z_0}{2} + z_0 2^b\right) \bmod \left(2^{b+1}-1\right)\right) \bmod (2^{m_3}-1) = \left(S_2 + \frac{z-z_0}{2} + z_0 2^{m_3-1}\right) \bmod (2^{m_3}-1)$, следовательно, значение $\left(\frac{z-z0}{2}\right) \bmod (2^{m_3}-1)$ равно,

$$\frac{z-z_0}{2} \bmod (2^{m_3}-1) = \left(S_{33} - S_2 - z_0 2^{m_3-1}\right) \bmod \left(2^{m_{12}-1}\right) \tag{2.26}$$

Из формулы (2.26) вычислим значения получим,

$$S_{11} = \left[\begin{array}{l} (S_2 + z) \bmod (2^{m_{12}}-1) \\ \left(S_2 + z - 2^{m_{12}-1} + 1\right) \bmod (2^{m_{12}}-1) \end{array} \right. . \tag{2.27}$$

Выражая из формулы (2.27) значения $|z|_{2^{m_{12}-1}}$, получим

$$z \bmod (2^{m_{12}}-1) = \left[\begin{array}{l} (S_{11} + S_2) \bmod (2^{m_{12}}-1) \\ (S_{11} - S_2 - z + 2^{m_{12}}-1) \bmod (2^{m_{12}}-1) \end{array} \right. . \tag{2.28}$$

Используя китайскую теорему об остатках восстановим четыре возможных значения z. Проверяя каждое полученное значение z на предположение при котором оно получилось, определим истинное значение z [41].
Следовательно используя облака №2 и №4 восстановим исходное значение информации.

Полный цикл работы описанного выше алгоритма условно можно разделить на два этапа:

1 Этап. Разделение информации.

2 Этап. Восстановление информации.

Приведем примеры работы алгоритма при доступности различных серверов хранения информации.

Пример 1. 1 Этап. Разделение информации ($b = 32$ бит).

На вход блока Информация поступает слово S длинной в $2b-1 = 63$ бита $S = 5850495393454642923$.

Вычислим

$$S_1 = S \bmod \left(2^b - 1\right) = S \bmod \left(2^{32} - 1\right) = 2015197563,$$
$$S_2 = S \bmod \left(2^b\right) = S \bmod \left(2^{32}\right) = 653022955,$$
$$S_3 = S \bmod \left(2^{b+1} - 1\right) = S \bmod \left(2^{33} - 1\right) = 1334110259,$$
$$m_{12} = \left\lfloor \frac{b+1}{2} \right\rfloor = 16,\ m_3 = \left\lfloor \frac{b+2}{2} \right\rfloor = 17.$$

Запускается три параллельных процесса

Первый процесс

На вход блока Доля 1 поступает значение $S_1 = 2015197563$. Производится вычисление проекции S_1:

$$S_{11} = S_1 \bmod (2^{m_{12}} - 1) = S_1 \bmod \left(2^{16} - 1\right) = 61848 S_1,$$
$$S_{12} = S_1 \bmod (2^{m_{12}}) = S_1 \bmod \left(2^{16}\right) = 31099,$$
$$S_{13} = S_1 \bmod \left(2^{m_{12}+1} - 1\right) = |S_1| \bmod \left(2^{17} - 1\right) = 112009,$$

Второй процесс

На вход блока Доля 2 поступает значение $S_2 = 653022955$. Производится вычисление проекции S_2:

$$S_{21} = S_2 \bmod \left(2^{m12} - 1\right) = S_{12} \bmod \left(2^{16} - 1\right) = 32215,$$
$$S_{22} = S_2 \bmod \left(2^{m12}\right) = S_2 \bmod \left(2^{16}\right) = 22251,$$
$$S_{23} = S_2 \bmod \left(2^{m12+1} - 1\right) = S_2 \left(2^{17} - 1\right) = 27233,$$

Третий процесс

На вход блока Доля 3 поступает значение $S_3 = 1334110259$. Производится вычисление проекции S_3:

$$S_{31} = S_3 \bmod \left(2^{m3} - 1\right) = S_3 \bmod \left(2^{17} - 1\right) = 69621,$$
$$S_{32} = S_3 \bmod \left(2^{m3}\right) = S_3 \bmod \left(2^{17}\right) = 59443,$$
$$S_{33} = S_3 \bmod \left(2^{m3+1} - 1\right) = S_3 \bmod \left(2^{18} - 1\right) = 64532,$$

Таким образом, результатом работы алгоритма является девять частей проекции исходной информации. Произведем распределение частей проекций между пятью облаками по схеме представленной на рисунке 2.4:

2 Этап. Восстановление информации

Произведем восстановление информации при доступности Облако 1, Облако 2, Облако 4, Облако 5. В начале производится запуск двух параллельных процессов:

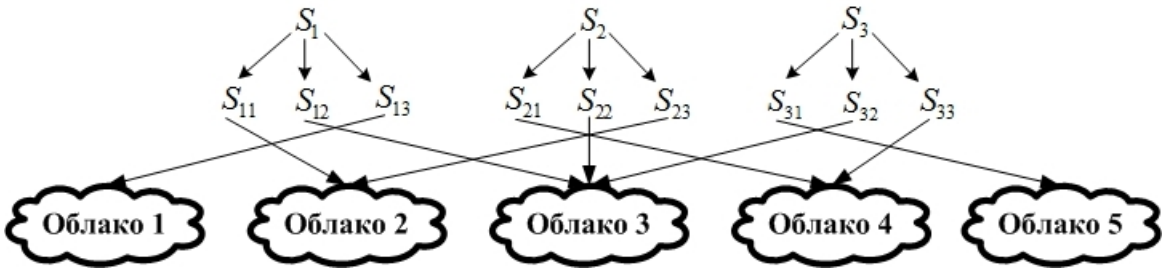

Рисунок 2.4 — Схема распределения частей проекции между серверами хранения информации

В результате работы первого процесса производится восстановление проекции S_1 по формуле (2.29)

$$S_1 = \left((S_{11} - S_{13}) \cdot \left(2^{m_{12}+1} - 1\right) + S_{13}\right) \bmod \left((2^{m_{12}} - 1) \cdot \left(2^{m_{12}+1} - 1\right)\right) \quad (2.29)$$

$$S_{11} = 61848, S_{13} = 112009, m_{12} = 16$$

$$S_1 = \left((61848 - 112009) \cdot \left(2^{17} - 1\right) + 112009\right) \bmod \left(\left(2^{17} - 1\right) \cdot \left(2^{16} - 1\right)\right) =$$
$$= 2015197563.$$

Результатом работы второго процесса будет получение восстановленного значения проекции S_3, восстановление производится по формуле (2.30)

$$S_3 = \left((S_{31} - S_{33}) \cdot \left(2^{m_3+1} - 1\right) + S_{33}\right) \bmod \left((2^{m_3} - 1) \cdot \left(2^{m_3+1} - 1\right)\right) \quad (2.30)$$

$$S_{31} = 69621, S_{33} = 64532, m_3 = 17$$

$$S_3 = \left((69621 - 64532) \cdot \left(2^{18} - 1\right) + 64532\right) \bmod \left(\left(2^{17} - 1\right) \cdot \left(2^{18} - 1\right)\right) =$$
$$= 1334110259.$$

Выполним вычисление значения S на основе полученных значений S_1 и S_3 по формуле 2.31

$$S = \left((S_1 - S_3) \cdot \left(2^{n+1} - 1\right) + S_3\right) \bmod \left((2^n - 1) \cdot \left(2^{n+1} - 1\right)\right) \quad (2.31)$$

$$S_1 = 2015197563, S_3 = 1334110259, b = 32$$

$$S = \left((2015197563 - 1334110259) \cdot \left(2^{33} - 1\right) +\right.$$
$$\left. +1334110259\right) \bmod \left(\left(2^{32} - 1\right) \cdot \left(2^{33} - 1\right)\right) = 58504953934546429 23.$$

В результате работы алгоритма получили восстановленное значение S.

Приведем несколько примеров работы алгоритма на этапе восстановления информации, при доступности различных серверов хранения информации.

Пример 2.

Рассмотрим случай, когда имеется доступ к двум облачным хранилищам Облако 2 и Облако 4, т.е. мы имеем доступ к четырем частям проекций: S_{11}, S_{21}, S_{23} и S_{33}, где $S_{11} = 48427$, $S_{21} = 27233$, $S_{23} = 24742$, $S_{33} = 164191$.

1. Используя значения $S_{21} = 22251$, $S_{23} = 27233$ вычислим S_2.

$$S_2 = \left((S_{21} - S_{23})\left(2^{18} - 1\right) + S_{33}\right) \bmod \left(2^{16} - 1\right) \cdot \left(2^{18} - 1\right) = 653022955$$

2. Вычислим z.

$$z \bmod (2_{17} - 1) = (S_{11} - S_2) \bmod (2_{17} - 1) = 21194$$

Пусть z_0, тогда

$$z \bmod \left(2^{19} - 1\right) = \left[\begin{array}{l} (2 \cdot (S_{33} - S_2)) \bmod \left(2^{17} - 1\right) = 281389 \\ \left(2 \cdot \left(S_{33} - S_2 + 2^{16} - 1\right)\right) \bmod \left(2^{19} - 1\right) = 412459 \end{array} \right. ,$$

Пусть $z_0 = 1$, тогда

$$z \bmod \left(2^{19} - 1\right) = \left[\begin{array}{l} \left(2 \cdot \left(S_{33} - S_2 - 2^{15}\right) + 1\right)\left(2^{19} - 1\right) = 215854 \\ \left(2 \cdot \left(S_{33} - S_2 + 2^{15} - 1\right) + 1\right) \bmod \left(2^{19} - 1\right) = 346924 \end{array} \right. ,$$

Вычислим значения $P = \left(2^{17} - 1\right) \cdot \left(2^{19} - 1\right)$, тогда $P_1 = 2^{19} - 1$ и $P_2 = 2^{17} - 1$.

$$k_1 = \left|P_1^{-1}\right|_{p_1} \cdot P_1 = 45812722347,\ k_2 = \left|P_2^{-1}\right| \cdot 22906099031.$$

Найдем значения z:

1. $z \xrightarrow{RNS} (21194, 281389)$, тогда
   $z = (a_1 \cdot k_1 + a_2 \cdot k_2) \bmod P = 340453652 < 2^{33}$ удовлетворяет условию $z < 2^{33}$.
2. $z \xrightarrow{RNS} (21194, 412459)$, тогда
   $z = (a_1 \cdot k_1 + a_2 \cdot k_2) \bmod P = 41653397069 > 2^{33}$ не удовлетворяет условию $z < 2^{33}$.

3. $z \xrightarrow{RNS} (21194, 215854)$, тогда
   $z = (a_1 \cdot k_1 + a_2 \cdot k_2) \bmod P = 11793527632 > 2^{33}$ удовлетворяет условию $z < 2^{33}$.
4. $z \xrightarrow{RNS} (21194, 215854)$, тогда
   $z = (a_1 \cdot k_1 + a_2 \cdot k_2) \bmod P = 11793527632 > 2^{33}$ удовлетворяет условию $z < 2^{33}$.

Следовательно, $z = 340543652$, восстановим значение $S = z \cdot 2^{34} + S_2 = = 5850495393454642923$.

Предложенный в 2.4, алгоритм позволяет надежно хранить информацию в облачных хранилищах. Данный алгоритм позволяет производить восстановление хранимой информации при выходе из строя или недоступности одного или нескольких облачных хранилищ. Для определения практической применимости разработанного алгоритма проведем моделирование работы на различных режимах функционирования.

Исходя из полученных значений сохранения данных в облачных серверах по отдельности, как поступает большое количество пользователей, и сразу во все хранилища, при помощи предложенного алгоритма, можно сделать вывод о том, что средняя скорость записи увеличилась на 25%. Использование схем разделения данных не значительно увеличивает объем хранимой информации, поэтому можно говорить об увеличении скорости записи на 17%, относительно стандартной записи информации на облачный сервер таблицы 5, 6, рисунок 2.5.

Таблица 5 — Скорость сохранения файлов в облачные хранилища с использованием СРД (килобайт в секунду)

n	10 KB	100 KB	1 MB	10 MB	100 MB
1024	7.6	74.0	620	1340	1397
8192	3.8	61.7	558.1	1325	1273
16384	9.0	68.3	515.1	1030	1391
32768	10.6	62.5	526.7	1367	1350

Произведем чтение информации из каждого взято по отдельности хранилища таблица 7. Как показано в пункте 3.1.1., что невозможность получения доступа к нескольким облачным хранилищам не является ограничением по работе с хранимой информацией. Использование схем разделения секрета позволяет

Таблица 6 — Средняя скорость сохранения файлов в облачные хранилища без СРД (килобайт в секунду)

Облачный сервис	10 KB	100 KB	1 MB	10 MB	100 MB
Mega	4.3	51.2	313.6	751.6	738.7
DropBox	3.6	28.8	219.9	631.9	1029.5
YandexDisk	9.8	121.17	1089.83	1973.89	2393.6
OneDrive	2.7	60,2	271,9	562	1496.9
Box	2.8	26.3	204.8	1137.7	1711.6
GoogleDrive	4.5	46.2	411.85	1073.9	1443.2

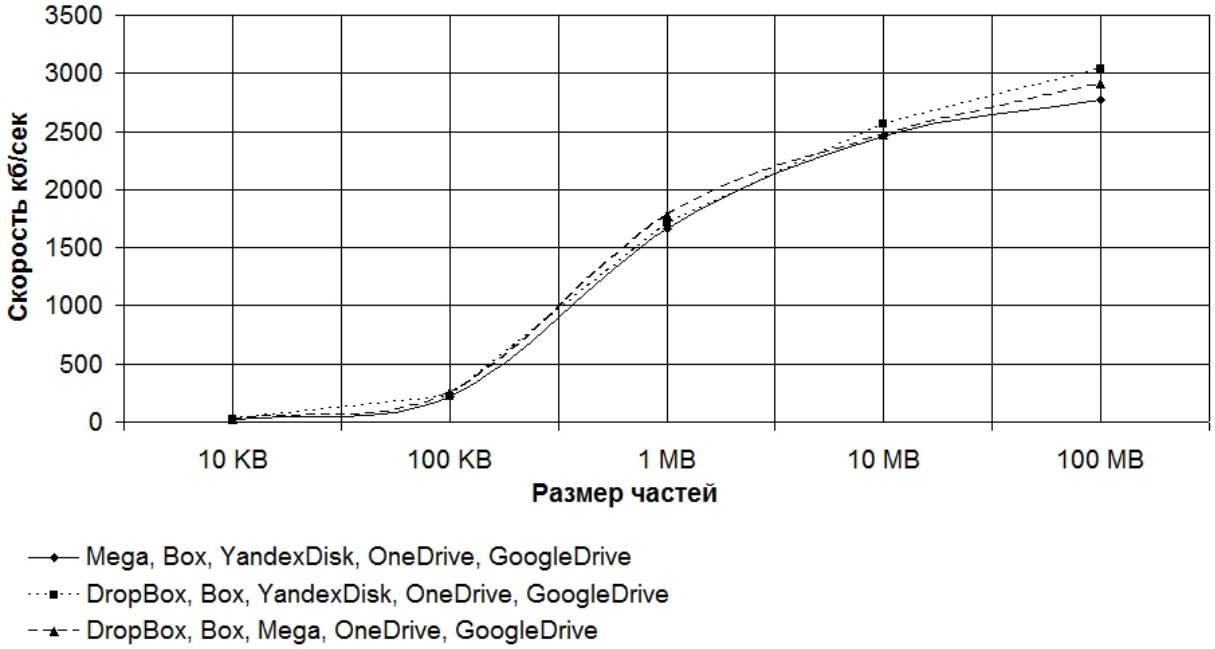

Рисунок 2.5 — Скорость сохранения данных при доступности четырех облачных серверов

восстановить хранимую информацию при доступности трех или мене облачных серверов. Проведем моделирование ситуации, когда доступно три облачных сервиса, рисунок 2.6. Как видно из рисунков скорость чтения информации из хранилищ увеличилась на 40%, но если учесть небольшую избыточность информации, то мы будем иметь 23% прирост скорости чтения информации относительно стандартного чтения информации из облачных серверов.

Таблица 7 — Средняя скорость чтения файлов в облаках без СРД (килобайт в секунду)

Облачный сервис	10 KB	100 KB	1 MB	10 MB	100 MB
Mega	1.48	39.83	265.27	287.25	747.35
DropBox	7.46	31.54	465.44	504.92	1226.40
YandexDisk	11.36	123.44	1163.63	2523.94	2687.58
OneDrive	1.69	41.49	433.86	1227.73	1210.19
Box	2.74	26.59	162.78	258.26	1014.50
GoogleDrive	3.08	52.07	511.97	2031.61	2192.93

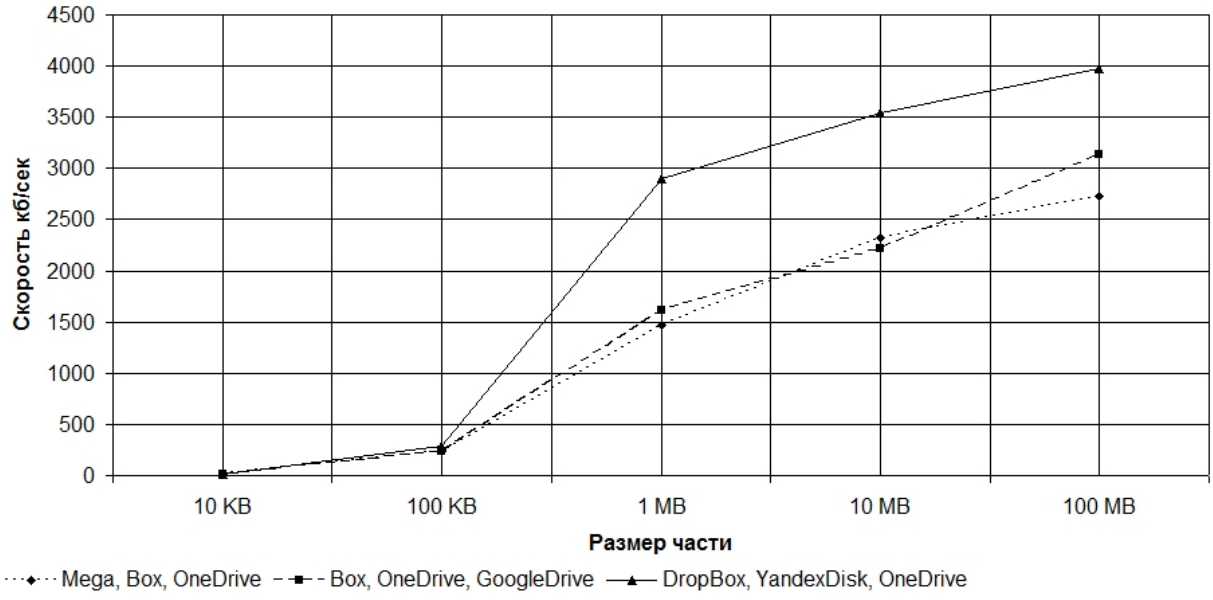

Рисунок 2.6 — Скорость сохранения данных при доступности трех облачных серверов

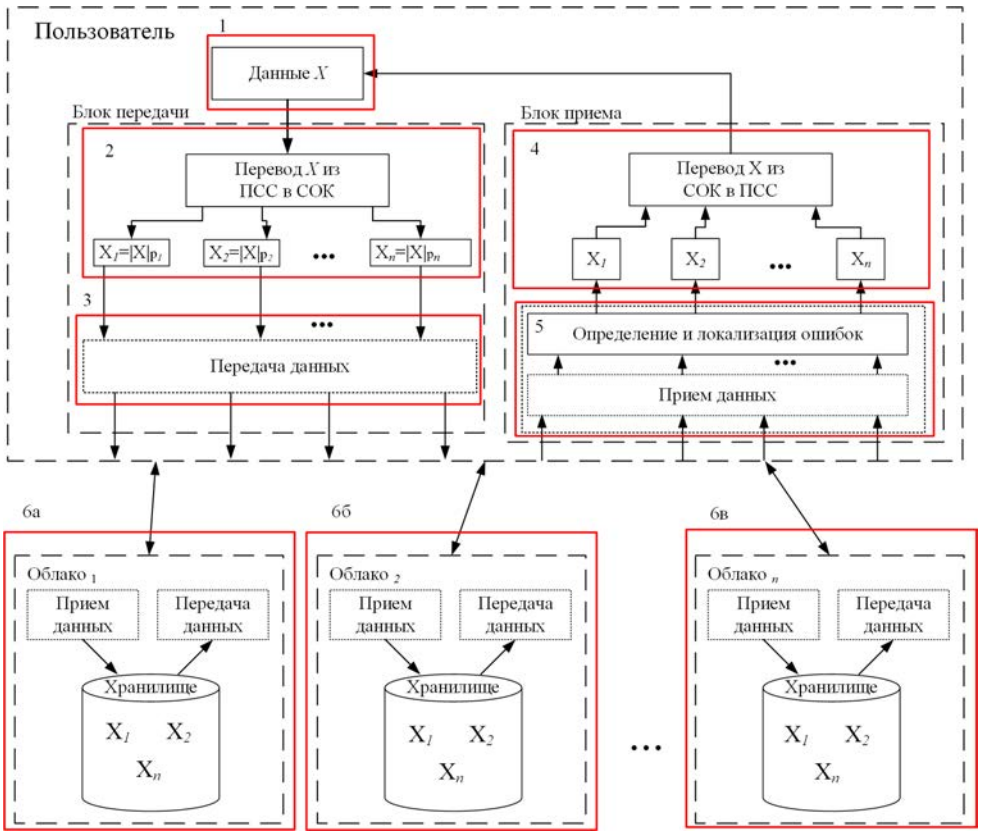

Рисунок 2.7 — Структура облачного хранилища основанного на одноуровневой системе остаточных классов

2.5 Разработка модели хранения и обработки больших данных в облаках

2.5.1 Модификация модели обработки и хранения данных в облачной среде на основе одноуровневой системы остаточных классов

На рисунке 2.7 представлена обобщенная схема обработки и хранения данных в облачной среде на основе одноуровневой системы остаточных классов.

Рассмотрим подробнее каждый из блоков приведенной схемы рисунок 2.7.

Данные 1 необходимо передать для хранения и обработки в облака 6а-6г. Для этого данные 1 конвертируются из СОК в ПСС при помощи алгоритмов перевода. После преобразования и разделения данных они через блок приема/-передачи данных 3 отправляются для хранения и обработки в облака 6а-6г. При необходимости получения данных пользователем направляется запрос в облака 6а-6г и через блок приема/передачи данных 3 данные приходят на блок определения и локализации ошибок 5 и производится определение и локализация

ошибок. После блока приема/передачи 3 данные передаются на блок 4, где производится восстановление данных и затем пользователь получает данные.

Основным недостатком схемы приведенной на рисунке 2.7 является высокая избыточность данных. Для решения проблемы высокой избыточности данных применяется избыточная СОК совместно со схемами распределения информации (СРД) [6, 11, 13]. Такой способ организации информационных систем базируется на дублировании или разделении информационных процессов между различными исполняющими узлами (облаками). Преимуществами распределенных систем является эффективность обработки за счет использования большого количества взаимодополняющих вычислительных узлов и разгрузки центрального узла и повышенная доступность, что достигается посредствам дублирования вычислительных узлов и узлов хранения данных [21]. Основной задачей СРД является разделение файлов на n частей таким образом, чтобы его можно было восстановить используя любые k из них $(m < k)$, при этом размер получаемых файлов должен быть в k раз меньше размера исходного. СРД базируются на идеи кодов стирания (EC), которые позволяют исправить ошибку в случае, если заранее известно в каком именно кодовом символе произошла ошибка [58]. EC коды хорошо сочетаются с распределенными системами хранения в которых можно отследить факт выхода из строя носителя или узла [70]. На рисунке 2.8 приведена схема облачного хранилища основанного на одноуровневой системе остаточных классов и схемах распределения данных.

Рассмотрим подробнее каждый из блоков приведенной схемы рисунок 2.8

Данные 1 необходимо передать для хранения и обработки в облака 6а-6г. Для этого данные 1 конвертируются из СОК в ПСС при помощи алгоритмов перевода и распределения данных СРД. После преобразования и разделения данных они через блок приема/передачи данных 3 отправляются для хранения и обработки в облака 6а-6г. При необходимости получения данных пользователем направляется запрос в облака 6а-6г и через блок приема/передачи данных 3 данные приходят на блок определения и локализации ошибок 5, производится определение и локализация ошибок и определяется возможность восстановления исходных данных из полученных проекций данных. После блока приема/передачи 3 данные передаются на блок 4, где производится восстановление данных и затем пользователь получает данные.

Построение систем хранения и обработки данных может производится при помощи различного числа модулей. В системе k рабочих оснований и r кон-

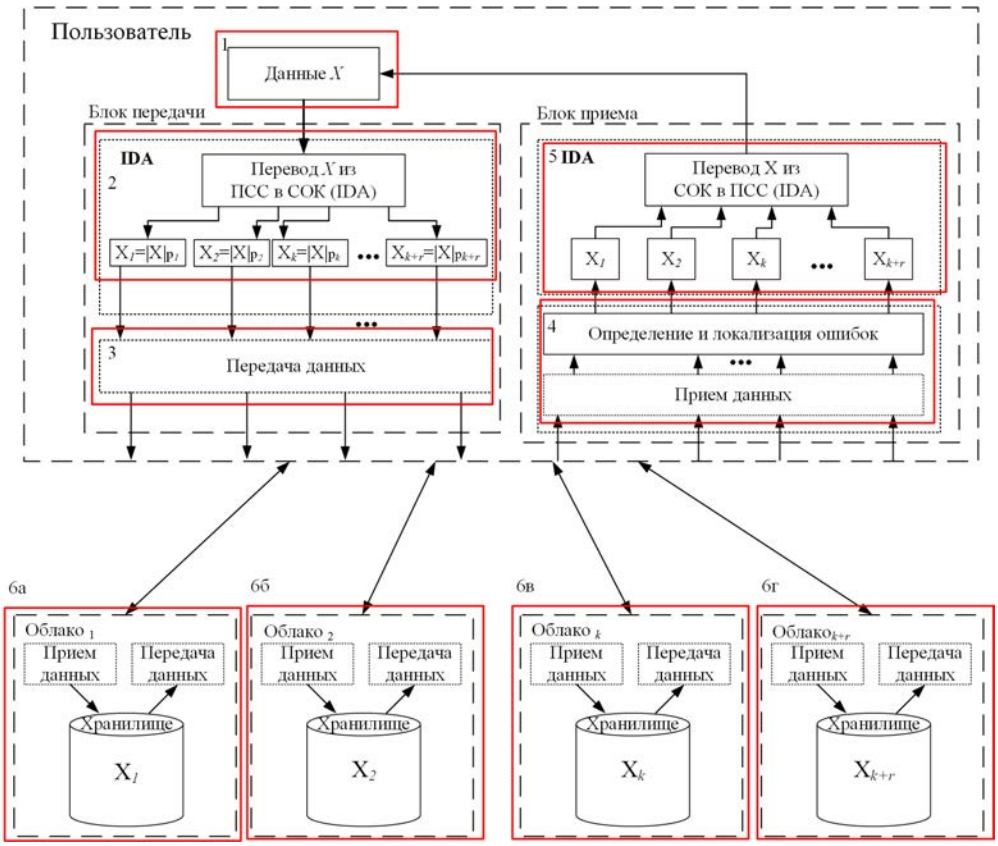

Рисунок 2.8 — Структура облачного хранилища основанного на одноуровневой системе остаточных классов и схемах распределения данных

трольных оснований, причем $n = k + r$. Построение системы обработки данных может производится как с полным копированием рабочего диапазона $(k = r)$, так и с частичным копированием $(k > r)$. Произведем расчеты полной надежности схемы обработки и хранения информации с использованием СОК и СРД для распределенного хранения и обработки данных, а так же произведем расчет доступного пространства при различных конфигурациях системы.

Зафиксируем колличество доступных облачных хранилищ и произведем расчет надежности системы со следующими параметрами построения:

1. Параметры СОК будем брать равными: $k = 2, 3, 4, 5, 6, 7, 8$, где r изменяется от 1 до 3.
2. Вероятность полного отказа системы обработки и хранения данных будем вычислять по формуле (2.32).

$$P_1 = \sum_{i=n-k+1}^{n} C_n^i P_d^i q^{n-i}, \tag{2.32}$$

где $q = 1 - P_d$, P_d – вероятность отказа одного облачного хранилища.

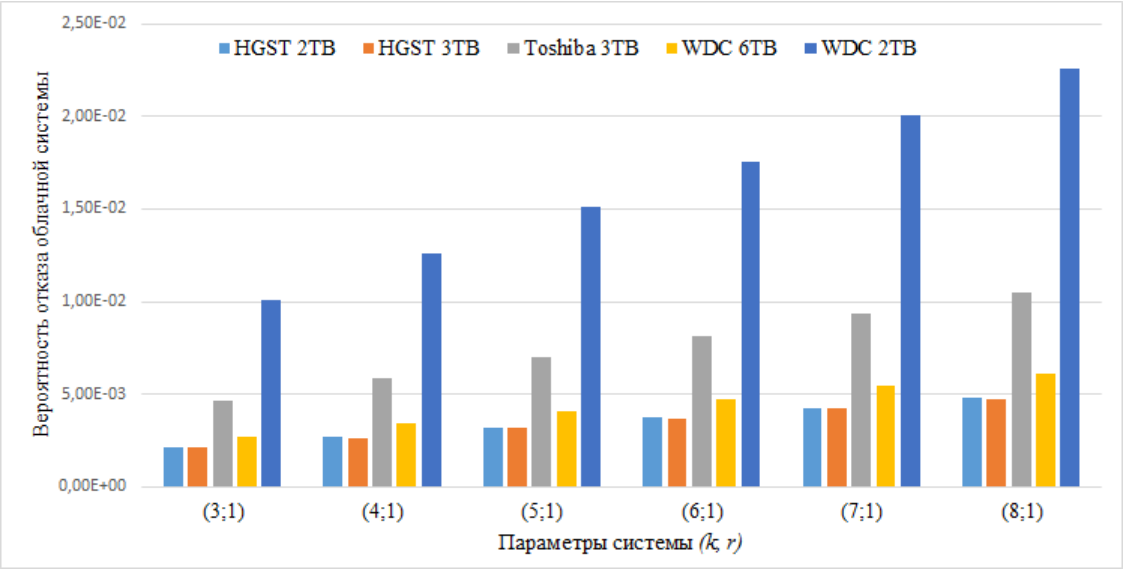

Рисунок 2.9 — Вероятности отказов работы системы облачных серверов на основе дисков марки WDC

3. Доступное дисковое пространство $V_{Д}$ будем вычислять по формуле (2.33).

$$V_{Д} = \frac{V_{ЖД} \times k}{n}, \tag{2.33}$$

где $V_{ЖД}$ – общий объем жесткого диска.

Результаты расчетов приведены на рисунке 2.9.

Построение одноуровневых моделей обработки данных на облачных серверах с использованием системы остаточных классов позволяет повысить надежность хранимой и обрабатываемых данных. Использование моделей обработки данных совместно с одномерными схемами распределенного хранения данных, основанных на системе остаточных классах позволяет строить надежные модели хранения. В случае использования рабочей модели с $k = 2$ рабочими основаниями, и $r = 7$ контрольными основаниями (при условии, что взяты модули равной битности), избыточность будет составлять $\approx 350\%$ надежность данной схемы будет составлять $P_O \approx 0.6038 \cdot 10^{-25}$. При использовании рабочей модели с $k = 8$ рабочими основаниями, и $r = 1$ контрольным основанием (схема $(8, 1)$), избыточность будет составлять $\approx 12\%$ надежность данной схемы будет составлять $P_O \approx 0.1011 \cdot 10^{-4}$. В модели обработки и хранения данных используемой компанией «Google Inc.» [56] избыточность составляет $\approx 300\%$, при надежности схемы равной $P_O \approx 0.1531 \cdot 10^{-11}$. Если использовать модель обработки данных на основе системы остаточных классов с $k = 4$ рабочими основаниями, и $r = 4$ контрольными основаниями, избыточность данной модели будет состав-

Таблица 8 — Относительное сравнение систем обработки данных компании «Google Inc.» [56], «ClaverSalfe» и на одномерной СОК

Модель	Избыточность	Вероятность отказа
	WDC, 6 TB	
«Google Inc.» [56]	$\approx 300\%$	$0.3144 \cdot 10^{-11}$
«ClaverSalfe»	$\approx 33\%$	0.0027
Одномерная СОК	$\approx 180\%$	$0.1828 \cdot 10^{-13}$
	WDC, 2 TB	
«Google Inc.» [56]	$\approx 300\%$	$0.1619 \cdot 10^{-9}$
«ClaverSalfe»	$\approx 33\%$	0.0100
Одномерная СОК	$\approx 180\%$	$0.1295 \cdot 10^{-10}$
	HGST, 3 TB	
«Google Inc.» [56]	$\approx 300\%$	$0.1492 \cdot 10^{-11}$
«ClaverSalfe»	$\approx 33\%$	0.0021
Одномерная СОК	$\approx 180\%$	$0.5285 \cdot 10^{-14}$
	HGST, 2 TB	
«Google Inc.» [56]	$\approx 300\%$	$0.1531 \cdot 10^{-11}$
«ClaverSalfe»	$\approx 33\%$	0.0021
Одномерная СОК	$\approx 180\%$	$0.5513 \cdot 10^{-14}$
	Toshiba, 3 TB	
«Google Inc.» [56]	$\approx 300\%$	$0.1609 \cdot 10^{-10}$
«ClaverSalfe»	$\approx 33\%$	0.0046
Одномерная СОК	$\approx 180\%$	$0.2775 \cdot 10^{-12}$

лять $\approx 100\%$ надежность данной схемы будет составлять $P_O \approx 0.1235 \cdot 10^{-12}$. Использование одномерных схем разделения данных совместно с системой остаточных классов позволяет уменьшить избыточность данных в 3 раза и повысить надежность в 1.9 раз относительно подхода представленного в работе [56]. Использование построенной модели для обработки и хранения больших данных позволит повысить отказоустойчивость и надежность данных, без использования дополнительного дискового пространства и вычислительных мощностей облачной системы [6].

2.5.2 Разработка модели обработки и хранения данных в облачной среде на основе двухуровневой системы остаточных классов

Мультиоблако (multi-cloud) системы представляют собой возможную альтернативу использования единого облака. В данном случае вместо одного облачного провайдера используется сразу несколько, что позволяет минимизировать риски потери, компрометации и раскрытия данных, а так же повысить надежность хранения данных. Однако в данном случае особого подхода требует обработка данных. В работе [41] предложено использование избыточной системы остаточных классов для распределения вычислений между различными облачными провайдерами. Использование СОК имеет ряд преимуществ. Во-первых, отдельный облачный провайдер не получает достаточной информации для восстановления исходных данных. Во-вторых, избыточное представление в СОК позволяет применять методы контроля и локализации ошибок в коде для повышения надежности системы [5, 10, 73].

Применение СОК базируется на распределении данных между различными вычислителями на основе остаточного представления. Все вычисления, проводимые облачными серверами, производятся независимо. Тем самым достигается максимальная эффективность и надежность операций. Повышение эффективности и надежности операций накладываемых на мультиоблачную систему для хранения и обработки данных приводит к модели двухуровневой системы остаточных классов. Каждый уровень такой системы представляет собой обособленную полноценную систему остаточных классов. Остатки первого уровня служат числами для второго, соответственно диапазон и модули первого превышают диапазон и модули второго. Каждый уровень при этом может решать свою конкретную задачу в зависимости от того, какая цель стоит перед всей системой в целом.

Использование двухуровневой СОК при проектировании мультиоблачной системы позволяет разделить задачи каждого из уровней. Основные вычисления производятся на втором уровне, следовательно, необходимо подбирать для него систему оснований так, чтобы арифметические операции на нем были максимально эффективны. Этого можно добиться, используя специальные наборы оснований [77]. Однако выбор оснований для каждого уровня является сложной задачей. Основная причина – это множество условий, накладываемых од-

новременно на оба уровня СОК. Кроме того, необходимо учитывать, что взаимодействие двух отдельных СОК такого рода может повлечь дополнительные накладные расходы при вычислениях.

Пусть на первом уровне используются модули $\{p_1, p_2, \ldots, p_n\}$ с диапазоном $P = p_1 \cdot p_2 \cdot \ldots \cdot p_n$, что представляет собой общий диапазон вычислений. Модуль с номером i ассоциирован с отдельным облачным провайдером для всех $i = 1, 2, \ldots, n$. Модули, соответствующие второму уровню для данного провайдера будем обозначать как $\{p_{i,1}, p_{i,2}, \ldots, p_{i,n_i}\}$, а их диапазон обозначим как $P_i = p_{i,1} \cdot p_{i,2} \cdot \ldots \cdot p_{i,n_i}$.

Как было показано в 2.5.1, избыточная СОК позволяет строить надежные схемы хранения и обработки данных в облачной среде. Рассмотрим построение двухуровневых схем хранения и обработки информации с использованием избыточной СОК. На рисунке 2.10 представлена схема обработки и хранения данных в облачной среде на основе двухуровневой системы остаточных классов [6].

Схема обработки и хранения данных в облачной среде представленная на рисунке 2.10 состоит из облачной (рисунок 2.12) и пользовательской (рисунок 2.11) частей, рассмотрим работу схемы.

Работа схемы представленной на рисунке 2.11: данные X необходимо передать для хранения и обработки в облако. Для этого данные X переводятся из ПСС в СОК $X = \{X_1, X_2, \ldots, X_k, X_{k+1}, \ldots, X_{k+r}\}$ с набором модулей $\{p_1, p_2, \ldots, p_k, \ldots, p_{k+r}\}$, и при помощи алгоритмов распределения информации (СРД) производится передача в мультиоблако i. После получения i-м мультиоблаком части данных X_i в блоке передачи производится перевод и разделение данных X_i, $X_i = \{X_{i,1}, X_{i,2}, \ldots, X_{i,k}, X_{i,k+1}, \ldots, X_{i,k+r}\}$ с набором модулей $\{p_{i,1}, p_{i,2}, \ldots, p_{i,k}, \ldots, p_{i,k+r}\}$, при помощи СРД, после этого производится передача данных X_i,j для хранения в i, j-облако. После запроса пользователя на получение данных X_i,j, i, j-облако передает $X_{i,j}$ на блок приема i-мультиоблака. В блоке приема производится определение и локализация ошибок и при потери одной или нескольких частей $X_{i,j}$ определяется возможность восстановления данных X_i. После восстановления данных, X_i отправляются пользователю. Блок приема после получения частей данных X_i производит определение и локализацию ошибок и при потери одной или нескольких частей X_i определяется возможность и восстановление данных X, и пользователь получает данные X.

Данная двухуровневая схема хранения и обработки данных имеет высокую надежность.

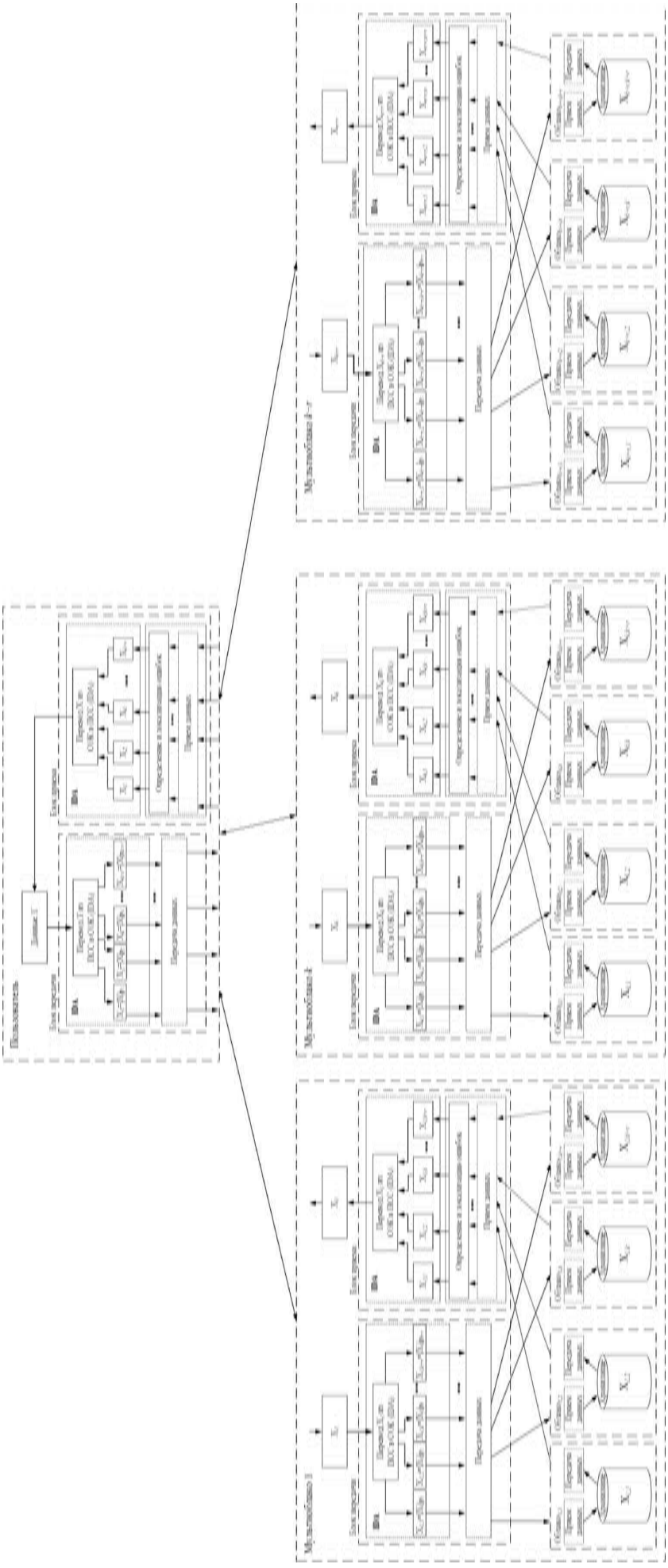

Рисунок 2.10 — Схема мульти облачного хранилища основанного на двухуровневой системе остаточных классов

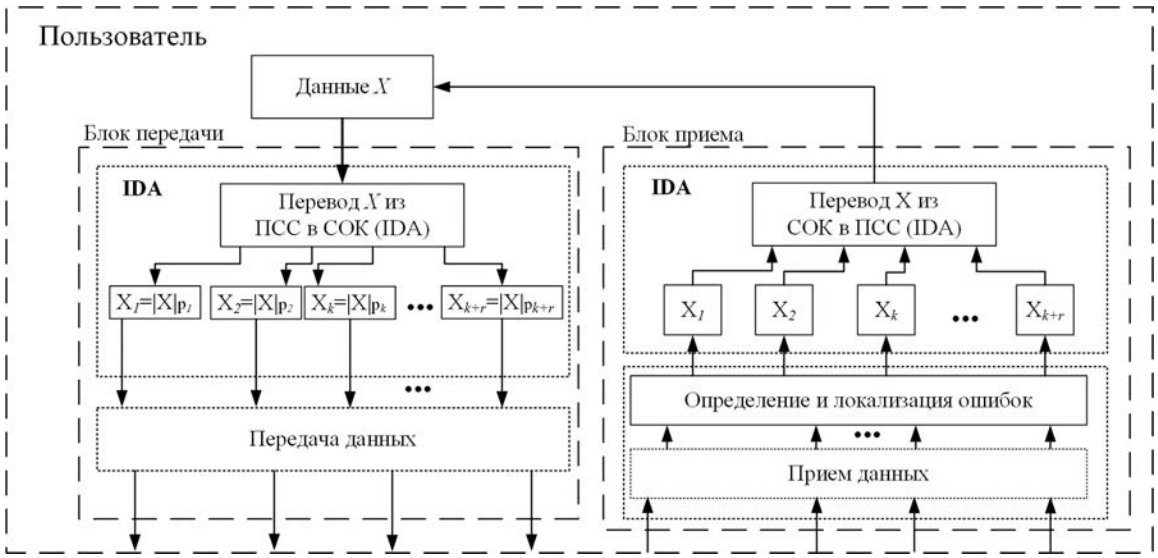

Рисунок 2.11 — Схема пользовательской части модели облачного хранилища

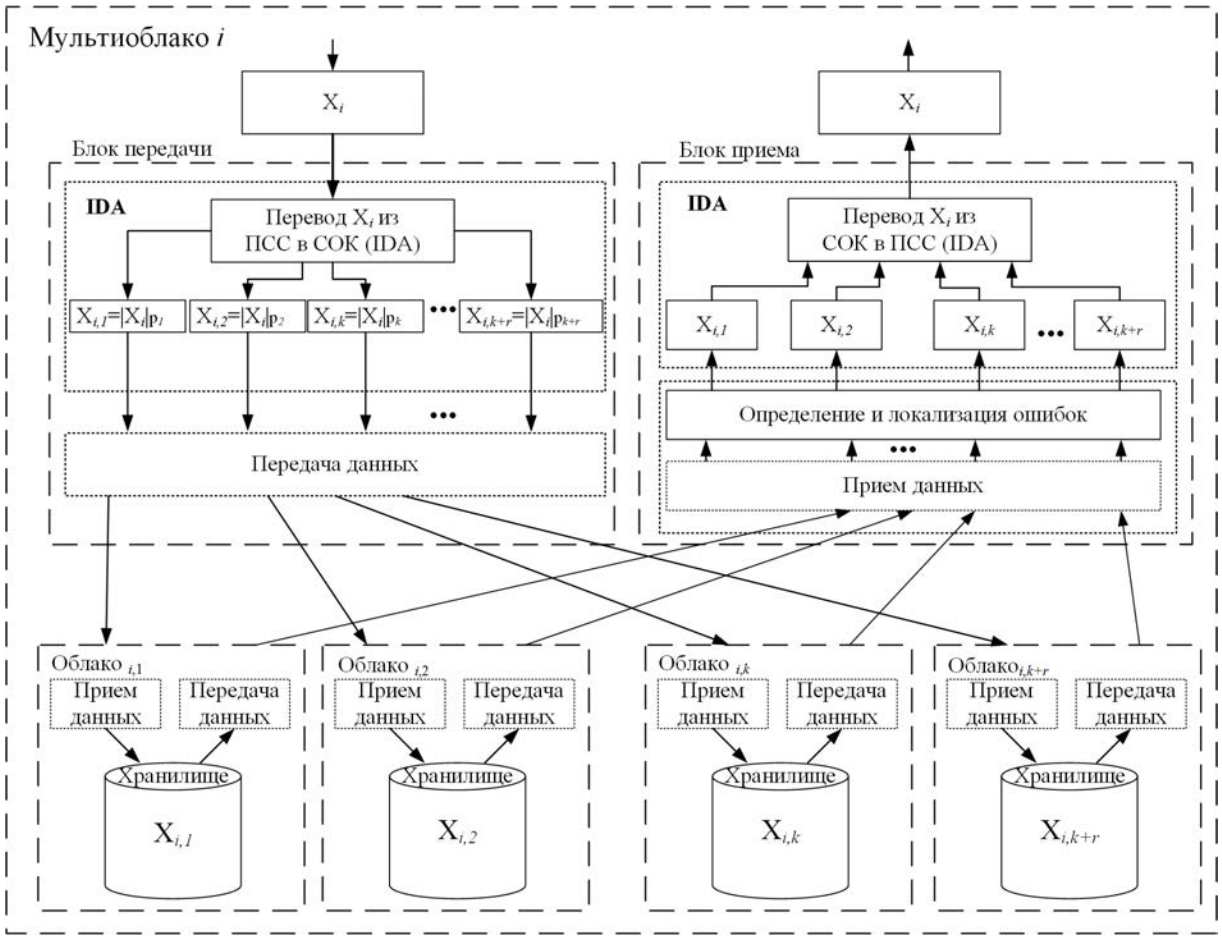

Рисунок 2.12 — Схема облачной части модели мульти облачного хранилища

Рассмотрим пример работы двухуровневой схемы представленной на рисунке 2.10.

Пусть заданы модули СОК для первого уровня $p_1 = 17$, $p_2 = 19$, $p_3 = 23$, $p_4 = 25$, диапазон первого уровня СОК $P = 17 \cdot 19 \cdot 23 \cdot 25 = 185\,725$.

Модули для второго уровня $\pi_1 = 3$, $\pi_2 = 5$, $\pi_3 = 7$, диапазон второго уровня СОК $\Pi = 3 \cdot 5 \cdot 7 = 105$.

Вычислим необходимые константы для первого уровня:

Найдем P_i: $P_1 = \frac{P}{p_1} = 10\ 925$, $P_2 = \frac{P}{p_2} = 9\ 775$, $P_3 = \frac{P}{p_3} = 8\ 075$, $P_4 = \frac{P}{p_4} = 7\ 429$.

