ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ «МИСИС»

Хаммуд Обадах

Модели и алгоритмы автономного распределения данных и управления доступом на базе смарт-контрактов

Специальность 2.3.1 – Системный анализ,
управление и обработка информации, статистика

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Научный руководитель
к.т.н. Тарханов И.А.

Москва 2024

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность

Информационные системы обмена данными на основе блокчейн (DApps) становятся все популярнее. По данным DApps Industry Report 2023 количество пользователей таких систем в течении только одного года выросло в 2 раза и достигло 4 миллиона, а капитализация рынка DApps приложений превысила 100 миллиардов долларов. Особое развитие такого рода системы получили в финансовом секторе и индустрии игр. При этом случаи внедрения блокчейн в государственных системах также увеличиваются. Основным преимуществом DApps по сравнению с централизованными системами является то, что они позволяют организовать доверенный обмен информаций между участниками, которые изначально друг другу не доверяют. Но очевидны и проблемы таких систем – это избыточность данных, скорость их обработки, а также в некоторых случаях повышенное энергопотребление. Именно необходимость многократно дублировать данные хранящиеся в блокчейн на всех узлах системы является существенным ограничением для внедрения DApps в реальных секторах экономки.

Требования к обработке данных в современных информационных системах со временем резко возрастают по мере увеличения масштабов этих систем и размера ежедневно генерируемых ими данных. В случае распределенных систем возникает необходимость балансировки нагрузки между узлами системы, чтобы избежать проблем с увеличением нагрузки на отдельные узлы системы, количества данных хранящихся в ней, а также количества узлов, где эти данные могут хранится.

Равномерное распределение данных является сложной задачей, поскольку объекты имеют разные размеры и наличие большого количества разнородных объектов может привести к высоким требованиям к ресурсам всех узлов системы (загрузке процессора, количестве выделяемой для хранения памяти на носителях), а также к пропускной способности между узлами. Известные научные подходы и распространенные облачные платформы не позволяют балансировать нагрузку без многократного дублирования данных на узлах одновременно с сохранением возможности увеличения узлов системы и количества объектов в ней (свойство масштабируемости системы).

Основным элементом DApps являются смарт-контракты, которые могут выполняться на узлах блокчейна и по сути являются самоисполняющимися программами. Именно с помощью смарт-контрактов возможно организовать балансировку нагрузки без участия пользователя. Однако в научной литературе не описаны подходы, которые решали бы такую задачу. В первую очередь это обусловлено наличием технических ограничений языка программирования смарт-контрактов.

Необходимо отметить, что, как и в любой информационной системе, в DApps нужно управлять доступом к объектам системы. В случае распределённой системы важно тратить на это минимум ресурсов при увеличении количества объектов и узлов в ней, а также сохранить устойчивость от сбоев, т.е. она не должна содержать централизованных компонент. Наиболее распространенные модели контроля доступа в информационных системах (DAC, RBAC) являются гибкими и удобными для администрирования доступа к объектам, но по сути являются централизованными и не подходят для использования в DApps.

В данной работе предлагаются алгоритмы и модели распределения данных, которые позволяют реализовать с их помощью DApps и одновременно обойти противоречие в избыточности данных. Также работа посвящена созданию модели управления доступом к данным, которая минимизирует размер хранилища данных прав доступа к объектам DApps, обеспечивая при этом способность увеличивать количество узлов и общий объем данных системы без ограничений.

Важным аспектом внедрения DApps помимо проблемы избыточности данных является оценка целесообразности внедрения такого рода систем по сравнению с системами централизованной архитектуры. Основным критерием здесь является обеспечение определенного уровня надежности (не хуже, чем в централизованных системах) при минимуме ресурсов. Для корректного сравнения надежности распределенных DApps и централизованных систем необходимо разработать новый подход к расчету надежности, который учитывает особенности работы балансировщика нагрузки, что является одной из задач данной работы.

Рассматриваемый в данной работе класс информационных систем не подходит для задач хранения объектов, требующих непосредственной обработки внутри DApps (прямое изменение данных внутри системы, запись видео непосредственно на диски системы и т.д.). Все операции чтения, записи и удаления объектов осуществляются только через балансировщик нагрузки, а размер файла должен быть заранее известен и не должен постепенно изменяться, так как размер файла является важным параметром для выбора местоположения файла.

В работе решается научная задача создания системного подхода для разработки DApps систем, который позволяет избавиться от избыточности данных в распределённой среде при этом обеспечить равномерное распределение данных по узлам системы, необходимую надежность системы (не хуже, чем в централизованных системах).

Целью работы

Целью работы является разработка моделей и алгоритмов управления распределением данных и контроля доступа в децентрализованной информационной системе, которые отличаются от известных тем, что:

- обеспечивают равномерную автономную балансировку нагрузки при распределении данных для любого заданного количества узлов с помощью смарт-контрактов,

- позволяют минимизировать размер хранилища на 25% по сравнению с информационными системами с полным резервным копированием каждого узла и с надежностью в целом не хуже, чем в информационных системах с централизованной архитектурой.

Идея работы заключается в сочетании гибридной архитектуры системы, в которой часть данных хранится в блокчейн, а часть в локальных хранилищах узлов системы. Управление распределением данных реализуется набором алгоритмов, созданных для автономного выполнения в среде смарт-контрактов блокчейн. Для минимизации ресурсов применяется метод стирающего кодирования (Erasure Coding), а также алгоритмы уменьшения размера данных и кэширования запросов, позволяющие управлять доступом к объектам в распределенной системе.

Задачи:

- Исследование параметров эффективности функционирования известных методов распределения данных, таких как общий размер необходимого дискового пространства для хранения объектов и их резервных копий, возможностей увеличения количества узлов, равномерность распределения данных по узлам, надежность системы в целом.

- Выявление ограничений известных моделей контроля доступа, которые можно применять в распределенных системах, и способов снятия этих ограничений.

- Разработка модели распределения данных, которая уменьшает использование памяти на узлах в условиях динамического изменения топологии сети и учитывает ограничения, накладываемые смарт-контрактами.

- Разработка распределенной модели управления доступом, решающей проблему увеличения количества узлов и объектов хранения для DApps систем без использования централизованных компонентов.

- Разработка модели расчета надежности для оценки предложенной модели в определенном состоянии и сравнения с другими подходами.

Методы исследования включают теоретико-множественные методы системного анализа для разработки предлагаемой модели, принципы распределения вероятностей и теории вероятностей, модели надежности, теорию игр, теорию принятия решений, методы математического моделирования, теоретико-информационный анализ систем хранения и распространения файлов, а также систем контроля доступа. Принципы статистики также используются для оценки полученных результатов и сравнения с другими системами.

Объектом исследования являются распределенные информационные системы, которые решают задачу конфиденциального обмена данными между участниками, которые не доверяют

друг другу.

Предметом исследования являются системные показатели эффективности функционирования децентрализованных систем и методы управления распределением данных, а также технологические ограничения применения данных методов при работе с блокчейн.

Научные положения, выносимые на защиту:

1) Разработанная модель распределения данных в DApps и 4 алгоритма балансировки нагрузки позволяют организовать равномерное распределение данных по узлам и уменьшить суммарный объем необходимого места на всех узлах на 25% по сравнению с системами полного резервного копирования.

2) Предложенная модель и алгоритмы контроля доступа позволяют минимизировать размер данных хранящихся в блокчейн и избежать их централизованного хранения, что решает задачу увеличения количества узлов и объектов в системе без ущерба для надежности и производительности.

3) Разработанная модель надежности учитывает автоматическое восстановление данных на основе количества узлов, доступного пространства на узлах и времени передачи данных, что позволяет корректно рассчитывать надежность распределённых систем на основе блокчейн и сравнивать их с системами другой архитектуры.

