ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
|(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)»
КАФЕДРА ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

На правах рукописи

Иваничкина Людмила Владимировна

МАТЕМАТИЧЕСКИЕ МОДЕЛИ НАДЕЖНОСТИ И МЕТОДЫ ЕЕ ПОВЫШЕНИЯ В СОВРЕМЕННЫХ РАСПРЕДЕЛЕННЫХ ОТКАЗОУСТОЙЧИВЫХ СИСТЕМАХ ХРАНЕНИЯ ДАННЫХ

Специальность 05.13.11 - Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание учётной степени
кандидата технических наук

Научный руководитель:
кандидат физико-математических наук
Коротаев Кирилл Сергеевич

Москва – 2018

Оглавление

Введение

В диссертационной работе рассматриваются математические модели надежности современных распределенных отказоустойчивых систем хранения данных. Особое внимание уделяются влиянию различных факторов на надежность хранения данных и методам ее повышения.

Актуальность темы

Надежное хранение больших объемов информации является одним из основополагающих требований современной информационной инфраструктуры. Объем информации, производимой человечеством, удваивается каждые полтора – два года, при этом все большая часть этой информации хранится не на устройствах пользователя, а в 'облачных' хранилищах. Стремительный рост вычислительной мощности компьютеров, опережающий потребности отдельных пользователей, создает предпосылки для развития облачных сервисов, а также способствует консолидации информационной инфраструктуры предприятий на базе гиперконвергентных кластерных решений, сочетающих надежное распределенное хранилище данных с возможностью запуска виртуальных машин. Подобные решения оказываются предпочтительными, поскольку обеспечивают более высокую надежность, чем выделенные сервера, за счет развитых систем резервирования и возможности миграции данных в случае отказа оборудования.

Степень разработанности темы

Построение надежного хранилища, способного не только противостоять отказам оборудования, но и обеспечивать высокую скорость доступа к данным для множества одновременно работающих с ним клиентов, является серьезной технической проблемой, не имеющей одного единственно верного решения. Всякое решение является компромиссом с точки зрения множества факторов. Успешное достижение такого компромисса возможно только на базе теоретических изысканий, имитационного моделирования и опыта практической реализации СХД. Задача осложняется еще и тем, что математических моделей, адекватно описывающих СХД, построенную на базе описанных выше принципов, практически не существовало.

Известные до настоящего времени математические модели надежности СХД основаны на представлении системы в виде цепей Маркова. Зная набор состояний системы и вероятности переходов между ними в единицу времени, можно вычислить среднее время перехода системы в поглощающее состояние, соответствующее отказу (потере данных в

нашем случае). Это время называется средним временем наработки до отказа. Мы будем использовать его в качестве количественной метрики надежности СХД. В дальнейшем для краткости изложения мы не будем делать различий между надежность и средним временем наработки до отказа (потери данных) для исследуемой системы.

Марковская модель адекватно описывает надежность RAID-массива, но совершенно не подходит для описания СХД с разбиением данных на фрагменты, поскольку они уже не могут рассматриваться, как эволюционирующие независимо друг от друга Марковские процессы. Отказ одного диска в такой системе приводит к потере сразу множества блоков данных, а их восстановление происходит не независимо, а последовательно со скоростью, которая зависит от суммарной производительности дисков, оставшихся неповрежденными.

Цель работы

Целью настоящего исследования является построение и анализ математических моделей надежности распределенного отказоустойчивого хранилища данных с разбиением данных на фрагменты и реализация такого хранилища на базе полученных в ходе исследования результатов. Особое внимание уделено методам повышения надежности СХД.

Методы исследования

Исследование начинается с изучения основных результатов, полученных в Марковской модели надежности хранения данных. Рассматривается метод моделирования стохастической системы путем сведения ее к эргодической с последующим рассмотрением ее стационарного режима. Этот метод используется для исследования разработанной нами математической модели СХД, более адекватной рассматриваемой системе, чем Марковская модель. Полученные в ходе исследования результаты верифицируются путем сравнения с результатами симуляции на моделях, имитирующих реальную СХД, а также путем сравнения со статистикой, накопленной при использовании реальной системы хранения данных, созданной нами на основе разработанных математических моделей.

Для построения основных моделей работы используются теория вероятностей, теория алгоритмов, широко используется аппарат математического анализа.

Новизна работы

1. Предложена математическая модель надежности СХД с разбиением данных на фрагменты, более адекватно описывающая реально существующие системы

хранения данных, чем Марковская модель. Разработан приближенный метод оценки надежности СХД, вытекающий из свойств модели.

2. Разработан комплекс имитационных моделей надежности СХД, отражающий реальную архитектуру и особенности реализации подобных систем.
3. Детально исследовано влияние различных факторов на надежность и масштабируемость СХД. Впервые изучено теоретически и проверено с помощью имитационных моделей влияние различных политик размещения дисковых блоков на надежность хранилища. Получены количественные оценки влияния скрытых повреждений на надежность СХД. Предложены методы борьбы со скрытыми повреждениями и даны оценки их эффективности.
4. На базе полученных теоретических результатов построена распределенная система хранения данных, гарантирующая высокую надежность за счет использования различных схем обеспечения избыточности и оптимизации надежности.

Научная и практическая ценность работы

Разработанные нами методы построения математических моделей реальных СХД зарекомендовали себя удобным инструментом анализа надежности подобных систем, значительно более адекватным реальной архитектуре хранилища, чем применявшиеся ранее Марковские модели. Применение этих методов позволило нам обосновать уже известные из практики способы повышения надежности и масштабируемости СХД, а также получить новые, ранее неизвестные результаты.

Построенная нами на базе полученных теоретических результатов система хранения данных Acronis Storage показала свою высокую надежность и коммерческую эффективность при хранении данных корпоративных клиентов в десятках стран мира. Созданная нами СХД позволяет объединить до 10,000 дисков в единое хранилище, обеспечивающее высокую надежность, автоматическое восстановление при отказах оборудования и производительность, сравнимую с производительностью локального дискового накопителя при одновременной работе тысяч клиентов. В настоящее время в различных инсталляциях Acronis Storage хранится более 100 петабайт пользовательских данных по всему миру. За более чем 5 лет эксплуатации системы не было ни одного случая потери данных вследствие отказа оборудования, что демонстрирует эффективность технических решений, созданных на базе полученных в ходе исследования теоретических результатов.

Положения, выносимые на защиту

1. Математическая модель надежности СХД с разбиением данных на фрагменты, более адекватно описывающая реально существующие системы хранения данных, чем Марковская модель.
2. Комплекс имитационных моделей надежности СХД, отражающий реальную архитектуру и особенности реализации подобных систем.
3. Получены теоретически и проверены на моделях оценки влияния различных факторов на надежность и масштабируемость СХД, а именно влияния различных политик размещения дисковых блоков и скрытых повреждений.
4. Практическая реализация распределенной системы хранения данных, гарантирующей высокую надежность за счет использования различных схем обеспечения избыточности и оптимизации надежности.

Апробация результатов работы

Результаты диссертационного исследования и побочные результаты докладывались, обсуждались и получили одобрение специалистов на российских и международных научных конференциях:

- IX Международная научно-практическая конференция "Актуальные проблемы науки XXI века", г.Москва, 2016;
- Международная научно-практическая конференция «Приоритетные задачи и стратегии развития естественных и математических наук», г.Тольятти, 2016;
- IV Международная научно-практическая конференция «Наука XXI века: проблемы и перспективы», г.Уфа, 2016;
- VIII Международная научно-практическая конференция «Инновационные технологии нового тысячелетия», г.Киров, 2016;
- Конференция «Облачные вычисления. Образование. Исследования. Разработка», г.Москва, 2015;
- 58-я научная конференция МФТИ, г.Долгопрудный, 2015.

Результаты работы реализованы в виде программного комплекса по оценке надёжности СХД с помощью приближенно-аналитических моделей, а также вычислительных имитационных методов статистических испытаний. Исходный код доступен в публичном репозитории [46-47].

Представленные в диссертации, разработанные нами методы и программные средства повышения надёжности и высокой доступности СХД включены в реализуемые ООО «Проект ИКС» технологии и программное обеспечение распределенных и высокопроизводительных вычислительных систем для хранения и обработки больших данных.

Диссертационная работа была выполнена рамках проведения прикладных научных исследований, выполняемых по Соглашению с Министерством науки и образования о предоставлении субсидии №14.579.21.0010 от "05"июня 2014 года по теме "Технология и программное обеспечение распределенных и высокопроизводительных вычислительных систем. Хранение и обработка больших данных".

Публикации по теме дисертации

По материалам данного диссертационного исследования были опубликованы работы [1-4,10-15]. В списке изданий присутствуют рекомендованные ВАК ([4]), индексируемые системами WebOfScience и SCOPUS ([10-14]), два свидетельства о регистрации программы ([2,3]) и заявка на патент [15]. В работах [2-4,10,11,13], выполненных в соавторстве, автору принадлежит материал, связанный с постановкой задачи, проведениями исследований и формулировкой математических моделей. В работе [12] вклад автора заключался в разработке имитационной модели, проектировании и реализации вычислительных алгоритмов, проведении вычислений и анализе результатов. В работе [14] вклад автора заключался в разработке и реализации алгоритма распределения дисковых блоков с учетом областей отказов, проектировании и реализации имитационной модели для оценки надежности хранилища, проведении вычислений и анализе полученных результатов.

Структура и объём диссертации

Диссертация состоит из введения, четырех глав, заключения и списка использованных источников. Общий объем дисертации 130 страниц, она включает 12 таблиц и 34 рисунка. Библиография включает 47 наименований.

Личный вклад автора

Все представленные в диссертации результаты получены лично автором.

1. Современные системы хранения данных и проблема моделирования их надежности

Предъявляемые к современным системам хранения данных (СХД) требования выходят далеко за пределы возможностей единичного дискового накопителя как по объему хранимой информации, так и по надежности и производительности. Перечислим наиболее важные из них:

1. *Масштабируемость*. Емкость хранилища должна легко наращиваться путем добавления в систему новых дисков и серверов. При этом не должно происходить ухудшения параметров производительности и надежности при увеличении размеров системы (до определенного предела).
2. *Отказоустойчивость*. Система должна сохранять работоспособность при выходе из строя отдельных компонент, поскольку с ростом числа компонент вероятность таких событий растет, и они перестают быть исключительными.
3. *Надежность*. Данные, записанные клиентом не должны меняться самопроизвольно или в результате отказов компонент системы. Если данные недоступны, клиент должен либо ожидать их доступности, либо получить соответствующее ситуации сообщение об ошибке.
4. *Способность к самовосстановлению*. После отказа компоненты (например, диска) система должна автоматически (без вмешательства оператора) провести необходимые процедуры восстановления целостности данных, чтобы отказ еще одной компоненты не привел к катастрофической потере данных.
5. *Универсальность*. Система должна предоставлять возможность доступа к данным с помощью различных протоколов доступа – файлового уровня (POSIX, NFS), блочного уровня (iSCSI, SMB/CIFS) или специализированных протоколов доступа уровня объектного хранилища (Amazon S3).

Методы построения отказоустойчивых СХД на базе нескольких дисковых накопителей зародились примерно 3 десятилетия назад. Это были так называемые RAID-массивы – ряд дисков, объединенных в единое хранилище [18, 19]. Повышение надежности такого хранилища достигалось за счет хранения избыточной информации (контрольных сумм) на одном или нескольких дополнительных дисках. При отказе одного из дисков данные можно было восстановить, используя эту избыточную информацию. Подобные решения справлялись со своими задачами пока необходимое число дисков в хранилище не превышало десятка. Проблема, возникающая при дальнейшем увеличении числа дисков, связана с тем, что вероятность отказа одного из множества дисков в единицу времени прямо

пропорциональна их количеству. Если к этому моменту не завершится восстановление после предыдущего отказа, данные могут быть потеряны.

Получение приемлемой надежности в хранилище данных, объединяющем тысячи дисков, требовало принципиально иных технических решений. Таким решением стало разбиение сохраненных данных на фрагменты, где каждый фрагмент хранится в виде набора дисковых блоков на различных дисках в СХД с необходимой для обеспечения надежности избыточностью. Увеличение надежности такого хранилища по сравнению с RAID-массивом достигается за счет того, что в случае отказа одного из дисков, поврежденные фрагменты оказываются распределены по всем дискам в такой системе. Соответственно, восстановление избыточности после потери диска также происходит одновременно на всех дисках в системе. Значит, время восстановления уменьшается с ростом числа дисков, эффективно компенсируя увеличение вероятности отказа любого из дисков с ростом их количества.

Первой промышленной системой, построенной на этом принципе, была кластерная файловая система корпорации Google [28], предназначенная для внутреннего использования специализированными приложениями корпорации Google. Аналогичные идеи легли в основу СХД Azure, разработанной корпорацией Microsoft [43], а также системы с открытым исходным кодом CEPH [36].

В системе, состоящей из сотен серверов и тысяч дисков, отказы оборудования и дисков в частности являются вполне рядовыми событиями. Соответственно, программное обеспечение такой системы должно обеспечить автоматическое восстановление после отказа оборудования, чтобы свести к минимуму вероятность потери данных. В связи с этим построение адекватной математической модели надежности является решающим фактором для создания отказоустойчивой и масштабируемой системы хранения данных.

Модели надежности дисковых RAID-массивов строились на базе цепей Маркова [42], где рассматривается эволюция дискового массива, как последовательность переходов между состояниями, различающимися количеством поврежденных дисков. При этом мы можем считать, что вероятность перехода из одного состояния в другое в единицу времени никак не зависит от предыстории. Это допущение и позволяет нам рассматривать эволюцию системы, как цепь Маркова, что существенно упрощает задачу оценки надежности хранения данных. Как будет показано в главе 2, знание вероятностей перехода между состояниями в единицу времени позволяет нам легко найти среднее время до потери данных в аналитическом виде.

В современных СХД данные делятся на фрагменты, которые сохраняются в виде дисковых блоков, распределенных по дискам системы в соответствии с некоторым

алгоритмом размещения. При этом надежность хранения достигается путем сохранения дополнительных блоков контрольных сумм, обеспечивающих избыточность. Надежность хранения каждого такого фрагмента может быть адекватно описана в рамках той же модели Марковской цепи. Однако при этом остается нерешенным вопрос о том, каким образом зная среднее время до потери конкретного фрагмента, вычислить среднее время до потери любого из множества фрагментов в системе. Например, в работе [44] делается предположение, что это время можно приближенно определить разделив среднее время до потери фрагмента на общее число вариантов размещения дисковых блоков, принадлежащих различным фрагментам. При этом полученные результаты могут только качественно отражать надежность реальной системы с неизвестной степенью достоверности. Авторы работы [45] по тем же причинам отказались от рассмотрения среднего времени до потери данных, как показателя надежности системы, и ограничились рассмотрением вероятности потери данных при восстановлении из критического состояния, т.е. такого состояния, когда вследствие дисковых отказов часть фрагментов данных имеет нулевую избыточность.

Цель данной диссертационной работы – восполнить указанный пробел в теории надежности СХД, построив математическую модель, способную давать количественный оценки среднего времени до потери данных. В отличие от существующих моделей, описывающих эволюцию отдельного фрагмента данных, с последующей экстраполяцией результатов на всю систему, созданная нами математическая модель описывает эволюцию системы, как целого. Это позволило нам получить целый ряд новых аналитических оценок для надежности и масштабируемости СХД, а также детально рассмотреть влияние размещения дисковых блоков по дискам на результирующую надежность системы. Наконец, мы рассмотрели влияние скрытых повреждений на надежность системы и предложили ряд способов борьбы с ними.

2. Математические модели надежности систем хранения данных

Данная глава посвящена рассмотрению математических моделей надежности СХД. Она начинается с описания классической Марковской модели надежности. Далее мы обсудим степень адекватности классической модели реальным системам хранения данных и перейдем к рассмотрению разработанной автором математической модели надежности СХД более полно описывающей реальные системы хранения данных. Для проверки адекватности предложенной математической модели мы сравним ее предсказания с результатами имитационного моделирования с использованием разработанного нами комплекса моделей, отражающего реальную архитектору и особенности реализации СХД.

2.1. Отказоустойчивые схемы хранения данных

Современные системы хранения данных состоят из многих компонент, каждая из которых подвержена отказам. Для того, чтобы обеспечить устойчивость системы хранения данных к отказам отдельных компонент, применяют резервирование и вносят избыточность в схему хранения данных, так, чтобы их можно было восстановить даже в случае утраты части хранимых данных. Простейший подход к обеспечению отказоустойчивости заключается в хранении каждого блока данных x в n копиях. В случае утраты одной из них мы сразу же восстанавливаем утраченную копию копированием одной из оставшихся. Эта схема называется репликацией, она схематически представлена на рисунке ниже.

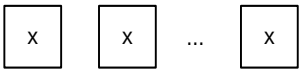

Рисунок 1. Хранение данных в репликах

Недостаток репликации – в низкой эффективности использования дискового пространства, поскольку данные в итоге занимают в n раз больше места.

Второй распространённый способ повышения надежности хранения данных заключается в добавлении блока контрольной суммы. Пусть мы разбили данные на k блоков $x_1, x_2, .. x_k$ и добавили к ним блок контрольной суммы, вычисленный по формуле $p = x_1 + x_2 + .. + x_k$. Тогда при потере любого блока, мы можем решить это уравнение относительно утраченного блока и восстановить его.

Рисунок 2. Хранение данных с контрольной суммой

На практике использование обычной арифметики неудобно, поскольку для сохранения результата сложения может понадобиться больше битов, чем требовалось для

хранения слагаемых. Поэтому для вычисления контрольной суммы используется некоторая операция над конечным множеством целых чисел (например, 8-битных), которая обладает теми же свойствами, что и сложение. Простейший вариант, который и используется на практике – операция исключающего ИЛИ. Подобная схема позволяет восстановить утраченные данные при потере одного блока, но повреждение большего числа блоков приводит к потере данных.

Чтобы иметь возможность восстановить данные при потере более одного блока, дополним цепочку блоков одной или несколькими дополнительными контрольными суммами. Пример такой цепочки для двух контрольных сумм показан на рисунке ниже.

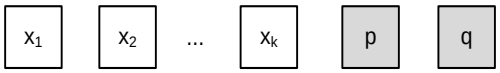

Рисунок 3. Хранение данных с несколькими контрольными суммами

Пусть p по прежнему вычисляется как сумма всех информационных блоков данных, а q – как взвешенная сумма с некоторыми коэффициентами:

$$p = x_1 + x_2 + .. + x_k$$

$$q = \alpha_1 x_1 + \alpha_2 x_2 + .. + \alpha_k x_k$$

При потере двух блоков мы сможем решить эту систему относительно утраченных блоков (при подходящем выборе коэффициентов α_i). Для этого нам понадобится обратимая операция умножения на конечном множестве целых чисел. Простейший пример такой операции – умножение по модулю простого числа – неудобен на практике, поскольку в двоичной системе счисления мы имеем дело с множествами с числом элементов равным степени ее основания, то есть числа 2. Поэтому на практике используются поля Галуа [5, с.186], предоставляющие операции сложения и умножения над множеством с числом элементов, равным 2^q, где q – произвольное целое число.

Абстрагируясь от конкретного алгоритма формирования избыточных блоков, все рассмотренные выше схемы хранения данных можно охарактеризовать двумя числами: n – общее количество блоков в цепочке (кортеже) и k – количество уникальных информационных блоков данных. При этом гарантируется восстановление целостности данных при потере до $(n-k)$ блоков. Схема хранения данных, обладающая этими свойствами часто обозначается просто, как (n,k) схема. Разность $n-k$ мы будем называть избыточностью и обозначать буквой m. Часто (n,k) схему обозначают как $k+m$, подчеркивая тот факт, что она образована путем добавления m блоков контрольных сумм к k информационным блокам.

В следующих главах мы рассмотрим методы расчета надежности хранения данных в схемах с избыточностью, используя различные допущения по поводу процессов потери блоков данных и их восстановления.

2.2. Расчет надежности хранения данных в Марковской модели

Один из способов аналитического решения задачи определения надежности хранилища данных заключается в представлении системы в виде цепи Маркова – случайного процесса, происходящего на конечном множестве состояний, для которого вероятность попасть в некоторое состояние зависит только от занимаемого перед этим состояния [6]. Это определение означает, что будущая эволюция Марковской цепи зависит только от ее состояния в текущий момент времени и никак не зависит от предыстории – от того, как система попала в текущее состояние. Свойство независимости от предыстории существенно упрощает рассуждения, поэтому Марковские цепи повсеместно используются для моделирования надежности сложных систем [7]. В дальнейшем такие модели мы будем называть просто Марковскими.

Рассмотрим для примера простейший вариант избыточного хранения данных – хранение блока данных в 2 экземплярах (репликах).

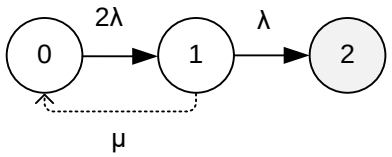

Рисунок 4. Граф состояний Марковской цепи для двух реплик

У такой системы всего 3 состояния. В состоянии 0 она полностью исправна. В состоянии 1 повреждена одна реплика, в состоянии 2 – две реплики. Из состояния 1 возможен переход в состояние 0, поскольку мы можем восстановить потерянную реплику путем копирования оставшейся. В состоянии 2 данные необратимо потеряны, поскольку у нас не осталось реплик для восстановления. Такие состояния мы будем называть *поглощающими (терминальным)*. Подобная система рано или поздно попадет в одно из поглощающих состояний и останется в нем навсегда. Наша задача – вычислить среднее время перехода в поглощающее состояние системы, которая в начальный момент времени находится в неповрежденном состоянии. Для краткости переход в поглощающее состояние мы будем называть просто *отказом*.

Ниже мы приводим расчет среднего времени до перехода в состояние отказа для Марковской модели. Затем мы обсудим, насколько такая модель адекватна реальным системам хранения данных (СХД) и рассмотрим альтернативные подходы к моделированию надежности СХД.

2.2.1. Аналитическое решение задачи о среднем времени до отказа

Эволюция Марковской модели однозначно определяется вероятностями перехода между состояниями. В случае дискретной эволюции системы используется матрица вероятностей переходов $\boldsymbol{P}$, элемент P_{ij} которой равен вероятности перехода из состояния i в состояние j [6, с.93]. Эволюция реальной системы обычно непрерывна во времени. Для описания такой системы используются Марковские цепи с непрерывным временем. Для них имеет смысл говорить о матрице вероятностей перехода за некоторый интервал времени τ: $\boldsymbol{P}(\tau)$. При этом $\boldsymbol{P}(0) = \boldsymbol{I}$, где $\boldsymbol{I}$ – единичная матрица, что иллюстрирует тот факт, что за нулевое время система имеет нулевую вероятность перейти в любое состояние, отличное от текущего. Поскольку $\boldsymbol{P}(\tau)$ зависит от τ, более удобной для рассмотрения оказывается ее производная по параметру τ, которую называют матрицей интенсивностей переходов:

$$\boldsymbol{Q} = \frac{d\boldsymbol{P}(\tau)}{d\tau}\Big|_{\tau=0} = \lim_{\tau\to 0}\frac{\boldsymbol{P}(\tau) - I}{\tau}$$

Соответственно, для бесконечно малого dt:

$$P_{ij}(dt) = dt * Q_{ij}\ (i \neq j),\ P_{ii}(dt) = 1 + dt * Q_{ii} \quad (2.2.1)$$

Отсюда понятен смысл элементов матрицы $\boldsymbol{Q}$. Недиагональные элементы Q_{ij} равны вероятности перехода из состояния i в состояние j *за единицу времени*. Диагональные элементы Q_{ii} равны вероятности ухода из состояния i в любое другое (со знаком минус) *за единицу времени*. Соответственно, сумма любой строки матрицы $\boldsymbol{Q}$ равна нулю. Это свойство также следует из того, что сумма любой строки матрицы $\boldsymbol{P}$ равна единице, как вероятность перехода из начального состояния в *любое другое*, включая то же состояние. Соответственно, сумма элементов строки матрицы $\boldsymbol{Q}$ равняется сумме производных по времени элементов матрицы $\boldsymbol{P}$ или производной от их суммы, то есть нулю, как производная от постоянной величины. В нашем примере системы из 2-х реплик с 3 состояниями матрица интенсивностей переходов имеет вид:

$$Q = \begin{bmatrix} -2\lambda & 2\lambda & 0 \\ \mu & -(\lambda+\mu) & \lambda \\ 0 & 0 & 0 \end{bmatrix}$$

Здесь параметры λ и μ – это величины, обратные среднему времени наработки до потери реплики и среднему времени ее восстановления соответственно. Множитель 2 для перехода из состояния 0 в состояние 1 иллюстрирует тот факт, что вероятность отказа в единицу времени любой из 2-х реплик в 2 раза больше вероятности отказа каждой из них.

Пусть $p_i(t)$ – вероятность обнаружить систему в состоянии i в момент времени t. Вероятность $p_i(t+dt)$ для момента времени $t+dt$ можно найти, разложив p_i в сумму условных вероятностей следующим образом:

$$p_i(t+dt) = \sum_j p_j(t) P_{ji}(\tau)$$

Подставив сюда (2.2.1), получим:

$$p_i(t+dt) = p_i(t) + dt * \sum_j p_j(t) Q_{ji}$$

Отсюда получаем уравнение Колмогорова [4, с.124]:

$$\frac{d}{dt} p_i = \sum_j p_j(t) Q_{ji} \qquad (2.2.2)$$

Решив систему дифференциальных уравнений (2.2.2), мы можем получить зависимость вероятности обнаружить систему в состоянии i от времени для произвольной Марковской цепи и найти среднее время перехода в состояние отказа – см. например [17, с.328-333].

Если все, что нас интересует – это среднее время наработки до отказа, то решать эту систему нет необходимости. Есть простой способ свести задачу к системе линейных уравнений. Пусть мы наблюдаем эволюцию системы начиная с состояния 0. После того, как она попадает в поглощающее состояние, мы сразу же переводим ее обратно в состояние 0 и продолжаем наблюдать за ее эволюцией. Если за время T система k раз попала в поглощающее состояние, то среднее время наработки до отказа (Mean Time To Failure - MTTF) может быть вычислено по формуле:

$$T_F = T/k$$

Если мы наблюдаем эволюцию ансамбля из N независимых систем, то

$$T_F = TN/k \qquad (2.2.3)$$

Нас будет интересовать эта величина в пределе $k \to \infty$ (и соответственно $T \to \infty$). При этом предполагается, что мы поддерживаем ансамбль в квазистационарном состоянии, постоянно переводя системы, попавшие в поглощающее состояние, обратно в состояние 0. Рассуждая таким образом, легко получить следующий метод вычисления среднего времени наработки до отказа[1].

Пусть $\widetilde{\boldsymbol{Q}}$ – матрица интенсивностей переходов, в которой удалены строки и столбцы, соответствующие поглощающим состояниям, а $\boldsymbol{t}$ – результат решения системы линейных уравнений $\boldsymbol{t} * \widetilde{\boldsymbol{Q}} = [-1, 0, \ldots 0]$. Тогда среднее время наработки до отказа равно:

[1] Без доказательства он приведен в [16, с.130]. Доказательство, несколько отличающееся от приведенного здесь, можно найти в монографии [3838, с. 507].

$$T_F = \sum_i t_i$$

Докажем это утверждение. Пусть мы наблюдаем за ансамблем из N независимых систем, где в состоянии i находится n_i экземпляров нашей системы. Без ограничения общности мы можем считать N произвольным и сколь угодно большим. По определению $\tilde{\boldsymbol{Q}}$ в единицу времени $n_i * \tilde{Q}_{ij}$ экземпляров системы переходят из состояния i в состояние j. Эту величину мы будем называть *потоком* из состояния i в состояние j. Будем считать, что эволюция ансамбля достаточно далека от начального состояния, так что система достигла квазистационарного состояния, когда n_i не зависят от времени. Значит, потоки между состояниями плюс поток извне F_j в сумме равны нулю для каждого состояния j:

$$\sum_i n_i * \tilde{Q}_{ij} + F_j = 0 \qquad (2.2.4)$$

Единственный источник внешнего потока – это перевод экземпляров системы из поглощающих состояний в состояние 0, то есть $F_0 = f, F_j = 0\ (j \neq 0)$. Здесь f – суммарный поток в поглощающие состояния. Как видно из (2.2.3), он просто равен $f = \frac{k}{T} = \frac{N}{T_F}$, откуда

$$T_F = \frac{N}{f} \qquad (2.2.5)$$

где N – общее число экземпляров системы $N = \sum_i n_i$. Перепишем (2.2.4) в виде

$$\sum_i n_i * \tilde{Q}_{ij} = [-f, 0, 0, \dots 0] \qquad (2.2.6)$$

Мы можем решить это уравнение, получив квазистационарное распределение экземпляров системы по состояниям, для любого f, они будут отличаться лишь общим количеством экземпляров. Пусть t_i – такое решение для $f = 1$. Тогда из (2.2.5) сразу следует, что $T_F = \sum_i t_i$.

В силу линейности (2.2.6) выбор конкретного значения f никак не влияет на полученный ответ. Действительно, для произвольного f решение (2.2.6) можно выразить через решение для $f = 1$:

$$n_i = f * t_i$$

Отсюда $N = f * \sum_i t_i$ и снова $T_F = \frac{N}{f} = \sum_i t_i$. Значит, эта формула будет оставаться справедливой для произвольного N. Можно дополнительно показать, что t_i – это среднее время, которое система проводит в состоянии i прежде, чем попасть в поглощающее состояние (см. [38]).

Фактически суть нашего метода нахождения T_F сводится к тому, что мы преобразовали Марковскую цепь с поглощающими состояниями к *эргодической*, т.е. такой, для которой вероятность попасть из любого состояния в любое другое отлична от нуля. При этом T_F будет равно величине, обратной средней по времени вероятности попадания в

поглощающее состояние исходной системы. Это среднее по времени для эргодической системы равно среднему по ансамблю, которое легко вычислить, рассматривая стационарное состояние системы.