Найдем значения $SQ_1 = 10\ 925 + 9\ 775 + 8\ 075 + 7\ 429 = 36\ 204$, $N_{1,1} = \lceil \log_2 80 \rceil = 7$

Вычислим параметры приближенного метода:

$$k_{1,1} = \left|P_1^{-1}\right|_{p_1} P_1 = 152950,\ k_{1,2} = \left|P_2^{-1}\right|_{p_2} P_2 = 166175$$
$$k_{1,3} = \left|P_3^{-1}\right|_{p_3} P_3 = 96900,\ k_{1,4} = \left|P_4^{-1}\right|_{p_4} P_4 = 141151$$
$$\overline{\overline{k_{1,1}}} = \left\lceil \frac{\left|P_1^{-1}\right|_{p_1} 2^{N_{1,1}}}{p_1} \right\rceil = 106,\ \overline{\overline{k_{1,2}}} = \left\lceil \frac{\left|P_2^{-1}\right|_{p_2} 2^{N_{1,1}}}{p_2} \right\rceil = 115$$
$$\overline{\overline{k_{1,3}}} = \left\lceil \frac{\left|P_3^{-1}\right|_{p_3} 2^{N_{1,1}}}{p_3} \right\rceil = 67,\ \overline{\overline{k_{1,4}}} = \left\lceil \frac{\left|P_4^{-1}\right|_{p_4} 2^{N_{1,1}}}{p_4} \right\rceil = 98$$

Вычислим необходимые константы для второго уровня:

Найдем Π_i: $\Pi_1 = \frac{\Pi}{\pi_1} = 35$, $\Pi_2 = \frac{\Pi}{\pi_2} = 21$, $\Pi_3 = \frac{\Pi}{\pi_3} = 15$.

Найдем значения $SQ_2 = 35 + 21 + 15 = 71$, $N_{1,2} = \lceil \log_2 12 \rceil = 4$

Вычислим параметры приближенного метода:

$$k_{2,1} = \left|\Pi_1^{-1}\right|_{\pi_1} \cdot \Pi_1 = 70,\ k_{2,2} = \left|\Pi_2^{-1}\right|_{\pi_2} \cdot \Pi_2 = 21$$
$$k_{2,3} = \left|\Pi_3^{-1}\right|_{\pi_3} \cdot \Pi_3 = 15$$
$$\overline{\overline{k_{2,1}}} = \left\lceil \frac{\left|\Pi_1^{-1}\right|_{\pi_1} 2^{N_{1,2}}}{\pi_1} \right\rceil = 11,\ \overline{\overline{k_{2,2}}} = \left\lceil \frac{\left|\Pi_2^{-1}\right|_{\pi_2} 2^{N_{1,2}}}{\pi_2} \right\rceil = 4,$$
$$\overline{\overline{k_{2,3}}} = \left\lceil \frac{\left|\Pi_3^{-1}\right|_{\pi_3} 2^{N_{1,2}}}{\pi_3} \right\rceil = 3.$$

Число $X = 185724$ нам нужно передать для хранения в облако. Для этого вначале производится перевод из ПСС в СОК на первом уровне $X \xrightarrow{COK} \{16, 18, 22, 24\}$. После этого каждую проекцию данных x_i передают в облако:

– облако 1: $x_1 = 16 \xrightarrow{COK} \{1, 1, 2\}$
– облако 2: $x_2 = 18 \xrightarrow{COK} \{0, 3, 4\}$

– облако 3: $x_3 = 22 \xrightarrow{COK} \{1, 2, 1\}$
– облако 4: $x_4 = 24 \xrightarrow{COK} \{0, 4, 3\}$

После получения запроса на выдачу данных производится их восстановление, для этого в каждом мультиоблаке производится восстановление x_i проекции данных.

Облако 1:

$$\sum_{i=1}^{n} x_{1,i}k_{2,i} = 1 \cdot 70 + 1 \cdot 21 + 2 \cdot 15 = 121.$$
$$\sum_{i=1}^{n} x_{1,i}\overline{\overline{k_{2,i}}} = 1 \cdot 11 + 1 \cdot 4 + 1 \cdot 3 = 18.$$
$$r = \left\lfloor \tfrac{18}{2^4} \right\rfloor = 1,$$
$$x_1 = 121 - 1 \cdot 105 = 16$$

Облако 2:

$$\sum_{i=1}^{n} x_{2,i}k_{2,i} = 0 \cdot 70 + 3 \cdot 21 + 4 \cdot 15 = 123.$$
$$\sum_{i=1}^{n} x_{2,i}\overline{\overline{k_{2,i}}} = 0 \cdot 11 + 3 \cdot 4 + 4 \cdot 3 = 24.$$
$$r = \left\lfloor \tfrac{24}{2^4} \right\rfloor = 1,$$
$$x_2 = 123 - 1 \cdot 105 = 18$$

Облако 3:

$$\sum_{i=1}^{n} x_{3,i}k_{2,i} = 1 \cdot 70 + 2 \cdot 21 + 1 \cdot 15 = 127.$$
$$\sum_{i=1}^{n} x_{3,i}\overline{\overline{k_{2,i}}} = 1 \cdot 11 + 2 \cdot 4 + 1 \cdot 3 = 22.$$
$$r = \left\lfloor \tfrac{22}{2^4} \right\rfloor = 1,$$
$$x_3 = 127 - 1 \cdot 105 = 22$$

Облако 4:

$$\sum_{i=1}^{n} x_{4,i}k_{2,i} = 0 \cdot 70 + 4 \cdot 21 + 3 \cdot 15 = 129.$$
$$\sum_{i=1}^{n} x_{4,i}\overline{\overline{k_{2,i}}} = 0 \cdot 11 + 4 \cdot 4 + 3 \cdot 3 = 25.$$
$$r = \left\lfloor \tfrac{25}{2^4} \right\rfloor = 1,$$
$$x_4 = 129 - 1 \cdot 105 = 24$$

После восстановления проекций данных облака отправляют данные пользователю, где производится восстановлене данных:

$$\sum_{i=1}^{n} \overline{k_i} x_i = 16 \cdot 106 + 18 \cdot 115 + 22 \cdot 67 + 24 \cdot 98 = 7592$$

$$r = \left\lfloor \frac{7592}{2^7} \right\rfloor = 59,$$

$X = 10957774 - 59 \cdot 185725 = -1$, так как $X < 0$ то $X = -1 + 185725 = 185724$

Затем пользователь получает данные и производит их обработку.

Проведем расчет надежности мультиоблачной модели хранения и обработки данных представленной на рисунке 2.10, расчет будем производить по формуле (2.35), результаты расчетов приведены в приложении Ж, таблицы 23, 24, 25, 26, 22.

$$P_2 = \sum_{i=n_2-k_2}^{n_2} C_{n_2}^i P_0^i q_4^{n_2-i} \tag{2.34}$$

вероятность безотказной работы мультиоблачной модели хранения и обработки данных будет равняться

$$P_{2,1} = P_1 \cdot P_2 = \sum_{i=n_2-k_2}^{n_2} C_{n_2}^i P_0^i q^{n_2-i} \cdot \sum_{i=n-k+1}^{n} C_n^i P_d^i q^{n-i}, \tag{2.35}$$

где $q = 1 - P_d$

Построение многоуровневых моделей обработки данных на облачных серверах с использованием избыточной СОК позволяет повысить надежность хранимых и обрабатываемых данных. Использование многоуровневых моделей обработки данных на основанных избыточной СОК и СРД, позволяет строить модели надежнее одноуровневых моделей и модели компании «Google Inc.» [56]. При построении модели обработки данных в используется избыточная СОК, минимальная вероятность отказа оборудования может составлять $8.7 \cdot 10^{-98}$ при использовании схем $(2,9);(3,9)$ в этом случае избыточность будет составлять $\approx 1250\%$, эта схема мало пригодна для решения поставленной задачи, так как имеет очень большую избыточность хранимых и обрабатываемых данных в облачной среде. Применение схемы обработки и хранения, с параметрами схемы, $(6,8);(6,9))$, в этом случае вероятность отказа составит $7.534 \cdot 10^{-12}$ при избыточности $\approx 100\%$.

2.6 Выводы по второй главе

1. Предложена архитектура мультиоблачной системы хранения и обработки больших данных основанная на принципах модулярной арифметики. Предложенная модификация облачной системы хранения и обработки данных позволяет повысить надежность и отказоустойчивость хранимых и обрабатываемых данных.
2. Разработаны одноуровневые и двухуровневые модели надежного хранения больших данных и по результатам моделирования проведенный сравнительный анализ показал преимущество моделей построенных на базе модулярной арифметики над моделями основанных на ПСС.
3. Модифицирована модель проверки корректности проверки результата вычислений при использовании системы остаточных классов. Предложен новый способ скрытия модулей системы остаточных классов при передачи данных облачным серверам для хранения или обработки.

Глава 3. РАЗРАБОТКА МАТЕМАТИЧЕСКОЙ МОДЕЛИ НАДЕЖНОЙ СИСТЕМЫ ХРАНЕНИЯ И ОБРАБОТКИ БОЛЬШИХ ДАННЫХ В ОБЛАКАХ

3.1 Повышение отказоустойчивости облачного сервера функционирующего в системе остаточных классов путем перераспределения обрабатываемых данных

Рассматриваемая облачная система приведенная на рисунках 2.8, 2.10 основанная на принципах СОК имеет p_i облачных серверов, которые предназначены для хранения и обработки больших данных по k рабочим и r контрольными основаниям (частям). Один из подходов к решению проблемы повышения надежности облачной системы основан на перераспределении частей данных при отказе части рабочих или контрольных каналов [3, 8]. При этом под надежностью будем понимать способность облачной системы сохранять работоспособность при отказе одного или нескольких серверов обработки и хранения данных, при снижении в допустимых пределах некоторых показателей качества функционирования. Данная особенность построения модели хранения и обработки больших данных, позволяет строить облачную систему с постоянными отказами рабочих или контрольных серверов функционирующих в СОК. При постоянных отказах рабочих и контрольных оснований облачная система переходит в состояние S_V, что влечет за собой необходимость перераспределения данных между серверами и нахождения множества $N = \{Q_V\}$ для распределения больших данных. Если для каждого состояния облачной системы выбирается решение о перераспределении данных Q_V между облачными серверами, проблема заключается в нахождении оптимального варианта перераспределения данных между облачными серверами для каждого сервера $S_V \in S$. Она может быть решена как проблема оптимизации по одному из показателей, принимаемому в качестве целевой функции при заданных ограничениях на остальные показатели. Рассмотрим постановку по распределению данных между облачными серверами функционирующие в СОК.

Допустим, что в процессе работы облачной системы существуют отказы облачных серверов, и сервера являются независимыми. Пусть $S(t)$ – состояние

облачной системы в момент времени t:

$$S(t) = d_1, d_2, \ldots, d_n, \tag{3.1}$$

где n – количество каналов облачной системы.

$$d_i = \begin{cases} 0, & \text{если } i\text{-й сервер в рабочем состоянии} \\ 1, & \text{если } i\text{-й сервер в нерабочем состоянии} \end{cases}$$

Рассмотрим интервал времени $[t_0, t_z]$. Пусть $S(t_0) = 0, 0, \ldots, 0$ – начальное состояние облачной системы, начальное распределение системы равно

$$A_{0_i} = \{|\alpha_1|_{p_i}, |\alpha_2|_{p_i}, \ldots, |\alpha_L V|_{p_n}\},$$

где $|\alpha_l|_{p_i}$ – данные, соответствующие i-му облачному серверу; $i = \overline{1, n}$; $l = \overline{1, L}$; L – число задач. Множество $P = \{p_1, p_2, \ldots, p_n\}$ является множеством всех рабочих и контрольных серверов облачной системы.

В процессе эксплуатации облачной системы в момент времени t_k $(t_0 < t_k \leqslant t_z)$ часть серверов может отказать. Тогда множество P можно разбить на два подмножества:

– $P^O_{t_k}$ – подмножество всех отказавших серверов облачной системы

– $P^P_{t_k}$ – подмножество всех отказавших серверов облачной системы

Пусть S_V – состояние в момент t_k; тогда

$$A_V = \{|\alpha_1|^+_{p_i}, |\alpha_2|^+_{p_i}, \ldots, |\alpha_L V|^+_{p_i}\},$$

где $i = \overline{1, n_p}$ – множество всех данных обеспечивающие допустимые пределы показателей качества хранимых данных, которые могут храниться в облачной системе в состоянии S_V, иными словами для обработки которых в облачной системе есть необходимое количество серверов, $n_p = n - n_O$ – количество работоспособных облачных серверов в состоянии S_V, n_O – количество отказавших облачных серверов

$$A_i = \{\big||\alpha_j|^+_{p_1}\big|^+_{p_i}, \big||\alpha_j|^+_{p_2}\big|^+_{p_i}, \ldots, \big||\alpha_j|^+_{p_n}\big|^+_{p_i}\},$$

где $j = \overline{1, L_V}$, $\big||\alpha_j|^+_{p_1}\big|^+_{p_i} \in A_V$ – подмножество данных которые может хранить и обрабатывать работоспособный облачный сервер, если облачная система находится в состоянии S_V.

В облачной системе с постоянными отказами, работоспособные облачные серверы используются для хранения и обработки данных. При отказе серверов

производится реконфигурация облачной системы, с целю исключения всех отказавших облачных серверов. При этом производится новое перераспределение между рабочими облачными серверами данных, которые не были отправлены выбывшим серверам.

При переходе облачной системы из состояния $S(t_0)$ в состояние $S(t_k) = S_V$ относительно конкретных отказавших и работоспособных облачных серверов могут быть приняты различные решения, которые определяются значимостью данных и работоспособными серверами.

Введем следующие обозначения: A_V^O – множество собственных данных отказавших облачных серверов для состояния S_V; A_V^P – множество всех собственных данных работоспособных облачных серверов для состояния S_V.

При наступлении события S_V производится перераспределение данных, а по отношению к данным множества A_O^P принимается одно из следующих решений:

- U_O^C – все данные множества A_O^P сохраняются в облачной системе если $n_O \leqslant l$, где $l = \frac{d_{min}}{2}$ – количество исправляемых ошибок (частей), d_{min} – минимальное кодовое расстояние избыточного кода СОК;
- U_O^O все данные множества A_O^P сохраняются, а часть отбрасывается если $1 < n_O < n_{O,\text{доп}}$, где $n_{O,\text{доп}}$ – допустимое число отказавших облачных серверов;
- U_O^O часть данных множества A_O^P сохраняются, а часть отбрасывается если $n_O = n_{O,\text{доп}}$.

Будем считать отказавшие облачные серверы фиктивными, а их собственные данные приравняем к нулю. Одновременно по отношению к данным множества A_P^V принимается одно из следующих решений:

- U_P^C – все данные множества A_P^V сохраняются и продолжается выполнение поставленной задачи;
- U_P^{C-I} – все данные множества A_P^V сохраняются в облачной системе, часть данных хранится и обрабатывается на назначенных серверах, а часть перераспределяется.

При выполнении неравества $n_O > n_{O,\text{доп}}$ происходит полный отказ облачной системы. Множество U возможных решений, которые могут быть приняты при перераспределении данных в состоянии S_V, определяется множеством всех возможных комбинаций рассмотренных выше решений для данных под-

Таблица 9 — Зависимость показателей облачной системы, функционирующей в СОК, от принятого решения при деградации рабочих и контрольных каналов

Решение	Отказы серверов		Показатели[1]	
	Рабочих	Контрольных	Надежность	Достоверность
$U_1 = U_O^C U_P^C$	Нет	Да	0	*
	Да	Нет	0	*
	Да	Да	0	*
$U_2 = U_O^C U_P^{C-I}$	Нет	Да	Отказ всей системы	
	Да	Нет		
	Да	Да		
$U_3 = U_O^O U_P^C$	Нет	Да	0	*
	Да	Нет	0	*
	Да	Да	*	*
$U_4 = U_O^O U_P^{C-I}$	Нет	Да	0	0
	Да	Нет	0	1
	Да	Да	0	0
$U_5 = U_O^{C-O} U_P^C$	Нет	Да	0	0
	Да	Нет	0	1
	Да	Да	0	0
$U_6 = U_O^{C-I} U_P^{C-I}$	Нет	Да	0	0
	Да	Нет	0	1
	Да	Да	0	*

1) 0 – уменьшение показателя, 1 – увеличение показателя, * – сохранение данного показателя

множеств A_P^V, A_O^V:

$$U = \left\{(U_O^C U_P^O), (U_C^O U_P^{C-I}), (U_O^O U_P^C), (U_O^O U_P^{C-I}), (U_O^{C-O} U_C^P), (U_O^{C-O} U_P^{C-I})\right\} \quad (3.2)$$

Рассмотрим отказы по рабочим и контрольным каналам, которые возможны при переходе облачной системы построенной на основе СОК, в состояние S_V и изменение показателей качества функционирования облачной системы при перераспределении в ней данных в соответствии с одним из решений U, таблица 9

В случае принятия решения $U_1 = U_O^C U_P^C$ все данные множества обрабатываются работоспособными каналами, а полученный результат выполнения

операции с некоторым искажением восстанавливается за счет корректирующих свойств кода, т.е. алгоритмическим путем.

Решение $U_2 = U_O^C U_P^{C-I}$ не имеет смысла (отказ всей системы) поскольку сохранение данных множества A_V^O не дает возможности перераспределять данные множества A_V^P.

Остальные решения используются для организации перераспределения данных в облачной системе.

Реализация любого из этих решений предполагает исключение из облачной системы отказавших серверов путем блокировки их входов и выходов и перераспределение данных между рабочими серверами. При этом в облачной системе предполагается наличие дополнительных програмных, апаратурных и временных ресурсов [8, 53].

Рассмотрим показатели функциональной мощности (E) и временной показатель отдельного сервера (T) как показатели качества работы облачной системы, к которым предъявляются определенные требования при осуществления перераспределения данных.

Номинальные $E^{\text{н}}$, $T^{\text{н}}$ и предельно допустимые $E^{\text{доп}}$, $T^{\text{доп}}$ значения принятых показателей качества функционирования определяют область работоспособности облачной системы как подмножество $M_p = \{y_\mu\}$ таких y_μ состояний для которых

$$E^{\text{н}} \geqslant E_\mu \geqslant E^{\text{доп}} \text{ и } T^{\text{н}} \geqslant T_\mu \geqslant T^{\text{доп}}. \tag{3.3}$$

Сформулируем задачу обеспечения работоспособности облачной системы при отказе части облачных серверов. Пусть при известной структуре облачной системы работающей на основе СОК с заданными облачными серверами выполняется алгоритм хранения и обработки данных, представленный в виде множества A_{0_i} и начального состояния для всех серверов системы S_0. Тогда при отказе некоторой части серверов задача заключается в нахождении стратегии перераспределения данных между рабочими серверами с обеспечением выполнения требований к показателям надежности облачной системы. Возможны два основных вида организации перераспределения данных: статистическое и динамическое перераспределение.

Статистический способ предполагает, что до начала работы облачной системы для некоторого заданного подмножества $S = S_V$ его состоянии в памяти находятся оптимальные планы распределения облачных серверов.

При переходе облачной системы в состояние $S_V \in S$ в ее облачных серверах начинается обработка даных соответствующих распределению G_V.

Если подмножество S не может быть задано либо его размерность требует недопустимо большого объема памяти для хранения всех допустимых программ, применяется динамическое перераспределение каналов. Время перестройки в этом случае может значительно превысить время перестройки при статическом способе.

Возможно сочетание статического и динамического способов перераспределения данных. Его суть состоит в том, что на первом этапе производится статическое перераспределение данных, для некоторого заранее сформулированного множества состояний системы $S_1(t)$, а на втором определяются наиболее вероятные переходы $S_1(t) \to S_2(t)$, формируется новый план перераспределения данных для всех возможных в процессе работы системы состояний. Учитываю требования к облачным системам по быстродействию, будем использовать статистическое перераспределение данных отказавших серверов облачной системы между рабочими серверами.

3.2 Функциональное представление параметров облачной системы функционирующей в системе остаточных классов

Качество работы облачной системы построенной на основе СОК характеризуется многими параметрами. Для реализации и решения задач оптимальной структуры облачной системы используются функциональные представления параметров в виде зависимости значений параметров от базовых величин, принимаемых в качестве аргументов [8]. Очевидно, что в зависимости от перераспределения данных между рабочими и информационными облачными серверами p_i изменяются как значения T (T – время выполнения алгоритма A), так и значения величин достоверности и точности вследствии изменения числа каналов $\{p_i \,|\, i \in k\}$ и $\{p_j \,|\, j \in r\}$.

Следующие показатели работы облачной системы являются важными в ее работе.

Функциональная мощность. Будем считать, что облачные системы предназначены для выполнения набора задач

$$\Pi = \{\pi_j\}, \quad i = \overline{1,n}.$$

Каждая задача задается совокупностью условий сложившейся ситуации по потребителям или источникам информации, выполнение которых определяется функциональными возможностями облачной системы. Поэтому одним из показателей оценивания облачной системы является функциональная мощность (E) определяемая количеством задач (запросов) которые может обработать облачная система в данный момент времени или в заданный интервал времени. Этот показатель может зависеть от числа решаемых задач, их сложности, важности, точности получаемых результатов и других характеристик [8].

Функциональная мощность облачной системы в состоянии S_V может быть определена на формуле [8]:

$$E_V = \sum_{|\alpha_j|^+_{p_i} \in A_V} \omega_i \tag{3.4}$$

где ω_i – вес разряда $|\alpha_j|^+_{p_i} \in A_V$ характеризующий его важность для решаемой задачи; A_V – множество разрядов, которые облачная система может обработать в состоянии S_V.

Реальное время или скорость выполнения задач облачной системой оценивается временными показателями T_i. К ним можно отнести показатели производительности, пропускной способности, среднего времени пребывания запроса на решение любой из допустимых задач, времени однократного выполнения заданного алгоритма функционирования и подобные им показатели.

Облачная система работающая в СОК обладает возможностью реконфигурации своей структуры при отказах отдельных рабочих или контрольных облачных серверов. Перераспределение данных решаемых задач между рабочими облачными серверами позволяет сохранить работоспособность всей облачной системы за счет снижения в допустимых пределах каких-либо показателей качества его функционирования, в том числе и T_i.

Под производительностью облачного сервера p_i будем понимать среднее время $(T_{iV}, i = T_i, \ldots, n)$ обработки данных одного разряда (части) в СОК и суммарное время T^{Σ}_{iV} обработки в облачном сервере p_i всех данных, назначенных ему в состоянии S_V. При этом T^{Σ}_{iV} включает время необходимое для коррекции результата.

Полагая известным среднее время T_i^O обработки данных облачным сервером p_i при исходном состоянии S_0 облачной системы, матрица $\|\Delta T_{i,j}\|$ (где $\Delta T_{i,j}$ – приращение среднего времени обработки данных в p_i при передачи ему данных $|\alpha_j|_{p_i}^+$, $j = 1, \ldots, L_i$, $i = 1, 2, \ldots, n$) и время, необходимое для коррекции результата, можно определить производительность.

Стоимость облачной системы определяется как [8]

$$C = C_1 + C_2 = nC_p + \sum_{i=1}^{n} \sum_{V_j \in \Omega} C_{|\alpha_j|_{p_i}^+}, \tag{3.5}$$

где C_p – составляющая стоимости одного облачного сервера, не зависящая от обрабатываемых им данных; $C_{|\alpha_j|_{p_i}^+}$ – составляющая стоимости одного облачного сервера зависящая от обрабатываемых им данных $|\alpha_j|_{p_i}^+$ эта составляющая связана с затратами на реализацию перераспределения данных. Функциональные представления для вероятности безотказной работы за некоторое время $R_{COK}(t)$ будем оценивать по полному отказу. Полным отказом облачной системы в момент времени t будем называть событие $F = \{E(t) < E^{\text{доп}}\}$, где $E^{\text{доп}}$ – минимально допустимое значение функциональной мощности.

Облачная система функционирующая в СОК состоит из k рабочих и r контрольных облачных серверов. При отказе рабочих серверов они могут быть заменены контрольными и наоборот. Предъявленное к облачной системе требование по обеспечению гарантированной защиты от выдачи недостоверного результата обуславливает необходимость сохранения работоспособности k_{min} рабочих и хотя бы одного контрольного облачных серверов. При этом допускается минимальная разрядность при которой с учетом важности решаемой задачи обеспечивается реализация всех заданных алгоритмов отвечающих требованиям по точности. При требуемой разрядности $\sum_{i=1}^{n} \omega_i$ величина ошибки результата реализации любого из алгоритмов δA не должна превосходить допустимую ошибку $\delta A_{\text{доп}}$:

$$\delta A_i \leqslant \delta A_{\text{доп}},\ k = \overline{1, b}, \tag{3.6}$$

где b – количество конечных результатов реализации алгоритмов.

3.3 Модификация численного метода Червякова для перевода чисел из системы остаточных классов в позиционную систему счисления за счет использования ранга числа Акушского

Современное состояние развития инфокоммуникационных технологий в области обработки и передачи данных характеризуется интенсивным внедрением новых принципов и подходов к обработке информации. Одним из путей по увеличению быстродействия вычислительных средств обусловило создание вычислительных систем с параллельной структурой. Вместе с тем возникла необходимость и целесообразность использования кодов с параллельной структурой. К числу таких кодов относят непозиционные коды – коды, основанные на модулярной арифметике, то есть коды, в которых числа представляются в системах остаточных классов (СОК).

Полной системой вычетов по модулю p называется совокупность p целых чисел, содержащая точно по одному представителю из каждого класса вычетов по модулю p. Каждый класс вычетов по модулю p содержит в точности одно из чисел совокупности всех возможных остатков от деления на p: $0, 1, \ldots, p-1$. Множество $\{0, 1, \ldots, p-1\}$ называется полной системой наименьших неотрицательных вычетов по модулю p [14].

Приведенной системой вычетов по модулю p, является система чисел взятых по одному и только по одному из каждого класса, взаимно простого с модулем.

Преимуществом модулярного представления числа, заключается в том, что операции сложения, вычитания и умножения выполняются очень просто и параллельно. Пусть заданы числа A и B по формуле (**??**):

$$A \equiv \alpha_1 \left(\mathrm{mod} p_1\right),\ A \equiv \alpha_2 \left(\mathrm{mod} p_2\right),\ \ldots\ , A \equiv \alpha_n \left(\mathrm{mod} p_n\right)$$
$$B \equiv \beta_1 \left(\mathrm{mod} p_1\right),\ B \equiv \beta_2 \left(\mathrm{mod} p_2\right),\ \ldots\ , B \equiv \beta_n \left(\mathrm{mod} p_n\right)$$

Тогда операции сложения, умножения и вычитания можно будет производить по формулам

$$\begin{gathered} A \pm B = (\alpha_1, \alpha_2, \ldots, \alpha_n) \pm (\beta_1, \beta_2, \ldots, \beta_n) = \\ = \left(\left((\alpha_1 \pm \beta_1) \left(\mathrm{mod} p_1\right)\right), \left((\alpha_2 \pm \beta_2) \left(\mathrm{mod} p_2\right)\right), \ldots, \left((\alpha_n \pm \beta_n) \left(\mathrm{mod} p_n\right)\right)\right) \end{gathered} \tag{3.7}$$

$$A \cdot B = (\alpha_1, \alpha_2, \ldots, \alpha_n) \cdot (\beta_1, \beta_2, \ldots, \beta_n) = \\ = (((\alpha_1 \cdot \beta_1) (\mathrm{mod} p_1)), ((\alpha_2 \cdot \beta_2) (\mathrm{mod} p_2)), \ldots, ((\alpha_n \cdot \beta_n) (\mathrm{mod} p_n))) \tag{3.8}$$

Операции сложения, вычитания и умножения в СОК производятся независимо и параллельно, следовательно, на основе данной системы счисления можно создать полностью гомоморфную систему кодирования. Системы кодирования именно такого вида требуются при организации облачных вычислений, так как позволяют защитить данные при удаленном выполнении математических действий.

Область чисел, над которыми можно выполнять операции модулярной арифметики – это множество чисел P, каждое из которых не превышает произведения выбранных модулей $\prod_{i=1}^{n} p_i$ [14].

С целью упрощения процесса перевода чисел из модулярного представления в позиционное представление чисел рассмотрим приближенный метод, который позволяет за счет замены точного значения на приближенное перейти от дорогостоящей операции взятия остатка по большому модулю к взятию дробной части числа и абсолютно правильно реализовать основные классы процедур принятия решений: проверка равенства (неравенства) двух значений; сравнение двух значений (больше, меньше), которые обеспечивают решение основного круга задач, возникающих при аппаратной или программной реализации реальных процессов.

Суть приближенного метода состоит в использовании относительной величины исходного числа к полному диапазону КТО, которая связывает позиционное число X с его представлением в остатках $(x_1, x_2, \ldots, x_n)$ следующим выражением:

$$X = \left| \sum_{i=1}^{n} \frac{P}{p_i} \left| P_i^{-1} \right|_{p_i} x_i \right|_P, \tag{3.9}$$

где x_i – наименьшие неотрицательные вычеты числа по модулям системы остаточных классов $p_1, p_2, \ldots, p_n$, $P = \prod_{i=1}^{n} p_i$, $\left| P_i^{-1} \right|_{p_i}$ – мультипликативная инверсия P_i относительно p_i и для всех $i = \overline{1,n}$ выполняется равенство $P = \frac{P}{p_i}$.

Если формулу (3.9) разделить на диапазон СОК P, то получим приближенное значение:

$$\frac{X}{P} = \left| \sum_{i=1}^{n} \frac{\left| P_i^{-1} \right|_{p_i}}{p_i} x_i \right|_1, \tag{3.10}$$

где для всех $i = \overline{1,n}$ выражение $\frac{\left|P_i^{-1}\right|_{p_i}}{p_i}$ – константы выбранной системы, а x_i – разряды числа, представленного в СОК, при этом значение каждой суммы будет в интервале $[0, 1)$. Конечный результат суммы определяется после суммирования и отбрасывания целой части числа с сохранением дробной части суммы. Дробная часть может быть записана также как $X \bmod 1$, потому что $X = \lfloor X \rfloor + X \bmod 1$. Количество разрядов дробной части числа определяется максимально возможной разностью между соседними числами. В работе [43] показано, что при точности вычислений в N бит восстановление чисел по формуле (3.10) будет корректным, где $N = \lceil \log_2(\rho P) \rceil$ и $\rho = -n + \sum_{i=1}^{n} p_i$.

Аппаратная реализации арифметических операций умножения и сложения вещественных чисел требует в среднем в 3.5 раза больше аппаратных ресурсов чем выполнение этих же операций с целыми числами того же размера, поэтому произведем переход от вещественных чисел к целым, формула (3.10) примет вид:

$$X = \left\lfloor \frac{\left| \sum_{i=1}^{n} \overline{k_i} x_i \right|_{2^N} P}{2^N} \right\rfloor, \tag{3.11}$$

где $\overline{k_i} = \left\lceil \frac{\left|P_i^{-1}\right|_{p_i} 2^N}{p_i} \right\rceil$.

Тогда операция взятия дробной части в формуле (3.10) заменится на операцию взятия младших N бит числа в формуле (3.11), а операция взятия остатка от деления по большому модулю диапазона СОК P заменится на умножение и операцию сдвига вправо на N бит числа.

Исследуем вопрос о размере N.

Теорема 1. *Формула* (3.11) *верна при выборе* N *равном:*

$$N = \left\lceil \log_2 \left(\left(-2n + \sum_{i=1}^{n} p_i \right) P + SQ \right) \right\rceil, \tag{3.12}$$

где $SQ = \sum_{i=1}^{n} P_i$.

Доказательство.

Пусть $\overline{k_i} = \left\lceil \frac{\left|P_i^{-1}\right|_{p_i} 2^N}{p_i} \right\rceil = \frac{\left|P_i^{-1}\right|_{p_i} 2^N}{p_i} + R_i$, где $0 \leqslant R_i < \frac{m_i - 1}{m_1}$.

Вычислим значение $\sum_{i=1}^{n} \overline{k_i} x_i$:

$$\begin{aligned}\sum_{i=1}^{n} \overline{k_i} x_i &= \sum_{i=1}^{n-1} \left(\frac{\left|P_i^{-1}\right|_{p_i} 2^N}{p_i} + R_i \right) x_i = \\ &= \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i} 2^N}{p_i} x_i + \sum_{i=1}^{n} R_i x_i = \\ &= 2^N \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i} 2^N}{p_i} x_i + \sum_{i=1}^{n} R_i x_i.\end{aligned} \tag{3.13}$$

Значение $\left| \sum_{i=1}^{n} \overline{k_i} x_i \right|_{2^N}$ равно:

$$\left| \sum_{i=1}^{n} \overline{k_i} x_i \right|_{2^N} = \sum_{i=1}^{n} \overline{k_i} x_i - \left\lfloor \sum_{i=1}^{n} \frac{\overline{k_i} x_i}{2^N} \right\rfloor \cdot 2^N \tag{3.14}$$

Подставляя (3.13) в (3.14) получим:

$$\left| \sum_{i=1}^{n} \overline{k_i} x_i \right|_{2^N} = 2^N \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i + \sum_{i=1}^{n} R_i x_i - \left\lfloor \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i + \frac{\sum_{i=1}^{n} R_i x_i}{2^N} \right\rfloor 2^N \tag{3.15}$$

Подставим формулу (3.15) в правую часть формулы (3.11), получим:

$$\begin{aligned}&\left\lfloor \frac{\left| \sum_{i=1}^{n} \overline{k_i} x_i \right|_{2^n} P}{2^N} \right\rfloor = \\ &= \left\lfloor P \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i + \frac{P}{2^N} \sum_{i=1}^{n} R_i x_i \right\rfloor - \left\lfloor \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i + \frac{\sum_{i=1}^{n} R_i x_i}{2^N} \right\rfloor P\end{aligned} \tag{3.16}$$

Учитывая, что по Китайской теореме об остатках:

$$X = P \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i - \left\lfloor \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i \right\rfloor P \tag{3.17}$$

Формула (3.11) будет эквивалентна формуле (3.17), если будут выполняться два условия:

1. $\left\lfloor P \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i + \frac{P}{2^N} \sum_{i=1}^{n} R_i x_i \right\rfloor = P \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i,$

   эквивалентно: $\frac{P}{2^N} \sum_{i=1}^{n} R_i x_i < 1.$

2. $\left\lfloor \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i + \frac{\sum_{i=1}^{n} R_i x_i}{2^N} \right\rfloor = \left\lfloor \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i \right\rfloor$, эквивалентно $\frac{\sum_{i=1}^{n} R_i x_i}{2^N} < \frac{1}{P}$.

Условия 1 и 2 равносильны, следовательно, необходимо и достаточного, чтобы выполнялось условие:

$$\frac{\sum_{i=1}^{n} R_i x_i}{2^N} < \frac{1}{P}. \tag{3.18}$$

Из неравенства (3.18) следует, что необходимым и достаточным условием является:

$$2^N > P \sum_{i=1}^{n} R_i x_i \tag{3.19}$$

Оценив правую часть неравенства (3.19), получим:

$$\begin{aligned} P \sum_{i=1}^{n} R_i x_i &< P \sum_{i=1}^{n} \frac{m_i - 1}{m_i} (m_i - 1) = \\ &= P \sum_{i=1}^{n} (m_1 - 1) - P \sum_{i=1}^{n} \left(1 - \frac{1}{m_i}\right) = \\ &= -nP + P \sum_{i=1}^{n} m_i - nP + SQ = \\ &= \left(-2n + \sum_{i=1}^{n} m_i\right) P + SQ. \end{aligned} \tag{3.20}$$

Из формулы (3.20) и неравенства (3.19) следует, что если выбрать $N = \left\lceil \log_2 \left(\left(-2n + \sum_{i=1}^{n} p_i \right) P + SQ \right) \right\rceil$, то формула (3.11) является верной.

Теорема доказана.

Покажем, что $N = \left\lceil \log_2 \left(\left(-2n + \sum_{i=1}^{n} p_i \right) P + SQ \right) \right\rceil \leqslant \lceil \log_2 (\rho P) \rceil$.

Для этого найдем разницу

$$\rho P - \left(\left(-2n + \sum_{i=1}^{n} p_i \right) P + SQ \right) = nP - SQ = \sum_{i=1}^{n} (P - P_i) > 0. \tag{3.21}$$

Из формулы (3.21) следует, что полученная оценка значения N является более точной, чем оценка из работы [43].

Согласно Китайской теореме об остатках значение X может быть вычислено по формуле (3.9) или:

$$X = \sum_{i=1}^{n} P_i \left|P_i^{-1}\right|_{p_i} x_i - \underbrace{\left\lfloor \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i \right\rfloor}_{r_X} P. \tag{3.22}$$

где r_X – ранг числа, целое положительное число, показывающее во сколько раз диапазон системы был превзойден при переходе от представления числа в системе остаточных классов к его представлению через систему ортогональных базисов [43].

Из формулы (3.22) следует, что при вычислении r_X требуется либо использовать дорогостоящую операцию целочисленного деления, либо работать с вещественным числами с точностью N, для корректного определения ранга числа.

Для эффективной реализации алгоритма вычисления ранга числа используем подход основанный на одновременном применении приближенного метода и модульного сумматора, который позволит существенно понизить точность вычислений:

$$r = \left\lfloor \sum_{i=1}^{n} \overline{\overline{k}}_i x_i / 2^{N_1} \right\rfloor, \tag{3.23}$$

где $\overline{\overline{k}}_i = \left\lceil \frac{\left|P_i^{-1}\right|_{p_i} 2^{N_1}}{p_i} \right\rceil$.

Исследуем вопрос связи между значениями N_1, r и r_X.

Теорема 2. *1. Если $N_1 = N$, то $r_X = r$.*

2. Если $N_1 = \lceil \log_2 \rho \rceil$, то $r_X = r$ или $r_X = r - 1$, где $\rho = \sum_{i=1}^{n} p_i - n$.

Доказательство

Пусть $\overline{\overline{k}}_i = \left\lceil \frac{\left|P_i^{-1}\right|_{p_i} 2^{N_1}}{p_i} \right\rceil = \frac{\left|P_i^{-1}\right|_{p_i} 2^{N_1}}{p_i} + R'$ где $0 \leqslant R'_i \leqslant \frac{m_i - 1}{m_i}$.

Вычислим значение $\sum_{i=1}^{n} \overline{\overline{k}}_i x_i$:

$$\begin{aligned} \sum_{i=1}^{n} \overline{\overline{k}}_i x_i = \sum_{i=1}^{n} \left(\frac{\left|P_i^{-1}\right|_{p_i} 2^{N_1}}{p_i} + R' \right) x_i = \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i} 2^{N_1}}{p_i} x_i + \sum_{i=1}^{n} R'_i x_i = \\ = 2^{N_1} \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i + \sum_{i=1}^{n} R'_i x_i \end{aligned} \tag{3.24}$$

Подставляя формулу (3.23) в (3.24), получим

$$r = \left\lfloor \frac{\sum\limits_{i=1}^{n} \overline{\overline{k_i}} x_i}{2^{N_1}} \right\rfloor = \left\lfloor \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} x_i + \frac{\sum\limits_{i=1}^{n} R'_i x_i}{2^{N_1}} \right\rfloor \tag{3.25}$$

Из формул (3.22) и (3.25) следует, что $r = r_x$ при $\frac{\sum\limits_{i=1}^{n} R'_i x_i}{2^{N_1}} < \frac{1}{P}$. Согласно теореме 2, данное неравенство выполняется при $N_1 = N$.

Если $r_x = r$ или $r_x = r - 1$, то из формулы (3.25) следует, что достаточным условием является $\frac{\sum\limits_{i=1}^{n} R'_i x_i}{2^{N_1}} < 1$, следовательно, $\sum\limits_{i=1}^{n} R'_i x_i < 2^{N_1}$.

Так как $\sum\limits_{i=1}^{n} R'_i x_i < \sum\limits_{i=1}^{n} (p_i - 1) = -n + \sum\limits_{i=1}^{n} p_i = \rho$, то $N_1 = \lceil \log_2 \rho \rceil$ будет достаточным для выполнения второго условия теоремы 2.

Схема аппаратной реализации представлена на 3.1.

Пример 1. Пусть заданы модули СОК $p_1 = 17$, $p_2 = 19$, $p_3 = 23$, $p_4 = 25$. Диапазон СОК $P = 17 \cdot 19 \cdot 23 \cdot 25 = 185\,725$.

Вычислим P_i: $P_1 = \frac{P}{p_1} = 10\,925$, $P_2 = \frac{P}{p_2} = 9\,775$, $P_3 = \frac{P}{p_3} = 8\,075$, $P_4 = \frac{P}{p_4} = 7\,429$.

Найдем значения SQ : $SQ = 10\,925 + 9\,775 + 8\,075 + 7\,429$.

Вычислим параметры приближенного метода:

$$N = \lceil \log_2 14151304 \rceil = 24,\ N_1 = \lceil \log_2 80 \rceil = 7$$
$$k_1 = \left|P_1^{-1}\right|_{p_1} P_1 = 152950,\ k_2 = \left|P_2^{-1}\right|_{p_2} P_2 = 166175$$
$$k_3 = \left|P_3^{-1}\right|_{p_3} P_3 = 96900,\ k_4 = \left|P_4^{-1}\right|_{p_4} P_4 = 141151$$
$$\overline{k_1} = \left\lceil \frac{\left|P_1^{-1}\right|_{p_1} 2^N}{p_1} \right\rceil = 13816531,\ \overline{k_2} = \left\lceil \frac{\left|P_2^{-1}\right|_{p_2} 2^N}{p_2} \right\rceil = 15011194$$
$$\overline{k_3} = \left\lceil \frac{\left|P_3^{-1}\right|_{p_3} 2^N}{p_3} \right\rceil = 8753331,\ \overline{k_4} = \left\lceil \frac{\left|P_4^{-1}\right|_{p_4} 2^N}{p_4} \right\rceil = 12750685$$
$$\overline{\overline{k_1}} = \left\lceil \frac{\left|P_1^{-1}\right|_{p_1} 2^{N_1}}{p_1} \right\rceil = 106,\ \overline{\overline{k_2}} = \left\lceil \frac{\left|P_2^{-1}\right|_{p_2} 2^{N_1}}{p_2} \right\rceil = 115$$
$$\overline{\overline{k_3}} = \left\lceil \frac{\left|P_3^{-1}\right|_{p_3} 2^{N_1}}{p_3} \right\rceil = 67,\ \overline{\overline{k_4}} = \left\lceil \frac{\left|P_4^{-1}\right|_{p_4} 2^{N_1}}{p_4} \right\rceil = 98$$

Пусть в СОК заданы числа $X \xrightarrow{COK} \{16, 18, 22, 24\}$ и $Y \xrightarrow{COK} \{1, 2, 3, 4\}$.

1. Вычислим значения X и Y с использованием приближенного метода

$$16 \cdot 13816531 + 18 \cdot 15011194 + 22 \cdot 8753331 + 24 \cdot 2750685 = 749855710.$$

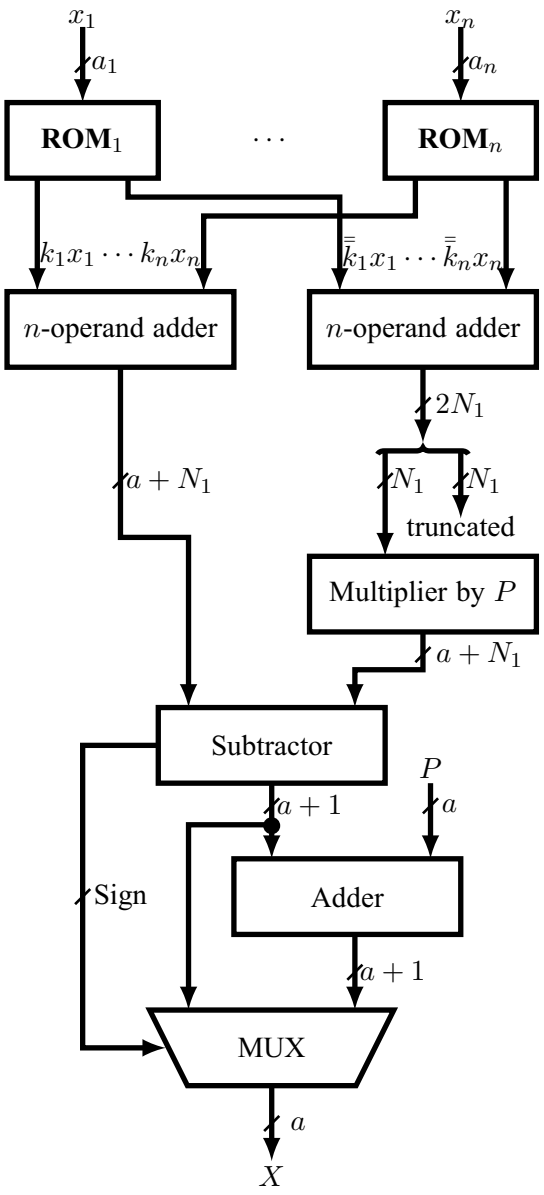

Рисунок 3.1 — Аппаратная реализация алгоритма перевода числа из СОК в ПСС на основе ранга числа, вычисляемого с помощью приближенного метода

$$|749855710|_{2^{24}} = 16777182.$$
$$X = \left\lfloor \frac{16777182 \cdot 185725}{2^{24}} \right\rfloor = 185724.$$
$$1 \cdot 13816531 + 2 \cdot 15011194 + 3 \cdot 8753331 + 4 \cdot 2750685 = 81101652.$$
$$|81101652|_{2^{24}} = 3661140.$$
$$Y = \left\lfloor \frac{3661140 \cdot 185725}{2^{24}} \right\rfloor = 40529.$$

2. Вычислим значения X и Y на основе теоремы **??** и ранга функции по формуле (3.23).

$$\sum_{i=1}^{n} k_i x_i = 16 \cdot 152950 + 18 \cdot 166175 + 22 \cdot 96900 + 24 \cdot 141151 = 10957774.$$
$$r_X = \left\lfloor \frac{10957774}{185725} \right\rfloor = 58$$

$$X = 10957774 - 58 \cdot 185725 = 185724$$
$$\sum_{i=1}^{n} k_i y_i = 1 \cdot 152950 + 2 \cdot 166175 + 3 \cdot 96900 + 4 \cdot 141151 = 1340604.$$
$$r_Y = \left\lfloor \frac{1340604}{185725} \right\rfloor = 7$$
$$Y = 1340604 - 7 \cdot 185725 = 40529$$

3. Вычислим значения X и Y с использованием теоремы **??** и ранга числа, вычисленного с помощью приближенного метода с точностью N.

$$\sum_{i=1}^{n} \overline{\overline{k_i}} x_i = 16 \cdot 13816531 + 18 \cdot 15011194 + 22 \cdot 8753331 + 24 \cdot 12750685 = 989855710$$
$$r = \left\lfloor \frac{989855710}{2^{24}} \right\rfloor = 58 = r_X,$$
$$X = 10957774 - 58 \cdot 185725 = 185724$$
$$\sum_{i=1}^{n} \overline{\overline{k_i}} y_i = 1 \cdot 13816531 + 2 \cdot 15011194 + 3 \cdot 8753331 + 4 \cdot 12750685 = 121101652$$
$$r = \left\lfloor \frac{121101652}{2^{24}} \right\rfloor = 7 = r_Y,$$
$$Y = 1340604 - 7 \cdot 185725 = 40529$$

4. Вычислим значения X и Y с использованием КТО и ранга числа, вычисленного с помощью приближенного метода с точностью N_1.

$$\sum_{i=1}^{n} \overline{\overline{k_i}} x_i = 16 \cdot 106 + 18 \cdot 115 + 22 \cdot 67 + 24 \cdot 98 = 7592$$
$$r = \left\lfloor \frac{7592}{2^7} \right\rfloor = 59,$$
$X = 10957774 - 59 \cdot 185725 = -1$, так как $X < 0$ то $X = -1 + 185725 = 185724$
$$\sum_{i=1}^{n} \overline{\overline{k_i}} y_i = 1 \cdot 106 + 2 \cdot 115 + 3 \cdot 67 + 4 \cdot 98 = 929$$
$$r = \left\lfloor \frac{929}{2^7} \right\rfloor = 7$$
$X = 1340604 - 7 \cdot 185725 = 40529$, так как $X < 0$ то $X = -1 + 185725 = 185724$.

Так как $Y \geqslant 0$ то результат остается без изменений.

Из примера 1 можно сделать вывод о том, что применение случая 2 из теоремы 2 позволяет уменьшить размер чисел при вычислении ранга числа и не использовать дорогостоящую с точки зрения аппаратных ресурсов операцию целочисленного деления.

3.4 Разработка модели отказоустойчивой обработки и хранения больших данных в облачной среде на основе двухуровневой системы остаточных классов

Мультиоблако (multi-cloud) системы представляют собой возможную альтернативу использования единого облака. В данном случае вместо одного облачного провайдера используется сразу несколько, что позволяет минимизировать риски потери, компрометации и раскрытия данных, а так же повысить надежность хранения данных. Однако в данном случае особого подхода требует обработка данных. В работе [41] предложено использование избыточной системы остаточных классов для распределения вычислений между различными облачными провайдерами. Использование СОК имеет ряд преимуществ. Во-первых, отдельный облачный провайдер не получает достаточной информации для восстановления исходных данных. Во-вторых, избыточное представление в СОК позволяет применять методы контроля и локализации ошибок в коде для повышения надежности системы.

Применение СОК базируется на распределении данных между различными вычислителями на основе остаточного представления. Все вычисления, проводимые облачными серверами, производятся независимо. Тем самым достигается максимальная эффективность и надежность операций. Повышение эффективности и надежности операций накладываемых на мультиоблачную систему для хранения и обработки данных приводит к модели двухуровневой системы остаточных классов. Каждый уровень такой системы представляет собой обособленную полноценную систему остаточных классов. Остатки первого уровня служат числами для второго, соответственно диапазон и модули первого превышают диапазон и модули второго. Каждый уровень при этом может решать свою конкретную задачу в зависимости от того, какая цель стоит перед всей системой в целом.