Новизна научных исследований заключается в следующем:

1) Предложены модель распределения данных и оперирующие этой моделью 4 алгоритма балансировки нагрузки, которые минимизируют избыточность данных и отличаются от известных тем, что гарантируют справедливое распределение данных между узлами, а также обеспечивают возможность автоматического восстановления данных на узлах за счет того, что сочетают метод стирающего кодирования, распределение по виртуальным кластерам и реализацию алгоритмов балансировки в среде смарт-контрактов в автономном режиме.

2) Предложены новая модель и алгоритмы управления доступом в распределенной системе, базирующиеся на известной DAC и RBAC моделях, которые в отличие от известных моделей разграничения доступа минимизируют необходимое пространство для хранения на балансировщике нагрузки и распределяют данные по узлам таким образом, чтобы поддерживать увеличение количества узлов и хранящихся в системе объектов на основе кэширования.

3) Предложена новая модель оценки надежности распределенных систем, основанная на модели оценки надежности для RAID и отличающийся от известных тем, что учитывает возможность восстановления потерянных данных на существующих узлах на основе информации о свободной памяти на узлах, количестве узлов и времени восстановления данных в системе.

Обоснованность и достоверность результатов исследования обеспечиваются: репрезентативностью исходных данных для моделирования поведения системы основанной на данных полученных из реального проекта коммерческой компании; использованием современного программного обеспечения, оборудования и апробированных методик для оценки надежности и моделирования.

Практическая значимость

Предложенные модели, включая модель распространения данных, а также модель контроля доступа, могут применяться в государственных и корпоративных системах многих отраслей экономики:

1) Для государственных систем предложенные модели могут быть полезны в сценариях, когда необходимо организовать проверку и аудит со стороны государства деятельности компаний, где существуют ограничения по ресурсам и требования к надежности. Например, контроль лицензирования различных видов деятельности (финансы, торговля требующими государственного регулирования товарами, оказание эксклюзивных услуг), сбор налоговых пошлин, предоставление финансовой отчетности для получения льгот и т. д.

2) Предложенные модели и алгоритмы могут быть использованы в качестве инфраструктуры облачных сервисов для корпоративного обмена данными, которые требуют хранения больших объемов файлов транзакций, изображений, аудио и видео библиотек. Они могут лечь в основу нового поколения распределённых систем для хранения любых информационных объектов или выполнять функции распределённой базы данных.

3) Предложенные модели также могут использоваться в DApps сервисах ориентированных на конечных пользователей. Например, в сервисах управления и предоставления доступа к данным обучения нейронных сетей, NFT системам, в которых хранятся файлы, метаданные файлов или сложные структурированные объекты.

Реализация выводов и рекомендаций работы

Основные положения диссертации прошли апробацию в проектах ООО «РИИТ». Подсистема распределённого хранения подключена к программной платформе «Рубикон», которая обеспечивает защищенный обмен данными между организациями, что подтверждается соответствующим актом внедрения. Зарегистрирована программа ЭВМ «DecStore: балансировщик нагрузки на базе смарт-контракта для распределённого хранения данных», регистрационный номер 2023682504 от 26.10.2023.

Публикации. Материалы диссертации опубликованы в 10 научных работах, в том числе в 3-х рекомендованных ВАК РФ и одна работа из перечня изданий, индексируемых в международных библиографических базах данных Scopus и Web of Science.

Соответствие паспорту специальности

Соответствует П5 «Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта», т.к. одной из задач диссертации является разработка программного и алгоритмического обеспечения системы распределения данных (обработки информации).

Соответствует П9 «Разработка проблемно-ориентированных систем управления, принятия решений и оптимизации технических объектов», т.к. в диссертации решается задача разработки модели управления доступом в децентрализованной системе.

Соответствует П11 «Методы и алгоритмы прогнозирования и оценки эффективности, качества, надежности функционирования сложных систем управления и их элементов», т.к. в диссертации решается задача оценки надежности функционирования систем на основе блокчейн.

Объем и структура диссертации. Диссертация состоит из введения, 4 глав, заключения, библиографического списка из 136 наименований и представлена на 148 страницах, включая 43 рисунков, 16 таблиц.

Содержание работы

В главе 1 — ПРОБЛЕМЫ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ РАСПРЕДЕЛЕНИЯ ДАННЫХ И УПРАВЛЕНИЯ ДОСТУПОМ К НИМ В ДЕЦЕНТРАЛИЗОВАННЫХ СИСТЕМАХ

Приводится обзор проектов и научных подходов изучающие вопрос управления данными в распределенных системах, которые рассматривают набор параметров: минимальный размер необходимых данных, показатели надежности и возможность масштабируемости (рост количества узлов и объектов в системе). Также приводится обзор научных подходов к моделям контроля доступа на основе блокчейна. В данной работе используются следующие обозначения.

Таблица 1 - Необходимые показатели для анализа систем

Показатель	Символ	Единица измерения
Множество узлов в системе	$\boldsymbol{N}$	Узел
Надежность системы	R	Процент
Объем свободного места, доступного для хранения объектов	$S`$	Байт
Отклонение значения размера хранилища в системе от равномерного распределения	ΔS	Байт
Размер входных объектов в системе	S_{Input}	Байт

Размер выходных объектов в системе	S_{Output}	Байт
Размер данных контроля доступа	$S_{AccessControl}$	Байт
Размер метаданных	$S_{Metadata}$	Байт
Размер дополнительных данных, используемых для повышения доступности/надежности объектов	$S_{Reducancy}$	Байт

Размер используемой системы хранения S_{Output} можно рассчитать следующим образом:

$$S_{Output} = S_{Input} + S_{Reducancy} + S_{Metadata} + S_{AccessControl} \quad (1)$$

Однако S_{Input} определяется размером входящего объекта, поэтому его невозможно оптимизировать, поскольку пользователи определяют, какие объекты хранить. По сравнению с входными объектами метаданные и данные контроля доступа считаются небольшими:

$$S_{Metadata} + S_{AccessControl} \ll S_{Input} \quad (2)$$

$$S_{Output} \approx S_{Input} + S_{Reducancy} \quad (3)$$

С учетом этих параметров систему распределения можно представить как $DistrSystem < S_{Output}, \Delta S, R, \boldsymbol{N} >$, где $\Delta S = \sum_{i=1}^{|N|} |\Delta S_i|$, $\Delta S_i = (SS_i - S_i) \rightarrow 0$: $\forall i \in \boldsymbol{N}$, $SS_i = \frac{C_i * S}{S` + S}$: $\forall i \in N$, C_i— размер хранилища на узле i. Соответственно, цель данной работы может быть определена как результат следующей оптимизации:

$$max_R min_{S_{Output}, \Delta S}\left(DistrSystem < S_{Output}, \Delta S, R, \boldsymbol{N} >\right) \quad (4)$$

В данной работе алгоритм стирающего кодирования используется предложенной моделью и алгоритмами для минимизации необходимого размера хранилища. RAID-5 и RAID-6 считаются одними из самых популярных реализаций стирающего кодирования в хранилищах данных. Несмотря на то, что они ориентированы на распределение данных по физическим дискам, эту концепцию можно расширить для использования на уровне узлов системы вместо физических дисков, что обеспечивает параллельную систему, что важно в контексте рассмотрения распределенных систем. Существуют и другие классы алгоритмов кодирования, которые могут минимизировать размер данных больше, чем на 25%, а также другие виды RAID. Однако RAID-5 и RAID-6 рассматриваются как базовые реализации, которые можно использовать для базовых сравнений и они представляют сбалансированное решение между минимизацией размер данных и скоростью их восстановления. В таблице 2 и 3 представлены результаты анализа проектов и научных подходов. Знак +/- означает, что свойство поддерживается, но не полностью.