На практике полезно иметь формулу, выражающую T_F через матрицу интенсивностей $\widetilde{\boldsymbol{Q}}$ в явном виде. Пусть $\widetilde{\boldsymbol{Q}}$ – матрица интенсивностей переходов, в которой удалены строки и столбцы, соответствующие поглощающим состояниям, а $\boldsymbol{Q}^1$ – матрица, полученная из нее заменой первого столбца на единичный столбец. Тогда среднее время наработки до отказа равно:

$$T_F = -\frac{det(\boldsymbol{Q}^1)}{det(\widetilde{\boldsymbol{Q}})} \qquad (2.2.7)$$

Докажем это утверждение. Обозначим $\mathbf{J} = [1, 0, \dots 0]$, $\mathbf{I} = \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$. Тогда $T_F = \boldsymbol{t} * \boldsymbol{I}$, где $\boldsymbol{t}$ – решение системы уравнений $\boldsymbol{t} * \widetilde{\boldsymbol{Q}} = -\boldsymbol{J}$. Пусть $\boldsymbol{r}$ – решение системы уравнений $\widetilde{\boldsymbol{Q}} * \boldsymbol{r} = \boldsymbol{I}$ Тогда $T_F = \boldsymbol{t} * \boldsymbol{I} = -\boldsymbol{J} * \widetilde{\boldsymbol{Q}}^{-1} * \widetilde{\boldsymbol{Q}} * \boldsymbol{r} = -\boldsymbol{J} * \boldsymbol{r} = -r_0$. Выразив r_0 по формуле Крамера, получаем формулу (2.2.7).

Применив полученные формулы к нашему примеру системы с двумя репликами, легко получить следующие результаты:

$$t_0 = \frac{\lambda+\mu}{2\lambda^2} \qquad t_1 = \frac{1}{\lambda} \qquad T_F^{(2)} = \frac{3\lambda+\mu}{2\lambda^2}$$

Что полностью согласуется с решением [17, с.334], полученным с помощью преобразования Лапласа уравнений Колмогорова (2.2.2). Для наглядности полезно переписать полученный результат в следующем виде:

$$T_F^{(2)} = \frac{1}{\lambda} * \frac{3+\mu/\lambda}{2} \qquad (2.2.8)$$

Здесь $1/\lambda$ – это среднее время наработки до отказа одной реплики, а отношение μ/λ равно отношению времени наработки до отказа реплики к времени ее восстановления. Таким образом, выигрыш в надежности за счет репликации оказывается тем больше, чем быстрее мы способны восстановить утраченную реплику.

Результаты применения формулы (2.2.7) для (n, k) схемы в зависимости от избыточности $m = n - k$ приведены в следующей таблице.

Таблица 1. Среднее время наработки до отказа в Марковской модели для различных (n, k) схем хранения

схема хранения	*среднее время наработки до отказа*
(n, n)	$T_F^{(n,n)} = \frac{1}{\lambda * n}$

$(n, n-1)$	$T_F^{(n,n-1)} = \frac{1}{\lambda * n * (n-1)} * \left(2n - 1 + \frac{\mu}{\lambda}\right)$
$(n, n-2)$	$T_F^{(n,n-2)} = \frac{1}{\lambda * n * (n-1) * (n-2)} * \left(3n^2 - 6n + 2 + (2n-2)\frac{\mu}{\lambda} + \frac{\mu^2}{\lambda^2}\right)$
$(n, n-3)$	$T_F^{(n,n-3)} = \frac{1}{\lambda * n * (n-1) * (n-2) * (n-3)} * \left(4n^3 - 18n^2 + 22n - 6 + (3n^2 - 9n + 6)\frac{\mu}{\lambda} + (2n-3)\frac{\mu^2}{\lambda^2} + \frac{\mu^3}{\lambda^3}\right)$
$(n, n-4)$	$T_F^{(n,n-4)} = \frac{1}{\lambda * n * (n-1) * (n-2) * (n-3) * (n-4)} * \left(5n^4 - 40n^3 + 105n^2 - 100n + 24 + (4n^3 - 24n^2 + 44n - 24)\frac{\mu}{\lambda} + (3n^2 - 12n + 12)\frac{\mu^2}{\lambda^2} + (2n-4)\frac{\mu^3}{\lambda^3} + \frac{\mu^4}{\lambda^4}\right)$

В предельном случае $\lambda/\mu \to 0$, то есть когда скорость потери блоков данных много меньше скорости их восстновления,

$$T_F^{(n,n-m)} \approx \frac{1/\lambda}{n*(n-1)*\ldots*(n-m)} * \frac{\mu^m}{\lambda^m} \qquad (2.2.9)$$

То есть среднее время до потери данных оказывается пропорциональным среднему времени до потери блока $1/\lambda$ с коэффициентом, пропорциональным отношению скорости восстановления μ к скорости потери блоков λ в степени, равной избыточности m. В ссследующей главе мы рассмотрим более простой способ получения этого результата для произвольной (n, k) схемы, не опирающийся на точное решение (2.2.7).

Приближенная формула (2.2.9) часто встречается в литературе, посвященной надежности СХД без ссылок на точный результат (2.2.7), например в [18, с.111], [19, с.158]. Использование формулы (2.2.9), когда время восстановления велико, например, как в случае RAID массива, когда восстанавливается диск целиком, может приводить к существенному искажению оценок надежности системы. А в частном случае отсутствия восстановления $(\mu = 0)$ формула (2.2.9) дает совершенно абсурдный результат $T_F = 0$.

Чтобы проверить справедливость нашего подхода и в частном случае $\mu = 0$, решим задачу нахождения времени наработки до отказа воспользовавшись альтернативным, чисто вероятностным методом. Начнем с вычисления времени наработки до отказа единственного блока данных. Пусть $R^{(1)}(t)$ – вероятность обнаружить неповрежденный блок данных в момент времени t (функция надежности) [17, с.22]. При этом $R^{(1)}(0) = 1$. Вероятность обнаружить неповрежденный блок данных в момент времени $t + dt$ для бесконечно малого dt можно вычислить по формуле:

$$R^{(1)}(t + dt) = R^{(1)}(t) * (1 - \lambda dt)$$

Здесь λdt – вероятность отказа на временном интервале dt при интенсивности отказов в единицу времени, равной λ. Соответственно $1 - \lambda dt$ - вероятность того, что блок останется неповрежденным к концу временного интервала dt при условии, что он был неповрежден в его начале (отсюда множитель $R^{(1)}(t)$). Таким образом, получаем дифференциальное уравнение:

$$\frac{d}{dt}R^{(1)}(t) = -\lambda R^{(1)}(t)$$

С учетом начального условия $R^{(1)}(0) = 1$, получаем решение:

$$R^{(1)}(t) = e^{-\lambda t}$$

Пусть τ – случайная величина, равная времени наработки до отказа исследуемой системы. Вероятность того, что $t < \tau < t + dt$ есть вероятность отказа на временном интервале $t \ldots t + dt$, а она в свою очередь равна $R(t) - R(t + dt)$ в силу определения функции надежности $R(t)$. Значит, τ имеет плотность распределения $\rho(t)$, равную отношению этой вероятности к величине интервала dt, то есть равную $-R'_t$. Отсюда легко найти среднее значение τ, которое является средним временем наработки до отказа:

$$T_F = \langle\tau\rangle = \int_0^\infty \rho(t) * t dt = -\int_0^\infty R'_t * t dt = -\int_0^\infty t * dR = -[tR]_0^\infty + \int_0^\infty R dt$$

Считая, что $\lim_{t\to\infty} tR = 0$, получаем, что среднее время до отказа равно площади под графиком функции надежности:

$$T_F = \int_0^\infty R dt \qquad (2.2.10)$$

В случае единственного блока, получаем:

$$T_F^{(1)} = \int_0^\infty e^{-\lambda t} dt = \left[-\frac{e^{-\lambda t}}{\lambda}\right]_0^\infty = \frac{1}{\lambda}$$

Мы доказали, что среднее время наработки до отказа блока равно величине, обратной количеству отказов в единицу времени. Рассмотрим более сложный случай с двумя блоками-репликами. Потеря данных в этом случае произойдет только в случае отказа обоих

блоков. Учитывая, что вероятность отказа блока (функция отказа) равна $F(t) = 1 - R(t)$, для 2-х реплик получаем:

$$F^{(2)}(t) = 1 - R^{(2)}(t) = \left[F^{(1)}(t)\right]^2 = \left[1 - R^{(1)}(t)\right]^2$$

Отсюда:

$$R^{(2)}(t) = 1 - \left[1 - e^{-\lambda t}\right]^2 = 2e^{-\lambda t} - e^{-2\lambda t}$$

Снова применяя (2.2.10), получаем:

$$T_F^{(2)} = \frac{2}{\lambda} - \frac{1}{2\lambda} = \frac{3}{2\lambda}$$

что полностью соответствует результату (2.2.8) при $\mu = 0$.

Таким образом, формула (2.2.7) дает нам общее решение задачи нахождения среднего времени наработки до отказа в Марковской модели с произвольным набором состояний. Однако, есть 2 обстоятельства, вследствие которых эта формула редко применяется на практике. Во-первых, точное аналитическое решение часто оказывается слишком громоздким, как видно из Таблица 1. Во-вторых, что гораздо более важно, Марковская модель по большей части лишь качественно описывает реальные хранилища данных. В следующих разделах мы рассмотрим метод решения задачи нахождения среднего времени наработки до отказа в приближении малости интенсивности отказов по сравнению со скоростью восстановления ($\lambda \ll \mu$), а также перейдем к альтернативным способам моделирования хранилища данных более адекватно описывающим реальные системы хранения данных.

2.2.2. Асимптотическое решение в приближении малости интенсивности отказов

Количество членов в аналитическом выражении для времени наработки до отказа может расти экспоненциально с увеличением количества состояний системы, поскольку определители в числителе и знаменателе (2.2.7) содержат по $K!$ членов, где K – число непоглощающих состояний. Это обстоятельство делает получение аналитического результата практически бессмысленным.

В то же время, точный аналитический ответ часто является избыточным, поскольку

- вероятность отказа различных компонент обычно известна нам лишь по порядку величины
- представление о постоянстве вероятности отказов, а также о процессе восстановления, как имеющем независящую от времени интенсивность переходов, являются грубым приближением

На практике гораздо полезнее иметь простое и наглядное выражение, верное в *асимптотическом приближении*, в предположении, что интенсивность отказов мала по сравнению с интенсивностью процессов восстановления (точное определение будет дано ниже). Нами была разработана оригинальная методика получения такого асимптотического приближения в аналитическом виде для произвольного графа состояний системы.

Рассмотрим направленный граф состояний системы. Вершинами этого графа будут состояния, а ребрами – переходы между состояниями, вызванные отказами или восстановлением. Соответственно, ребра графа состояний мы будем называть *ребрами отказов* или *ребрами восстановления* в зависимости от того, чем вызван соответствующий переход между состояниями. С каждым ребром, связывающим состояние i с состоянием j, мы свяжем *интенсивность перехода*, равную вероятности перехода $i \to j$ в единицу времени при условии, что система находится в состоянии i. Интенсивности отказов будем, как и прежде обозначать буквой λ, а восстановления – буквой μ.

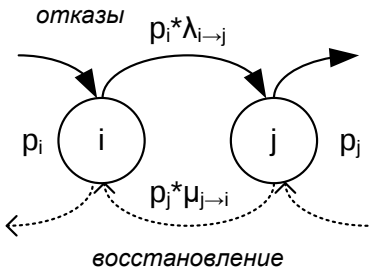

Рисунок 5. Часть графа состояний системы с переходами отказов (сплошные линии) и восстановления (пунктирные линии).

Как и в 2.2.1, будем рассматривать квазистационарный ансамбль из некоторого произвольного количества N статистически независимых экземпляров нашей системы. Пусть p_i – количество экземпляров системы в состоянии i или *заселенность* этого состояния. Количество переходов из состояния i в состояние j в единицу времени мы будем называть *потоком* из состояния i в состояние j. Он равен произведению заселенности состояния i на соответствующую интенсивность перехода, то есть $p_i * \lambda_{i\to j}$ для переходов отказов и $p_i * \mu_{i\to j}$ для переходов восстановления.

Рассмотрим способ построения асимптотического решения на нашем простом примере с 2 репликами (Рисунок 4). Выберем p_0 произвольно. Тогда p_1 мы сможем найти из уравнения $2\lambda * p_0 - (\mu + \lambda) * p_1 = 0$. Отбрасывая асимптотически малые члены, получаем $p_1 = \frac{2\lambda}{\mu} p_0$. Поток в поглощающее состояние (2): $f = \lambda * p_1 = \frac{2\lambda^2}{\mu} p_0$. Отсюда, следуя (2.2.5), получаем $T_F = \frac{p_0 + p_1}{f} = \frac{\mu}{2\lambda^2}$, где мы учли, что p_1 асимптотически мало по сравнению с p_0. В нашем случае последнее утверждение следует из полученного выше результата $p_1/p_0 = \frac{2\lambda}{\mu}$. В общем случае можно показать, что все p_i $(i \neq 0)$ асимптотически малы по сравнению с p_0. Для этого найдем верхнюю и нижнюю границы для p_i.

Введем следующие обозначения. Пусть λ_{min} - минимальная интенсивность отказов по всем непоглощающим состояниям системы, λ_{max} - максимальная интенсивность отказов по всем непоглощающим состояниям системы с учетом кратности переходов, то есть если из состояния есть несколько переходов отказов, то в λ_{max} мы учитываем сумму их интенсивностей. Аналогично, пусть μ_{min}- минимальная интенсивность восстановления по всем непоглощающим состояниям системы (кроме состояния 0), μ_{max}- максимальная интенсивность восстановления по всем непоглощающим состояниям системы (кроме состояния 0) с учетом кратности переходов. Будем считать, что отношения $\lambda_{max}/\lambda_{min}$ и μ_{max}/μ_{min} ограничены по величине некоторой произвольной константой:

$$\frac{\lambda_{max}}{\lambda_{min}} \leq C, \quad \frac{\mu_{max}}{\mu_{min}} \leq C \qquad (2.2.11)$$

Пусть $\gamma = \lambda_{max}/\mu_{min}$. Мы будем говорить, что А асимптотически мало по сравнению с В, если $\lim_{\gamma\to 0} A/B = 0$. Будем говорить, что А асимптотически равно В, если $\lim_{\gamma\to 0} A/B = 1$. Наша задача - найти метод построения решения асимптотически равного аналитическому решению (2.2.7).

Назовем *уровнем деградации* состояния системы длину кратчайшего пути в него из состояния 0 по ребрам отказов в графе состояний. Уровень деградации 0 состоит из единственного состояния 0. Остальные уровни деградации могут содержать и больше

одного состояния, как показано на рисунке ниже, где ребра отказов показаны сплошной линией, а ребра восстановления – пунктиром. На рисунке показаны только 2 уровня деградации, остальные уровни, включая поглощающие состояния, для простоты опущены.

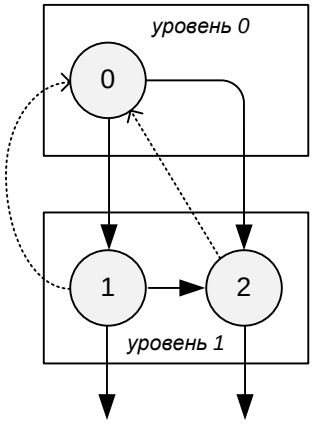

Рисунок 6. Состояния системы, сгруппированные по уровням деградации.

Предположим, что наша система обладает следующими свойствами:

1. В каждое состояние, кроме состояния 0, ведет хотя бы один переход отказа[2].
2. Переходы, соответствующие восстановлению, никогда не приводят к увеличению уровня деградации[3].
3. Из каждого состояния есть как минимум один переход восстановления, который приводит к уменьшению уровня деградации. Только такие переходы мы будем учитывать в μ_{min}. При этом всегда $\mu_{min} > 0$, так что выполняется (2.2.11).

Заметим, что эти свойства выполняются автоматически, если в каждом состоянии система представляет некоторый набор скрытых компонент, которые могут находиться как в исправном, так и в неисправном состоянии. Тогда уровень деградации – это просто количество неисправных компонент. Свойство (1) означает, что в состояние, где какие-то компоненты неисправны, можно попасть из состояния, где хотя бы один из неисправных компонент был исправен. Свойство (3) означает, что из состояния, где какие-то компоненты неисправны можно попасть в состояние, где хотя бы один их них исправен.

Сгруппируем состояния системы по уровням деградации и рассмотрим переходы в некоторый уровень $k > 0$. Пусть D_k – множество состояний уровня деградации k. Обозначим $P_k = \sum_{i \in D_k} p_i$ – суммарная заселенность уровня деградации k.

[2] Это необходимое и достаточное условие для того, чтобы каждому состоянию можно было присвоить уровень деградации.

[3] По сути это требование исключительно терминологическое, поскольку если такой переход восстановления все таки найдется, мы сможем назвать его переходом деградации.

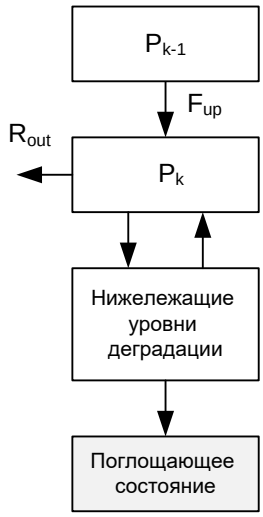

Рисунок 7. Баланс потоков для уровня деградации k.

Заселенность уровня деградации $k-1$ обозначим как P_{k-1}. В соответствии с рисунком назовем его вышележащим, а уровни деградации $> k$ назовем нижележащими. В силу свойства конструктивного восстановления единственными переходами с вышележащего уровня деградации являются переходы отказов. Их суммарный поток $F_{up} \leq \lambda_{max} P_{k-1}$. Переходы восстановления образуют поток $R_{out} \geq \mu_{min} P_k$. Суммарный поток в нижележащие уровни в целом положителен, что следует из нулевого баланса потоков нижележащих уровней и из того, что из них существует поток отказов в поглощающее состояние и, возможно, поток восстановления в уровни выше k-го. Таким образом, уравнение баланса потока уровня k можно записать в виде: $F_{up} - R_{out} > 0$. Отсюда:

$$\lambda_{max} P_{k-1} \geq F_{up} > R_{out} \geq \mu_{min} P_k$$

$$\lambda_{max} P_{k-1} > \mu_{min} P_k$$

$$P_k < \frac{\lambda_{max}}{\mu_{min}} P_{k-1}$$

$$P_k < \gamma P_{k-1}$$

Применяя рекурсивно эту формулу, получаем, что $P_k < \gamma^k P_0$. Но $P_0 = p_0$, так как нулевой уровень деградации состоит из единственного состояния 0, а $p_i \leq P_k (i \in D_k)$. Значит

$$p_i < \gamma^k p_0 \ (i \in D_k) \qquad (2.2.12)$$

Чтобы найти нижнюю границу заселенности произвольного состояния, рассмотрим баланс потока для некоторого состояния i $(i \in D_k)$.

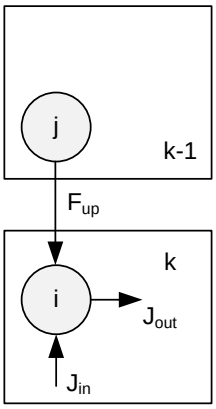

Рисунок 8. Баланс потоков для состояния i на уровне деградации k.

Заметим, что на уровне $k-1$ найдется состояние j ($j \in D_{k-1}$), которое связано с состоянием i ребром отказа. Это ребро дает поток $F_{up} \geq \lambda_{min} P_{k-1}^{min}$, где мы ввели обозначение P_k^{min} – минимальная заселенность состояния на уровне k. Исходящий поток из состояния i: $J_{out} \leq (\mu_{max} + \lambda_{max}) p_i$. Условие баланса потока для состояния i можно записать в виде: $F_{up} + J_{in} - J_{out} = 0$, где $J_{in} \geq 0$ соответствует входящим потокам от переходов отказов и восстановления, оставшихся неучтенными. Таким образом:

$$\lambda_{min} P_{k-1}^{min} \leq F_{up} \leq J_{out} \leq (\mu_{max} + \lambda_{max}) p_i$$

$$\lambda_{min} P_{k-1}^{min} \leq (\mu_{max} + \lambda_{max}) p_i$$

$$p_i \geq \frac{\lambda_{min}}{\mu_{max} + \lambda_{max}} P_{k-1}^{min}$$

Иначе говоря

$$P_k^{min} = \frac{\lambda_{min}}{\mu_{max} + \lambda_{max}} P_{k-1}^{min} \geq \frac{\gamma}{\mathrm{C}^2 + \gamma \mathrm{C}} P_{k-1}^{min}$$

где мы учли неравенства (2.2.11). Считая, что $\gamma < 1$, воспользуемся неравенством $\mathrm{C}^2 + \gamma \mathrm{C} < \mathrm{C}^2 + \mathrm{C} < (C+1)^2$ и перепишем для простоты этот результат в виде

$$P_k^{min} > \frac{\gamma}{(C+1)^2} P_{k-1}^{min}$$

Рекурсивно применяя эту формулу, получаем, что

$$p_i > \frac{\gamma^k}{(C+1)^{2k}} p_0 \quad (i \in D_k) \qquad (2.2.13)$$

Объединяя обе полученные оценки (2.2.12) и (2.2.13), приходим к неравенству:

$$\frac{\gamma^k}{(C+1)^{2k}} p_0 < p_i < \gamma^k p_0 \; (i \in D_k) \qquad (2.2.14)$$

Таким образом, мы получили верхний и нижний пределы заселенности p_i произвольного состояния i. Отсюда непосредственно следует, что:

1. все p_i $(i \neq 0)$ асимптотически малы по сравнению с p_0

2. любой p_i асимптотически мал по сравнению с p_j, если состояние j имеет уровнень деградации меньший, чем состояние i
3. никакой p_i не является асимптотически малым по сравнению с p_j, если состояния i и j имеют одинаковый уровнень деградации

Полученные результаты позволяют найти асимптотическое решение в случаях, более сложных, чем наш пример с 2 репликами. Рассмотрим систему с 3 репликами, граф состояний которой показан на рисунке ниже. Как и раньше, состояние с индексом 0 соответствует системе без потерянных блоков данных. В состоянии 1 потеряна одна реплика, в состоянии 2 – две. В состоянии 3 потеряны все реплики, восстановление данных невозможно, так что это состояние является поглощающим.

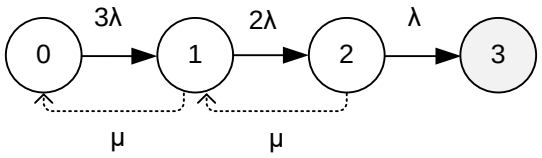

Рисунок 9. Граф состояний для 3-х реплик.

Мы не можем, начав с p_0 последовательно вычислить p_1 и p_2, поскольку p_1 через входящий поток восстановления зависит не только от p_0, но и от p_2. Иначе говоря, граф переходов содержит цикл, не содержащий состояние 0. Мы собираемся доказать, что можно отбросить переходы, вклад которых в заселенность состояний является пренебрежимо малым, и получить граф без таких циклов (Рисунок 10).

Лемма 1

Вклад перехода с потоком J в заселенность уровня не превосходит $|J|/\mu_{min}$, если он не является исходящим переходом восстановления.

Действительно, баланс потоков для состояния i можно записать в виде $J_{in} - \sum_j \mu_{ij} p_i - F_{out} = 0$, где J_{in} – все входящие потоки, суммирование производится по всем исходящим переходам восстановления $i \to j$, а F_{out} – исходящие потоки отказов. Значит $p_i = (J_{in} - F_{out})/\sum_j \mu_{ij}$. Если из числителя убрать некоторый поток J, то результат изменится на $\Delta p_i = |J|/\sum_j \mu_{ij} \le |J|/\mu_{min}$.

Лемма 2

Вклад исходящего потока отказов в заселенность состояния асимптотически мал.

Действительно, с учетом Леммы 1 в данном случае $\frac{\Delta p_i}{p_i} \le \frac{\lambda_{max}}{\mu_{min}} = \gamma$.

Лемма 3

Вклад входящего потока отказов в заселенность состояния на уровне k асимптотически мал, если уровень деградации его источника больше или равен k.

Действительно, с учетом (2.2.14) и Леммы 1 в данном случае $\frac{\Delta p_i}{p_i} \leq \frac{\lambda_{max}}{\mu_{min}}(C+1)^{2k} = \gamma(C+1)^{2k}$.

Лемма 4

Вклад входящего потока восстановления в заселенность состояния на уровне k асимптотически мал, если уровень деградации его источника больше k.

Действительно, с учетом (2.2.14) и Леммы 1 в данном случае $\frac{\Delta p_i}{p_i} \leq \frac{\mu_{max}}{\mu_{min}}\gamma(C+1)^{2k} \leq \gamma C(C+1)^{2k}$.

Фактически эта лемма означает, что потоки восстановления с нижележащих уровней можно перенаправить в состояние 0, полученный граф состояний будет асимптотически эквивалентен исходному. Для нашего примера с 3 репликами такой эквивалентный граф показан на рисунке ниже.

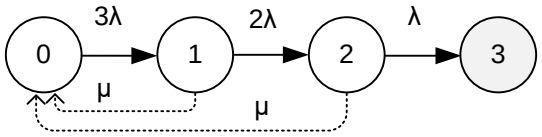

Рисунок 10. Граф состояний для 3-х реплик, где восстановление всегда происходит в состояние 0.

Процесс вычисления асимптотического решения в данном случае выглядит следующим образом:

1. Выберем p_0 произвольно
2. Пренебрегая исходящим потоком отказов получаем $p_1 = \frac{3\lambda}{\mu}p_0$
3. Пренебрегая исходящим потоком отказов получаем $p_2 = \frac{2\lambda}{\mu}p_1 = \frac{6\lambda^2}{\mu^2}p_0$
4. Находим поток переходов в поглощающее состояние $f = \lambda p_2 = \frac{6\lambda^3}{\mu^2}p_0$
5. Находим среднее время наработки до отказа $T_F = \frac{p_0}{f} = \frac{\mu^2}{6\lambda^3}$

Легко проверить, что результат асимптотически равен аналитически точному результату (2.2.7), причем независимо от того, на какой уровень происходит восстановление с уровня k, – на уровень 0 или $k-1$.

В общем случае наш алгоритм сводится к следующим шагам:

1. Обходя граф состояний по ребрам отказов присваиваем каждому состоянию уровень деградации, равный длине кратчайшего пути в это состояние из состояния 0.
2. Отбрасываем ребра, вносящие асимптотический малый вклад в заселенность состояний в соответствии с леммами 3 и 4. Получаем граф без циклов (за исключением циклов, проходящих через состояние 0).

3. Производим топологическую сортировку состояний в соответствии с полученным на шаге (2) графом. Получается список состояний, такой что, для любого перехода $i \to j$, оставшегося после шага (2), состояние i всегда находится в списке перед состоянием j.
4. Произвольно выбираем заселенность состояния 0[4].
5. Пользуясь леммой 2, последовательно вычисляем заселенность остальных непоглощающих состояний в порядке их следования в списке, полученном на шаге (3).
6. Находим суммарный поток переходов f в поглощающие состояния.
7. Находим среднее время наработки до отказа по формуле $T_F = \frac{p_0}{f}$

[4] Состояние 0 располагается первым в списке, полученном на шаге (3), поскольку это единственное состояние без входящих переходов.

2.2.3. Анализ надежности (n,k) схемы в асимптотическом приближении

Применим описанную методику к общему случаю (n,k) схемы хранения данных.

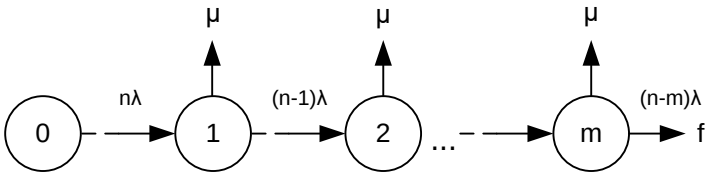

Рисунок 11. К расчету надежности (n,k) схемы в асимптотическом приближении.

Пусть p_i – заселенность состояния i. Выберем p_0 произвольно. Для состояния $i \neq 0$, мы можем пренебречь исходящим потоком отказов и входящим потоком восстановления и рассмотреть баланс входящего потока отказов и исходящего потока восстановления, как показано на Рисунок 11. Конечное состояние переходов восстановления намеренно не показано на рисунке, поскольку вклад этого перехода в заселенность конечного состояния пренебрежимо мал. Это может либо состояние $i-1$, тогда граф состояний принимает вид, как на Рисунок 9, либо состояние 0, тогда получается граф состояний подобный показанному на Рисунок 10. При этом в приближении $\lambda \ll \mu$ результирующее среднее время наработки до отказа получается одинаковым.

Таким образом, рассмотрев баланс потока отказов и восстановления для состояния i, получаем соотношение:

$$(n-i+1)\lambda p_{i-1} = \mu p_i$$

Отсюда $p_i = (n-i+1)\frac{\lambda}{\mu}p_{i-1}$ или, применяя этот результат рекурсивно:

$$p_i = \frac{\lambda^i}{\mu^i} n(n-1)\dots(n-i+1)p_0 = \frac{\lambda^i}{\mu^i}\frac{n!}{(n-i)!}p_0 \qquad (2.2.15)$$

Поток переходов f в поглощающее состояние определяется заселенностью состояния $m = n-k$:

$$f = (n-m)\lambda p_m = \frac{\lambda^{m+1}}{\mu^m}\frac{(n-m)*n!}{(n-m)!}p_0 = \frac{\lambda^{m+1}}{\mu^m}\frac{n!}{(n-m-1)!}p_0$$

Среднее время наработки до отказа найдем по формуле $T_F = \frac{p_0}{f}$:

$$T_F = \frac{\mu^m}{\lambda^{m+1}}\frac{(n-m-1)!}{n!} = \frac{\mu^{n-k}}{\lambda^{n-k+1}}\frac{(k-1)!}{n!}$$

что в точности соответствует результатам из Таблица 1 в пределе $\lambda/\mu \to 0$.

Пусть $T_1 = 1/\lambda$ – среднее время наработки до отказа одного блока, иными словами – время до потери данных при хранении без избыточности, $T_R = 1/\mu$ – среднее время восстановления поврежденного блока. Тогда полученную формулу для наглядности можно переписать в виде:

$$T_F = T_1 \frac{(k-1)!}{n!}\left[\frac{T_1}{T_R}\right]^{n-k} \qquad (2.2.16)$$

Иначе говоря, надежность хранения при использовании (n,k) схемы по сравнению с хранением без избыточности возрастает в количество раз, пропорциональное отношению $\frac{T_1}{T_R} = \frac{\mu}{\lambda}$ в степени, равной избыточности $n-k$.

Заметим, что коэффициент $(k-1)!/n!$ в формуле (2.2.16) уменьшается при увеличении n примерно как n в степени, равной избыточности плюс 1. То есть, увеличивая количество информационных блоков при постоянной избыточности, мы увеличиваем эффективность хранения данных, но уменьшаем надежность. Максимальная надежность (т.е. максимальное T_F) достигается для схемы репликации, где $k = 1$. На рисунке ниже показана зависимость надежности хранения данных с избыточностью 2 от накладных расходов на их хранение - количества дополнительных данных в процентах к исходному размеру данных. За единицу на шкале надежности принята надежность хранения в 3-х репликах.

