Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Московский физико-технический институт
(государственный университет)»

Факультет радиотехники и кибернетики

Кафедра теоретической и прикладной информатики

Исследование и разработка методов параллельного доступа к распределенному хранилищу данных с помощью протокола iSCSI

Выпускная квалификационная работа
(бакалаврская работа)

Направление подготовки: 03.03.01 Прикладные математика и физика

Выполнил:
студент 211 группы _____ Вилкин Михаил Сергеевич

Научный руководитель: _____ Иваничкина Людмила Владимировна

Москва 2016

Оглавление

Глава 1

Введение

Распределенное хранилище — это модель онлайн-хранилища, некий виртуальный носитель информации, в котором данные хранятся и обрабатываются на многочисленных серверах, распределенных в сети. В настоящее время подобные хранилища получили большое распространение, так как они очень удобны для пользования: доступность данных (необходим только доступ к сети Интернет), отсутствие необходимости заботиться об аппаратной и программной поддержке накопителей, возможность организации совместного пользования.

Организация доступа к данным может быть произведена различными способами, но наиболее популярные протоколы — это FCP (Fibre Channel Protocol), FCIP (Fibre Channel over TCP/IP) и iSCSI (Internet Small Computer System Interface). У каждого из них есть свои преимущества и недостатки, но для исследования был выбран именно протокол iSCSI [1], потому что он сочетает в себе такие качества, как наибольшая популярность и распространенность, простота в пользовании и высокая производительность.

Также iSCSI широко используется в области виртуализации [2], например, является одним из стандартных протоколов, используемых VMware ESX при работе с удаленным хранилищем данных.

Глава 2

Постановка задачи

2.1 Описание протокола

iSCSI — это сквозной (end-to-end) протокол, который базируется на протоколе TCP/IP (Transmission Control Protocol /Internet Protocol) и разработан для для управления и взаимодействия систем хранения данных, серверов и клиентов.

В iSCSI принята своя терминология для субъектов взаимодействия. Полный список терминов и используемых сокращений можно найти в официальной документации [1]. Приведем лишь основные и наиболее часто используемые в работе понятия:

1 Инициатор (Initiator) — клиент, который запрашивает доступ к хранилищу данных и к самим данным

2 Таргет (Target) — сервер, который организует все взаимодействие и имеет непосредственный доступ к данным

3 LU (Logical Unit) — устройство для хранения данных

4 LUN (Logical Unit Number) — адрес устройства в сети хранения

iSCSI строится на двух широко используемых протоколах: SCSI (Small Computer System Interface) — протокол для физического подключения и обмена данными между компьютером и периферийными устройствами (в данном случае хранилищем) и IP — сетевой протокол, который на сегодняшний день наиболее широко используется в сетях Ethernet. Использование первого протокола обеспечивает совместимость с существующими решениями в операционных системах и приложениях, а второго — передачу команд в глобальном масштабе (сеть Интернет). Полный стек используемых протоколов в процессе взаимодействия отображен на рисунке 2.1.

Рис. 2.1: Стек протоколов iSCSI

2.2 iSCSI цикл

Более детально полный процесс взаимодействия инициатора и таргета рассмотрен в [3] и [4]. Остановимся лишь на основных моментах, важных для составления общей картины.

Все начинается с того, что инициатор ищет таргет, к которому хочет подключиться. В терминологии iSCSI это процесс называется discovery (обнаруже-

ние). Для упрощения этого процесса используется преимущество глобальной адресации протокола IP. Все устройства iSCSI обладают идентификаторами двух типов: iSCSI-имя и iSCSI-адрес. Все таргеты и инициаторы получают постоянные имена, которые указывают на конкретное устройство независимо от местоположения или IP-адреса. iSCSI-адрес, в свою очередь, состоит из IP-адреса, номера порта (3260 — общепринятый порт для iSCSI) и имени устройства.

Далее следует процесс аутентификации. Существует два основных способа ограничения доступа к таргету. Первый заключается в том, что сам таргет можно сконфигурировать таким образом, чтобы он мог принимать подключения только от определенных инициаторв или, наоборот, от всех, кроме некоторых. Изначально эти списки заносятся в тергет при настройке и при необходимости могут быть изменены в процессе работы. Второй способ заключается в использовании CHAP (Challenge Handshake Authentication Protocol) аутентификации, поддержка которой заложена в протоколе. В зависимости от требуемой политики безопасности можно выбрать один из следующих уровней безопасности: manual CHAP authentication (таргет и инициатор проверяют друг друга и у каждого инициатора свой отдельный секретный ключ), one-way CHAP authentication (только таргет проверяет подлинность инициатора) или не использовать данный метод вовсе.

После следует процесс обмена параметрами взаимодействия в текстовом виде (по типу: “ключ=значение;”), которые каждый из субъектов парсит и запоминает в данные сессии. Так, например, инициатор изначально узнает об устройстве, которое экспортируется таргетом, и его параметрах.

Затем следует сам процесс работы с данными. Все взаимодействие происходит с помощью PDU (Protocol Data Unit) различных назначений, которые являются базовыми единицами обмена данными во всем iSCSI-цикле. Каждый такой unit, в свою очередь, состоит из заголовка BHS (Basic Header Segment), дополнительного заголовка AHS (Additional Header Segment) в случае его необходимости и данных, которые необходимо передать таргету, опять же в

случае их необходимости (например, для операций на запись). Вся информация о том, что с этими данными необходимо сделать, содержится в заголовках. После обработки требуемых данных таргет формирует ответ, который строится аналогичным образом, и отправляет его инициатору.

Когда все необходимые операции выполнены и клиент инициирует завершение работы, происходит отправление соответствующего запроса на таргет. Это необходимо для корректного освобождения выделенной под сессию памяти, очистки всех кешей и сохранения данных на диске.