Использование двухуровневой СОК при проектировании мультиоблачной системы позволяет разделить задачи каждого из уровней. Основные вычисления производятся на втором уровне, следовательно, необходимо подбирать для него систему оснований так, чтобы арифметические операции на нем были максимально эффективны. Этого можно добиться, используя специальные наборы оснований. Однако выбор оснований для каждого уровня является сложной зада-

чей. Основная причина – это множество условий, накладываемых одновременно на оба уровня СОК. Кроме того, необходимо учитывать, что взаимодействие двух отдельных СОК такого рода может повлечь дополнительные накладные расходы при вычислениях [22, 74].

Пусть на первом уровне используются модули $\{p_1, p_2, \ldots, p_n\}$ с диапазоном $P = p_1 \cdot p_2 \cdot \ldots \cdot p_n$, что представляет собой общий диапазон вычислений. Модуль с номером i ассоциирован с отдельным облачным провайдером для всех $i = 1, 2, \ldots, n$. Модули, соответствующие второму уровню для данного провайдера будем обозначать как $\{p_{i,1}, p_{i,2}, \ldots, p_{i,n_i}\}$, а их диапазон обозначим как $P_i = p_{i,1} \cdot p_{i,2} \cdot \ldots \cdot p_{i,n_i}$.

На рисунке 2.10 представлена схема обработки и хранения данных в облачной среде на основе двухуровневой системы остаточных классов. Схема обработки и хранения данных в облачной среде представленная на рисунке 2.10 состоит из облачной (рисунок 2.12) и пользовательской (рисунок 2.11) частей, рассмотрим работу схемы. Работа данной схемы подробно описана в параграфе 2.5.2

Проведем расчет надежности мультиоблачной модели хранения и обработки данных представленной на рисунке 2.10, расчет будем производить по формуле (3.26), результаты расчетов приведены в приложении Ж, таблицы 23, 24, 25, 26, 22.

$$P_4 = \sum_{i=n_2-k_2}^{n_2} C_{n_2}^i P_0^i q_4^{n_2-i} \tag{3.26}$$

при условиях:

1. $P_0 = \sum\limits_{i=n_1-k_1}^{n_1} C_{n_1}^i P_d^i q_0^{n_1-i}$;
2. $R = \frac{n_1 \cdot n_2}{k_1 \cdot k_2}$;
3. $q_4 = 1 - P_0, q_0 = 1 - P_d$
4. $\begin{cases} P_4 < 1.6 \cdot 10^{-10} \\ R < 2.5 \end{cases}$

Построение многоуровневых моделей обработки данных на облачных серверах с использованием избыточной СОК позволяет повысить надежность хранимых и обрабатываемых данных. Использование многоуровневых моделей обработки данных на основанных избыточной СОК и СРД, позволяет строить модели надежнее одноуровневых моделей и модели компании «Google Inc.» [56]. При построении модели обработки данных в используется избыточная СОК,

Таблица 10 — Сравнение систем построенных на основе одноуровневой и двухуровневой системы остаточных классов

Диски	Модель	Избыточность	Вероятность отказа	Схема
WDC, 6 TB	Двухуровневая $(k_1, r_1) - (k_2)$ схема	$\approx 125\%$	$0.3357 \cdot 10^{-13}$	$(4,5) - (4)$
	Двухуровневая $(k_1, r_1) - (k_2, r_2)$ схема	$\approx 60\%$	$0.3377 \cdot 10^{-14}$	$(7,1) - (5,2)$
	Одноуровневая (k, r) схема	$\approx 80\%$	$0.1828 \cdot 10^{-13}$	$(5,4)$
WDC, 2 TB	Двухуровневая $(k_1, r_1) - (k_2)$ схема	$\approx 125\%$	$0.1551 \cdot 10^{-10}$	$(4,5) - (3)$
	Двухуровневая $(k_1, r_1) - (k_2, r_2)$ схема	$\approx 60\%$	$0.8859 \cdot 10^{-11}$	$(7,1) - (5,2)$
	Одноуровневая (k, r) схема	$\approx 80\%$	$0.1295 \cdot 10^{-10}$	$(5,4)$
HGST, 3 TB	Двухуровневая $(k_1, n_1) - (k_2)$ схема	$\approx 125\%$	$0.1355 \cdot 10^{-14}$	$(4,5) - (3)$
	Двухуровневая $(k_1, r_1) - (k_2, r_2)$ схема	$\approx 60\%$	$0.7621 \cdot 10^{-15}$	$(7,1) - (5,2)$
	Одноуровневая (k, r) схема	$\approx 80\%$	$0.5285 \cdot 10^{-14}$	$(5,4)$
HGST, 2 TB	Двухуровневая $(k_1, r_1) - (k_2)$ схема	$\approx 125\%$	$0.1425 \cdot 10^{-14}$	$(4,5) - (3)$
	Двухуровневая $(k_1, r_1) - (k_2, r_2)$ схема	$\approx 60\%$	$0.8017 \cdot 10^{-15}$	$(7,1) - (5,2)$
	Одноуровневая (k, r) схема	$\approx 80\%$	$0.5513 \cdot 10^{-14}$	$(5,4)$
Toshiba, 3 TB	Двухуровневая $(k_1, r_1) - (k_2)$ схема	$\approx 125\%$	$0.1561 \cdot 10^{-12}$	$(4,5) - (3)$
	Двухуровневая $(k_1, r_1) - (k_2, r_2)$ схема	$\approx 60\%$	$0.8826 \cdot 10^{-13}$	$(7,1) - (5,2)$
	Одноуровневая (k, r) схема	$\approx 80\%$	$0.2775 \cdot 10^{-12}$	$(5,4)$

минимальная вероятность отказа оборудования может составлять $1.4 \cdot 10^{-112}$ при использовании схем $(2,7); (3,6)$ в этом случае избыточность будет составлять $\approx 1250\%$, эта схема мало пригодна для решения поставленной задачи, так как имеет очень большую избыточность хранимых и обрабатываемых данных в облачной среде. Применение схемы обработки и хранения, с параметрами схемы, $(6,2); (6,3))$, в этом случае вероятность отказа составит $4.382 \cdot 10^{-13}$ при избыточности $\approx 100\%$.

Построение схем при которых на одном из слоев будет применяться классическая СОК, а на другом избыточная СОК позволяет строить схемы менее надежные, чем одномерные схемы, поэтому данные схемы нецелесообразно применять для решения поставленных задач.

3.5 Выводы по третьей главе

1. Для повышения отказоустойчивости разработанных методов при отказе облачного сервера, применяется перераспределение обрабатываемых данных между доступными серверами. Введение небольшой избыточности позволяет производить обработку или восстановление хранимых

данных в случае отказа контрольных серверов. Разработанные методы надежного и отказоустойчивого хранения построенные на базе двухуровневой СОК, которые имеют преимущество в надежности, отказоустойчивости и избыточности хранимых данных.

2. Разработан метод декодирования данных из системы остаточных классов в позиционную систему счисления. Предложен новый алгоритм вычисления ранга числа на основе приближенного метода. Разработана новая вычислительная архитектура для перевода чисел из системы остаточных классов в позиционную систему счисления, основанная на вычислении ранга числа с использованием приближенного метода. Доказана корректность разработанного алгоритма.
3. Разработана модель хранения построенная на основе СОК. Данная модель позволяет производить восстановление хранимых и обрабатываемых данных в случае выхода из строя одного или нескольких облачных серверов. Использование многоуровневых моделей обработки данных на системе остаточных классах позволяет строить модели надежнее одноуровневых моделей и моделей основанных на ПСС.

Глава 4. МОДЕЛИРОВАНИЕ СИСТЕМЫ ХРАНЕНИЯ И ОБРАБОТКИ БОЛЬШИХ ДАННЫХ В ОБЛАЧНОЙ СРЕДЕ НА ОСНОВЕ СИСТЕМЫ ОСТАТОЧНЫХ КЛАССОВ

4.1 Модель распределенной обработки данных основанная на системе остаточных классов

Некоторые из традиционных и новых облачных сервисов приложений включают в себя социальные сети, веб-хостинг, доставку контента и обработку инструментальных данных в режиме реального времени. Каждый из этих типов приложений имеет различные требования к составу, конфигурации и развертыванию. Количественная оценка эффективности политик обеспечения (планирования и распределения) в реальной облачной вычислительной среде (Amazon EC2 [96], Microsoft Azure [36], Google App Engine [56]) для разных моделей приложений в переходных условиях чрезвычайно сложна, потому что:

1. облака демонстрируют различные требования, модели поставок, размеры и ресурсы системы (аппаратное обеспечение, программное обеспечение, сеть);
2. пользователи имеют гетерогенные, динамические и конкурирующие требования QoS;
3. Приложения имеют различные характеристики производительности, рабочей нагрузки и динамического масштабирования приложений.

Использование реальных инфраструктур, таких как Amazon EC2 и Microsoft Azure, для сопоставления производительности приложения (пропускной способности, стоимости) в переменных условиях часто ограничивается жесткостью инфраструктуры. Следовательно, это позволяет воспроизводить результаты, на которым можно доверять очень сложно. Также невозможно выполнить эксперименты по бенчмаркингу в повторяющихся, надежных и масштабируемых средах с использованием реальных облачных сред.

Более жизнеспособной альтернативой является использование инструментов моделирования. Эти инструменты открывают возможность оценки гипотезы (исследование сравнительного тестирования приложений) в контролируемой среде, где можно легко воспроизвести результаты. Учитывая, что ни один из

существующих распределенных (в том числе Grid и Network) симуляторов системы [29, 51, 75] не предлагает среду, которая может быть непосредственно использована для моделирования облачных вычислительных сред, нами был разработан симулятор CloudStorageSim кторый позволяет производить моделирование и эксперименты с реальными облачными вычислительными инфраструктурами.

CloudStorageSim позволяет:

1. проводить моделирование крупномасштабных облачных вычислительных сред, включая центры обработки данных, на одном физическом вычислительном узле;
2. строить автономную платформу для моделирования облаков, брокеров услуг, обработки и распределения данных;
3. поддерживает моделирование сетевых соединений между элементами имитируемой системой;
4. проводить моделирование федеративной облачной среды, которая использует межсетевые ресурсы как из частных, так и государственных доменов, что является критическим для исследований, связанных с облачными отказами и автоматическим масштабированием приложений.

Некоторые из уникальных особенностей CloudStorageSim:

1. наличие механизма виртуализации, который помогает в создании и управлении несколькими, независимыми и совместно размещенными виртуализованными службами на узле центра обработки данных
2. гибкость переключения между пространственно-разделенным и распределенным по времени распределением процессорных ядер виртуализованных служб.

Для создания надежной и безопасной модели хранения данных в распределенной облачной структуре нами будут использоваться избыточная СОК и коды исправления ошибок. Хранилище данных обладает следующими свойствами: надежность, целостность, распределение данных и коды исправления ошибок.

Пусть i-е облако соответствует модулю избыточной СОК p_i вида $2^b - \alpha_i$, где b – длина модуля, а α_i – малое целое число. Значения b и α_i выбираются согласно доступным вычислительным ресурсам i-го облака, требуемому уровню безопасности и надежности. Количество облаков равняется n. Используя $(k,\, n)$ схему разделения данных мы можем восстановить данные при доступности любых k облаков. Поскольку диапазон ИСОК равен P, где $L = \lceil \log_2 P \rceil \approx k \cdot b$ обозначает размер данных. В этом случае каждый облачный провайдер полу-

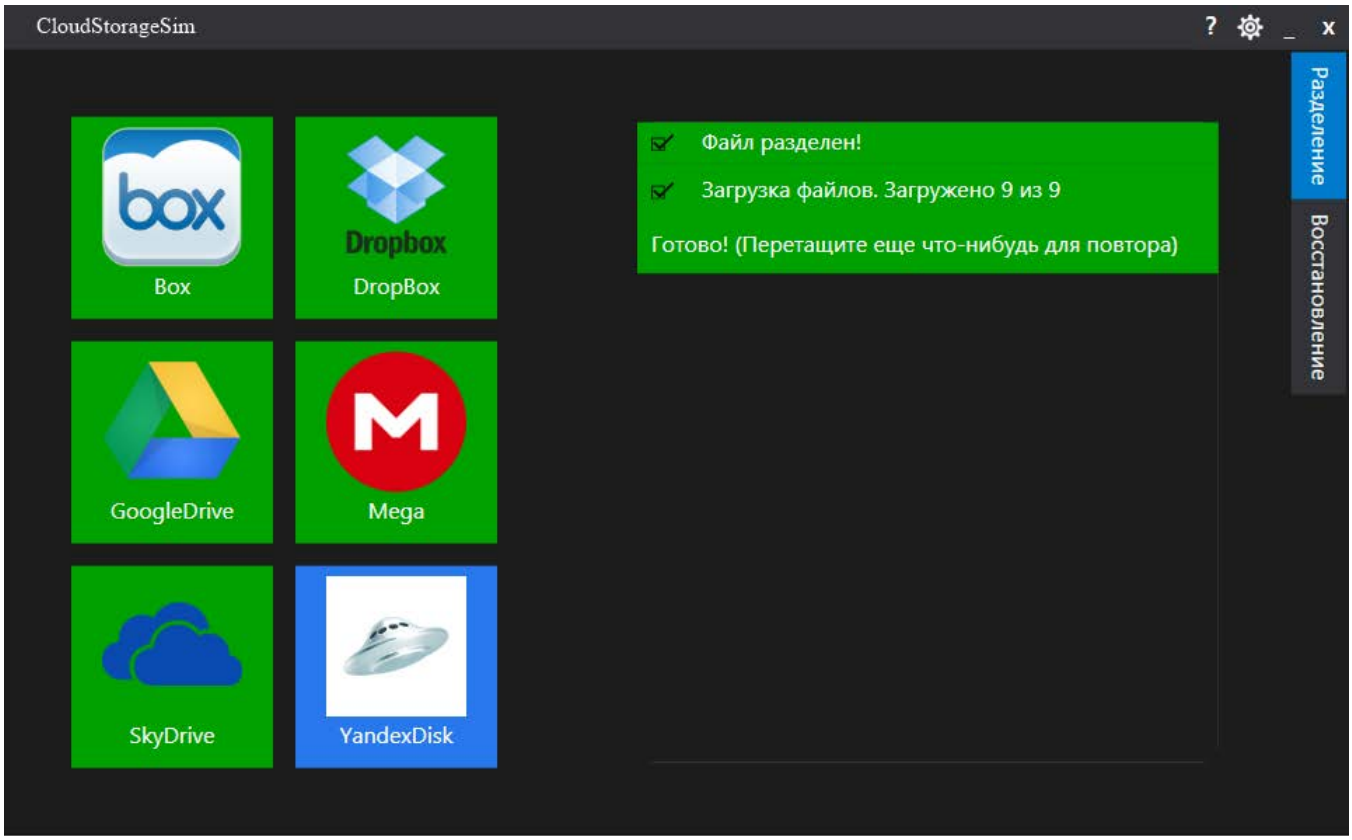

Рисунок 4.1 — Внешний вид среды для моделирования распределенного облачного хранения данных CloudStorageSim

чает кусок данных, который состоит из идентификатора канала, свойств блока, проекции исходных данных и упрощенной цифровой подписи.

В процессе эксплуатации облачные серверы могут быть недоступными в течении некоторого промежутка времени. Например, доступ к информации сервиса Amazon в 2009 году был ограничен в течение длительного времени из-за DDoS-атак. В 2013 году у сервисов Amazon, Microsoft и Google произошла серия отключений облачных серверов. Технические сбои и потери данных из-за сбоев происходили у компаний Amazon, Dropbox, Microsoft, Google и Yandex. В первом квартале 2014 года у сервиса Dropbox наблюдались перебои в обслуживании. В связи с банкротством компании Nirvanix в 2013 году большое количество пользователей потеряли свои данные.

Для противодействию и устранению DDoS атак веб-сервис Greatfire.org тратит до 30 000 долларов в день. Одна из самых мощных DDoS атак произошла в октябре 2016 года. Пользователи не могли получить доступ к своим данным около 11 часов. Согласно отчету об атаках DDoS в первом квартале 2016 года, наибольшая атака длилась 197 часов или 8,2 дня. Вероятность потери информации для различных параметров системы показана на рисунке 4.2.

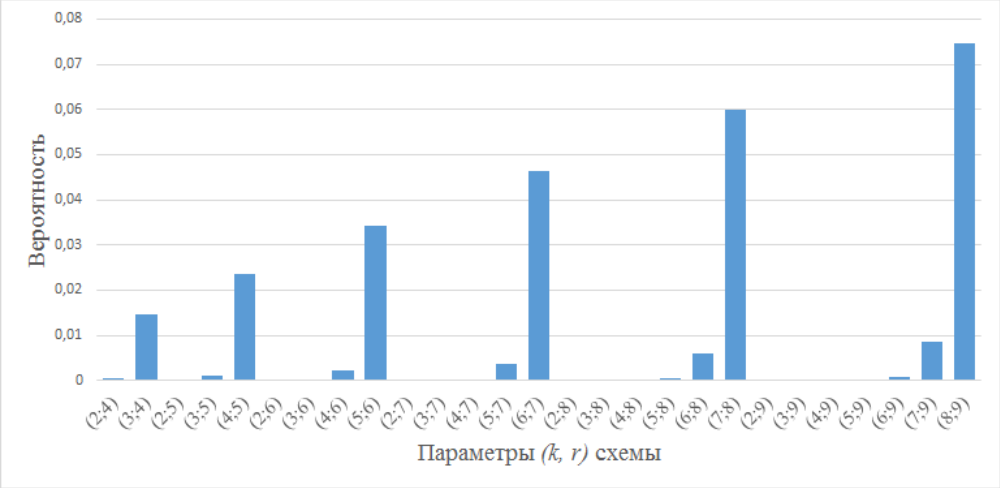

Рисунок 4.2 — Вероятность потери информации при различных параметров избыточной СОК $(k,\ r)$

Можно сделать вывод о том, что предложенная схема хранения данных обеспечивает наименьшую вероятность потери данных с параметрами избыточной СОК $(k,\ n)$ в пределах от $(2,\ n)$ до $(n-2,\ n)$, где $n = 4, 5, \ldots, 9$. Наибольшая вероятность потери данных наблюдается при параметрах системы $(n,\ n)$.

Выполнение резервного копирования данных в большом хранилище является важной задачей по хранению данных в облачной среде. Необходимое количество бит для сохранения, в худшем случае $(p_1 - 1, p_2 - 1, \ldots, p_n - 1)$, будет приблизительно равняться $\sum_{i=1}^{n} \log_2 p_i$. Входные данные L приблизительно равны $\sum_{i=1}^{k} \log_2 p_i$. Произведем вычисление избыточности как отношение сохраненных закодированных данных к исходному размеру данных, пусть модули СОК удовлетворяют условию:

$$2^{b-1} < p_1 < p_2 < \cdots < p_n < 2^b$$

Тогда избыточность будет удовлетворять следующему неравенству

$$\frac{\sum_{i=1}^{n}(b-1)}{\sum_{i=1}^{k} b} < \frac{\sum_{i=1}^{n} \log_2 p_i}{\sum_{i=1}^{k} \log_2 p_i} \leqslant \frac{\sum_{i=1}^{n} b}{\sum_{i=1}^{k} b}$$

Следовательно, избыточность примерно равняется отношению $\frac{n}{k}$, параметры избыточности СОК показаны на рисунке 4.3.

Для анализа скорости кодирования/декодирования данных нами был использован ЦОД СКФУ с техническими характеристиками: ОС: Ubuntu 16.04.4

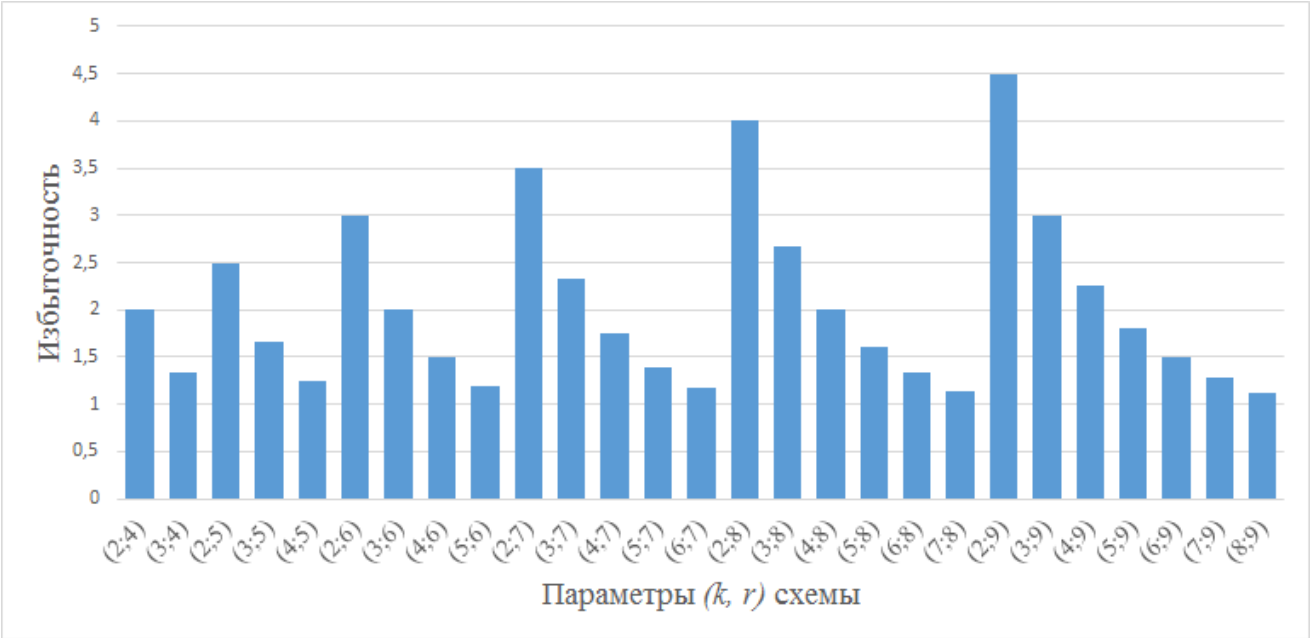

Рисунок 4.3 — Избыточность СОК при различных параметрах $(k,\ r)$

LTS x86_64, Процессор: Intel Xeon E5-2690V4 2.6 ГГц, Оперативная память: 125.92 Гб, Жесткий диск: 1024 Гб, 7200 об/мин.

Основная цель алгоритмов предложенная в работе [44], состоит в том, чтобы найти остаток деления L-битного слова по модулю вида $2^{b_i} \pm \alpha_i$. Если $b_1 = b_2 = \cdots b_n = b$, то эффективный алгоритм основан на нейронной сети конечного кольца. Его последняя модификация представленная в работе [41] позволяет достичь алгоритмической сложности $b \cdot \log_2 k$. Следовательно, мы предполагаем, что для вычисления остальной части деления требуется примерно $b \cdot \log_2 k$ бит операций. Чтобы представить L-разрядное число в СОК, требуется выполнить операцию поиска остатка деления по модулю СОК n раз. Таким образом, общее количество бит-операций – $n \cdot b \cdot \log_2 k$. Поскольку размер входного блока равен $L \approx k \cdot b$ битам, количество блоков в 1 МБ составляет $\frac{2^{23}}{L} \approx \frac{2^{23}}{k \cdot b}$. Поэтому, чтобы кодировать 1 Мб данных, требуются $\frac{2^{23} \cdot n \cdot b \cdot \log_2 k}{k \cdot b} = \frac{2^{23} \cdot n \cdot \log_2 k}{k}$ операций.

Скорость кодирования данных в МБ/с может быть рассчитана по формуле:

$$V_C = \frac{2^{30} \cdot k}{2^{23} \cdot n \cdot \log_2 k} = \frac{k \cdot 2^7}{n \cdot \log_2 k}$$

Зависимость скорости кодирования данных от параметров схемы показана на рисунке 4.4.

Пользователь может выбрать требуемые параметры для обеспечения требуемого значения скорости кодирования данных.

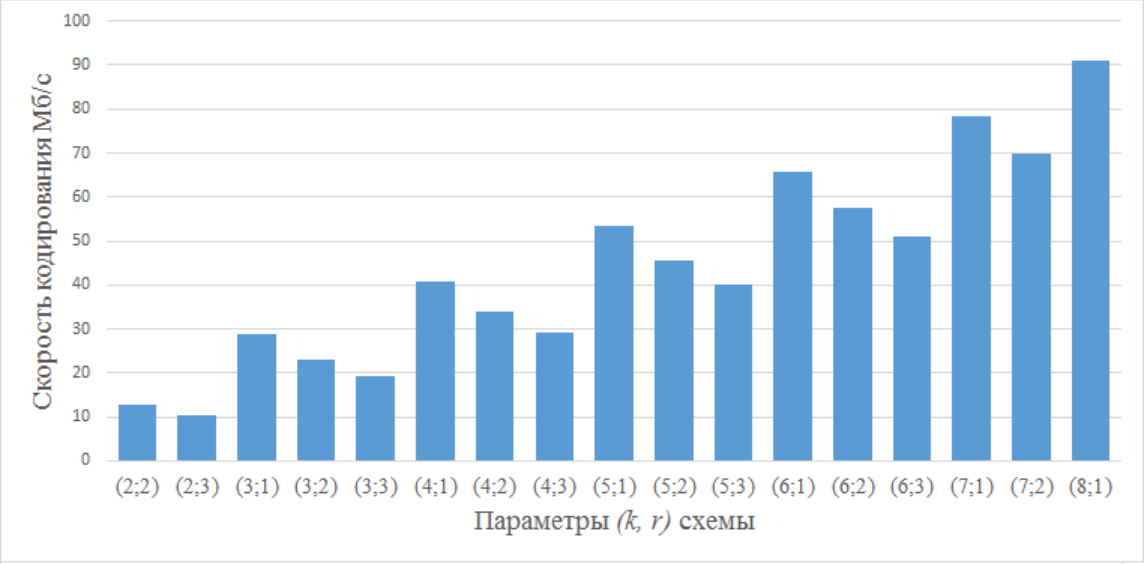

Рисунок 4.4 — Скорость кодирования данных (Мб/с) при различных параметрах $(k,\, r)$ избыточной СОК

Когда данные декодируются без ошибок, согласно КТО алгоритмическая сложность будет равняться $O\left(L^2\right)$, для выполнения данной операции необходимо провести примерно $L^2 \approx k^2 \cdot b^2$ битовых операций. В случае если мы имеем r избыточных модулей СОК мы можем обнаружить r и исправить $r-1$ ошибок.

Чтобы обнаружить и локализовать ошибку, мы используем алгоритм, основанный на проекциях. Для вычисления проекции мы используем КТО, поэтому одна проекция вычисляется примерно в $L^2 \approx k^2 \cdot b^2$-битных операциях.

Так как число проекций равняется C_n^{k+1}, чтобы обнаружить и локализовать ошибку нам нужно выполнить $C_n^{k+1} \cdot k^2 \cdot b^2$ битных операции. Чтобы закодировать 1 Мб данных требуется $\frac{2^{23} C_n^{k+1} \cdot k^2 \cdot b^2}{k \cdot b}$ битовых операции. Следовательно, скорость декодирования будет равняться

$$V_D = \frac{2^{30}}{2^{23} C_n^{k+1} \cdot k \cdot b} = \frac{2^7}{C_n^{k+1} \cdot k \cdot b}.$$

Зависимость скорости декодирования данных от параметров схемы показана на рисунке 4.5 для $b = \{8, 16, 32\}$.

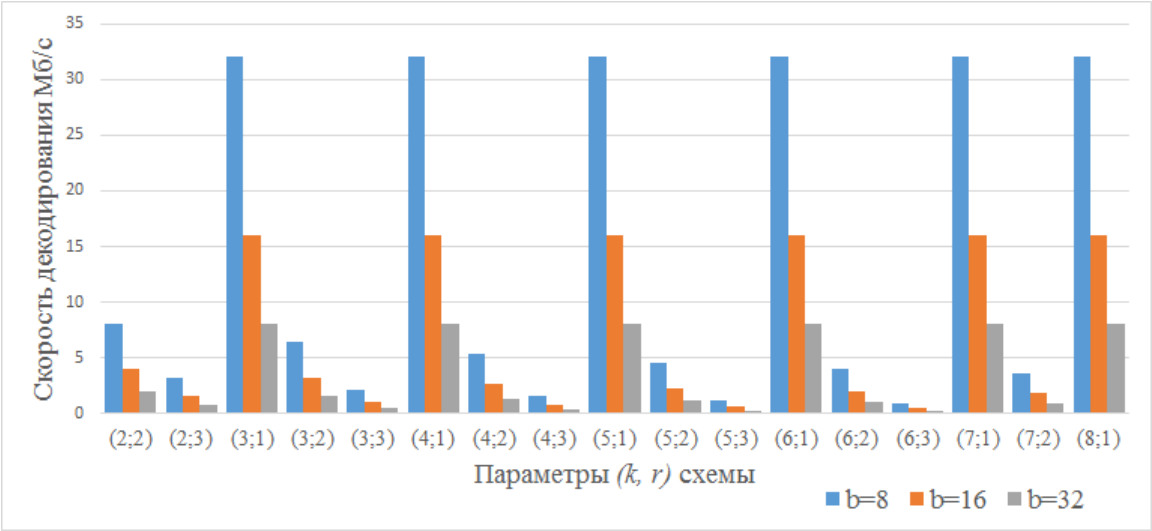

Рисунок 4.5 — Скорость декодирования данных (Мб/с) при различных параметрах $(k,\ r)$ избыточной СОК для $b = 8, 16, 32$

4.2 Обнаружение и коррекция ошибок, управление арифметическими операциями в системе остаточных классов

Используя теорему 2, мы предлагаем новый метод декодирования данных на основе аппроксимации ранга и кодов исправления ошибок (AR-ECC).

Из уравнения (3.22), следует, что $\sum_{i=1}^{n} k_i x_i$ можно представить в виде:

$$\sum_{i=1}^{n} k_i x_i = X - r_X P. \tag{4.1}$$

Предположим, что во время вычисления $E \xrightarrow{\text{ИСОК}} (e_1, e_2, \ldots, e_n)$ произошла ошибка, и пользователь получил значение $X + E$ вместо X. Тогда по формуле (4.1), имеем:

$$\sum_{i=1}^{n} k_i \left(x_i + e_i\right) = X + E + r_X P + r_E P.$$

Без ограничения общности мы предполагаем, что множество модулей избыточной СОК находятся в порядке возрастания, $p_1 < p_2 < \cdots < p_n$. Так как в ИСОК k-модули включены в динамический диапазон, а r является избыточными , где $k + r = n$, то

$$X < \prod_{i=1}^{k} p_i = R.$$

Значение $\left\lfloor \frac{E}{R} \right\rfloor$ равно $\left\lfloor \frac{X+E}{R} \right\rfloor$ или $\left\lfloor \frac{X+E}{R} \right\rfloor - 1$. Если $\left\lfloor \frac{E+X}{R} \right\rfloor = 0$, то $E = 0$. Поэтому мы можем использовать значение $\left\lfloor \frac{E+X}{R} \right\rfloor$, чтобы определить, правильный ли результат, если или нет ошибки.

В связи с тем, что ошибка имеет вид: $E = \beta P_I$, где $P_I = \prod_{i \in I} p_i$, β целое число в интервале $\left[0, /, \frac{P}{P_I - 1}\right]$, а I – множество модулей избыточной СОК, которые не имеют ошибки. Значение $\left\lfloor \frac{E}{R} \right\rfloor$ может использоваться как синдром ошибки, где каждое значение $\left\lfloor \frac{E}{R} \right\rfloor$ однозначно определяет E и I.

Мы производим предварительные вычисления: сортируем значения всех возможных ошибок $\left\lfloor \frac{E}{R} \right\rfloor$ в порядке возрастания и сопоставляем с E. Если мы используем двоичный поиск в отсортированном массиве значений $\left\lfloor \frac{E}{R} \right\rfloor$, мы находим E и устанавливаем I логарифмический размер временного массива.

Пусть $X' = X + E$ и $X' \xrightarrow{\text{ИСОК}} (x'_1, x'_2, \ldots, x'_n)$. Используя уравнение (3.22) вычислим:

$$\left\lfloor \frac{X'}{R} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^{n} \left| P_i^{-1} \right|_{p_i} \cdot P_i \cdot x'_i - r_{X'} P}{R} \right\rfloor \tag{4.2}$$

Так как P/R является целым числом, то согласно свойству целое число может быть выведено как общий коэффициент, то уравнение (4.2) можно переписать:

$$\left\lfloor \frac{X'}{R} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^{n} \left| P_i^{-1} \right|_{p_i} \cdot P_i \cdot x'_i}{R} \right\rfloor - \frac{r_{X'} P}{R} \tag{4.3}$$

Пусть $P/R = M$ и $0 \leqslant \left\lfloor \frac{E}{R} \right\rfloor < M$, уравнение (4.3) эквивалентно

$$\left\lfloor \frac{X'}{R} \right\rfloor = \left| \left\lfloor \frac{\sum_{i=1}^{n} \left| P_i^{-1} \right|_{p_i} \cdot P_i \cdot x'_i}{R} \right\rfloor - \frac{r_{X'} P}{R} \right|_M = \left| \left\lfloor \frac{\sum_{i=1}^{n} \left| P_i^{-1} \right|_{p_i} \cdot P_i \cdot x'_i}{R} \right\rfloor \right|_M \tag{4.4}$$

Из определенияP_i следует, что для всех $i = \overline{k+1, n}$ значение $\frac{P_i}{R}$ является целым числом. Тогда, согласно свойству поля, целое число можно вывести как общий коэффициент, а уравнение (4.4) можно переписать:

$$\left\lfloor \frac{X'}{R} \right\rfloor = \left| \left\lfloor \frac{\sum_{i=1}^{n} \left| P_i^{-1} \right|_{p_i} \cdot P_i \cdot x'_i}{R} \right\rfloor + \sum_{i=k+1}^{n} \frac{\left| P_i^{-1} \right|_{p_i} \cdot P_i}{R} \cdot x'_i \right|_M \tag{4.5}$$

В уравнении (4.5), значение $\left\lfloor \frac{\sum_{i=1}^{n} \left| P_i^{-1} \right|_{p_i} \cdot P_i \cdot x'_i}{R} \right\rfloor$ может быть вычислено согласно теореме 2, для всех $i = \overline{k+1, n}$ значения $\frac{\left| P_i^{-1} \right|_{p_i} \cdot P_i}{R}$ являются предварительно вычисленными константами.

Пример 2. Пусть мы имеем схему $(3, 5)$ и множество модулей – $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, $p_5 = 11$. Параметры избыточной СОК равны $R = 2 \cdot 3 \cdot 5 = 30$ и $P = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 = 2310$. Существует число представленное в избыточной СОК $X = 8 \to (0, 2, 3, 1, 8)$.

Рассмотрим предложенный подход для обнаружения, локализации и исправления ошибки.

Предварительные вычисления: $M = \frac{P}{R}$,

$$P_1 = \frac{P}{p_1} = 1155, P_2 = \frac{P}{p_2} = 770,$$
$$P_3 = \frac{P}{p_3} = 462, P_4 = \frac{P}{p_4} = 330, P_5 = \frac{P}{p_5} = 210;$$
$$\left|P_1^{-1}\right|_{p_1} \cdot P_1 = 1 \cdot 1155 = 1155,$$
$$\left|P_2^{-1}\right|_{p_2} \cdot P_2 = 2 \cdot 770 = 1540,$$
$$\left|P_3^{-1}\right|_{p_3} \cdot P_3 = 3 \cdot 462 = 1386,$$
$$\left|P_4^{-1}\right|_{p_4} \cdot P_4 = 1 \cdot 770 = 330,$$
$$\left|P_5^{-1}\right|_{p_5} \cdot P_5 = 1 \cdot 210 = 210.$$

Составим таблицу ошибок в зависимости от значений $\left\lfloor \frac{W}{R} \right\rfloor$:

0: нет ошибки;

38, 39: ошибка в w_1;

25, 26, 51, 52: ошибка в w_2;

15, 16, 30, 31, 46, 47, 61, 62: ошибка в w_3;

11, 22, 33, 44, 55, 66: ошибка в w_4;

7, 14, 21, 28, 35, 42, 49, 56, 63: ошибка в w_5.

Пусть вектор ошибки E равен $E = (0, 0, 0, 1, 0)$, то получим $X' = X + E \to (0, 2, 3, 2, 8)$, мы вычислим $\left\lfloor \frac{X'}{R} \right\rfloor$ используя уравнение (4.5), и получим:

$$\left\lfloor \frac{X'}{R} \right\rfloor = \left| \left\lfloor \frac{1155 \cdot 0 + 1540 \cdot 2 + 1386 \cdot 3}{30} \right\rfloor + 11 \cdot 2 + 7 \cdot 8 \right|_{77} = 11.$$

Так как $\left\lfloor \frac{X'}{R} \right\rfloor = 11$, то ошибка находится в w_4 и $E = (0, 0, 0, 1, 0)$. Следовательно, значение X равно $X \to (0, 2, 3, 1, 8)$.

Применим теорему 2 для вычисления $\left\lfloor \frac{\sum_{i=1}^{n} \left|P_i^{-1}\right|_{p_i} \cdot P_i \cdot x_i'}{R} \right\rfloor$.

$$k_1 = \left\lceil \frac{2^3 \cdot \left|P_1^{-1}\right|_{p_1} \cdot P_1}{30} \right\rceil = 308,$$
$$k_2 = \left\lceil \frac{2^3 \cdot \left|P_2^{-1}\right|_{p_2} \cdot P_2}{30} \right\rceil = 411,$$
$$k_3 = \left\lceil \frac{2^3 \cdot \left|P_3^{-1}\right|_{p_3} \cdot P_3}{30} \right\rceil = 370.$$

$$\left\lfloor \frac{\sum_{i=1}^{k} \left|P_i^{-1}\right| \cdot P_i \cdot x_i'}{R} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^{3} k_i x_i'}{8} \right\rfloor = \left\lfloor \frac{308 \cdot 0 + 411 \cdot 2 + 370 \cdot 3}{8} \right\rfloor = 241$$

Основными арифметическими операциями в большинстве алгоритмов являются сложение, вычитание, умножение и деление. Сдвиг вправо – это операция деления на степень двух, а сдвиг влево – это операция умножения на степень двух. Например, унарная алгебраическая операция отрицания «-a» может быть представлена как двоичная алгебраическая операция с использованием выражения $0 - a$.

Если мы обозначим i-ю двоичную операцию O_i, с операндами X_i и Y_i, ее можно записать в виде: $O_i = f_i\left(X_i, Y_i\right)$.

Обозначим последовательность двоичных алгебраических операций O_i выполняемых соответственно над операндами $X_1, X_2, \ldots, X_m$ как $F\left(X_1, X_2, \ldots, X_m\right)$. Поскольку контроль результатов промежуточных вычислений является сложной задачей, а вероятность ошибок в вычислениях высока, нам нужен метод проверки результата $F\left(X_1, X_2, \ldots, X_m\right)$. Для управления результатами вычислений могут использоваться AN коды [61]. Однако они требуют избыточности информации и не могут быть адаптированы для распределенных систем.

Избыточная СОК позволяет производить контроль вычислений с меньшей избыточностью информации [42].

Пусть истинный результат $F\left(X_1, X_2, \ldots, X_m\right)$ имеет вид: $T \to (t_1, t_2, \ldots, t_n)$. Предположим, что полученный результат равен $A \to (a_1, a_2, \ldots, a_n)$. Если для всех $i = \overline{1, n}$, $t_i = a_i$, то полученный результат является верным, в противном случае – неверным. Чтобы проверить правильность результата $F\left(X_1, X_2, \ldots, X_m\right)$, мы используем метод масштабирования и расширения базы в СОК.

Например, из первых $n-1$ вычетов $(a_1, a_2, \ldots, a_{n-1})$ мы вычисляем $\overline{a_n}$. Если $\overline{a_n} = a_n$, то результат $F(X_1, X_2, \ldots, X_m)$ верный, в противном случае – неверный.

Чтобы найти истинный результат, мы используем метод обнаружения и локализации ошибок рассмотренный выше. Следовательно полученные результаты не имеют ошибок, которые не могут быть обнаружены при помощи СОК.

Ошибка может быть исправлена не только в случае перекрытия ошибок модулей при вычислениях по одному модулю, но и в случае единственного отказа.

Теорема 3. *Пусть $p_1, p_2, \ldots, p_n$ – множество модулей СОК, $X \to (x_1, x_2, \ldots, x_n)$ и $Y \to (y_1, y_2, \ldots, y_n)$ – числа представленные в СОК, и их ранги соответственно равны r_X и r_Y, то имеет место равенство:*

$$r_{X+Y} = r_X + r_Y - \sum_{x_i+y_i \geqslant p_i} \left|P_i^{-1}\right|_{p_i}. \tag{4.6}$$

Доказательство. Из определения ранга r_{X+Y} следует

$$r_{X+Y} = \left\lfloor \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i} |x_i + y_i|_{p_i}}{p_i} \right\rfloor \tag{4.7}$$

Учитывая, что для всех $\forall i \in \overline{1,n}$ справедливо равенство

$$|x_i + y_i|_{p_i} = \begin{cases} x_i + y_i - p_i, \text{ если } x_i + y_i \geqslant p_i, \\ x_i + y_i, \text{ иначе.} \end{cases}$$

тогда уравнение (4.7) можно переписать в виде:

$$r_{X+Y} = \left\lfloor \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i} x_i}{p_i} + \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i} y_i}{p_i} - \sum_{x_i+y_i \geqslant p_i} \left|P_i^{-1}\right|_{p_i} \right\rfloor. \tag{4.8}$$

Так как целое число A может быть представлено как сумма целой и дробной части, то есть $A = \lfloor A \rfloor + \{A\}$, то, согласно свойству поля, целочисленная часть является общим фактором. Следовательно, уравнение (4.8) можно переписать в следующем виде:

$$\begin{aligned} r_{x+y} = \left\lfloor \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i} x_i}{p_i} \right\rfloor + \left\lfloor \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i} y_i}{p_i} \right\rfloor - \sum_{x_i+y_i \geqslant p_i} \left|P_i^{-1}\right|_{p_i} + \\ + \left\lfloor \left\{ \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i} x_i}{p_i} \right\} + \left\{ \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i} y_i}{p_i} \right\} \right\rfloor \end{aligned} \tag{4.9}$$

Согласно работе [72], $\left\{\sum_{i=1}^{n}\frac{\left|P_i^{-1}\right|_{p_i}x_i}{p_i}\right\}=\frac{X}{P}$ и $\left\{\sum_{i=1}^{n}\frac{\left|P_i^{-1}\right|_{p_i}y_i}{p_i}\right\}=\frac{Y}{P}$.

Если подставить значения $r_X=\left\lfloor\sum_{i=1}^{n}\frac{\left|P_i^{-1}\right|_{p_i}x_i}{p_i}\right\rfloor$ и $r_Y=\left\lfloor\sum_{i=1}^{n}\frac{\left|P_i^{-1}\right|_{p_i}y_i}{p_i}\right\rfloor$ в уравнение (4.9), получим:

$$r_{X+Y}=r_X+r_Y-\sum_{x_i+y_i\geqslant p_i}\left|P_i^{-1}\right|_{p_i}+\left\lfloor\frac{X}{P}+\frac{Y}{P}\right\rfloor. \tag{4.10}$$

Поскольку условие применимости СОК равно $X+Y<P$, то $\left\lfloor\frac{X}{P}+\frac{Y}{P}\right\rfloor=0$, а формула (4.10) равна уравнению (4.6).

Пример 3. Пусть у нас есть набор модулей СОК $p_1=2$, $p_2=3$, $p_3=5$, с динамическим диапазоном СОК $P=2\cdot3\cdot5=30$. Два числа в СОК представляются как: $X\to(0,2,3)$ и $Y\to(1,2,4)$. Мы рассмотрим несколько случаев использования теоремы 3 для проверки результата. Мы используем коэффициенты для данного СОК которые вычисляются в примере 1: $r_X=1$, $r_Y=1$, $\left|P_1^{-1}\right|_{p_1}=1$, $\left|P_2^{-1}\right|_{p_2}=1$, $\left|P_3^{-1}\right|_{p_3}=1$, $\left|P_1^{-1}\right|_{p_1}P_1=15$, $\left|P_2^{-1}\right|_{p_2}P_2=10$, $\left|P_3^{-1}\right|_{p_3}P_3=6$.

Случай 1. $W=x+Y\to(1,1,2)$. Вычислим сумму $\sum_{i=1}^{3}k_iw_i=4\cdot1+3\cdot1+2\cdot2=11$, тогда $R_W=\left\lfloor\frac{11}{8}\right\rfloor=1$, $W=\sum_{i=1}^{3}P_i\left|P_i^{-1}\right|_{p_i}w_i-R_WP=15\cdot1+10\cdot1+6\cdot2-30=7$. Поэтому, согласно теореме 2, получаем: $r_W^*=R_W=1$. Проверим корректность результата при помощи теоремы 3 и уравнения (4.6). Получаем: $r_W=r_X+r_Y-\left|P_2^{-1}\right|_{p_2}-\left|P_3^{-1}\right|_{p_3}=1+1-1-1=0$.

Поскольку $r_W^*\neq r_W$, значит есть ошибка. В этом случае $X=8$ и $Y=29$, а $W=X+Y=8+29=37$. Произошло переполнение динамического диапазона СОК, что и привело к обнаружению ошибки.

Случай 2. $W=X+X+E\to(1,1,1)$, где E – вектор ошибок, равный $E\to(1,0,0)$. Вычислим сумму $\sum_{i=1}^{3}k_iw_i=4\cdot1+3\cdot1+2\cdot1=9$, тогда $R_W=\left\lfloor\frac{9}{8}\right\rfloor=1$, $W=\sum_{i=1}^{3}P_i\left|P_i^{-1}\right|_{p_i}w_i-R_WP=15\cdot1+10\cdot1+6\cdot1-30=1$.

Поэтому, согласно теореме 2, получаем $r_W^*=R_W=1$. Мы проверяем корректность результата с теоремой 3 и уравнением (4.6). Получаем: $r_W=r_X+r_Y-\left|P_2^{-1}\right|_{p_2}-\left|P_3^{-1}\right|_{p_3}=1+1-1-1=0$. Поскольку $r_W^*\neq r_W$, возникает ошибка.

Случай 3. $W=X+X\to(0,1,1)$. Вычислим сумму $\sum_{i=1}^{3}k_iw_i=4\cdot0+3\cdot1+2\cdot1=7$, тогда $R_W=\left\lfloor\frac{7}{8}\right\rfloor=0$, $W=\sum_{i=1}^{3}P_i\left|P_i^{-1}\right|_{p_i}w_i-R_WP=15\cdot0+$

$10 \cdot 1 + 6 \cdot 1 = 16$. Согласно теореме 2 получаем: $r_W^* = R_W = 0$. Мы проверяем корректность результата с помощью теоремы 3 и формулы (4.6). Получаем: $r_W = r_X + r_Y - \left|P_2^{-1}\right|_{p_2} - \left|P_3^{-1}\right|_{p_3} = 1+1-1-1 = 0$. Так как $r_W^* = r_W$, результат правильный.

Как показано в примере 3, теорему 3 можно использовать для проверки результата арифметических операций даже при минимальной избыточности (n, n). Однако в случае ошибки она не может быть исправлена. В примере 2 рассматривается подход, который не только обнаруживает ошибку, но и исправляет ее.

4.3 Параметры конфигурации системы моделирования надежного хранения данных в облачной среде

Способы обнаружения сбоев в распределенных хранилищах данных и средствах связи обычно основаны на кодах исправления ошибок, кодах стирания и кодах восстановления, а так же их модификациях.

В отличие от существующих методов, коды исправления ошибок в СОК могут эффективно обнаруживать, исправлять ошибки и контролировать вычисления. Они обладают надежностью и сохраняют целостность данных, что делает их применимыми для хранения больших данных и обработки данных.

Однако коды коррекции ошибок в СОК имеют один существенный недостаток: высокая сложность обнаружения и локализация ошибок. Чтобы решить эту проблему, мы предлагаем приближенный метод вычисления рангов чисел, сортировки массива относительных значений и использования двоичного поиска.

Предлагаемый подход AR-ECC снижает сложность от линейного до логарифмического значения мощности множества всех возможных чисел в СОК.

Наша модель является настраиваемой. Для определения и конфигурации параметров необходимо учитывать следующие критерии оптимизации: точность, масштабируемость, надежность, конфиденциальность, безопасность и производительность.

Точность. Точность вычислений зависит от параметра N. Точность $N \geqslant \log_2(\rho \cdot P)$ позволяет точно вычислить ранг числа [43]. Однако, соглас-

но теореме 2, точность $N = \lceil \log_2(\rho) \rceil$ позволяет эффективно аппроксимировать ранг числа с отклонением менее 1 от истинного результата.

Масштабируемость и надежность. Если возникает ошибка, мы используем свойство масштабируемости облака для поддержания определенного уровня надежности. Мы восстанавливаем или исправляем потерянные фрагменты данных, хранящиеся в одном или нескольких облаках, используя AR-ECC. Мы определяем параметры системы хранения для желаемой надежности с использованием подхода, описанного в разделе **??**, и объема памяти, необходимого для хранения данных с использованием коэффициента резервирования из раздела **??**.

Конфиденциальность и безопасность. Чтобы сделать схему асимптотически идеальной, Barzu [24] предложил использовать алгоритм Asmuth-Bloom, обеспечивающий высокую степень защиты данных. Однако этот алгоритм неприменим для распределенного хранилища, поскольку он требует избыточности хранения данных, как в схеме Шамира.

В СОК вычислительная безопасность системы зависит от параметров k, n, b_i и α_i. Его можно оценить по формуле $\frac{V}{V_{\bar{I}}}$, где V – объем данных, а $V_{\bar{I}}$ – максимальный объем данных, которые могут быть пропущены неавторизованному пользователю.