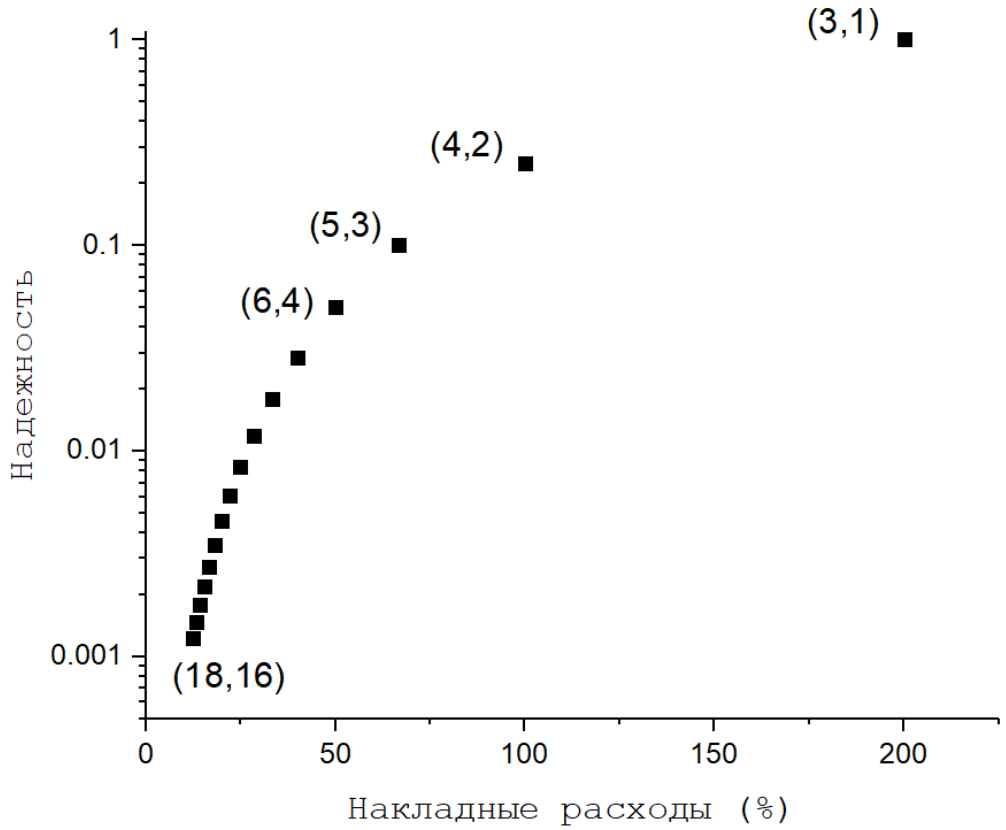

Рисунок 12. Взаимосвязь надежности и накладных расходов для различных схем хранения данных с избыточностью 2.

Как видно из рисунка, при увеличении количества информационных блоков уменьшение накладных расходов сопровождается значительным падением надежности, которое может достигать нескольких порядков.

Мы рассмотрели метод приближенного решения задачи нахождения среднего времени наработки до отказа в приближении малости интенсивности отказов по сравнению со скоростью восстановления ($\lambda \ll \mu$). В следующей главе с помощью имитационного моделирования мы изучим, насколько Марковская модель адекватна реальной системе хранения данных, а также рассмотрим альтернативную модель точнее описывающую реальные хранилища данных.

2.3. Моделирование реального распределенного многодискового хранилища данных

2.3.1. Особенности реального многодискового хранилища

Реальное хранилище данных состоит из множества дисков, и это обстоятельство существенно меняет расчет надежности такого хранилища. Основной причиной потери блоков в такой системе становится отказ целого диска, приводящий к одновременной потере **всех хранящихся на этом диске блоков**. Соответственно, представление Марковской модели об отказах блоков, как происходящих независимо и с постоянной вероятностью, становится неверным. Неверным становится и представление о восстановлении, как о вероятностном процессе, происходящим с постоянной скоростью. В реальности восстановление блоков после отказа диска происходит не одновременно, а последовательно, причем скорость этого процесса оказывается ограничена производительностью диска. С другой стороны, обычно восстановление происходит одновременно на всех оставшихся дисках в системе, как показано на рисунке ниже.

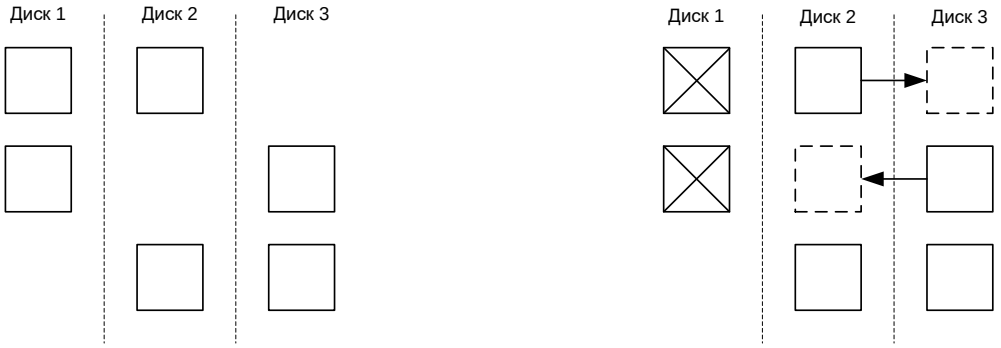

Рисунок 13. Восстановление после потери диска в многодисковом хранилище.

Данные каждого хранящегося в системе файла разбиваются на один или более диапазонов (чанков), размер которых ограничен некоторой величиной порядка от десятков мегабайт до нескольких гигабайт. Каждый чанк хранится в виде кортежа из дисковых блоков в соответствии с выбранной схемой хранения. Разбиение файла на чанки позволяет с одной стороны ограничить время, необходимое для восстановления утраченного блока данных, с другой стороны – равномерно распределить блоки по дискам. На рисунке показан пример распределения кортежей из 2-х блоков по 3 дискам. Каждый горизонтальный ряд соответствует кортежу. Мы никогда не храним более одного блока из кортежа на одном диске, чтобы исключить их одновременную потерю при отказе диска. Как видно из рисунка, преимущество такого равномерного распределения в том, что восстановление поврежденных кортежей при отказе диска может происходить одновременно на всех оставшихся в системе дисках. Как мы увидим в дальнейшем, такая одновременность

восстановления приводит к тому, что надежность системы растет с увеличением числа дисков, хотя вероятность отказа любого из дисков увеличивается.

Таким образом, непосредственное использование результатов, полученных в Марковской модели для одного чанка, в реальном многодисковом хранилище оказывается невозможным. Чтобы проверить, насколько результаты Марковской модели могут быть применимы к реальному хранилищу, мы создали несколько имитационных моделей такого хранилища.

2.3.2. Имитационные модели реальной распределенной системы хранения данных (СХД)

Мы разработали 2 имитационные модели – простую, работающую в асимптотическом приближении большого числа дисков и чанков, и более сложную, учитывающую конкретное распределение чанков по дискам. Каждая модель предполагает, что в системе имеется N дисков, данные хранятся в C чанках, каждый из которых представлен кортежем из n блоков. Данные хранятся в схеме (n, k), то есть чанк считается потерянным, если в нем утрачено более $n - k$ блоков. Предполагается, что отказы дисков в системе происходят с независящей от времени вероятностью λ, то есть $\lambda = 1/T_1$, где T_1 – среднее время до отказа любого из имеющихся в системе дисков или, иными словами, среднее время до потери данных при хранении их в единственной реплике. Восстановление утраченных блоков происходит одновременно на всех работоспособных дисках в системе, при этом чанки на каждом диске восстанавливаются последовательно, причем один чанк восстанавливается за время $T_R = 1/\mu$. Как правило, время восстановления чанка T_R определяется скоростью доступа к диску. Результатом работы каждой модели является среднее время до потери данных T_F. При этом потеря данных фиксируется, когда хотя бы в одном чанке остается меньше, чем k неповрежденных блоков.

Простая (асимптотическая) модель, назовем ее модель А, хранит число чанков p_i на каждом уровне деградации, индекс которого i равен числу потерянных блоков в чанке. Отказ диска приводит к тому, что с уровня i на уровень $i + 1$ переходит $\frac{n-i}{N} p_i$ чанков. Здесь $(n - i)p_i$ – число блоков, принадлежащих чанкам p_i. Мы считаем что они равномерно распределены по N дискам (пренебрегая вышедшими из строя). Потеря данных происходит, если в результате оказывается, что $p_{n-k+1} > 0$. Восстановление с уровня i на уровень $i - 1$ происходит с постоянной скоростью, равной $\frac{N/(k+1)}{T_R} = \mu N/(k + 1)$. Здесь $N/(k + 1)$ – количество чанков, которые восстанавливаются одновременно, с учетом того, что для восстановления достаточно k блоков, плюс еще один диск будет вовлечен в процесс восстановления, поскольку на нем мы воссоздаем утраченный блок. Как и в реальной системе, в нашей модели первыми восстанавливаются чанки с максимального уровня деградации.

Более сложная модель, назовем ее модель В, хранит набор блоков для каждого чанка и планирует восстановление таким образом, чтобы каждый диск в каждый момент времени участвовал в восстановлении не более одного чанка. Опять же это соответствует ограничениям реальных СХД.

На рисунке ниже представлены результаты моделирования для системы из 2500 чанков на 50 дисках со схемой хранения (5,3). Каждая точка на графике была получена при усреднении результатов 200 симуляций. Сплошные точки соответствуют модели B, полые – модели A. В модели A надежность хранилища оказывается несколько выше, поскольку мы не учитываем влияние конечности числа дисков и чанков на скорость репликации. В частности, в модели B число одновременно восстанавливаемых чанков должно быть целым числом, поэтому некоторые диски оказываются не вовлечены в процесс восстановления. Модель A этих ограничений не учитывает. График построен в безразмерных координатах с тем, чтобы подтвердить зависимость T_F от T_1 и T_R из формулы (2.2.16). Как видно из графика, в выбранных координатах зависимость действительно оказывается линейной, так что $\sqrt{\frac{T_F}{T_1}} = \alpha \frac{T_1}{T_R}$, то есть $T_F = T_1 * \alpha^2 \left[\frac{T_1}{T_R}\right]^2$, что с точностью до коэффициента соответствует формуле (2.2.16).

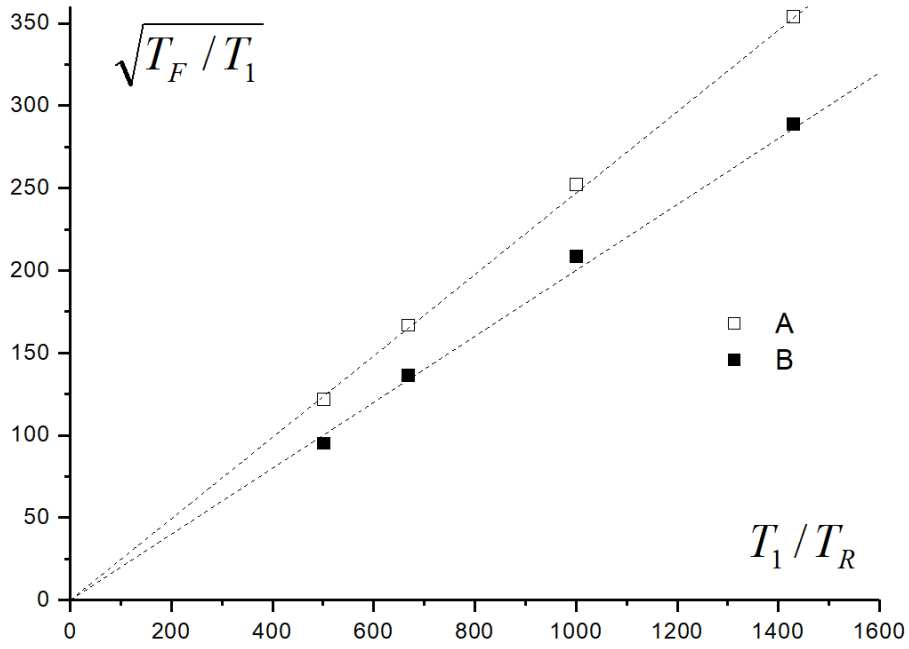

Рисунок 14. Зависимость среднего времени до потери данных T_F от среднего времени до отказа диска T_1 в линеаризованных координатах.

Таким образом, Марковская модель надежности, построенная для одного чанка, качественно соответствует результатам моделирования надежности всей системы, хотя она никак не учитывает одновременность потери блоков данных на вышедшем из строя диске, а также последовательное восстановление утраченных блоков со скоростью, которая определяется скоростью доступа к диску. Теоретически мы могли бы построить Марковскую модель всего хранилища, учитывающую состояния каждого блока данных в нем, но количество состояний в такой модели было бы настолько большим, что получение

с ее помощью каких-либо результатов стало бы практически невозможным. В следующей главе мы рассмотрим альтернативную модель поведения реального хранилища.

2.3.3. Математическая модель надежности реальной СХД

Поскольку описанная выше Марковская модель дает лишь качественное соответствие реальной системе, мы разработали математическую модель, позволяющую получать приближенные оценки среднего времени до отказа в реальном многодисковом хранилище.

Пусть P_i – усредненное по времени количество чанков на уровне деградации i, $\lambda = 1/T_1$ – суммарная интенсивность дисковых отказов, $\mu = 1/T_R$ – скорость восстановления, $\varphi = \frac{N}{k+1}\mu$ – суммарная по N дискам скорость восстановления в предположении, что чанков всегда достаточно, чтобы восстановление происходило на всех дисках одновременно. При этом в восстановлении одного чанка участвуют k блоков плюс один вновь создаваемый блок. Поскольку мы никогда не располагаем более одного блока из одного кортежа на одном диске, в процессе восстановления чанка участвует $k+1$ диск. Соответственно, одновременно могут восстанавливаться $N/(k+1)$ чанков.

Чтобы вычислить скорость восстановления на уровне деградации i, введем еще одну характеристику F_i – вероятность обнаружить ненулевое количество чанков на уровне деградации i. Тогда поток восстановления с уровня деградации i на уровень $i-1$ просто равен φF_i, поскольку восстановление идет с постоянной скоростью φ пока на уровне деградации i есть поврежденные чанки. Поток отказов с уровня i на уровень $i+1$ можно вычислить по формуле $J_i = P_i \frac{n-i}{N}\lambda$. Здесь $P_i(n-i)$ – суммарное количество неповрежденных блоков на уровне i, соответственно $P_i(n-i)/N$ – среднее количество таких блоков на одном диске, а λ – вероятность выхода диска из строя в единицу времени.

Чтобы упростить расчеты, мы предположим, что P_i и F_i не зависят от времени, то есть система находится в состоянии динамического равновесия. Если считать, что в начальный момент времени у нас нет поврежденных блоков, то это условие очевидно не выполняется, но можно ожидать, что спустя некоторое время количество поврежденных блоков приблизится к равновесному. Рассмотрение равновесной конфигурации системы интересно и с практической точки зрения, если нас интересует не среднее время до *первого* отказа после начала работы полностью исправной системы, а среднее время до отказа относительно произвольного момента времени. При этом потеря данных возможно уже происходила в прошлом нашей системы, нас лишь интересует среднее время до потери данных в ее будущем. Заметим, что разница между первым отказом и произвольным отказом отсутствовала в ранее рассмотренной Марковской модели чанка постольку, поскольку после первого отказа чанк попадал в поглощающее состояние и его эволюция прекращалась. В системе, состоящей из множества чанков, мы можем продолжать

рассмотрение ее эволюции и после потери данных одного или нескольких чанков. Поскольку точка отсчета времени до очередного отказа может быть выбрана произвольно, среднее время до следующего отказа равно среднему времени между последовательными отказами. Для поддержания системы в состоянии динамического равновесия мы будем восполнять уход чанков в поглощающее состояние за счет добавления в систему такого же количества неповрежденных чанков.

В состоянии динамического равновесия поток отказов с вышележащего уровня уравновешивается потоком восстановления, то есть $\varphi F_i = J_{i-1} = P_{i-1}\frac{n-i+1}{N}\lambda$, откуда

$$F_i = P_{i-1}\frac{n-i+1}{N}\frac{\lambda}{\varphi} \qquad (2.3.1)$$

При этом мы пренебрегаем потоком восстановления с нижележащего уровня, считая его пренебрежимо малым.

Вероятность потери данных в единицу времени p_F может быть найдена из вероятности обнаружить ненулевое количество чанков на уровне деградации $m = n - k$ следующим образом:

$$p_F = F_m * \lambda \qquad (2.3.2)$$

Эта формула является следствием того, что для потери данных необходимо и достаточно, чтобы в момент возникновения дисковой ошибки на уровне деградации m оставалось ненулевое количество чанков. При этом мы считаем, что их всегда достаточное количество, чтобы обнаружить хотя бы один на любом диске. Искомое время наработки до отказа T_F – это величина, обратная p_F.

Для решения задачи нам не хватает уравнения, связывающего P_i на соседних уровнях деградации. Пусть $p_i(t)$ – количество чанков на уровне деградации i в момент времени t, тогда P_i равно среднему по времени значению p_i: $P_i = \langle p_i(t)\rangle_t$. Заметим, что чанки на уровне i могут появиться только вследствие отказа диска в некоторый момент времени t', их начальное количество равно $p_i(t') = \frac{n-i+1}{N}p_{i-1}(t')$. После этого $p_i(t)$ линейно уменьшается до нуля за время $\tau = p_i(t')/\varphi$. При этом мы пренебрегаем вероятностью отказа в течение интервала времени τ, считая, что $F_i \ll 1$. Усредним $p_i(t)$ за период времени T:

$$P_i = \langle p_i(t)\rangle_t = \frac{1}{T}\int_0^T p_i(t)dt$$

Вклад одного дискового отказа в этот интеграл равен $\int_{t'}^{t'+\tau} p_i(t)dt = \frac{\tau}{2}p_i(t') = \frac{1}{2\varphi}\left[\frac{n-i+1}{N}p_{i-1}(t')\right]^2$, а среднее число таких дисковых отказов за время T равно λT. Отсюда:

$$P_i = \langle p_i(t) \rangle_t = \frac{\lambda}{2\varphi} \left[\frac{n-i+1}{N} \right]^2 \langle p_{i-1}(t)^2 \rangle_t \qquad (2.3.3)$$

Таким образом, наша задача свелась к нахождению связи между средним значением p_i и его средним квадратом. Рассматривать p_i как случайную величину неудобно, поскольку ее плотность вероятности не является непрерывной, - с вероятностью $1 - F_i$ величина p_i равна 0. Рассмотрим непрерывно распределенную случайную величину $\tilde{p}_i$, которая определена в случае если $p_i > 0$ и равна p_i. По определению F_i: $\langle p_i \rangle = F_i \langle \tilde{p}_i \rangle$, $\langle p_i^2 \rangle = F_i \langle \tilde{p}_i^2 \rangle$

Воспользуемся соотношением $\langle \tilde{p}_i^2 \rangle = \langle \tilde{p}_i \rangle^2 + D[\tilde{p}_i]$, где $D[\tilde{p}_i]$ – дисперсия $\tilde{p}_i$. Перепишем его в виде: $\langle \tilde{p}_i^2 \rangle = \alpha_i \langle \tilde{p}_i \rangle^2$, где $\alpha_i = 1 + \frac{D[\tilde{p}_i]}{\langle \tilde{p}_i \rangle^2}$. Тогда $\langle p_i^2 \rangle = F_i \langle \tilde{p}_i^2 \rangle = F_i \alpha_i \langle \tilde{p}_i \rangle^2 = \frac{\alpha_i}{F_i} \langle p_i \rangle^2$, а (2.3.3) можно переписать в следующем виде:

$$P_i = \frac{\alpha_{i-1}}{2} \frac{\lambda}{\varphi} \left[\frac{n-i+1}{N} \right]^2 \frac{P_{i-1}^2}{F_{i-1}} \qquad (2.3.4)$$

Для уровня 0, пренебрегая чанками на других уровнях, можем считать, что p_0 всегда равно общему количеству чанков C, соответственно $P_0 = C$, $F_0 = 1$. Дисперсия p_0 равна 0, значит в этом случае $\alpha_0 = 1$. В общем случае α_i является функцией i. Для уровня $i = 1$ начальное значение $p_i(t')$ после отказа диска всегда одинаково (поскольку $p_0 = C$), значит плотность распределения $\tilde{p}_1$ постоянна в диапазоне от нуля до максимального значения, равного Cn/N. Для такого распределения $D[\tilde{p}_1] = \frac{1}{3} \langle \tilde{p}_1 \rangle^2$, соответственно $\alpha_1 = \frac{4}{3}$.

С учетом того, что $P_0 = C$, $F_0 = 1$, используя (2.3.1), (2.3.2), (2.3.4), решим задачу нахождения вероятности отказа p_F и времени наработки до отказа $T_F = 1/p_F$ для первых 3-х уровней деградации. Сначала последовательно найдем выражения для P_i, F_i. Результаты приведены в следующей таблице.

Таблица 2. Параметры математической модели для первых трех уровней деградации

$P_0 = C$	$F_0 = 1$
$P_1 = \frac{C^2}{2} \frac{\lambda}{\varphi} \frac{n^2}{N^2}$	$F_1 = C \frac{\lambda}{\varphi} \frac{n}{N}$
$P_2 = \frac{C^3}{6} \frac{\lambda^2}{\varphi^2} \frac{n^3 (n-1)^2}{N^5}$	$F_2 = \frac{C^2}{2} \frac{\lambda^2}{\varphi^2} \frac{n^2 (n-1)}{N^3}$

	$F_3 = \frac{C^3}{6}\frac{\lambda^3}{\varphi^3}\frac{n^3(n-1)^2(n-2)}{N^6}$

Подставим $\lambda = \frac{1}{T_1}$, $\varphi = \frac{N}{k+1}\mu = \frac{N}{k+1}\frac{1}{T_R}$ и найдем среднее время наработки до отказа $T_F = \frac{1}{p_F} = \frac{1}{F_m * \lambda} = \frac{T_1}{F_m}$ для значений избыточности $m = n - k$ от 0 до 3:

Таблица 3. Среднее время наработки до потери данных в математической модели

Схема хранения	*Время наработки до отказа*
(n, n)	$T_F = T_1$
$(n, n-1)$	$T_F = T_1 * \frac{T_1}{CT_R} * \frac{N^2}{n^2}$
$(n, n-2)$	$T_F = 2T_1 * \left[\frac{T_1}{CT_R}\right]^2 * \frac{N^5}{n^2(n-1)^3}$
$(n, n-3)$	$T_F = 6T_1 * \left[\frac{T_1}{CT_R}\right]^3 * \frac{N^9}{n^3(n-1)^2(n-2)^4}$

Сравним полученные результаты с формулой (2.2.16), полученной в Марковской модели. Как и в формуле (2.2.16), результат демонстрирует зависимость от отношения T_1/T_R в степени, равной избыточности схемы хранения. Но при этом степень в зависимости от количества блоков n оказывается значительно больше. Количество чанков C входит в формулу для T_F только в произведении с T_R, что вполне ожидаемо, поскольку если общее число чанков увеличить, а время восстановления одного чанка уменьшить в том же отношении, то результат не должен измениться.

2.3.4. Сравнение надежности СХД с разбиением данных на блоки с надежностью классического RAID-массива

Интересно сравнить полученные результаты с надежностью классического RAID-массива, где единицей хранения данных выступает не блок, а диск целиком [7,8]. Будем считать, что $N = R * n$, так что мы поделили наш набор из N дисков на R независимых RAID-массивов. Надежность каждого отдельного RAID-массива можно адекватно описать в рамках Марковской модели, поскольку набор одинаковых дисков с точки зрения надежности ничем не отличается от цепочки блоков данных. Значит, мы можем применить формулу (2.2.16), записав ее в виде:

$$\tilde{T}_F = T_d \frac{(k-1)!}{n!}\left[\frac{T_d}{T_r}\right]^{n-k} \qquad (2.3.5)$$

Здесь $\tilde{T}_F$ – среднее время до отказа (потери данных) для одного RAID-массива, T_d – среднее время до отказа диска, T_r – среднее время восстановления диска. Перейдем от этих величин, специфичных для отдельного RAID-массива, к величинам, общим для всего кластера, которые фигурируют в Таблица 3. Вероятность отказа в единицу времени одного RAID-массива есть $1/\tilde{T}_F$, а любого из R независимых массивов - $R/\tilde{T}_F$. То есть, среднее время до потери данных в случае R независимых массивов есть

$$T_F^{RAID} = \frac{\tilde{T}_F}{R} \qquad (2.3.6)$$

Рассуждая аналогично, приходим к выводу, что

$$T_d = T_1 * N \qquad (2.3.7)$$

В кластере, где данные хранятся в C чанках, на N дисках хранится $C * n$ блоков данных. Пусть размер этих данных составляет долю ρ от суммарной емкости кластера ($0 < \rho \leq 1$). Тогда $C * \frac{n}{N} = \rho * B$, где B – количество доступных блоков на одном диске. Среднее время восстановления диска в RAID-массиве можно выразить через среднее время восстановления одного дискового блока как

$$T_r = B * T_R = \frac{Cn}{\rho N} * T_R \qquad (2.3.8)$$

Подставляя (2.3.7), (2.3.8) в (2.3.5), с учетом (2.3.6), получаем:

$$T_F^{RAID} = T_1 \frac{(k-1)!}{(n-1)!}\left[\frac{\rho T_1}{C T_R}\right]^{n-k} * \frac{N^{2(n-k)}}{n^{n-k}} \qquad (2.3.9)$$

где мы дополнительно учли, что $R = N/n$.

Формула (2.3.9) позволяет нам определить среднее время до потери данных при тех же параметрах, что и Таблица 3, но с учетом того, что хранение данных будет осуществляться в наборе независимых RAID-массивов. Чтобы количественно оценить выигрыш в надежности вследствие разбиения данных на чанки с последующим

равномерным распределением их по дискам, вычислим отношение T_F/T_F^{RAID} для различных значений избыточности. Результат приведен в следующей таблице.

Таблица 4. Выигрыш в надежности вследствие разбиения данных на чанки.

Схема хранения (n, k)	*Выигрыш в надежности вследствие разбиения на чанки* T_F/T_F^{RAID}
(n, n)	1
$(n, n-1)$	$\frac{1}{\rho} * \frac{n-1}{n}$
$(n, n-2)$	$\frac{2}{\rho^2} * \frac{(n-2)N}{(n-1)^2}$
$(n, n-3)$	$\frac{6}{\rho^3} * \frac{(n-3)N^3}{(n-1)(n-2)^3}$

Как видно из таблицы, при хранении без избыточности разбиение на чанки не дает никакого выигрыша в надежности, что вполне объяснимо, поскольку отказ любого диска всегда приводит к потере данных, так что $T_F = T_F^{RAID} = T_1$. При хранении с избыточностью 1 выигрыш появляется только с уменьшением фактора заполненности кластера ρ. Это происходит потому, что при разбиении на чанки мы восстанавливаем только реально хранимые данные, а не весь диск целиком, как в случае с RAID-массивом. Выигрыш в надежности оказывается обратно пропорционален ρ в степени, равной избыточности схемы хранения, поскольку скорость восстановления входит в выражения для T_F в той же степени. Дополнительный выигрыш в надежности, пропорциональный[5] N/n для избыточности 2 и $(N/n)^3$ для избыточности 3, дает параллельное восстановление при отказе диска, в котором участвуют все работоспособные диски в системе (Рисунок 13). Таким образом, разбиение данных на чанки оказывается тем более выгодно, чем больше в системе дисков и чем меньше заполненность кластера. Поэтому в современных распределенных СХД разбиение данных на чанки используется повсеместно.

[5] Здесь имеется ввиду пропорциональность в пределе $N/n \to \infty$.

2.3.5. Зависимость надежности СХД от количества дисков (масштабирование)

Одним из важнейших требований к современных СХД является требование неухудшения показателей производительности и надежности при расширении системы за счет добавления к ней новых элементов – серверов и дисков. Это свойство называется масштабируемостью. На практике оно всегда выполняется лишь до определенного предела. Однако при разработке системы необходимо обязательно убедиться в том, что система обладает свойством масштабируемости пока этот предел не достигнут.

Проанализируем зависимость надежности хранилища с разбиением данных на чанки от количества дисков в системе N. Время наработки до отказа растет при увеличении количества дисков за счет увеличения скорости восстановления. Однако при этом уменьшается и T_1 обратно пропорционально N. В реальных системах рост скорости восстановления может быть ограничен производительностью сети передачи данных, поэтому в больших системах используют многосвязные конфигурации с несколькими сетевыми адаптерами у каждого узла сети, избегая немасштабируемых конфигураций вроде звезды с единственным маршрутизатором в ее центре. В следующей таблице мы сравнили зависимость среднего времени наработки до отказа T_F от числа дисков N для различных значений избыточности в предположении постоянного количества данных ($C = const$), а также в предположении постоянной заполненности дисков ($C \sim N$), т.е. когда количество данных хранимых на СХД растет пропорционально количеству дисков.

Таблица 5. Зависимость среднего времени наработки до потери данных от количества дисков в СХД.

Схема хранения	*Постоянное количество данных* ($C = const$)	*Постоянная заполненность дисков* ($C \sim N$)
(n, n)	$T_F \sim 1/N$	$T_F \sim 1/N$
$(n, n-1)$	$T_F \sim const$	$T_F \sim 1/N$
$(n, n-2)$	$T_F \sim N^2$	$T_F \sim const$
$(n, n-3)$	$T_F \sim N^5$	$T_F \sim N^2$

Как видно из таблицы, избыточности 2 достаточно, чтобы надежность не падала при добавлении в систему новых дисков при условии отсутствия прочих факторов, ограничивающих скорость восстановления, вроде производительности сети. Это объясняет популярность такого формата хранения в реальных системах. При хранении с избыточностью 3 наблюдается даже рост среднего времени наработки до отказа с увеличением количества дисков, поэтому такой формат оказывается предпочтителен, если требуется масштабируемость на очень большое количество дисков.

Заметим, что при хранении данных в наборе RAID-массивов надежность всегда падает при масштабировании кластера обратно пропорционально количеству дисков в нем. Это видно уже из (2.3.6), поскольку для отдельного RAID-массива время наработки до отказа $\tilde{T}_F$ не меняется при добавлении новых дисков, в то время как количество RAID-массивов R растет пропорционально количеству дисков. Поэтому на практике RAID-массивы не применяются, когда требуется масштабируемость системы.

2.3.6. Расчет среднего времени наработки до отказа для реального хранилища.