В случае необходимости инициатор может вновь восстановить соединение и тогда все вышеописанные операции повторяются. Полная схема взаимодействия со стороны инициатора и со стороны таргета представлена в виде конечных автоматов на рисунках 2.2 и 2.3 соответственно [5].

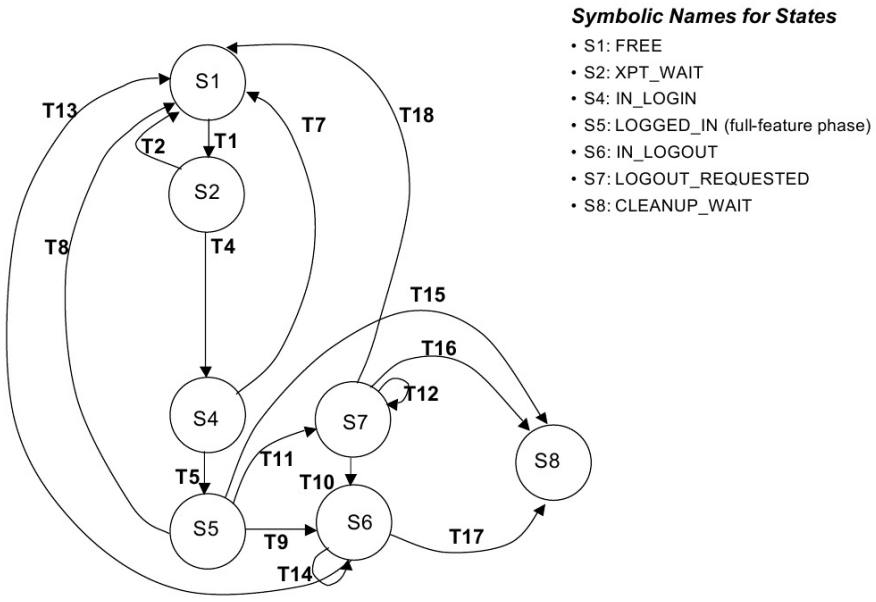

Рис. 2.2: Диаграмма состояний инициатора

Все вышеперечисленные характеристики позволяют находить iSCSI применения в различных способах организации SAN (Storage Area Network — сеть хранения данных) и вытеснять системы, основанные на Fibre Channel [6]. Существует множество подходов к подключению сервера: использование NIC (network interface controller — сетевая карта) с драйвером iSCSI, TOE (TCP

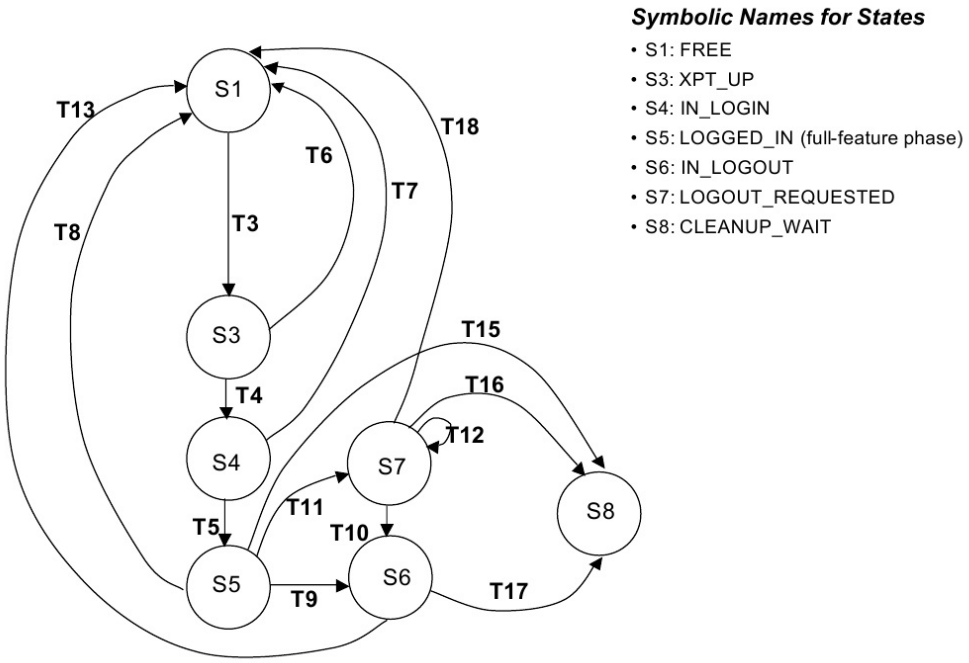

Рис. 2.3: Диаграмма состояний таргета

Offload Engine — механизмы разгрузки TCP) NIC с драйвером iSCSI или, как поступают традиционные производители FC адаптеров, HBA (Host Bus Adapter — адаптеры шины узла) — что позволяет выбирать лучший вариант в зависимости от необходимых параметров и ограничений. Если попытаться изобразить схему iSCSI SAN в самом общем виде, то получается следующая картина (см. рисунок 2.4).