Например, если для всех $i = \overline{1, n}$ выполняется условие $b_i = const$, и существует сговор $k - 1$ облаков, то мощность множества всех возможных комбинаций входных данных больше или равна 2^{b-1}.

Представление данных. Важной проблемой является минимизация вычислительных затрат, связанных с реализацией арифметических операций. Для выбора модулей СОК, мы должны учесть, что модули являются попарно взаимно простыми. модули равны $2^{b_i} \pm \alpha_i$, где b_i соответствует количеству сохраненных и обработанных данных в i-м облаке, α_i является целым числом в двоичном представлении с фиксированным числом бит, равным «1». Подход из раздела **??** оценивает скорость кодирования и декодирования данных. Если в системе хранения данных нет ошибок, скорость декодирования значительно возрастает, потому что нет необходимости выполнять дорогостоящую операцию поиска ошибки.

В разделе 4.2 описано, как увеличить скорость декодирования данных при помощи оптимизации метода поиска ошибки с использованием AR. Для вычисления уравнения (4.5), мы используем КТО. Требуется, примерно, $M^2 \approx r^2 \cdot b^2$

– битные операции, где $r = n - k$. Так как значение уравнения (4.5) равняется M, то чтобы обнаружить и локализовать ошибку нам нужны $\log_2 M \cdot r \cdot b$ битовых операций. Чтобы кодировать 1 Мб данных нам необходимо $2^{23}\frac{\log_2 M \cdot r \cdot b + r^2 \cdot b^2}{k \cdot b} = 2^{23}\frac{\log_2 (r \cdot b) \cdot r + r^2 \cdot b}{k}$ битовых операции. Следовательно, скорость декодирования будет составлять

$$V_D = \frac{2^{30} \cdot k}{2^{23} \cdot (\log_2 (r \cdot b) \cdot r + r^2 \cdot b)} = \frac{2^7 \cdot k}{\log_2 (r \cdot b) \cdot r + r^2 \cdot b}.$$

Предложенный метод увеличивает скорость декодирования данных с

$$V_D = \frac{2^7}{C_n^{k+1} \cdot b \cdot k} \quad \text{до} \quad V_D = \frac{2^7 \cdot k}{\log_2 (r \cdot b) \cdot r + r^2 \cdot b}$$

Чтобы обнаружить и исправить хотя бы одну ошибку, k должно удовлетворять неравенству $k \leqslant n - 1$ на рисунке 4.6 показывается скорость декодирования данных в зависимости от параметров, которые удовлетворяют этому условию.

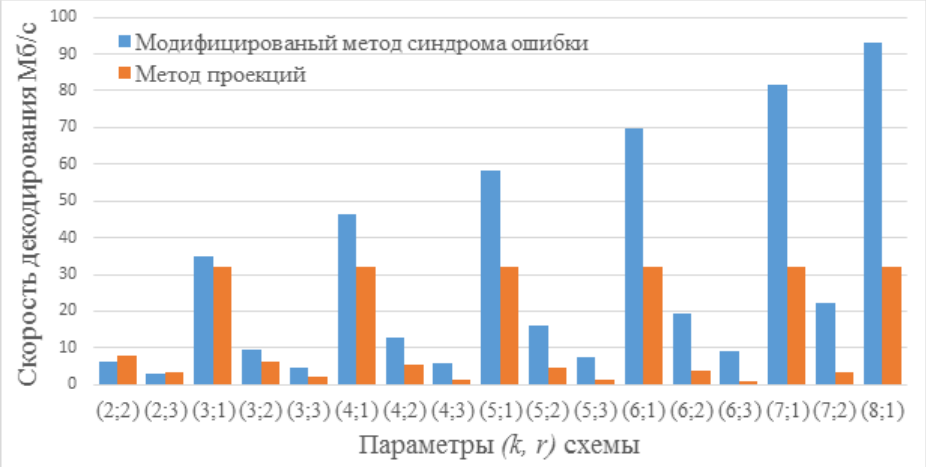

Рисунок 4.6 — Скорость декодирования данных (Мбит/с) на основе AR-ECC и избыточной СОК в худшем сценарии восстановления данных с максимальным количеством ошибок в сравнении с настройками RRNS (k, r) для $b = 8$.

Чтобы увеличить скорость кодирования и декодирования данных, мы можем использовать многоядерные процессоры или несколько виртуальных машин. Если мы выберем значения b_i кратным машинному слову, можно использовать эффективную реализацию модульной арифметики с распределенными операциями и нейронной сетью конечного кольца. Это уменьшает сложность нахождения остатка деления от квадратичного до линейного.

4.4 Расчет надежности алгоритмов хранения данных в облаках их сравнительная оценка

Произведем расчет надежности хранения данных для различных систем.

Данные по отказам в работе жестких дисков полученных от компании «Backblaze» за 3 года (2014–2016), были проанализированы и обработаны. По каждому году были проведены расчеты возможность отказа рисунок 4.7, 4.8, 4.9.

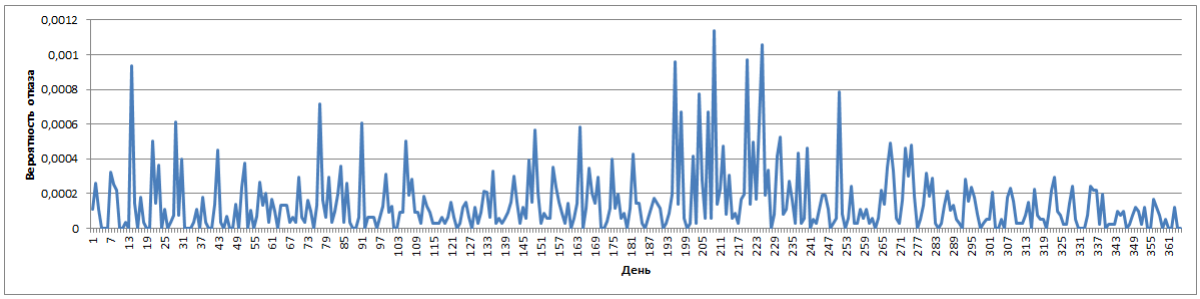

Рисунок 4.7 — Вероятности отказа жестких дисков за 2014 год

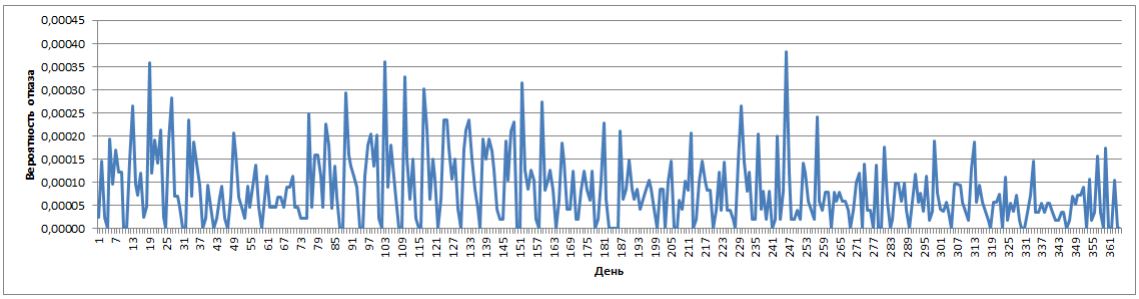

Рисунок 4.8 — Вероятности отказа жестких дисков за 2015 год

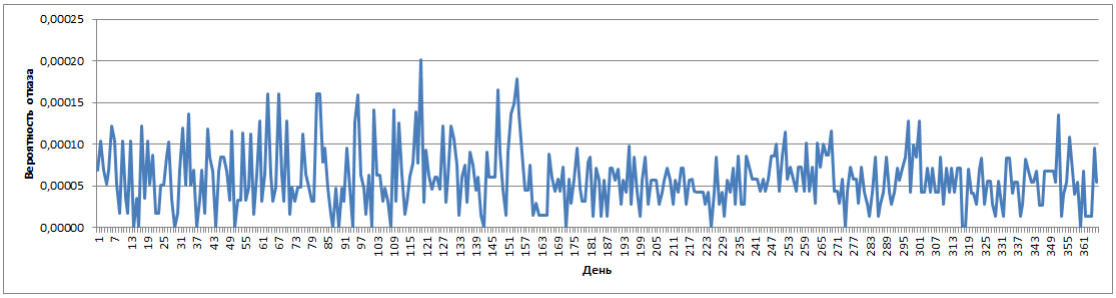

Рисунок 4.9 — Вероятности отказа жестких дисков за 2016 год

По расчитанным данным отказов, путем проведения интерполирования и усреднения полученных данных по каждому году, была построена система для моделирования отказов облачных систем. На основе разработанной системы бы-

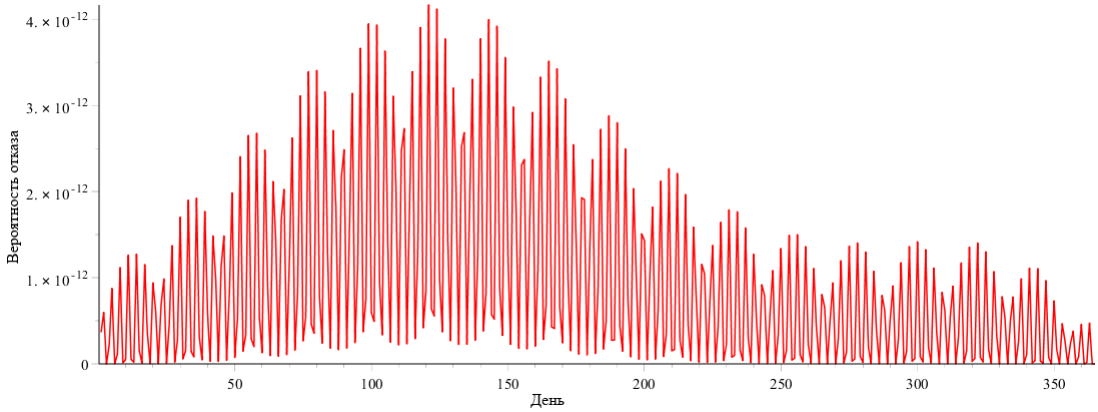

Рисунок 4.10 — Вероятности отказа жестких дисков в системе хранения данных BigTable [35]

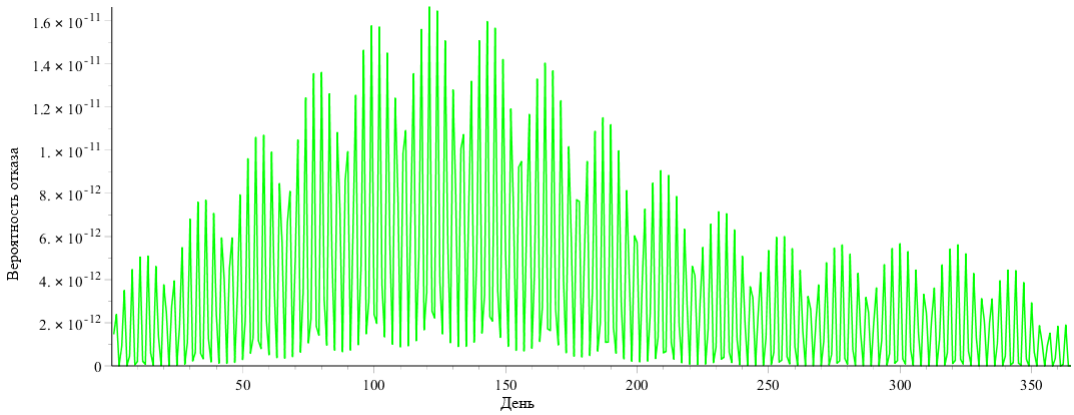

Рисунок 4.11 — Вероятности отказа жестких дисков в системе хранения данных DepSky [25]

ло проведено моделирование надежности хранения данных в следующих системах:

1. однослойная (8, 2);
2. двухслойная, 1 слой (6, 3), 2 слой (6);
3. двухслойная 1 слой (6, 2), 2 слой (6, 3);
4. BigTable [35];
5. DepSky [25];
6. RACS ((2, 1), (4, 1), (5, 2)) [15];

Результаты моделирования системы хранения в выше перечисленных системах представлены на рисунках 4.10–4.17.

По результатам проведенного моделирования можно сделать вывод о том, что система построенная на основе СОК (рисунок 4.15, 4.16) позволяет хранить данные с большей надежностью в сравнении с представленными системами.

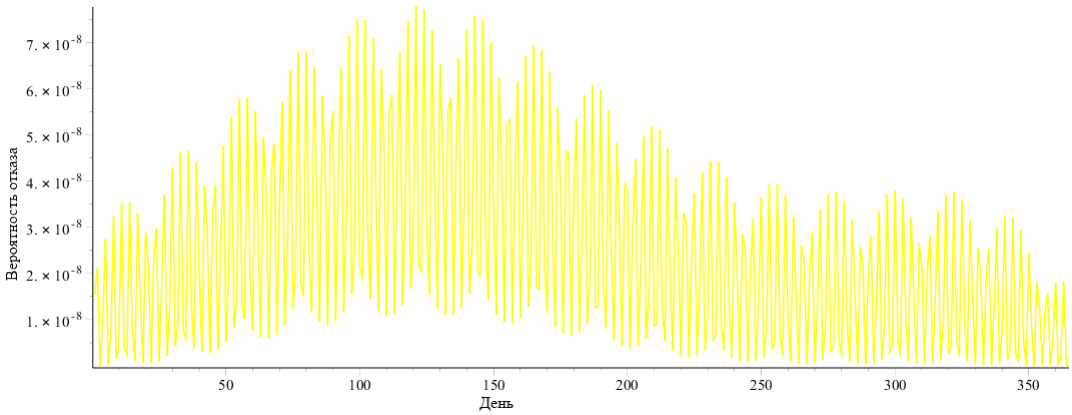

Рисунок 4.12 — Вероятности отказа жестких дисков в системе хранения данных RACS(2, 1) [15]

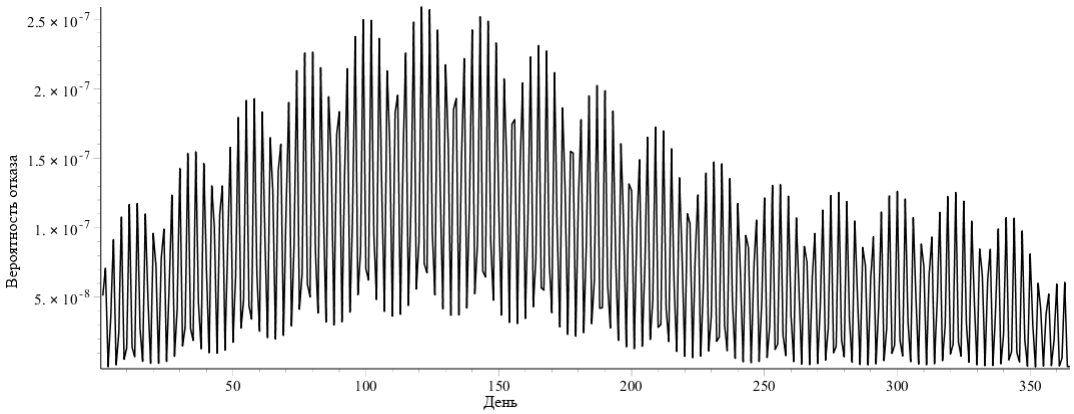

Рисунок 4.13 — Вероятности отказа жестких дисков в системе хранения данных RACS(4, 1) [15]

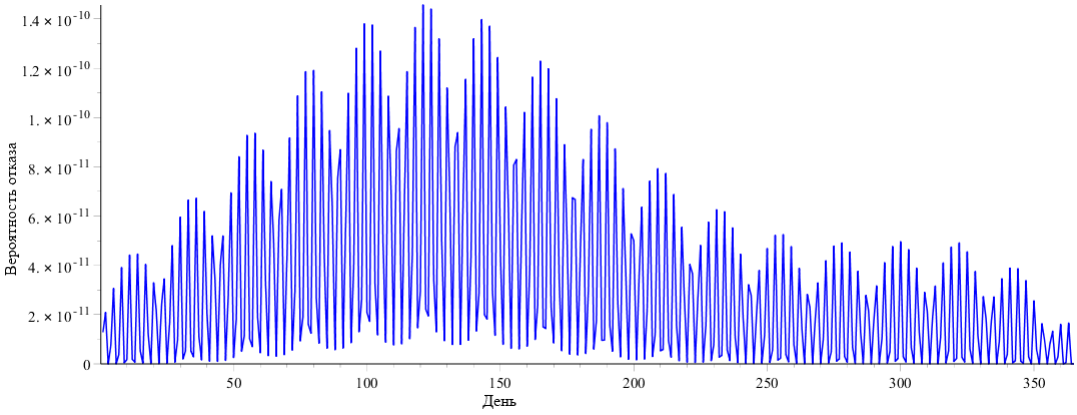

Рисунок 4.14 — Вероятности отказа жестких дисков в системе хранения данных RACS(5, 2) [15]

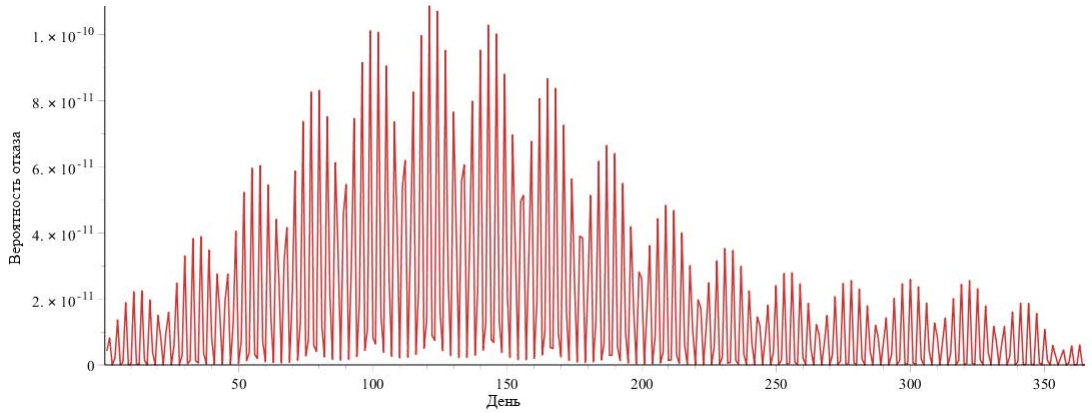

Рисунок 4.15 — Вероятности отказа жестких дисков в системе хранения данных использующую СОК (1-й слой (6, 3), 2-й слой 6)

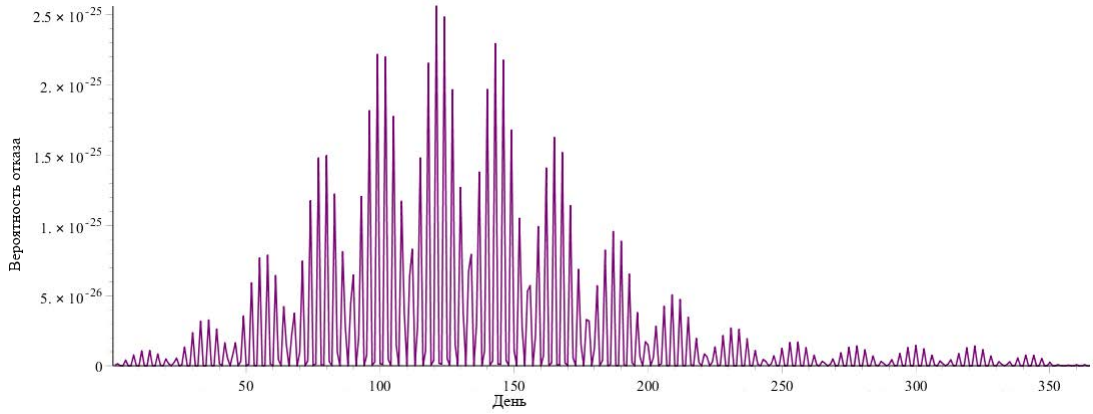

Рисунок 4.16 — Вероятности отказа жестких дисков в системе хранения данных использующую СОК (1-й слой (6, 2), 2-й слой (6, 3))

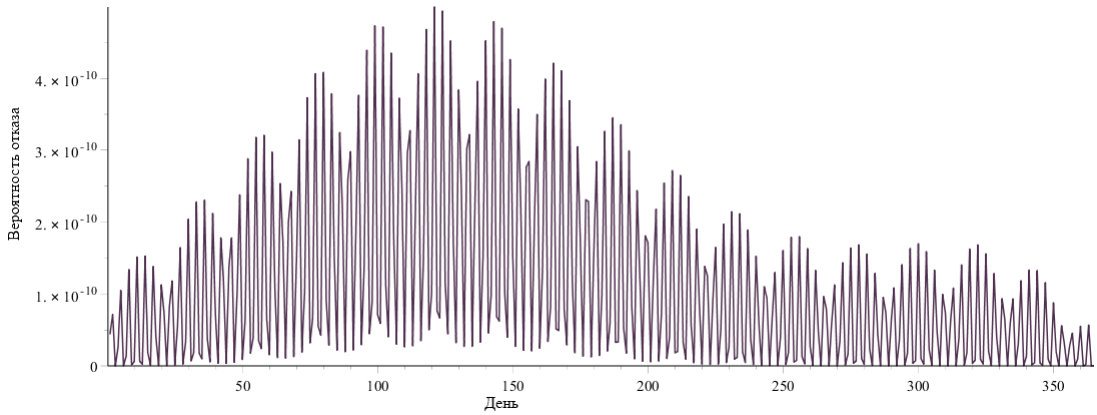

Рисунок 4.17 — Вероятности отказа жестких дисков в системе хранения данных использующую СОК (8, 2)

При построении математической модели надежности облака, учитываются следующие свойства СОК:

– независимость частей данных, что позволяет независимо обрабатывать части данных и считать их независимыми элементами;
– равноправность частей данных, что дает возможность избыточные каналы считать элементами резерва для остальных;
– малоразрядность, что обуславливает несущественный разброс надежностных характеристик облаков.

Для анализа надежности функционирования облака на основе СОК целесообразно использовать математические соотношения теории надежности и различные способы введения структурной избыточности. При вычислении надежностных характеристик предполагается, что

– в начальный момент времени имеются $n = k + r$ облаков, где k – количество рабочих, r – количество контрольных облаков;
– минимальное количество облаков (частей), необходимых для правильного восстановления данных равно $n - (r + 1)$; отказ более чем $r - 1$ облаков рассматривается как отказ всей облачной системы;
– отказы облаков являются статистически независимыми событиями.

Задача построения математической модели надежности может быть сформулирована следующим образом.

Данные представленные в СОК и передающиеся в облако для обработки и хранения конвертируются при помощи k рабочих и r контрольных оснований. При потери или недоступности рабочих частей, они могут быть заменены контрольными.

Требование по обеспечению гарантированной защиты от выдачи недостоверного результата обуславливает необходимость сохранения в работоспособном состоянии хотябы одного контрольного и k рабочих частей.

Таким образом, надежная структура облачного хранения и обработки данных будет соответствовать способу скользящего резервирования, при котором резервные элементы находятся в нагруженном состоянии. При принятии допущений о простейшем потоке отказов элементов и их равной надежности (описанной в), учитывая, что отказавшие элементы не восстанавливаются приходим к выражению для расчета вероятности безотказной работы следующего вида:

$$R_1 = \sum_{i=0}^{r-1} P'^{k+r-i} \left(1 - P'\right)^i, \tag{4.11}$$

где R_1 – вероятность безотказной работы облачной системы, на основе одноуровневой СОК, в течении заданного времени, P – вероятность безотказной работы одного облака.

$$R_2 = \sum_{i=0}^{r-1} P'^{k+r-i} \left(1 - P'\right)^i, \tag{4.12}$$

$$R_{1,2} = R_1 \cdot R_2, \tag{4.13}$$

где $R_{1,2}$ – вероятность безотказной работы облачной системы, на основе двухуровневой СОК, в течении заданного времени, P – вероятность безотказной работы одного облака.

При распределении времени функционирования облака до отказа по экспоненциальному закону вероятность безотказной работы в течении заданного времени будет определяться как:

$$P'(t) = e^{-\lambda_{TP} \cdot t}. \tag{4.14}$$

Тогда

- интенсивность отказов можно считать постоянной величиной;
- кроме интенсивности отказов отсутствуют другие исходные данные для расчета;
- полученные результаты пригодны для инженерных оценок отказоустойчивости.

Обозначим интенсивность отказа одного облака λ_P, и выбрав в качестве среднего числа разрядов в конкретной системе

$$\gamma_{cp} = \left[\sum_{i=1}^{n} \frac{\gamma_i}{n} \right], \tag{4.15}$$

где n – число облаков (оснований СОК) в системе, γ_i – количество разрядов в n-м канале, $[\,]$ – целая часть числа, определим

$$\lambda_{TP} = \gamma_{cp} \cdot \lambda_{P'}, \tag{4.16}$$

где λ_{TP} – интенсивность отказов одного облака.

Расчетные данные для системы с различным соотношением числа рабочих и контрольных оснований (рисунки 4.18, 4.19) свидетельствуют о существенном выигрыше в надежности при использовании резерва.

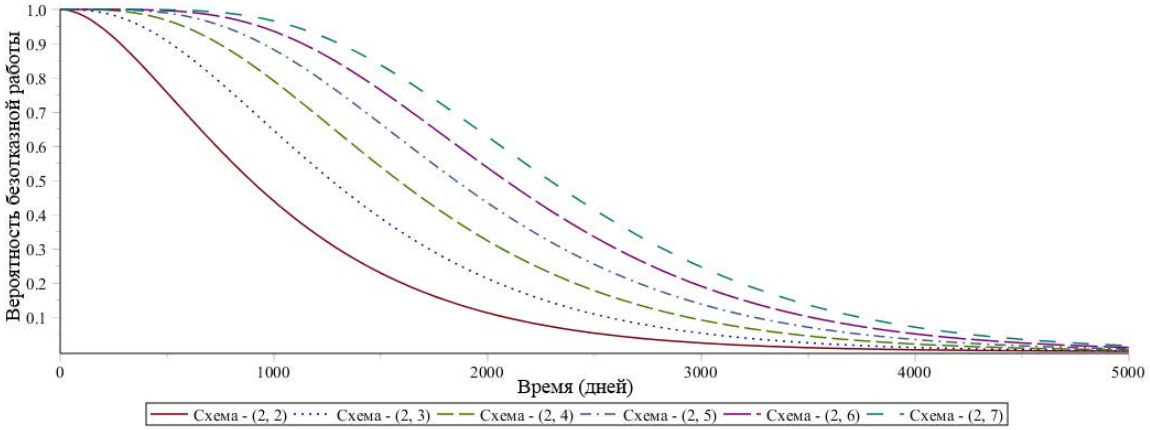

Рисунок 4.18 — Зависимость вероятности безотказной работы системы от соотношения числа рабочих (первая цифра в скобках) и контрольных (вторая цифра) облаков при интенсивности отказов $\lambda = 8 \cdot 10^{-5}$

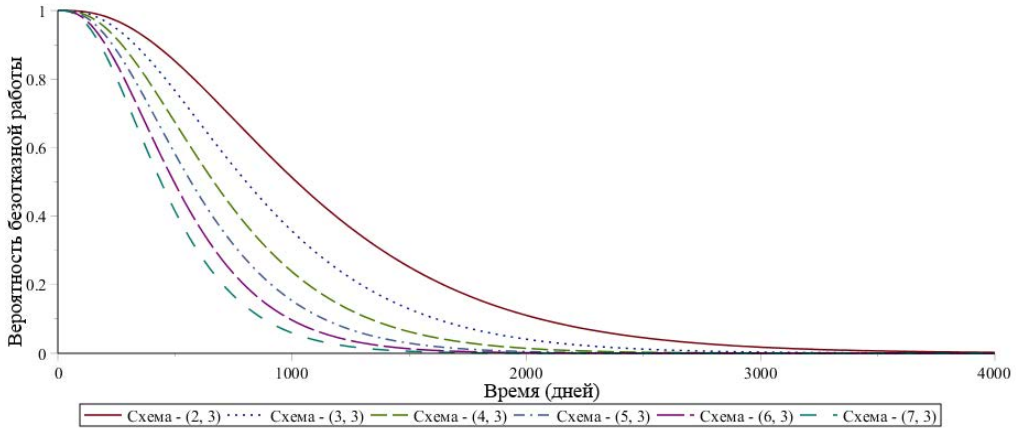

Рисунок 4.19 — Зависимость вероятности безотказной работы системы от соотношения числа рабочих (первая цифра в скобках) и контрольных (вторая цифра) облаков при интенсивности отказов $\lambda = 8 \cdot 10^{-5}$

Наибольший интерес представляет сравнение по выбранному показателю систем функционирующих на базе СОК, ПСС, систем «Google» и «ClaverSalfe», и обеспечивающие равные диапазоны представления чисел. Тогда при интенсивности отказов одного разряда, равной λ_P, интенсивность отказов ПСС

$$\lambda_{nc} = \gamma_{nc}\lambda_P, \tag{4.17}$$

где $\lambda_{nc} = 32$ – число разрядов позиционного процессора.

Определение и оперативная коррекция ошибочных результатов с помощью позиционных систем возможны лишь, при условии одновременной работы нескольких вычислительных устройств по принципу голосования; для сравнения выбирается мажоритарная вычислительная структура, функционирующая по принципу «2» из «3» и обеспечивающая маскирование одиночных сбоев и

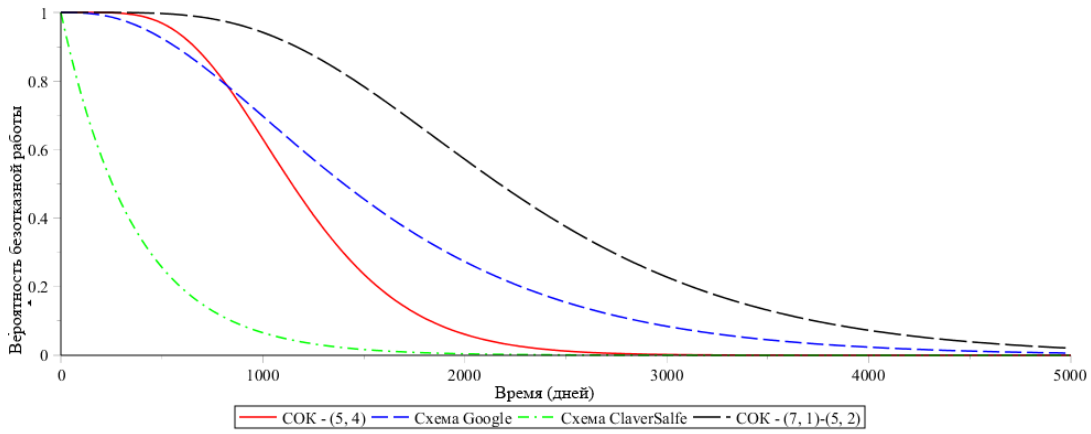

Рисунок 4.20 — График сравнительной оценки вероятности безотказной работы системы построенной с использованием различных алгоритмов

отказов. Вероятность безотказной работы такой системы без учета надежности элемента голосования задается выражением [8]

$$P_{\text{ПСС}} = 3P'^{2}_{\text{ПСС}} - 2P'^{3}_{\text{ПСС}}, \tag{4.18}$$

где $P_{\text{ПСС}}$ – вероятность безотказной работы позиционного процессора.

На рисунке 4.20 приведены графики различных систем обработки и хранения данных в облачной среде. Наилучшими характеристиками вероятности безотказной работы является система построенная на двухуровневой СОК, которая позволяет сохранять высокую надежность, через 1000 дней после начала работы она составляет 94%.

Теперь определим величину M средней наработки на отказ облачной системы, функционирующей в СОК (время за которое выйдут из строя γ облаков), а так же ожидаемое время M_1 первого отказа в данном облаке:

$$M = \sum_{i=n-r+1}^{n} \frac{1}{\lambda_{TP} \cdot i}; \quad M_1 = \frac{1}{\lambda_{pm}}. \tag{4.19}$$

При этом можно использовать нормированное значение M:

$$m = \frac{M}{M_1} = \sum_{i=n-r+1}^{n} \frac{1}{i} \tag{4.20}$$

Анализ полученных зависимостей свидетельствует о преимуществе в надежности алгоритмов хранения и обработки данных на основе СОК перед существующими системами при существенном выигрыше в избыточности. Таким

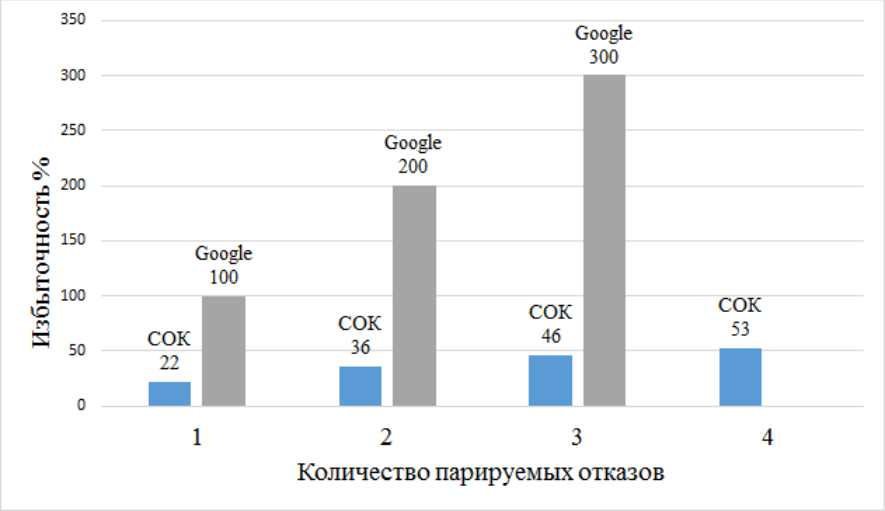

Рисунок 4.21 — Соотношение избыточности оборудования позиционных и модулярных алгоритмов для различного числа парируемых отказов

образом, при использовании алгоритмов обработки и хранения данных на основе СОК можно получить значительное увеличение надежности системы без дополнительных аппаратурных затрат при снижении избыточности.

Оценка надежности алгоритмов хранения и обработки данных в облачной среде построенных на базе СОК и сравнение их с аналогичными по качественным функциональным характеристикам существующих позиционными системами свидетельствует об их существенном преимуществе, что объясняется наличием эффекта поэлементно скользящего резервирования. Соотношение избыточного оборудования позиционных и модулярных схем обработки и хранения данных для различных параметров парируемых отказов.

При длительной работе облачной системы важно знать число отказов, ожидаемых за конкретный период. Эти данные необходимы для выбора наилучшей конфигурации мультиоблачной системы. Для определения числа случайных событий за некоторый период времени применим распределение Пуассона. При этом полный список вероятностей имеет следующий вид: вероятность нуля отказов – $e^{-\lambda \cdot t}$, вероятность одного отказа – $\lambda \cdot t^{-\lambda \cdot t}$, двух отказов – $\frac{(\lambda \cdot t)^2}{2!} \cdot e^{-\lambda \cdot t}$, трех отказов – $\frac{(\lambda \cdot t)^3}{3!} \cdot e^{-\lambda \cdot t}$, n отказов – $\frac{(\lambda \cdot t)^n}{n!} \cdot e^{-\lambda \cdot t}$. Для получения количества месяцев с 1, 2, ... отказами, необходимо умножить указанные вероятности на число периодов T на рассматриваемом отрезке времени t. При расчете по распределению Пуассона имеют место те же ограничения, что и для экспоненциального распределения. В таблице 11 приведены расчеты распределения отказов за 36 месяцев и вероятности их появления и не появления, а на рисунке **??** – диаграммы распределения отказов за определенный период времени.

Таблица 11 — Распределение отказов и вероятностей их появления

Наименование	Кол-во отказов в месяц		
	0	1	2
Количество месяцев	35.3	0.019	0.0001
Вероятность появления месяца с 0, 1, 2, и.т.д. отказами	0.98	0.69	0.0068
Вероятность не появления месяца с 0, 1, 2, и.т.д. отказами	0.02	0.31	0.9932

Анализ данных таблицы показывает, что в течении 35.3 месяца мультиоблако на базе СОК будет работать безотказно, почти весь период эксплуатации.

Будем считать, что один отказ приводит к выходу из строя одного облачного сервера; тогда количество отказов будет определяться количество ошибочных разрядов конечных вычислений. Полученные соотношения позволяют выбрать избыточный код СОК с двумя контрольными основаниями обнаруживающий две и исправляющий одну ошибку. Обнаружение ошибки должно производится после получения каждого конечного результата и следовательно не должно влиять на производительность. Коррекция ошибок и восстановление правильного результата при полученных надежностных характеристиках будут возникать достаточно редко. Проведенная оценка надежности облачной системы функционирующей на базе СОК с реконфигурируемой структурой, и сравнение с аналогичными по качественным характеристикам системам свидетельствуют об их преимуществе.

4.5 Выводы по четвертой главе

1. Разработана среда моделирования «CloudStorageSim» позволяющая производить моделирование хранения и обработки данных с реальными облачными системами у различных провайдеров. Среда моделирования позволяет в процессе моделирования вводить ошибки, искажения или утери частей данных, для проведения оценки надежности, отказоустой-

чивости и избыточности данных. Проведено моделирование и сравнение разработанных методов с существующими, которое показало превосходство разработанных методов по таким важным характеристикам: отказоустойчивость, надежность, избыточность, доступность хранимых и обрабатываемых данных в облачной среде.

2. Предложена эффективная реализация алгоритма обнаружения ошибок с использованием приближенного метода. Разработана новая конфигурируемая схема хранения данных, основанная на системе остаточных классов, кодах исправления ошибок и схемах распределения данных. Мы предоставляем теоретическую основу для расчета вероятности потери информации, избыточности данных, скорости кодирования/декодирования и параметров конфигурации для соответствия различным объектным предпочтениям, рабочим нагрузкам и облачным свойствам. Мы показали, как предлагаемая схема позволяет настроить надежность, избыточность и сократить накладные расходы на хранение данных путем выбора параметров СОК.

   На основе предложенной аппроксимации ранга мы разработали новый метод декодирования данных AR-ECC, который уменьшает сложность с $O\left(L^2\right)$ до $O\left(L \cdot \log L\right)$ и размер коэффициентов от $\lceil \log\left(\rho \cdot P\right) \rceil$ до $\lceil \log\left(\rho\right) \rceil$ бит. Используя свойства приближенного значения и арифметических свойств СОК, мы разработали новый метод обнаружения ошибок, коррекции и контроля результатов вычислений. Через 3 года вероятность отказа облачной системы, построенной на основе двухуровневой СОК будет в 5,1 раза меньше чем у системы построенной на базе GFS.

Заключение

В диссертации проведено решение актуальной научной задачи по разработке математических методов моделирования, хранения и обработки данных большой разрядности с высокой надежностью в облачной среде на основе системы остаточных классов. Решена актуальная задача повышения надежности обрабатываемых и хранимых данных в облаках за счет использования системы остаточных классов и схем разделения данных.

Проведенное в диссертации исследование, а также результаты экспериментов и моделирование дали возможность получить следующие основные научные и практические результаты.

1. В ходе анализа математических методов хранения и обработки больших данных, применяемых в облачных системах, установлена возможность реализации данных операций в модулярном базисе, что обеспечивает повышенную производительность и отказоустойчивость разрабатываемых на их основе алгоритмов.
2. Особенность СОК заключается в реализации высокопроизводительных алгоритмов в сочетании с возможностью контроля и исправления ошибок. Многие ученые отмечают высокую эффективность системы остаточных классов при реализации процедур, применяемых в облачных системах хранения и обработки данных. В первую очередь это связано с представлением данных в непозиционной форме, что позволяет обрабатывать части параллельно и независимо. Такое представление дает преимущество в отказоустойчивости и при правильной реализации ведет к уменьшению нагрузки на серверы.
3. Показано, что применение модулярной арифметики, широко развиваемой и применяемой в настоящее время, позволяет производить построение надежных и отказоустойчивых структур хранения и обработки больших данных. Рассмотрены основные методы распределенного хранения и обработки данных и показаны преимущества совместного использования системы остаточных классов и схем распределения данных для повышения отказоустойчивости и надежности. Разработка методов надежного и отказоустойчивого хранения больших данных в облачной среде

на основе системы остаточных классов и схем распределения данных является основной задачей исследования.

4. Разработана архитектура мультиоблачной системы хранения и обработки больших данных основанная на принципах модулярной арифметики. Предложенная модификация облачной системы хранения и обработки данных позволяет повысить надежность и отказоустойчивость хранимых и обрабатываемых данных.
5. Разработаны одноуровневые и двухуровневые модели надежного хранения больших данных и по результатам моделирования проведенный сравнительный анализ показал преимущество моделей построенных на базе модулярной арифметики над моделями основанных на позиционной системе счисления.
6. Для повышения отказоустойчивости разработанных методов при отказе облачного сервера, применяется перераспределение обрабатываемых данных между доступными серверами. Введение небольшой избыточности позволяет производить обработку или восстановление хранимых данных в случае отказа контрольных серверов. Разработанные методы надежного и отказоустойчивого хранения построенные на базе двухуровневой системы остаточных классов, которые имеют преимущество в надежности, отказоустойчивости и избыточности хранимых данных.
7. Разработан метод декодирования данных из системы остаточных классов в позиционную систему счисления. Предложен новый алгоритм вычисления ранга числа на основе приближенного метода. Разработана новая вычислительная архитектура для перевода чисел из системы остаточных классов в позиционную систему счисления, основанная на вычислении ранга числа с использованием приближенного метода.
8. Разработана модель хранения построенная на основе системы остаточных классов. Данная модель позволяет производить восстановление хранимых и обрабатываемых данных в случае выхода из строя одного или нескольких облачных серверов. Использование многоуровневых моделей обработки данных на системе остаточных классах позволяет строить модели надежнее одноуровневых моделей и моделей основанных на позиционная система счисления.
9. Разработана среда моделирования «CloudStorageSim» позволяющая производить моделирование хранения и обработки данных с реальными

облачными системами у различных провайдеров. Среда моделирования позволяет в процессе моделирования вводить ошибки, искажения или утери частей данных, для проведения оценки надежности, отказоустойчивости и избыточности данных. Проведено моделирование и сравнение разработанных методов с существующими, которое показало превосходство разработанных методов по таким важным характеристикам: отказоустойчивость, надежность, избыточность, доступность хранимых и обрабатываемых данных в облачной среде.

10. На основе предложенной аппроксимации ранга мы разработали новый метод декодирования данных AR-ECC, который уменьшает сложность с $O\left(L^2\right)$ до $O\left(L \cdot \log L\right)$ и размер коэффициентов от $\lceil \log\left(\rho \cdot P\right) \rceil$ до $\lceil \log\left(\rho\right) \rceil$ бит. Используя свойства приближенного значения и арифметических свойств системы остаточных классов, мы разработали новый метод обнаружения ошибок, коррекции и контроля результатов вычислений. Через 3 года дней вероятность отказа облачной системы, построенной на основе двухуровневой СОК будет в 5,1 раза меньше чем у системы построенной на базе GFS.
11. Разработана новая конфигурируемая схема хранения данных, основанная на системе остаточных классов, кодах исправления ошибок и схемах распределения данных. Мы предоставляем теоретическую основу для расчета вероятности потери информации, избыточности данных, скорости кодирования/декодирования и параметров конфигурации для соответствия различным объектным предпочтениям, рабочим нагрузкам и облачным свойствам. Мы показали, как предлагаемая схема позволяет настроить надежность, избыточность и сократить накладные расходы на хранение данных путем выбора параметров системы остаточных классов.

Обозначения и сокращения

КТО	Китайская теорема об остатках
ПСС	позиционная система счисления
СОК	система остаточных классов
СРД	схема распределения данных
GFS	«Google File System», файловая система Гугл использующая репликацию данных
БД	база данных

Список литературы

1. Червяков, Н.И. Новая схема хранения информации в облачной среде на основе системы остаточных классов и схем разделения секрета / Н.И. Червяков, М.Г. Бабенко, Н.Н. Кучеров, В.А. Кучуков, Н.Н. Кучукова // Современная наука и инновации. – 2017. – №4. – С. 28–34.
2. Червяков, Н.И. Алгебраические аспекты эффективной реализации методов защиты информации в облачных вычислениях с использованием системы остаточных классов / Н.И. Червяков, М.Г. Бабенко, Н.Н. Кучеров // Инфокоммуникационные технологии. – 2016. – Т.14. – №4. – С. 343-349.
3. Червяков, Н.И. Методы масштабирования модулярных чисел, используемые при цифровой обработке сигналов / Н.И. Червяков, П.А. Сахнюк, А.В. Шапошников, С.А. Ряднов // Инфокоммуникационные технологии. – 2006 – Т. 4. – С. 15–24.
4. Батура, Т.В. Облачные технологии: основные модели, приложения, концепции и тенденции развития / Т.В. Батура, Ф.А. Мурзин, Д.Ф. Семич // Программные продукты и системы. – 2014. – Т. 107. – № 3.
5. Червяков, Н.И. Применение корректирующих кодов СОК для диагностики работы модулярных процессоров / Н.И. Червяков, М.Г. Бабенко, Н.Н. Кучеров // Наука. Инновации. Технологии. – 2014. – №. 3. – С. 24 – 39.
6. Кучеров, Н.Н. Разработка архитектуры облачной системы отказоустойчивого хранения данных на основе системы остаточных классов / Н.Н. Кучеров // Современная наука и инновации. – 2018. – №3. – С. 29–35.
7. Исупов, К.С. Метод выполнения немодульных операций в системе остаточных классов на основе интервальных позиционных характеристик / К.С. Исупов // Фундаментальные исследования. – 2013. – № 4-3. – С. 566-570.
8. Червяков, Н.И. Модулярные параллельные вычислительные структуры нейропроцессорных систем / Н.И. Червяков, П.А. Сахнюк, А.В. Шапошников, С.А. Ряднов. – М.: ФИЗМАТЛИТ. – 2003. – 288 С.

9. Оцоков, Ш.А. Ускорение высокоточных вычислений за счет распараллеливания операций округления в комплексе систем счисления / Ш.А. Оцоков // Информационные технологии. – 2015. – Т. 21. – №5. – С. 352 – 356.

10. Акушский, И.Я. Машинная арифметика в остаточных классах / И.Я. Акушский, Д.И. Юдицкий. – М.: – Сов. радио, 1968. – 440 С.

11. Червяков, Н.И. Исследование эффективных методов перевода чисел из системы остаточных классов в позиционную систему счисления на FPGA / Н.И. Червяков, В.А. Кучуков, Н.Н. Кучеров, Н.Н. Кучукова // Современная наука и инновации. – 2017. – №3. – С. 46–53.

12. Медведев, А. Облачные технологии: тенденции развития, примеры исполнения/ А. Медведев // Современные технологии автоматизации. – 2013. – №. 2. – С. 6.

13. Червяков, Н.И. Эффективная реализация операции вычисления остатка от деления многоразрядных чисел на FPGA / Н.И. Червяков, А.С. Назаров, Ю.В. Черногорова, Н.Н. Кучеров // Современная Наука и Инновации. – 2018. – №1. – С. 15-22.

14. Червяков, Н.И. Нейрокомпьютеры в системе остаточных классов. Кн. 11: учеб. пособие для вузов / Н.И. Червяков, П.А. Сахнюк, А.В. Шапошников, А.Н. Макоха. – М.: Радиотехника, 2003. – 272 с.

15. Abu-Libdeh, H. RACS: a case for cloud storage diversity / H. Abu-Libdeh, L. Princehouse, H. Weatherspoon // Proceedings of the 1st ACM symposium on Cloud computing. – ACM, 2010. – P. 229-240.

16. Adya, A. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment / A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer, R.P. Wattenhofer //ACM SIGOPS Operating Systems Review. – 2002. – vol. 36. – №. SI. – P. 1-14.

17. Alonso, G. Distributed data management in workflow environments / G. Alonso, B. Reinwald, C. Mohan. – Research Issues in Data Engineering, Proceedings. Seventh International Workshop on. IEEE, 1997. – P. 82-90.

18. Amrhein, D. Cloud computing for the enterprise: Part 1: Capturing the cloud / D. Amrhein, S. Quint. – DeveloperWorks, IBM. – 2009. – vol. 8. – P. 121-126.

19. Asmuth, C. A modular approach to key safeguarding / C. Asmuth, J. Bloom. – IEEE Transactions on information theory. – 1983. – vol. 29. – №. 2. – P. 208-210.

20. Ateniese, G. Improved proxy re-encryption schemes with applications to secure distributed storage / G. Ateniese, K. Fu, M. Green, S. Hohenberger // ACM Transactions on Information and System Security (TISSEC). – 2006. – vol. 9. – №. 1. – P. 1-30.

21. Babenko, M. Development of a Control System for Computations in BOINC with Homomorphic Encryption in Residue Number System / M. Babenko, N. Kucherov, A. Tchernykh, N. Chervyakov, E. Nepretimova, I. Vashchenko // International Conference BOINC-Based High Performance Computing: Fundamental Research and Development, BOINC: FAST 2017. – 2017. – Vol. 1973. – P. 77-84.

22. Babenko, M. Unfairness correction in P2P grids based on residue number system of a special form / M. Babenko, N. Chervyakov, A. Tchernykh, N. Kucherov, M. Shabalina, I. Vashchenko, G. Radchenko, D. Murga // 2017 28th International Workshop on Database and Expert Systems Applications (DEXA), 2017. – P. 147-151.

23. Baru, C. The SDSC storage resource broker / C. Baru, R. Moore, A. Rajasekar, M. Wan // CASCON First Decade High Impact Papers. – IBM Corp., 2010. – P. 189-200.