Воспользовавшись разработанной нами математической моделью, рассчитаем среднее время наработки до отказа для реального хранилища, состоящего из дисков, емкостью 10Tb с производительностью операций чтения/записи 100Mb/сек. Пусть T_d – среднее время наработки до отказа одного диска. С помощью Таблица 3 найдем среднее время наработки до потери данных для нескольких характерных значений T_d, например для 1 года, 2, 5 и 10 лет, что соответствует диапазону от крайне ненадежных дисков до дисков с надежностью выше среднего уровня. При этом мы будем считать, что восстановление происходит с максимальной скоростью, так что время восстановления одного диска

$$T_r = \frac{10\ \text{Tb}}{100\ \text{Mb/сек}} = 10^5\ \text{сек} \approx 0.003\ \text{года}$$

Рассмотрим наихудший с точки зрения отказоустойчивости случай, когда все диски заполнены практически полностью. Тогда, для схемы $(n, n-2)$ время наработки до потери данных не будет зависеть от количества дисков:

$$T_F^{(n,n-2)} = \frac{2}{n^2(n-1)^3}\frac{T_d^3}{T_r^2} \qquad (2.3.10)$$

Для схемы $(n, n-3)$ результат будет выглядеть следующим образом:

$$T_F^{(n,n-3)} = \frac{6N^2}{n^3(n-1)^2(n-2)^4} * \frac{T_d^4}{T_r^3} \qquad (2.2.11)$$

Результаты для $N = 25$ и 500 сведены в следующей таблице. В левой колонке указана схема кодирования и количество дисков (для схем с избыточностью 3). Результаты для схем с избыточностью 2 и 3 сгруппированы так, что в соседних строках таблицы оказываются результаты для схем кодирования, обеспечивающих одинаковую эффективность с точки зрения занимаемого дискового пространства, как например схемы $(6, 4)$ и $(9, 6)$.

Таблица 6. Расчетное среднее время наработки до потери данных для различных комбинаций параметров СХД.

Среднее время наработки до отказа диска		*1 год*	*2 года*	*5 лет*	*10 лет*
$(\mathbf{3, 1})$		$3 * 10^3$ лет	$2.5 * 10^4$ лет	$4 * 10^5$ лет	$3 * 10^6$ лет
$(\mathbf{6, 4})$		50 лет	400 лет	$6 * 10^3$ лет	$5 * 10^4$ лет
$(\mathbf{9, 6})$	$\boldsymbol{N = 25}$	$1 * 10^3$ лет	$2 * 10^4$ лет	$8 * 10^5$ лет	$1.2 * 10^7$ лет
	$\boldsymbol{N = 500}$	$6 * 10^5$ лет	$8 * 10^6$ лет	$3 * 10^8$ лет	$5 * 10^9$ лет
$(\mathbf{12, 10})$		1 год	10 лет	150 лет	$1 * 10^3$ лет
$(\mathbf{18, 15})$	$\boldsymbol{N = 25}$	1 год	20 лет	800 лет	$1.2 * 10^4$ лет

	$\boldsymbol{N = 500}$	500 лет	$8 * 10^3$ лет	$3 * 10^5$ лет	$5 * 10^6$ лет

Как видно из таблицы, варьируя параметры хранилища, легко получить время наработки до потери данных в диапазоне от одного года до возраста Земли. Такой огромный диапазон результатов объясняется высокими степенями входящих в формулы (2.3.10) и (2.3.11) параметров. Это обстоятельство делает способность теории давать адекватные предсказания надежности реального хранилища особенно актуальной с практической точки зрения.

Стоит отметить несколько практически важных результатов, которые непосредственно следуют из полученных результатов:

1. Надежность хранилища при использовании избыточности 2 быстро падает с ростом n – количества дисков в кортеже дисковых блоков. Значения $n > 10$ не следует использовать, поскольку они приводят к неприемлимо низкой надежности.
2. Выигрыш в надежности от использования схем кодирования с избыточностью 3 наблюдается только при использовании большого числа дисков. Так что схемы кодирования с $n > 10$ и избыточностью 3 можно использовать для повышения эффективности использования дискового пространства только в случаях, когда количество дисков N составляет несколько сотен и более.

2.3.7. Сравнение предсказаний математической модели с результатами имитационного моделирования

Чтобы проверить выводы описанной выше математической модели, мы смоделировали поведение кластера из 50 дисков с 2500 чанками со схемами хранения (5,3) и (5,2) для различной интенсивности дисковых отказов. Результаты, полученные для моделей A и B, показаны на рисунке ниже. Сплошными линиями показаны результаты, вычисленные в соответствии с математической моделью, точками показаны результаты симуляции – квадратами для схемы (5,3), кружками – для схемы (5,2). Каждая точка на графике была получена в результате усреднения результатов 200 симуляций. При этом моделирование не выявило статистически значимых различий между средним временем до первой потери данных и среднем временем между последовательными событиями потери данных. Для наглядности график построен в безразмерных координатах, где и интенсивность дисковых отказов и результирующее среднее время наработки до отказа соотнесены с временем восстановления блока T_R.

Верхний график (окружности) построен по результатам симуляции для схемы (5,2) в модели A. Симуляция в модели B не проводилась, поскольку для данной схемы хранения с высокой избыточностью она потребовала бы чрезвычайно длительных расчетов. Сплошной линией показаны результаты расчета в соответствии с Таблица 3. Ниже квадратами показаны результаты симуляции для схемы (5,3). Полыми квадратами показаны результаты для модели A, сплошными – для модели B. Во втором случае время наработки до отказа оказывается несколько меньше вследствие влияния конечности числа дисков на скорость восстановления. В частности, в модели B число одновременно восстанавливаемых чанков должно быть целым числом, поэтому некоторые диски оказываются не вовлечены в процесс восстановления. Модель A этих ограничений не учитывает. Нижняя сплошная линия показывает результаты расчета для схемы (5,3) в соответствии с Таблица 3. Как видно из рисунка, разработанная нами математическая модель с хорошей точностью совпадает с результатами симуляции.

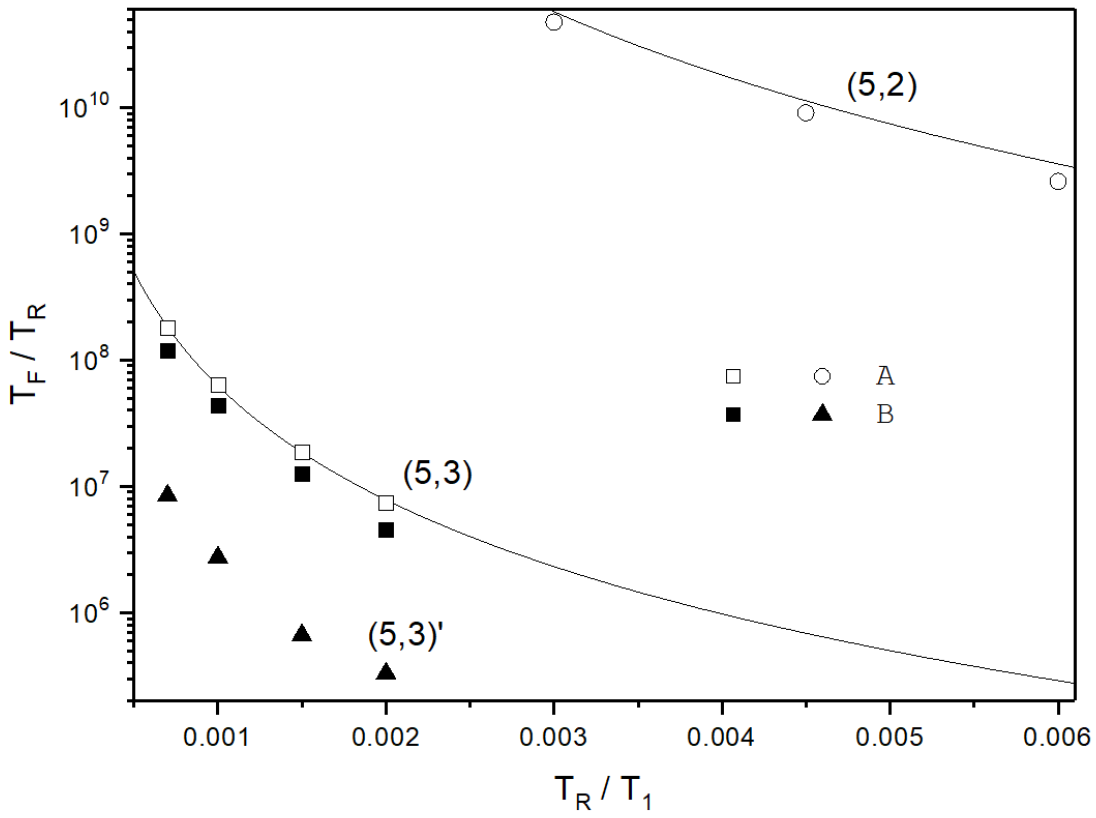

Рисунок 15. Результаты симуляции надежности СХД (символы) в сравнении с расчетом по формулам из Таблица 3 (линии).

Отдельно на графике показаны результаты симуляции, когда чанки для восстановления выбирается случайным образом без упорядочивания по уровню деградации. Такие точки на графике изображены треугольниками и обозначены как схема (5,3)'. Как видим, надежность системы без приоритезации восстановления оказалась более чем на порядок хуже надежности системы с приоритетным восстановлением чанков на больших уровнях деградации. Заметим, что приоритетность восстановления была нами неявно использована при построении теоретической модели, когда мы рассматривали восстановление на каждом уровне деградации, как идущее с постоянной скоростью независимо от остальных уровней. На самом деле постоянна только суммарная скорость восстановления. Рассматривая восстановление на конкретном уровне деградации i, мы пренебрегали восстановлением на уровнях $> i$, считая количество чанков на них пренебрежимо малым. Уровни деградации $< i$, которыми очевидно нельзя пренебречь, мы при этом не рассматривали в силу того, что их восстановление откладывалось до окончания восстановления на текущем уровне i.

Таким образом, мы разработали математическую модель надежности СХД, учитывающую отказы дисков и последовательное восстановление потерянных блоков со скоростью, ограниченной скоростью доступа к диску. При этом мы предположили, что в

системе достаточно много чанков и они достаточно равномерно распределены по дискам, чтобы любой дисковый сбой затрагивал хотя-бы один чанк на любом диске, в том числе уже поврежденный, если они есть. В следующей главе мы уточним нашу модель в случаях, если это условие не выполняется, и рассмотрим преимущества таких конфигураций. Кроме того, мы выйдем за рамки допущения о том, что выход из строя дисков являются единственным источником отказов и рассмотрим более общую модель отказов, которая имеет важные следствия для обеспечения надежности хранилища.

3. Оптимизации надежности СХД

В предыдущей главе мы рассмотрели классическую Марковскую модель надежности хранения данных и построили математическую модель реального многодискового хранилища, более адекватно описывающую реальные СХД. В данной главе мы рассмотрим теоретически и проверим на моделях оценки влияния различных факторов на надежность и масштабируемость СХД, а именно влияния различных политик размещения дисковых блоков и скрытых повреждений. Полученные в данной главе результаты позволят нам глубже понять механизмы потери данных в реальных СХД и оптимизировать параметры хранилища для обеспечения максимальной надежности.

3.1. Группы размещения

Рассмотренная в (2.3.3) математическая модель основана на предположении о том, что на любом уровне деградации на каждом диске достаточно чанков, чтобы отказ любого диска привел к повреждению хотя бы одного чанка. Однако для пользовательских файлов это предположение может оказаться неверным. Представим себе, что один из пользователей системы владеет единственным файлом с единственным (n, k) чанком. Допустим, в системе произошла потеря данных вследствие отказа $n - k + 1$ дисков. Вероятность того, что при этом окажется поврежден конкретный (n, k) чанк равна:

$$p = \frac{n}{N} * \frac{n-1}{N-1} * \ldots * \frac{k}{N-n+k}$$

Первый член в этом произведении есть вероятность того, что первый отказавший диск окажется среди n дисков, где хранятся блоки, принадлежащие пользователю. Второй член есть вероятность того, что второй отказавший диск окажется среди $n - 1$ дисков, где хранятся блоки, оставшиеся неповрежденными после первого отказа, и т.д. Значит, с точки зрения пользователя надежность хранилища, как среднее время наработки до потери его единственного чанка, будет как минимум в $1/p$ раз больше, чем среднее время до потери хотя бы одного чанка во всем хранилище. На этом простом наблюдении основана идея *групп размещения* [20].

Вместо того, чтобы располагать чанки по дискам случайным образом, сгенерируем некоторое количество кортежей из n дисков и будем использовать для размещения чанков только эти кортежи, которые мы будем называть группами размещения. Даже если файл пользователя состоит из большого количества чанков, мы можем использовать для них небольшое количество групп размещения, так что надежность хранения этого файла существенно повысится. При этом, однако, возникает ряд вопросов

- как такой способ размещения чанков скажется на надежности хранилища в целом?
- какого количества групп размещения достаточно для обеспечения приемлемой надежности хранилища в целом?

Наивный ответ на эти вопросы, который неявно подразумевался в [20], заключается в том, что надежность хранилища должна только вырасти от ограничения числа возможных дисковых кортежей, так как при этом уменьшается вероятность того, что отказ некоторого набора дисков затронет хотя бы один существующий дисковый кортеж. Однако такое объяснение полностью игнорирует влияние скорости восстановления на надежность СХД.

3.1.1. Имитационная модель надежности СХД с учетом групп размещения

Чтобы найти ответ на поставленные вопросы, мы добавили в нашу имитационную модель В алгоритм генерации дисковых кортежей, использующий группы размещения. Для создания групп размещения аналогично [20] мы использовали следующий *алгоритм генерации дисковых кортежей*:

1. все N дисков заносятся в список в случайном порядке
2. список дисков делится на $\lfloor N/n \rfloor$ непересекающихся групп размещения по n дисков в каждой (остаток игнорируется)
3. процедура повторяется, начиная с шага 1 до тех пор, пока не будет сгенерировано необходимое количество групп размещения

Для проверки влияния количества групп размещения на надежность хранилища мы смоделировали поведение кластера из 50 дисков с 2500 чанками со схемой хранения (5,3). Относительная интенсивность дисковых отказов составляла $\frac{\lambda}{\mu} = \frac{T_R}{T_1} = 0.0015$. Полученная зависимость относительного времени наработки до отказа T_F/T_R от количества групп размещения K показана на следующем рисунке. Пунктирной линией показан результат симуляции с использованием полностью случайного размещения дисковых кортежей.

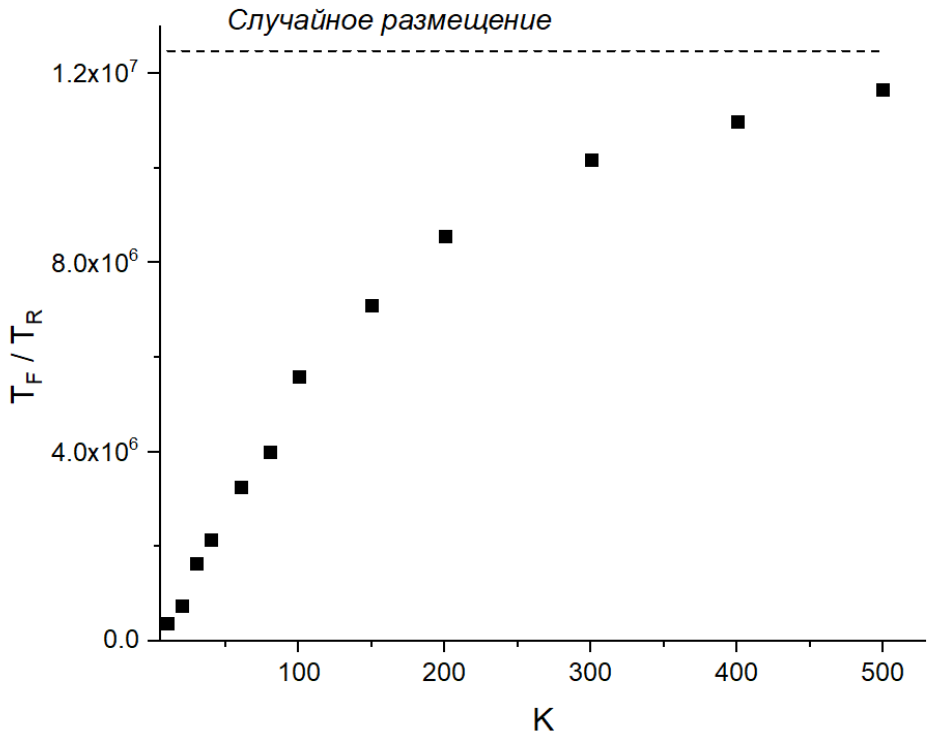

Рисунок 16. Зависимость среднего времени наработки до отказа от количества групп размещения.

Как видно из графика, зависимость надежности хранилища, выраженной в среднем времени наработки до отказа, от количества групп размещения оказывается противоположной ожидаемой из наивных соображений. Надежность увеличивается с увеличением количества групп размещения, плавно приближаясь к надежности хранилища при случайном размещении дисковых кортежей (пунктир). Чтобы понять природу этой зависимости и оценить необходимое количество групп размещения для получения приемлемой надежности, мы доработали нашу математическую модель хранилища с тем, чтобы она учитывала возможность выбора дисковых кортежей из конечного набора групп размещения.

3.1.2. Математическая модель хранилища с использованием групп размещения

Группа размещения представляет собой множество дисков. Для краткости мы будем называть такое множество просто кортежем. Рассмотрим множество из K кортежей $T = \{t\}$, каждый из которых представляет собой случайную выборку из n элементов множества дисков $D = \{d\}$. Пусть $N = |D|$ - количество дисков. Пусть элементы множества D – это просто номера дисков $D = \{1,2,\dots N\}$. Назовем покрытием множества кортежей $\delta(T)$ множество дисков, полученное как объединение всех кортежей из множества T: $\delta(T) = \bigcup_{t\in T} t$. Нас будет интересовать математическое ожидание количества дисков в таком покрытии, назовем его, следуя принятому в литературе (см. [20]) названию, *разбросом* $S_0 = \langle|\delta(T)|\rangle$. Чтобы найти S_0, вычислим вероятность того, что случайно выбранный из множества D диск d не принадлежит покрытию $\delta(T)$. Для этого он не должен принадлежать ни одному из кортежей $t \in T$. Вероятность того, что случайно выбранный диск не принадлежит кортежу из n случайно выбранных дисков равна $1 - \frac{n}{N}$. Для K случайно выбранных кортежей вероятность того, что диск не принадлежит ни одному из них равна $\left[1 - \frac{n}{N}\right]^K$. Значит, искомый разброс может быть найден по формуле:

$$S_0 = N\left(1 - \left[1 - \frac{n}{N}\right]^K\right) \qquad (3.1.1)$$

Пользуясь первым замечательным пределом, в приближении $N \gg 1$, получаем:

$$S_0 \approx N\left(1 - e^{-\frac{nK}{N}}\right) \qquad (3.1.2)$$

Полученный результат позволяет нам ответить на вопрос, какова вероятность того, что поврежденный диск окажется принадлежащим хотя бы одному кортежу в ситуации, когда количество кортежей (чанков или групп размещения) меньше или порядка количества дисков. Однако, даже если чанков изначально много больше, чем дисков, аналогичный вопрос остается актуальным в ситуации, когда произошло несколько дисковых ошибок и мы интересуемся вероятностью того, что найдется чанк, изначально содержавший все потерянные диски и хотим оценить скорость восстановления после такого события. Для того, чтобы получить оценки для описанной ситуации, введем понятие связанного множества кортежей:

$$T_{\{d_1,d_2\dots d_q\}} = \{t | t \in T, d_1 \in t, d_2 \in t \dots d_q \in t\}$$

Иначе говоря, это множестве кортежей, каждый из которых содержит множество дисков $\{d_1, d_2 \dots d_q\}$. При этом мы дополнительно потребуем, чтобы связанное множество кортежей было непустым. Для этого необходимо и достаточно, чтобы $d_1 \in \delta(T), d_2 \in$

$\delta(T_{\{d_1\}}) \dots d_q \in T_{\{d_1,d_2 \dots d_{q-1}\}}$. Следующий рисунок схематически иллюстрирует множество дисков, образующих покрытие связанного множества трехдисковых кортежей, содержащих диск с номером i.

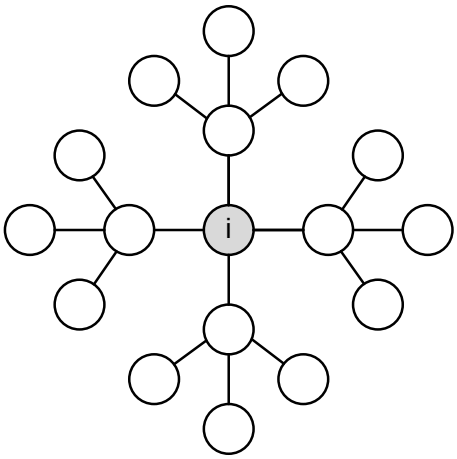

Рисунок 17. Множество дисков, образующее покрытие кортежей, содержащих i-й диск.

После связывания второго диска с номером j получим множество, схематически показанное на рисунке ниже.

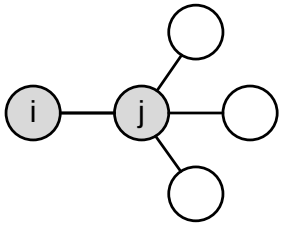

Рисунок 18. Множество дисков, образующих покрытие кортежей, содержащих диски i и j.

Разбросом порядка q назовем математическое ожидание количества дисков в покрытии связанного множества кортежей: $S_q = \langle \left| \delta \left(T_{\{d_1,d_2 \dots d_q\}} \right) \right| \rangle$. Найдем общую формулу для вычисления S_q, считая, что это некоторая функция от количества дисков, размера кортежа и их количества, вид которой мы и собираемся определить. Пусть мы выбрали диск $d_1 \in \delta(T)$ и хотим найти $S_1 = \langle \left| \delta(T_{\{d_1\}}) \right| \rangle$. Сначала найдем среднее количество кортежей в $T_{\{d_1\}}$: $K_1 = \langle \left| T_{\{d_1\}} \right| \rangle_{d_1 \in \delta(T)}$. Заметим, что если просуммировать $\left| T_{\{d_1\}} \right|$ по всем дискам, то мы получим суммарное количество кортежей помноженное на n - количество дисков в кортеже (поскольку каждый кортеж в итоге будет посчитан n раз): $\sum_{d_1 \in D} \left| T_{\{d_1\}} \right| = K * n$. Значит, среднее значение $\left| T_{\{d_1\}} \right|$ можно напрямую вычислить по формуле:

$$K_1 = \frac{\sum_{d_1 \in D} |T_{\{d_1\}}|}{|\delta(T)|} = \frac{K * n}{S_0}$$

Для нахождения S_1 воспользуемся следующим трюком. Исключим из рассмотрения диск d_1, и удалим его из всех кортежей в $T_{\{d_1\}}$. Задача сведется к нахождению разброса S_0 для K_1 кортежей по $n-1$ дисков в каждом, случайно выбранных из $N-1$ дисков:

$$S_1(N, n, K) = 1 + S_0(N-1, n-1, \frac{K * n}{S_0})$$

Аналогично рассуждая, можно получить рекурсивные формулы для вычисления количества кортежей K_q и их разброса S_q после ‘связывания‘ q дисков. Пусть нам уже известны K_{q-1} и S_{q-1}. Исключив из рассмотрения $q-1$ связанных дисков, получим K_{q-1} кортежей по $n-q+1$ дисков в каждом, случайно выбранные из $N-q+1$ дисков с разбросом, равным $S_{q-1}-q+1$. После связывания q-го диска получим среднее число кортежей:

$$K_q = \frac{K_{q-1} * (n-q+1)}{S_{q-1}-q+1}$$

Исключив из рассмотрения q-й связанный диск, сведем задачу к нахождению разброса K_q кортежей по $n-q$ дисков в каждом, случайно выбранных из $N-q$ дисков:

$$S_q = q + S_0(N-q, n-q, K_q) \qquad (3.1.3)$$

С помощью этого выражения формулу для K_q можно переписать в виде:

$$K_q = \frac{K_{q-1} * (n-q+1)}{S_0(N-q+1,\ n-q+1,\ K_{q-1})} \qquad (3.1.4)$$

С помощью (3.1.2), (3.1.3), (3.1.4) получим приближенные формулы для $q = 1, 2$ для случая $K \gg N$, когда мы можем считать, что $S_0(N, n, K) = N$. Результаты для K_1, K_2, S_1, S_2 сведены в следующей таблице:

Таблица 7. Среднее число дисковых кортежей и их разброс после связывания одного и двух дисков.

$K_1 = \frac{Kn}{N}$	$S_1 = 1 + (N-1)\left(1 - e^{-\frac{Kn(n-1)}{N(N-1)}}\right)$
$K_2 = \frac{Kn(n-1)}{N(N-1)}\left(1 - e^{-\frac{Kn(n-1)}{N(N-1)}}\right)^{-1}$	$S_2 = 2 + (N-2)\left(1 - \exp\left(-\frac{Kn(n-1)(n-2)}{N(N-1)(N-2)}\left(1 - e^{-\frac{Kn(n-1)}{N(N-1)}}\right)^{-1}\right)\right)$

Чтобы проверить полученные нами выводы, мы провели численный эксперимент. На множестве из $N = 50$ дисков мы создавали группы размещения в виде кортежей по $n = 5$

дисков в каждой и оценивали разброс S_1, S_2 методом Монте-Карло. Полученная зависимость разброса S_1 (квадратные точки) и S_2 (круглые точки) от количества групп размещения K показана на следующем графике.

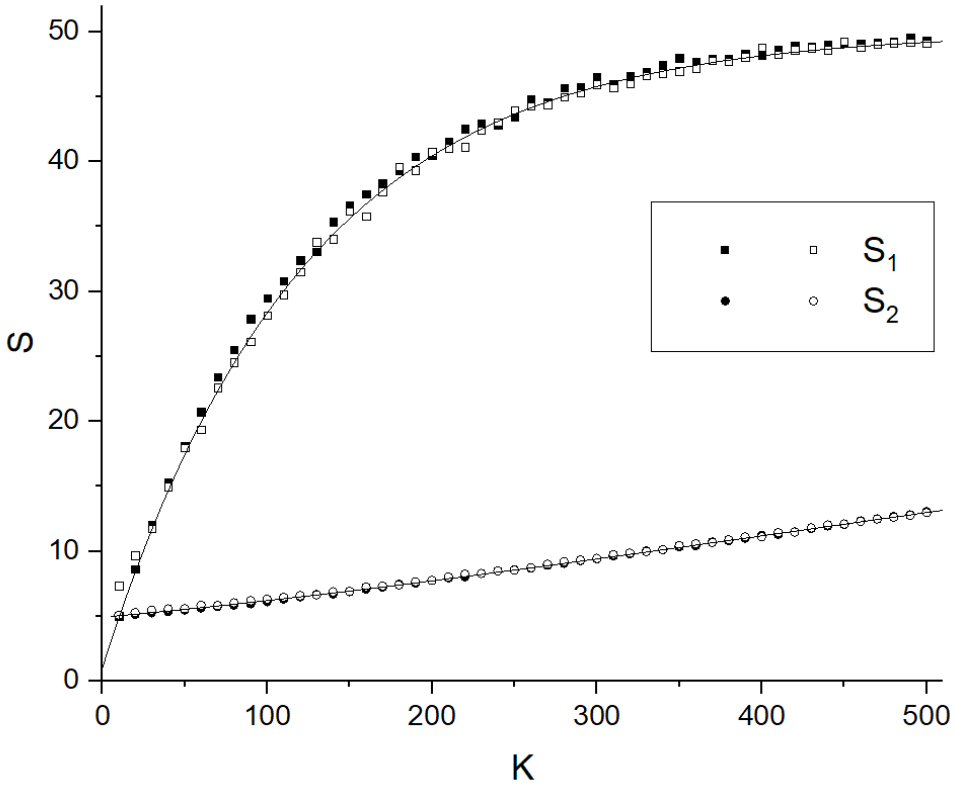

Рисунок 19. Зависимость разброса кортежей после связывания одного (S_1) и двух дисков (S_2) от числа кортежей K.

Для генерации групп размещения мы использовали 2 методики. Первая использовала *алгоритм генерации дисковых кортежей,* описанный в разделе 3.1.1. Вторая методика заключалась в полностью случайном выборе групп размещения. Результаты, полученные для первой методики, показаны сплошными точками, для второй – полыми. Как видно из рисунка, способ генерации групп размещения не играет существенной роли - разница проявляется только при очень малом количестве групп размещения. Сплошными линиями показаны результаты вычисления разброса по формулам из Таблица 7. Как видно из графика, наш анализ весьма точно соответствует результатам численного эксперимента.

Рассмотрим теперь, как использование групп размещения влияет на надежность хранилища. Наша математическая модель надежности хранилища, описанная в разделе 2.3.3, неявно предполагала, что на каждом уровне деградации количество чанков много больше количества дисков N, поэтому разброс этих чанков близок к N. При

использовании групп размещения это условие может нарушаться, поэтому нам необходимо учесть разброс чанков на каждом уровне деградации в наших уравнениях.

Поток отказов с уровня i на уровень $i+1$ не должен измениться если учесть, что разброс чанков $S_i < N$, поскольку уменьшение вероятности повреждения диска компенсируется увеличением среднего количества чанков, приходящихся на один диск. Уравнение (2.3.1) мы перепишем в виде:

$$F_i = P_{i-1}\frac{n-i+1}{N}\frac{\lambda}{\varphi_i} \qquad (3.1.5)$$

Где φ_i – суммарная скорость восстановления с уровня i, которая может зависеть от разброса чанков на этом уровне.

Уравнение (2.3.2) для вероятности потери данных приобретает следующий вид ($m = n - k$):

$$p_F = F_m * \lambda * \frac{S_m}{N} \qquad (3.1.6)$$

Где мы учли, что число дисковых отказов в единицу времени, приводящих к потере блоков на уровне i, равно $\lambda * \frac{S_i}{N}$. Уравнение (2.3.3) по тем же причинам принимает вид:

$$P_i = \langle p_i(t)\rangle_t = \frac{\lambda}{2\varphi_i}\frac{S_{i-1}}{N}\left[\frac{n-i+1}{S_{i-1}}\right]^2 \langle p_{i-1}(t)^2\rangle_t = \frac{\lambda}{2\varphi_i}\frac{(n-i+1)^2}{N*S_{i-1}}\langle p_{i-1}(t)^2\rangle_t$$

Откуда получаем аналог уравнения (2.3.4):

$$P_i = \frac{\alpha_{i-1}}{2}\frac{\lambda}{\varphi_i}\frac{(n-i+1)^2}{N*S_{i-1}}\frac{P_{i-1}^2}{F_{i-1}} \qquad (3.1.7)$$

Результаты для первых 3-х уровней деградации приведены в следующей таблице:

Таблица 8. Параметры математической модели надежности СХД с учетом влияния разброса кортежей S_i на i-м уровне деградации.