Последнее, что стоит отметить — это преимущества iSCSI перед другими способами организации распределенных хранилищ данных (таких, как Fibre Channel, SCSI и тому подобные) [4]. Во-первых, это защищенность данных, так как для создания резервного копирования на значительном удалении от основной системы необходимы низкие стартовые вложения. Во-вторых, это возможность сведения разрозненных дисков в сети интернет в единый пул, что позволяет централизованно управлять ресурсами. В-третьих, это высокая доступность, так как множественные пути передачи IP-пакетов обеспечивают постоянное соединение даже в случае выхода из строя некоторых компонентов. И самое главное преимущество: возможность сохранения большого количества средств на построение новых специализированных сетей для

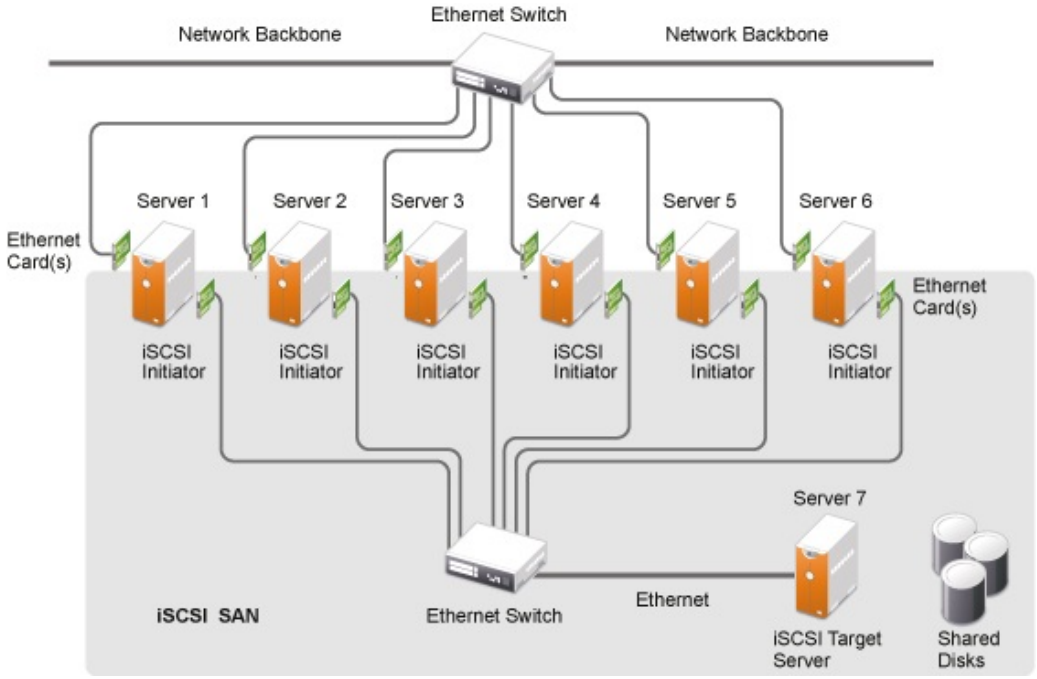

Рис. 2.4: Пример хранилища с использованием iSCSI SAN

обеспечения хранения данных за счет возможности использования уже имеющейся IP-инфраструктуры.

2.3 Постановка проблемы

Одним из наиболее важных параметров работы с хранилищем данных является скорость доступа к самим данным. Эта скорость ограничивается как аппаратными — сетевой картой, устройством сервера, так и программными — способом взаимодействия — характеристиками. Эти проблемы относятся, в частности, и к доступу к распределенному хранилищу с помощью iSCSI протокола. Все предыдущие исследования в этой области направлены исключительно на то, чтобы оптимизировать работу самого протокола [7] [8], и не решают фундаментальную проблему отсутствия параллельного доступа к хранилищу. Тема данной работы — исследование возможности параллельного доступа к распределенной системе.

При использовании нескольких LUN'ов решение достаточно просто и очевид-

но - экспортировать их большим числом таргетов, что приводит к разгрузке отдельных элементов системы. Проблемы начинаются в тот момент, когда все инициаторы захотят доступиться до одного LUN'а, соответственно будут обращаться к одному таргету. Поэтому конкретная цель данной работы — исследовать возможности паралелльного доступа к распределенной системе, представленной одним устройством (одним LUN'ом). Схематичное изображение исследуемой схемы представлено на рисунке 2.5.

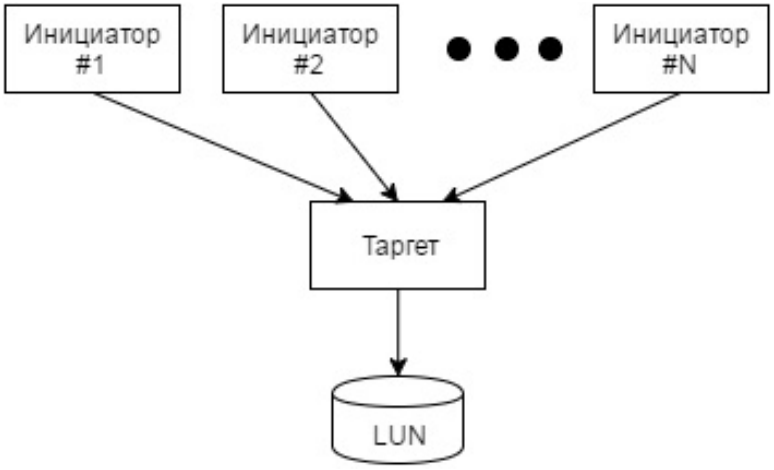

Рис. 2.5: Схема доступа к одному LUN'у

Для решения вышеописанной проблемы было предложено несколько подходов: multipath и stripping. Первый метод уже реализован во многих iSCSI клиентах, включая наиболее распространенные iSCSI клиенты в операционных системах Windows и Linux, а также в продукте компании VMware, поэтому задача — изучить существующие решения и исследовать их влияние на производительность.

Для решения задачи вторым методом необходимо реализовать систему, позволяющую нескольким таргетам, а не одному, как есть сейчас, корректно работать с одним разделом блочного устройства. Преимущество второго метода — вся логика выносится в серверную часть, поэтому для клиента нет никаких усложнений в работе, но как это скажется на сложности и производительности системы также подлежит исследованию.