24. Barzu, M. Compact sequences of co-primes and their applications to the security of CRT-based threshold schemes / M. Barzu, F.L. Ţiplea, C.C. Drăgan // Information Sciences. – 2013. – vol. 240. – P. 161-172.

25. Bessani, A. DepSky: dependable and secure storage in a cloud-of-clouds / A. Bessani, M. Correia, B. Quaresma, F. Andre, P. Sousa // ACM Transactions on Storage (TOS). – 2013. – vol. 9. – №. 4. – P. 12.

26. Bowers, K.D. HAIL: A high-availability and integrity layer for cloud storage / K.D. Bowers, A. Juels, A. Oprea // Proceedings of the 16th ACM conference on Computer and communications security. – ACM, 2009. – P. 187-198.

27. Brantner, M. Building a database on S3 / M. Brantner, D. Florescu, D. Graf, D. Kossmann, T. Kraska // Proceedings of the 2008 ACM SIGMOD international conference on Management of data. – ACM, 2008. – P. 251-264.

28. Buyya, R. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility / R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, & I. Brandic // Future Generation computer systems. – 2009. – vol. 25. – №. 6. – P. 599-616.

29. Buyya, R. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing / R. Buyya, M. Murshed // Concurrency and computation: practice and experience. – 2002. – vol. 14. – №. 13-15. – P. 1175-1220.

30. Buyya, R. The gridbus toolkit for service oriented grid and utility computing: An overview and status report / R. Buyya, S. Venugopal // Grid Economics and Business Models, 2004. GECON 2004. 1st IEEE International Workshop on. – IEEE, 2004. – P. 19-66.

31. Cai, M. A Peer-to-Peer replica location service based on a distributed hash table / M. Cai, A. Chervenak, M. Frank // Proceedings of the 2004 ACM/IEEE conference on Supercomputing. – IEEE Computer Society, 2004. – P. 56.

32. Carlin, S. Cloud computing security / S. Carlin, K. Curran //Pervasive and Ubiquitous Technology Innovations for Ambient Intelligence Environments. – IGI Global, 2013. – P. 12-17.

33. Catteddu, D. Cloud Computing: benefits, risks and recommendations for information security / D. Catteddu // Web application security. – Springer, Berlin, Heidelberg, 2010. – P. 17.

34. Celesti, A. Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems / A. Celesti, M. Fazio, M. Villari, A. Puliafito // Journal of Network and Computer Applications. – 2016. – vol. 59. – P. 208-218.

35. Chang, F. Bigtable: A distributed storage system for structured data / F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, R.E. Gruber // ACM Transactions on Computer Systems (TOCS). – 2008. – vol. 26. – № 2. – P. 4.

36. Chappell, D. Introducing the Azure services platform / D. Chappell //White paper, Oct. – 2008. – vol. 1364. – №.11.

37. Chen, Y. Collaborative detection of DDoS attacks over multiple network domains / Y. Chen, K. Hwang, W.S. Ku //IEEE Transactions on Parallel & Distributed Systems. – 2007. – №. 12. – P. 1649-1662.

38. Chervenak, A. Data placement for scientific applications in distributed environments / A. Chervenak, E. Deelman, M. Livny, M.H. Su, R. Schuler, S. Bharathi, G. Mehta, K. Vahi // Proceedings of the 8th IEEE/ACM International Conference on Grid Computing. – IEEE Computer Society, 2007. – P. 267-274.

39. Chervenak, A. Giggle: a framework for constructing scalable replica location services / A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, H. Stockinger, B. Tierney // Supercomputing, ACM/IEEE 2002 Conference. – IEEE, 2002. – P. 58-58.

40. Chervenak, A. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets / A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke // Journal of network and computer applications. – 2000. – vol. 23. – №. 3. – P. 187-200.

41. Chervyakov, N. AR-RRNS: Configurable Reliable Distributed Data Storage Systems for Internet of Things to Ensure Security / N. Chervyakov, M. Babenko, A. Tchernykh, N. Kucherov, V. Miranda-López, J.M. Cortés-Mendoza // Future Generation Computer Systems. – 2017. – doi.org/10.1016/j.future.2017.09.061.

42. Chervyakov, N. Development of Information Security's Theoretical Aspects in Cloud Technology with the Use of Threshold Structures / N. Chervyakov, M. Babenko, M. Deryabin, A. Garianina. – IEEE 2014 International Conference Engineering and Telecommunication (EnT), 2014. – P. 38-42.

43. Chervyakov, N.I. An approximate method for comparing modular numbers and its application to the division of numbers in residue number systems / N.I. Chervyakov, M.G. Babenko, P.A. Lyakhov, I.N. Lavrinenko // Cybernetics and Systems Analysis. – 2014. – vol. 50. – №. 6. – P. 977-984.

44. Chervyakov, N.I. An efficient method of error correction in fault-tolerant modular neurocomputers / N.I. Chervyakov, P.A. Lyakhov, M.G. Babenko, A.I. Garyanina, I.N. Lavrinenko, A.V. Lavrinenko, M.A. Deryabin // Neurocomputing. – 2016. – vol. 205. – P. 32-44.

45. Dean, J. MapReduce: simplified data processing on large clusters / J. Dean, S. Ghemawat // Communications of the ACM. – 2008. – vol. 51. – №. 1. – P. 107-113.

46. Deelman, E. Data management challenges of data-intensive scientific workflows / E. Deelman,A. Chervenak // 8th IEEE International Symposium on. Cluster Computing and the Grid, 2008. CCGRID'08. – IEEE, 2008. – P. 687-692.

47. Deelman, E. Pegasus: Mapping scientific workflows onto the grid / E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, M. Livny // Grid Computing. – Springer, Berlin, Heidelberg, 2004. – P. 11-20.

48. Deelman, E. The cost of doing science on the cloud: the montage example / E. Deelman, G. Singh, M. Livny, B. Berriman, J. Good // Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, 2008. – P. 1-12.

49. Dilley, J. Globally distributed content delivery / J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, B. Weihl // IEEE Internet Computing. – 2002. – vol. 6. – №. 5. – P. 50-58.

50. Dimakis, A.G. A survey on network codes for distributed storage / A.G. Dimakis, K. Ramchandran, Y. Wu, C. Suh // Proceedings of the IEEE. – 2011. – vol. 99. – №. 3. – P. 476-489.

51. Dumitrescu, C.L. GangSim: a simulator for grid scheduling studies / C.L. Dumitrescu, I. Foster // IEEE International Symposium on. Cluster Computing and the Grid, 2005. – IEEE, 2005. – vol. 2. – P. 1151-1158.

52. Erkin, Z. Generating private recommendations efficiently using homomorphic encryption and data packing / Z. Erkin, T. Veugen, T. Toft, R.L. Lagendijk // IEEE transactions on information forensics and security. – 2012. – vol. 7. – №. 3. – P. 1053-1066.

53. Foster, I. The Grid 2: Blueprint for a future computing infrastructure / I. Foster, C. Kesselman. – Elsevier, Waltham: Morgan Kaufmann Publishers, 2004. – P. 737.

54. Francois, J. FireCol: a collaborative protection network for the detection of flooding DDoS attacks / J. Francois, I. Aib, R. Boutaba // IEEE/ACM Transactions on Networking (TON). – 2012. – vol. 20. – №. 6. – P. 1828-1841.

55. Gentry, C. Computing arbitrary functions of encrypted data / C. Gentry // Communications of the ACM. – 2010. – vol. 53. – №. 3. – P. 97-105.

56. Ghemawat, S. The Google file system. / S. Ghemawat, H. Gobioff, S.T. // ACM. – 2003. – vol. 37. – №. 5. – P. 29-43.

57. Glatard, T. Flexible and efficient workflow deployment of data-intensive applications on grids with moteur / T. Glatard, J. Montagnat, D. Lingrand, X. Pennec // The International Journal Of High Performance Computing Applications. – 2008. – vol. 22. – №. 3. – P. 347-360.

58. Goldreich, O. Chinese remaindering with errors / O. Goldreich, D. Ron, M. Sudan // Proceedings of the thirty-first annual ACM symposium on Theory of computing. – ACM, 1999. – P. 225-234.

59. Gomathisankaran, M. HORNS: A homomorphic encryption scheme for Cloud Computing using Residue Number System / M. Gomathisankaran, A. Tyagi, K. Namuduri. – IEEE Annual Conference Information Sciences and Systems (CISS), 2011. – P. 1-5.

60. González, Y.D. Gestión de la calidad en las telecomunicaciones. un acercamiento a la norma TL 9000 / Y.D. González, C.A. Calderón // Revista Telem@tica. – 2013. – vol. 12. – №. 1. – P. 23-31.

61. Grangetto, M. Joint source/channel coding and MAP decoding of arithmetic codes / M. Grangetto, P. Cosman, G. Olmo // IEEE Transactions on Communications. – 2005. – vol. 53. – №. 6. – P. 1007-1016.

62. Grobauer, B. Understanding cloud computing vulnerabilities / B. Grobauer, T. Walloschek, E. Stocker // IEEE Security & Privacy. – 2011. – vol. 9. – №. 2. – P. 50-57.

63. Grossman, R. Data mining using high performance data clouds: experimental studies using sector and sphere / R. Grossman, Y. Gu // Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. – ACM, 2008. – P. 920-927.

64. Grossman, R.L. Compute and storage clouds using wide area high performance networks / R.L. Grossman, Y. Gu, M. Sabala, W. Zhang // Future Generation Computer Systems. – 2009. – vol. 25. – №. 2. – P. 179-183.

65. Herodotou, H. Starfish: a self-tuning system for big data analytics / H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F.B. Cetin, S. Babu // CIDR 2011 - 5th Biennial Conference on Innovative Data Systems Research. – 2011. – vol. 11. – №. 2011. –P 261-272.

66. Hoffa, C. On the use of cloud computing for scientific workflows / C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, J. Good // IEEE Fourth International Conference on eScience. – IEEE, 2008. – P. 640-645.

67. Inmon, W.H. Building the Data Warehouse: Getting Started / W.H. Inmon. – Wiley Publishing, Inc., Indianapolis, Indiana, 2005. – P. 576.

68. Jablonski, S. Dalton: An infrastructure for scientific data management / S. Jablonski, O. Curé, M.A. Rehman, B. Volz // International Conference on Computational Science. – Springer, Berlin, Heidelberg, 2008. – P. 520-529.

69. Jullien, G. Complex digital signal processing over finite rings / G. Jullien, R. Krishnan, W. Miller // IEEE transactions on circuits and systems. – 1987. – vol. 34. – №. 4. – P. 365-377.

70. Juve, G. Data sharing options for scientific workflows on amazon ec2 / G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B.P. Berman, P. Maechling //Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis. – IEEE Computer Society, 2010. – P. 1-9.

71. Kondo, D. Cost-benefit analysis of cloud computing versus desktop grids / D. Kondo, B. Javadi, P. Malecot, F. Cappello, D.P. Anderson // IEEE International Symposium on Parallel & Distributed Processing (IPDPS 2009). – 2009. – vol. 9. – P. 1-12.

72. Kong, Z. Decentralized coding algorithms for distributed storage in wireless sensor networks / Z. Kong, S.A. Aly, E. Soljanin // IEEE Journal on Selected Areas in Communications. – 2010. – vol. 28. – №. 2. – P. 261-267.

73. Kucherov, N. Towards reliable low cost distributed storage in multi-clouds / N. Chervyakov, M. Babenko, A. Tchenykh, I. Dvoryaninova, N. Kucherov // 2017 International Siberian Conference on Control and Communications (SIBCON), 2017. – P. 1-6.

74. Kucherov, N.N. A high-speed residue-to-binary converter based on approximate Chinese Remainder Theorem / N.N. Kucherov, V.A. Kuchukov, N.N.Kuchukova, A.E. Shangina // 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018. – P. 325-328.

75. Legrand, A. Scheduling distributed applications: the simgrid simulation framework / A. Legrand, L. Marchal, H. Casanova // 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRIDí03). – IEEE, 2003. – P. 138.

76. Li, M. Data security and privacy in wireless body area networks / M. Li, W. Lou, K. Ren // IEEE Wireless communications. – 2010. – vol. 17. – №. 1. – P. 51-58.

77. Lin, H.Y. A secure decentralized erasure code for distributed networked storage / H.Y. Lin, W.G. Tzeng // IEEE transactions on Parallel and Distributed Systems. – 2010. – vol. 21. – №. 11. – P. 1586-1594.

78. Lin, H.Y. A secure erasure code-based cloud storage system with secure data forwarding / H.Y. Lin, W.G. Tzeng // IEEE transactions on parallel and distributed systems. – 2012. – vol. 23. – №. 6. – P. 995-1003.

79. Lin, S.J. Novel polynomial basis and its application to reed-solomon erasure codes / S.J. Lin, W.H. Chung, Y.S. Han // 2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS). – IEEE, 2014. – P. 316-325.

80. Liu, D.T. GridDB: a data-centric overlay for scientific grids / D.T. Liu, M.J. Franklin // Proceedings of the Thirtieth international conference on Very large data bases – VLDB Endowment, 2004. – vol. 30. – P. 600-611.

81. Ludäscher B. Scientific workflow management and the Kepler system / B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee, J. Tao, Y. Zhao // Concurrency and Computation: Practice and Experience. – 2006. – vol. 18. – №. 10. – P. 1039-1065.

82. Mignotte, M. How to share a secret / M. Mignotte // Workshop on Cryptography. – Springer, Berlin, Heidelberg, 1982. – P. 371-375.

83. Oram, A. Peer-to-peer: Harnessing the power of disruptive technologies / A. Oram // SIGMOD Record. – 2003. – vol. 32. – №. 2. – P. 57.

84. Ozsu, M.T. Distributed database systems: where are we now? / M.T. Ozsu, P. Valduriez // Computer. – 1991. – vol. 24. – №. 8. – P. 68-78.

85. Pang, L.J. A new (t, n) multi-secret sharing scheme based on Shamir's secret sharing / L.J. Pang, Y.M. Wang // Applied Mathematics and Computation. – 2005. – vol. 167. – №. 2. – P. 840-848.

86. Parakh, A. Online data storage using implicit security / A. Parakh, S. Kak // Information Sciences. – 2009. – vol. 179. – №. 19. – P. 3323-3331.

87. Parakh, A. Space efficient secret sharing for implicit data security / A. Parakh, S. Kak // Information Sciences. – 2011. – vol. 181. – №. 2. – P. 335-341.

88. Patil, S. Extended Proactive Secret Sharing using Matrix Projection Method / S. Patil, N. Rana, D. Patel, P. Hodge // International Journal of Scientific & Engineering Research. – 2013. – vol. 4. – №. 6. – P. 2024-2029.

89. Patterson, D.A. A case for redundant arrays of inexpensive disks (RAID) / D.A. Patterson, G. Gibson, R.H. Katz // ACM. – 1988. – vol. 17. – №. 3. – P. 109-116.

90. Quezada-Pina, A. Adaptive parallel job scheduling with resource admissible allocation on two-level hierarchical grids / A. Quezada-Pina, A. Tchernykh, J.L. González-García, A. Hirales-Carbajal, J.M. Ramírez-Alcaraz, U. Schwiegelshohn, R. Yahyapour, V. Miranda-López // Future Generation Computer Systems. – 2012. – vol. 28. – №. 7. – P. 965-976.

91. Rabin, M.O. Efficient dispersal of information for security, load balancing, and fault tolerance / M.O. Rabin // Journal of the ACM (JACM). – 1989. – vol. 36. – №. 2. – P. 335-348.

92. Ruj, S. DACC: Distributed access control in clouds / S. Ruj, A. Nayak, I. Stojmenovic // 2011 International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11 (TrustCom 2011). – IEEE, 2011. – P. 91-98.

93. Samanthula, B.K. A secure data sharing and query processing framework via federation of cloud computing / B.K. Samanthula, Y. Elmehdwi, G. Howser, S. Madria // Information Systems. – 2015. – vol. 48. – P. 196-212.

94. Sathiamoorthy, M. Xoring elephants: Novel erasure codes for big data / M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A.G. Dimakis, R. Vadali, S. Chen, D. Borthakur // Proceedings of the VLDB Endowment. – VLDB Endowment, 2013. – vol. 6. – №. 5. – P. 325-336.

95. Schmuck, F.B. GPFS: A Shared-Disk File System for Large Computing Clusters / F.B. Schmuck, R.L. Haskin // FAST. – 2002. – vol. 2. – №. 19. – P. 231–244.

96. Schreiber, T. Amazon Web Services Security / T. Schreiber. – Seminarthesis of Ruhr-Universität Bochum. – 2011. – P. 25.

97. Shah, N.B. Interference alignment in regenerating codes for distributed storage: Necessity and code constructions / N.B. Shah, K.V. Rashmi, P.V. Kumar, K. Ramchandran // IEEE Transactions on Information Theory. – 2012. – vol. 58. – №. 4. – P. 2134-2158.

98. Soltis, S.R. The global file system. / S.R. Soltis, T.M. Ruwart, M.T. OKeefe // 5th NASA Goddard Conference on Mass Storage Systems and Technologies – 1996. – vol. 2. – P. 319-342.

99. Stockinger, H. File and object replication in data grids / H. Stockinger, A. Samar, K. Holtman, B. Allcock, I. Foster, B. Tierney // Cluster Computing. – 2002. – vol. 5. – №. 3. – P. 305-314.

100. Szabo, N.S. Residue arithmetic and its applications to computer technology. / N.S. Szabo, R.I. Tanaka. – McGraw-Hill, New York, 1967. – P. 103-104.

101. Szalay, A. 2020 Computing: Science in an exponential world / A. Szalay, J. Gray // Nature. – 2006. – vol. 440. – №. 7083. – P. 413.

102. Tatebe, O. Grid datafarm architecture for petascale data intensive computing / O. Tatebe, Y. Morita, S. Matsuoka, N. Soda, S. Sekiguchi // 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002. – IEEE, 2002. – P. 102-102.

103. Tchernykh, A. Online bi-objective scheduling for IaaS clouds ensuring quality of service / A. Tchernykh, L. Lozano, U. Schwiegelshohn, P. Bouvry, J.E. Pecero, S. Nesmachnow, A.Y. Drozdov // Journal of Grid Computing. – 2016. – vol. 14. – №. 1. – P. 5-22.

104. Tchernykh, A. Towards mitigating uncertainty of data security breaches and collusion in cloud computing / A. Tchernykh, M. Babenko, N. Chervyakov, J.M. Cortes-Mendoza, N. Kucherov, V. Miranda-Lopez, M. Deryabin, I. Dvoryaninova, G. Radchenko // 2017 28th International Workshop on Database and Expert Systems Applications (DEXA), 2017. – P. 137-141.

105. Tchernykh, A. Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability / A. Tchernykh, U. Schwiegelsohn, E. Talbi, M. Babenko // Journal of Computational Science. – 2016. – doi.org/10.1016/j.jocs.2016.11.011.

106. Tsakalozos, K. Flexible use of cloud resources through profit maximization and price discrimination / K. Tsakalozos, H. Kllapi, E. Sitaridi, M. Roussopoulos, D. Paparas, A. Delis // IEEE 27th International Conference on. Data Engineering (ICDE), – IEEE, 2011. – P. 75-86.

107. Venugopal, S. A grid service broker for scheduling distributed data-oriented applications on global grids / S. Venugopal, R. Buyya, L. Winton // Proceedings

of the 2nd workshop on Middleware for grid computing. – ACM, 2004. – P. 75-80.

108. Venugopal, S. A taxonomy of data grids for distributed data sharing, management, and processing / S. Venugopal, R. Buyya, K. Ramamohanarao // ACM Computing Surveys (CSUR). – 2006. – vol. 38. – №. 1. – P. 3.

109. Vouk A.M. Cloud computing–issues, research and implementations / A.M. Vouk // Journal of computing and information technology. – 2008. – vol. 16. – №. 4. – P. 235-246.

110. Wang, C. Toward secure and dependable storage services in cloud computing / C. Wang, Q. Wang, K. Ren, N. Cao, W. Lou // IEEE transactions on Services Computing. – 2012. – vol. 5. – №. 2. – P. 220-232.

111. *Weiss A.* Weiss, A. Computing in the clouds / A. Weiss // netWorker. – 2007. – vol. 11. – №. 4. – P. 16-25.

112. Wu, H. Network security for virtual machine in cloud computing / H. Wu, Y. Ding, C. Winer, L. Yao // 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT). – IEEE, 2010. – P. 18-21.

113. Wylie, J.J. Survivable information storage systems / J.J. Wylie, M.W. Bigrigg, J.D. Strunk, G.R. Ganger, H. Kiliccote, P.K. Khosla // Computer. – 2000. – vol. 33. – №. 8. – P. 61-68.

114. Yang, C.C. A (t, n) multi-secret sharing scheme / C.C. Yang, T.Y. Chang, M.S. Hwang // Applied Mathematics and Computation. – 2004. – vol. 151. – №. 2. – P. 483-490.

115. Yu, S. Can we beat DDoS attacks in clouds? / S. Yu, Y. Tian, S. Guo, O.D. Wu //IEEE Transactions on Parallel and Distributed Systems. – 2014. – vol. 25. – №. 9. – P. 2245-2254.

116. Yuan, D. Computation and Storage in the Cloud: Understanding the Trade-offs / D. Yuan, Y. Yang, J. Chen. – Elsevier Newnes, 2012. – P. 137.

117. Zhang, X. A privacy leakage upper bound constraint-based approach for cost-effective privacy preserving of intermediate data sets in cloud / X. Zhang, C. Liu,

S. Nepal, S. Pandey, J. Chen // IEEE Transactions on Parallel and Distributed Systems. – 2013. – vol. 24. – №. 6. – P. 1192-1202.

118. Zhang, Y. Cross-VM side channels and their use to extract private keys / Y. Zhang, A. Juels, M.K. Reiter, T. Ristenpart // Proceedings of the 2012 ACM conference on Computer and communications security. – ACM, 2012. – P. 305-316.

Приложение А

Программная реализация многоуровневой дизъюнктивной схемы разделения секрета

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Numerics;
using Api.Clouds.Extensions;
using Api.Clouds.Types;

namespace Api.Clouds.SecretShareSystem
{
  public class Mode1
  {
    public static IEnumerable<ExtendedBigInteger> Split(
       ExtendedBigInteger input, int n)
    {
      BigInteger S1, S2, S3;

      BigInteger Temp = (BigInteger.One << n) - 1;
      if (BigInteger.Compare(input.Value, Temp) > -1)
      {
        S2 = input.Value & Temp;
        S1 = S2 + (input.Value >> n);

        if (BigInteger.Compare(Temp, S1) != 1) { S1 = S1 - Temp;
            }

        Temp = (BigInteger.One << (n + 1)) - 1;
        S3 = (Temp & input.Value) + (input.Value >> (n + 1));
        if (BigInteger.Compare(Temp, S3) < 1) { S3 = S3 - Temp;
           }
      }
      else
      {
        S1 = input.Value;
        S2 = input.Value;
```

```
      S3 = input.Value;
    }

    return new List<ExtendedBigInteger>() { new
       ExtendedBigInteger(input.IsNegative, S1),
      new ExtendedBigInteger(input.IsNegative, S2),
      new ExtendedBigInteger(input.IsNegative, S3) };
  }

  public static ExtendedBigInteger Compose(ExtendedBigInteger
     N1, int n1Index, ExtendedBigInteger N2, int n2Index, int
     n)
  {
    if (n1Index == 0)
    {
      if (n2Index == 1)
      {
        return ComposeS1S2(N1, N2, n);
      }
      else if (n2Index == 2)
      {
        return ComposeS1S3(N1, N2, n);
      }
      else
      {
        return null;
      }
    }
    else if (n1Index == 1)
    {
      return ComposeS2S3(N1, N2, n);
    }
    else
    {
      return null;
    }
  }
  public static ExtendedBigInteger ComposeS1S2(
     ExtendedBigInteger S1, ExtendedBigInteger S2, int n)
  {
    BigInteger z = S1.Value - S2.Value;
    if (z.Sign == -1) { z = z + ((BigInteger.One << n) - 1); }
    BigInteger result = (z << n) + S2.Value;
```

```
      return new ExtendedBigInteger(S1.IsNegative, result);
    }
    public static ExtendedBigInteger ComposeS1S3(
       ExtendedBigInteger S1, ExtendedBigInteger S3, int n)
    {
      BigInteger P = (BigInteger.One << (2 * n + 1)) - (
        BigInteger.One << (n + 1)) - (BigInteger.One << n) + 1;
      BigInteger result = ((S1.Value - S3.Value) << (n + 1)) + 2
          * S3.Value - S1.Value;
      if (result.Sign == -1) { result = result + P; }
      if (BigInteger.Compare(P, result) != 1) { result = result
        - P; }

      return new ExtendedBigInteger(S1.IsNegative, result);
    }
    public static ExtendedBigInteger ComposeS2S3(
       ExtendedBigInteger S2, ExtendedBigInteger S3, int n)
    {
      BigInteger Temp = S3.Value - S2.Value;
      BigInteger result = (Temp << (n + 1)) + S2.Value;
      if (result.Sign == -1) { result = result + (BigInteger.One
          << (2 * n + 1)) - (BigInteger.One << n); }
      if (BigInteger.Compare((BigInteger.One << (2 * n + 1)) - (
        BigInteger.One << n), result) != 1) { result = result -
          ((BigInteger.One << (2 * n + 1)) - (BigInteger.One <<
        n)); }

      return new ExtendedBigInteger(S2.IsNegative, result);


    }
  }

  using Api.Clouds.Extensions;
  using Api.Clouds.ModuleBase;
  using Api.Clouds.SecretShareSystem;
  using Api.Clouds.Types;
  using Api.Clouds.Views;
  using Caliburn.Micro;
  using Microsoft.Win32;
  using NLog;
  using System;
```

```
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;


namespace Api.Clouds.ViewModels
{
class ShellViewModel : Screen
{
  private List<FileStream> activeStreams;
  private readonly Logger _logger = NLog.LogManager.
     GetCurrentClassLogger();
  private readonly string[][] sharingMap = new string[5][] {
     new string[] { "shade02.shd" },
  new string[] { "shade00.shd", "shade12.shd" },
  new string[] { "shade01.shd", "shade11.shd", "shade21.shd"
     },
  new string[] { "shade10.shd", "shade22.shd" },
  new string[] { "shade20.shd" }};
  private List<string> _CloudFilesList;
  struct ModuleFilepair
{
  public IModule module;
  public string FileName;
}

public ShellViewModel()
{
  Modules = new BindableCollection<IModule>();
  ModulesToUpload = new BindableCollection<IModule>();
  ModulesToDownload = new BindableCollection<IModule>();
  _CloudFilesList = new List<string>();
  CloudFilesList = new BindableCollection<string>();

  var types = ModuleInstanceFactory.GetExportedTypes();
  foreach (var type in types)
    {
    Modules.Add(
    ModuleInstanceFactory.CreateInstance(type.ToString())
    );
```

```
        }

    if (Modules.Count >= 5)
    {
      for (int i = 0; i < 5; i++)
    {
      ModulesToUpload.Add(Modules[i]);
    }
      for (int i = 0; i < 3; i++)
        {
        if (i == 2)
    {
    ModulesToDownload.Add(Modules[i + 1]);
    continue;
    }
    ModulesToDownload.Add(Modules[i]);
    }
    }
    activeStreams = new List<FileStream>();
    }

    #region Properties
    #region Private
    private int n = 1024;
    private string _FilePath;
    private string _ResultOutput = "";
    private string _NameOfFileToDownload;
    private string _NameOfFileToCompose;
    private bool _SplitOperationStarted;
    private bool _LocalSplitOperationComplete;
    private int _UploadedFilesCounts;
    private bool _UploadProcessComplete;
    private int _DownloadedFilesCount;
    private bool _DownloadProcessStarted;
    private bool _DownloadProcessComplete;
    private bool _ComposeProcessStarted;
    private bool _ComposeProcessComplete;
    #endregion

    #region Public
    public BindableCollection<IModule> Modules { get; set; }
    public int ModuleN
    {
```

```
get
{
return n;
}
set
{
if (n == value) return;
n = value;
NotifyOfPropertyChange(() => ModuleN);
}
}
public BindableCollection<IModule> ModulesToUpload { get; set;
    }
public BindableCollection<IModule> ModulesToDownload { get;
   set; }
public BindableCollection<string> CloudFilesList { get; set; }
public string ResultOutput
{
get
{
return _ResultOutput;
}
set
{
if (_ResultOutput == value) return;
_ResultOutput = value;
NotifyOfPropertyChange(() => ResultOutput);
}
}
public string NameOfFileToCompose
{
get
{
return _NameOfFileToCompose;
}
set
{
if (_NameOfFileToCompose == value) return;
_NameOfFileToCompose = value;
NotifyOfPropertyChange(() => NameOfFileToCompose);
}
}
```

```
#region Visual states
public bool SplitOperationStarted
{
get
{
return _SplitOperationStarted;
}
set
{
if (_SplitOperationStarted == value) return;
_SplitOperationStarted = value;
NotifyOfPropertyChange(() => SplitOperationStarted);
}
}
public bool LocalSplitOperationComplete
{
get
{
return _LocalSplitOperationComplete;
}
set
{
if (_LocalSplitOperationComplete == value) return;
_LocalSplitOperationComplete = value;
NotifyOfPropertyChange(() => LocalSplitOperationComplete);
}
}
public int UploadedFilesCount
{
get
{
return _UploadedFilesCounts;
}
set
{
if (_UploadedFilesCounts == value) return;
_UploadedFilesCounts = value;
NotifyOfPropertyChange(() => UploadedFilesCount);
}
}
public bool UploadProcessComplete
{
get
```

```
    {
    return _UploadProcessComplete;
    }
    set
    {
    if (_UploadProcessComplete == value) return;
    _UploadProcessComplete = value;
    NotifyOfPropertyChange(() => UploadProcessComplete);
    }
    }
    public int DownloadedFilesCount
    {
    get
    {
    return _DownloadedFilesCount;
    }
    set
    {
    if (_DownloadedFilesCount == value) return;
    _DownloadedFilesCount = value;
    NotifyOfPropertyChange(() => DownloadedFilesCount);
    }
    }
    public bool DownloadProcessStarted
    {
    get
    {
    return _DownloadProcessStarted;
    }
    set
    {
    if (_DownloadProcessStarted == value) return;
    _DownloadProcessStarted = value;
    NotifyOfPropertyChange(() => DownloadProcessStarted);
    }
    }
    public bool DownloadProcessComplete
    {
    get
    {
    return _DownloadProcessComplete;
    }
    set
```

```
{
if (_DownloadProcessComplete == value) return;
_DownloadProcessComplete = value;
NotifyOfPropertyChange(() => DownloadProcessComplete);
}
}
public bool ComposeProcessStarted
{
get
{
return _ComposeProcessStarted;
}
set
{
if (_ComposeProcessStarted == value) return;
_ComposeProcessStarted = value;
NotifyOfPropertyChange(() => ComposeProcessStarted);
}
}
public bool ComposeProcessComplete
{
get
{
return _ComposeProcessComplete;
}
set
{
if (_ComposeProcessComplete == value) return;
_ComposeProcessComplete = value;
NotifyOfPropertyChange(() => ComposeProcessComplete);
}
}
#endregion
#endregion
#endregion

#region Commands
public async Task Authenticate(string ModuleName)
{
var module = Modules.SingleOrDefault(x => x.ModuleName ==
   ModuleName);
if (module == null)
{
```

```
360     _logger.Error("ShellViewModel: module not loaded.");
        return;
        }

        if (module.AuthComplete == null)
365     {
        module.AuthComplete += (sender, e) =>
        {
        if (((IModule)sender).IsAuthenticated == true)
        {
370     AppendMessage("Модуль " + ((IModule)sender).ModuleName + ": Ав
           торизованно.");
        _CloudFilesList.AddRange(((IModule)sender).GetFiles().Distinct
           ());
        AggregateAvalibleFiles();
        }
        else
375     {
        AppendMessage("Модуль " + ((IModule)sender).ModuleName + ": Ав
           торизация провалилась.");
        }
        };
        }
380
        await module.Authenticate();
        }
        public async Task FileDropped(DragEventArgs e)
        {
385     if (e.Data.GetDataPresent(DataFormats.FileDrop))
        {
        // Note that you can have more than one file.
        string[] files = (string[])e.Data.GetData(DataFormats.FileDrop
           );

390     if (files != null && files.Any())
        {
        SplitOperationStarted = false;
        LocalSplitOperationComplete = false;
        UploadProcessComplete = false;
395
        var filePath = files.First();
        SplitOperationStarted = true;
        await SplitProcess(filePath);
```

```
    LocalSplitOperationComplete = true;
    await UploadProcess();
    UploadProcessComplete = true;
    }
    }
    }
    public async Task SaveFile()
    {
    string newFileName = null;
    var dialog = new SaveFileDialog();
    dialog.FileName = NameOfFileToCompose;
    if (dialog.ShowDialog() == true)
    {
    newFileName = dialog.FileName;
    }
    else
    {
    return;
    }
    if (newFileName == null)
    {
    MessageBox.Show("Ошибка");
    return;
    }

    await StartCompose(newFileName);
    }
    public void OpenSplitSettingsDialog()
    {
    var dialog = new SplitSettingsViewModel(Modules, n);
    dialog.ModulesToDownload.AddRange(this.ModulesToDownload);
    dialog.ModulesToUpload.AddRange(this.ModulesToUpload);
    var dialogView = new SplitSettingsView() { DataContext =
       dialog };
    dialogView.ShowDialog();
    ModulesToUpload.Clear();
    ModulesToUpload.AddRange(dialog.ModulesToUpload);
    ModulesToDownload.Clear();
    ModulesToDownload.AddRange(dialog.ModulesToDownload);
    n = dialog.N;
    }
    public async Task UpdateFilesInCloudsList()
    {
```

```
    await Task.Run(() =>
    {
    _CloudFilesList.Clear();
    Parallel.ForEach(Modules, x =>
    {
    if (!x.IsAuthenticated) return;
    _CloudFilesList.AddRange(x.GetFiles().Distinct());
    });
    AggregateAvalibleFiles();
    });
    }
    #endregion

    #region Utils
    #region Processes
    /// <summary>
    /// Процесс разделения файла
    /// </summary>
    /// <param name="filePath"></param>
    /// <returns></returns>
    private async Task SplitProcess(string filePath)
    {
    await Task.Run(() =>
    {
    List<Task> Tasks = new List<Task>();
    AppendMessage("Начало.");
    if (!File.Exists(filePath))
    {
    AppendMessage("Выбранный файл не существует");
    return;
    }

    AppendMessage("Разделяем файл...");
    var splitStartTime = DateTime.Now;
    Split(filePath, ModuleN);
    var splitEndTime = DateTime.Now;
    var splitElapsedTime = (splitEndTime - splitStartTime).
       TotalSeconds;
    AppendMessage("Файл разделен за " + splitElapsedTime + " секун
       д(ы).");
    var fileSize = new FileInfo(filePath).Length;
    AppendMessage("Скорость разделения: " + ((fileSize / 1024) /
       splitElapsedTime) + " килобайт в секунду.");
```

```
    });
    }
    /// <summary>
    /// Процесс выгрузки частей разделенного файла в облака
    /// </summary>
    /// <returns></returns>
    private async Task UploadProcess()
    {
    await Task.Run(() =>
    {
    UploadedFilesCount = 0;
    List<Task> Tasks = new List<Task>();
    AppendMessage("Выгружаем файлы в облака...");
    var uploadStartTime = DateTime.Now;
    if (ModulesToUpload.Count != 5)
    {
    AppendMessage("Набор модулей для загрузки подобран не верно. О
       тмена операции.");
    return;
    }
    for (int i = 0; i < sharingMap.Count(); i++)
    {
    var module = ModulesToUpload[i];
    module.UploadProgressComplete = (sender, e) =>
    {
    if (e.Result == OperationResultEnum.OK)
    {
    UploadedFilesCount++;
    AppendMessage(e.FileName + " загружен...");
    }
    else
    {
    AppendMessage("Ошибка при загрузке " + e.FileName + ". Модуль:
        " + sender.ToString() + ". См.лог.");
    }
    };

    foreach (var shareName in sharingMap[i])
    {
    var fileName = Directory.GetFiles("Upload").SingleOrDefault(x
       => x.Contains(shareName));
    if (fileName == null)
    {
```

```
AppendMessage("Один из файлов для загрузки отсутствует. Имя: "
    + shareName);
return;
}
fileName = Path.GetFileName(fileName);
var stream = new FileStream(Path.Combine("Upload", fileName),
   FileMode.Open, FileAccess.Read);
activeStreams.Add(stream);
Tasks.Add(module.UploadFileAsync(stream, fileName));
}
}

Task.WaitAll(Tasks.ToArray());
var uploadEndTime = DateTime.Now;
var elapsedUploadTime = (uploadEndTime - uploadStartTime).
   TotalSeconds;
AppendMessage("Все файлы загружены за " + elapsedUploadTime +
   " секунд(ы).");

foreach (var stream in activeStreams)
{
stream.Close();
}
activeStreams.Clear();

long commonShadesSize = 0;
foreach (var file in new DirectoryInfo("Upload").GetFiles())
   commonShadesSize += (file.Length / 1024);
AppendMessage("Скорость загрузки: " + commonShadesSize /
   elapsedUploadTime + " килобайт в секунду.");
});
}
/// <summary>
/// Процесс восстановления файла
/// </summary>
/// <param name="TargetFileName">Новое имя файла</param>
/// <returns></returns>
public async Task StartCompose(string TargetFileName)
{
await Task.Run(() =>
{
DownloadedFilesCount = 0;
DownloadProcessStarted = false;
```

```
    DownloadProcessComplete = false;
    ComposeProcessComplete = false;
560 ComposeProcessStarted = false;

    if (!ModulesToDownload.Any())
    {
    AppendMessage("Выберите сначала облака из которых восстанавлив
       ать файлы.");
565 return;
    }
    if (!Directory.Exists("Download")) Directory.CreateDirectory("
       Download");
    foreach (var file in (new DirectoryInfo("Download").GetFiles()
       ))
    {
570 file.Delete();
    }

    DownloadProcessStarted = true;
    var files = prepareFilesToCompose(ModulesToDownload,
       NameOfFileToCompose);
575 if (files == null)
    {
    AppendMessage("При выборе файлов для восстановления исходного
       файла возникла ошибка.");
    return;
    }
580
    var tasks = new List<Task>();
    foreach (var pair in files)
    {
    tasks.Add(DowloadFile(pair.module, pair.FileName));
585 }
    Task.WaitAll(tasks.ToArray());
    DownloadProcessStarted = false;
    DownloadProcessComplete = true;

590 ComposeProcessStarted = true;
    Compose(NameOfFileToCompose, n, TargetFileName);
    ComposeProcessStarted = false;
    ComposeProcessComplete = true;
    });
595 }
```

```
#endregion

/// <summary>
/// Записывает сообщение в строку консоли (ResultOutput)
/// Отправляет сообщение в лог с уровнем Info
/// </summary>
/// <param name="message">сообщение</param>
private void AppendMessage(string message)
{
ResultOutput += DateTime.Now.ToString("hh:mm:ss") + ": " +
   message + "\n";
_logger.Info(message);
}
/// <summary>
/// Скачивает файл в папку Download
/// </summary>
/// <param name="module">Модуль, с помощью которого будет прои
   зводиться скачивание</param>
/// <param name="fileName">Имя скачиваемого файла</param>
/// <returns></returns>
private async Task DowloadFile(IModule module, string fileName
   )
{
await Task.Run(() =>
{
bool completed = false;
if (Path.GetExtension(fileName) != ".shd")
{
fileName = fileName + ".shd";
}
AppendMessage("Начало скачивания. Модуль: " + module.
   ModuleName + ". Имя файла: " + fileName);

DateTime startTime = DateTime.Now;
DateTime endTime = DateTime.Now;

if (!Directory.Exists("Download"))
{
Directory.CreateDirectory("Download");
}
FileStream fs = null;
try
{
```

```
635     fs = new FileStream(Path.Combine("Download", fileName),
           FileMode.Create);
        }
        catch (Exception ex)
        {
        _logger.Error("ShellViewModel: Download method. Error while
           trying to create file.", ex);
640     return;
        }
        activeStreams.Add(fs);
        if (module != null && module.IsAuthenticated)
        {
645     module.DownloadProgressComplete += (sender, e) =>
        {
        ThreadPool.QueueUserWorkItem(new WaitCallback((treadSender) =>
        {
        if (e.FileName == fileName)
650     {
        if (e.Result == OperationResultEnum.ERROR)
        {
        AppendMessage("Ошибка загрузки файла. Модуль: " + ((IModule)
           sender).ModuleName + ". Имя файла: " + fileName);
        return;
655     }
        DownloadedFilesCount++;
        var stream = activeStreams.SingleOrDefault(x => x.Name.
           Contains(fileName));
        if (stream != null)
        {
660     try
        {
        stream.Flush();
        stream.Close();
        }
665     catch (ObjectDisposedException ex)
        {
        _logger.Warn("Module " + ((IModule)sender).ModuleName + "
           disposed stream of file " + fileName);
        }
        activeStreams.Remove(stream);
670     }
        endTime = DateTime.Now;
        var elapsedTime = endTime - startTime;
```

```
AppendMessage("Файл загружен за " + elapsedTime.TotalSeconds +
    " секунд. Модуль: "
+ module.ToString() + ". Имя файла: " + fileName);
var fileSize = new FileInfo(Path.Combine("Download", fileName)
   ).Length / 1024;
AppendMessage("Средняя скорость загрузки: " + fileSize /
   elapsedTime.TotalSeconds + " килобайт в секунду");
completed = true;
}
}));
};
}
startTime = DateTime.Now;
module.DownloadFileAsync(fs, fileName);
while (!completed)
{
Thread.Sleep(100);
}
});
}
/// <summary>
/// Собирает подходящие для восстановления файлы
/// </summary>
/// <param name="fileName">Имя файла для восстановления</param
   >
/// <returns></returns>
private List<List<ExtendedFileStream>> AggregateFiles(string
   fileName)
{
var files = Directory.GetFiles("Download").Where(x => x.
   Contains(fileName)).ToList();
if (files.Count() < 4)
{
_logger.Error("ShellViewModel: Compose cant be perfomed. Not
   enough shadows.");
return null;
}

List<string> filesWithoutExt = new List<string>();
files.ForEach(x => filesWithoutExt.Add(
(Path.GetExtension(x) == ".shd") ? Path.
   GetFileNameWithoutExtension(x) : x));
```

```
List<ExtendedFileStream> streams = new List<ExtendedFileStream
   >();
for (int i = 0; i < filesWithoutExt.Count; i++)
{
var firstIndex = int.Parse(filesWithoutExt[i][filesWithoutExt[
   i].Length - 2].ToString());
var secondIndex = int.Parse(filesWithoutExt[i][filesWithoutExt
   [i].Length - 1].ToString());
var stream = new ExtendedFileStream(files[i], firstIndex,
   secondIndex);
streams.Add(stream);
}

List<List<ExtendedFileStream>> resultStreams = new List<List<
   ExtendedFileStream>>();
for (int i = 0; i < 3; i++)
{
var tmp = streams.Where(x => x.FirstIndex == i).ToList();
if (tmp.Count > 0)
{
resultStreams.Add(tmp);
}
}

return resultStreams;
}
/// <summary>
/// Обрабатывает список всех файлов в облаках и возвращает спи
   сок файлов, которые возможно восстановить
/// </summary>
private void AggregateAvalibleFiles()
{
var tmpList = new List<string>();
_CloudFilesList.ForEach(x =>
{
if (x != null && x.Contains("_shade"))
{
var name = x.Substring(0, x.IndexOf("_shade"));
tmpList.Add(name);
}
});

CloudFilesList.Clear();
```

```
745     foreach (var name in tmpList)
        {
        var nameCount = tmpList.Where(x => x.Equals(name)).Count();
        if (nameCount >= 4 && !CloudFilesList.Contains(name))
        {
750     CloudFilesList.Add(name);
        }
        }
        }
        /// <summary>
755     /// получает номер тени
        /// </summary>
        /// <param name="fileName">имя файла из которого нужно получит
           ь индекс (без расширения)</param>
        /// <param name="indexLevel">уровень индекса (первый=1, второй
           =2)</param>
        /// <returns>индекс</returns>
760     private static int getShadeIndex(string fileName, int
           indexLevel)
        {
        var Name = (Path.GetExtension(fileName) == ".shd") ? Path.
           GetFileNameWithoutExtension(fileName) : fileName;
        if (indexLevel == 1)
        {
765     return int.Parse(Name[Name.Length - 2].ToString());
        }
        else
        {
        return int.Parse(Name[Name.Length - 1].ToString());
770     }
        }
        /// <summary>
        /// Возвращает 2 пары файлов для восстановления файла
        /// </summary>
775     /// <param name="Modules"></param>
        /// <param name="fileNameToCompose"></param>
        /// <returns></returns>
        private static List<ModuleFilepair> prepareFilesToCompose(
           IEnumerable<IModule> Modules, string fileNameToCompose)
        {
780     List<ModuleFilepair> avalibleFiles = new List<ModuleFilepair
           >();
        List<ModuleFilepair> files = new List<ModuleFilepair>();
```

```
foreach (var module in Modules)
{
var fileNames = module.GetFiles();
foreach (var fileName in fileNames)
{
if (fileName != null && fileName.Contains(fileNameToCompose))
{
avalibleFiles.Add(new ModuleFilepair()
{
module = module,
//FileName = (Path.GetExtension(fileName) == ".shd") ? Path.
   GetFileNameWithoutExtension(fileName) : fileName
FileName = fileName
});
}
}
}
int pairCount = 0;
for (int i = 0; i < 3; i++)
{
var filesForPairing = avalibleFiles.Where(x => i ==
   getShadeIndex(x.FileName, 1));
if (filesForPairing.Count() >= 2)
{
files.Add(filesForPairing.ElementAt(0));
files.Add(filesForPairing.ElementAt(1));
pairCount++;
if (pairCount >= 2)
break;
}
}
if (files.Count < 2)
{
return null;
}

return files;
}
#endregion

#region SecretSharing
public void Split(string fileName, int n)
{
```

```
    var bufferSize = ((2 * n) / 8) - 1;
    var buffer = new byte[bufferSize];
    var lowModules = new List<int>()
    {
    (n+1)/2,
    (n+1)/2,
    (n+2)/2,
    };

    using (var fs = new FileStream(fileName, FileMode.Open,
       FileAccess.Read))
    {
    if (!Directory.Exists("Upload"))
    {
    Directory.CreateDirectory("Upload");
    }
    foreach (System.IO.FileInfo file in new DirectoryInfo("Upload"
       ).GetFiles()) file.Delete();

    //Создаем потоки
    List<List<FileStream>> outStreams = new List<List<FileStream
       >>();
    for (int i = 0; i < 3; i++)
    {
    outStreams.Add(new List<FileStream>());
    for (int j = 0; j < 3; j++)
    {
    outStreams.ElementAt(i).Add(
    new FileStream(
    Path.Combine("Upload", Path.GetFileName(fileName) + "_shade" +
        i + j + ".shd"),
    FileMode.Create));
    }
    }

    //Разделяем и пишем
    while (fs.Read(buffer, 0, bufferSize) > 0)
    {
    ExtendedBigInteger s = new ExtendedBigInteger(buffer, true);
    List<ExtendedBigInteger> firstShadows = Mode1.Split(s, n).
       ToList();

    //Parallel.For(0, 3, i =>
```

```
    //{
    for (int i = 0; i < 3; i++)
    {
    List<ExtendedBigInteger> secondShadows = Mode1.Split(
       firstShadows[i], lowModules[i]).ToList();
    for (int j = 0; j < 3; j++)
    {
    //Parallel.For(0, 3, j =>
    //{
    byte[] shadeData = secondShadows[j].ToByteArray(false);
    int shadeDataLength = shadeData.Length;
    outStreams[i][j].Write(BitConverter.GetBytes(shadeDataLength),
        0, 4);
    outStreams[i][j].Write(shadeData, 0, shadeDataLength);
    outStreams[i][j].Flush();
    }//);
    }//);
    Array.Clear(buffer, 0, buffer.Length);
    }

    //Закрываем потоки
    foreach (var list in outStreams)
    {
    foreach (var stream in list)
    {
    stream.Close();
    }
    }
    }
    }
    public void Compose(string fileName, int n, string
       TargetFileName)
    {
    AppendMessage("Начинаем восстанавливать файл...");
    var bufferSize = ((2 * n) / 8) - 1;
    var lowModules = new List<int>()
    {
    (n+1)/2,
    (n+1)/2,
    (n+2)/2,
    };
```

```
    List<List<ExtendedFileStream>> inputStreams = AggregateFiles(
       fileName);
    if (inputStreams == null)
    {
    _logger.Error("ShellViewModel: No files aggregated or files
       count lesser then 4");
    return;
    }

    AppendMessage("Имя восстанавливаемого ϕайла: " + fileName);
    using (var fs = new FileStream(TargetFileName, FileMode.Create
       ))
    {
    _logger.Info("Start composition");
    ExtendedBigInteger s;

    ExtendedBigInteger[] firstLayerTmpData = new
       ExtendedBigInteger[2];
    ExtendedBigInteger[] secondLayerTmpData = new
       ExtendedBigInteger[2];
    while (inputStreams[0][0].Read(new byte[4], 0, 4) > 0)
    {
    inputStreams[0][0].Position -= 4;

    for (int i = 0; i < 2; i++)
    {
    for (int j = 0; j < 2; j++)
    {
    byte[] lengthBuffer = new byte[4];
    byte[] buffer = new byte[1];

    inputStreams[i][j].Read(lengthBuffer, 0, 4);
    int length = BitConverter.ToInt32(lengthBuffer, 0);