$P_0 = C$	$F_0 = 1$
$P_1 = \frac{C^2}{2}\frac{\lambda}{\varphi_1}\frac{n^2}{NS_0}$	$F_1 = C\frac{\lambda}{\varphi_1}\frac{n}{N}$
$P_2 = \frac{C^3}{6}\frac{\lambda^2}{\varphi_1\varphi_2}\frac{n^3(n-1)^2}{N^2S_0^2S_1}$	$F_2 = \frac{C^2}{2}\frac{\lambda^2}{\varphi_1\varphi_2}\frac{n^2(n-1)}{N^2S_0}$
	$F_3 = \frac{C^3}{6}\frac{\lambda^3}{\varphi_1\varphi_2\varphi_3}\frac{n^3(n-1)^2(n-2)}{N^3S_0^2S_1}$

Как видим, λ и N входят во все выражения в степенях противоположного знака, что закономерно, поскольку теперь N влияет только на вероятность отказа единичного диска, остальные свойства кластера зависят от S_i.

Подставим в Таблица 8 выражения для скорости (вероятности в единицу времени) дисковых отказов $\lambda = \frac{1}{T_1}$, и восстановления $\varphi_i = \frac{S_i}{k+1}\mu = \frac{S_i}{k+1}\frac{1}{T_R}$, где мы учли, что в востановлении на уровне деградации i участвуют S_i дисков. Таким образом, найдем среднее время наработки до отказа $T_F = \frac{1}{p_F} = \frac{N}{F_m * S_m * \lambda} = \frac{T_1 * N}{F_m * S_m}$ для значений избыточности $m = n - k$ от 0 до 3:

Таблица 9. Среднее время наработки до потери данных с учетом влияния разброса кортежей S_i на i-м уровне деградации.

Схема хранения	*Время наработки до отказа*
(n, n)	$T_F = T_1 \frac{N}{S_0}$
$(n, n-1)$	$T_F = T_1 * \frac{T_1}{CT_R} * \frac{N^2}{n^2}$
$(n, n-2)$	$T_F = 2T_1 * \left[\frac{T_1}{CT_R}\right]^2 * \frac{N^3 * S_0 * S_1}{n^2(n-1)^3}$
$(n, n-3)$	$T_F = 6T_1 * \left[\frac{T_1}{CT_R}\right]^3 * \frac{N^4 * S_0^2 * S_1^2 * S_2}{n^3(n-1)^2(n-2)^4}$

Как видно из сравнения Таблица 9 и Таблица 3, для схемы $(n, n-2)$ среднее время наработки до отказа T_F отличается от полученного без учета разброса множителем $S_0 S_1 / N^2$. Чтобы проверить полученный результат, мы наложили на результаты симуляции, показанные на Рисунок 16, кривую, полученную умножением результата для случайного распределения чанков на $S_0 S_1 / N^2$, считая что случайное распределение чанков соответствует нашей математической модели без учета влияния групп размещения. Результат показан на следующем рисунке сплошной линией. Как видно из рисунка, разработанная нами теоретическая модель демонстрирует хорошую степень соответствия результатам симуляции. Полученное по результатам симуляции время наработки до отказа оказывается несколько меньше теоретического вследствие влияния конечности числа дисков на скорость восстановления. В частности, число одновременно восстанавливаемых

чанков должно быть целым числом, поэтому некоторые диски оказываются не вовлечены в процесс восстановления.

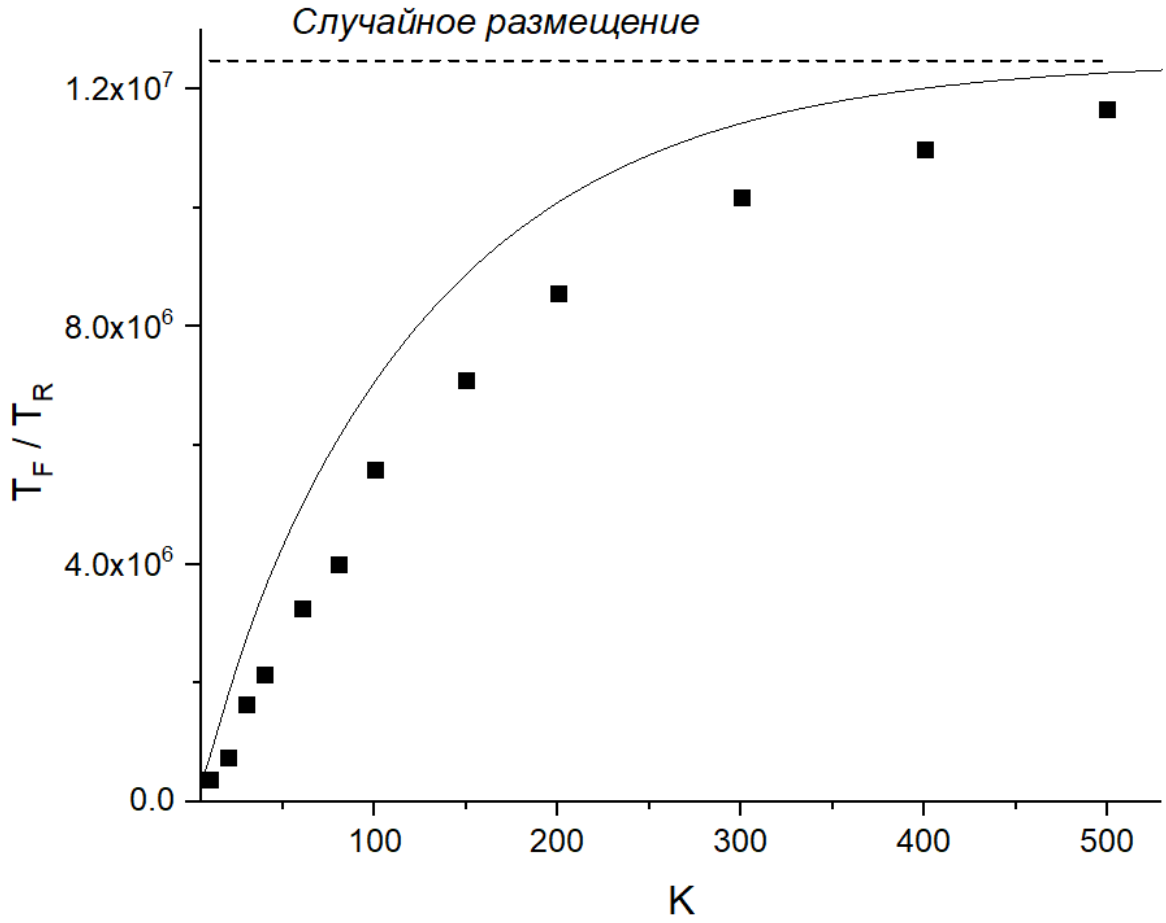

Рисунок 20. Зависимость среднего времени наработки до отказа от количества групп размещения (символы) в сравнении с теоретической моделью (сплошная кривая).

Качественно понять природу полученной зависимости можно следующим образом. Вероятность потери данных вследствие дискового отказа определяется разбросом дисковых кортежей на критическом уровне деградации $m = n - k$ и средним количеством чанков на этом уровне деградации. При уменьшении разброса S_m мы увеличиваем время восстановления (поскольку уменьшается число участвующих в нем дисков), но в то же время уменьшаем вероятность отказа любого из S_m дисков. Значит, надежность хранилища, выраженная в среднем времени наработки до отказа (потери данных), не должна зависеть от S_m, что и демонстрирует Таблица 9. Предположим теперь, что мы уменьшили разброс S_i, где $i < m$. Рассуждая аналогично, приходим к выводу, что при этом вероятность повреждения чанков на уровне деградации i в единицу времени вследствие очередного дискового отказа не изменится. Однако при этом среднее количество чанков, которые перейдут на уровень деградации $i + 1$ вследствие очередного дискового отказа, должно

увеличиться обратно пропорционально уменьшению S_i (считая общее количество чанков на уровне i постоянным), соответственно увеличится время восстановления, а значит и вероятность потери данных в хранилище.

Таким образом, для получения приемлемой надежности важно, чтобы количество групп размещения K было достаточно большим. Допустим, мы хотим получить среднее время наработки до отказа, равное 1/2 от этого параметра для случайного размещения чанков. Тогда для схемы $(n, n-2)$ имеем уравнение $\frac{S_0 S_1}{N^2} = 1/2$. Используя приближенные соотношения, полученные из Таблица 7,

$$S_0 \approx N, \qquad S_1 \approx N\left(1 - e^{-\frac{Kn(n-1)}{N(N-1)}}\right)$$

приходим к соотношению:

$$\frac{Kn(n-1)}{N(N-1)} \approx \ln(2)$$

Отсюда необходимое количество групп размещения:

$$K_{1/2}^{(n,n-2)} \approx 0.7 * \frac{N(N-1)}{n(n-1)}$$

Для получения надежности на уровне 3/4 от надежности хранилища для случайного размещения чанков, коэффициент в этой формуле необходимо изменить на $\ln(4)$, то есть

$$K_{3/4}^{(n,n-2)} \approx 1.4 * \frac{N(N-1)}{n(n-1)}$$

Для схемы хранения $(n, n-3)$ среднее время наработки до отказа с учетом групп размещения отличается от такового для случайного размещения чанков множителем $S_0^2 S_1^2 S_2 / N^5$, как следует из Таблица 9. Допустим, мы хотим получить среднее время наработки до отказа, равное 1/2 от этого параметра для случайного размещения чанков. Используя приближенные соотношения, полученные из Таблица 7,

$$S_0 \approx S_1 \approx N, \qquad S_2 \approx N\left(1 - e^{-\frac{Kn(n-1)(n-2)}{N(N-1)(N-2)}}\right)$$

приходим к соотношению:

$$\frac{Kn(n-1)(n-2)}{N(N-1)(N-2)} \approx \ln(2)$$

Отсюда необходимое количество групп размещения:

$$K_{1/2}^{(n,n-3)} \approx 0.7 * \frac{N(N-1)(N-2)}{n(n-1)(n-2)}$$

Для получения надежности на уровне 3/4 от надежности хранилища для случайного размещения чанков, коэффициент в этой формуле необходимо изменить на $\ln(4)$, то есть

$$K_{3/4}^{(n,n-3)} \approx 1.4 * \frac{N(N-1)(N-2)}{n(n-1)(n-2)}$$

Таким образом, проведенный нами анализ показывает, что, являясь интересным механизмом оптимизации надежности хранения данных с точки зрения пользователя, группы размещения не увеличивают надежность всего хранилища. Более того, требование сохранения надежности хранилища на приемлемом уровне накладывает ограничение снизу на общее количество групп размещения, причем это количество растет примерно пропорционально количеству дисков в системе в степени, равной избыточности.

В следующей главе мы рассмотрим еще один важный фактор, который необходимо учитывать для оптимизации надежности хранения данных – скрытые ошибки. Их опасность в том, что они никак не проявляют себя вплоть до попытки чтения данных. Поэтому, чтобы обеспечить эффективное восстановление, мы вынуждены читать данные, даже если их никто не запрашивает. И даже в этом случае вероятность потери данных вследствие скрытых ошибок в реальных системах оказывается не меньше, чем вследствие полного отказа диска.

3.2. Скрытые ошибки и скраббинг

Дисковые ошибки не всегда сводятся к полному отказу диска. Вторая категория ошибок – ошибки чтения сектора или группы секторов, которые чаще всего являются следствием врожденных дефектов или приобретенных повреждений поверхности вращающегося диска, либо запоминающих транзисторов твердотельного носителя информации. Такие ошибки, как и повреждения диска, следствием которых они являются, называют скрытыми, поскольку они никак не проявляют себя вплоть до попытки чтения поврежденной области диска. В данном разделе мы рассмотрим влияние скрытых ошибок на надежность хранилища, постоим математическую модель надежности СХД с учетом скрытых ошибок, а также рассмотрим методы оптимизации надежности хранилища с учетом наличия скрытых повреждений.

3.2.1. Влияние скрытых ошибок на надежность СХД

Отличительной особенность скрытых ошибок является то, что они остаются незамеченными до попытки чтения данных с диска, поэтому они и называются скрытыми или латентными. Понятно, что если данные не читаются с диска, то рано или поздно мы можем столкнуться с ситуацией, когда данные, хранящиеся в (n, k) схеме, уже невозможно восстановить, поскольку более чем $n - k$ блоков потеряно из-за скрытых ошибок. А при наличии дисковых отказов возможна ситуация, когда после отказа p блоков $(p \leq n - k)$ данные невозможно восстановить вследствие ошибок чтения других q блоков, так что $p + q > n - k$. Поэтому для надежного хранения данных критически важно постоянно проверять целостность блоков данных, сохраненных на каждом диске. Обычно это делается путем последовательного чтения блоков данных и сравнения их содержимого с контрольной суммой, которая как правило хранится отдельно от данных. При этом фиксируются как ошибки, приводящие к невозможности чтения данных, так и ошибки, приводящие к чтению искаженных данных. Этот процесс обычно называется скраббингом [21].

Как показывает практика, среднее время наработки до отказа конкретного сектора превышает время наработки до отказа всего диска на несколько порядков. Однако при этом среднее время наработки до возникновения ошибки чтения любого из множества секторов диска оказывается в несколько раз меньше времени наработки до отказа диска [21, 22]. Это обстоятельство имеет весьма существенные следствия для оценок надежности хранилища данных, которые мы рассмотрим в следующем разделе.

3.2.2. Математическая модель надежности СХД с учетом скрытых ошибок

Существующие модели надежности СХД, учитывающие наличие скрытых ошибок, основаны исключительно на Марковской модели (например [27]). Такого рода модели не вполне адекватно описывают существующие системы, поскольку они не учитывают скоррелированность отказа дисковых блоков при выходе диска из строя, а также считают, что восстановление всех поврежденных блоков происходит независимо, хотя на практике они восстанавливаются последовательно со скоростью, ограниченной скоростью доступа к диску. Вышеперечисленные факторы учитывает разработанная нами математическая модель надежности СХД, рассмотренная в главе 2.3.3. Чтобы количественно охарактеризовать влияние скрытых ошибок на надежность хранилища, мы доработали нашу математическую модель надежности с учетом наличия скрытых ошибок. Результат мы сформулируем в форме следующей теоремы, весьма важной для практических реализаций СХД.

Теорема о скрытых ошибках

Пусть T_B - среднее время до возникновения ошибки чтения любого блока на диске, T_d - среднее время до отказа диска целиком. Пусть T_R – время, необходимое для восстановления утраченного блока, τ – время, необходимое для проверки блока на наличие скрытых ошибок, причем блоки восстанавливаются и проверяются последовательно один за другим. Пусть вероятность потери данных в единицу времени вследствие одних только дисковых отказов, без учета скрытых ошибок, равна p_F. Обозначим вероятность потери данных в единицу времени с учетом скрытых ошибок как $\tilde{p}_F$. Тогда при использовании (n, k) схемы хранения справедливо следующее неравенство:

$$\tilde{p}_F > p_F\left(1 + \frac{k}{k+1} * \frac{T_d}{T_B} * \frac{\tau}{T_R}\right)$$

Следствие

Пусть $T_B < T_d/2$, что для реально используемых дисков выполняется практически всегда (см. [21, 22]). Предположим, что восстановление происходит с максимальной скоростью, ограниченной производительностью диска, чтобы минимизировать p_F. Поскольку скорость проверки диска на наличие скрытых ошибок тоже ограничена производительностью диска, τ не может быть меньше, чем T_R. Тогда $\tilde{p}_F > 2p_F$, иначе говоря, скрытые ошибки являются основным источником потери данных.

Доказательство

Рассмотрим возникновение скрытых ошибок как Марковский процесс независящий от дисковых отказов и восстановления после них. Этот Марковский процесс затрагивает отдельные блоки и имеет всего 2 состояния. В состоянии 0 блок неповрежден, в состоянии 1 – поврежден (содержит скрытую ошибку). Состояние 1 не является поглощающим, поскольку блок обычно можно восстановить, используя блоки с других дисков из того же кортежа (чанка). Ситуации, когда блок невозможно восстановить, относительно редки, и на данном этапе мы ими пренебрегаем. Наша промежуточная цель – найти долю поврежденных блоков ρ, иначе говоря, вероятность того, что случайно выбранный блок окажется поврежден.

Пусть $\gamma = 1/T_b$ – скорость появления скрытых ошибок, соответственно T_b – среднее время до появления скрытой ошибки конкретного блока, $\sigma = 1/T_S$ – скорость восстановления поврежденных блоков, а T_S – средний период скраббинга, т.е. время, необходимое для проверки всех блоков.

Рассмотрим квазистационарное состояние ансамбля из K статистически независимых блоков, где p_0 неповрежденных блоков находятся в состоянии 0, а p_1 поврежденных блоков находятся в состоянии 1, так что $p_0 + p_1 = K$. Тогда искомая вероятность обнаружить поврежденный блок $\rho = \lim\limits_{K\to\infty} \frac{p_1}{K}$. За бесконечно малый интервал времени dt количество блоков $dt * \gamma p_0$ перейдет из состояния 0 в состояние 1, в то время как количество блоков $dt * \sigma p_1$ перейдет из состояния 1 в состояние 0. В квазистационарном ансамбле эти количества равны, так что $\gamma p_0 = \sigma p_1$, откуда $p_0 = \frac{\sigma}{\gamma} p_1$, значит $\left(1 + \frac{\sigma}{\gamma}\right) * p_1 = K$, отсюда $p_1 = K / \left(1 + \frac{\sigma}{\gamma}\right)$. Считая, что $\gamma \ll \sigma$, для ρ получаем выражение:

$$\rho \approx \frac{\gamma}{\sigma} = \frac{T_S}{T_b} \qquad (3.2.1)$$

В дальнейшем нам понадобится величина, производная от ρ, обозначим ее $\rho_q^{(l)}$ – вероятность того, что среди случайно выбранных l блоков обнаружится не менее q поврежденных. Чтобы вычислить эту вероятность в общем случае, начнем с нахождения $\rho_1^{(l)}$. Заметим, что $\rho_1^{(l)}$ в сумме с вероятность обнаружить все l блоков неповрежденными равна 1. Вероятность найти l блоков неповрежденными есть $(1-\rho)^l$. Значит $\rho_1^{(l)} = 1 - (1-\rho)^l$. Считая, что $\gamma \ll \sigma$, а значит $\rho \ll 1$, получаем:

$$\rho_1^{(l)} \approx l * \rho \qquad (3.2.2)$$

Чтобы найти $\rho_q^{(l)}$ для произвольного q воспользуемся следующим приемом. Пусть мы обнаружили l блоков, среди которых есть k поврежденных. Вероятность обнаружить такой набор блоков, выбрав его случайным образом равна $\rho_k^{(l)}$. Отбросим k поврежденных

блоков. Среди оставшихся $l-k$ блоков с вероятностью $\rho_m^{(l-k)}$ мы сможем найти еще m поврежденных блоков. Это означает, что для любых k и m таких, что $k+m \le l$ выполняется соотношение:

$$\rho_{k+m}^{(l)} = \rho_k^{(l)} * \rho_m^{(l-k)} \qquad (3.2.3)$$

В частности $\rho_q^{(l)} = \rho_1^{(l)} * \rho_{q-1}^{(l-1)}$. Рекурсивно применяя это соотношение, получаем:

$$\rho_q^{(l)} = \rho_1^{(l)} * \rho_1^{(l-1)} * \dots * \rho_1^{(l-q+1)}$$

Или с учетом (3.2.2):

$$\rho_q^{(l)} \approx \frac{l!}{(l-q)!}\,\rho^q \qquad (3.2.4)$$

Перейдем теперь к рассмотрению эволюции чанков, вызванной дисковыми отказами. Следуя подходу, развитому в главе 2.3.3, рассмотрим набор уровней деградации соответственно количеству потерянных блоков, для каждого из которых мы будем следить за средним количеством чанков P_i и вероятностью F_i обнаружить это количество чанков отличным от нуля.

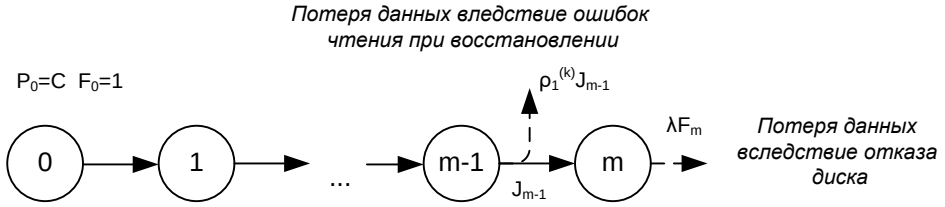

Рисунок 21. Граф состояний, учитывающий потерю данных вследствие ошибок чтения при восстановлении.

Пусть $m = n-k$ – индекс последнего (критического) уровня деградации, J_{m-1} – поток чанков с уровня $m-1$ на уровень m вследствие дисковых отказов (т.е. количество переходов в единицу времени). Если среди этих чанков обнаружится чанк со скрытой ошибкой, его восстановление будет невозможно. Таких чанков в единицу времени $p'_F = \rho_1^{(k)} * J_{m-1}$. Поскольку поток отказов J_{m-1} компенсируется потоком восстановления φF_m, то $p'_F = \rho_1^{(k)}\varphi F_m$, где $\varphi = \frac{N}{k+1}\frac{1}{T_R}$ – суммарная скорость восстановления в кластере из N дисков. Сравним p'_F со скоростью появления ошибок вследствие дисковых отказов p_F (2.3.2):

$$\frac{p'_F}{p_F} = \frac{\rho_1^{(k)}\varphi F_m}{\lambda F_m} = \rho_1^{(k)}\frac{\varphi}{\lambda} \qquad (3.2.5)$$

Здесь $\lambda = 1/T_1$ – вероятность дисковых отказов в единицу времени, T_1 – среднее время до отказа диска в кластере, T_R – время восстановления одного чанка. Таким образом, из (3.2.1), (3.2.2), (3.2.5) получаем:

$$\frac{p'_F}{p_F} = \frac{k}{k+1} * \frac{T_S}{T_b} \frac{NT_1}{T_R}$$

Заметим, что $NT_1 = T_d$ – среднее время наработки до отказа конкретного диска. Пусть B – количество использованных блоков на диске. Тогда среднее время до возникновения ошибки чтения любого из них равно $T_B = T_b/B$. Отсюда:

$$\frac{p'_F}{p_F} = \frac{k}{k+1} * \frac{T_S * T_d}{B * T_B * T_R}$$

Заметим, что $\frac{T_S}{B} = \tau$ – время проверки одного дискового блока. То есть:

$$\frac{p'_F}{p_F} = \frac{k}{k+1} * \frac{T_d}{T_B} * \frac{\tau}{T_R} \qquad (3.2.6)$$

В действительности скорость потери данных вследствие ошибок чтения при восстановлении всегда больше, чем p'_F, поскольку наш анализ не учитывал возможность появления нескольких ошибок чтения одновременно. В общем случае для суммарной скорости потери данных можно написать следующее выражение:

$$\tilde{p}_F = p_F + \beta + \sum_{i=1}^{m} \gamma_i \qquad (3.2.7)$$

Здесь p_F – скорость потери данных вследствие дисковых отказов, β – скорость потери данных вследствие накопления скрытых ошибок чтения блоков, $\gamma_i = \rho_{m-i+1}^{(n-i)} * J_{i-1}$ – скорость появления необратимо поврежденных блоков при переходе с уровня деградации $i-1$ на уровень i вследствие обнаружения в оставшихся $n-i$ блоках $m-i+1$ скрытых ошибок (так что в сумме оказывается потеряно $m+1$ блоков). Иначе говоря, γ_i – это скорость потери данных вследствие скрытых ошибок при восстановлении i утраченных блоков. Таким образом, p'_F из (3.2.6) есть γ_m в формуле (3.2.7). Отсюда следует, что $\tilde{p}_F > p_F + p'_F$, то есть

$$\tilde{p}_F > p_F \left(1 + \frac{k}{k+1} * \frac{T_d}{T_B} * \frac{\tau}{T_R}\right)$$

что и требовалось доказать.

Найдем выражение для суммарной скорости потери данных из (3.2.7). С учетом (3.2.1), (3.2.4) имеем:

$$\gamma_i = \frac{(n-i)!}{(k-1)!} \rho^{m-i+1} * J_{i-1} = \frac{(n-i)!}{(k-1)!} \left(\frac{T_S}{T_b}\right)^{m-i+1} * P_{i-1} * \frac{n-i+1}{NT_1} = \frac{(n-i+1)!}{(k-1)!} \left(\frac{T_S}{T_b}\right)^{m-i+1} * \frac{P_{i-1}}{T_d} \qquad (3.2.8)$$

Где мы по традиции обозначили P_{i-1} – количество чанков, у которых потеряно $i-1$ блоков вследствие дисковых отказов.

Чтобы найти β, заметим, что в единицу времени появляется Cn/T_b новых скрытых ошибок, где C- общее количество чанков. Данные будет невозможно восстановить, если в

одном чанке с поврежденным блоком уже есть критическое количество скрытых ошибок, равное избыточности $m = n - k$, вероятность этого события равна $\rho_m^{(n-1)}$. Отсюда:

$$\beta = \rho_m^{(n-1)} * \frac{Cn}{T_b} = \frac{n!}{(k-1)!} * \left(\frac{T_S}{T_b}\right)^m * \frac{C}{T_b} \qquad (3.2.9)$$

В следующей таблице приведены результаты применения формулы (3.2.7) с учетом (3.2.8), (3.2.9), а также результатов из Таблица 2 для различных схем хранения данных.

Таблица 10. Суммарная скорость потери данных с учетом скрытых ошибок.

Схема хранения (n, k)	*Суммарная скорость потери данных* $\tilde{p}_F$
(n, n)	$\frac{Cn}{T_b} + \frac{N}{T_d}$
$(n, n-1)$	$Cn(n-1)\frac{T_S}{T_b}\left(\frac{1}{T_b} + \frac{1}{T_d}\right) + Cn^2\frac{T_R}{T_d^2}$
$(n, n-2)$	$Cn(n-1)(n-2)\frac{T_S^2}{T_b^2}\left(\frac{1}{T_b} + \frac{1}{T_d}\right) + \frac{C^2}{2N^2}n^2(n-1)^2\left[(n-2)\frac{T_S}{T_b} + (n-1)\frac{T_R}{T_d}\right]\frac{T_R}{T_d^2}$
$(n, n-3)$	$Cn(n-1)(n-2)(n-3)\frac{T_S^3}{T_b^3}\left(\frac{1}{T_b} + \frac{1}{T_d}\right) + \frac{C^2}{2N^2}n^2(n-1)(n-2)^2(n-3)\frac{T_S^2}{T_b^2}\frac{T_R}{T_d^2} + \frac{C^3}{6N^5}n^3(n-1)^2(n-2)^3\left[(n-3)\frac{T_S}{T_b} + (n-2)\frac{T_R}{T_d}\right]\frac{T_R^2}{T_d^3}$

Чтобы проверить полученные результаты, мы добавили учет скрытых ошибок в имитационную модель A и смоделировали скорость потери данных для хранилища из $N = 50$ дисков с $C = 2500$ чанками со схемой хранения $(5,3)$ для случая, когда скраббинг происходит с максимально возможной скоростью, так что $\tau = T_R$, то есть когда проверка блока данных занимает столько же времени, сколько восстановление утраченного блока. При этом мы предположили для определенности, что среднее время до возникновения ошибки чтения любого блока на диске $T_B = 10^5 T_R$. Результат в зависимости от

интенсивности дисковых отказов показан на рисунке ниже. График построен в безразмерных координатах. По горизонтальной оси отложено отношение среднего времени жизни диска до его отказа T_d к времени восстановления утраченного блока T_R. По вертикальной оси – отношение среднего время наработки до потери данных $T_F = 1/\tilde{p}_F$ к времени восстановления блока T_R. Сплошной линией показан результат расчета в соответствии с Таблица 10. Точками показаны результаты моделирования. Каждая точка получена усреднением результатов 200 симуляций. Пунктиром для сравнения показаны результаты, рассчитанные при рассмотрении только дисковых отказов или только скрытых ошибок. Как видно из рисунка, наша математическая модель демонстрирует отличное соответствие результатам моделирования.

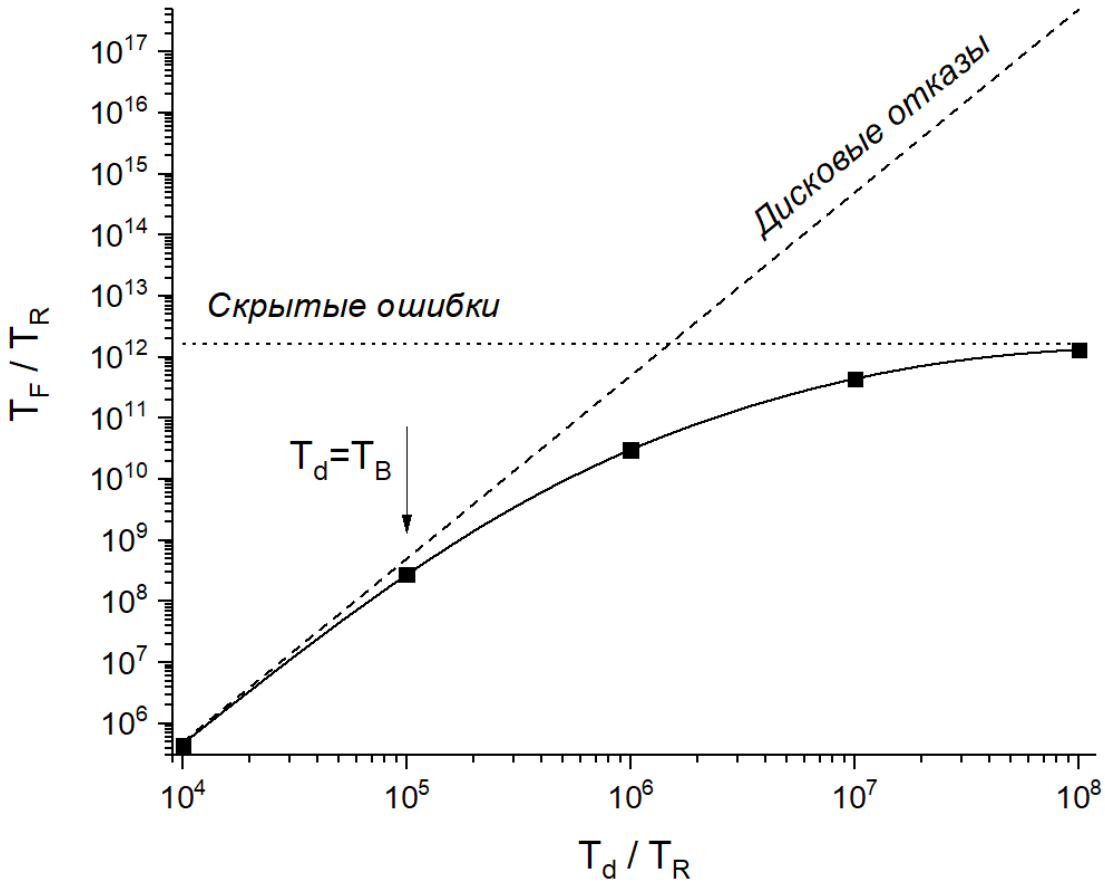

Рисунок 22. Среднее время до потери данных при наличии скрытых ошибок по результатам моделирования (символы) в сравнении с предсказанным математической моделью (сплошная линия).