Глава 3

Подход 1: Multipath

3.1 Описание технологии

Multipath Input/Output — технология подключения узлов сети хранения данных, в которой к каждому узлу (устройству) от центрального процессора идет несколько маршрутов. Например, в нашем случае — это подключение одного SCSI-диска к нескольким SCSI-контроллерам. Схематическое изображение системы с использованием multipath доступа приведено на рисунке 13.1.

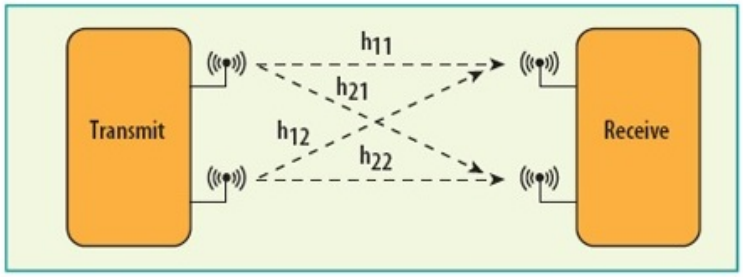

Рис. 3.1: Схема multipath доступа

Данная технология используется для повышения отказоустойчивости (в случае выхода из строя любого из устройств в одном из маршрутов, другие остаются в работоспособном состоянии), для увеличения производительности (можно посылать независимые запросы и получать ответы параллельно

по нескольким каналам) и для распределения нагрузки.

3.2 Исследование iSCSI multipath

Все вышеописанное относится и к системам, основывающимся на протокле iSCSI [9]. Одной из особенностей исследуемой технологии является вынос всей логики в клиентскую часть. В случае iSCSI это позволяет использовать любой таргет. Хотя в протоколе и не заложена поддержка multipath в явном виде, в нем также нет ничего, что могло бы помешать реализовать данный метод. Поэтому для исследования был взыбран Linux SCSI target framework (tgt), как наиболее популярный в использовании, с открытым исходным кодом (open-source) и простой в администрировании.

Если говорить об инициаторах, то во многих этот метод уже заведомо реализован и поддерживается. Это относится и к наиболее распространненым инициаторам: клиенты в операционных системах Windows и Linux, а также в продуктах компании VMware, которые и были выбраны для исследования. Поэтому основная задача — изучить существующие решения и исследовать их влияние на производительность.

Для моделирования была выбрана схема, представленная на рисунке 3.2.

Как выяснилось, сам по себе open-iscsi (стандартный Linux инициатор) не умеет подключаться к нескольким IP-адресам одного таргета — подобное подключение приводит к появлению нескольких блочных устройств, и инициатор не может самостоятельно определить, что это одно и то же устройство. Однако, нашлось простое решение — использовать дополнительную утилиту multipathd, которая помогает в поиске дисков с одинаковыми идентификаторами (UUID — Universally Unique Identifier) и обработке их таким образом, каким и положено в multipath технологии.

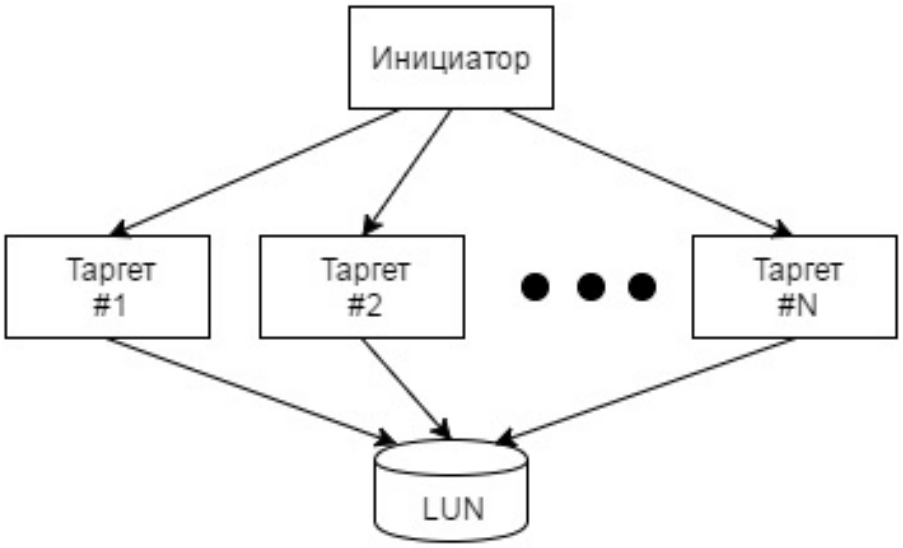

Рис. 3.2: Схема multipath iSCSI подключения

3.3 Результаты

Сравнительные результаты исследования использования multipath технологии для различных инициаторов приведены на диаграммах 3.3, 3.4.

Следует отметить, что к приведенным результатам не стоит относиться как к абсолютным величинам по ряду причин: для тестирования был использован обычный HDD 7200 об/мин, одна сетевая карта и один Ethernet кабель (1 Гбит/с), тестирование проводилось в локальной сети. Поэтому, это скорее сравнительные результаты производительности различных инициаторов.

Для более точных результатов необходима модернизация исследуемой системы — использование нескольких сетевых карт, чтобы было несколько не только виртуальных каналов связи, но и физических, и доступ к хранилищу, находящемуся вне локальной сети, чтобы принималось решение об оптимальных маршрутах и эффект гонки и непоследовательности доставки пакетов повлиял на скорости операций.