    Array.Resize(ref buffer, length);
    Array.Clear(buffer, 0, buffer.Length);

    inputStreams[i][j].Read(buffer, 0, length);
    secondLayerTmpData[j] = new ExtendedBigInteger(buffer, false);
    }
    firstLayerTmpData[i] = Mode1.Compose(secondLayerTmpData[0],
       inputStreams[i][0].SecondIndex,
```

```
    secondLayerTmpData[1], inputStreams[i][1].SecondIndex,
       lowModules[i]);
    }
    s = Mode1.Compose(firstLayerTmpData[0], inputStreams[0][0].
       FirstIndex,
940 firstLayerTmpData[1], inputStreams[1][0].FirstIndex, n);
    byte[] data = s.ToByteArray(true);
    if (data.Length != bufferSize)
    {
    Array.Resize(ref data, bufferSize);
945 }
    fs.Write(data, 0, data.Length);
    fs.Flush();

    //Parallel.For(0, 3, i =>
950 //{
    //for (int i = 0; i < 3; i++)
    //{
    //    //Parallel.For(0, 3, j =>
    //    //{
955 //    for (int j = 0; j < 3; j++)
    //    {
    //        byte[] lengthBuffer = new byte[4];
    //        byte[] buffer = new byte[1];
    //        inputStreams[i][j].Read(lengthBuffer, 0, 4);
960 //        int length = BitConverter.ToInt32(lengthBuffer, 0);

    //        Array.Resize(ref buffer, length);
    //        Array.Clear(buffer, 0, buffer.Length);

965 //        inputStreams[i][j].Read(buffer, 0, length);
    //        secondData[j] = new ExtendedBigInteger(buffer, false
       );
    //    }//);
    //    firstData[i] = Mode1.ComposeS1S2(secondData[0],
       secondData[1], lowModules[i]);
    //}//);
970 //s = Mode1.ComposeS1S2(firstData[0], firstData[1], n);
    //byte[] data = s.ToByteArray(true);
    //if (data.Length != bufferSize)
    //{
    //    Array.Resize(ref data, bufferSize);
975 //}
```

```
    //fs.Write(data, 0, data.Length);
    //fs.Flush();
    }

    for (int i = 0; i < 2; i++)
    {
    for (int j = 0; j < 2; j++)
    {
    inputStreams.ElementAt(i).ElementAt(j).Close();
    }
    }
    }
    AppendMessage("Файл восстановлен.");
    }
    #endregion
    }



    }

    using Caliburn.Micro;
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.Composition;
    using System.ComponentModel.Composition.Hosting;
    using System.ComponentModel.Composition.ReflectionModel;
    using System.IO;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;

    namespace Api.Clouds.ModuleBase
    {
    public class CompositionContainerSingleton
    {
    [ImportMany(typeof(IModule))]
    public static readonly CompositionContainer Instance =
       GetContainer();

    CompositionContainerSingleton() { }

    private static CompositionContainer GetContainer()
    {
```

```
        string pluginsPath = Path.Combine(AppDomain.CurrentDomain.
            BaseDirectory, "Modules");
        var modulesDirs = Directory.EnumerateDirectories(pluginsPath);
        AggregateCatalog catalog = new AggregateCatalog();
        foreach(var directory in modulesDirs)
        {
        DirectoryCatalog directoryCatalog = new DirectoryCatalog(Path.
            Combine(pluginsPath, directory), "Api.Clouds.*.dll");
        catalog.Catalogs.Add(directoryCatalog);
        }
        AssemblySource.Instance.AddRange(catalog.Parts.Select(part =>
            ReflectionModelServices.GetPartType(part).Value.Assembly));
        var container = new CompositionContainer(catalog);
        return container;
        }
        }
        }


        using System;
        using System.Collections.Generic;
        using System.IO;
        using System.Linq;
        using System.Text;
        using System.Threading.Tasks;

        namespace Api.Clouds.ModuleBase
        {
        public interface IModule : IDisposable
        {

        #region Properties
        bool IsAuthenticated { get; set; }
        string ModuleName { get; set; }
        string IconName { get; }
        #endregion

        #region Authentication
        Task Authenticate(AccountInfo accountInfo);
        Task Authenticate();
        EventHandler<AccountInfo> AuthComplete { get; set; }
        #endregion
```

```
#region Async Upload
Task UploadFileAsync(Stream stream, string fileName);
EventHandler<OperationProgressChangedEventArgs>
   UploadProgressChanged { get; set; }
EventHandler<OperationResult> UploadProgressComplete { get;
   set; }
#endregion

#region Async Download
Task DownloadFileAsync(Stream outStream, string fileName);
EventHandler<OperationProgressChangedEventArgs>
   DownloadProgressChanged { get; set; }
EventHandler<OperationResult> DownloadProgressComplete { get;
   set; }
#endregion

#region Sync Methods
void Upload(Stream stream, string fileName);
void Download(Stream outStream, string fileName);
List<string> GetFiles();
#endregion
}

}

using Api.Clouds.ModuleBase;
using System;
using System.Collections.Generic;
using System.ComponentModel.Composition.Primitives;
using System.ComponentModel.Composition.ReflectionModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Api.Clouds.ModuleBase
{
public class ModuleInstanceFactory
{
public static IModule CreateInstance(string contractName)
{
return CompositionContainerSingleton.Instance.
   GetExportedValues<IModule>().SingleOrDefault(x => x.GetType
   ().FullName == contractName);
```

```
    }

    public static IEnumerable<Type> GetExportedTypes()
    {
    return CompositionContainerSingleton.Instance.Catalog.Parts
    .Select(part => ComposablePartExportType<IModule>(part))
    .Where(t => t != null).ToList();
    }

    private static Type ComposablePartExportType<T>(
       ComposablePartDefinition part)
    {

    if (part.ExportDefinitions.Any(
    def => def.Metadata.ContainsKey("ExportTypeIdentity") &&
    def.Metadata["ExportTypeIdentity"].Equals(typeof(T).FullName))
       )
    {
    return ReflectionModelServices.GetPartType(part).Value;
    }
    return null;
    }
    }
    }

    using Api.Clouds.ModuleBase;
    using Box.V2;
    using Box.V2.Config;
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.Composition;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows;
    using System.Linq;
    using Box.V2.Auth;
    using Box.V2.Models;
    using System.IO;
    using NLog;
    using System.Threading;
    using System.ComponentModel;

```

```
namespace Api.Clouds.Modules.Box
{
[Export(typeof(IModule)), PartCreationPolicy(CreationPolicy.
   NonShared)]
[ExportModule(ModuleName = "Box", AuthenticationType =
   AuthenticationType.OAuth)]
class ModuleMain : IModule, INotifyPropertyChanged
{
private readonly Logger _logger = LogManager.
   GetCurrentClassLogger();
BoxClient Client;
OAuthSession session = null;
Thread SessionMaintainer;


#region Properties
private bool _IsAuthenticated;
public bool IsAuthenticated
{
get
{
return _IsAuthenticated;
}
set
{
if (_IsAuthenticated == value) return;
_IsAuthenticated = value;
if (PropertyChanged != null)
{
PropertyChanged(this, new PropertyChangedEventArgs("
   IsAuthenticated"));
}
}
}
public string ModuleName
{
get
{
return "Box";
}
set
{
```

```
1175    }
        }
        public string IconName
        {
        get
1180    {
        return AppDomain.CurrentDomain.BaseDirectory+@"Resources\
           Images\box-icon.png";
        }
        }
        #endregion
1185
        #region Sync
        /// <summary>
        /// Синхронная выгрузка файла
        /// </summary>
1190    /// <param name="stream">Содержимое файла</param>
        /// <param name="fileName">Имя файла</param>
        public void Upload(Stream stream, string fileName)
        {
        _logger.Info("Box module: Upload file started. FileName: " +
           fileName);
1195    if (!IsAuthenticated)
        {
        _logger.Warn("Box module: Upload file canceled. Not
           authenticated.");
        return;
        }
1200
        BoxFileRequest request = new BoxFileRequest()
        {
        Name = fileName,
        Parent = new BoxRequestEntity() { Id = "0" },
1205    };

        var folderItems = Client.FoldersManager.GetFolderItemsAsync("0
           ", 50).Result;
        var oldFile = folderItems.Entries.SingleOrDefault(x => x.Name.
           Equals(fileName));
        if (oldFile != null)
1210    {
        var deleted = Client.FilesManager.DeleteAsync(oldFile.Id).
           Result;
```

```
    if (deleted != true)
    {
    _logger.Error("Box module: deletetion file failed! FileName: "
        + fileName);
    return;
    }
    }

    try
    {
    var file = Client.FilesManager.UploadAsync(request, stream).
       Result;
    }
    catch (Exception ex)
    {
    _logger.Error("Box module: upload failed. FileName: " +
       fileName, ex);
    }
    _logger.Info("Box modules: File uploaded! FileName: " +
       fileName);
    }
    /// <summary>
    /// Синхронная загрузка файла
    /// </summary>
    /// <param name="outStream">Поток для записи файла</param>
    /// <param name="fileName">Имя файла</param>
    public void Download(Stream outStream, string fileName)
    {
    _logger.Info("Box module: File download start. FileName: " +
       fileName);
    if (!IsAuthenticated)
    {
    _logger.Warn("Box module: File download canceled. Not
       authenticated!");
    return;
    }

    var folderItems = Client.FoldersManager.GetFolderItemsAsync("0
       ", 50).Result;
    var file = folderItems.Entries.SingleOrDefault(x => x.Name.
       Equals(fileName));
    if (file == null)
    {
```

```
_logger.Warn("Box module: File download canceled. File not
   found!");
return;
}
else
{
try
{
(Client.FilesManager.DownloadStreamAsync(file.Id).Result).
   CopyTo(outStream);
}
catch (Exception ex)
{
_logger.Error("Box module: File download error!", ex);
return;
}
}
}
/// <summary>
/// Запрос списка файлов
/// </summary>
/// <returns>список имен файлов</returns>
public List<string> GetFiles()
{
if(!IsAuthenticated)
{
_logger.Warn("Box module: GetFiles failed. Not Authenticated."
   );
return null;
}

List<BoxItem> folderItems = null;
try
{
folderItems = Client.FoldersManager.GetFolderItemsAsync("0",
   50).Result.Entries;
} catch(Exception ex)
{
_logger.Error("Box module: GetFiles method. Error getting
   files list.", ex);
return null;
}
```

```
1285    List<string> result = new List<string>();
        folderItems.ForEach(x=> result.Add(x.Name));

        return result;
        }
1290    #endregion

        #region Async
        /// <summary>
        /// Асинхронная выгрузка файла
1295    /// </summary>
        /// <param name="stream">Содержимое файла</param>
        /// <param name="fileName">Имя файла</param>
        public async Task UploadFileAsync(Stream stream, string
           fileName)
        {
1300    _logger.Info("Box module: Async upload started. FileName: " +
           fileName);
        if (!IsAuthenticated)
        {
        _logger.Warn("Box module: Async upload canceled. Not
           authenticated!");
        if (this.UploadProgressComplete != null)
1305    {
        this.UploadProgressComplete(this, new OperationResult() {
           Result = OperationResultEnum.ERROR, FileName = fileName });
        }
        return;
        }
1310
        BoxFileRequest request = new BoxFileRequest() //Создаем запрос
        {
        Name = fileName,
        Parent = new BoxRequestEntity() { Id = "0" },
1315    };

        var folderItems = await Client.FoldersManager.
           GetFolderItemsAsync("0", 50); //Запрашиваем список файлов с
            сервера
        var oldFile = folderItems.Entries.SingleOrDefault(x => x.Name.
           Equals(fileName)); //Проверяем, занято ли имя
        if (oldFile != null)
1320    {
```

```
    var deleted = await Client.FilesManager.DeleteAsync(oldFile.Id
       );
    if (!deleted)
    {
    _logger.Warn("Box module: Async upload canceled. File exists,
       but not deleted. FileName: " + fileName);
    if (this.UploadProgressComplete != null)
    {
    this.UploadProgressComplete(this, new OperationResult() {
       Result = OperationResultEnum.ERROR, FileName = fileName });
    }
    return;
    }
    }

    try
    {
    BoxFile boxFile = await Client.FilesManager.UploadAsync(
       request, stream);
    }
    catch (Exception ex)
    {
    _logger.Error("Box module: Async upload failed!", ex);
    if (this.UploadProgressComplete != null)
    {
    this.UploadProgressComplete(this, new OperationResult() {
       Result = OperationResultEnum.ERROR, FileName = fileName });
    }
    }

    _logger.Info("Box module: Async upload complete! FileName: " +
        fileName);
    if (this.UploadProgressComplete != null)
    {
    this.UploadProgressComplete(this, new OperationResult() {
       Result = OperationResultEnum.OK, FileName = fileName });
    }
    }
    /// <summary>
    /// Асинхронная загрузка файла
    /// </summary>
    /// <param name="outStream">Поток для записи файла</param>
    /// <param name="fileName">Имя файла</param>
```

```
public async Task DownloadFileAsync(Stream outStream, string
   fileName)
{
if (fileName == null)
throw new ArgumentException("fileName");
if (!IsAuthenticated)
{
_logger.Warn("Box module: Async download canceled. Not
   authenticated!");
if (this.DownloadProgressComplete != null)
{
this.DownloadProgressComplete(this, new OperationResult() {
   Result = OperationResultEnum.ERROR, FileName = fileName });
}
return;
}

var folderItems = await Client.FoldersManager.
   GetFolderItemsAsync("0", 50);
var file = folderItems.Entries.SingleOrDefault(x => x.Name.
   Equals(fileName));
if (file == null)
{
_logger.Warn("Box module: Async download failed. File not
   found!");
if (this.DownloadProgressComplete != null)
{
this.DownloadProgressComplete(this, new OperationResult() {
   Result = OperationResultEnum.ERROR, FileName = fileName });
}
return;
}

try
{
(await Client.FilesManager.DownloadStreamAsync(file.Id)).
   CopyTo(outStream);
}
catch (Exception ex)
{
_logger.Error("Box module: Async download failed.", ex);
if (this.DownloadProgressComplete != null)
{
```

```
    this.DownloadProgressComplete(this, new OperationResult() {
       Result = OperationResultEnum.ERROR, FileName = fileName });
    }
    return;
    }

    _logger.Info("Box module: Async download complete.");
    if (this.DownloadProgressComplete != null)
    {
    this.DownloadProgressComplete(this, new OperationResult() {
       Result = OperationResultEnum.OK, FileName = fileName });
    }
    }
    #endregion

    #region Auth
    /// <summary>
    /// Запуск процесса авторизации
    /// </summary>
    /// <returns></returns>
    public async Task Authenticate()
    {
    await Task.Run(() =>
    {
    _logger.Info("Box module: Auth start.");
    var config = new BoxConfig("od301ujquxljs8grwvse15gf1tvem91r",
        "CffDanRRe5OwthHM6EdWAvID79QhDCRD", new Uri("https://
       youcantgetthisninjas.com"));
    Client = new BoxClient(config);
    _logger.Info("Box module: Client created.");

    Application.Current.Dispatcher.Invoke(() =>
    {
    LoginScreen ls = new LoginScreen(config.AuthCodeUri);
    ls.AuthComplete += AuthCodeReceived;
    ls.Show();
    });
    _logger.Info("Box module: login screen shown.");
    });
    }
    /// <summary>
    /// Not implemented
    /// </summary>
```

```
        /// <param name="accountInfo"></param>
        /// <returns></returns>
        public async Task Authenticate(AccountInfo accountInfo)
        {
1435    await Authenticate();
        }
        private async void AuthCodeReceived(object sender, string
           authCode)
        {
        _logger.Info("Box module: auth code recieved: " + authCode);
1440
        try
        {
        session = await Client.Auth.AuthenticateAsync(authCode);
        _logger.Info("Box module: session created. Expires in " +
           session.ExpiresIn);
1445    }
        catch (Exception ex)
        {
        _logger.Error("Box module: session creation failed.", ex);
        }
1450    finally
        {
        if (session != null)
        {
        IsAuthenticated = true;
1455    if (AuthComplete != null)
        this.AuthComplete(this, new AccountInfo());

        SessionMaintainer = new Thread(new ThreadStart(MaintainSession
           ));
        }
1460    }
        }
        private void MaintainSession()
        {
        if (!IsAuthenticated)
1465    {
        _logger.Warn("Box module: Account maintainer failed. Not
           authenticated.");
        }
        if (session == null)
        {
```

```
1470        _logger.Warn("Box module: Account maintainer failed. Session
               null.");
            }

            Thread.Sleep(session.ExpiresIn - 100);
            session = Client.Auth.RefreshAccessTokenAsync(session.
               RefreshToken).Result;
1475        _logger.Warn("Box module: Account maintainer: session updated.
               ");
            }
            #endregion

            #region Events
1480        /// <summary>
            /// Событие возникает при успешном завершении авторизации
            /// </summary>
            public EventHandler<AccountInfo> AuthComplete { get; set; }
            /// <summary>
1485        /// Not implemented
            /// </summary>
            public EventHandler<OperationProgressChangedEventArgs>
               UploadProgressChanged { get; set; }
            /// <summary>
            /// Not implemented
1490        /// </summary>
            public EventHandler<OperationResult> UploadProgressComplete {
               get; set; }
            /// <summary>
            /// Not implemented
            /// </summary>
1495        public EventHandler<OperationProgressChangedEventArgs>
               DownloadProgressChanged { get; set; }
            /// <summary>
            /// Not implemented
            /// </summary>
            public EventHandler<OperationResult> DownloadProgressComplete
               { get; set; }
1500        #endregion

            #region IModule
            public void Dispose()
            {
1505        SessionMaintainer.Abort();
```

```
    }
    #endregion

    public event PropertyChangedEventHandler PropertyChanged;
1510 }
    }

    using System;
    using System.Threading.Tasks;
1515 using Api.Clouds.ModuleBase;
    using System.ComponentModel.Composition;
    using DropNet;
    using DropNet.Models;
    using System.Windows.Threading;
1520 using System.Windows;
    using System.IO;
    using NLog;
    using System.Collections.Generic;
    using System.ComponentModel;
1525
    namespace Api.Clouds.Modules.DropBox
    {
    [Export(typeof(IModule)), PartCreationPolicy(CreationPolicy.
       NonShared)]
    [ExportModule(ModuleName = "DropBox", AuthenticationType =
       AuthenticationType.OAuth)]
1530 class ModuleMain : IModule, INotifyPropertyChanged
    {
    private static Logger _logger = LogManager.
       GetCurrentClassLogger();
    private DropNetClient Client;

1535 #region Properties
    private bool _IsAuthenticated;
    public bool IsAuthenticated
    {
    get
1540 {
    return _IsAuthenticated;
    }
    set
    {
1545 if (_IsAuthenticated == value) return;
```

```
        _IsAuthenticated = value;
        if (PropertyChanged != null)
        {
        PropertyChanged(this, new PropertyChangedEventArgs("
            IsAuthenticated"));
        }
        }
        }
        public string ModuleName
        {
        get
        {
        return "DropBox";
        }
        set
        {

        }
        }
        public string IconName
        {
        get
        {
        return AppDomain.CurrentDomain.BaseDirectory + @"Resources\
            Images\dropbox-icon.png";
        }
        }
        #endregion

        #region IModule
        [ImportingConstructor]
        public ModuleMain()
        {
        }
        public void Dispose()
        {
        throw new NotImplementedException();
        }
        #endregion

        #region Sync
        public void Upload(Stream stream, string fileName)
        {
```

```
_logger.Info("DropBox module: Upload start. FileName: " +
   fileName);
if (!IsAuthenticated) return;

try
{
var metaData = Client.Delete(fileName);
_logger.Warn("DropBox module: File already exists. Deleting...
    FileName: " + fileName);
}
catch (Exception ex)
{

}

try
{
using (MemoryStream mc = new MemoryStream())
{
stream.CopyTo(mc);
Client.UploadFile("/", fileName, mc.ToArray());
}
}
catch (DropNet.Exceptions.DropboxException ex)
{
_logger.Error("DropBox module: Upload failed! FileName" +
   fileName, ex);
return;
}
_logger.Info("DropBox module: File uploaded! FileName: " +
   fileName);
}
public void Download(Stream outStream, string fileName)
{
_logger.Info("DropBox module: Download file started. FileName:
    " + fileName);
if (!IsAuthenticated) return;

try
{
var bytes = Client.GetFile("/" + fileName);
outStream.Write(bytes, 0, bytes.Length);
```

```
1625    }
        catch (DropNet.Exceptions.DropboxException ex)
        {
        _logger.Error("DropBox module: Download file failed! FileName:
            " + fileName, ex);
        if (this.DownloadProgressComplete != null)
1630    {
        this.DownloadProgressComplete(this, new OperationResult() {
           Result = OperationResultEnum.ERROR, FileName = fileName });
        }
        return;
        }
1635    _logger.Info("DropBox module: File downloaded." + fileName);
        if (this.DownloadProgressComplete != null)
        {
        this.DownloadProgressComplete(this, new OperationResult() {
           Result = OperationResultEnum.OK, FileName = fileName });
        }
1640    }
        public List<string> GetFiles()
        {
        if (!IsAuthenticated)
        {
1645    _logger.Warn("DropBox module: GetFiles failed. Not
           authenticated.");
        return null;
        }
        List<MetaData> metaDataContents = null;

1650    try
        {
        metaDataContents = Client.GetMetaData("/").Contents;
        }
        catch (Exception ex)
1655    {
        _logger.Error("DropBox module: GetFiles method. Error getting
           files list.", ex);
        return null;
        }

1660    List<string> result = new List<string>();
        metaDataContents.ForEach(x => result.Add(x.Name));
```

```
        return result;
        }
1665    #endregion

        #region Async
        public async Task UploadFileAsync(Stream stream, string
            fileName)
        {
1670    if (!IsAuthenticated) return;

        await Task.Run(() =>
        {
        Upload(stream, fileName);
1675
        if (this.UploadProgressComplete != null)
        {
        this.UploadProgressComplete(this, new OperationResult() {
            Result = OperationResultEnum.OK, FileName = fileName });
        }
1680    });
        }
        public async Task DownloadFileAsync(Stream outStream, string
            fileName)
        {
        if (!IsAuthenticated) return;
1685    await Task.Run(() =>
        {
        _logger.Info("DropBox module: Download file started. FileName:
            " + fileName);

        Download(outStream, fileName);
1690
        //Client.GetFileAsync("/", s =>
        //{
        //    using (MemoryStream mc = new MemoryStream(s.RawBytes))
        //    {
1695    //        mc.CopyTo(outStream);
        //    }
        //    _logger.Info("DropBox module: File downloaded. FileName"
            + fileName);
        //    this.DownloadProgressComplete(this, OperationResult.OK);
        //},
1700    //f =>
```

```
        //{
        //    _logger.Error("DropBox module: File download failed!
            FileName" + fileName, f);
        //    this.DownloadProgressComplete(this, OperationResult.
            ERROR);
        //});
1705    });
        }
        #endregion

        #region Authentication
1710    public async Task Authenticate(Api.Clouds.ModuleBase.
            AccountInfo accountInfo)
        {
        await Task.Run(() =>
        {
        Client = new DropNetClient("yjujusp8utymm3v", "bc5bkzaeo5sya0t
            ");
1715    Client.UseSandbox = true;
        Client.UserLogin = new UserLogin { Token = (accountInfo as
            AccountDropBox).Token, Secret = (accountInfo as
            AccountDropBox).Secret };
        IsAuthenticated = true;
        this.AuthComplete(this, accountInfo);
        });
1720    }
        public async Task Authenticate()
        {
        await Task.Run(() =>
        {
1725    Client = new DropNetClient("yjujusp8utymm3v", "bc5bkzaeo5sya0t
            ");
        Client.UseSandbox = true;
        Client.GetToken();
        var url = Client.BuildAuthorizeUrl();

1730    Application.Current.Dispatcher.Invoke(() =>
        {
        LoginScreen ls = new LoginScreen(url);
        ls.Show();
        ls.AuthCompleted += ls_AuthCompleted;
1735    });
        });
```

```
}

void ls_AuthCompleted(object sender, string e)
{
var accessToken = Client.GetAccessToken();
var accInfo = new AccountDropBox() { Token = accessToken.Token
   , Secret = accessToken.Secret };
IsAuthenticated = true;
if (this.AuthComplete != null)
{
this.AuthComplete(this, accInfo);
}
}
#endregion

#region Events
/// <summary>
/// Событие возникает при успешном завершении авторизации
/// </summary>
public EventHandler<Api.Clouds.ModuleBase.AccountInfo>
   AuthComplete { get; set; }
/// <summary>
/// Not implemented
/// </summary>
public EventHandler<OperationProgressChangedEventArgs>
   UploadProgressChanged { get; set; }
/// <summary>
/// Not implemented
/// </summary>
public EventHandler<OperationResult> UploadProgressComplete {
   get; set; }
/// <summary>
/// Not implemented
/// </summary>
public EventHandler<OperationProgressChangedEventArgs>
   DownloadProgressChanged { get; set; }
/// <summary>
/// Not implemented
/// </summary>
public EventHandler<OperationResult> DownloadProgressComplete
   { get; set; }
#endregion
```

```
    public event PropertyChangedEventHandler PropertyChanged;
1775 }
    }
```

Приложение Б

Аппаратная реализация многоуровневой дизъюнктивной схемы разделения секрета

```
----------------------------------------------------------------

-- Create Date: 19.05.2015 21:52:34
-- Module Name: root - Behaviora

----------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std;
use IEEE.numeric_bit;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity root is
Port ( secret : in bit_vector  (0 to 511);
N : in integer range 0 to 65535;
share1 : out bit_vector  (0 to 511);
share2 : out bit_vector  (0 to 511);
share3 : out bit_vector  (0 to 511));
end root;

architecture Behavioral of root is

begin
process Share(secret, N)
variable Temp : bit_vector  (511 downto 0);
```

```
variable S1 : bit_vector  (511 downto 0);
variable S2 : bit_vector  (511 downto 0);
variable S3 : bit_vector  (511 downto 0);
begin

if (secret >= Temp) then
S2 := secret & Temp;
S1 := S2 and (secret ror N);

if (Temp <= S1) then
S1 := S1 - Temp;
end if;

Temp := 1 rol (n+1) -1;

S3 := (Temp & secret) + (secret ror (n+1));

if (Temp <= S3) then
S3 := S3 - Temp;
end if;
else
S1 := secret;
S2 := secret;
S3 := secret;
end if;

share1 <= S1;
share2 <= S2;
share3 <= S3;
end process;

end Behavioral;


‘timescale 1ns / 1ps


module root(
input [0:63] secret,
output s1_1,
output s1_2,
output s1_3,
output s2_1,
```

```
output s2_2,
output s2_3,
output s3_1,
output s3_2,
output s3_3
);

parameter n=32;

function [0:63] split;
input [0:63] value;
input [0:16] split_param;
input [0:3] share_number;
begin
if (share_number == 1)
split = value % ( (2**split_param)-1 );
else if (share_number == 2)
split = value % ( 2**split_param );
else
split = value % ( (2 ** (split_param+1) ) -1 );
end
endfunction

reg [0:63] s1, s2, s3;
reg [0:16] m12, m3;
reg [0:32] s1_1, s1_2, s1_3, s2_1, s2_2, s2_3, s3_1, s3_2, s3_3;

always @(secret)
begin
s1 = split(secret, n, 1);
s2 = split(secret, n, 2);
s3 = split(secret, n, 3);
end

always @(s1)
begin
m12 = (n+1)/2;

s1_1 = split(s1, m12, 1);
s1_2 = split(s1, m12, 2);
s1_3 = split(s1, m12, 3);
end
```

```
always @(s2)
begin
m12 = (n+1)/2;

s2_1 = split(s2, m12, 1);
s2_2 = split(s2, m12, 2);
s2_3 = split(s2, m12, 3);
end

always @(s3)
begin
m3 = (n+2)/2;

s3_1 = split(s3, m3, 1);
s3_2 = split(s3, m3, 2);
s3_3 = split(s3, m3, 3);
end

endmodule
```

Приложение В

Аппаратная реализация узлов контроля работы облачных узлов

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
library UNISIM;
use UNISIM.VComponents.all;

entity piosk_ohibki is
port (
A1 :in std_logic_vector ( 54 downto 0);
A2  :in std_logic_vector ( 54 downto 0);
A3  :in std_logic_vector ( 54 downto 0);
A4  :in std_logic_vector ( 54 downto 0);
A5  :in std_logic_vector ( 54 downto 0);
a :out bit
);

end piosk_ohibki;

architecture Behavioral of piosk_ohibki is
constant P_rab  :unsigned ( 216 downto 0) := "100000000000000
0000000000000000000000000000011010101100000000000000000000000
0000000000000001111111110011010000000000000000000000000000000
1000001111010111101000101000000000000000000000000110001110101
0010011101110110111";
constant P_poln :unsigned ( 270 downto 0) := "100000000000000
0000000000000000000000000000010011001011000000000000000000000
0000000000001000110110101011011000000000000000000000000000001
1111100001000011000001001000000000000000000000110110010000001
1011111101011001001000000000000100100010011010110010100000000
0001101000011";
```

```
constant B1    :unsigned ( 269 downto 0) := "10011111010111
000011000011110000001101010100001101000111010100100111000010101
000011110000110001110111110011101000111011101011001001000
000110001010001110110000011000111001001101001000010101101101100
001110010111000010010110101111011100100101101010000011110011
110111010101";
constant B2    :unsigned ( 269 downto 0) := "110011111001011
001010000010000000111000111001101100000110010001011110110001011
111011000110110100100010000001001000110100001110111000110110101
100111010011010000100010000011111010111100010000111011101101011
101110101100100010111100001111011000110100001000010101000011000
0000010";
constant B3    :unsigned ( 269 downto 0) := "101011101111101
001000001000010000100111110110101010011110110001010100000110000
111000011101100111000101101111101000110101100010111001100110010
010010111000011000111111100011110000001100001011010011000111010
010010001010010100001011110110100111011101110000010110100001010
1100101";
constant B4    :unsigned ( 269 downto 0) := "111100010001000
010000000000101010000001100100001100001100010101101100101001000
001100100101011001010011100010001010000101000100111110010100100
100010000001001101100000011001110111101010101001101010100011010
000111001000101011010000110011010110111111111110111001001000000
1110001";
constant B5    :unsigned ( 269 downto 0) := "111100010000001
010111101101100100000011001100101000011111111110010010011001111
111100001001110100100010100101011011010011010011100010110101111
101001001110101010010101101110100111001011000111110010101110000
000111110100010111111011010100000001011101001010110111011000011
1100000";
constant r     :unsigned ( 2 downto 0) := "101";
signal A_pol   :unsigned ( 324 downto 0) := (others =>'0');
--  signal Raz :unsigned ( 48 downto 0) := (others =>'0');
begin

--  A_pol <= ((B1*unsigned(A1)+B2*unsigned(A2)+B3*unsigned(A3)+B4*
   unsigned(A4)+B5*unsigned(A5))-r*P_poln);  -- 1
--  A_pol <= ((((B1*unsigned(A1))+(B2*unsigned(A2)))+(((B3*
   unsigned(A3))+(B4*unsigned(A4)))+(B5*unsigned(A5))))-r*P_poln);
     -- 2
--  A_pol <= (((B1*unsigned(A1))+(B2*unsigned(A2))+(B3*unsigned(A3
   )))+((B4*unsigned(A4))+(B5*unsigned(A5)))-r*P_poln);  -- 3
```

```
A_pol <= (((((B1*unsigned(A1))+(B2*unsigned(A2)))+((B3*unsigned(A3
   ))+(B4*unsigned(A4))))+(B5*unsigned(A5)))-r*P_poln);  -- 4

p0:process (A_pol)
begin
if (A_pol > P_poln) then a <='0';
else a <='1';
end if;
end process;

end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

--use work.CrtOperations.reducedMult;

entity crt_converter is
Port (
a1 : in  STD_LOGIC_VECTOR (5 downto 0);
a2 : in  STD_LOGIC_VECTOR (5 downto 0);
a3 : in  STD_LOGIC_VECTOR (5 downto 0);
a4 : in  STD_LOGIC_VECTOR (5 downto 0);
A  : out STD_LOGIC_VECTOR (44 downto 0)
);
constant bc:   natural := 52;
constant pbc:  natural := 6;     -- for a (inputs)
constant rbc:  natural := 45;    -- for A (output)

constant k1:  unsigned (bc-1 downto 0)     := "
   1010010100101001010010100101001010010100101001010011";
constant k2:  unsigned (bc-1 downto 0)     := "
   1111100100010100110000011011101011001111100100010101";
```

```
constant k3:  unsigned (bc-1 downto 0)     := "
   11111001000101001100000110111010110011111100100010101";
constant k4:  unsigned (bc-1 downto 0)     := "
   11100010001110111000100011101110001000111011100010001";
constant P :   unsigned (rbc-1 downto 0)    := "
   10000011111001111011001000110001000100100010 1";
end crt_converter;

architecture Behavioral of crt_converter is

signal ka1 :   unsigned (bc+pbc-1 downto 0) := (others => '0');
signal ka2 :   unsigned (bc+pbc-1 downto 0) := (others => '0');
signal ka3 :   unsigned (bc+pbc-1 downto 0) := (others => '0');
signal ka4 :   unsigned (bc+pbc-1 downto 0) := (others => '0');

signal ka12:   unsigned (bc-1 downto 0)     := (others => '0');
signal ka34:   unsigned (bc-1 downto 0)     := (others => '0');

signal A11 :   unsigned (bc+rbc-1 downto 0) := (others => '0');

begin

ka1  <= k1 * unsigned(a1);
ka2  <= k2 * unsigned(a2);
ka3  <= k3 * unsigned(a3);
ka4  <= k4 * unsigned(a4);

ka12 <= unsigned(ka1(bc-1 downto 0)) + unsigned(ka2(bc-1 downto 0)
   );
ka34 <= unsigned(ka3(bc-1 downto 0)) + unsigned(ka4(bc-1 downto 0)
   );

A11 <= (( ka12 + ka34 ) * P);

A <= std_logic_vector(A11(bc+rbc-1 downto bc));
end Behavioral;

----------------------------------------------------------------------

-- Company:
-- Engineer:
--
-- Create Date:    20:08:05 01/16/2015
```

```
-- Design Name:
-- Module Name:    mrc_converter - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity mrc_converter is
Port (
A1 :in std_logic_vector ( 16 downto 0);
A2  :in std_logic_vector ( 16 downto 0);
A3  :in std_logic_vector ( 16 downto 0);
A4  :in std_logic_vector ( 16 downto 0);
A5  :in std_logic_vector ( 16 downto 0);
--    a6 : in   STD_LOGIC_VECTOR (5 downto 0);
--    a7 : in   STD_LOGIC_VECTOR (5 downto 0);
--    a8 : in   STD_LOGIC_VECTOR (5 downto 0);
A  : out  bit
);

constant pbc1:  natural := 17;
constant pbc2:  natural := 17;
constant pbc3:  natural := 17;
```

```
constant pbc4:  natural := 17;
constant pbc5:  natural := 17;




constant p1  : unsigned (pbc1-1 downto 0)     := "
   10000110011010101";
constant p2  : unsigned (pbc2-1 downto 0)     := "
   10000110011111111";
constant p3  : unsigned (pbc3-1 downto 0)     := "
   10000110100001111";
constant p4  : unsigned (pbc4-1 downto 0)     := "
   10000110100010001";
constant p5  : unsigned (pbc5-1 downto 0)     := "
   10000110100011011";
--  constant p6  : unsigned (pbc-1 downto 0)     := "110101";
--  constant p7  : unsigned (pbc-1 downto 0)     := "111011";
--  constant p8  : unsigned (pbc-1 downto 0)     := "111101";
--
--constant b11 : unsigned (pbc-1 downto 0)     := "
   00000000000000000001";

constant b12 : unsigned (pbc2-1 downto 0)     := "
   00010000000000110";
constant b22 : unsigned (pbc2-1 downto 0)     := "
   01101101010010101";

constant b13 : unsigned (pbc3-1 downto 0)     := "
   00011000001111100";
constant b23 : unsigned (pbc3-1 downto 0)     := "
   00100010111100100";
constant b33 : unsigned (pbc3-1 downto 0)     := "
   01110110011111001";
--
constant b14 : unsigned (pbc4-1 downto 0)     := "
   01010011000101000";
constant b24 : unsigned (pbc4-1 downto 0)     := "
   01100101110001110";
constant b34 : unsigned (pbc4-1 downto 0)     := "
   00001000011100001";
constant b44 : unsigned (pbc4-1 downto 0)     := "
   01001100101100000";
```

```
constant b15 : unsigned (pbc5-1 downto 0)     := "
   00100110010110000";
constant b25 : unsigned (pbc5-1 downto 0)     := "
   00000011001100100";
constant b35 : unsigned (pbc5-1 downto 0)     := "
   01101000101100110";
constant b45 : unsigned (pbc5-1 downto 0)     := "
   01000000010010101";
constant b55 : unsigned (pbc5-1 downto 0)     := "
   00010111101011011";

constant rc2 : unsigned (pbc1 - 1 downto 0)   := "
   10000110011010101";
constant rc3 : unsigned (pbc1+pbc2 - 2 downto 0) := "
   10001101001111010110001000101011";
constant rc4 : unsigned (pbc1+pbc2+pbc3 - 3 downto 0) := "
   10010100011100011100011010101001010110110000101";
constant rc5 : unsigned (pbc1+pbc2+pbc3+pbc4 - 3 downto 0) :=
"01001110000000101011010010100111000100001010
10100100100011011010101";

constant con_sr :unsigned (3 downto 0):= (others =>'0');

end mrc_converter;

architecture Behavioral of mrc_converter is

signal   aa1  : unsigned (2*pbc1-1 downto 0)          := (others
   => '0');

signal   ab21 : unsigned (pbc1+pbc2-1 downto 0)          := (
   others => '0');
signal   ab22 : unsigned (2*pbc2-1 downto 0)          := (others
   => '0');
signal   aa2  : unsigned (2*pbc2-1 downto 0)          := (others
   => '0');

signal   ab31 : unsigned (pbc3+pbc1-1 downto 0)          := (
   others => '0');
signal   ab32 : unsigned (pbc3+pbc2-1 downto 0)          := (
   others => '0');
```

```
signal   ab33 : unsigned (2*pbc3-1 downto 0)          := (others
   => '0');
signal   aa31 : unsigned (pbc3+pbc2-1 downto 0)          := (
   others => '0');
signal   aa32 : unsigned (2*pbc3-1 downto 0)          := (others
   => '0');
signal   aa3  : unsigned (2*pbc3-1 downto 0)          := (others
   => '0');
--
signal   ab41 : unsigned (pbc4+pbc1-1 downto 0)          := (
   others => '0');
signal   ab42 : unsigned (pbc4+pbc2-1 downto 0)          := (
   others => '0');
signal   ab43 : unsigned (pbc4+pbc3-1 downto 0)          := (
   others => '0');
signal   ab44 : unsigned (pbc4+pbc4-1 downto 0)          := (
   others => '0');
signal   aa41 : unsigned (pbc4+pbc2-1 downto 0)          := (
   others => '0');
signal   aa42 : unsigned (2*pbc4-1 downto 0)          := (others
   => '0');
signal   aa4  : unsigned (2*pbc4-1 downto 0)          := (others
   => '0');
--
signal   ab51 : unsigned (pbc5+pbc1-1 downto 0)          := (
   others => '0');
signal   ab52 : unsigned (pbc5+pbc2-1 downto 0)          := (
   others => '0');
signal   ab53 : unsigned (pbc5+pbc3-1 downto 0)          := (
   others => '0');
signal   ab54 : unsigned (pbc5+pbc4-1 downto 0)          := (
   others => '0');
signal   ab55 : unsigned (2*pbc5-1 downto 0)          := (others
   => '0');
signal   aa51 : unsigned (pbc5+pbc2-1 downto 0)          := (
   others => '0');
signal   aa52 : unsigned (pbc5+pbc4-1 downto 0)          := (
   others => '0');
signal   aa53 : unsigned (2*pbc5-1 downto 0)          := (others
   => '0');
signal   aa5  : unsigned (2*pbc5-1 downto 0)          := (others
   => '0');
```

```
signal          t2  : unsigned (pbc2-1 downto 0)      := (others
   => '0');
signal          t3  : unsigned (pbc3-1 downto 0)      := (others
   => '0');
signal          t4  : unsigned (pbc4-1 downto 0)      := (others
   => '0');

signal          mm1 : unsigned (pbc1-1 downto 0)      := (others
   => '0');
signal          mm2 : unsigned (pbc2-1 downto 0)      := (others
   => '0');
signal          mm3 : unsigned (pbc3-1 downto 0)      := (others
   => '0');
signal          mm4 : unsigned (pbc4-1 downto 0)      := (others
   => '0');
signal          mm5 : unsigned (pbc5-1 downto 0)      := (others
   => '0');

signal           w1 : unsigned (pbc5+pbc4+pbc2+pbc1+pbc3-3 downto
   0)    := (others => '0');
signal           w2 : unsigned (pbc4+pbc2+pbc1+pbc3-3 downto 0)
      := (others => '0');




function MAX (LEFT, RIGHT: INTEGER) return INTEGER is
begin
if LEFT > RIGHT then return LEFT;
else return RIGHT;
end if;
end MAX;
-- this internal procedure computes UNSIGNED division
-- giving the quotient and remainder.
procedure DIVMOD (signal NUM : unsigned; XDENOM: UNSIGNED; signal
   XQUOT, XREMAIN: out UNSIGNED) is
variable TEMP: UNSIGNED(NUM'LENGTH downto 0);
variable QUOT: UNSIGNED(MAX(NUM'LENGTH, XDENOM'LENGTH)-1 downto 0)
   ;
alias DENOM: UNSIGNED(XDENOM'LENGTH-1 downto 0) is XDENOM;
variable TOPBIT: INTEGER;
begin
TEMP := "0"&NUM;
```

```
QUOT := (others => '0');
TOPBIT := -1;
for J in DENOM'RANGE loop
if DENOM(J)='1' then
TOPBIT := J;
exit;
end if;
end loop;
assert TOPBIT >= 0 report "DIV, MOD, or REM by zero" severity
   ERROR;

for J in NUM'LENGTH-(TOPBIT+1) downto 0 loop
if TEMP(TOPBIT+J+1 downto J) >= "0"&DENOM(TOPBIT downto 0) then
TEMP(TOPBIT+J+1 downto J) := (TEMP(TOPBIT+J+1 downto J))
-("0"&DENOM(TOPBIT downto 0));
QUOT(J) := '1';
end if;
assert TEMP(TOPBIT+J+1)='0'
report "internal error in the division algorithm"
severity ERROR;
end loop;
XQUOT <= RESIZE(QUOT, XQUOT'LENGTH);
XREMAIN <= RESIZE(TEMP, XREMAIN'LENGTH);
end DIVMOD;
begin

mm1 <= unsigned(a1);

ab21 <= b12 * unsigned(a1);
ab22 <= b22 * unsigned(a2);
aa2  <= ab21 + ab22;

--  mm2 <= aa2 mod p2;

DIVMOD(aa2, p2, t2, mm2);

ab31 <= b13 * unsigned(a1);
ab32 <= b23 * unsigned(a2);
ab33 <= b33 * unsigned(a3);
aa31 <= ab31 + ab32;
aa32 <= ab33 + t2;
aa3  <= aa31 + aa32;
```

```
--  mm3 <= aa3 mod p3;

DIVMOD(aa3, p3, t3, mm3);

ab41 <= b14 * unsigned(a1);
ab42 <= b24 * unsigned(a2);
ab43 <= b34 * unsigned(a3);
ab44 <= b44 * unsigned(a4);
aa41 <= ab41 + ab42;
aa42 <= ab43 + ab44;
aa4  <= aa41 + aa42 + t3;
--
--  mm4 <= aa4 mod p4;
--
DIVMOD(aa4, p4, t4, mm4);

ab51 <= b15 * unsigned(a1);
ab52 <= b25 * unsigned(a2);
ab53 <= b35 * unsigned(a3);
ab54 <= b45 * unsigned(a4);
ab55 <= b55 * unsigned(a5);
aa51 <= ab51 + ab52;
aa52 <= ab53 + ab54;
aa53 <= ab55 + t4;
aa5  <= aa51 + aa52 + aa53;

mm5 <= aa5 mod p5;

--  w1 <= rc5*mm5 + mm1 + rc2*mm2;
--  w2 <= rc4*mm4+rc3*mm3;
--
--  A <=  std_logic_vector(w1 + w2);


p0:process (mm5)
begin
if (mm5 = con_sr) then a <='0';
else a <='1';
end if;
end process;


end Behavioral;
```

Приложение Г

Система контроля вычислений в облачной среде с использованием системы остаточных классов

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net.Sockets;
using System.Net;
using System.Threading;

namespace Client

{

class client
{
static void Main(string[] args)
{
ClientHelper helper = new ClientHelper();
helper.InitializeConnection();

}
}
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
```

```
namespace Client
{
class ClientHelper
{

IPEndPoint ipEndpoint = new IPEndPoint(IPAddress.Parse("
   192.168.173.1"), 80);
Socket socket;
private Thread listener, sender;
public string msg, name;


public void InitializeConnection()
{
try
{
socket = new Socket(ipEndpoint.AddressFamily, SocketType.Stream,
   ProtocolType.Tcp);
socket.Connect(ipEndpoint);
listener = new Thread(new ThreadStart(Listen));
listener.Start();
sender = new Thread(new ThreadStart(Send));
sender.Start();
}
catch (Exception ex)
{

}
}

public void Listen()
{
byte[] bytes = new byte[1024];
try
{
FileRecieve();
string input = name;
byte[] flag = new byte[1];
flag[0] = 1;
socket.Send(flag);
FileRecieve();
RunFile StartFile = new RunFile();
StartFile.RunerFile(input);
```

```
SendFile(input);
}
catch (Exception ex)
{
Console.WriteLine("Ошибка: " + ex.Message);
}
}

public void Send()
{
while (true)
{
Console.Write("> ");
msg = Console.ReadLine();
byte[] msgs = Encoding.UTF8.GetBytes(msg);
socket.Send(msgs);
}
}

public void FileRecieve()
{
name = null;
byte[] header = new byte[64];
socket.Receive(header);
Int64 size = 0;
FileHeader file = new FileHeader(name, size);
file = FileHeaderConverter.ConvertToHeader(header);
name = file.name;
using (var fs = new FileStream(name, FileMode.Create))
{

Int64 count;
int rebuffer;
while ((count = fs.Position) < file.size)
{
byte[] buffer = new byte[8192];
rebuffer = socket.Receive(buffer);
fs.Write(buffer, 0, rebuffer);
fs.Flush();
}
}
}
```

```
public void SendFile(string name)
{
try
{
long count = 0;
name = name + "_output.txt";
using (var fs = new FileStream(name, FileMode.Open))
{
FileHeader file = new FileHeader(name, fs.Length);
socket.Send(FileHeaderConverter.ConvertToByte(file));
int rebuffer;
while ((count) < file.size)
{
var buffer = new byte[8192];
if ((rebuffer = fs.Read(buffer, 0, 8192)) == 8192)
{
socket.Send(buffer);
count += 8192;
}
else
{
fs.Position = count;
var bufferEND = new byte[rebuffer];
fs.Read(bufferEND, 0, rebuffer);
socket.Send(bufferEND);
count += 8192;
}
}
}
}
catch (Exception ex)
{
Console.WriteLine("Ошибка: " + ex.Message);
}
}
}

public class FileHeader
{
public string name;
public Int64 size;
public FileHeader(string name, Int64 size)
{
```

```
this.name = name;
this.size = size;
}
}

public class FileHeaderConverter
{
public static byte[] ConvertToByte(FileHeader fileHeader)
{
byte[] data = new byte[64];

Encoding.ASCII.GetBytes(fileHeader.name)
.CopyTo(data, 0);

BitConverter.GetBytes(fileHeader.size)
.CopyTo(data, 32);

return data;
}

public static FileHeader ConvertToHeader(byte[] data)
{
string name = Encoding.ASCII.GetString(data, 0, 32);
name = name.Substring(0, name.IndexOf('\0'));
Int64 size = BitConverter.ToInt64(data, 32);

var header = new FileHeader(name, size);
return header;
}
}
}

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;



namespace Client
{
class RunFile
```

```
{
public void RunerFile(String file)
{
try
{
Process RunSendFile = new Process();
RunSendFile.StartInfo.FileName = "cmd";

RunSendFile.StartInfo.Arguments = @"/c test.exe<"+file+">"+file+"
   _output.txt & exit";
RunSendFile.Start();
RunSendFile.WaitForExit();
}
catch (UnauthorizedAccessException ex)
{
Console.WriteLine(ex);
}
}

void RunSendFile_Exited(object sender, EventArgs e)
{
Console.WriteLine("Child procces end");
}
}
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net.Sockets;
using System.Net;
using System.Threading;

namespace Server_v0._1a
{
class Program
{

static void Main(string[] args)
{
ServerHelper helper = new ServerHelper();
```

```
helper.InitiateTCPListener();
}
}
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Sockets;
using System.Net;
using System.Threading;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Diagnostics;

namespace Server_v0._1a
{
class ServerHelper
{
public int ClientCount;
public Socket serverSocket, clientSocket;
public Thread listener, sender;
IPEndPoint IpEndPoint = new IPEndPoint(IPAddress.Parse("
   192.168.173.1"), 80);
public void InitiateTCPListener()
{
serverSocket = new Socket(IpEndPoint.AddressFamily, SocketType.
   Stream, ProtocolType.Tcp);
serverSocket.Bind(IpEndPoint);
serverSocket.Listen(15);
ClientCount = 0;
while (true)
{
clientSocket = serverSocket.Accept();
ClientCount++;
var client = new Client(clientSocket, ClientCount);
client.ThreadAborted += ThreadAborted;
new Thread(new ThreadStart(client.SendClient)).Start();
}
}

```