Стрелкой на рисунке обозначена абсцисса, соответствующая интенсивности дисковых отказов, когда среднее время до отказа конкретного диска оказывается равным среднему времени до отказа любого дискового блока на нем. Суммарная скорость потери

данных при этом составляет 1.825 от скорости потери данных вследствие одних только дисковых отказов. Этот результат полностью соответствует доказанной выше теореме которая дает минимальное значение этого отношения, равное 1.75 для $k = 3$.

Интересно, что скорость потери данных вследствие накопления скрытых ошибок, представленная членом β в формуле (3.2.7), составляет при этом всего $3*10^{-4}$ от скорости потери данных вследствие дисковых отказов. Это говорит о том, что в формуле (3.2.7) доминирующим является последний член, соответствующий ошибкам при восстановлении утраченных блоков вследствие наличия скрытых ошибок в оставшихся блоках. Чтобы проверить это, мы построили график доли различных механизмов потери данных в суммарной интенсивности отказов $\tilde{p}_F$ из (3.2.7). Результат, вычисленный в соответствии с Таблица 10, показан на следующем рисунке.

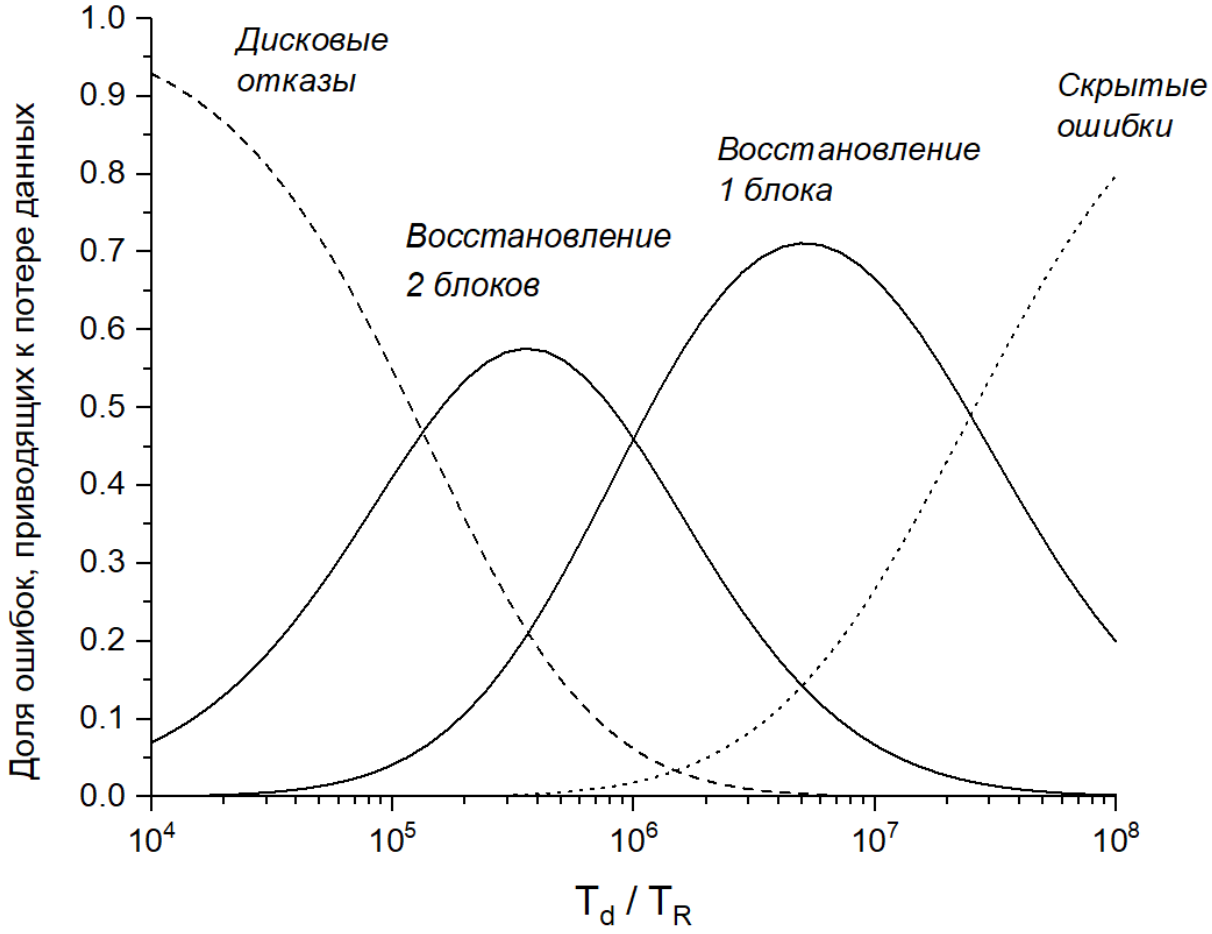

Рисунок 23. Относительный вклад различных механизмов потери данных при наличии скрытых ошибок в зависимости от интенсивности дисковых отказов.

Штриховой линией показан вклад дисковых отказов в суммарную интенсивность потери данных $p_F/\tilde{p}_F$, пунктиром – вклад накопления скрытых ошибок чтения блоков $\beta/\tilde{p}_F$, сплошными линиями – вклад ошибок при восстановлении одного и двух утраченных

блоков - $\gamma_1/\tilde{p}_F$ и $\gamma_2/\tilde{p}_F$, возникающих вследствие наличия скрытых ошибок в оставшихся блоках. В соответствии с Таблица 10:

$$\beta = Cn(n-1)(n-2)\frac{T_S^2}{T_b^3}$$

$$\gamma_1 = Cn(n-1)(n-2)\frac{T_S^2}{T_b^2 T_d}$$

$$\gamma_2 = \frac{C^2}{2N^2}n^2(n-1)^2(n-2)\frac{T_S T_R}{T_b T_d^2}$$

$$p_F = \frac{C^2}{2N^2}n^2(n-1)^3\frac{T_R^2}{T_d^3}$$

Как видно из рисунка, каждый из 4-х механизмов потери данных, соответствующих 4-м слагаемым в (3.2.7), доминирует в определенном диапазоне интенсивности дисковых отказов, так что суммарная интенсивность потери данных оказывается значительно больше, чем результат простой суперпозиции потери данных вследствие одних только дисковых отказов и скрытых ошибок.

Рассмотрим, как суммарная скорость потери данных зависит от скорости скраббинга. Следующий график показывает, как меняется среднее время до потери данных в зависимости от скорости появления скрытых ошибок и скорости скраббинга.

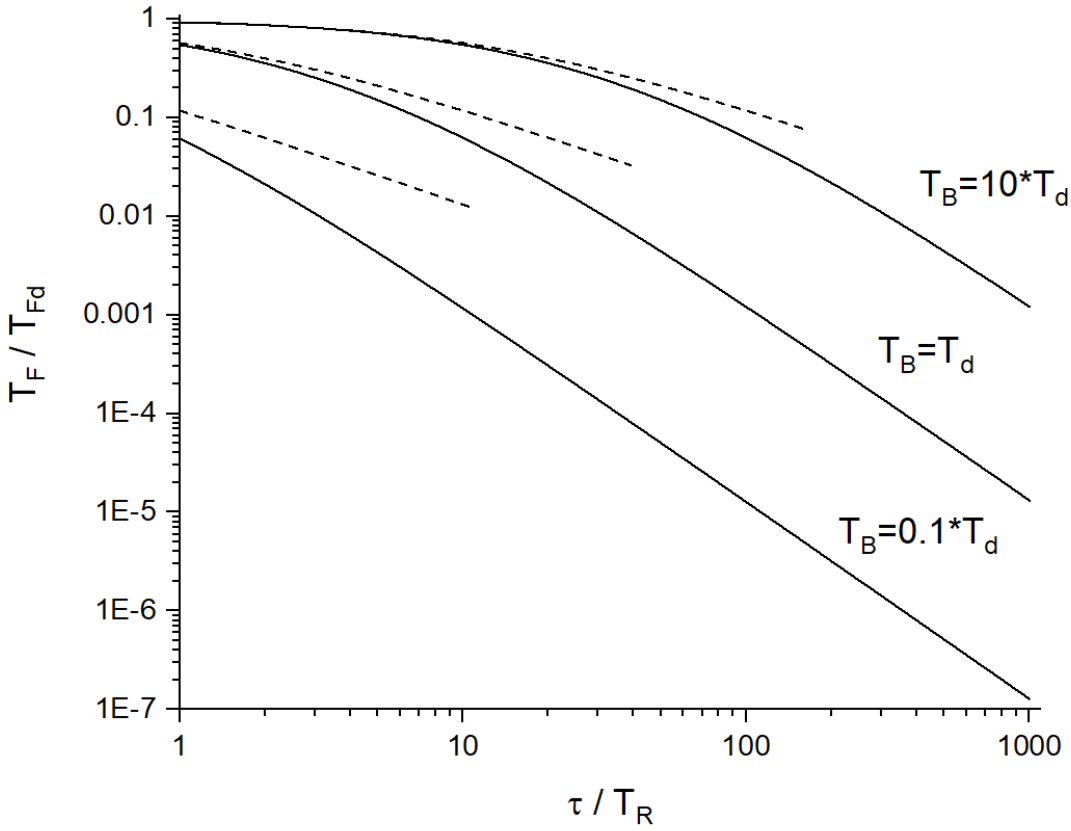

Рисунок 24. Ухудшение надежности хранилища за счет скрытых ошибок как функция отношения времени τ, которое тратится на проверку блока, к времени его восстановления.

По горизонтальной оси отложено отношение времени τ, затраченного на проверку одного блока, к времени его восстановления T_R. По вертикальной оси отложено отношение результирующего среднего времени до потери данных $T_F = 1/\tilde{p}_F$ к среднему времени до потери данных $T_{Fd} = 1/p_F$, вычисленному с учетом только дисковых отказов. Это отношение показывает, насколько скрытые ошибки ухудшают надежность хранилища данных. Сплошной линией показаны зависимости, рассчитанные для среднего времени до возникновения скрытой ошибки в любом дисковом блоке равного $10T_d$, T_d и $\frac{1}{10}T_d$, где T_d – среднее время до отказа конкретного диска, которое было выбрано равным $T_d = 10^5 T_R$. Пунктирной линией показана верхняя граница T_F/T_{Fd}, рассчитанная в соответствии с теоремой, доказанной выше, то есть $\left(1 + \frac{k}{k+1} * \frac{T_d}{T_B} * \frac{\tau}{T_R}\right)^{-1}$. Как видно из графика, при увеличении τ среднее время до потери данных падает быстрее, чем следует из доказанной нами теоремы, вследствие доминирования членов β и γ_1 в сумме (3.2.7).

Чтобы продемонстрировать зависимость $\tilde{p}_F$ от скорости скраббинга более наглядно, перепишем выражение для $\tilde{p}_F$ в схеме хранения $(n, n-2)$ из Таблица 10 в виде квадратного трехчлена по параметру τ/T_R:

$$\frac{\tilde{p}_F}{p_F} = 1 + \frac{n-2}{n-1} * \frac{T_d}{T_B} * \frac{\tau}{T_R} + \frac{2N(n-2)}{B(n-1)^2} * \left(1 + \frac{T_d}{BT_B}\right) * \left(\frac{T_d}{T_B}\right)^2 * \left(\frac{\tau}{T_R}\right)^2 \qquad (3.2.10)$$

Здесь мы обозначили B – число блоков на диске, $T_B = T_b/B$ - среднее время до возникновения скрытой ошибки в любом дисковом блоке, τ – время, необходимое для проверки одного блока. Первые два члена этого выражения дают оценку снизу для $\tilde{p}_F$ из теоремы выше для частного случая $k = n - 2$. Из полученного выражения видно, что квадратичный по параметру τ/T_R член растет с увеличением количества дисков N, то есть наличие скрытых ошибок ограничивает масштабируемость хранилища. Следующий рисунок иллюстрирует эту зависимость.

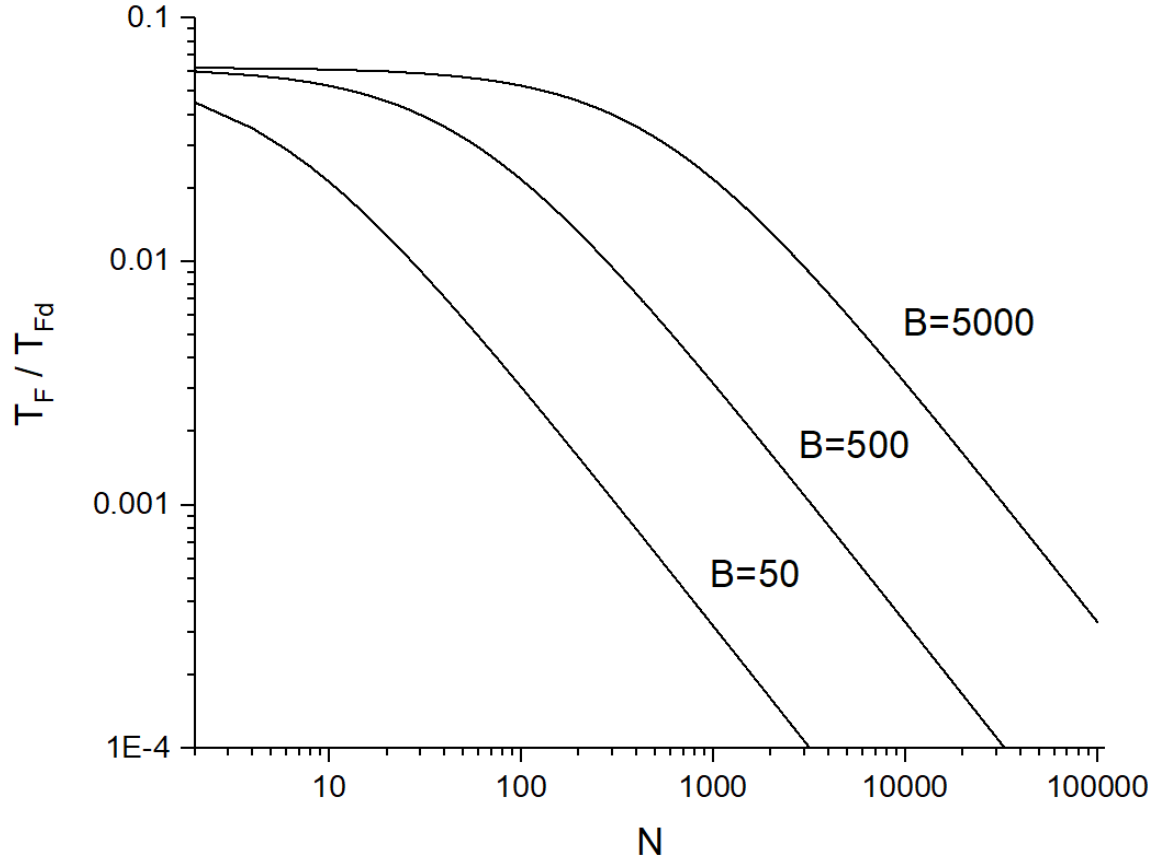

Рисунок 25. Падение надежности хранилища с увеличением количества дисков N вследствие наличия скрытых ошибок в зависимости от количества блоков B, на которое поделен каждый диск.

По горизонтальной оси отложено количество дисков в хранилище, по вертикальной – отношение результирующего среднего времени до потери данных $T_F = 1/\tilde{p}_F$ к среднему времени до потери данных $T_{Fd} = 1/p_F$, вычисленному с учетом только дисковых отказов.

Среднее время до отказа диска было выбрано равным $T_d = 10^5 T_R$, среднее время до появления скрытой ошибки на диске мы положили равным $T_B = T_d/2$, время, необходимое для проверки одного блока $\tau = 10T_R$. На графике показаны зависимости, вычисленные по формуле (3.2.10) для различного числа блоков на диске B. Как видно из графика, начиная с некоторого порогового значения N' наблюдается обратно-пропорциональная зависимость среднего времени до потери данных от N. Это пороговое значение нетрудно найти из (3.2.10), положив последний член равным сумме предшествующих членов (при этом T_F равно половине от асимптоты при $N \to 0$). Таким образом, получаем уравнение:

$$\frac{2N'(n-2)}{B(n-1)^2} * \left(1 + \frac{T_d}{BT_B}\right) * \left(\frac{T_d}{T_B}\right)^2 * \left(\frac{\tau}{T_R}\right)^2 = 1 + \frac{n-2}{n-1} * \frac{T_d}{T_B} * \frac{\tau}{T_R}$$

Отсюда:

$$N' = \frac{B(n-1)^2}{2(n-2)}\left(1 + \frac{n-2}{n-1} * \frac{T_d}{T_B} * \frac{\tau}{T_R}\right)\left(1 + \frac{T_d}{BT_B}\right)^{-1} * \left(\frac{T_d}{T_B}\right)^{-2} * \left(\frac{\tau}{T_R}\right)^{-2} \quad (3.2.11)$$

Для выбранных нами параметров $N' \approx B/10$. Как видно из (3.2.10), увеличивая количество блоков, мы уменьшаем квадратичный член по параметру τ/T_R, что положительно сказывается на надежности хранилища и его масштабируемости. Природа этой зависимости очевидна – чем меньше блок, тем меньше вероятность появления скрытой ошибки в нем. На практике, однако, произвольное уменьшение размера блока оказывается нежелательным, поскольку суммарный размер связанных с блоками метаданных растет при этом обратно пропорционально размеру блока. Альтернативный подход, лишенный этого недостатка, заключается в разбиении блоков на более мелкие фрагменты исключительно на стадии восстановления, как показано на рисунке ниже.

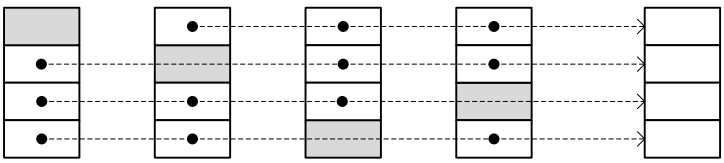

Рисунок 26. Восстановление с разбиением блоков на более мелкие фрагменты, чтобы уменьшить вероятность повреждения каждого из них.

Предположим, у чанка с $k = 3$ сохранились 4 блока, но все они содержат по одной скрытой ошибке. Если разбить блоки на фрагменты так, чтобы в каждом кортеже из фрагментов было не более одного поврежденного фрагмента (показаны серым цветом), то восстановление тем не менее оказывается возможным.

3.2.3. Расчет среднего времени наработки до отказа для реального хранилища с учетом скрытых ошибок

Рассчитаем среднее время до потери данных для реального хранилища, состоящего из $N = 1000$ дисков емкостью 10Tb каждый с производительностью операций чтения/записи 100Mb/сек. Для определенности рассмотрим схему хранения $(6, 4)$, где данные разбиты на фрагменты размером 256Mb. Будем считать, что вероятность скрытого повреждения диска равна вероятности его полного отказа. Результаты, вычисленные по формулам из Таблица 10, для различных значений среднего времени до отказа одного диска T_d и скорости скраббинга сведены в следующей таблице.

Таблица 11. Расчетное среднее время наработки до потери данных с учетом скрытых ошибок для различных комбинаций параметров СХД.

Среднее время наработки до отказа диска	*1 год*	*2 года*	*5 лет*	*10 лет*
Без учета скрытых ошибок	50 лет	400 лет	$6 * 10^3$ лет	$5 * 10^4$ лет
Скраббинг 50Mb/сек	20 лет	150 лет	$2 * 10^3$ лет	$2 * 10^4$ лет
Скраббинг 10Mb/сек	5 лет	40 лет	600 лет	$5 * 10^3$ лет
Скраббинг 5Mb/сек	2.5 года	20 лет	300 лет	$2.5 * 10^3$ лет
Скраббинг 1Mb/сек	4 месяца	2.5 года	40 лет	300 лет

Как видно из Таблица 11, результирующая надежность хранилища быстро падает с уменьшением скорости скраббинга. Уже при скорости скраббинга в $1/10$ от полной производительности диска надежность хранилища падает в 10 раз по сравнению с надежностью в отсутствие скрытых повреждений. Это означает, что для обеспечения приемлимой надежности необходимо использовать максимальную скорость скраббинга, которая еще не приводит к заметному ухудшению производительности хранилища с точки зрения пользователя. Кроме того, важно выбрать схему хранения данных так, тобы среднее время наработки на отказ без учета скрытых повреждений не менее чем на порядок превышало требуемое минимальное значение.

3.2.4. Оптимизация надежности с учетом скрытых ошибок

Таким образом, наличие скрытых ошибок существенно ухудшает надежность хранилища данных и его масштабируемость, а также требует принятия специальных мер для их обнаружения – скраббинга. При этом производительность диска, накладывая ограничения на скорость скраббинга, не позволяет нам нивелировать негативный эффект от наличия скрытых ошибок, так что в большинстве реальных систем хранения данных они будут являться основным источником потери данных. Оптимизация надежности хранилища при этом требует принятия целого комплекса мер, например:

1. Выбрать скорость скраббинга как максимальную, не приводящую к заметному падению производительности кластера для пользователя. При этом скраббинг может иметь более низкий приоритет, чем дисковые операции, инициированные пользователем.
2. Оперативно заменять диски, на которых обнаружены скрытые ошибки, поскольку практика показывает, что вероятность повторного появления скрытых ошибок на таких дисках становится существенно выше (см. [22]).
3. Разбивать дисковые блоки на более мелкие фрагменты при восстановлении, чтобы уменьшить вероятность ошибки при восстановлении, когда число доступных блоков больше критического $(n - k)$.
4. Использовать оптимизированные стратегии скраббинга, например, использующие свойство локализации ошибок. При этом диск делится на фрагменты размером несколько мегабайт, а они объединяются в локальные группы. Первыми проверяются первые фрагменты во всех группах, затем следующие за ними и т.д. Такая стратегия позволяет уменьшить время обнаружения области скрытых ошибок, если ее размер превышает размер фрагмента (см. [26]).
5. Мониторить S.M.A.R.T. статистику диска, чтобы обнаружить симптомы появления скрытых ошибок до того, как они приведут к реальным ошибкам чтения данных (см. [23, 25]).
6. Использовать хранение с избыточностью на уровне диска, чтобы обеспечить восстановление поврежденных блоков без обращения к другим дискам (см. [24]).

3.3. Учет особенностей аппаратной инфраструктуры – области отказов

Отказы оборудования не всегда сводятся к одним только дисковым отказам. Например, у сервера может отказать система питания или материнская плата, что приведет к недоступности всех подключенных к нему дисков. Стойка, на которой расположен сервер, может полностью отключиться вследствие аварии в системе питания. Аварии в системе питания или охлаждения могут привести к отключению и целого ряда стоек, а пожар в машинном зале центра обработки данных (ЦОД) может привести к отключению или повреждению находящихся в нем серверов. Учет подобного рода факторов при распределении блоков данных по дискам может существенно улучшить надежность СХД.

Будем называть областью отказов совокупность оборудования, подверженную одновременному отказу вследствие некоторых катастрофических событий. Для простоты мы выделим всего 5 возможных типов областей отказа:

- Диск
- Сервер
- Стойка
- Ряд стоек
- Машинный зал

Следующий рисунок иллюстрирует области отказа на примере типичного машинного зала ЦОД.

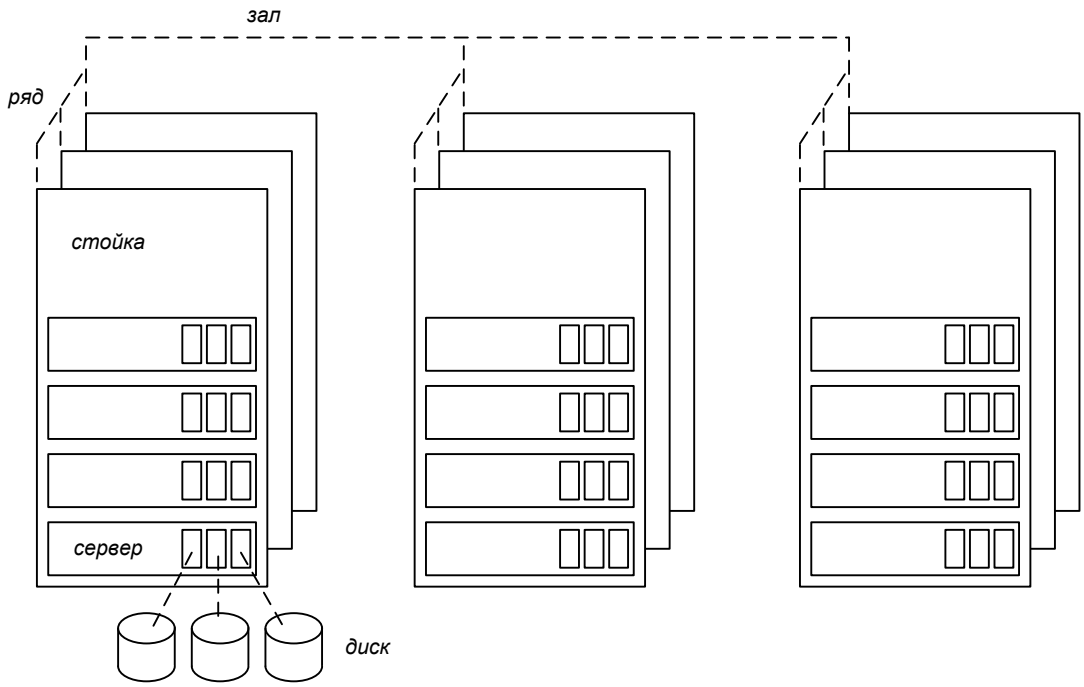

Рисунок 27. Иерархия областей отказа на примере типичного машинного зала ЦОД.

Как видно из рисунка, области отказа образуют иерархию, где области отказа нижнего уровня (например, диски) являются частью областей отказа более высокого уровня (например, сервера). На каждом уровне иерархии области отказов образуют некоторое конечное множество (дисков, серверов, стоек и т.д.). Каждое такое множество областей отказов мы будем называть доменом отказов (следуя принятой для систем баз данных терминологии – см. [9, с.152]). Мы будем рассматривать 5 различных доменов отказов в соответствии с типами областей отказа, перечисленными выше.

Дисковый кортеж *устойчив к отказам* домена D, если все входящие в него диски принадлежат различным областям отказов из домена D. Это означает, что выход из строя любой области отказа домена D приведет к отказу не более одного диска в таком дисковом кортеже. Любой дисковый кортеж устойчив к отказам дискового домена. Дисковый кортеж, куда входят диски из различных серверов, устойчив к отказам серверного (а также и дискового) домена и т.д. Максимальный уровень домена отказов, к которым устойчив дисковый кортеж, мы будем для краткости называть просто его *доменом отказов*. Чем выше уровень домена D в иерархии, тем большую надежность обеспечивает дисковый кортеж, устойчивый к отказам из D, при равной избыточности схемы хранения. Следовательно, домен отказов является важным параметром при распределении блоков данных по дисковым кортежам.

С учетом вышеизложенного можно сделать вывод о том, что дисковые кортежи необходимо создавать так, чтобы они были устойчивы к отказам домена самого верхнего уровня иерархии. Однако на практике приходится учитывать и другие факторы. Во-первых, создание дискового кортежа длины n, устойчивого к отказам домена D, может оказаться невозможным, если количество областей отказа в домене D меньше, чем n. Например, создание кортежа длины n, устойчивого к отказам стойки, будет невозможно, если количество стоек в системе меньше n. Вторым фактором, который необходимо учитывать при выборе домена отказов, является сетевая топология. Если все дисковые кортежи устойчивы к отказам домена самого верхнего уровня из возможных, то в случае отказа весь поток данных, необходимых для восстановления после потери дисковых блоков, пройдет через самый верхний уровень сетевой топологии. Если на этом уровне окажется единственный маршрутизатор, как показано на рисунке ниже, его пропускная способность может оказаться фактором, ограничивающим скорость восстановления всей системы. Негативное влияние этого ограничения скорости восстановления с большой вероятностью перевесит эффект от устойчивости к отказам домена самого верхнего уровня, вероятность которых крайне мала.

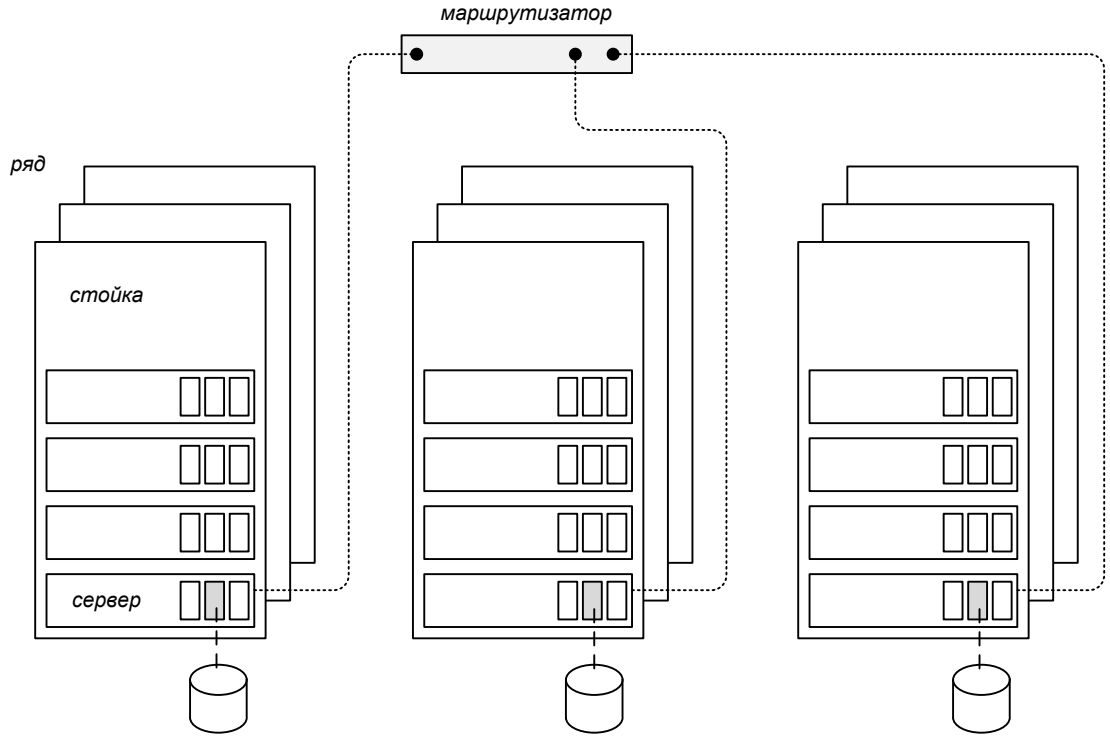

Рисунок 28. Дисковый кортеж, устойчивый к отказам домена самого верхнего уровня, оказывается связан через самый верхний уровень сетевой топологии.