Однако, все равно видно, что при использовании исследуемой технологии производительность возрастает даже в случае "локального"тестирования, при-

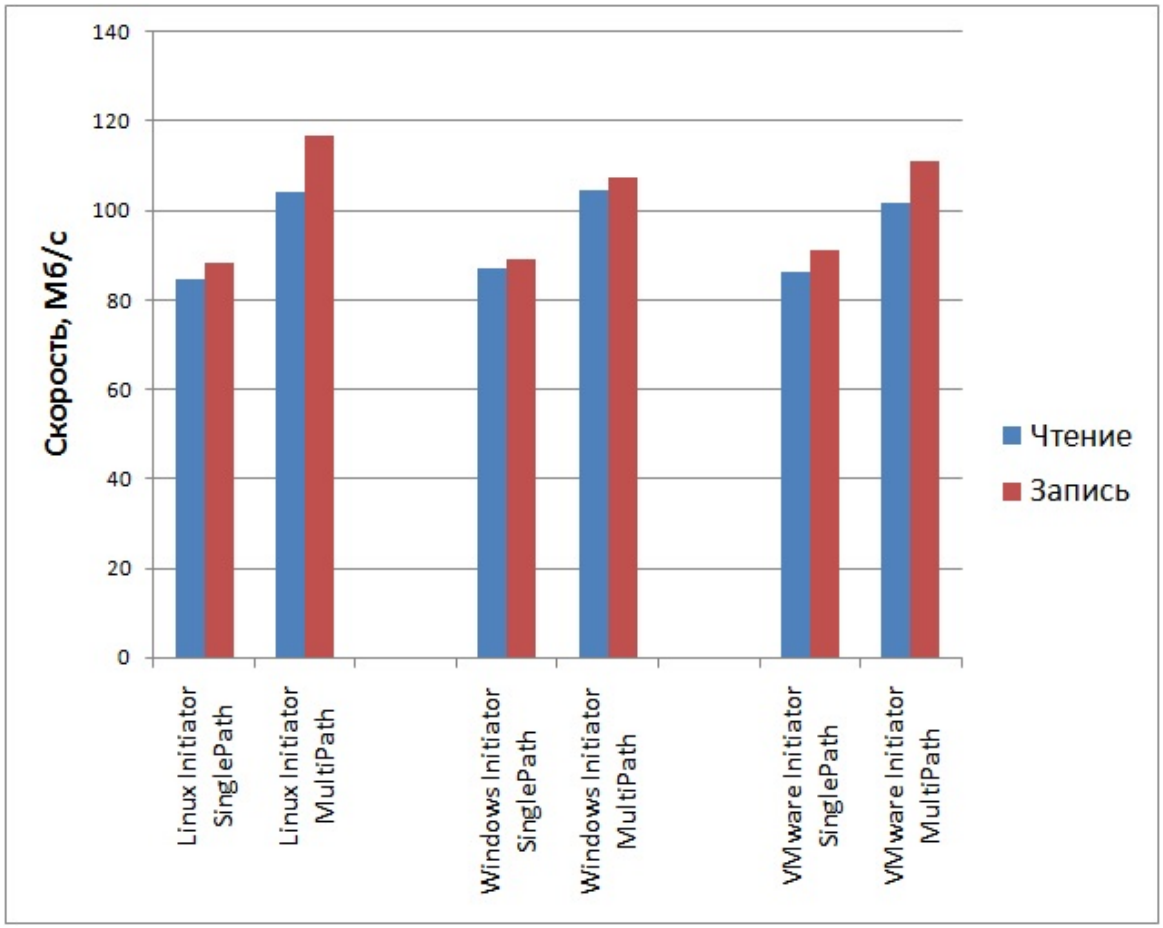

Рис. 3.3: Производительность multipath iSCSI (Мб/с)

чем достаточно заметно (порядка 20%).

Также возрастает и отказоустойчивость системы, так как появлется множество путей доступиться от инициатора к LUN'у, что в условиях большого количества узлов является важных фактором.