Таблица 2 — Анализ научных подходов хранения и распределения данных

	$R \rightarrow \max$	$\lvert N \rvert \neq const$	$S_{Output} \rightarrow \min$	$\Delta S \rightarrow 0$
Ceph	+/-	+	-	+
Raid5	-	-	+	-
Raid6	-	-	+	-
Storj	-	+	-	-
IPFS	-	+	+/-	-
Lustre	-	+	-	-

Таблица 3 — Анализ проектов хранения и распределения данных

	$R \rightarrow \max$	$\lvert N \rvert \neq const$	$S_{Output} \rightarrow \min$	$\Delta S \rightarrow 0$
HDFS	-	+	-	+
Gluster	+	+ / -	-	+/-
MS Azure	+	+/-	-	+/-
Amazon AWS	+	+/-	-	+/-

В конце обзора был сделан вывод, что обсуждаемые исследования не предлагают решения проблемы динамической балансировки нагрузки без многократного дублирования данных одновременно с сохранением возможности масштабирования системы и надежностью не хуже, чем в централизованных системах с одной резервной копией. В целом обсуждаемые научные исследования фокусируются на одном или подмножестве обсуждаемых параметров и не обеспечивают сбалансированное состояние этих параметров.

В обзоре также обсуждались научные подходы создания моделей контроля доступа на основе блокчейна. В целом научные подходы сосредоточены на том, как модели контроля доступа могут быть реализованы в DApps, как правила доступа между объектами и субъектами системы структурируются с использованием смарт-контрактов или как разделить несколько компонентов между смарт-контрактами. Задача минимизации размера хранения данных контроля доступа в блокчейне не рассматривается. Был сделан вывод, что существующие исследования не решают проблему контроля доступа на базе DAC и RBAC моделей в DApps, т.к. используемые в этих моделях матрицы доступа субъектов к объектам системы нельзя хранить в блокчейн при увеличении размера системы.

Результаты обзора опубликованы в статьях [1, 4, 5, 7, 8].

В главе 2 — МОДЕЛЬ РАСПРЕДЕЛЕНИЯ ДАННЫХ, АЛГОРИТМЫ БАЛАНСИРОВКИ ДАННЫХ И МОДЕЛЬ КОНТРОЛЯ ДОСТУПА К НИМ ДЛЯ МИНИМИЗАЦИИ ОБЩЕГО РАЗМЕРА ДАННЫХ СИСТЕМЫ

Использовались следующие определения:

Таблица 4 - Определения, необходимые для предлагаемых алгоритмов и моделей

VC	Виртуальный кластер в системе
VD	Виртуальный диск в системе
$\boldsymbol{VD_S}$	Множество виртуальных дисков в системе
$\boldsymbol{VD`_S}$	Множество пустых слотов виртуальных дисков в системе
$\boldsymbol{VC}$	Множество виртуальных кластеров в системе
$\boldsymbol{VD_{n(i)}}$	Подмножество $\boldsymbol{VD_S}$ расположенных в узле i
$\boldsymbol{VD_{n(i,\,x)}}$	Подмножество $\boldsymbol{VD_S}$ расположенных в узле i, которые имеют состояние x
$\boldsymbol{VD`_{n(i)}}$	Подмножество $\boldsymbol{VD`_S}$ расположенных в узле i.
$state(d)$	Функция, возвращающая состояние виртуального диска d. VD представляет одно из состояний hot(горячим), cold(холодным) или normal (обычным)
$\boldsymbol{VD_{index(i,j)}}$	Элемент множества $\boldsymbol{VD_S}$, расположенный в узле i и имеющий порядок j
$\boldsymbol{VD_{S(x)}}$	Подмножество $\boldsymbol{VD_S}$, которые имеют состояние x
$vd_{vc,\,p}$	Элемент множества $\boldsymbol{VD_S}$, представляющий vd, принадлежащий виртуальному кластеру vc и имеющий часть p. p может быть 1 для первой части, 2 для второй части, 3 для стирающего кодирования
$\boldsymbol{VC_{n(i)}}$	Подмножество $\boldsymbol{VC}$, которым принадлежат узлы i
$\boldsymbol{VC_i}$	Элемент множества $\boldsymbol{VC}$ с индексом i
$fsr(i)$	(Free space to request ratio) функция, возвращающая отношение свободного места к запросу для узла i
vc_d	Элемент множества $\boldsymbol{VC}$, который содержит конкретный виртуальный диск d
$p(d)$	Функция, возвращающая номер части виртуального диска d
$mcvd(x)$	(Mean count of VDs) – функция, которая возвращает среднее количество VD в состоянии x на узел
$MTTF_{system}$	Время наработки на отказ всей системы
$MTTF_{node}$	Время наработки на отказ узла
$MPRC$	Максимально возможное количество процессов восстановления
P_{SFANF}	Вероятность отказа системы
$MTTR$	Среднее время восстановления узла

Архитектура типовой DApps системы состоит из следующих частей (см. Рисунок 1). Архитектуру децентрализованной системы хранения, использующей блокчейн для децентрализованного управления, можно представить как на рисунке 2.

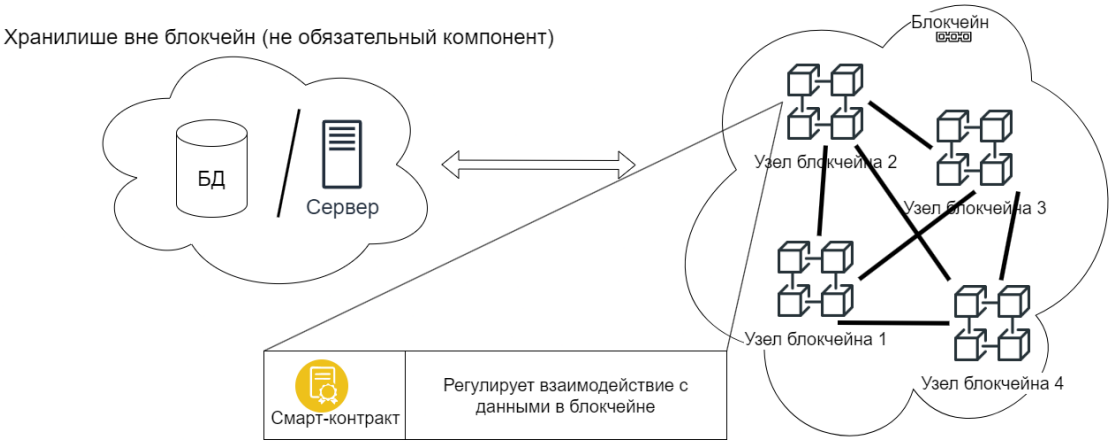

Рисунок 1 – Архитектура DApps системы

1. **Балансировщик нагрузки:** Балансировщик нагрузки обеспечивает необходимые функции управления и распределения данных. Он отвечает за следующие задачи:

a) Управление узлами хранения: управление существующими узлами, контроль процессов присоединения и выхода к узлам и т. д.

b) Управление данными: управление размещением существующих файлов (или объектов) и определение места добавления новых файлов. Предлагаемая модель распределения использует виртуальные объекты хранения, называемые виртуальными кластерами (VC) и виртуальными дисками (VD), которые будут объяснены в пункте 2 «Узлы хранения». Балансировщик нагрузки отвечает за процессы, связанные с этими виртуальными объектами, например за их распространение. Разработано четыре алгоритма распределения данных между узлами.