Таким образом, при создании дисковых кортежей не следует ориентироваться ни на домен отказов самого нижнего – дискового – уровня (поскольку отказ сервера является достаточно частым событием), ни на домен самого верхнего уровня (поскольку рост загруженности сети может нивелировать все преимущества в устойчивости к катастрофическим событиям). Оптимальной является по-видимому возможность гибкой настройки этого параметра (также, как и схемы хранения данных) клиентом на уровне отдельных файлов и/или папок на файловой системе в зависимости от его потребностей в надежности хранения данных. Максимально надежным является хранение в репликах (схема $(n, 1)$), устойчивых к отказам в домене самого верхнего уровня. Однако, такая схема неэффективна как с точки зрения использования дискового пространства, так и с точки зрения нагрузки на сетевую инфраструктуру. Оптимальным по соотношению надежности и накладных расходов способом хранения данных является (n, k) схема с серверным доменом отказов. Схемы хранения с дисковым доменом отказов имеет смысл использовать только для данных, целостность и доступность которых не являются критичными для клиента, например, логов.

4. Внедрение результатов исследования надежности СХД

Результаты данного исследования надежности СХД нашли непосредственное применение при создании кластерного хранилища данных Acronis Storage, реализованного автором в составе коллектива разработчиков. В данной главе мы перечислим основные требования к созданной СХД, изучим лежащую в ее основе архитектуру, особенности ее реализации, рассмотрим достигнутые в ходе внедрения результаты и направления дальнейшего развития системы.

4.1. Основные требования к СХД

При создании СХД Acronis Storage ставилась задача удовлетворить потребности в хранении данных для широкого спектра приложений, таких как

- облачные сервисы, предоставляющие доступ к виртуальным машинам и контейнерам;
- сервера виртуальных машин предприятий;
- отказоустойчивые файловые хранилища предприятий;
- отказоустойчивые хранилища систем резервного копирования.

Для этого СХД должна удовлетворять следующим основным требованиям:

6. Масштабируемость. Емкость хранилища должна легко наращиваться путем добавления в систему новых дисков и серверов. При этом не должно происходить ухудшения параметров производительности и надежности при увеличении размеров системы (до определенного предела).
7. Отказоустойчивость (высокодоступность). Система должна сохранять работоспособность при выходе из строя отдельных компонент, поскольку с ростом числа компонент вероятность таких событий растет, и они перестают быть исключительными. Высокодоступность подразумевает, что система не просто устойчива к отказам, но и продолжает предоставлять доступ для клиентов в случае отказа отдельных компонент.
8. Надежность (консистентность). Данные, записанные клиентом не должны меняться самопроизвольно или в результате отказов компонент системы. Если данные недоступны, клиент должен либо ожидать их доступности, либо получить соответствующее ситуации сообщение об ошибке.
9. Способность к самовосстановлению. После отказа компоненты (например, диска) система должна автоматически (без вмешательства оператора) провести необходимые процедуры восстановления целостности данных, чтобы отказ еще одной компоненты не привел к катастрофической потере данных.
10. Универсальность. Система должна предоставлять разнообразные способы доступа к данным для клиентов – файловый доступ, доступ по сетевым протоколам удаленного доступа типа NFS, iSCSI, а также доступ по протоколу объектного хранилища (Amazon S3).

4.2. Архитектура созданной СХД

4.2.1. Базовые принципы построения

Требования к вместительности и надежности современных СХД выходят далеко за рамки возможностей отдельного диска. Удовлетворить им можно только создав систему, объединяющую емкости сотен дисков и при этом гарантирующую надежность, существенно превышающую надежность каждого из них. Исторически первыми системами подобного рода были RAID-массивы [7,8]. Они обеспечивали избыточность хранения либо за счет сохранения данных одновременно на два диска (RAID1), либо за счет хранения контрольной суммы на дополнительном диске (RAID5), либо двух независимых контрольных сумм на двух дополнительных дисках (RAID6). Последний вариант широко используется и поныне, поскольку позволяет гибко варьировать накладные расходы меняя количество дисков, где хранятся данные, и при этом гарантирует сохранность данных при выходе из строя одного или двух произвольных дисков из этого набора. Однако RAID-массивы не удовлетворяют большинству требований, перечисленных в 4.1 по следующим причинам:

- Восстановление после отказа диска требует вмешательства оператора для замены диска и активации процедуры восстановления.
- Процедура восстановления требует воссоздания содержимого целого диска, что занимает значительное время, в течение которого использование системы клиентами невозможно.
- Расширение системы путем простого добавления новых дисков невозможно.

На практике несколько отдельных RAID6-массивов иногда объединяют для построения одного или нескольких комбинированных томов, чтобы получить доступный объем хранилища, выходящий за рамки возможностей одного RAID-массива, но, как было отмечено в разделе 2.3.4, это приводит к падению показателей надежности, обратно пропорциональному количеству дисков в такой конфигурации.

Как было показано в разделе 2.3.4, разбиение файла на фрагменты фиксированной длины (чанки) с последующим распределением их между дисками в системе позволяет достичь гораздо лучших показателей надежности, а главное, удовлетворить требованию масштабируемости, то есть обеспечить как минимум сохранение показателей надежности при увеличении количества дисков в системе при условии, что схема хранения имеет избыточность не менее 2. Первой промышленной системой, построенной на этом принципе, была кластерная файловая система корпорации Google [28], предназначенная для

внутреннего использования специализированными приложениями корпорации Google. В отличие от нее СХД Acronis Storage изначально разрабатывалась как универсальная кластерная файловая система, которая может быть использована любыми приложениями без их адаптации.

4.2.2. Основные компоненты системы

Для доступа к дисковым блокам в распределенной системе необходим специальный сервис, предоставляющий сетевой интерфейс для этого. В нашей системе этим сервисом является сервис чанков или CS (chunk service). Каждый диск в системе связан с единственным CS, который имеет монопольный доступ к этому диску и предоставляет сетевой доступ к нему для клиентов. Каждый чанк в системе представлен некоторым дисковым кортежем, т.е. набором дисковых блоков, хранящихся на некотором наборе CS[6]. Для того, чтобы клиенты могли обратиться к этим CS, где-то должна храниться информация о соответствии между файлами, чанками и CS, на которых хранятся соответствующие дисковые блоки. Эта информация, очевидно, не может храниться на самом CS. Для ее хранения используется отдельный сервис метаданных или MDS (Meta Data Service). Он же используется и для хранения дерева имен файловой системы, а также для координации доступа к файлам и восстановления после сбоев. Все CS периодически посылают MDS регистрационные сообщения, так что MDS имеет полную информацию о том, какие CS находятся в рабочем состоянии, и знает их сетевые адреса. Эту информацию он предоставляет клиентам с тем, чтобы они могли обратиться к конкретным CS для доступа к данным конкретного файла. Наконец, третий компонент системы – *агент*, предоставляющий пользователям доступ к файловой системе посредством монтирования ее дерева имен в локальную файловую систему пользователя. Таких агентов в кластере может быть произвольное количество – как минимум по одному на каждый компьютер, имеющий доступ к СХД. Следующий рисунок иллюстрирует основные компоненты системы и их взаимодействие.

[6] Точнее на дисках, обслуживаемых этими CS. Для простоты в дальнейшем мы не будем делать различий между диском и CS, который его обслуживает.

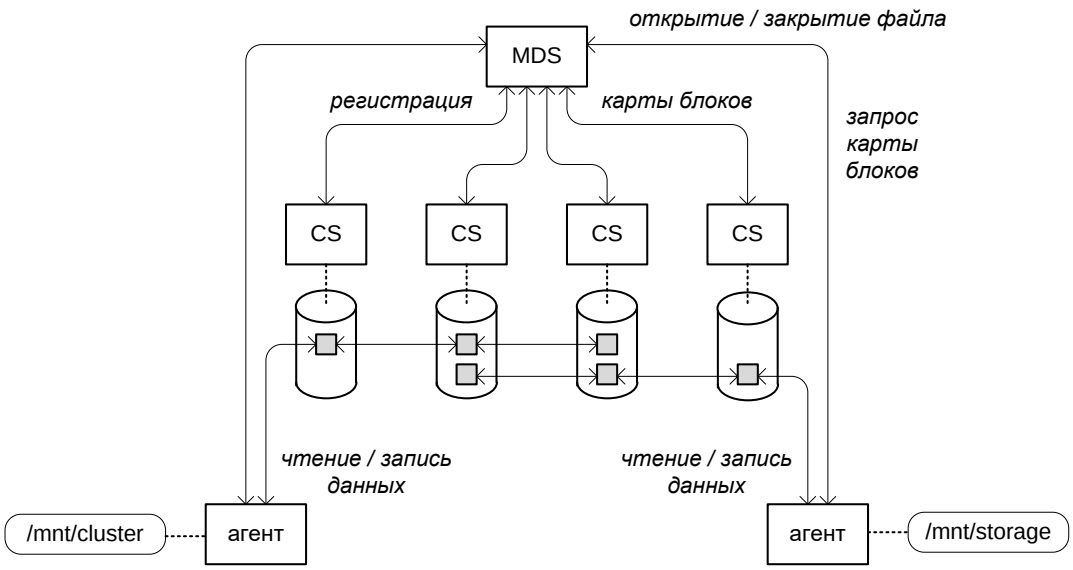

Рисунок 29. Основные компоненты СХД Acronis Storage.

Прежде, чем получить доступ к хранилищу, клиент запускает специальный сервис, который мы будем называть *агентом*. После этого все содержимое хранилища становится доступно клиенту в виде содержимого папки в его локальной файловой системе. Эта процедура называется *монтированием*. При этом сама папка, куда смонтирована файловая система хранилища, может быть различной у различных клиентов. Более того, клиент при желании может, запустив несколько агентов, смонтировать хранилище несколько раз в разные папки, например, для лучшей балансировки. При обращении к файлу в хранилище для чтения или записи агент сначала посылает запрос MDS на открытие файла. MDS координирует доступ к файлам со стороны клиентов с тем, чтобы файл мог быть открыт на запись максимум одним клиентом, в то время как на чтение его могут открывать и несколько клиентов одновременно при условии, что он не открыт на запись. В случае успешного открытия файла клиент запрашивает у MDS карту дисковых блоков для чанка, который он собирается читать или писать. Поскольку информацией о распределении дисковых блоков по серверам чанков владеет только MDS, перед тем, как ответить клиенту, он распространяет карту блоков по тем CS, где хранятся эти блоки. Те запоминают ее в оперативной памяти и предоставляют доступ к блокам клиенту. Чтобы исключить несанкционированный доступ к файлу со стороны клиента, который когда-то получил карту блоков, но был лишен доступа к файлу, карта блоков имеет актуальную версию, которая сравнивается с версией карты блоков клиента прежде, чем ему будет разрешен доступ к данным. Имея карту блоков, клиент может обратиться напрямую к серверам чанков для чтения или записи данных.

4.2.3. Обеспечение отказоустойчивости

Каждый CS примерно раз в секунду присылает MDS регистрационное сообщение, которое информирует MDS о том, что CS готов к работе, сообщает его сетевой адрес, а также информацию об общем объеме и текущей заполненности диска. Если CS перестает присылать регистрационные сообщения, MDS ждет некоторое время, после чего начинает процедуру восстановления утраченной избыточности для чанков, имеющих дисковые блоки на этом CS. Для этого он распространяет работающим CS карты блоков для этих чанков, где неработающий CS исключен, а вместо него добавлен новый CS, на котором будет создан новый дисковый блок взамен утраченного. При выборе нового CS учитываются требования к домену отказов (см. главу 3.3) для файла, которому принадлежит дисковый блок. Файлы наследуют домен отказов, как и схему хранения, от директории, в которой они создаются. Директория наследует эти параметры от родительской директории. Кроме того, пользователь может менять эти параметры на директориях и файлах. При условии соблюдения требований к домену отказов новый CS выбирается из числа менее заполненных и максимально близких к другим CS того же кортежа с учетом сетевой топологии, чтобы минимизировать нагрузку на сетевую инфраструктуру (см. 3.3). Новый CS никогда не выбирается из числа тех, которые уже участвуют в восстановлении какого-то дискового блока, чтобы исключить падение скорости доступа к вращающемуся диску вследствие одновременного обращения к разным областям диска. Эта стратегия позволяет производить восстановление утраченных блоков с максимальной скоростью и, соответственно, достичь максимальных показателей надежности. При восстановлении потерянных дисковых блоков первыми восстанавливаются чанки с максимальным количеством утраченных блоков. Как было показано в главе 2.3.7, это позволяет существенно повысить надежность всей системы, поскольку чем больше блоков потеряно, тем выше риск необратимой потери данных при будущих дисковых отказах.

Поскольку процесс восстановления после сбоев CS координируется единственным MDS, его надежность становится критически важным фактором для надежности всего хранилища. А с учетом того, что он координирует и доступ к файлам со стороны клиентов, его постоянная доступность абсолютно необходима для обеспечения постоянной доступности СХД. Столь высокие требования к надежности и доступности MDS требуют специальных технических решений. В настоящее время традиционным подходом к построению подобных отказоустойчивых сервисов является концепция реплицированного конечного автомата. Сервис, отказоустойчивость которого необходимо обеспечить, запускается в нескольких экземплярах на разных серверах. Если сервис не имеет своего

состояния, то его экземпляры могут работать одновременно, обрабатывая клиентские запросы. В случае MDS это очевидно не так. Обрабатывая запросы клиентов, он в общем случае меняет свое состояние, например, создает или удаляет новые файлы или дисковые блоки. Проблема создания подобных отказоустойчивых сервисов была впервые подробно исследована Лесли Лампортом [29, 30] и Фредом Шнайдером [32]. Идея предложенного ими метода состоит в том, чтобы реализовать сервис в виде конечного автомата, который меняет свое состояние в ответ на некоторые события. Предложенный Лесли Лампортом протокол – PAXOS – позволяет гарантировать, что все работоспособные сервисы будут обрабатывать один и тот же поток событий. Соответственно, мы можем гарантировать, что в результате обработки этих событий состояние сервисов будет изменяться синхронно. Наша реализация этой концепции применительно к MDS схематично показана на следующем рисунке.

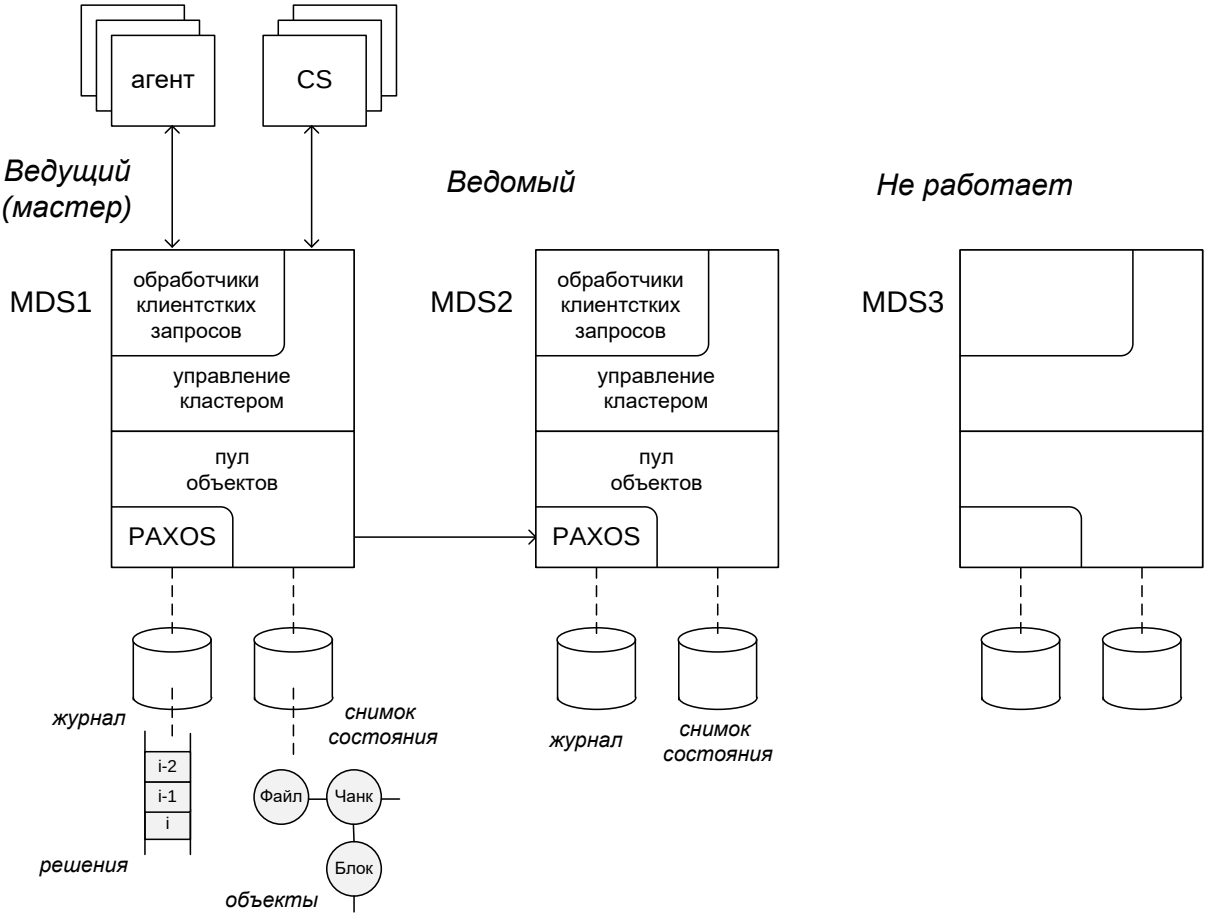

Рисунок 30. Отказоустойчивая архитектура MDS

На рисунке показаны три MDS, которые образуют отказоустойчивый кластер, связанный протоколом PAXOS. Этот протокол обеспечивает так называемый *консенсус*, то есть согласованное принятие некоторого *решения* подверженными отказам процессами. Каждый процесс может предложить свое *решение*, протокол гарантирует, что какое-то решение из числа предложенных в конечном итоге будет принято, о нем узнают все

процессы, и это решение не изменится впоследствии (например, вследствие отключения питания). В нашем случае эти решения и играют роль событий, которые меняют состояние MDS. Таким образом, протокол консенсуса обеспечивает синхронное и согласованное изменение состояния всех работоспособных MDS.

Для работы протокола консенсуса необходим кворум. Кворумом называют такую конфигурацию процессов – участников протокола консенсуса, которая может существовать в не более чем одном экземпляре. В простейшем случае, который и реализован в нашей системе, кворумом является простое большинство от общего числа MDS. То есть, если в системе три MDS, как показано на Рисунок 30, то в кворуме будет не менее двух из них. Значит, система сохранит работоспособность при выходе из строя не более одного MDS. В случае, если в системе 5 MDS, то сохранение работоспособности гарантируется при выходе из строя не более двух из них.

Для достижения консенсуса каждое решение, прежде чем оно будет принято, записывается в *журнал* каждым участником протокола. Журналом называется форма постоянного хранилища записей, обладающая двумя простыми свойствами:

- Если запись с номером i записана в журнал, то из него всегда можно будет прочитать все записи с номерами $j \leq i$.
- Из журнала никогда не читаются записи, которые не были в него записаны.

Как видно, свойства журнала выполняются в случае, если мы просто добавляем записи в конец некоторого списка (например, дискового файла). При этом правда требуются некоторые усилия для строгого выполнения второго свойства с учетом того, что запись может быть произведена не полностью (например, вследствие отключения питания). Обычно, чтобы гарантировать второе свойство, к записям добавляется заголовок с контрольной суммой. После этого незаконченные записи обнаруживаются путем сравнения сохраненной контрольной суммы с вычисленной после прочтения записи.

Процесс, который инициирует принятие некоторого решения (назовем его инициатором), рассылает его всем активным участникам протокола с тем, чтобы они записали это решение в свои журналы. Как только решение оказалось записано в журналы у множества участников, составляющего кворум, оно принимается. После этого в любом кворуме найдется участник, у которого в журнале будет записано это принятое решение. Данное свойство позволяет гарантировать, что однажды принятое решение впоследствии не изменится и не будет потеряно. Остальные детали протокола PAXOS обеспечивают механизм, гарантирующий, что участники, которые были неактивны во время принятия определенного решения, впоследствии узнают о нем, причем порядок принятых решений, о которых они узнают, будет таким же, как и у остальных участников протокола.

Число сообщений, необходимых для работы протокола консенсуса, можно свести к минимуму, если гарантировать, что в течение некоторого времени роль инициатора будет играть один определенный процесс из числа активных участников протокола (см. [31]). Такой процесс называют ведущим или мастером, соответственно остальные процессы называют ведомыми. Именно мастер MDS обрабатывает все запросы клиентов и регистрационные сообщения от CS. Смена мастера происходит, когда ведомые MDS обнаруживают отсутствие сообщений от мастера в течение некоторого времени.

Описанный выше протокол консенсуса имеет один фундаментальный недостаток. Журнал, куда участники записывают принятые решения, неограниченно растет в ходе функционирования системы. Это приводит к неограниченному использованию дискового пространства (поскольку журнал должен переживать отключения питания, а значит сохраняться на диске). Вторая связанная с этим проблема менее очевидна, но не менее серьезна. Поскольку при перезапуске сервис должен перечитать свой журнал, его время старта будет расти пропорционально длине журнала. Поэтому для работы системы абсолютно необходимо иметь механизм, ограничивающий рост журнала. Роль такого механизма традиционно играют снимки (snapshots) состояния системы. После того, как мы сохранили в каком-то виде состояние системы на некоторый момент времени, мы можем отбросить все записи журнала, соответствующие решениям, принятым до этого момента, поскольку они в совокупности привели систему в то состояние, которое мы сохранили. Для восстановления состояния системы после перезапуска сервиса нам достаточно восстановить сохраненный снимок состояния, после чего прочитать журнал решений, принятых после создания этого снимка и, соответственно, обновить состояние сервиса.

Состояние MDS обновляется подсистемами, обеспечивающими обработку клиентских запросов и управление кластером (Рисунок 30). Именно они являются источниками решений, принимаемых протоколом консенсуса (например, решений о создании или удалении файлов). Для сохранения совокупного состояния этих разнородных подсистем полезно иметь единый универсальный механизм. Роль такого механизма играет разработанный нами *пул объектов*, тесно интегрированный в протокол консенсуса. Каждая подсистема, модифицирующая в результате своей работы состояние MDS, регистрирует в этом пуле один или несколько *типов объектов*. Тип характеризуется набором функций для преобразования объекта в массив байт для сохранения в составе снимка состояния системы и для обратного преобразования. Таким образом, нам удалось создать единый механизм создания снимков состояния всей системы практически независимый от подсистем, создающих и обновляющих это состояние.

Текущее состояние MDS хранится исключительно в оперативной памяти соответствующего процесса, при этом журнал и снимок состояния позволяют воссоздать это состояние в случае перезапуска процесса. Такая архитектура с одной стороны обеспечивает минимальное время обработки клиентских запросов, с другой стороны размер доступной физической памяти сервера накладывает ограничения на масштабируемость системы. Мы рассмотрим эти ограничения более подробно в главе 4.3.

4.2.4. Особенности реализации различных схем хранения данных

Реализованная нами СХД поддерживает различные схемы хранения данных, которые можно разделить на 2 больших класса – это *реплики* и схемы с *избыточным кодированием* или (n, k) схемы, например RAD6 ($n = k + 2$) и коды Рида-Соломона с $n - k > 2$ (см. 2.1). Эти 2 класса принципиально отличаются по способу хранения данных и организации доступа к ним по причинам, которые мы рассмотрим в данной главе.

При хранение данных в репликах каждый блок просто записывается в нескольких экземплярах на разных CS. Для этого мы используем схему цепочечной репликации, описанную в [33]. Все запросы на запись данных от клиентского агента всегда поступают на первый CS в цепочке, как показано на следующем рисунке. Каждый CS инициирует запись данных на диск и сразу же пересылает запрос следующему CS в цепочке. Прежде, чем ответить агенту или предыдущему CS в цепочке, каждый CS дожидается окончания записи на свой диск, а также получения ответа от следующего CS в цепочке. Это означает, что первый CS в цепочке сигнализирует агенту об успешно завершенной записи только после того, как запись будет произведена на все диски в цепочке.

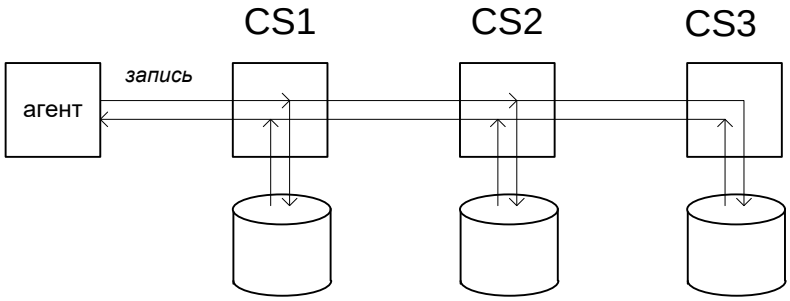

Рисунок 31. Запись реплик по цепочке.

Запись реплик по цепочке позволяет сериализовать записи в пересекающиеся области файла и гарантировать, что после завершения операций записи все дисковые блоки в цепочке имеют одинаковое содержимое. В отличие от запросов на запись данных запросы на чтение не распространяются по цепочке. Они присылаются одному CS, который выбирается клиентом из соображений минимизации нагрузки на диск и сетевую инфраструктуру.

Хранение данных в репликах обеспечивает максимальную скорость доступа и максимальную надежность по сравнению с другими схемами с той же избыточностью (см. 2.2.3). Платой за этой является наименьшая эффективность использования дискового пространства. Более экономичными с точки зрения использования дискового пространства (хотя и менее надежными и быстрыми) являются (n, k) схемы хранения или схемы с избыточным кодированием.

Хранение данных с избыточным кодированием принципиально отличается от хранения в репликах, поскольку блоки контрольных сумм, которые обеспечивают эту избыточность, должны обновляться одновременно с информационными блоками. При этом вся инфраструктура доступа к данным должна гарантировать, что клиент никогда не обнаружит набор дисковых блоков в частично обновленном состоянии, поскольку при этом могут оказаться искаженными не только данные, еще не записанные клиентом, но и данные, которые он уже записал, но которые имеют общие контрольные суммы с данными, запись которых не была завершена. Набор записанных клиентом данных, имеющий общую контрольную сумму, мы будем называть *страйпом*. Область данных, записанную клиентом в результате одной операции записи обычно называют *икстентом*. Следующий рисунок иллюстрирует страйп с двумя икстентами в схеме хранения $(3,2)$[7].

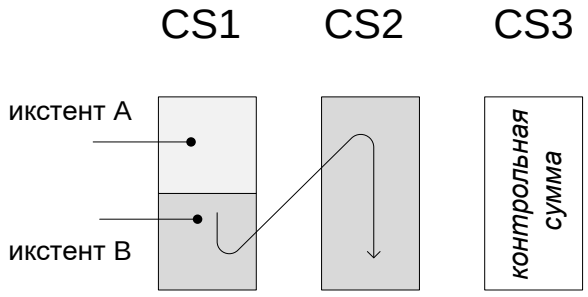

Рисунок 32. Страйп с двумя икстентами и схемой (3,2)

В случае, если запись икстента B прервется, икстент A может остаться с неправильной контрольной суммой, поскольку она общая для A и B, что впоследствии может привести к необратимому повреждению данных, причем у клиента даже не будет способа обнаружить это повреждение.

Для того, чтобы исключить описанную выше ситуацию СХД обычно организую запись страйпов так, чтобы они всегда записывались целиком и никогда не модифицировались впоследствии. При этом максимальная производительность дисковой подсистемы достигается, если данные просто пишутся подряд. Файловые системы, построенные по этому принципу, называются лог-структурированными ([39,40]). Следуя этому подходу, агент записывает страйпы подряд путем добавления в конец некоторого файла, который мы будем называть *дата-логом*, поскольку данные в него пишутся строго последовательно. Для поиска данных в нем по смещению в файле, с которым имеет дело клиент, необходима отдельная таблица, хранящая информацию о размерах и смещениях всех записанных икстентов, а также позиции данных этих икстентов в дата-логе. Наконец, чтобы не хранить данные, которые уже переписаны и недоступны для клиента, необходима

[7] На практике такая схема не используется в силу низкой надежности.

процедура *уборки мусора* (garbage collection), которая освобождает место на диске, занятое уже неактуальными данными в дата-логе. Освободить занятое на диске место можно только освободив целый чанк. Вероятность того, что он весь будет состоять из неактуальных данных крайне мала. Поэтому сборщик мусора руководствуется следующим алгоритмом:

- Он находит чанк, у которого отношение размера актуальных данных к размеру всех записанных данных меньше некоторого порога (3/4). Размер актуальных и записанных данных хранится для каждого чанка в отдельной таблице.
- Все данные, которые все еще актуальны, из этого чанка перемещаются в конец дата-лога. Соответственно обновляется таблица икстентов.
- После копирования всех актуальных данных чанк освобождается. При этом вся информация о нем удаляется из метаданных файла, а место, занятое его блоками на чанк серверах, становится свободным.

Как видно из вышеизложенного, реализация каждого (n, k) файла представляет собой небольшую базу данных с целым набором дополнительных таблиц. Для того, чтобы иметь возможность хранить всю дополнительную информацию, файл реализован в виде директории, в которой хранятся дата лог и служебные таблицы метаданных, а также журнал операций (см. ниже). Для клиента эта директория выглядит как один файл. Мы будем называть ее контейнером. Следующий рисунок иллюстрирует основные компоненты такого контейнера и их взаимодействие при операции записи.