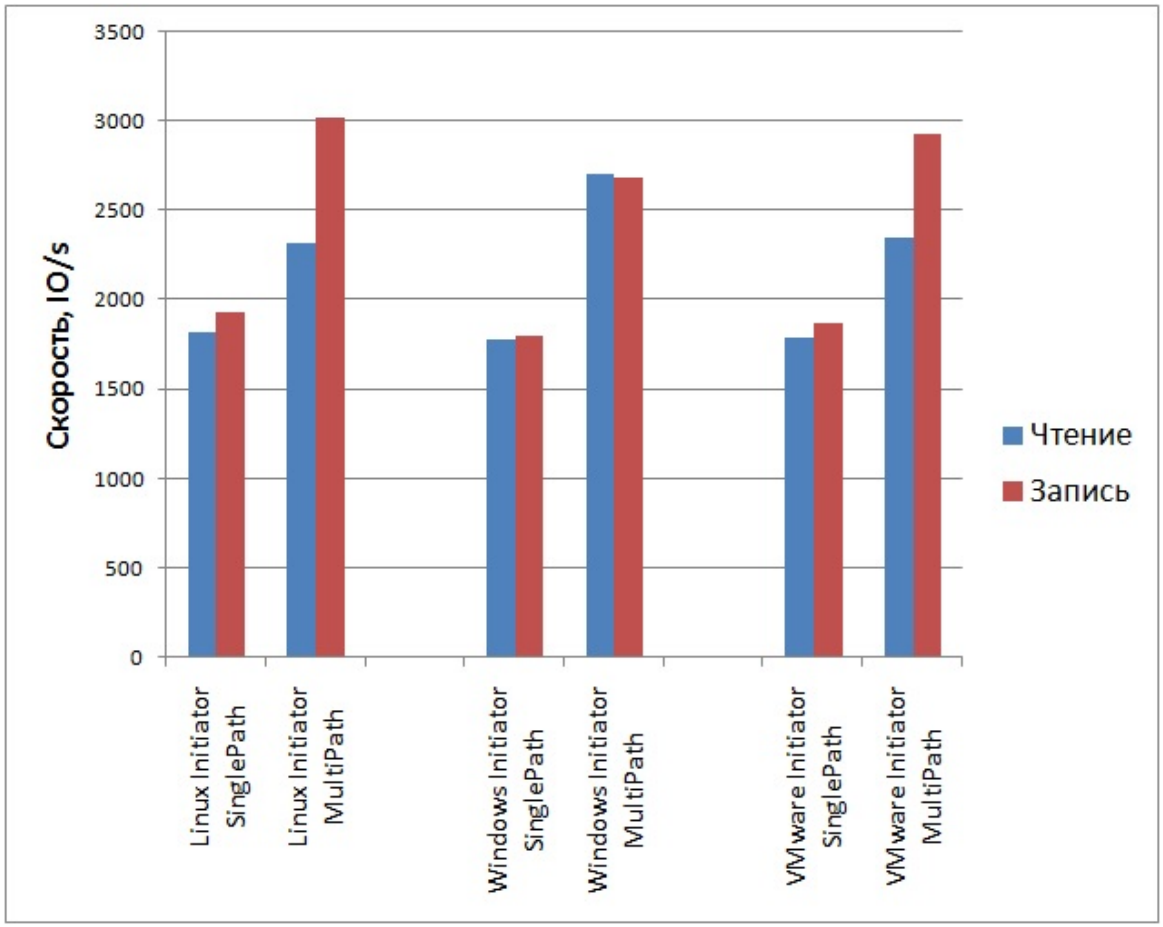

Рис. 3.4: Производительность multipath iSCSI (IO/s)

Глава 4

Подход 2: Stripping

4.1 Описание RAID технологии

Для работы с большими объемами данных (а значит, и с большим количеством устройств для хранения) часто бывает удобно объединять диски для достижения каких-либо целей. Для этого была введена технология виртуализации данных RAID (Redundant Array of Independent Disks — избыточный массив независимых дисков), которая объединяет несколько дисков в один логический элемент для достижения определенных целей (повысить надежность, отказоустойчивость, производительность).

Выделяют следующие базовые уровни RAID [10]:

1 RAID 0 (stripping) — распределение данных, то есть информация разбивается на блоки фиксированной длины и записывается на несколько дисков поочередно;

2 RAID 1 (mirroring) — клонирование данных, то есть информация записывается одновременно на несколько дисков, которые являются полными копиями друг друга;

3 RAID 2 — по сути тот же RAID 0, только вводятся дополнительные

диски для хранения блоков четности (кодов Хемминга) для повышения отказоустойчивости;

4 RAID 3 — по сути тот же RAID 2, только нужно меньше дополнительных дисков, что уменьшает перерасход, но препятствует коррекции ошибок на лету;

5 RAID 4 — по сути тот же RAID 3, только работает с блоками большего размера;

6 RAID 5 — по сути тот же RAID 4, только данные для коррекции находятся не на одном диске, а происходит чередование при выборе;

7 RAID 6 — по сути тот же RAID 5, только теперь данные коррекции хранятся не на одном из дисков, а на двух, которые, в свою очередь, также выбираются чередованием.

4.2 Описание stripping технологии

Для исследования была выбрана stripping (RAID 0) технология, поэтому на ней стоит остановиться несколько более детально [10]. Схема дискового массива с использованием данной технологии приведена на рисунке 4.1.

Процесс работы устроен следующим образом: информация разбивается на блоки фиксированной длины (Ai) и записывается на несколько дисков поочередно циклически. Считывать же данные можно в параллель с каждого диска независимо. Из этого вытекают следующие преимущества данной технологии: повышается пропускная способность последовательного ввода-вывода за счет одновременной нагрузки нескольких интерфейсов и снижается латентность случайного доступа за счет возможности параллельного доступа.

Но у этой схемы есть и очевидный недостаток: при выходе из строя хотя бы одного из дисков, теряется вся информация, что требует восстановления

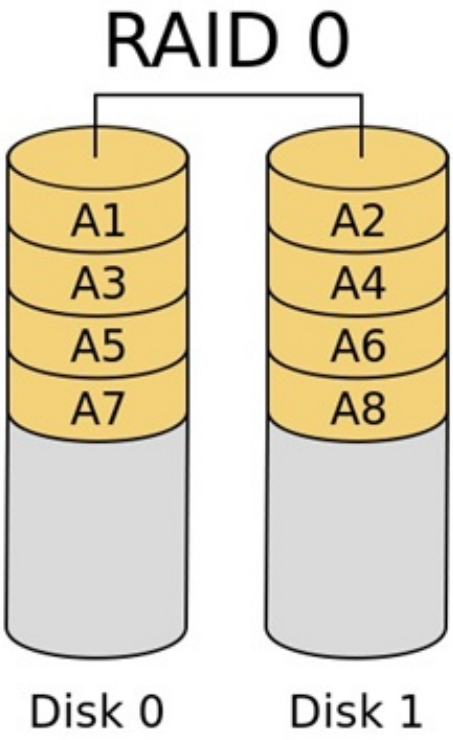

Рис. 4.1: Схема RAID 0 диска

данных с резервных носителей (например, может помочь использование более продвинутых RAID систем).

4.3 Проблема масштабируемости iSCSI

Как уже было упомянуто ранее, в настоящее время наиболее популярными способами организации iSCSI систем являются: 1) использование одного таргета, который экспортирует все необходимые устройства (в этом случае каждый инициатор теоретически может получить доступ к каждому из них), 2) использование одного таргета для каждого отдельного устройства (в этом случае для работы с несколькими устройствами необходимо подключаться к отвечающим за них таргетам). Схематичные изображения этих систем приведены на рисунках 4.2 и 4.3 соответственно.