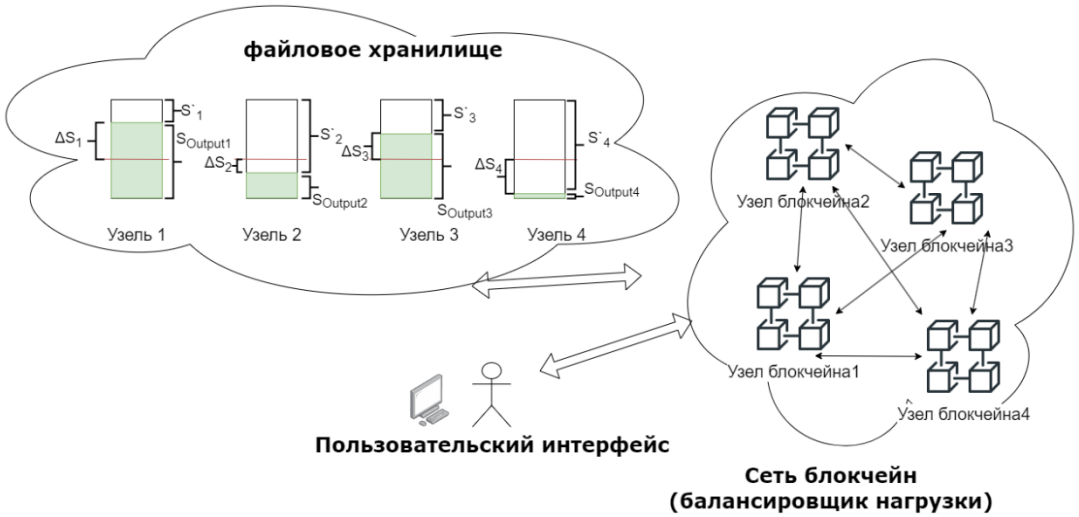

Рисунок 2 – Архитектура децентрализованной системы распределения данных на основе блокчейна

c) Управление пользователями: добавление и удаление пользователей, а также управление контролем доступа.

2. **Узлы хранения**: Система хранения объектов системы (в обобщенном случае - файлов) состоит из узлов (одноранговая сеть P2P, где все узлы имеют одинаковую роль). Файлы организованы в виртуальные кластеры (VC). В свою очередь, каждый из которых состоит из трёх виртуальных дисков фиксированного размера (VD). Это сделано для того, чтобы при увеличении VC и файлов в системе на основе рассматриваемой модели время работы алгоритмов управления распределением файлов по виртуальным дискам не зависело от общего количества файлов в системе. Очевидно, что количество VD в системе будет гораздо меньше количество файлов в ней. Важность минимизации сложности работы алгоритмов возрастает, поскольку предлагаемые алгоритмы реализованы на смарт-контрактах. Когда файл загружается пользователем или внешней информационной системой в систему хранения, он разбивается на две части и рассчитывается соответствующий стирающий код. Важно отметить, что размер стирающего кода равен размеру половины файла. В каждом кластере виртуальные диски организованы как индексированный массив. Первый VD используется для хранения первых половин файлов, второй VD для соответствующих вторых половин и третий для хранения стирающего кода. Такая схема распределения данных приводит к минимизации необходимого хранилища резервных данных на 25% по сравнению с системами хранения с одним сервером резервного копирования. На рисунке 3 представлена концепция предлагаемой модели.

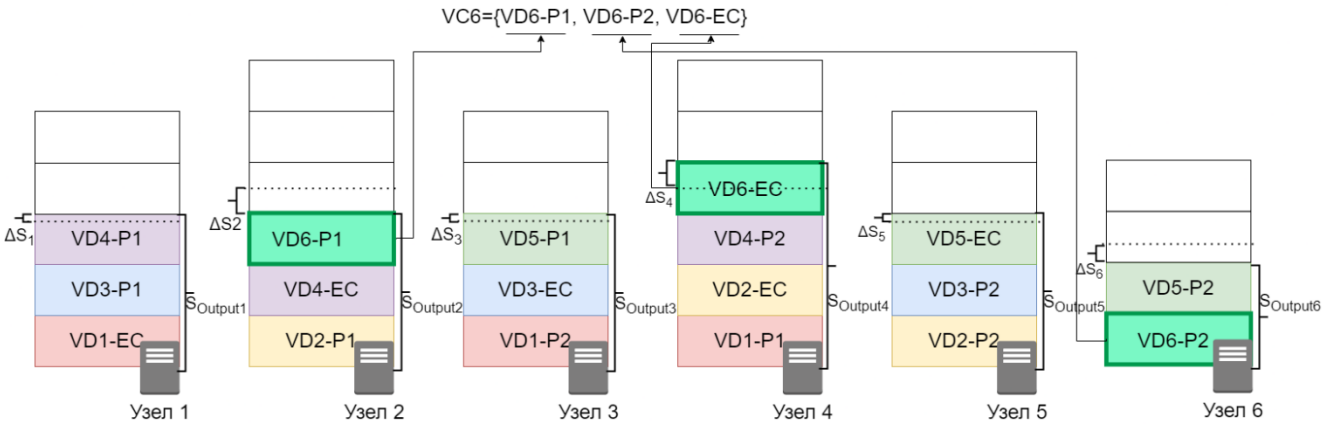

Рисунок 3 – Распределение VCs и VDs по узлам хранения

3. **Пользовательский интерфейс:** позволяет пользователю взаимодействовать с системой. Пользовательский интерфейс также отвечает за операции разделения и объединения файлов вместо узлов хранения, чтобы снизить нагрузку на систему.

Алгоритмы балансировки данных

При работе со смарт-контрактами существует несколько ограничений, которые учитываются при реализации алгоритмов балансировки данных, поскольку эти алгоритмы работают на смарт-контрактах:

1. Ограничения языка программирования: не все языки программирования поддерживаются в сети блокчейн: Многие сети блокчейнов используют язык программирования, специально разработанный для смарт-контрактов, например Solidity.
2. Сложные типы данных и структуры не поддерживаются, что ограничивает возможности работы с переменными. Например, сеть блокчейн Ethereum (Solidity) ограничивает сложность алгоритмов за счет расходования комиссий (Ethereum gas), а блокчейн платформа Hyperledger ограничивает максимальное время выполнения транзакций.
3. Ограниченные библиотеки и функциональность: смарт-контракты имеют ограниченные библиотеки и функциональность по сравнению с программированием общего назначения. Например, Solidity (один из самых популярных языков для написания смарт-контрактов) не поддерживает модификаторы, наследование, перегрузку функций и операторов, бесконечные циклы, рекурсии и числа с плавающей точкой.

Предложенные алгоритмы учитывают несколько ограничений при балансировке данных, в том числе поддержание равного распределения данных на основе свободного пространства узлов и размера самих данных. Также в смарт-контрактах реализуется требование о том, чтобы система не содержала на одном узле двух VD, принадлежащих одному и тому же VC, чтобы сделать возможность восстановления данных в случае отказа узлов и проверка состояния использования VD (горячие – VD, к которым обращаются чаще всего, холодные – VD используются для стирающего кодирования и обычные VD – остальные VD).

Было разработано 4 алгоритма:

1. Алгоритм добавления новых узлов (A1): ребалансирует данные при добавлении новых узлов.
2. Алгоритм добавления файлов (A2): выбирает или создаёт VC на узлах, на которых будет размещаться новый файл
3. Алгоритм восстановления потерянных данных (A3): восстанавливает содержимого потерянного узла на остальных узлах с использованием определенных условий.
4. Алгоритм возобновления узлов (A4): выполняет ребалансировку данных на узлах в тот момент, когда ранее отказавший узел повторно подключается к системе.

Рассмотрим детальнее алгоритм А3, который применяется для восстановления данных на других узлах в случае сбоя одного из узлов системы (см. Рисунок 4). Процесс восстановления инициализируется при следующих условиях:

- VD не может сосуществовать в одном узле с другим VD, принадлежащим тому же VC.

- Данные должны распределяться справедливо, так как процесс восстановления данных должен учитывать все доступные узлы.

Рисунок 4 – Алгоритм восстановления потерянных данных (А3).