```
public void ThreadAborted(object sender, EventArgs e)
{
ClientCount--;
}

public TcpClient ClientSocket { get; set; }
}

public class Client
{
private Socket socket;
private int number;
public EventHandler ThreadAborted;
public Client(Socket Socket, int Number)
{
this.socket = Socket;
this.number = Number;
}
public string name = null;

public void FileRecieve()
{
name = null;
byte[] header = new byte[64];
socket.Receive(header);
Int64 size = 0;
FileHeader file = new FileHeader(name, size);
file = FileHeaderConverter.ConvertToHeader(header);
name = file.name;
using (var fs = new FileStream(name, FileMode.Create))
{

Int64 count;
int rebuffer;
while ((count = fs.Position) < file.size)
{
byte[] buffer = new byte[8192];
rebuffer = socket.Receive(buffer);
fs.Write(buffer, 0, rebuffer);
fs.Flush();
}
}
}
```

```
public void SendFile(string name)
{
try
{
long count = 0;
using (var fs = new FileStream(name, FileMode.Open))
{
FileHeader file = new FileHeader(name, fs.Length);
socket.Send(FileHeaderConverter.ConvertToByte(file));
int rebuffer;
while ((count) < file.size)
{
var buffer = new byte[8192];
if ((rebuffer = fs.Read(buffer, 0, 8192)) == 8192)
{
socket.Send(buffer);
count += 8192;
}
else
{
fs.Position = count;
var bufferEND = new byte[rebuffer];
fs.Read(bufferEND, 0, rebuffer);
socket.Send(bufferEND);
count += 8192;
}
}
}
}
catch (Exception ex)
{
Console.WriteLine("Ошибка: " + ex.Message);
}
}

public void SendClient()
{
if (socket == null)
throw new InvalidOperationException("Сокет не может быть равен нул
   ю!");
try
{
```

```
name = "input" + number + ".txt";
Stopwatch stopWatch = Stopwatch.StartNew();
SendFile(name);
name = null;
byte[] flag = new byte[1];
socket.Receive(flag);

string name1 = "test.exe";
SendFile(name1);
FileRecieve();
stopWatch.Stop();
long frequency = Stopwatch.Frequency;
long ts =1000000000L* stopWatch.ElapsedTicks / frequency;
Console.WriteLine("\nВремя работы : {0}мс.", ts);
using (StreamWriter sw = new StreamWriter(File.Open("input" +
   number + ".txt_output.txt", FileMode.Append)))
{
sw.Write(ts);
}
Console.WriteLine(number+": Done");
}
catch (Exception ex)
{

socket.Close();

if (ThreadAborted != null)
{
this.ThreadAborted(this, new EventArgs());
}

Thread.CurrentThread.Abort();
}
}


}

public class FileHeader
{
public string name;
public Int64 size;
public FileHeader(string name, Int64 size)
{
```

```
this.name = name;
this.size = size;
}
}

public class FileHeaderConverter
{
public static byte[] ConvertToByte(FileHeader fileHeader)
{
byte[] data = new byte[64];

Encoding.ASCII.GetBytes(fileHeader.name)
.CopyTo(data, 0);

BitConverter.GetBytes(fileHeader.size)
.CopyTo(data, 32);

return data;
}

public static FileHeader ConvertToHeader(byte[] data)
{
string name = Encoding.ASCII.GetString(data, 0, 32);
name = name.Substring(0, name.IndexOf('\0'));
Int64 size = BitConverter.ToInt64(data, 32);

var header = new FileHeader(name, size);
return header;
}
}

public class ClientListener
{
public string data;
private Socket clientS;
public ClientListener(Socket socket)
{
clientS = socket;
}

public void Listen()
{
```

```
if (clientS == null)
throw new InvalidOperationException("ClientS не может быть null!")
    ;

ServerHelper help = new ServerHelper();
while (true)
{
try
{
data = null;
byte[] bytes = new byte[1024];
int bytesRec = clientS.Receive(bytes);//получение байтового сообще
    ния
data += Encoding.UTF8.GetString(bytes, 0, bytesRec);//перекодировк
    а в символы
Console.Write("\n" + clientS.RemoteEndPoint + ">> " + data + "\n>"
    );