Стоит отметить, что у каждой из представленных схем есть большой недо-

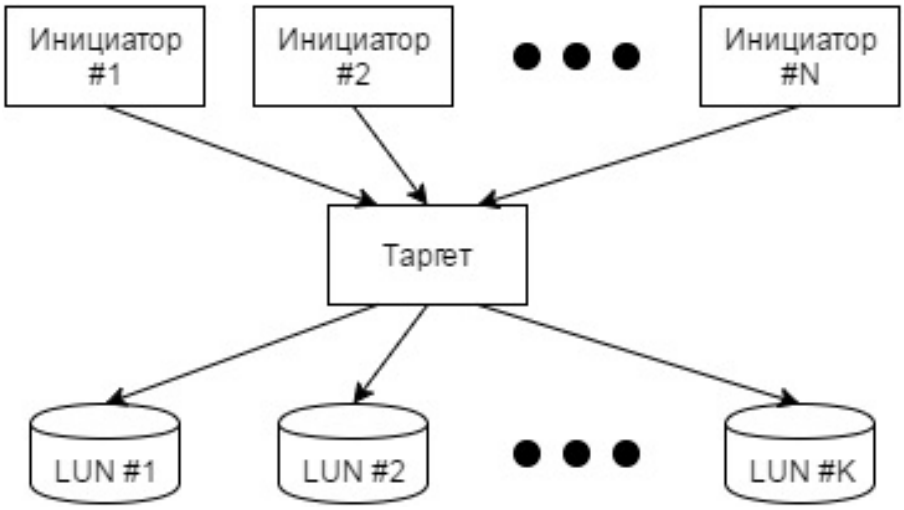

Рис. 4.2: Схема организации №1

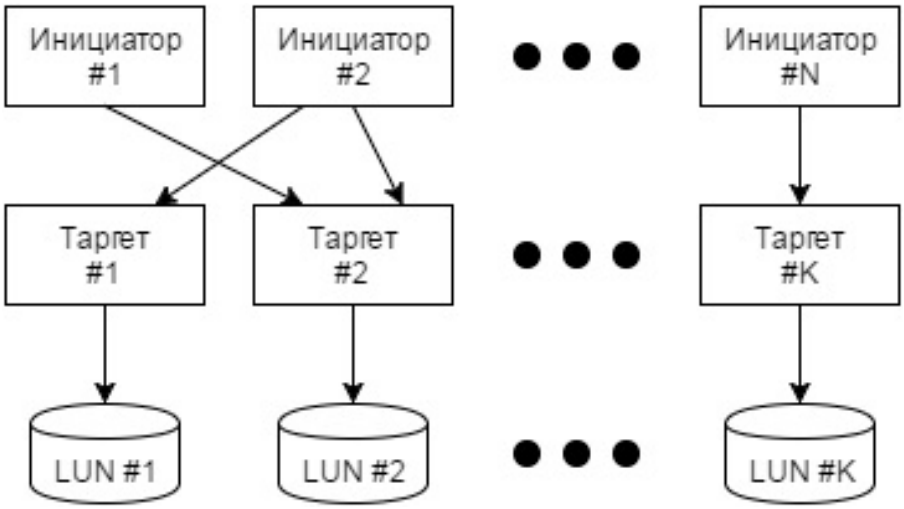

Рис. 4.3: Схема организации №2

статок — плохая масштабируемость, так как добавление нового устройства никак не улучшает работу с предыдущими. Более того, в первом случае у нас усложняется логика определения устройства, с которым инициатор хочет взаимодействовать, так как увеличивается список поддерживаемых устройств, а во втором возникают дополнительные накладные расходы на работу нового таргета. С этой проблемой удалось справиться следующим образом.

4.4 Архитектура решения

Для устранения вышеописанного недостатка протокола была предложена схема, изображенная на рисунке 4.4.

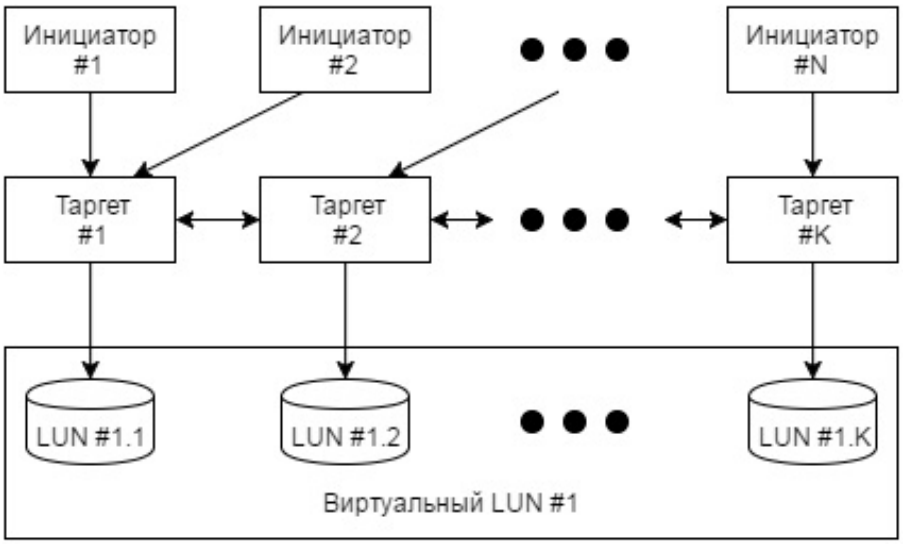

Рис. 4.4: Схема организации №3

Эту схему следует рассмотреть более детально. Вводится новый элемент — виртуальный LUN, который и отвечает за масштабируемость всей схемы. По сути, виртуальный LUN — это набор обычных LUN'ов, организованных методом stripping, то есть это реализация RAID 0 хранилища, за одним исключением, что здесь нет одного контроллера, который бы определял, на каком из дисков нужно искать интересующий инициатора блок. В качестве контроллера выступает набор из таргетов, каждый из которых, в свою очередь, отвечает за работу с одним из дисков из массива.