Математически эту задачу можно рассматривать как классическую задачу теории игр, поскольку выигрыш может быть достигнут с помощью стратегии (способа распределения данных). Следовательно, при большом количестве VD и узлов, задачу восстановления

невозможно решить за полиномиальное время. В диссертации обсуждаются различные параметры игры, включая набор игроков, состояния игры, результаты и вероятность перехода. Обсуждается, как необходимо рассматривать все возможные переходы в виде дерева и изучать результаты каждого случая для существующих параметров (требуется проверить все возможные варианты). Эта проблема считается NP-полной проблемой. Реализация такого алгоритма в среде смарт-контрактов нецелесообразна. Вместо этого предлагается новый алгоритм. В этом алгоритме введена функция fsr, которая помогает ограничить количество узлов, которые могут восстановить потерянные виртуальные диски.

$$fsr(i) = \frac{|\boldsymbol{VD`}_{\boldsymbol{n(i)}}|}{\left|\boldsymbol{VD}_{\boldsymbol{n(lost\_node\_id)}}\right| - \left|\boldsymbol{VC}_{\boldsymbol{n(i)}} \cap \boldsymbol{VC}_{\boldsymbol{n(lost\_node\_id)}}\right|} \quad (5)$$

$$hdvc(i) = \left|\boldsymbol{VD}_{\boldsymbol{n(i,hot)}}\right| \quad (6)$$

$$m = \max\{fsr(i)\colon\ i = 1..|N|\backslash\{lost\_node\_id\}\} \quad (7)$$

$$j = fsr^{-1}(m) \quad (8)$$

$$l(j) = \begin{cases} \min\{hdvc(k)\}\colon \forall k \in j & state(d) = hot \\ \max\{hdvc(k)\}\colon \forall k \in j & else \end{cases} \quad (9)$$

$$Selected\_node\_for\_d = l^{-1}(j) \quad (10)$$

Предложенные модели и методы распределения данных опубликованы в статьях [3, 5, 9].

Модель контроля доступа

Предлагаемая модель контроля доступа минимизирует количество данных в блокчейн и количество обращений к определенным узлам системы для вычисления правил доступа. Модель контроля доступа состоит из следующих ключевых моментов:

- Каждый VD снабжен блоком, который называется «Компонент хранения разрешений». Для каждого пользователя, имеющего доступ хотя бы к одному файлу в указанном VD, в этом блоке создается дерево. Это дерево представляет файлы, организованные в каталогах, к которым пользователь имеет доступ через DAC. Единицы хранения, расположенные на VD, принадлежащих одному VC, имеют одинаковую копию деревьев.
- Каждый узел хранения содержит одно дерево для каждого пользователя, которое представляет собой комбинацию его деревьев DAC на VD, существующих в этом узле хранения.
- На стороне блокчейна у каждого пользователя хранится хэш, который представляет собой общий хэш корня его дерева Меркла. Если файлов миллион и пользователей 100, будет сохранено только 100 хэшей. Роли хранятся в блокчейне аналогично DAC. Для каждой

роли строится дерево Меркла, в котором файлы, доступные этой роли, рассматриваются как листья дерева.

В модели, предложенной в этом исследовании, точка информации о политике (PIP) не хранится на централизованном сервере. Когда PDP (точка принятия решения о политике, которая представлена в модели смарт-контрактом блокчейна) принимает решение, она получает информацию от двух объектов: пользователя и хранящихся в блокчейне записей. Информация о полном доступе хранится на узлах хранения, и пользователи синхронизируют свои собственные деревья доступа с деревьями, расположенными на узлах хранения. Точка применения политики (PEP), которая также представлена смарт-контрактом, отправляет в PDP доказательство дерева Меркла из запроса пользователя вместе с соответствующим ожидаемым корневым хэшем дерева Меркла.

Таким образом, PIP представлен тремя сущностями:

- Распределенная сеть узлов хранения, которая имеет полные политики и используется только для того, чтобы позволить пользователям синхронизировать свои политики.
- Сторона пользователя, где каждый пользователь хранит свои политики, будь то DAC или RBAC.
- Корневые хэши дерева Меркла как для пользователей, так и для ролей, которые хранятся в блокчейне.

Деревья пользователей состоят из трех типов узлов:

1. Корень представляет собой общий хэш дерева Меркла.
2. Листья представляют файлы (или каталоги, если политика разрешает пользователю доступ ко всем файлам в каталоге вместо выбора файлов вручную). Они могут иметь необязательное значение «compress».
3. Промежуточные узлы они могут быть каталоги или объединяющие узлы. Объединяющий узел — это узел, который объединяет два узла различных типов, причем любой из этих двух узлов может быть листом, каталогом или объединяющим узлом. Каждый узел дерева содержит две связанные структуры данных: хэш и значение. Хэш представляет собой хэш значения.

Основные этапы управления доступом можно описать следующим образом:

1. В каждом узле хранения создается дерево для каждого пользователя. В этом узле имеется набор файлов, к которым имеет доступ пользователь, которые существуют на виртуальных дисках с определенным заранее выбранным индексом. Для каждого пользователя все деревья с VD в узле объединяются в одно дерево, сжимаются и преобразуются в бинарное дерево. Эти деревья используются для DAC.

2. Также каждый узел хранения имеет несколько деревьев ролей. Существует дополнительная копия каждого дерева ролей, расположенная на другом узле. Все узлы хранения имеют одинаковое количество деревьев. Блокчейн распределяет деревья и их копии по узлам хранения.

3. Пользователи синхронизируют свои деревья с существующими на узлах хранения. Сюда входят их личные деревья (DAC) и деревья ролей (RBAC).

4. Пользователи используют свои копии деревьев для доступа к файлам, генерируя доказательство Меркла.

5. Блокчейн обрабатывает процессы, связанные с распространением и восстановлением деревьев в различных ситуациях, чтобы обеспечить балансировку системы и возможность восстановления файлов или автоматического восстановления на другой узел при потере узла хранения.

Для этого, были разработаны следующие алгоритмы:

1. Алгоритм создания дерева Меркла: создает дерево контроля доступа каждого пользователя на каждом узле хранения. Каждое дерево имеет политику доступа только к файлам, существующим на соответствующем узле

2. Алгоритм сжатия деревьев: уменьшает количество узлов в деревьях через слияния узлов, имеющих только один дочерний узел, будь то конечный или промежуточный узел.

3. Алгоритм преобразования дерева в двоичное: преобразует дерево в двоичное дерево через путем добавления родительских узлов, когда количество узлов на уровне превышает два

4. Алгоритм проверки доступа пользователей: предоставляет или запрещает доступ к файлу.

5. Алгоритм кеширования локального дерева доступа: позволяет пользователям кэшировать свои политики контроля доступа и синхронизировать измененные политики

Были обсуждены другие алгоритмы, основанные на различных событиях, таких как восстановление узлов, возобновление и добавление узлов.

Предлагаемая модель опубликована в статье [1].

Ограничения и требования:

Существуют некоторые ограничения и требования, которые следует учитывать при реализации модели распределения данных, алгоритмов балансировки, модели контроля доступом в DApps:

1. Количество узлов должно быть более 3 узлов. Чем больше узлов, тем выше надежность.

2. Процент использования хранилища узлов: системные узлы не могут быть использованы полностью, чтобы иметь достаточно места для восстановления данных при отказе узлов.

3. Максимальный размер файла определяется емкостью виртуального диска *2, поскольку файлы разделяются.

В главе 3 — МОДЕЛЬ ОЦЕНКИ НАДЕЖНОСТИ ПРЕДЛАГАЕМОЙ СИСТЕМЫ

Делается вывод, что существующие методы оценки надежности систем не учитывают возможность восстановления данных динамически распределять их поровну по узлам MPRC раз. MPRC зависит от используемого пространства в системе, количества узлов и количества свободного места в системе. Следовательно, нужно разработать новую модель. За основу вычисления надежности была взята известная модель расчета надежности для RAID-5.