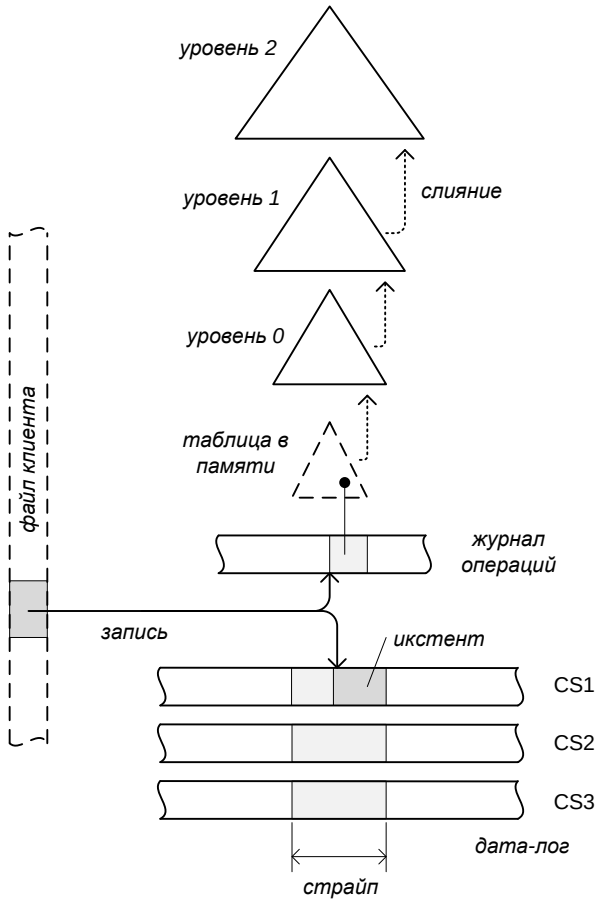

Рисунок 33. Внутренняя структура контейнера, реализующего избыточное кодирование клиентского файла.

Данные, которые клиент записывает в то, что он считает файлом (мы будем называть его *виртуальным* файлом), сначала накапливаются в буфере в памяти агента, где из них формируется страйп. Страйп записывается в соответствие с выбранной схемой кодирования на необходимое количество CS-ов, формируя постоянно растущий дата-лог. После того, как страйп записан, в журнал операций сохраняется информация обо всех записанных в составе страйпа икстентах, чтобы клиент мог впоследствии их найти в дата-логе по смещению в своем виртуальном файле. Журнал операций играет ту же роль, что и в классических системах баз данных (см. например [34], гл. 8), обеспечивая атомарность обновления метаданных. Одновременно с записью в журнал, метаданные добавляются в таблицу икстентов в памяти агента, чтобы их можно было впоследствии найти.

Постоянно растущий журнал операций необходимо периодически очищать, сохраняя метаданные в другие дисковые структуры, имеющие ограниченный размер и оптимизированные для быстрого поиска. Эта процедура называется *ротацией*. В случае, если ротация сохраняет метаданные в единственную таблицу, время ротации оказывается пропорционально ее размеру, а вставка метаданных в произвольные места этой таблицы может увеличить время ротации на порядки величины вследствие ограничений диска на

количество возможных операций в секунду. Современный подход к решению этой проблемы заключается в использовании *дерева таблиц* (см. [35]). При таком подходе таблицы никогда не изменяются, а только пересоздаются, в результате удается добиться лучшей производительности диска за счет строго последовательного доступа к нему. Чтобы амортизировать накладные расходы на пересоздание большой таблицы при вставке небольшого количества метаданных, используется иерархия таблиц. Таблицы верхнего уровня имеют больший размер, но пересоздаются реже. Таблицы пересоздаются в результате операции, которая называется *слиянием*. В ходе слияния таблица уровня j объединяется с таблицей уровня $j+1$, результат сохраняется в виде новой таблицы уровня $j+1$, а уровень j остается пустым. Слияние происходит, когда размер таблицы на уровне j превышает некоторый порог. На нижнем уровне этой иерархии расположена таблица метаданных, хранящаяся в памяти (Рисунок 33). Поиск в дереве таблиц происходит, начиная с нижнего уровня иерархии. Если элемент не найден в таблице нижнего уровня, производится поиск на вышележащем уровне и т.д. Аналогичным образом хранятся и другие служебные таблицы, например, таблица чанков, где учитывается занятое каждым чанком дисковое пространство, а также его часть, занятая актуальными данными, еще не перезаписанными клиентом. Эта таблица используется сборщиком мусора для выбора чанка, который будет освобожден после копирования актуальных данных в конец дата-лога.

Таблицы метаданных, а также журнал операций хранятся в виде файлов в репликах с той же избыточностью, что установлена для дата лога (то есть со схемой хранения $(n-k+1, 1)$). Это гарантирует максимальную надежность, при этом общая эффективность падает незначительно, поскольку размер метаданных обычно пренебрежимо мал по сравнению с размером дата лога. Все файлы метаданных вместе с дата логом хранятся в директории – контейнере. Агент, предоставляющий доступ к СХД, скрывает ее содержимое от пользователя, создавая иллюзию, что это и есть файл, с которым работает пользователь.

Если клиент хранит в файле данные с низкой энтропией (легко сжимаемые), например, текст, то он может активизировать режим сжатия данных при сохранении в файл. Для этого используется специальный системный атрибут файла. Если он установлен, то все икстенты, размер которых превышает некоторый порог, сжимаются перед сохранением в дата-лог. Соответственно, в таблице икстентов сохраняется информация о размере икстента до и после сжатия. Специальный эвристический алгоритм используется для оценки эффективности сжатия, чтобы не тратить процессорные ресурсы на сжатие данных с высокой энтропией. Для этого производится пробное сжатие небольшого фрагмента икстента.

Описанная архитектура позволила нам при использовании избыточного кодирования добиться производительности операций записи при пиковых нагрузках сравнимой с производительностью при хранении данных в репликах. Средняя производительность с учетом сборки мусора обычно составляет 1/3 от производительности при хранении данных в репликах. В следующей главе мы рассмотрим особенности архитектуры чанк-серверов, которые позволили нам улучшить производительность и реализовать непрерывный контроль целостности данных (скраббинг), критически важный для обеспечения надежности хранилища, как показано в главе 3.2.

4.2.5. Гибридное хранилище данных

В настоящее время вращающиеся диски имеют оптимальное соотношение емкости, производительности и цены, что и определяет их повсеместное использование в системах хранения данных. При этом они имеют и свои недостатки. Главный из них – крайне ограниченное число возможных операций доступа к диску в секунду (порядка 100). Это ограничение связано с необходимостью механического позиционирования головки диска, а также с необходимостью ожидания момента, когда диск повернется под углом, необходимым для того, чтобы адресуемый дисковый сектор оказался под головкой диска. Поэтому производительность диска резко падает при уменьшении среднего размера блока данных перемещаемых с диска или на него, а также при увеличении количества клиентов, одновременно использующих диск, поскольку при этом растет количество одновременных обращений в различные области диска и, соответственно, растет количество необходимых перемещений головки диска. Последнее обстоятельство особенно актуально по 2 причинам:

1. Количество клиентов в кластере может существенно превышать количество дисков. Соответственно, каждый из них столкнется с падением производительности, вызванным активностью других клиентов.
2. Сумарное число операций чтения и записи в кластере пропорционально числу дисков в кортежах для используемой схемы кодирования (n для схемы (n,k)). Наиболее эффективные с точки зрения использования дискового пространства схемы кодирования имеют наибольшее число дисков в кортеже. Значит клиенты будут стремится использовать максимально возможное n, усугубляя в n раз проблему, обозначенную в п.1.

В отличие от вращающихся дисков, твердотельные накопители (SSD) имеют на 3-4 порядка большее количество возможных операций доступа в секунду. Но их использование для построения СХД экономически невыгодно вследствие более высокой стоимости. Однако, есть простой и эффективный способ повышения производительности хранилища, если наряду с вращающимися дисками использовать в нем и SSD накопители[8]. Этот способ называется *журналированием записей*. Все данные, которые клиенты пишут на CS, записываются не сразу на вращающийся диск, а помещаются в файл на SSD, который мы называем *журналом*, поскольку записи помещаются в него строго одна за другой. Когда журнал заполняется выше определенного порога, происходит его *ротация* – данные перемещаются на

[8] Подобное хранилище, построенное на базе разнородных носителей информации, называется *гибридным*.

вращающийся диск, а занятое место освобождается для новых записей. В ходе ротации данные помещаются на диск не в том порядке, в каком они записывались в журнал, а в порядке, обусловленном их расположением на вращающемся диске. При этом соседние икстенты объединяются, что уменьшает результирующее количество дисковых операций. Следующий рисунок иллюстрирует эту идею.

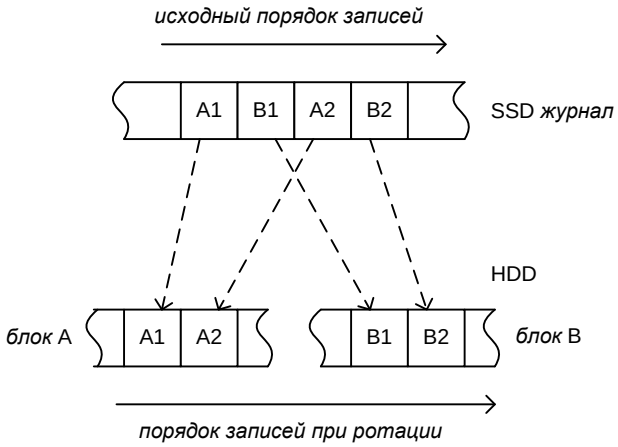

Рисунок 34. Переупорядочивание записей при ротации SSD журнала.

На рисунке показаны икстенты, записанные 2 клиентами в 2 различных дисковых блока. Если бы они сразу записывались на вращающийся диск, это потребовало бы 4 обращений к нему. При использовании SSD журнала объединение икстентов позволяет сократить количество обращений до 2, т.е. до количества клиентов. Фактически использование журнала сериализует одновременные обращения к диску со стороны клиентов позволяя каждому получить его 'долю' от производительности диска, исключив дополнительное падение производительности вследствие одновременного обращения к диску разных клиентов.

Использование SSD журнала позволяет не только улучшить производительность операций чтения и записи, но и повысить надежность хранения данных. Во-первых, журналирование позволяет нивелировать падение скорости восстановления, вызванного поступлением запросов на доступ к данным со стороны клиента во время восстановления дискового блока. Во-вторых, журналирование позволило нам реализовать непрерывную проверку целостности данных (скраббинг), критически важную для построения надежного хранилища (см. 3.2). Для того, чтобы иметь возможность проверки целостности сохраненного блока данных, необходимо отдельно от него[9] хранить контрольную сумму. Проблема заключается в том, что без использования журнала надежная реализация этой схемы оказывается невозможна. Действительно, в случае внезапного отключения питания

[9] Возможна реализация и с совместным хранением, но она оказывается неэффективна вследствие нарушения выравнивания при сохранении данных вместе с дополнительной информацией.

во время записи данных или контрольной суммы, эта пара неизбежно останется в несогласованном состоянии. При этом у нас не будет способа определить, возникла ли эта несогласованность как результат незавершенной записи или вследствие повреждения диска. Журналирование данных решает эту проблему за счет того, что незавершенная в результате внезапного выключения системы ротация журнала просто повторяется с самого начала при последующем старте CS, так что все записи – и завершенные и нет – повторяются снова.

Таким образом, разработанная нами технология построения гибридного СХД позволяет не только достичь лучших показателей производительности, но и радикально улучшить показатели надежности хранилища за счет ускорения восстановления и раннего детектирования скрытых повреждений дисковых накопителей. При этом использование дополнительных твердотельных накопителей приводит к минимальному удорожанию системы, поскольку один SSD может использоваться совместно несколькими CS, работающими на одном сервере, что эффективно амортизирует его стоимость.

4.2.6. Оптимизация параметров хранилища для обеспечения максимальной надежности.

В процессе реализации системы хранения данных Acronis Storage мы провели всестороннюю проверку выбранных при ее проектировании параметров с учетом разработанной нами теоретической модели. В результате был принят ряд важных с точки зрения оптимизации надежности хранилища технических решений:

1. Мы отказались от использования схем хранения с избыточностью 1, как не обеспечивающих масштабирования системы (см. 2.3.5).
2. В СХД реализовано кодирование с избыточностью 2 и 3, последнее рекомендуется для использования при наличии большого количества дисков.
3. Процедура восстановления после дисковых отказов обеспечивает последовательное восстановление утраченных дисковых блоков, так что каждый диск в любой момент времени участвует в востановлении не более одного блока. Это позволяет достичь максимально возможной производительности дисковых операций, соответственно сократив время восстановления и повысив надежность хранилища. При этом первыми восстанавливаются фрагменты данных с максимальным количеством утраченных блоков.
4. Используется максимально возможное количество групп размещения (см. 3.1).
5. В СХД используется непрерывный скраббинг со скоростью порядка 1/10 от максимальной производительности дисков для раннего обнаружения скрытых повреждений дисков.
6. Схема хранения данных, а также домен отказов для их размещения в хранилище, настраиваются пользователем на уровне папок на файловой системе, так чтобы при необходимости обеспечить желаемый баланс между эффективностью хранения и надежностью в зависимости от цены потенциальной утраты конкретных данных пользователя.

В следующих главах мы подробнее остановимся на достигнутых нами результатах, а также рассмотрим перспективы дальнейшего развития системы.

4.3. Достигнутые в ходе внедрения результаты

Разработанная нами система хранения данных Acronis Storage зарекомендовала себя эффективным и надежным решением для корпоративных клиентов, обеспечивающим производительность операций доступа к данным на уровне производительности локального диска. Благодаря использованию современных технологий хранения данных и наших собственных разработок на базе разработанного нами теоретического фундамента, изложенного в главах 2 и 3, нам удалось свести к минимуму вероятность потери данных вследствие отказов оборудования. На данный момент в хранилищах, построенных на базе технологии Acronis Storage, содержится более 100 петабайт пользовательских данных. В состав каждого из них входит до 2500 дисков. При этом за всю историю эксплуатации (около 5 лет) этих хранилищ не было ни одного случая потери данных вследствие отказа оборудования.

Помимо надежности и производительности интересным как с теоретической, так и с практической точки зрения аспектом является вопрос о масштабируемости системы и ее пределах. Как было показано в главе 2.3.5, использование избыточности не менее 2 гарантирует масштабируемость надежности системы при отсутствии прочих факторов, которые могут ее ограничивать. Одним из таких факторов является пропускная способность сетевой инфраструктуры. Для оптимизации нагрузки на нее и сохранении надежности на требуемом уровне наша СХД имеет специальный алгоритм создания дисковых кортежей, учитывающий области отказов (см. главу 3.3). Пользователь может менять требуемый уровень надежности, задавая домен отказов на уровне папок и отдельных файлов. Доменом отказов по умолчанию является сервер. Это означает, что диски в одном кортеже никогда не выбираются из числа подключенных к одному серверу, так что отказ любого сервера приводит к выходу из строя не более одного диска в кортеже. При этом учитывается также *локальность* размещения дисков в кортеже – предпочтение отдается кортежам, где диски наиболее близки друг к другу в терминах аппаратной инфраструктуры, например, расположены в пределах одной стойки центра обработки данных. Это позволяет нам оптимально использовать сетевую инфраструктуру.

Даже при условии, что сеть не является фактором, ограничивающим масштабируемость системы, существует ограничение, ограничивающее масштабируемость, заложенное в самой архитектуре системы. Это ограничение, накладываемое центральным сервисом метаданных (MDS). Центральный сервис метаданных необходим по многим причинам:

– для установления единого пространства имен;

– для установления единого пула чанк серверов с возможностью балансировки нагрузки между ними;
– для координации восстановления после отказа оборудования;

Поскольку центральный сервер существует в единственном экземпляре, его ресурсы неизбежно будут являться фактором, ограничивающим масштабируемость системы. Наиболее очевидным ресурсом является оперативная память, где содержится все текущее состояние MDS. Большая часть памяти MDS занята объектами, представляющими файлы, чанки и дисковые блоки. Их размеры приведены в следующей таблице.

Таблица 12. Размеры базовых объектов в памяти MDS

Объект	Размер в оперативной памяти
Файл	304 + 128 * число компонент пути + длина пути
Чанк	240
Дисковый блок	64

Таким образом, размер состояния кластера в памяти MDS напрямую зависит от количества файлов, их среднего размера, а также размера чанков и схемы хранения. В среднем можно рассчитывать на то, что для 1 петабайта доступного места в кластере потребуется 1 гигабайт оперативной памяти. Менее очевидным фактором является ограниченность ресурсов процессора. Код, обновляющий метаданные MDS, исполняется на единственном процессорном ядре. Это позволяет избежать блокировок, неизбежных при работе с метаданными из нескольких потоков. В результате обновление большого количества метаданных может занимать значительное время и даже привести к неработоспособности кворума в алгоритме PAXOS (см. 4.2.3). Наконец, третий фактор, ограничивающий масштабируемость MDS – это время, необходимое для восстановления его состояния после старта. Если оно становится слишком большим, система также может стать неработоспособной.

Перечисленные выше проблемы являются общими для всех существующих в настоящее время СХД. Единственным решением, применяющимся на практике является использование нескольких MDS (шардирование), где каждый хранит лишь часть общих метаданных, например как в СХД СЕРН [36]. Фактически этот подход лишь отодвигает предел масштабирования системы, поскольку остаются операции, требующие вовлечения всех MDS, например, листинг корневой директории. В нашем случае эмпирическим пределом масштабирования является 10000 дисков по 10 терабайт, что дает нам 100 петабайт доступного дискового пространства. В следующей главе мы обсудим пути дальнейшего развития нашей СХД и расширения пределов ее масштабирования.

4.4. Направления дальнейшего развития

Дальнейшее развитие СХД Acronis Storage будет происходит по 2 тесно связанным между собой направлениям – внедрение более эффективных способов организации хранения данных и расширение пределов масштабирования системы.

4.4.1. Внедрение более эффективных способов организации хранения данных

Схема хранения данных с избыточным кодированием, описанная в главе 4.2.4, имеет ряд недостатков:

- она увеличивает количество метаданных, которые хранятся на MDS, поскольку вместо одного файла ему приходится хранить информацию о примерно 10 разных файлах;
- она реализована на агенте, соответственно, пока агент не работает с файлом, никакие действия с ним невозможны. Это затрудняет реализацию таких операций, как снапшоты и изменения схемы кодирования. Уборка мусора также реализована агентом, поэтому она производится только в то время, пока файл открыт на запись, замедляя при этом клиентские операции чтения и записи файла;
- сборка мусора является весьма сложной и не всегда эффективной процедурой.

Журналирование данных, описанное в главе 4.2.5, эффективно сериализует операции записи, но никак не помогает операциям чтения данных, которые по-прежнему приводят к деструктивной интерференции между клиентами, осуществляющими параллельное чтение данных одного и того же CS.

С связи с вышеизложенным перспективным представляется подход, когда избыточное кодирование данных осуществляется на чанк сервере. При этом данные сначала сохраняются в репликах в быстром SSD хранилище, играющем роль журнала в терминах главы 4.2.5. Только после того, как активность клиента прекращается, эти данные сохраняются на более медленный вращающийся диск в более эффективном формате с избыточным кодированием. В случае, если клиент активно читает какую-то область закодированных данных, мы можем переместить ее в быстрое хранилище для ускорения доступа. Таким образом, вместо простого журнала в SSD хранилище организуется кэш для 'горячих' данных, которые перемещаются на более медленный диск по мере 'охлаждения'.

Данный подход позволит нам существенно упростить процедуру уборки мусора в закодированных данных, поскольку мы можем выбрать размер страйпа достаточно большим и сделать его, а не чанк, структурной единицей выделения и освобождения дискового пространства. Тогда новый страйп будет просто записываться на место, освободившееся после удаления ставшего неактуальным страйпа без переноса актуальных данных на новое место, как в схеме, описанной в главе 4.2.4.

4.4.2. Расширение пределов масштабирования системы

Основным фактором, ограничивающим масштабирование нашей системы, является необходимость хранения информации о чанках в памяти MDS. Реализация идей, изложенных в 4.4.1, позволит нам существенно увеличить размер чанка, поскольку он перестанет играть роль единицы размещения данных. Чанк будет обладать внутренней структурой, информация о которой будет храниться в CS. В результате мы сможем уменьшить количество информации, хранящейся в MDS. Кроме того, вместо хранения всей информации о цепочке дисковых блоков в каждом чанке мы можем хранить в нем ссылку на *группу размещения* (см. главу 3.1), что также уменьшит размер информации, которая хранится в памяти MDS для каждого чанка.

Дальнейшее расширение пределов масштабирования системы возможно за счет разбиения ее на отдельные подкластера или *ячейки* с группой объединенных протоколом PAXOS серверов метаданных в каждом из них. Разделение пространства имен между ячейками может осуществляться статически или динамически (см. [37]).

Заключение

Можно выделить следующие основные результаты проделанной работы и вытекающие из них выводы:

1. Предложена математическая модель надежности хранения данных в многодисковом хранилище с разбиением данных на фрагменты с заданной избыточностью более полно описывающее реальные СХД, чем модель Марковских цепей.
2. Разработан комплекс имитационных моделей надежности СХД, отражающий реальную архитектуру подобных систем. С помощью имитационного моделирования показана важность приоритезации восстановления утраченных дисковых блоков для оптимизации надежности хранилища.
3. Исследовано влияние избыточности на масштабирование надежности хранилища с ростом числа дисков. Показано, что для сохранения надежности при добавлении дисков в систему необходимо использование значения избыточности не менее 2.
4. На базе созданной математической модели исследовано влияние ограничения числа возможных вариантов размещения дисковых блоков (групп размещения) на надежность хранилища. Показано, что такое ограничение уменьшает надежность хранилища. Найдена нижняя граница для количества групп размещения, обеспечивающее сохранение надежности хранилища на приемлемом уровне.
5. На базе созданной математической модели исследовано влияние скрытых повреждений на надежность хранилища. Сформулирована и доказана теорема, дающая нижнюю границу на вероятность отказа при наличии скрытых повреждений. Предложены методы борьбы со скрытыми повреждениями и дана оценки их эффективности.
6. Исследовано влияние особенностей аппаратной инфраструктуры на надежность СХД, предложена методика учета этих особенностей при распределении блоков данных по дискам в системе.
7. Полученные результаты использованы при создании комплекса программ реализующего отказоустойчивое, масштабируемое хранение данных.

Список литературы

1. Иваничкина, Л.В. Методы обеспечения надежности хранения сверхбольших объемов данных в распределенной системе. 58 научная конференция МФТИ. сб. науч. тр. М.: МФТИ, 2015.
2. Иваничкина, Л. В., Бухтияров, П.А., Вареник, Р.В., Науменко, С.А., Непорада, А.Л. Имитационная модель надежности хранения данных, выполняющая симуляцию сбоев и процессов восстановления данных в распределенной системе хранения данных. Свидетельство о государственной регистрации программы для ЭВМ № 2015618800. 2015.
3. Иваничкина, Л.В., Бухтияров, П.А., Вареник, Р.В., Науменко, С.А., Непорада, А.Л. Имитационная модель процессов сбоев и восстановления данных для различных схем размещения данных в распределенной системе хранения. Свидетельство о государственной регистрации программы для ЭВМ № 2016613180. 2016.
4. Иваничкина, Л.В., А.П. Непорада. Модель надежности распределенной системы хранения данных в условиях явных и скрытых дисковых сбоев. Труды Института системного программирования РАН, том 27, выпуск 6, 2015 г. стр. 253–274.
5. Акритас А.Г. Основы компьютерной алгебры с приложениями. Пер. с англ. М.: Мир, 1994. 544 с.
6. Джон Дж. Камени, Дж. Лори Снелл. Конечные цепи Маркова. Пер. с англ. М.: Наука, 1970. 272 с.
7. Каштанов В.А., Медведев А.И. Теория надежности сложных систем. 2-е изд., перераб. М.: ФИЗМАТЛИТ, 2010. 608 с.
8. Вентцель Е.С. Исследование операций: задачи, принципы, методология. 2-е изд. М.: Наука, 1988. 208 с.
9. К. Дж. Дейт. Введение в системы баз данных. 7-е изд. Пер. с англ. Изд. "Вильямс". 2001. 1072 с.
10. L. Ivanichkina, A. Neporada. The reliability model of a distributed data storage in case of explicit and latent disk faults. Journal of Engineering and Applied Sciences. 2015. Vol 10(20). pp. 9713—9724.
11. L. Ivanichkina, A. Neporada. Mathematical methods and models of improving data storage reliability including those based on finite field theory. Contemporary Engineering Sciences. 2014. Vol 7(28). pp. 1589 – 1602.
12. L. Ivanichkina, A. Neporada. Computer Simulator of Failures in Super Large Data Storage. Contemporary Engineering Sciences. 2015. Vol 8(28). pp. 1679–1691.

13. L. Ivanichkina, V. Vinnikov. Comparative study of LRC and RS codes. Contemporary Engineering Sciences. 2016. Vol 9(21). pp. 1015-1029.
14. L. Ivanichkina, K. Korotaev, A. Neporada. Failure resilient data placement policies for distributed storages. Contemporary Engineering Sciences. 2016. Vol 9(30). pp. 1463-1489.
15. O. Volkov, A. Zaitsev, A. Kobets, L. Ivanichkina, K. Korotaev. Management of garbage data in distributed systems. US patent application 15/445,858. 2017.
16. Buzacott, J.A. Markov approach to finding failure times of repairable systems. Reliability, IEEE Transactions on reliability. 1970. Vol 19(4). pp. 128–134.
17. M. Rausand, A. Hoyland. System Reliability Theory. Models, Statistical Methods, and Applications. Second ed. Wiley, 2004. 636 p.
18. D. A. Patterson, G. A. Gibson, R. H. Katz, A case for redundant arrays of inexpensive disks (RAID). ACM SIGMOD Record. 1988. Vol 17(3). pp. 109-116.
19. P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, D. A. Patterson. RAID: High-Performance, Reliable Secondary Storage. ACM Computing Surveys. 1994. Vol 26(2). pp. 145-185.
20. A. Cidon, S. Rumble, R. Stutsman, S. Katti, John Ousterhout, M. Rosenblum. Copysets: reducing the frequency of data loss in cloud storage. Proceeding USENIX ATC. 2013. pp. 37-48.
21. T.J.E. Schwarz ; Qin Xin ; E.L. Miller ; D.D.E. Long ; A. Hospodor ; S. Ng. Disk Scrubbing in Large Archival Storage Systems. Proceedings MASCOTS. 2004. 10 p.
22. L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, J. Schindler. An analysis of latent sector errors in disk drives. Proceeding ACM SIGMETRICS. 2007. pp. 289-300.
23. E. Pinheiro, W. Weber, L. Barroso. Failure Trends in a Large Disk Drive Population. Proceedings of USENIX FAST. 2007. 13 p.
24. I. Iliadis, R. Haas, Xiao-Yu Hu, E. Eleftheriou. Disk Scrubbing Versus Intradisk Redundancy for RAID Storage Systems. Journal ACM Transactions on Storage. 2011. Vol 27(2). pp. 241-252.
25. J. Li, X. Ji, Y. Jia, B. Zhu, G. Wang, Z. Li, X. Liu. Hard Drive Failure Prediction Using Classification and Regression Trees. IEEE/IFIP International Conference on Dependable Systems and Networks. 2014. 12 p.
26. A. Oprea, A. Juels. A Clean-Slate Look at Disk Scrubbing. Proceedings of USENIX FAST. 2010. 14 p.

27. J. Pâris, A. Amer, D. D. E. Long, T. J. E. Schwarz. Evaluating the Impact of Irrecoverable Read Errors on Disk Array Reliability. Proceedings of Dependable Computing. 2009. pp. 379-384.
28. S. Ghemawat, H. Gobioff, S.-T. Leung. The Google File System. ACM SOSP. 2003. 15 p.
29. L. Lamport. The Part-Time Parliament. ACM Transactions on Computer Systems. 1998. Vol 16 (2). pp. 133–169.
30. L. Lamport. Paxos Made Simple. ACM SIGACT News. 2001. Vol 32(4). pp. 51-58.
31. L. Lamport. Fast Paxos. Distributed Computing. 2006. Vol 19 (2). pp. 79-103.
32. F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. ACM Computing Surveys. 1990. Vol 22(4). pp. 299–319.
33. R. v. Renesse, F. B. Schneider. Chain replication for supporting high throughput and availability. Proceedings OSDI. 2004. 14 p.
34. H. Garcia-Molina, J. D. Ullman, J. Widom. Database System Implementation. Prentice Hall. 2000. 653 p.
35. P. O'Neil, E. Cheng, D. Gawlick, E. O'Neil. The log-structured merge-tree (LSM-tree). Acta Informatica. 1996. Vol 33 (4). pp. 351–385.
36. S. A. Weil, S. A. Brandt, E. L. Miller, D. E. Long, C. Maltzahn. Ceph: a scalable, high-performance distributed file system. Proceedings OSDI. 2006. pp. 307-320.
37. S. A. Weil, K. T. Pollack, S. A. Brandt, E. L. Miller. Dynamic metadata management for petabyte-scale file systems. Proceedings ACM/IEEE SC. 2004.
38. K. S. Trivedi. Probability and Statistics with Reliability, Queueing, and Computer Science Applications, 2nd Edition. Wiley. 2002. 830 p.
39. M. Rosenblum and J. Ousterhout. The Design and Implementation of a Log-Structured File System. ACM Trans. on Computer Systems. Vol 10(1). 1992. pp. 26–52.
40. M. Seltzer, K. Bostic, M. McKusick, and C. Staelin. An Implementation of a Log-Structured File System for UNIX. In Proc. of the 1993 Winter USENIX. 1993. pp. 307-326.
41. D. Ford, F. Labelle, F. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, S. Quinlan, Availability in Globally Distributed Storage Systems, Proceedings of the 9th USENIX OSDI. 2010. 14 p.
42. M. Malhotra, K. S. Trivedi, Reliability Analysis of Redundant Arrays of Inexpensive Disks. Journal of Parallel and Distributed Computing. Vol 17. 1993. pp. 146-151.
43. B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, J. Haridas, C. Uddaraju, H. Khatri, A. Edwards, V. Bedekar,

S. Mainali, R. Abbasi, A. Agarwal, M. F. ul Haq, M. I. ul Haq, D. Bhardwaj, S. Dayanand, A. Adusumilli, M. McNett, S. Sankaran, K. Manivannan, L. Rigas. Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency. Proceeding SOSP. 2011. pp. 143-157.

44. Q. Lian, W. Chen, Z. Zhang. On the Impact of Replica Placement to the Reliability of Distributed Brick Storage Systems. Proceedings of the 25th IEEE International Conference on Distributed Computing Systems. 2005. pp. 187–196.
45. V. Venkatesan, I. Iliadis, X.-Yu Hu, R. Haas, C. Fragouli. Effect of Replica Placement on the Reliability of Large-Scale Data Storage Systems. Proceedings of IEEE MASCOTS. 2010. 10 p.
46. [Электронный ресурс] https://github.com/ludmilai/markov_model
47. [Электронный ресурс] https://github.com/ludmilai/storage_model