}
catch (Exception ex)
{


}
}
}
}
}
```

Приложение Д

Результаты расчетов вероятностей отказа двухуровневой модели (на 1-м уровне применяется классическая СОК, на 2-м уровне применяется избыточная СОК)

Таблица 12 — Расчет вероятностей отказа двухуровневой модели (на 1-м уровне применяется классическая СОК, на 2-м уровне применяется избыточная СОК), WDC 6 TB

(k,n)	Количество модулей на 1-м слое						
	3	4	5	6	7	8	9
2,4	8,316E-06	1,11E-05	1,39E-05	1,66E-05	1,94E-05	2,22E-05	2,49E-05
3,4	0,0081296	0,010825	0,013513	0,016193	0,018866	0,021532	0,024191
2,5	9,3E-09	1,24E-08	1,55E-08	1,86E-08	2,17E-08	2,48E-08	2,79E-08
3,5	1,385E-05	1,85E-05	2,31E-05	2,77E-05	3,23E-05	3,69E-05	4,16E-05
4,5	0,0101516	0,013513	0,016862	0,0202	0,023527	0,026842	0,030147
3,6	1,89E-08	2,52E-08	3,15E-08	3,78E-08	4,41E-08	5,04E-08	5,67E-08
4,6	2,077E-05	2,77E-05	3,46E-05	4,15E-05	4,85E-05	5,54E-05	6,23E-05
5,6	0,0121695	0,016193	0,0202	0,024191	0,028165	0,032124	0,036066
4,7	3,3E-08	4,4E-08	5,5E-08	6,6E-08	7,7E-08	8,8E-08	9,9E-08
5,7	2,906E-05	3,88E-05	4,84E-05	5,81E-05	6,78E-05	7,75E-05	8,72E-05
6,7	0,0141833	0,018866	0,023527	0,028165	0,032782	0,037377	0,041949
5,8	5,28E-08	7,04E-08	8,8E-08	1,06E-07	1,23E-07	1,41E-07	1,58E-07
6,8	3,874E-05	5,16E-05	6,46E-05	7,75E-05	9,04E-05	0,000103	0,000116
7,8	0,016193	0,021532	0,026842	0,032124	0,037377	0,042601	0,047797
6,9	7,89E-08	1,05E-07	1,32E-07	1,58E-07	1,84E-07	2,1E-07	2,37E-07
7,9	4,978E-05	6,64E-05	8,3E-05	9,96E-05	0,000116	0,000133	0,000149
8,9	0,0181986	0,024191	0,030147	0,036066	0,041949	0,047797	0,053608

Таблица 13 — Расчет вероятностей отказа двухуровневой модели (на 1-м уровне применяется классическая СОК, на 2-м уровне применяется избыточная СОК), WDC 2 TB

(k,n)	Количество модулей на 1-м слое						
	3	4	5	6	7	8	9
2,4	0,000115	0,000153	0,000191	0,00023	0,000268	0,000306	0,000344
3,4	0,029941	0,039721	0,049402	0,058986	0,068473	0,077864	0,087161
2,5	4,84E-07	6,45E-07	8,07E-07	9,68E-07	1,13E-06	1,29E-06	1,45E-06
3,5	0,000191	0,000255	0,000318	0,000382	0,000446	0,000509	0,000573
4,5	0,037285	0,049402	0,061366	0,07318	0,084845	0,096364	0,107737
2,6	1,8E-09	2,4E-09	3E-09	3,6E-09	4,2E-09	4,8E-09	5,4E-09
3,6	9,66E-07	1,29E-06	1,61E-06	1,93E-06	2,25E-06	2,58E-06	2,9E-06
4,6	0,000286	0,000381	0,000477	0,000572	0,000667	0,000763	0,000858
5,6	0,044574	0,058986	0,07318	0,087161	0,10093	0,114492	0,127849
3,7	4,2E-09	5,6E-09	7E-09	8,4E-09	9,8E-09	1,12E-08	1,26E-08
4,7	1,69E-06	2,25E-06	2,81E-06	3,38E-06	3,94E-06	4,5E-06	5,06E-06
5,7	0,0004	0,000533	0,000666	0,000799	0,000933	0,001066	0,001199
6,7	0,051807	0,068473	0,084845	0,10093	0,116732	0,132257	0,147508

Продолжение таблицы 13

(k,n)	Количество модулей на 1-м слое						
	3	4	5	6	7	8	9
4,8	8,4E-09	1,12E-08	1,4E-08	1,68E-08	1,96E-08	2,24E-08	2,52E-08
5,8	2,69E-06	3,59E-06	4,49E-06	5,39E-06	6,29E-06	7,19E-06	8,08E-06
6,8	0,000532	0,000709	0,000887	0,001064	0,001241	0,001418	0,001596
7,8	0,058986	0,077864	0,096364	0,114492	0,132257	0,149665	0,166724
5,9	1,53E-08	2,04E-08	2,55E-08	3,06E-08	3,57E-08	4,08E-08	4,59E-08
6,9	4,03E-06	5,38E-06	6,72E-06	8,07E-06	9,41E-06	1,08E-05	1,21E-05
7,9	0,000683	0,000911	0,001138	0,001366	0,001593	0,00182	0,002048
8,9	0,06611	0,087161	0,107737	0,127849	0,147508	0,166724	0,185507

Таблица 14 — Расчет вероятностей отказа двухуровневой модели (на 1-м уровне применяется классическая СОК, на 2-м уровне применяется избыточная СОК), HGST 3 TB

(k,n)	Количество модулей на 1-м слое						
	3	4	5	6	7	8	9
2,4	5,06E-06	6,75E-06	8,44E-06	1,01E-05	1,18E-05	1,35E-05	1,52E-05
3,4	0,006347	0,008454	0,010557	0,012655	0,014748	0,016837	0,018922
2,5	4,5E-09	6E-09	7,5E-09	9E-09	1,05E-08	1,2E-08	1,35E-08
3,5	8,43E-06	1,12E-05	1,41E-05	1,69E-05	1,97E-05	2,25E-05	2,53E-05
4,5	0,007928	0,010557	0,013178	0,015793	0,018401	0,021002	0,023596
3,6	9E-09	1,2E-08	1,5E-08	1,8E-08	2,1E-08	2,4E-08	2,7E-08
4,6	1,26E-05	1,69E-05	2,11E-05	2,53E-05	2,95E-05	3,37E-05	3,79E-05
5,6	0,009506	0,012655	0,015793	0,018922	0,02204	0,025149	0,028248
4,7	1,56E-08	2,08E-08	2,6E-08	3,12E-08	3,64E-08	4,16E-08	4,68E-08
5,7	1,77E-05	2,36E-05	2,95E-05	3,54E-05	4,13E-05	4,72E-05	5,31E-05
6,7	0,011082	0,014748	0,018401	0,02204	0,025666	0,029279	0,032878
5,8	2,49E-08	3,32E-08	4,15E-08	4,98E-08	5,81E-08	6,64E-08	7,47E-08
6,8	2,36E-05	3,15E-05	3,93E-05	4,72E-05	5,5E-05	6,29E-05	7,08E-05
7,8	0,012655	0,016837	0,021002	0,025149	0,029279	0,033391	0,037485
6,9	3,75E-08	5E-08	6,25E-08	7,5E-08	8,75E-08	1E-07	1,13E-07
7,9	3,03E-05	4,04E-05	5,05E-05	6,06E-05	7,07E-05	8,08E-05	9,1E-05
8,9	0,014225	0,018922	0,023596	0,028248	0,032878	0,037485	0,042071

Таблица 15 — Расчет вероятностей отказа двухуровневой модели (на 1-м уровне применяется классическая СОК, на 2-м уровне применяется избыточная СОК), HGST 2 TB

(k,n)	Количество модулей на 1-м слое						
	3	4	5	6	7	8	9
2,4	5,148E-06	6,86E-06	8,58E-06	1,03E-05	1,2E-05	1,37E-05	1,54E-05
3,4	0,0064011	0,008526	0,010646	0,012761	0,014872	0,016979	0,019081
2,5	4,5E-09	6E-09	7,5E-09	9E-09	1,05E-08	1,2E-08	1,35E-08
3,5	8,578E-06	1,14E-05	1,43E-05	1,72E-05	2E-05	2,29E-05	2,57E-05
4,5	0,007995	0,010646	0,013289	0,015926	0,018556	0,021178	0,023794
3,6	9,3E-09	1,24E-08	1,55E-08	1,86E-08	2,17E-08	2,48E-08	2,79E-08
4,6	1,286E-05	1,71E-05	2,14E-05	2,57E-05	3E-05	3,43E-05	3,86E-05
5,6	0,0095863	0,012761	0,015926	0,019081	0,022225	0,02536	0,028484
4,7	1,62E-08	2,16E-08	2,7E-08	3,24E-08	3,78E-08	4,32E-08	4,86E-08
5,7	1,8E-05	2,4E-05	3E-05	3,6E-05	4,2E-05	4,8E-05	5,4E-05
6,7	0,0111751	0,014872	0,018556	0,022225	0,025881	0,029523	0,033152

Продолжение таблицы 15

(k,n)	Количество модулей на 1-м слое						
	3	4	5	6	7	8	9
5,8	2,58E-08	3,44E-08	4,3E-08	5,16E-08	6,02E-08	6,88E-08	7,74E-08
6,8	2,399E-05	3,2E-05	4E-05	4,8E-05	5,6E-05	6,4E-05	7,2E-05
7,8	0,0127613	0,016979	0,021178	0,02536	0,029523	0,033669	0,037797
6,9	3,84E-08	5,12E-08	6,4E-08	7,68E-08	8,96E-08	1,02E-07	1,15E-07
7,9	3,083E-05	4,11E-05	5,14E-05	6,17E-05	7,19E-05	8,22E-05	9,25E-05
8,9	0,014345	0,019081	0,023794	0,028484	0,033152	0,037797	0,042421

Таблица 16 — Расчет вероятностей отказа двухуровневой модели (на 1-м уровне применяется классическая СОК, на 2-м уровне применяется избыточная СОК), Toshiba 3 TB

(k,n)	Количество модулей на 1-м слое						
	3	4	5	6	7	8	9
2,4	2,47E-05	3,29E-05	4,11E-05	4,94E-05	5,76E-05	6,58E-05	7,41E-05
3,4	0,013974	0,018588	0,023181	0,027752	0,032302	0,036831	0,041338
2,5	4,83E-08	6,44E-08	8,05E-08	9,66E-08	1,13E-07	1,29E-07	1,45E-07
3,5	4,11E-05	5,48E-05	6,85E-05	8,22E-05	9,59E-05	0,00011	0,000123
4,5	0,017437	0,023181	0,028892	0,034569	0,040213	0,045824	0,051403
3,6	9,63E-08	1,28E-07	1,61E-07	1,93E-07	2,25E-07	2,57E-07	2,89E-07
4,6	6,16E-05	8,22E-05	0,000103	0,000123	0,000144	0,000164	0,000185
5,6	0,020887	0,027752	0,034569	0,041338	0,04806	0,054734	0,061362
3,7	3E-10	4E-10	5E-10	6E-10	7E-10	8E-10	9E-10
4,7	1,68E-07	2,24E-07	2,81E-07	3,37E-07	3,93E-07	4,49E-07	5,05E-07
5,7	8,62E-05	0,000115	0,000144	0,000172	0,000201	0,00023	0,000259
6,7	0,024326	0,032302	0,040213	0,04806	0,055842	0,063561	0,071216
4,8	3E-10	4E-10	5E-10	6E-10	7E-10	8E-10	9E-10
5,8	2,69E-07	3,59E-07	4,49E-07	5,39E-07	6,29E-07	7,18E-07	8,08E-07
6,8	0,000115	0,000153	0,000191	0,00023	0,000268	0,000306	0,000344
7,8	0,027752	0,036831	0,045824	0,054734	0,063561	0,072305	0,080967
5,9	6E-10	8E-10	1E-09	1,2E-09	1,4E-09	1,6E-09	1,8E-09
6,9	4,04E-07	5,38E-07	6,73E-07	8,07E-07	9,42E-07	1,08E-06	1,21E-06
7,9	0,000148	0,000197	0,000246	0,000295	0,000344	0,000393	0,000443
8,9	0,031167	0,041338	0,051403	0,061362	0,071216	0,080967	0,090616

Приложение Е

Результаты расчетов вероятностей отказа двухуровневой модели (на 1-м уровне применяется избыточная СОК, на 2-м уровне применяется классическая СОК)

Таблица 17 — Расчет вероятностей отказа двухуровневой модели (на 1-м уровне применяется классическая СОК, на 2-м уровне применяется избыточная СОК), WDC 6 TB

(k,n)	Количество модулей на 2-м слое						
	3	4	5	6	7	8	9
2,4	2,49E-05	4,41E-05	6,89E-05	9,9E-05	0,000135	0,000175	0,000222
3,4	0,00813	0,010825	0,013513	0,016193	0,018866	0,021532	0,024191
2,5	8,45E-08	2E-07	3,89E-07	6,72E-07	1,06E-06	1,59E-06	2,25E-06
3,5	4,14E-05	7,34E-05	0,000115	0,000165	0,000224	0,000291	0,000368
4,5	0,010152	0,013513	0,016862	0,0202	0,023527	0,026842	0,030147
2,6	2,58E-10	8,14E-10	1,98E-09	4,1E-09	7,58E-09	1,29E-08	2,06E-08
3,6	1,69E-07	3,99E-07	7,77E-07	1,34E-06	2,12E-06	3,16E-06	4,49E-06
4,6	6,2E-05	0,00011	0,000171	0,000246	0,000334	0,000435	0,00055
5,6	0,01217	0,016193	0,0202	0,024191	0,028165	0,032124	0,036066
2,7	7,37E-13	3,1E-12	9,42E-12	2,34E-11	5,04E-11	9,8E-11	1,76E-10
3,7	6,02E-10	1,9E-09	4,61E-09	9,54E-09	1,76E-08	3E-08	4,79E-08
4,7	2,95E-07	6,96E-07	1,36E-06	2,34E-06	3,7E-06	5,5E-06	7,81E-06
5,7	8,67E-05	0,000154	0,000239	0,000344	0,000466	0,000607	0,000767
6,7	0,014183	0,018866	0,023527	0,028165	0,032782	0,037377	0,041949
2,8	2E-15	1,12E-14	4,27E-14	1,27E-13	3,19E-13	7,09E-13	1,43E-12
3,8	1,96E-12	8,24E-12	2,51E-11	6,21E-11	1,34E-10	2,6E-10	4,67E-10
4,8	1,2E-09	3,78E-09	9,2E-09	1,9E-08	3,51E-08	5,97E-08	9,53E-08
5,8	4,71E-07	1,11E-06	2,16E-06	3,73E-06	5,9E-06	8,77E-06	1,24E-05
6,8	0,000115	0,000204	0,000318	0,000457	0,00062	0,000807	0,001018
7,8	0,016193	0,021532	0,026842	0,032124	0,037377	0,042601	0,047797
2,9	5,25E-18	3,92E-17	1,86E-16	6,65E-16	1,95E-15	4,94E-15	1,12E-14
3,9	6E-15	3,36E-14	1,28E-13	3,8E-13	9,53E-13	2,12E-12	4,27E-12
4,9	4,41E-12	1,85E-11	5,62E-11	1,39E-10	3E-10	5,83E-10	1,05E-09
5,9	2,16E-09	6,79E-09	1,65E-08	3,41E-08	6,29E-08	1,07E-07	1,71E-07
6,9	7,05E-07	1,66E-06	3,24E-06	5,57E-06	8,81E-06	1,31E-05	1,86E-05
7,9	0,000148	0,000262	0,000409	0,000586	0,000795	0,001034	0,001303
8,9	0,018199	0,024191	0,030147	0,036066	0,041949	0,047797	0,053608

Таблица 18 — Расчет вероятностей отказа двухуровневой модели (на 1-м уровне применяется классическая СОК, на 2-м уровне применяется избыточная СОК), WDC 2 TB

(k,n)	Количество модулей на 2-м слое						
	3	4	5	6	7	8	9
2,4	0,00034	0,000602	0,000935	0,001338	0,00181	0,002351	0,002958
3,4	0,029941	0,039721	0,049402	0,058986	0,068473	0,077864	0,087161
2,5	4,29E-06	1,01E-05	1,96E-05	3,36E-05	5,29E-05	7,83E-05	0,000111

Продолжение таблицы 18

(k,n)	Количество модулей на 2-м слое						
	3	4	5	6	7	8	9
3,5	0,000565	0,000996	0,001545	0,002207	0,002982	0,003866	0,004856
4,5	0,037285	0,049402	0,061366	0,07318	0,084845	0,096364	0,107737
2,6	4,87E-08	1,52E-07	3,69E-07	7,58E-07	1,39E-06	2,35E-06	3,73E-06
3,6	8,53E-06	2E-05	3,88E-05	6,63E-05	0,000104	0,000154	0,000218
4,6	0,000843	0,001484	0,002298	0,003278	0,004421	0,005721	0,007175
5,6	0,044574	0,058986	0,07318	0,087161	0,10093	0,114492	0,127849
2,7	5,16E-10	2,15E-09	6,49E-09	1,6E-08	3,42E-08	6,6E-08	1,18E-07
3,7	1,13E-07	3,53E-07	8,52E-07	1,75E-06	3,2E-06	5,4E-06	8,56E-06
4,7	1,48E-05	3,48E-05	6,72E-05	0,000115	0,00018	0,000266	0,000374
5,7	0,001174	0,002064	0,00319	0,004543	0,006117	0,007903	0,009895
6,7	0,051807	0,068473	0,084845	0,10093	0,116732	0,132257	0,147508
2,8	5,2E-12	2,89E-11	1,09E-10	3,21E-10	8,01E-10	1,76E-09	3,53E-09
3,8	1,37E-09	5,69E-09	1,71E-08	4,21E-08	8,99E-08	1,73E-07	3,08E-07
4,8	2,24E-07	7E-07	1,69E-06	3,45E-06	6,31E-06	1,06E-05	1,68E-05
5,8	2,36E-05	5,52E-05	0,000106	0,000182	0,000285	0,000419	0,000589
6,8	0,001557	0,002733	0,004217	0,005997	0,008061	0,010398	0,012997
7,8	0,058986	0,077864	0,096364	0,114492	0,132257	0,149665	0,166724
2,9	5,06E-14	3,74E-13	1,76E-12	6,23E-12	1,81E-11	4,54E-11	1,02E-10
3,9	1,55E-11	8,59E-11	3,23E-10	9,52E-10	2,37E-09	5,2E-09	1,04E-08
4,9	3,06E-09	1,27E-08	3,82E-08	9,35E-08	1,99E-07	3,83E-07	6,8E-07
5,9	4,02E-07	1,25E-06	3,01E-06	6,14E-06	1,12E-05	1,88E-05	2,97E-05
6,9	3,52E-05	8,22E-05	0,000158	0,000269	0,000421	0,000619	0,000869
7,9	0,001992	0,003491	0,005377	0,007634	0,010244	0,013192	0,016462
8,9	0,06611	0,087161	0,107737	0,127849	0,147508	0,166724	0,185507

Таблица 19 — Расчет вероятностей отказа двухуровневой модели (на 1-м уровне применяется классическая СОК, на 2-м уровне применяется избыточная СОК), HGST 2 TB

(k,n)	Количество модулей на 2-м слое						
	3	4	5	6	7	8	9
2,4	1,54E-05	2,74E-05	4,27E-05	6,14E-05	8,35E-05	0,000109	0,000138
3,4	0,006401	0,008526	0,010646	0,012761	0,014872	0,016979	0,019081
2,5	4,12E-08	9,75E-08	1,9E-07	3,28E-07	5,2E-07	7,75E-07	1,1E-06
3,5	2,57E-05	4,55E-05	7,1E-05	0,000102	0,000139	0,000181	0,000229
4,5	0,007995	0,010646	0,013289	0,015926	0,018556	0,021178	0,023794
2,6	9,91E-11	3,13E-10	7,61E-10	1,58E-09	2,91E-09	4,96E-09	7,93E-09
3,6	8,23E-08	1,95E-07	3,79E-07	6,54E-07	1,04E-06	1,54E-06	2,19E-06
4,6	3,84E-05	6,82E-05	0,000106	0,000153	0,000208	0,000271	0,000342
5,6	0,009586	0,012761	0,015926	0,019081	0,022225	0,02536	0,028484
2,7	2,22E-13	9,35E-13	2,85E-12	7,07E-12	1,53E-11	2,97E-11	5,33E-11
3,7	2,31E-10	7,28E-10	1,77E-09	3,67E-09	6,78E-09	1,15E-08	1,84E-08
4,7	1,44E-07	3,4E-07	6,62E-07	1,14E-06	1,81E-06	2,69E-06	3,83E-06
5,7	5,38E-05	9,53E-05	0,000149	0,000214	0,00029	0,000378	0,000477
6,7	0,011175	0,014872	0,018556	0,022225	0,025881	0,029523	0,033152
2,8	4,76E-16	2,67E-15	1,01E-14	3,02E-14	7,6E-14	1,69E-13	3,42E-13
3,8	5,92E-13	2,49E-12	7,58E-12	1,88E-11	4,05E-11	7,88E-11	1,42E-10
4,8	4,61E-10	1,45E-09	3,54E-09	7,32E-09	1,35E-08	2,3E-08	3,67E-08
5,8	2,3E-07	5,43E-07	1,06E-06	1,82E-06	2,89E-06	4,3E-06	6,1E-06
6,8	7,16E-05	0,000127	0,000198	0,000284	0,000386	0,000502	0,000634
7,8	0,012761	0,016979	0,021178	0,02536	0,029523	0,033669	0,037797
2,9	9,81E-19	7,33E-18	3,49E-17	1,25E-16	3,65E-16	9,28E-16	2,11E-15
3,9	1,43E-15	7,99E-15	3,04E-14	9,04E-14	2,27E-13	5,05E-13	1,02E-12

Продолжение таблицы 19

(k,n)	Количество модулей на 2-м слое						
	3	4	5	6	7	8	9
4,9	1,33E-12	5,59E-12	1,7E-11	4,22E-11	9,09E-11	1,77E-10	3,17E-10
5,9	8,29E-10	2,61E-09	6,36E-09	1,31E-08	2,43E-08	4,13E-08	6,59E-08
6,9	3,44E-07	8,13E-07	1,58E-06	2,73E-06	4,32E-06	6,42E-06	9,12E-06
7,9	9,19E-05	0,000163	0,000254	0,000364	0,000495	0,000644	0,000813
8,9	0,014345	0,019081	0,023794	0,028484	0,033152	0,037797	0,042421

Таблица 20 — Расчет вероятностей отказа двухуровневой модели (на 1-м уровне применяется классическая СОК, на 2-м уровне применяется избыточная СОК), HGST 3 TB

(k,n)	Количество модулей на 2-м слое						
	3	4	5	6	7	8	9
2,4	1,51E-05	2,69E-05	4,2E-05	6,04E-05	8,21E-05	0,000107	0,000135
3,4	0,006347	0,008454	0,010557	0,012655	0,014748	0,016837	0,018922
2,5	4,02E-08	9,5E-08	1,85E-07	3,2E-07	5,07E-07	7,55E-07	1,07E-06
3,5	2,52E-05	4,48E-05	6,98E-05	0,0001	0,000136	0,000178	0,000225
4,5	0,007928	0,010557	0,013178	0,015793	0,018401	0,021002	0,023596
2,6	9,58E-11	3,02E-10	7,36E-10	1,52E-09	2,82E-09	4,8E-09	7,67E-09
3,6	8,02E-08	1,9E-07	3,7E-07	6,38E-07	1,01E-06	1,51E-06	2,14E-06
4,6	3,78E-05	6,71E-05	0,000105	0,00015	0,000204	0,000266	0,000336
5,6	0,009506	0,012655	0,015793	0,018922	0,02204	0,025149	0,028248
2,7	2,13E-13	8,97E-13	2,73E-12	6,78E-12	1,46E-11	2,84E-11	5,11E-11
3,7	2,23E-10	7,04E-10	1,71E-09	3,55E-09	6,55E-09	1,12E-08	1,78E-08
4,7	1,4E-07	3,32E-07	6,46E-07	1,11E-06	1,76E-06	2,63E-06	3,73E-06
5,7	5,29E-05	9,37E-05	0,000146	0,00021	0,000285	0,000372	0,000469
6,7	0,011082	0,014748	0,018401	0,02204	0,025666	0,029279	0,032878
2,8	4,52E-16	2,53E-15	9,65E-15	2,87E-14	7,23E-14	1,61E-13	3,25E-13
3,8	5,68E-13	2,39E-12	7,27E-12	1,8E-11	3,89E-11	7,56E-11	1,36E-10
4,8	4,46E-10	1,41E-09	3,42E-09	7,07E-09	1,31E-08	2,22E-08	3,55E-08
5,8	2,24E-07	5,3E-07	1,03E-06	1,78E-06	2,81E-06	4,19E-06	5,95E-06
6,8	7,04E-05	0,000125	0,000195	0,000279	0,000379	0,000494	0,000624
7,8	0,012655	0,016837	0,021002	0,025149	0,029279	0,033391	0,037485
2,9	9,25E-19	6,91E-18	3,29E-17	1,17E-16	3,44E-16	8,75E-16	1,99E-15
3,9	1,36E-15	7,59E-15	2,89E-14	8,6E-14	2,16E-13	4,8E-13	9,7E-13
4,9	1,28E-12	5,36E-12	1,63E-11	4,05E-11	8,72E-11	1,69E-10	3,04E-10
5,9	8,02E-10	2,53E-09	6,15E-09	1,27E-08	2,35E-08	3,99E-08	6,37E-08
6,9	3,36E-07	7,93E-07	1,54E-06	2,66E-06	4,21E-06	6,26E-06	8,89E-06
7,9	9,04E-05	0,00016	0,00025	0,000358	0,000486	0,000633	0,000799
8,9	0,014225	0,018922	0,023596	0,028248	0,032878	0,037485	0,042071

Таблица 21 — Расчет вероятностей отказа двухуровневой модели (на 1-м уровне применяется классическая СОК, на 2-м уровне применяется избыточная СОК), Toshiba 3 TB

(k,n)	Количество модулей на 2-м слое						
	3	4	5	6	7	8	9
2,4	7,37E-05	0,000131	0,000203	0,000292	0,000397	0,000517	0,000652
3,4	0,013974	0,018588	0,023181	0,027752	0,032302	0,036831	0,041338
2,5	4,31E-07	1,02E-06	1,98E-06	3,41E-06	5,4E-06	8,03E-06	1,14E-05
3,5	0,000122	0,000217	0,000338	0,000485	0,000657	0,000856	0,001079

Продолжение таблицы 21

(k,n)	Количество модулей на 2-м слое						
	3	4	5	6	7	8	9
4,5	0,017437	0,023181	0,028892	0,034569	0,040213	0,045824	0,051403
2,6	2,27E-09	7,14E-09	1,74E-08	3,58E-08	6,61E-08	1,12E-07	1,79E-07
3,6	8,59E-07	2,03E-06	3,94E-06	6,79E-06	1,07E-05	1,59E-05	2,26E-05
4,6	0,000183	0,000324	0,000505	0,000724	0,000981	0,001276	0,001608
5,6	0,020887	0,027752	0,034569	0,041338	0,04806	0,054734	0,061362
2,7	1,12E-11	4,68E-11	1,42E-10	3,52E-10	7,56E-10	1,47E-09	2,63E-09
3,7	5,28E-09	1,66E-08	4,03E-08	8,32E-08	1,53E-07	2,6E-07	4,15E-07
4,7	1,5E-06	3,54E-06	6,87E-06	1,18E-05	1,87E-05	2,77E-05	3,92E-05
5,7	0,000256	0,000453	0,000704	0,001008	0,001366	0,001775	0,002235
6,7	0,024326	0,032302	0,040213	0,04806	0,055842	0,063561	0,071216
2,8	5,22E-14	2,92E-13	1,11E-12	3,29E-12	8,24E-12	1,83E-11	3,68E-11
3,8	2,97E-11	1,24E-10	3,77E-10	9,32E-10	2E-09	3,88E-09	6,96E-09
4,8	1,05E-08	3,31E-08	8,03E-08	1,65E-07	3,05E-07	5,16E-07	8,22E-07
5,8	2,39E-06	5,64E-06	1,09E-05	1,88E-05	2,97E-05	4,4E-05	6,23E-05
6,8	0,000341	0,000602	0,000935	0,001338	0,001811	0,002352	0,002959
7,8	0,027752	0,036831	0,045824	0,054734	0,063561	0,072305	0,080967
2,9	2,36E-16	1,76E-15	8,32E-15	2,96E-14	8,66E-14	2,19E-13	4,97E-13
3,9	1,56E-13	8,72E-13	3,3E-12	9,8E-12	2,46E-11	5,44E-11	1,09E-10
4,9	6,65E-11	2,78E-10	8,44E-10	2,09E-09	4,48E-09	8,67E-09	1,55E-08
5,9	1,89E-08	5,93E-08	1,44E-07	2,96E-07	5,45E-07	9,23E-07	1,47E-06
6,9	3,58E-06	8,43E-06	1,63E-05	2,81E-05	4,42E-05	6,56E-05	9,27E-05
7,9	0,000437	0,000771	0,001197	0,001713	0,002316	0,003005	0,003778
8,9	0,031167	0,041338	0,051403	0,061362	0,071216	0,080967	0,090616

Приложение Ж

Результаты расчетов вероятностей отказа двухуровневой модели (на 1-м и 2-м уровнях применяется избыточная СОК)

Таблица 22 — Расчет вероятностей отказа двухуровневой модели (на 1-м и 2-м уровнях используется избыточная СОК), Toshiba 3 ТВ

	4,2	4,3	5,2	5,3	5,4	6,2	6,3	6,4	6,5
4,2	4,06E-10	0,000131	1,55E-15	1,13E-09	0,000203	4,79E-21	6,19E-15	2,53E-09	0,000292
4,3	3,29E-05	0,018588	6,43E-08	5,48E-05	0,023181	1,13E-10	1,28E-07	8,22E-05	0,027752
5,2	5,57E-15	1,02E-06	4,15E-23	2,57E-14	1,98E-06	2,25E-31	3,31E-22	8,66E-14	3,41E-06
5,3	6,77E-10	0,000217	2,58E-15	1,88E-09	0,000338	7,98E-21	1,03E-14	4,22E-09	0,000485
5,4	4,11E-05	0,023181	8,04E-08	6,85E-05	0,028892	1,41E-10	1,61E-07	0,000103	0,034569
6,2	6,88E-20	7,14E-09	1E-30	5,29E-19	1,74E-08	9,55E-42	1,59E-29	2,67E-18	3,58E-08
6,3	1,11E-14	2,03E-06	8,3E-23	5,15E-14	3,94E-06	4,51E-31	6,62E-22	1,73E-13	6,79E-06
6,4	1,02E-09	0,000324	3,87E-15	2,82E-09	0,000505	1,2E-20	1,55E-14	6,33E-09	0,000724
6,5	4,94E-05	0,027752	9,64E-08	8,22E-05	0,034569	1,69E-10	1,93E-07	0,000123	0,041338
7,2	7,92E-25	4,68E-11	2,25E-38	1,01E-23	1,42E-10	3,78E-52	7,17E-37	7,68E-23	3,52E-10
7,3	1,6E-19	1,66E-08	2,33E-30	1,23E-18	4,03E-08	2,23E-41	3,72E-29	6,23E-18	8,32E-08
7,4	1,95E-14	3,54E-06	1,45E-22	9,01E-14	6,87E-06	7,89E-31	1,16E-21	3,03E-13	1,18E-05
7,5	1,42E-09	0,000453	5,42E-15	3,94E-09	0,000704	1,68E-20	2,17E-14	8,86E-09	0,001008
7,6	5,76E-05	0,032302	1,12E-07	9,59E-05	0,040213	1,98E-10	2,25E-07	0,000144	0,04806
8,2	8,69E-30	2,92E-13	4,82E-46	1,85E-28	1,11E-12	1,42E-62	3,07E-44	2,1E-27	3,29E-12
8,3	2,11E-24	1,24E-10	6E-38	2,71E-23	3,77E-10	1,01E-51	1,91E-36	2,05E-22	9,32E-10
8,4	3,21E-19	3,31E-08	4,67E-30	2,47E-18	8,03E-08	4,46E-41	7,44E-29	1,25E-17	1,65E-07
8,5	3,12E-14	5,64E-06	2,32E-22	1,44E-13	1,09E-05	1,26E-30	1,85E-21	4,85E-13	1,88E-05
8,6	1,9E-09	0,000602	7,23E-15	5,26E-09	0,000935	2,23E-20	2,89E-14	1,18E-08	0,001338
8,7	6,58E-05	0,036831	1,29E-07	0,00011	0,045824	2,26E-10	2,57E-07	0,000164	0,054734
9,2	9,2E-35	1,76E-15	9,96E-54	3,27E-33	8,32E-15	5,17E-73	1,27E-51	5,55E-32	2,96E-14
9,3	2,61E-29	8,72E-13	1,45E-45	5,56E-28	3,3E-12	4,27E-62	9,21E-44	6,31E-27	9,8E-12
9,4	4,75E-24	2,78E-10	1,35E-37	6,09E-23	8,44E-10	2,27E-51	4,3E-36	4,61E-22	2,09E-09
9,5	5,78E-19	5,93E-08	8,4E-30	4,44E-18	1,44E-07	8,02E-41	1,34E-28	2,24E-17	2,96E-07
9,6	4,68E-14	8,43E-06	3,49E-22	2,16E-13	1,63E-05	1,89E-30	2,78E-21	7,28E-13	2,81E-05
9,7	2,44E-09	0,000771	9,3E-15	6,76E-09	0,001197	2,87E-20	3,71E-14	1,52E-08	0,001713
9,8	7,41E-05	0,041338	1,45E-07	0,000123	0,051403	2,54E-10	2,89E-07	0,000185	0,061362
	7,2	7,3	7,4	7,5	7,6	8,2	8,3	8,4	8,5
4,2	1,29E-26	2,6E-20	1,89E-14	4,95E-09	0,000397	3,15E-32	9,15E-26	1,04E-19	4,83E-14
4,3	1,85E-13	2,63E-10	2,25E-07	0,000115	0,032302	2,9E-16	4,94E-13	5,26E-10	3,59E-07
5,2	9,95E-40	2,86E-30	1,77E-21	2,37E-13	5,4E-06	3,8E-48	1,88E-38	2,28E-29	7,23E-21
5,3	2,15E-26	4,34E-20	3,15E-14	8,26E-09	0,000657	5,24E-32	1,52E-25	1,73E-19	8,06E-14
5,4	2,32E-13	3,29E-10	2,81E-07	0,000144	0,040213	3,62E-16	6,17E-13	6,58E-10	4,49E-07
6,2	6,92E-53	2,82E-40	1,49E-28	1,02E-17	6,61E-08	4,13E-64	3,49E-51	4,5E-39	9,74E-28
6,3	1,99E-39	5,71E-30	3,54E-21	4,74E-13	1,07E-05	7,6E-48	3,76E-38	4,56E-29	1,45E-20
6,4	3,22E-26	6,5E-20	4,73E-14	1,24E-08	0,000981	7,87E-32	2,29E-25	2,6E-19	1,21E-13
6,5	2,78E-13	3,95E-10	3,37E-07	0,000172	0,04806	4,35E-16	7,41E-13	7,89E-10	5,39E-07
7,2	4,49E-66	2,6E-50	1,17E-35	4,11E-22	7,56E-10	4,18E-80	6,03E-64	8,28E-49	1,22E-34
7,3	1,61E-52	6,58E-40	3,48E-28	2,39E-17	1,53E-07	9,63E-64	8,13E-51	1,05E-38	2,27E-27
7,4	3,48E-39	9,99E-30	6,19E-21	8,3E-13	1,87E-05	1,33E-47	6,59E-38	7,97E-29	2,53E-20
7,5	4,51E-26	9,11E-20	6,62E-14	1,73E-08	0,001366	1,1E-31	3,2E-25	3,64E-19	1,69E-13
7,6	3,24E-13	4,61E-10	3,93E-07	0,000201	0,055842	5,07E-16	8,64E-13	9,21E-10	6,28E-07
8,2	2,77E-79	2,28E-60	8,77E-43	1,58E-26	8,24E-12	4,04E-96	9,92E-77	1,45E-58	1,46E-41
8,3	1,2E-65	6,93E-50	3,12E-35	1,1E-21	2E-09	1,12E-79	1,61E-63	2,21E-48	3,26E-34

8,4	3,23E-52	1,32E-39	6,96E-28	4,77E-17	3,05E-07	1,93E-63	1,63E-50	2,1E-38	4,54E-27
8,5	5,57E-39	1,6E-29	9,91E-21	1,33E-12	2,97E-05	2,13E-47	1,05E-37	1,28E-28	4,05E-20
8,6	6,01E-26	1,21E-19	8,83E-14	2,31E-08	0,001811	1,47E-31	4,27E-25	4,85E-19	2,26E-13
8,7	3,71E-13	5,27E-10	4,49E-07	0,00023	0,063561	5,79E-16	9,88E-13	1,05E-09	7,18E-07
9,2	1,65E-92	1,93E-70	6,33E-50	5,82E-31	8,66E-14	3,8E-112	1,57E-89	2,46E-68	1,69E-48
9,3	8,32E-79	6,85E-60	2,63E-42	4,73E-26	2,46E-11	1,21E-95	2,98E-76	4,36E-58	4,39E-41
9,4	2,69E-65	1,56E-49	7,03E-35	2,47E-21	4,48E-09	2,51E-79	3,62E-63	4,97E-48	7,34E-34
9,5	5,81E-52	2,37E-39	1,25E-27	8,59E-17	5,45E-07	3,47E-63	2,93E-50	3,78E-38	8,18E-27
9,6	8,36E-39	2,4E-29	1,49E-20	1,99E-12	4,42E-05	3,19E-47	1,58E-37	1,91E-28	6,07E-20
9,7	7,73E-26	1,56E-19	1,13E-13	2,97E-08	0,002316	1,89E-31	5,49E-25	6,23E-19	2,9E-13
9,8	4,17E-13	5,93E-10	5,05E-07	0,000259	0,071216	6,52E-16	1,11E-12	1,18E-09	8,08E-07

	8,6	8,7	9,2	9,3	9,4	9,5	9,6	9,7	9,8
4,2	8,79E-09	0,000517	7,14E-38	2,83E-31	4,62E-25	3,36E-19	1,09E-13	1,45E-08	0,000652
4,3	0,000153	0,036831	4,36E-19	8,68E-16	1,11E-12	9,46E-10	5,38E-07	0,000197	0,041338
5,2	5,61E-13	8,03E-06	1,3E-56	1,02E-46	2,14E-37	1,32E-28	2,43E-20	1,19E-12	1,14E-05
5,3	1,47E-08	0,000856	1,19E-37	4,71E-31	7,7E-25	5,6E-19	1,81E-13	2,42E-08	0,001079
5,4	0,000191	0,045824	5,46E-19	1,09E-15	1,39E-12	1,18E-09	6,73E-07	0,000246	0,051403
6,2	3,22E-17	1,12E-07	2,13E-75	3,33E-62	8,9E-50	4,7E-38	4,91E-27	8,77E-17	1,79E-07
6,3	1,12E-12	1,59E-05	2,6E-56	2,04E-46	4,28E-37	2,65E-28	4,87E-20	2,38E-12	2,26E-05
6,4	2,2E-08	0,001276	1,79E-37	7,07E-31	1,16E-24	8,4E-19	2,71E-13	3,63E-08	0,001608
6,5	0,00023	0,054734	6,55E-19	1,3E-15	1,67E-12	1,42E-09	8,07E-07	0,000295	0,061362
7,2	1,73E-21	1,47E-09	3,25E-94	1,01E-77	3,46E-62	1,56E-47	9,25E-34	6,04E-21	2,63E-09
7,3	7,52E-17	2,6E-07	4,96E-75	7,77E-62	2,08E-49	1,1E-37	1,15E-26	2,05E-16	4,15E-07
7,4	1,96E-12	2,77E-05	4,55E-56	3,58E-46	7,48E-37	4,64E-28	8,52E-20	4,16E-12	3,92E-05
7,5	3,08E-08	0,001775	2,5E-37	9,89E-31	1,62E-24	1,18E-18	3,8E-13	5,08E-08	0,002235
7,6	0,000268	0,063561	7,64E-19	1,52E-15	1,94E-12	1,66E-09	9,42E-07	0,000344	0,071216
8,2	8,81E-26	1,83E-11	4,7E-113	2,93E-93	1,28E-74	4,91E-57	1,66E-40	3,96E-25	3,68E-11
8,3	4,6E-21	3,88E-09	8,66E-94	2,7E-77	9,22E-62	4,15E-47	2,47E-33	1,61E-20	6,96E-09
8,4	1,5E-16	5,16E-07	9,92E-75	1,55E-61	4,15E-49	2,19E-37	2,29E-26	4,09E-16	8,22E-07
8,5	3,14E-12	4,4E-05	7,28E-56	5,73E-46	1,2E-36	7,42E-28	1,36E-19	6,66E-12	6,23E-05
8,6	4,1E-08	0,002352	3,33E-37	1,32E-30	2,16E-24	1,57E-18	5,07E-13	6,77E-08	0,002959
8,7	0,000306	0,072305	8,73E-19	1,74E-15	2,22E-12	1,89E-09	1,08E-06	0,000393	0,080967
9,2	4,34E-30	2,19E-13	6,6E-132	8,2E-109	4,57E-87	1,49E-66	2,87E-47	2,5E-29	4,97E-13
9,3	2,64E-25	5,44E-11	1,4E-112	8,78E-93	3,84E-74	1,47E-56	4,98E-40	1,19E-24	1,09E-10
9,4	1,04E-20	8,67E-09	1,95E-93	6,07E-77	2,07E-61	9,35E-47	5,55E-33	3,62E-20	1,55E-08
9,5	2,71E-16	9,23E-07	1,79E-74	2,8E-61	7,48E-49	3,95E-37	4,13E-26	7,37E-16	1,47E-06
9,6	4,71E-12	6,56E-05	1,09E-55	8,59E-46	1,8E-36	1,11E-27	2,04E-19	9,99E-12	9,27E-05
9,7	5,27E-08	0,003005	4,29E-37	1,7E-30	2,77E-24	2,02E-18	6,51E-13	8,7E-08	0,003778
9,8	0,000344	0,080967	9,82E-19	1,95E-15	2,5E-12	2,13E-09	1,21E-06	0,000443	0,090616

Таблица 23 — Расчет вероятностей отказа двухуровневой модели (на 1-м и 2-м уровнях используется избыточная СОК), WDC 6 TB

	4,2	4,3	5,2	5,3	5,4	6,2	6,3	6,4	6,5
4,2	4,61E-11	4,41E-05	5,92E-17	1,28E-10	6,89E-05	6,16E-23	2,37E-16	2,88E-10	9,9E-05
4,3	1,11E-05	0,010825	1,26E-08	1,85E-05	0,013513	1,28E-11	2,51E-08	2,77E-05	0,016193
5,2	2,13E-16	2E-07	3,1E-25	9,85E-16	3,89E-07	3,29E-34	2,48E-24	3,32E-15	6,72E-07
5,3	7,68E-11	7,34E-05	9,87E-17	2,13E-10	0,000115	1,03E-22	3,94E-16	4,79E-10	0,000165
5,4	1,39E-05	0,013513	1,57E-08	2,31E-05	0,016862	1,6E-11	3,14E-08	3,46E-05	0,0202
6,2	8,86E-22	8,14E-10	1,46E-33	6,82E-21	1,98E-09	1,58E-45	2,33E-32	3,45E-20	4,1E-09
6,3	4,26E-16	3,99E-07	6,2E-25	1,97E-15	7,77E-07	6,58E-34	4,95E-24	6,64E-15	1,34E-06
6,4	1,15E-10	0,00011	1,48E-16	3,2E-10	0,000171	1,54E-22	5,91E-16	7,19E-10	0,000246
6,5	1,66E-05	0,016193	1,88E-08	2,77E-05	0,0202	1,92E-11	3,77E-08	4,15E-05	0,024191
7,2	3,44E-27	3,1E-12	6,42E-42	4,41E-26	9,42E-12	7,09E-57	2,05E-40	3,34E-25	2,34E-11
7,3	2,07E-21	1,9E-09	3,41E-33	1,59E-20	4,61E-09	3,69E-45	5,44E-32	8,04E-20	9,54E-09
7,4	7,45E-16	6,96E-07	1,08E-24	3,45E-15	1,36E-06	1,15E-33	8,66E-24	1,16E-14	2,34E-06
7,5	1,61E-10	0,000154	2,07E-16	4,48E-10	0,000239	2,16E-22	8,28E-16	1,01E-09	0,000344

7,6	1,94E-05	0,018866	2,2E-08	3,23E-05	0,023527	2,24E-11	4,4E-08	4,85E-05	0,028165
8,2	1,27E-32	1,12E-14	2,69E-50	2,71E-31	4,27E-14	3,03E-68	1,72E-48	3,08E-30	1,27E-13
8,3	9,16E-27	8,24E-12	1,71E-41	1,18E-25	2,51E-11	1,89E-56	5,47E-40	8,91E-25	6,21E-11
8,4	4,13E-21	3,78E-09	6,81E-33	3,18E-20	9,2E-09	7,37E-45	1,09E-31	1,61E-19	1,9E-08
8,5	1,19E-15	1,11E-06	1,74E-24	5,51E-15	2,16E-06	1,84E-33	1,39E-23	1,86E-14	3,73E-06
8,6	2,15E-10	0,000204	2,76E-16	5,97E-10	0,000318	2,87E-22	1,1E-15	1,34E-09	0,000457
8,7	2,22E-05	0,021532	2,51E-08	3,69E-05	0,026842	2,56E-11	5,02E-08	5,54E-05	0,032124
9,2	4,53E-38	3,92E-17	1,09E-58	1,61E-36	1,86E-16	1,25E-79	1,39E-56	2,75E-35	6,65E-16
9,3	3,81E-32	3,36E-14	8,07E-50	8,14E-31	1,28E-13	9,08E-68	5,15E-48	9,25E-30	3,8E-13
9,4	2,06E-26	1,85E-11	3,85E-41	2,65E-25	5,62E-11	4,25E-56	1,23E-39	2E-24	1,39E-10
9,5	7,44E-21	6,79E-09	1,23E-32	5,73E-20	1,65E-08	1,33E-44	1,96E-31	2,89E-19	3,41E-08
9,6	1,79E-15	1,66E-06	2,6E-24	8,27E-15	3,24E-06	2,76E-33	2,08E-23	2,79E-14	5,57E-06
9,7	2,77E-10	0,000262	3,55E-16	7,68E-10	0,000409	3,69E-22	1,42E-15	1,73E-09	0,000586
9,8	2,49E-05	0,024191	2,83E-08	4,16E-05	0,030147	2,88E-11	5,65E-08	6,23E-05	0,036066
	7,2	7,3	7,4	7,5	7,6	8,2	8,3	8,4	8,5
4,2	5,58E-29	3,35E-22	7,24E-16	5,63E-10	0,000135	4,59E-35	3,96E-28	1,34E-21	1,85E-15
4,3	1,22E-14	2,99E-11	4,39E-08	3,88E-05	0,018866	1,11E-17	3,25E-14	5,97E-11	7,03E-08
5,2	2,84E-43	4,17E-33	1,32E-23	9,09E-15	1,06E-06	2,11E-52	5,37E-42	3,33E-32	5,42E-23
5,3	9,3E-29	5,58E-22	1,21E-15	9,39E-10	0,000224	7,65E-35	6,61E-28	2,23E-21	3,08E-15
5,4	1,52E-14	3,74E-11	5,49E-08	4,84E-05	0,023527	1,38E-17	4,06E-14	7,47E-11	8,78E-08
6,2	1,3E-57	4,67E-44	2,18E-31	1,32E-19	7,58E-09	8,77E-70	6,55E-56	7,46E-43	1,43E-30
6,3	5,67E-43	8,34E-33	2,65E-23	1,82E-14	2,12E-06	4,23E-52	1,07E-41	6,66E-32	1,08E-22
6,4	1,4E-28	8,37E-22	1,81E-15	1,41E-09	0,000334	1,15E-34	9,91E-28	3,35E-21	4,63E-15
6,5	1,83E-14	4,48E-11	6,59E-08	5,81E-05	0,028165	1,66E-17	4,88E-14	8,96E-11	1,05E-07
7,2	5,54E-72	4,89E-55	3,36E-39	1,79E-24	5,04E-11	3,39E-87	7,45E-70	1,56E-53	3,51E-38
7,3	3,03E-57	1,09E-43	5,09E-31	3,08E-19	1,76E-08	2,05E-69	1,53E-55	1,74E-42	3,33E-30
7,4	9,93E-43	1,46E-32	4,64E-23	3,18E-14	3,7E-06	7,4E-52	1,88E-41	1,17E-31	1,9E-22
7,5	1,95E-28	1,17E-21	2,53E-15	1,97E-09	0,000466	1,61E-34	1,39E-27	4,68E-21	6,48E-15
7,6	2,13E-14	5,23E-11	7,69E-08	6,78E-05	0,032782	1,94E-17	5,69E-14	1,05E-10	1,23E-07
8,2	2,25E-86	4,87E-66	4,91E-47	2,32E-29	3,19E-13	1,3E-104	8,07E-84	3,11E-64	8,22E-46
8,3	1,48E-71	1,3E-54	8,95E-39	4,78E-24	1,34E-10	9,05E-87	1,99E-69	4,16E-53	9,36E-38
8,4	6,06E-57	2,18E-43	1,02E-30	6,17E-19	3,51E-08	4,09E-69	3,06E-55	3,48E-42	6,66E-30
8,5	1,59E-42	2,34E-32	7,42E-23	5,09E-14	5,9E-06	1,18E-51	3,01E-41	1,87E-31	3,03E-22
8,6	2,6E-28	1,56E-21	3,38E-15	2,63E-09	0,00062	2,14E-34	1,85E-27	6,25E-21	8,64E-15
8,7	2,44E-14	5,98E-11	8,79E-08	7,75E-05	0,037377	2,21E-17	6,5E-14	1,19E-10	1,41E-07
9,2	8,8E-101	4,68E-77	6,94E-55	2,88E-34	1,95E-15	4,4E-122	8,44E-98	5,97E-75	1,86E-53
9,3	6,76E-86	1,46E-65	1,47E-46	6,95E-29	9,53E-13	3,8E-104	2,42E-83	9,32E-64	2,47E-45
9,4	3,32E-71	2,93E-54	2,01E-38	1,08E-23	3E-10	2,04E-86	4,47E-69	9,36E-53	2,11E-37
9,5	1,09E-56	3,93E-43	1,83E-30	1,11E-18	6,29E-08	7,37E-69	5,5E-55	6,27E-42	1,2E-29
9,6	2,38E-42	3,5E-32	1,11E-22	7,64E-14	8,81E-06	1,78E-51	4,51E-41	2,8E-31	4,55E-22
9,7	3,35E-28	2,01E-21	4,34E-15	3,38E-09	0,000795	2,75E-34	2,38E-27	8,03E-21	1,11E-14
9,8	2,74E-14	6,72E-11	9,88E-08	8,72E-05	0,041949	2,49E-17	7,32E-14	1,34E-10	1,58E-07
	8,6	8,7	9,2	9,3	9,4	9,5	9,6	9,7	9,8
4,2	1E-09	0,000175	3,51E-41	4,12E-34	2E-27	4,33E-21	4,16E-15	1,65E-09	0,000222
4,3	5,16E-05	0,021532	9,67E-21	3,32E-17	7,31E-14	1,07E-10	1,05E-07	6,64E-05	0,024191
5,2	2,15E-14	1,59E-06	1,41E-61	5,7E-51	6,11E-41	1,94E-31	1,83E-22	4,57E-14	2,25E-06
5,3	1,67E-09	0,000291	5,84E-41	6,87E-34	3,34E-27	7,22E-21	6,93E-15	2,75E-09	0,000368
5,4	6,46E-05	0,026842	1,21E-20	4,15E-17	9,14E-14	1,34E-10	1,32E-07	8,3E-05	0,030147
6,2	4,17E-19	1,29E-08	5,12E-82	7,09E-68	1,67E-54	7,82E-42	7,21E-30	1,14E-18	2,06E-08
6,3	4,31E-14	3,16E-06	2,83E-61	1,14E-50	1,22E-40	3,88E-31	3,65E-22	9,14E-14	4,49E-06
6,4	2,5E-09	0,000435	8,77E-41	1,03E-33	5,01E-27	1,08E-20	1,04E-14	4,13E-09	0,00055
6,5	7,75E-05	0,032124	1,45E-20	4,97E-17	1,1E-13	1,61E-10	1,58E-07	9,96E-05	0,036066
7,2	7,54E-24	9,8E-11	1,7E-102	8,22E-85	4,28E-68	2,94E-52	2,66E-37	2,64E-23	1,76E-10
7,3	9,73E-19	3E-08	1,2E-81	1,65E-67	3,91E-54	1,82E-41	1,68E-29	2,65E-18	4,79E-08
7,4	7,53E-14	5,5E-06	4,94E-61	1,99E-50	2,14E-40	6,79E-31	6,39E-22	1,6E-13	7,81E-06
7,5	3,5E-09	0,000607	1,23E-40	1,44E-33	7,02E-27	1,52E-20	1,46E-14	5,78E-09	0,000767
7,6	9,04E-05	0,037377	1,69E-20	5,8E-17	1,28E-13	1,88E-10	1,84E-07	0,000116	0,041949
8,2	1,3E-28	7,09E-13	5,6E-123	9,1E-102	1,04E-81	1,05E-62	9,33E-45	5,85E-28	1,43E-12

8,3	2,01E-23	2,6E-10	4,6E-102	2,19E-84	1,14E-67	7,84E-52	7,09E-37	7,04E-23	4,67E-10
8,4	1,95E-18	5,97E-08	2,39E-81	3,31E-67	7,81E-54	3,65E-41	3,37E-29	5,31E-18	9,53E-08
8,5	1,21E-13	8,77E-06	7,91E-61	3,19E-50	3,42E-40	1,09E-30	1,02E-21	2,56E-13	1,24E-05
8,6	4,67E-09	0,000807	1,64E-40	1,92E-33	9,35E-27	2,02E-20	1,94E-14	7,71E-09	0,001018
8,7	0,000103	0,042601	1,93E-20	6,63E-17	1,46E-13	2,15E-10	2,11E-07	0,000133	0,047797
9,2	2,15E-33	4,94E-15	1,7E-143	9,7E-119	2,45E-95	3,64E-73	3,16E-52	1,25E-32	1,12E-14
9,3	3,89E-28	2,12E-12	1,7E-122	2,7E-101	3,13E-81	3,16E-62	2,8E-44	1,75E-27	4,27E-12
9,4	4,52E-23	5,83E-10	1E-101	4,93E-84	2,57E-67	1,76E-51	1,59E-36	1,59E-22	1,05E-09
9,5	3,5E-18	1,07E-07	4,3E-81	5,95E-67	1,41E-53	6,57E-41	6,06E-29	9,55E-18	1,71E-07
9,6	1,81E-13	1,31E-05	1,19E-60	4,79E-50	5,13E-40	1,63E-30	1,53E-21	3,84E-13	1,86E-05
9,7	6E-09	0,001034	2,1E-40	2,47E-33	1,2E-26	2,6E-20	2,5E-14	9,91E-09	0,001303
9,8	0,000116	0,047797	2,18E-20	7,46E-17	1,65E-13	2,42E-10	2,37E-07	0,000149	0,053608

Таблица 24 — Расчет вероятностей отказа двухуровневой модели (на 1-м и 2-м уровнях используется избыточная СОК), WDC 2 TB

	4,2	4,3	5,2	5,3	5,4	6,2	6,3	6,4	6,5
4,2	8,79E-09	0,000602	1,56E-13	2,43E-08	0,000935	2,25E-18	6,22E-13	5,46E-08	0,001338
4,3	0,000153	0,039721	6,45E-07	0,000255	0,049402	2,45E-09	1,29E-06	0,000381	0,058986
5,2	5,61E-13	1,01E-05	4,2E-20	2,58E-12	1,96E-05	2,29E-27	3,34E-19	8,67E-12	3,36E-05
5,3	1,46E-08	0,000996	2,6E-13	4,06E-08	0,001545	3,75E-18	1,04E-12	9,09E-08	0,002207
5,4	0,000191	0,049402	8,07E-07	0,000318	0,061366	3,06E-09	1,61E-06	0,000477	0,07318
6,2	3,22E-17	1,52E-07	1,02E-26	2,47E-16	3,69E-07	2,11E-36	1,61E-25	1,24E-15	7,58E-07
6,3	1,12E-12	2E-05	8,4E-20	5,17E-12	3,88E-05	4,59E-27	6,68E-19	1,73E-11	6,63E-05
6,4	2,2E-08	0,001484	3,9E-13	6,08E-08	0,002298	5,62E-18	1,56E-12	1,36E-07	0,003278
6,5	0,00023	0,058986	9,68E-07	0,000382	0,07318	3,67E-09	1,93E-06	0,000572	0,087161
7,2	1,73E-21	2,15E-09	2,29E-33	2,2E-20	6,49E-09	1,8E-45	7,27E-32	1,66E-19	1,6E-08
7,3	7,51E-17	3,53E-07	2,37E-26	5,76E-16	8,52E-07	4,91E-36	3,76E-25	2,89E-15	1,75E-06
7,4	1,96E-12	3,48E-05	1,47E-19	9,04E-12	6,72E-05	8,03E-27	1,17E-18	3,03E-11	0,000115
7,5	3,08E-08	0,002064	5,47E-13	8,52E-08	0,00319	7,87E-18	2,18E-12	1,91E-07	0,004543
7,6	0,000268	0,068473	1,13E-06	0,000446	0,084845	4,28E-09	2,25E-06	0,000667	0,10093
8,2	8,8E-26	2,89E-11	4,94E-40	1,87E-24	1,09E-10	1,47E-54	3,12E-38	2,11E-23	3,21E-10
8,3	4,6E-21	5,69E-09	6,12E-33	5,87E-20	1,71E-08	4,81E-45	1,94E-31	4,42E-19	4,21E-08
8,4	1,5E-16	7E-07	4,74E-26	1,15E-15	1,69E-06	9,83E-36	7,53E-25	5,79E-15	3,45E-06
8,5	3,14E-12	5,52E-05	2,35E-19	1,45E-11	0,000106	1,28E-26	1,87E-18	4,86E-11	0,000182
8,6	4,1E-08	0,002733	7,29E-13	1,14E-07	0,004217	1,05E-17	2,9E-12	2,55E-07	0,005997
8,7	0,000306	0,077864	1,29E-06	0,000509	0,096364	4,9E-09	2,58E-06	0,000763	0,114492
9,2	4,33E-30	3,74E-13	1,02E-46	1,53E-28	1,76E-12	1,16E-63	1,29E-44	2,58E-27	6,23E-12
9,3	2,64E-25	8,59E-11	1,48E-39	5,6E-24	3,23E-10	4,42E-54	9,37E-38	6,32E-23	9,52E-10
9,4	1,04E-20	1,27E-08	1,38E-32	1,32E-19	3,82E-08	1,08E-44	4,36E-31	9,94E-19	9,35E-08
9,5	2,7E-16	1,25E-06	8,54E-26	2,07E-15	3,01E-06	1,77E-35	1,36E-24	1,04E-14	6,14E-06
9,6	4,71E-12	8,22E-05	3,53E-19	2,17E-11	0,000158	1,93E-26	2,81E-18	7,28E-11	0,000269
9,7	5,27E-08	0,003491	9,37E-13	1,46E-07	0,005377	1,35E-17	3,73E-12	3,27E-07	0,007634
9,8	0,000344	0,087161	1,45E-06	0,000573	0,107737	5,51E-09	2,9E-06	0,000858	0,127849
	7,2	7,3	7,4	7,5	7,6	8,2	8,3	8,4	8,5
4,2	2,82E-23	1,22E-17	1,9E-12	1,07E-07	0,00181	3,21E-28	2E-22	4,86E-17	4,84E-12
4,3	8,67E-12	5,7E-09	2,25E-06	0,000533	0,068473	2,92E-14	2,31E-11	1,14E-08	3,59E-06
5,2	1,02E-34	2,9E-26	1,78E-18	2,37E-11	5,29E-05	3,91E-42	1,92E-33	2,3E-25	7,25E-18
5,3	4,7E-23	2,03E-17	3,16E-12	1,78E-07	0,002982	5,35E-28	3,33E-22	8,09E-17	8,07E-12
5,4	1,08E-11	7,13E-09	2,81E-06	0,000666	0,084845	3,66E-14	2,88E-11	1,42E-08	4,49E-06
6,2	3,31E-46	6,19E-35	1,5E-24	4,73E-15	1,39E-06	4,29E-56	1,66E-44	9,83E-34	9,77E-24
6,3	2,04E-34	5,79E-26	3,56E-18	4,73E-11	0,000104	7,82E-42	3,84E-33	4,6E-25	1,45E-17
6,4	7,05E-23	3,05E-17	4,75E-12	2,66E-07	0,004421	8,02E-28	4,99E-22	1,21E-16	1,21E-11
6,5	1,3E-11	8,55E-09	3,38E-06	0,000799	0,10093	4,39E-14	3,46E-11	1,71E-08	5,39E-06
7,2	1E-57	1,24E-43	1,18E-30	8,83E-19	3,42E-08	4,39E-70	1,34E-55	3,91E-42	1,23E-29
7,3	7,73E-46	1,44E-34	3,5E-24	1,1E-14	3,2E-06	1E-55	3,87E-44	2,29E-33	2,28E-23
7,4	3,56E-34	1,01E-25	6,23E-18	8,28E-11	0,00018	1,37E-41	6,72E-33	8,06E-25	2,54E-17

7,5	9,87E-23	4,27E-17	6,64E-12	3,73E-07	0,006117	1,12E-27	6,99E-22	1,7E-16	1,69E-11
7,6	1,52E-11	9,98E-09	3,94E-06	0,000933	0,116732	5,12E-14	4,04E-11	1,99E-08	6,29E-06
8,2	2,9E-69	2,35E-52	8,87E-37	1,57E-22	8,01E-10	4,28E-84	1,03E-66	1,48E-50	1,47E-35
8,3	2,68E-57	3,29E-43	3,15E-30	2,36E-18	8,99E-08	1,17E-69	3,58E-55	1,04E-41	3,28E-29
8,4	1,55E-45	2,89E-34	7,01E-24	2,21E-14	6,31E-06	2E-55	7,75E-44	4,59E-33	4,56E-23
8,5	5,7E-34	1,62E-25	9,97E-18	1,33E-10	0,000285	2,19E-41	1,07E-32	1,29E-24	4,06E-17
8,6	1,32E-22	5,69E-17	8,86E-12	4,97E-07	0,008061	1,5E-27	9,32E-22	2,27E-16	2,26E-11
8,7	1,73E-11	1,14E-08	4,5E-06	0,001066	0,132257	5,85E-14	4,61E-11	2,28E-08	7,19E-06
9,2	8,1E-81	4,3E-61	6,42E-43	2,69E-26	1,81E-11	4,02E-98	7,65E-78	5,43E-59	1,7E-41
9,3	8,71E-69	7,04E-52	2,66E-36	4,71E-22	2,37E-09	1,28E-83	3,09E-66	4,45E-50	4,41E-35
9,4	6,03E-57	7,41E-43	7,1E-30	5,3E-18	1,99E-07	2,63E-69	8,05E-55	2,35E-41	7,37E-29
9,5	2,78E-45	5,2E-34	1,26E-23	3,97E-14	1,12E-05	3,6E-55	1,39E-43	8,25E-33	8,2E-23
9,6	8,56E-34	2,43E-25	1,5E-17	1,99E-10	0,000421	3,28E-41	1,61E-32	1,93E-24	6,09E-17
9,7	1,69E-22	7,31E-17	1,14E-11	6,39E-07	0,010244	1,92E-27	1,2E-21	2,91E-16	2,91E-11
9,8	1,95E-11	1,28E-08	5,06E-06	0,001199	0,147508	6,58E-14	5,19E-11	2,56E-08	8,08E-06
	8,6	8,7	9,2	9,3	9,4	9,5	9,6	9,7	9,8
4,2	1,89E-07	0,002351	3,39E-33	2,87E-27	1,01E-21	1,57E-16	1,09E-11	3,11E-07	0,002958
4,3	0,000709	0,077864	9,51E-17	8,75E-14	5,18E-11	2,04E-08	5,38E-06	0,000911	0,087161
5,2	5,58E-11	7,83E-05	1,34E-49	1,05E-40	2,17E-32	1,33E-24	2,43E-17	1,18E-10	0,000111
5,3	3,15E-07	0,003866	5,66E-33	4,79E-27	1,68E-21	2,61E-16	1,81E-11	5,18E-07	0,004856
5,4	0,000887	0,096364	1,19E-16	1,09E-13	6,48E-11	2,56E-08	6,72E-06	0,001138	0,107737
6,2	1,49E-14	2,35E-06	4,8E-66	3,44E-54	4,22E-43	1,02E-32	4,91E-23	4,03E-14	3,73E-06
6,3	1,12E-10	0,000154	2,69E-49	2,1E-40	4,34E-32	2,67E-24	4,87E-17	2,36E-10	0,000218
6,4	4,72E-07	0,005721	8,48E-33	7,19E-27	2,52E-21	3,92E-16	2,71E-11	7,77E-07	0,007175
6,5	0,001064	0,114492	1,43E-16	1,31E-13	7,77E-11	3,07E-08	8,07E-06	0,001366	0,127849
7,2	3,69E-18	6,6E-08	1,6E-82	1,05E-67	7,65E-54	7,32E-41	9,24E-29	1,29E-17	1,18E-07
7,3	3,47E-14	5,4E-06	1,12E-65	8,03E-54	9,85E-43	2,39E-32	1,15E-22	9,41E-14	8,56E-06
7,4	1,95E-10	0,000266	4,71E-49	3,67E-40	7,6E-32	4,67E-24	8,51E-17	4,13E-10	0,000374
7,5	6,61E-07	0,007903	1,19E-32	1,01E-26	3,52E-21	5,48E-16	3,8E-11	1,09E-06	0,009895
7,6	0,001241	0,132257	1,66E-16	1,53E-13	9,07E-11	3,58E-08	9,41E-06	0,001593	0,147508
8,2	8,73E-22	1,76E-09	5,1E-99	3,08E-81	1,32E-64	4,99E-49	1,66E-34	3,9E-21	3,53E-09
8,3	9,84E-18	1,73E-07	4,26E-82	2,81E-67	2,04E-53	1,95E-40	2,46E-28	3,43E-17	3,08E-07
8,4	6,93E-14	1,06E-05	2,24E-65	1,61E-53	1,97E-42	4,77E-32	2,29E-22	1,88E-13	1,68E-05
8,5	3,13E-10	0,000419	7,53E-49	5,87E-40	1,22E-31	7,47E-24	1,36E-16	6,61E-10	0,000589
8,6	8,81E-07	0,010398	1,58E-32	1,34E-26	4,7E-21	7,31E-16	5,06E-11	1,45E-06	0,012997
8,7	0,001418	0,149665	1,9E-16	1,75E-13	1,04E-10	4,09E-08	1,08E-05	0,00182	0,166724
9,2	1,99E-25	4,54E-11	1,5E-115	8,66E-95	2,2E-75	3,28E-57	2,87E-40	1,14E-24	1,02E-10
9,3	2,62E-21	5,2E-09	1,52E-98	9,23E-81	3,96E-64	1,5E-48	4,97E-34	1,17E-20	1,04E-08
9,4	2,21E-17	3,83E-07	9,58E-82	6,33E-67	4,59E-53	4,39E-40	5,54E-28	7,71E-17	6,8E-07
9,5	1,25E-13	1,88E-05	4,03E-65	2,89E-53	3,54E-42	8,59E-32	4,12E-22	3,39E-13	2,97E-05
9,6	4,69E-10	0,000619	1,13E-48	8,81E-40	1,82E-31	1,12E-23	2,04E-16	9,91E-10	0,000869
9,7	1,13E-06	0,013192	2,04E-32	1,72E-26	6,04E-21	9,4E-16	6,51E-11	1,86E-06	0,016462
9,8	0,001596	0,166724	2,14E-16	1,97E-13	1,17E-10	4,6E-08	1,21E-05	0,002048	0,185507

Таблица 25 — Расчет вероятностей отказа двухуровневой модели (на 1-м и 2-м уровнях используется избыточная СОК), HGST 2 TB

	4,2	4,3	5,2	5,3	5,4	6,2	6,3	6,4	6,5
4,2	1,77E-11	2,74E-05	1,4E-17	4,9E-11	4,27E-05	9,05E-24	5,61E-17	1,1E-10	6,14E-05
4,3	6,86E-06	0,008526	6,12E-09	1,14E-05	0,010646	4,91E-12	1,22E-08	1,71E-05	0,012761
5,2	5,05E-17	9,75E-08	3,58E-26	2,34E-16	1,9E-07	1,85E-35	2,86E-25	7,88E-16	3,28E-07
5,3	2,95E-11	4,55E-05	2,34E-17	8,17E-11	7,1E-05	1,51E-23	9,36E-17	1,84E-10	0,000102
5,4	8,58E-06	0,010646	7,65E-09	1,43E-05	0,013289	6,14E-12	1,53E-08	2,14E-05	0,015926
6,2	1,3E-22	3,13E-10	8,22E-35	1E-21	7,61E-10	3,41E-47	1,31E-33	5,07E-21	1,58E-09
6,3	1,01E-16	1,95E-07	7,16E-26	4,67E-16	3,79E-07	3,7E-35	5,72E-25	1,58E-15	6,54E-07
6,4	4,42E-11	6,82E-05	3,51E-17	1,23E-10	0,000106	2,26E-23	1,4E-16	2,76E-10	0,000153
6,5	1,03E-05	0,012761	9,18E-09	1,72E-05	0,015926	7,37E-12	1,84E-08	2,57E-05	0,019081

7,2	3,13E-28	9,35E-13	1,76E-43	4,01E-27	2,85E-12	5,86E-59	5,62E-42	3,04E-26	7,07E-12
7,3	3,04E-22	7,28E-10	1,92E-34	2,34E-21	1,77E-09	7,95E-47	3,06E-33	1,18E-20	3,67E-09
7,4	1,77E-16	3,4E-07	1,25E-25	8,18E-16	6,62E-07	6,48E-35	1E-24	2,76E-15	1,14E-06
7,5	6,18E-11	9,53E-05	4,92E-17	1,72E-10	0,000149	3,17E-23	1,96E-16	3,86E-10	0,000214
7,6	1,2E-05	0,014872	1,07E-08	2E-05	0,018556	8,59E-12	2,14E-08	3E-05	0,022225
8,2	7,15E-34	2,67E-15	3,59E-52	1,53E-32	1,01E-14	9,59E-71	2,29E-50	1,74E-31	3,02E-14
8,3	8,34E-28	2,49E-12	4,7E-43	1,07E-26	7,58E-12	1,56E-58	1,5E-41	8,11E-26	1,88E-11
8,4	6,07E-22	1,45E-09	3,84E-34	4,68E-21	3,54E-09	1,59E-46	6,13E-33	2,36E-20	7,32E-09
8,5	2,83E-16	5,43E-07	2,01E-25	1,31E-15	1,06E-06	1,04E-34	1,6E-24	4,41E-15	1,82E-06
8,6	8,25E-11	0,000127	6,56E-17	2,29E-10	0,000198	4,22E-23	2,62E-16	5,15E-10	0,000284
8,7	1,37E-05	0,016979	1,22E-08	2,29E-05	0,021178	9,82E-12	2,45E-08	3,43E-05	0,02536
9,2	1,58E-39	7,33E-18	7,07E-61	5,62E-38	3,49E-17	1,51E-82	9,02E-59	9,58E-37	1,25E-16
9,3	2,15E-33	7,99E-15	1,08E-51	4,59E-32	3,04E-14	2,88E-70	6,88E-50	5,22E-31	9,04E-14
9,4	1,88E-27	5,59E-12	1,06E-42	2,41E-26	1,7E-11	3,52E-58	3,37E-41	1,82E-25	4,22E-11
9,5	1,09E-21	2,61E-09	6,91E-34	8,42E-21	6,36E-09	2,86E-46	1,1E-32	4,26E-20	1,31E-08
9,6	4,25E-16	8,13E-07	3,01E-25	1,96E-15	1,58E-06	1,55E-34	2,4E-24	6,62E-15	2,73E-06
9,7	1,06E-10	0,000163	8,43E-17	2,94E-10	0,000254	5,43E-23	3,37E-16	6,62E-10	0,000364
9,8	1,54E-05	0,019081	1,38E-08	2,57E-05	0,023794	1,11E-11	2,75E-08	3,86E-05	0,028484
	7,2	7,3	7,4	7,5	7,6	8,2	8,3	8,4	8,5
4,2	5,07E-30	4,92E-23	1,72E-16	2,16E-10	8,35E-05	2,58E-36	3,6E-29	1,97E-22	4,39E-16
4,3	3,68E-15	1,15E-11	2,14E-08	2,4E-05	0,014872	2,62E-18	9,8E-15	2,29E-11	3,42E-08
5,2	7,78E-45	2,35E-34	1,53E-24	2,16E-15	5,2E-07	2,82E-54	1,47E-43	1,88E-33	6,27E-24
5,3	8,46E-30	8,2E-23	2,86E-16	3,6E-10	0,000139	4,3E-36	6,01E-29	3,28E-22	7,32E-16
5,4	4,6E-15	1,43E-11	2,68E-08	3E-05	0,018556	3,28E-18	1,23E-14	2,86E-11	4,28E-08
6,2	1,07E-59	1,01E-45	1,23E-32	1,94E-20	2,91E-09	2,78E-72	5,41E-58	1,61E-44	8,05E-32
6,3	1,56E-44	4,7E-34	3,06E-24	4,32E-15	1,04E-06	5,65E-54	2,95E-43	3,75E-33	1,25E-23
6,4	1,27E-29	1,23E-22	4,29E-16	5,4E-10	0,000208	6,45E-36	9,01E-29	4,92E-22	1,1E-15
6,5	5,52E-15	1,72E-11	3,21E-08	3,6E-05	0,022225	3,94E-18	1,47E-14	3,43E-11	5,13E-08
7,2	1,38E-74	4,04E-57	9,21E-41	1,63E-25	1,53E-11	2,55E-90	1,86E-72	1,29E-55	9,64E-40
7,3	2,5E-59	2,35E-45	2,87E-32	4,54E-20	6,78E-09	6,48E-72	1,26E-57	3,76E-44	1,88E-31
7,4	2,72E-44	8,22E-34	5,36E-24	7,56E-15	1,81E-06	9,88E-54	5,15E-43	6,57E-33	2,19E-23
7,5	1,78E-29	1,72E-22	6,01E-16	7,56E-10	0,00029	9,04E-36	1,26E-28	6,88E-22	1,54E-15
7,6	6,44E-15	2E-11	3,75E-08	4,2E-05	0,025881	4,59E-18	1,72E-14	4,01E-11	5,99E-08
8,2	1,69E-89	1,54E-68	6,57E-49	1,31E-30	7,6E-14	2,2E-108	6,07E-87	9,86E-67	1,1E-47
8,3	3,68E-74	1,08E-56	2,46E-40	4,35E-25	4,05E-11	6,8E-90	4,96E-72	3,44E-55	2,57E-39
8,4	5,01E-59	4,71E-45	5,74E-32	9,07E-20	1,35E-08	1,3E-71	2,53E-57	7,52E-44	3,75E-31
8,5	4,36E-44	1,32E-33	8,58E-24	1,21E-14	2,89E-06	1,58E-53	8,25E-43	1,05E-32	3,51E-23
8,6	2,37E-29	2,3E-22	8,02E-16	1,01E-09	0,000386	1,2E-35	1,68E-28	9,18E-22	2,05E-15
8,7	7,36E-15	2,29E-11	4,28E-08	4,8E-05	0,029523	5,25E-18	1,96E-14	4,58E-11	6,85E-08
9,2	2E-104	5,69E-80	4,52E-57	1,01E-35	3,65E-16	1,9E-126	1,9E-101	7,26E-78	1,21E-55
9,3	5,08E-89	4,63E-68	1,97E-48	3,92E-30	2,27E-13	6,7E-108	1,82E-86	2,96E-66	3,3E-47
9,4	8,29E-74	2,43E-56	5,53E-40	9,8E-25	9,09E-11	1,53E-89	1,11E-71	7,75E-55	5,78E-39
9,5	9,01E-59	8,47E-45	1,03E-31	1,63E-19	2,43E-08	2,33E-71	4,55E-57	1,35E-43	6,76E-31
9,6	6,53E-44	1,97E-33	1,29E-23	1,81E-14	4,32E-06	2,37E-53	1,24E-42	1,58E-32	5,27E-23
9,7	3,04E-29	2,95E-22	1,03E-15	1,3E-09	0,000495	1,55E-35	2,16E-28	1,18E-21	2,64E-15
9,8	8,28E-15	2,58E-11	4,82E-08	5,4E-05	0,033152	5,9E-18	2,21E-14	5,15E-11	7,7E-08
	8,6	8,7	9,2	9,3	9,4	9,5	9,6	9,7	9,8
4,2	3,84E-10	0,000109	1,22E-42	2,32E-35	1,82E-28	6,37E-22	9,88E-16	6,34E-10	0,000138
4,3	3,2E-05	0,016979	1,8E-21	7,87E-18	2,21E-14	4,12E-11	5,13E-08	4,11E-05	0,019081
5,2	5,11E-15	7,75E-07	9,19E-64	7,61E-53	1,68E-42	1,09E-32	2,11E-23	1,09E-14	1,1E-06
5,3	6,4E-10	0,000181	2,04E-42	3,87E-35	3,04E-28	1,06E-21	1,65E-15	1,06E-09	0,000229
5,4	4E-05	0,021178	2,26E-21	9,84E-18	2,76E-14	5,15E-11	6,42E-08	5,14E-05	0,023794
6,2	6,14E-20	4,96E-09	6,22E-85	2,25E-70	1,39E-56	1,69E-43	4,07E-31	1,67E-19	7,93E-09
6,3	1,02E-14	1,54E-06	1,84E-63	1,52E-52	3,35E-42	2,19E-32	4,23E-23	2,17E-14	2,19E-06
6,4	9,59E-10	0,000271	3,05E-42	5,8E-35	4,56E-28	1,59E-21	2,47E-15	1,58E-09	0,000342
6,5	4,8E-05	0,02536	2,71E-21	1,18E-17	3,31E-14	6,18E-11	7,7E-08	6,17E-05	0,028484
7,2	6,87E-25	2,97E-11	3,9E-106	6,18E-88	1,07E-70	2,44E-54	7,31E-39	2,41E-24	5,33E-11
7,3	1,43E-19	1,15E-08	1,45E-84	5,24E-70	3,23E-56	3,94E-43	9,49E-31	3,91E-19	1,84E-08

7,4	1,79E-14	2,69E-06	3,22E-63	2,66E-52	5,86E-42	3,83E-32	7,4E-23	3,8E-14	3,83E-06
7,5	1,34E-09	0,000378	4,28E-42	8,13E-35	6,38E-28	2,23E-21	3,46E-15	2,22E-09	0,000477
7,6	5,6E-05	0,029523	3,16E-21	1,38E-17	3,86E-14	7,21E-11	8,98E-08	7,19E-05	0,033152
8,2	7,32E-30	1,69E-13	2,4E-127	1,6E-105	7,86E-85	3,34E-65	1,25E-46	3,3E-29	3,42E-13
8,3	1,83E-24	7,88E-11	1E-105	1,65E-87	2,85E-70	6,49E-54	1,95E-38	6,42E-24	1,42E-10
8,4	2,86E-19	2,3E-08	2,9E-84	1,05E-69	6,46E-56	7,88E-43	1,9E-30	7,81E-19	3,67E-08
8,5	2,86E-14	4,3E-06	5,14E-63	4,26E-52	9,38E-42	6,12E-32	1,18E-22	6,08E-14	6,1E-06
8,6	1,79E-09	0,000502	5,7E-42	1,08E-34	8,51E-28	2,97E-21	4,61E-15	2,96E-09	0,000634
8,7	6,4E-05	0,033669	3,61E-21	1,57E-17	4,41E-14	8,24E-11	1,03E-07	8,22E-05	0,037797
9,2	7,53E-35	9,28E-16	1,4E-148	4,1E-123	5,6E-99	4,43E-76	2,06E-54	4,36E-34	2,11E-15
9,3	2,2E-29	5,05E-13	7,1E-127	4,9E-105	2,36E-84	1E-64	3,75E-46	9,9E-29	1,02E-12
9,4	4,12E-24	1,77E-10	2,4E-105	3,71E-87	6,41E-70	1,46E-53	4,38E-38	1,45E-23	3,17E-10
9,5	5,15E-19	4,13E-08	5,22E-84	1,89E-69	1,16E-55	1,42E-42	3,42E-30	1,41E-18	6,59E-08
9,6	4,3E-14	6,42E-06	7,72E-63	6,39E-52	1,41E-41	9,18E-32	1,77E-22	9,12E-14	9,12E-06
9,7	2,3E-09	0,000644	7,33E-42	1,39E-34	1,09E-27	3,82E-21	5,93E-15	3,8E-09	0,000813
9,8	7,2E-05	0,037797	4,06E-21	1,77E-17	4,96E-14	9,27E-11	1,15E-07	9,25E-05	0,042421

Таблица 26 — Расчет вероятностей отказа двухуровневой модели (на 1-м и 2-м уровнях используется избыточная СОК), HGST 3 TB

	4,2	4,3	5,2	5,3	5,4	6,2	6,3	6,4	6,5
4,2	1,71E-11	2,69E-05	1,34E-17	4,74E-11	4,2E-05	8,45E-24	5,34E-17	1,07E-10	6,04E-05
4,3	6,75E-06	0,008454	5,97E-09	1,12E-05	0,010557	4,75E-12	1,19E-08	1,69E-05	0,012655
5,2	4,8E-17	9,5E-08	3,32E-26	2,22E-16	1,85E-07	1,67E-35	2,65E-25	7,49E-16	3,2E-07
5,3	2,85E-11	4,48E-05	2,23E-17	7,9E-11	6,98E-05	1,41E-23	8,89E-17	1,78E-10	0,0001
5,4	8,44E-06	0,010557	7,46E-09	1,41E-05	0,013178	5,94E-12	1,49E-08	2,11E-05	0,015793
6,2	1,22E-22	3,02E-10	7,43E-35	9,37E-22	7,36E-10	2,98E-47	1,19E-33	4,74E-21	1,52E-09
6,3	9,61E-17	1,9E-07	6,64E-26	4,44E-16	3,7E-07	3,35E-35	5,31E-25	1,5E-15	6,38E-07
6,4	4,27E-11	6,71E-05	3,34E-17	1,19E-10	0,000105	2,11E-23	1,33E-16	2,67E-10	0,00015
6,5	1,01E-05	0,012655	8,95E-09	1,69E-05	0,015793	7,12E-12	1,79E-08	2,53E-05	0,018922
7,2	2,87E-28	8,97E-13	1,55E-43	3,69E-27	2,73E-12	4,95E-59	4,96E-42	2,8E-26	6,78E-12
7,3	2,84E-22	7,04E-10	1,73E-34	2,19E-21	1,71E-09	6,95E-47	2,77E-33	1,11E-20	3,55E-09
7,4	1,68E-16	3,32E-07	1,16E-25	7,78E-16	6,46E-07	5,85E-35	9,28E-25	2,62E-15	1,11E-06
7,5	5,98E-11	9,37E-05	4,67E-17	1,66E-10	0,000146	2,96E-23	1,87E-16	3,73E-10	0,00021
7,6	1,18E-05	0,014748	1,04E-08	1,97E-05	0,018401	8,31E-12	2,09E-08	2,95E-05	0,02204
8,2	6,46E-34	2,53E-15	3,09E-52	1,38E-32	9,65E-15	7,83E-71	1,97E-50	1,57E-31	2,87E-14
8,3	7,66E-28	2,39E-12	4,14E-43	9,83E-27	7,27E-12	1,32E-58	1,32E-41	7,45E-26	1,8E-11
8,4	5,67E-22	1,41E-09	3,47E-34	4,37E-21	3,42E-09	1,39E-46	5,54E-33	2,21E-20	7,07E-09
8,5	2,69E-16	5,3E-07	1,86E-25	1,24E-15	1,03E-06	9,37E-35	1,49E-24	4,19E-15	1,78E-06
8,6	7,97E-11	0,000125	6,23E-17	2,21E-10	0,000195	3,95E-23	2,49E-16	4,98E-10	0,000279
8,7	1,35E-05	0,016837	1,19E-08	2,25E-05	0,021002	9,5E-12	2,39E-08	3,37E-05	0,025149
9,2	1,4E-39	6,91E-18	5,92E-61	5E-38	3,29E-17	1,2E-82	7,56E-59	8,52E-37	1,17E-16
9,3	1,94E-33	7,59E-15	9,26E-52	4,15E-32	2,89E-14	2,35E-70	5,91E-50	4,71E-31	8,6E-14
9,4	1,72E-27	5,36E-12	9,31E-43	2,21E-26	1,63E-11	2,97E-58	2,97E-41	1,68E-25	4,05E-11
9,5	1,02E-21	2,53E-09	6,24E-34	7,87E-21	6,15E-09	2,5E-46	9,97E-33	3,98E-20	1,27E-08
9,6	4,04E-16	7,93E-07	2,79E-25	1,87E-15	1,54E-06	1,4E-34	2,23E-24	6,29E-15	2,66E-06
9,7	1,03E-10	0,00016	8,01E-17	2,85E-10	0,00025	5,07E-23	3,2E-16	6,4E-10	0,000358
9,8	1,52E-05	0,018922	1,34E-08	2,53E-05	0,023596	1,07E-11	2,68E-08	3,79E-05	0,028248
	7,2	7,3	7,4	7,5	7,6	8,2	8,3	8,4	8,5
4,2	4,66E-30	4,6E-23	1,63E-16	2,09E-10	8,21E-05	2,33E-36	3,31E-29	1,84E-22	4,18E-16
4,3	3,53E-15	1,11E-11	2,09E-08	2,36E-05	0,014748	2,49E-18	9,4E-15	2,21E-11	3,34E-08
5,2	6,85E-45	2,12E-34	1,42E-24	2,05E-15	5,07E-07	2,42E-54	1,3E-43	1,7E-33	5,81E-24
5,3	7,77E-30	7,67E-23	2,72E-16	3,48E-10	0,000136	3,89E-36	5,52E-29	3,06E-22	6,96E-16
5,4	4,41E-15	1,38E-11	2,61E-08	2,95E-05	0,018401	3,12E-18	1,17E-14	2,77E-11	4,17E-08
6,2	9,06E-60	8,81E-46	1,11E-32	1,82E-20	2,82E-09	2,27E-72	4,57E-58	1,41E-44	7,27E-32
6,3	1,37E-44	4,24E-34	2,84E-24	4,11E-15	1,01E-06	4,85E-54	2,6E-43	3,39E-33	1,16E-23
6,4	1,17E-29	1,15E-22	4,08E-16	5,22E-10	0,000204	5,83E-36	8,28E-29	4,6E-22	1,04E-15

6,5	5,29E-15	1,66E-11	3,13E-08	3,54E-05	0,02204	3,74E-18	1,41E-14	3,32E-11	5,01E-08
7,2	1,12E-74	3,42E-57	8,12E-41	1,5E-25	1,46E-11	1,98E-90	1,5E-72	1,09E-55	8,49E-40
7,3	2,11E-59	2,06E-45	2,59E-32	4,24E-20	6,55E-09	5,29E-72	1,07E-57	3,28E-44	1,7E-31
7,4	2,4E-44	7,43E-34	4,97E-24	7,19E-15	1,76E-06	8,49E-54	4,54E-43	5,93E-33	2,03E-23
7,5	1,63E-29	1,61E-22	5,72E-16	7,31E-10	0,000285	8,17E-36	1,16E-28	6,43E-22	1,46E-15
7,6	6,17E-15	1,94E-11	3,65E-08	4,13E-05	0,025666	4,36E-18	1,64E-14	3,87E-11	5,84E-08
8,2	1,31E-89	1,26E-68	5,65E-49	1,18E-30	7,23E-14	1,6E-108	4,71E-87	8,05E-67	9,45E-48
8,3	2,98E-74	9,11E-57	2,16E-40	4E-25	3,89E-11	5,28E-90	4,01E-72	2,91E-55	2,27E-39
8,4	4,23E-59	4,11E-45	5,19E-32	8,48E-20	1,31E-08	1,06E-71	2,13E-57	6,57E-44	3,39E-31
8,5	3,84E-44	1,19E-33	7,95E-24	1,15E-14	2,81E-06	1,36E-53	7,27E-43	9,49E-33	3,25E-23
8,6	2,18E-29	2,15E-22	7,62E-16	9,75E-10	0,000379	1,09E-35	1,55E-28	8,58E-22	1,95E-15
8,7	7,05E-15	2,21E-11	4,17E-08	4,72E-05	0,029279	4,99E-18	1,88E-14	4,43E-11	6,68E-08
9,2	1,5E-104	4,49E-80	3,79E-57	8,95E-36	3,44E-16	1,3E-126	1,4E-101	5,73E-78	1,01E-55
9,3	3,94E-89	3,78E-68	1,69E-48	3,54E-30	2,16E-13	4,9E-108	1,41E-86	2,41E-66	2,84E-47
9,4	6,71E-74	2,05E-56	4,87E-40	9E-25	8,72E-11	1,19E-89	9,03E-72	6,54E-55	5,1E-39
9,5	7,61E-59	7,4E-45	9,33E-32	1,53E-19	2,35E-08	1,9E-71	3,84E-57	1,18E-43	6,11E-31
9,6	5,76E-44	1,78E-33	1,19E-23	1,72E-14	4,21E-06	2,04E-53	1,09E-42	1,42E-32	4,88E-23
9,7	2,8E-29	2,76E-22	9,8E-16	1,25E-09	0,000486	1,4E-35	1,99E-28	1,1E-21	2,51E-15
9,8	7,93E-15	2,49E-11	4,7E-08	5,31E-05	0,032878	5,61E-18	2,11E-14	4,98E-11	7,51E-08
	8,6	8,7	9,2	9,3	9,4	9,5	9,6	9,7	9,8
4,2	3,71E-10	0,000107	1,09E-42	2,1E-35	1,68E-28	5,95E-22	9,39E-16	6,13E-10	0,000135
4,3	3,15E-05	0,016837	1,7E-21	7,48E-18	2,11E-14	3,98E-11	5E-08	4,04E-05	0,018922
5,2	4,86E-15	7,55E-07	7,69E-64	6,54E-53	1,48E-42	9,88E-33	1,96E-23	1,03E-14	1,07E-06
5,3	6,18E-10	0,000178	1,81E-42	3,5E-35	2,79E-28	9,92E-22	1,57E-15	1,02E-09	0,000225
5,4	3,93E-05	0,021002	2,13E-21	9,35E-18	2,64E-14	4,98E-11	6,26E-08	5,05E-05	0,023596
6,2	5,73E-20	4,8E-09	4,91E-85	1,83E-70	1,17E-56	1,48E-43	3,68E-31	1,56E-19	7,67E-09
6,3	9,72E-15	1,51E-06	1,54E-63	1,31E-52	2,95E-42	1,98E-32	3,92E-23	2,06E-14	2,14E-06
6,4	9,27E-10	0,000266	2,71E-42	5,24E-35	4,19E-28	1,49E-21	2,35E-15	1,53E-09	0,000336
6,5	4,72E-05	0,025149	2,55E-21	1,12E-17	3,17E-14	5,98E-11	7,51E-08	6,06E-05	0,028248
7,2	6,31E-25	2,84E-11	2,9E-106	4,8E-88	8,66E-71	2,06E-54	6,44E-39	2,21E-24	5,11E-11
7,3	1,34E-19	1,12E-08	1,15E-84	4,28E-70	2,73E-56	3,44E-43	8,58E-31	3,65E-19	1,78E-08
7,4	1,7E-14	2,63E-06	2,69E-63	2,29E-52	5,17E-42	3,46E-32	6,85E-23	3,61E-14	3,73E-06
7,5	1,3E-09	0,000372	3,8E-42	7,34E-35	5,87E-28	2,08E-21	3,29E-15	2,14E-09	0,000469
7,6	5,5E-05	0,029279	2,98E-21	1,31E-17	3,7E-14	6,97E-11	8,76E-08	7,07E-05	0,032878
8,2	6,62E-30	1,61E-13	1,7E-127	1,2E-105	6,1E-85	2,73E-65	1,07E-46	2,98E-29	3,25E-13
8,3	1,68E-24	7,56E-11	7,8E-106	1,28E-87	2,31E-70	5,48E-54	1,72E-38	5,9E-24	1,36E-10
8,4	2,68E-19	2,22E-08	2,29E-84	8,56E-70	5,46E-56	6,88E-43	1,72E-30	7,3E-19	3,55E-08
8,5	2,72E-14	4,19E-06	4,31E-63	3,66E-52	8,27E-42	5,53E-32	1,1E-22	5,78E-14	5,95E-06
8,6	1,73E-09	0,000494	5,06E-42	9,79E-35	7,82E-28	2,78E-21	4,38E-15	2,86E-09	0,000624
8,7	6,29E-05	0,033391	3,4E-21	1,5E-17	4,23E-14	7,97E-11	1E-07	8,08E-05	0,037485
9,2	6,69E-35	8,75E-16	9,1E-149	2,9E-123	4,1E-99	3,5E-76	1,73E-54	3,88E-34	1,99E-15
9,3	1,99E-29	4,8E-13	5E-127	3,6E-105	1,83E-84	8,19E-65	3,22E-46	8,95E-29	9,7E-13
9,4	3,79E-24	1,69E-10	1,8E-105	2,88E-87	5,19E-70	1,23E-53	3,86E-38	1,33E-23	3,04E-10
9,5	4,82E-19	3,99E-08	4,12E-84	1,54E-69	9,83E-56	1,24E-42	3,09E-30	1,31E-18	6,37E-08
9,6	4,08E-14	6,26E-06	6,46E-63	5,49E-52	1,24E-41	8,3E-32	1,65E-22	8,67E-14	8,89E-06
9,7	2,23E-09	0,000633	6,51E-42	1,26E-34	1,01E-27	3,57E-21	5,64E-15	3,68E-09	0,000799
9,8	7,08E-05	0,037485	3,83E-21	1,68E-17	4,76E-14	8,96E-11	1,13E-07	9,1E-05	0,042071