Надёжность системы основана на вероятности отказа системы в течение года (8760 часов)[1], и рассчитывается как:

$$R = 1 - \frac{8760}{MTTF_{system}} \tag{11}$$

Также на основе предложенных алгоритмов был рассчитан MPRC:

$$MPRC = max\{n = 1..|\boldsymbol{N}| - 2: F(n) \geq 0\} + 1 \tag{12}$$

$$F(n) = F(n-1) - \frac{|\boldsymbol{VD_S}|}{|\boldsymbol{N}| - n + 1} - \frac{F(n-1)}{|\boldsymbol{N}| - n + 1} \tag{13}$$

$$F(0) = |\boldsymbol{VD`_S}| \tag{14}$$

Вероятность отказа системы рассчитывалась на основе предложенной модели и алгоритмов:

$$P_{SFANF} = \left(\frac{MTTR * MPRC}{MTTF_{node}}\right)^{MPRC} * \prod_{i=1}^{MPRC}(|\boldsymbol{N}| - i) \tag{15}$$

В результате было получено среднее время безотказной работы для предложенной модели:

$$MTTF_{system} = \left(\frac{MTTF_{node}}{MTTR * MPRC}\right)^{MPRC} * \frac{MTTF_{node}}{\prod_{i=0}^{MPRC}(|\boldsymbol{N}| - i)} \tag{16}$$

[1] Такой период используется в большинстве систем хранения, в частности для RAID.

Предполагая, что все узлы имеют одинаковое количество виртуальных дисков и свободных слотов для виртуальных дисков, надежность предлагаемой системы может быть выше надежности серверов полного резервного копирования при двух условиях:

$$R_{DecStore} > R_{fullBackup} \leftrightarrow \begin{cases} |\boldsymbol{N}| > 3 \\ \forall i \in \boldsymbol{N}: \boldsymbol{VD`}_{\boldsymbol{n(i)}} > \dfrac{2 * |\boldsymbol{VD_{n(i)}}|}{|\boldsymbol{N}| - 2} \end{cases} \quad (17)$$

Предложенная модель расчета надежности опубликована в статьях [2, 6].

В главе 4 — СРАВНИТЕЛЬНАЯ ОЦЕНКА НЕОБХОДИМЫХ РЕСУРСОВ, ПРОИЗВОДИТЕЛЬНОСТИ И НАДЕЖНОСТИ ПРЕДЛАГАЕМОЙ МОДЕЛИ РАСПРЕДЕЛЕНИЯ ДАННЫХ И КОНТРОЛЯ ДОСТУПА

Приводятся результаты оценки предложенных алгоритмов и моделей и их сравнение с другими известными подходами. Для оценки представленных алгоритмов и моделей использовались два метода: метод математического моделирования и подход, основанный на экспериментах. Далее разработанная система называется DecStore.

Производительность балансировщика нагрузки

Для проведения эксперимента был сформулирован результат базового сценария тестирования системы, состоящей из 3 узлов хранения, которая будет масштабироваться для использования в крупной финансовой организации. Целевые показатели получены из эксплуатируемой заказчиком системы с централизованной архитектурой, которую разработало ООО «РИИТ». В системе работает одновременно до 1500 пользователей и она состоит из 10 сетевых узлов.

Таблица 5 - Среднее время тестирования DecStore

Название теста	**Результат тестирования (мс)**	**Целевые показатели (мс)**
Добавление узла	974	5000
Добавление одного файла	1512	3000
Восстановление данных с потерянных узлов	65 041	600 000
Возобновление узла	1062	60 000

На основе проведенных тестов видно, что все основные операции выполняются относительно быстро по сравнению с ожидаемыми. Важно отметить, что для теста использовались виртуальные узлы, поэтому время, необходимое для передачи сетевого трафика, не учитывается. Необходимо отметить, что минимальные технические характеристики узлов

достаточно низкие (с использованием virtual box, 2 ГБ ОЗУ, 1 процессор, ОС LUbuntu), а это значит, что систему можно построить с минимальными затратами на инфраструктуру.

Процент сокращения места для распределения данных

На основе эксперимента удалось сформулировать правило, определяющее процент сокращения пространства при внедрении систему по сравнению с системами полного резервного копирования который состоит из B резервных копий:

$$Storage\ reduction\ percentage = 100 * \left(1 - \frac{\frac{3}{2}}{1+B}\right)\% \quad (18)$$

Таким образом, предлагаемая система использует на 25% меньше места для хранения по сравнению с одним сервером резервного копирования, при этом на 50% меньше места при использовании 2 резервных серверов. Рассмотрим следующий пример.

В системе полного резервного копирования, состоящей из 1 резервного сервера на каждый сервер, создается одна копия каждого объекта. Таким образом, для хранения файла размером 10 МБ потребуется 20 МБ. В предлагаемой системе файл разбивается на две половины (5 + 5) и рассчитывается стирающий код (5 МБ), таким образом, всего требуется 15 МБ по сравнению с системой полного резервного копирования с одним уровнем резервного копирования (на 25% меньше). В системе полного резервного копирования с двумя серверами резервного копирования требуется 30 МБ (что означает, что предлагаемой системе требуется на 50 % меньше).

Измерения размера данных и задержек, связанные с процессами контроля доступа

Для оценки временных характеристик DApps, реализующих предложенные модель и алгоритмы контроля доступа, был разработан ряд тестов эмуляции работы системы контроля доступа (DAC+RBAC) в разных архитектурах. Тесты были написаны на C++ на фреймворке QT, а деревья контроля доступа были представлены двумя разными способами: файлами SQLite и файлами JSON, поскольку технологии реализации влияют на результаты производительности. Тестовое окружение было развернуто под управлением Ubuntu 22.04 LTS на процессоре Core i7-8575u и 16 ГБ оперативной памяти.

В начальном эксперименте была поставлена цель оценить время создания объединенного дерева с точки зрения времени и места для хранения, а также сделать выбор технологий для его хранения и обработки. Эксперимент включал создание с нуля двух пользовательских деревьев DAC, состоящих из 500 новых файлов каждое, для создания объединенного дерева. Для создания хэшей узлов использовалась хэш-функция SHA-256. Тест проводился автоматически 100 раз. С такой единовременной нагрузкой сталкивается информационная система, где одновременно работает 1000 пользователей со средней степенью интенсивности работы с файлами (системы внешнего и внутреннего документооборота, CRM системы и т.д.) Среднее время этого процесса

составило 9,48 секунды в случае использования SQLite и 0,55 секунды в случае файлов JSON. Общий размер дерева в случае SQLite составляет 249,9 КБ, а при использовании JSON — 552 КБ. Реализация включала объединение 2 деревьев в одно, сжатие и преобразование в двоичный формат, чтобы минимизировать размер пользовательского запроса.

Чтобы дать оценку модели контроля доступа и сравнить с другими возможными реализациями, для тестирования были рассмотрены архитектуры 3-х систем:

1. Централизованная система: для ее реализации использовалась PHP-инфраструктура Laravel, поскольку она считается популярной платформой для реализации API. MySQL использовался для хранения политик доступа.
2. Blockchain Hyperledger: данные хранятся непосредственно в блокчейне, а не на внешнем ресурсе для хранения политик. Docker использовался для узлов Hyperledger.
3. DecStore: реализация модели контроля доступа, предложенной в этом исследовании.

Было создано 1000 файловых записей, а также 1000 пользователей. Всем пользователям был предоставлен доступ ко всем файлам (1000*1000). Было выполнено три тестовых случая:

1. Предоставление нового разрешения файлу: которое выражает скорость добавления нового правила доступа для пользователя x к файлу y (1x1).
2. Полный отзыв доступа пользователя: это означает отзыв всех типов доступа, предоставленных пользователю x, к любому файлу в системе (1x1000).
3. Проверка доступа: это таймер, необходимый системе для проверки наличия у пользователя x доступа к файлу y (1x1).

Результаты тестов приведены в таблице 6. По результатам тестов производительности можно сделать вывод, что скорость предлагаемой системы в различных операциях несущественно меньше по сравнению с другими системами, что не является критичным, так как в отличие от остальных архитектур DecStore поддерживает масштабируемость. Сделан вывод, что файлы JSON могут быть хорошим способом представления деревьев. Однако производительность может отличаться при реализации с использованием другого стека технологий.

Таблица 6 – Результат тестирования скорости процессов контроля доступа

Тест (измеряется в мс)	**DecStore**		**Блокчейн**	**Централизованный сервер (целевые показатели)**
	JSON	**SQLite**		
Предоставление нового разрешения файлу	157	190	100	60

Полный запрет доступа пользователя	330	330	300	60
Проверка доступа	130	140	100	10

Предлагаемая модель контроля доступа использует узлы хранения для хранения всех данных управления доступом, в то время как пользователи имеют копию своих собственных данных, связанных с доступом, что можно рассматривать как метод кэширования. Без этой техники кэширования осуществляется 4 запроса, т.к. пользователь отправляет запрос на доступ к файлу, затем запрос отправляется в блокчейн, после чего пересылается на узел хранения, который содержит информацию о доступе, после этого ответ отправляется обратно на узел блокчейна, и только после этого клиенту. При использовании кэширования необходимое количество запросов сокращается до 2 вместо 4.

Оценка надежности предлагаемой модели распределения данных

В предлагаемой модели распределения используется стирающий код для уменьшения избыточности данных. RAID-5 и RAID-6 — одни из наиболее часто используемых RAID-технологий, которые также используют стирающий код разными способами, отличными от предлагаемых модели и алгоритмов. Некоторые другие обсуждаемые технологии можно рассматривать как системы полного резервного копирования с несколькими резервными копиями с точки зрения надежности, поскольку они ориентированы на то, как узлы хранения и их репликации группируются и управляются.

С помощью математической модели расчета надежности произведено моделирование расчета надежности системы в сравнении с другими известными подходами. Для сравнения были взяты RAID-5, RAID-6 и системы с полным резервным копированием, где используется 1, 2 и 3 копии (ПРК-1, ПРК-2, ПРК-3 соответственно), что покрывает все случаи применения информационных систем, где коэффициент оперативной готовности должен быть не менее 0,99. Для расчета использовались следующие параметры емкости дисковых носителей = {2 ТВ, 3ТВ, 2ТВ, 3ТВ …} (где первый узел в системе имеет емкость 2 ТБ, второй — 3 ТБ и т. д. Узлы различаются по емкости, чтобы затруднить равномерное распределение данных), R_{rate} = 50%, Th = 30MB/s, Количество узлов варьируется от 4 до 13. Предполагается, что размер S_{input} равен 50% емкости системы. Т.к. $S_{metadata}$ и $S_{accesscontrol}$ не существенно, с точки зрения общего объема данных, то дальше ($S_{output} = S_{input} + S_{redunduncy}$). Так как результаты надежности близки к 1 (100%), то используется преобразование $-1 * \log(1 - R)$.

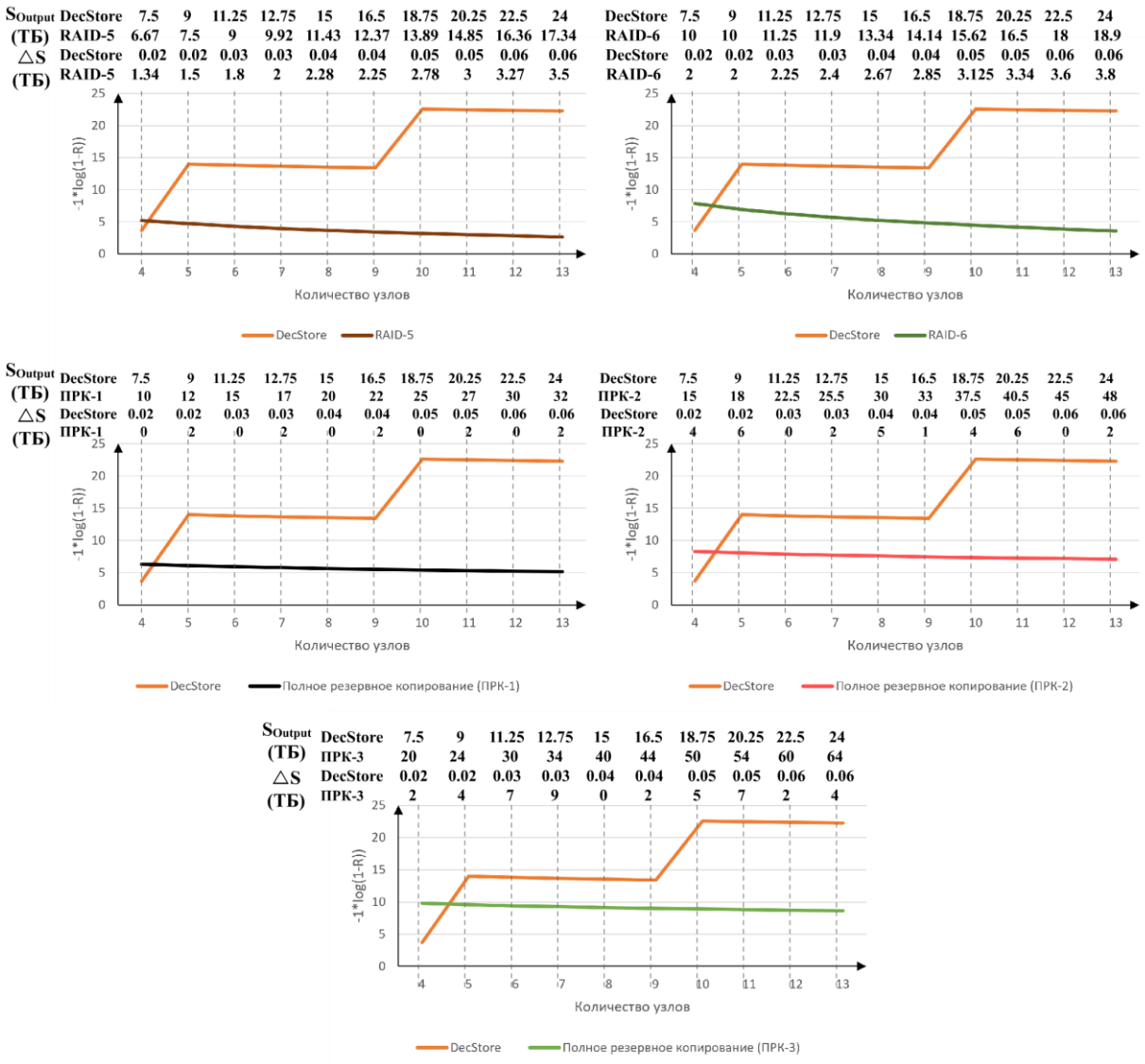

Рисунок 5 – Результаты расчета R, S_{Output}, ΔS для разных моделей распределения данных

В этих сценариях моделирования расчета надежности, рассматривается случай, когда в системе нет замены потерянным узлам. Например, в RAID-5 P_{SFANF} равен 1 после потери одного узла, поскольку потеря еще одного узла означает безвозвратную потерю данных.

В DecStore данные распределяются по узлам поровну, с небольшим ΔS, вызванным резервированием VD фиксированного размера. В RAID файлы распределяются поровну и распределяются по узлам, без учета того, что узлы с большей емкостью должны хранить больше объектов, что приводит к высокому ΔS. Системы полного резервного копирования не имеют встроенного метода распределения данных, поэтому предположим, что они также одинаково хранят файлы на всех узлах, а резервные копии дублируют их. Однако не все данные могут быть сохранены в предлагаемом сценарии. Например, в случае 2 резервных копий при наличии 4 узла $S_{input} = 50\% * (2 + 3 + 2 + 3)$ТБ $= 5$ТБ, а емкость системы 10ТБ. При этом для системы резервного копирования с двумя серверами резервного копирования требуется 15 ТБ для

хранения данных, что превышает доступное пространство. С двумя резервными копиями можно хранить только 2 ТБ, а четвертый узел остается непригодным для использования, так как у него нет резервных копий.

Учитывая общую надежность с учетом общих метаданных и контроля доступа, надежность DecStore не меняется, поскольку данные распределяются по узлам хранения и используют один и тот же метод восстановления данных (данные контроля доступа обрабатываются как другие объекты). Однако для других решений, использующих централизованный сервер для метаданных и контроля доступа, надежность снижается, поскольку зависит от единой точки отказа, и диаграмма надежности выглядит следующим образом:

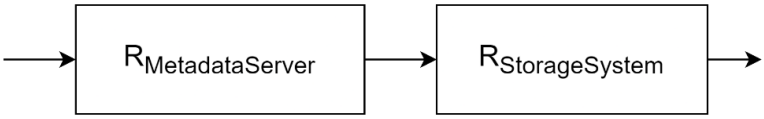

Рисунок 14 – Диаграмма надежности систем хранения данных с сервером централизованного управления

Исходя из этого, надежность этих систем можно рассчитать по следующей формуле:

$$R = R_{MetadataServer} * R_{StorageSystem} \quad (17)$$

Суммарная надежность систем $(100 * R)$ для узлов от 4 до 13 представлена в таблице 7.

Таблица 7 - Полная надежность систем с учетом контроля доступа и метаданных

	DecStore	RAID-5	RAID-6	ПРК-1	ПРК-2	ПРК-3
4	97.49	98.7	99.3	99.1	99.3	99.3
5	99.99991	98.4	99.2	99.07	99.3	99.3
6	99.9999	97.9	99.1	99.07	99.3	99.3
7	99.9998	97.4	98.9	98.9	99.3	99.3
8	99.9998	96.7	98.7	98.9	99.3	99.3
9	99.9998	96.09	98.4	98.8	99.3	99.3
10	99.99999998	95.2	98.1	98.8	99.3	99.3
11	99.99999998	94.4	97.7	98.8	99.3	99.3
12	99.99999998	93.5	97.1	98.7	99.3	99.3
13	99.99999997	92.4	96.5	98.7	99.2	99.3

Из графика и таблиц видно, что надежность DecStore повышается, когда система растет и увеличивает количество используемых серверов. В случае традиционных технологий, таких как

RAID-5 и RAID-6, с ростом количества узлов надежность становится меньше, т.к. вероятность выхода из строя 2-х или 3-х узлов одновременно увеличивается.

Результаты 4-й главы опубликованы в [1,2,3,4,5,6,7,9].

ЗАКЛЮЧЕНИЕ

В диссертационной работе решена важная научная и практическая задача внедрения DApps приложений - уменьшение избыточности циркулирующих в системе данных без потери надежности. В диссертационной работе был проведен комплексный анализ изучения методов и проектов распределения данных по набору параметров. Было установлено, что не существует научных подходов, которые решают задачу уменьшения избыточности данных без ущерба для надежности в распределенных системах. Анализ известных моделей контроля доступа в распределённых системах показал, что задача минимизации размера данных доступа, хранящийся на стороне блокчейна, на сегодняшний день не решается и для этого используются централизованные решения, что, в свою очередь, не позволяет увеличивать количество узлов и объем данных в системе.

Получены следующие научные результаты:

1) Разработана модель распределения данных в блокчейн и на узлах DApps систем, а также 4 алгоритма балансировки данных, которые оперируют этой моделью. Алгоритмы позволяют распределять данные по узлам автономно (без участия пользователя) и таким образом равномерно балансировать нагрузку. При этом балансировщик нагрузки работает в среде смарт-контрактов, что накладывает ограничения на вычислительную сложность алгоритмов. Модель распределения данных и алгоритмы балансировки позволяют уменьшить общее место хранения данных на 25% относительно систем с полным резервным копированием. Данное утверждение было подтверждено экспериментально при помощи реализации сценария поведения системы на 10 узлах и 1500 пользователей.

2) Реализация модели контроля доступа позволяет хранить только часть данных о доступе субъектов к объектам в блокчейн и таким образом минимизировать общий объем данных в реализуемой DApps системе. Новые алгоритмы создания дерева Меркла, сжатия дерева и кеширования оперируют данными этой модели и обеспечивают доступ к данным DApps в сети с нулевым доверием. Экспериментально показано, что при росте размера системы скорость обработки основных операций сравнима с централизованными системами.

3) Разработана модель надежности, которая учитывает возможности автоматического восстановления данных на других узлах системы, в то время как известные модели надежности эту возможность не рассматривают. Было проведено моделирование оценки надежности

рассматриваемой системы в сравнении с системами другой архитектуры и показано, что надежность рассматриваемой системы может превосходить надежность других систем при увеличении количества узлов и наличии достаточного свободного места на узлах для восстановления данных.

Основные положения диссертационной работы изложены в следующих опубликованных работах

В перечне, рекомендованном ВАК Минобрнауки России

1. Хаммуд О., Тарханов И. DAC модель распределенного управления доступом на основе блокчейна // Системы высокой доступности. 2024. – DOI. 10.18127/j20729472-202401-05.

2. Хаммуд О., Тарханов И.А. Анализ Надежности Распределенной Системы Хранения Файлов на Основе Блокчейн // Труды Института Системного Анализа Российской Академии Наук. 2023. – DOI. 10.14357/20790279230402.

3. Хаммуд О., Тарханов И.А. Методология Хранения Электронных Документов в Децентрализованных Блокчейн Приложениях (Dapps) // Труды Института Системного Анализа Российской Академии Наук. 2021. – DOI. 10.14357/20790279210205.

В других изданиях:

4. Hammoud O., Tarkhanov I. A method of data synchronization with Ethereum blockchain // Artificial societies. 2020. – Т. 15. – №. 3. – С. 9. – DOI. 10.18254/S207751800010671-7.

5. Hammoud O., Tarkhanov I.A. DecStore: a Blockchain-based File Storage System With Autoscaling // 2023 24th International Arab Conference on Information Technology (ACIT). Ajman, United Arab Emirates. IEEE, 2023. – С. 1–6. – DOI. 10.1109/ACIT58888.2023.10453678.

6. Hammoud O., Tarkhanov I.A. Estimating the Reliability of a DApps-Based Files Storage System // 2023 International Russian Automation Conference (RusAutoCon). 2023. – С. 82–87. – DOI. 10.1109/RusAutoCon58002.2023.10272732.

7. Hammoud O., Tarkhanov I.A. A Novel Blockchain-Integrated Distributed Data Storage Model with Built-in Load Balancing // 2022 IEEE 16th International Conference on Application of Information and Communication Technologies (AICT). 2022. – С. 1–6. – DOI. 10.1109/AICT55583.2022.10013548.

8. Hammoud O. Review of metadata storing and searching methods in DLT and distributed files systems // Law & Digital Technologies. 2022. – Т. 2. – №. 1. – С. 35. – DOI. 10.18254/S278229070018060-2.

9. Hammoud O., Tarkhanov I., Kosmarski A. An Architecture for Distributed Electronic Documents Storage in Decentralized Blockchain B2B Applications // Computers. 2021. – Т. 10. – №. 11. – С. 142. – DOI. 10.3390/computers10110142.

10. Hammoud O. Modeling the Reliability of Distributed Data Storage Systems // Artificial societies. 2024. – Т. 19. – №. 2. – С. 0. – DOI. 10.18254/S207751800031356-0.