Стоит отметить следующий важный момент: таргетов много, и к любому их них может подключиться инициатор для доступа к данным, поэтому, во-первых, необходимо, чтобы каждый из таргетов знал, какой виртуальный LUN экспортирует вся система. Во-вторых, если инициатор обращается к таргету (номер 1) и просит данные с устройства, за который отвечает другой таргет (номер 2), то первый не должен отправлять инициатора ко второму, а должен сам запросить эти данные и переправить их инициатору. Это условие

сделано для того, чтобы вся необходимая для организации схемы логика была вынесена исключительно в работу таргета, а инициаторы ничего об этом не знали, то есть чтобы у стандартных клиентов была возможность работать с подобного рода хранилищами.

Получается, что для достижения поставленной цели необходимо решить задачу синхронизации этого "распределенного" RAID контроллера. Для этого было решено ввести новую iSCSI команду по типу iscsi_login (для процесса подключения) или iscsi_data (для передачи данных). В ней хранится информация об операции, которую необходимо осуществить (запись или чтение данных) и то, с какими данными необходимо произвести эту работу (их длину и сдвиг внутри устройства, по которому они находятся).

Процесс работы в случае ситуации, рассмотренной выше, строится следующим образом: если таргет понимает, что от него требуют данные с устройства, за которое отвечает второй таргет, то первый отправляет эту команду второму и ждет ответа (в случае необходимости). Второй, в свою очередь, детектирует новую команду и работает с ней следующим образом — достает длину и сдвиг напрямую из команды (а не в процессе раскрутки и вычислений как в обычном цикле), выполняет необходимую операцию и отправляет результат, если он подразумевается в выполненной операции.

Теперь детально рассмотрим, как работа с новой командой межтаргетного взаимодействия интегрируется в обычный цикл обработки. На вход таргета приходит запрос от инициатора, который парсится, обрабатывается и доходит до момента обращения к устройству для выполнения полученной операции. Далее происходит дополнительная обработка — по длине и сдвигу внутри виртуального LUN'а вычисляются длины и сдвиги внутри реальных устройств, на которых хранятся данные. Напомним, что данные заполняются в дисковом массиве с помощью stripping метода, поэтому зная размер одного блока, который является конфигурационным параметром схемы, можно вычислить, с какими блоками предстоит работа, а зная какой таргет отвечает за какой спектр блоков, можно определить, какому из таргетов мы должны

передать этот запрос с помощью нововведенной команды.

Стоит обратить внимание еще на один момент: если происходит необходимость обращения к нескольким таргетам, то вся логика по распределению и сбору данных заложена именно в первом таргете, чтобы упросить весь процесс синхронизации и не вмешиваться в работу остальных таргетов.

Последнее, чему нужно уделить особое внимание — это организация виртуального LUN'а, что включает в себя его определение таргетом и возможность экспортировать, причем реальный LUN, о котором таргет также должен знать, должен быть скрыт от инициатора, чтобы избежать работы с ним напрямую и, соответственно, потери или порчи данных.

Первая задача решается следующим образом: когда таргет понимает, что администратор хочет добавить ему поддержку виртуального LUN'а, создается объект, который содержит информацию о виртуальном устройстве (размеры, дескрипторы и тому подобное), который добавляется в список экспортируемых LUN'ов. Если детектируется обращение именно к этому виртуальному устройству, то запускается дополнительная вышеописанная обработка и все операции либо перенаправляются к другим таргетам, либо обращение происходит к скрытому от инициатора LUN'у — части виртуального, иначе процесс ничем не отличается от обычного цикла обработки. Скрытие же (вторая задача) решается просто выделением подобных LUN'ов в дополнительный список, чтобы с ними можно было работать, но искать их приходилось иным способом.

4.5 Тонкости реализации

Теперь обратимся к реализации вышеописанного решения. Для этого был снова выбран Linux SCSI target framework (tgt), так как он является проектом с открытым исходным кодом (open-source), что позволяет свободно использовать его код и дополнять всем необходимым.

Для добавления виртуального LUN'а было решено использовать следующую схему: сначала с помощью команды tgtadm добавляется реальное устройство, с которым будет работать именно этот таргет, которое будет скрыто от инициаторов, и его дескриптор сохраняется для дальнейшего доступа к устройству. Далее аналогичным образом добавляется виртуальное устройство, которое детектируется по некоторому заранее определенному имени (например, "virt_dev"). Информация о виртуальном устройстве может быть либо задана изначально (в коде системы), либо подгружаться из дополнительных ресурс-файлов.

Теперь рассмотрим работу таргета, который принимает запросы от инициатора. Для минимизации влияния вносимых изменений в стабильную работу таргета было решено делать это в самый последний момент - непосредственно перед выполнением операции с устройством (метод bs_rdwr_request). Для определенности скажем, что были переопределены исключительно операции чтения-записи.

Операция чтения теперь выполняется следующим образом: сначала происходит пересчет длины и сдвига внутри виртуального устройства на длины и сдвиги внутри реальных устройств, за которые отвечают различные таргеты. Затем происходит проверка: если мы должны что-то прочитать со своего устройства, то читаем. Если же нам необходимо получить данные с других таргетов тоже, то на них отправляются соответствующие запросы и, после, принимаются ответы с запрашиваемыми данными. И в конце, когда все данные собраны, происходит соединение этих данных в один цельный блок с помощью обратного пересчета длин и сдвигов.

Операция записи выполняется аналогичным образом за одним исключением, что сначала происходит не только пересчет, но и разделение данных, и что после отсылки этих данных нет необходимости ждать ответов — нить просто завершает эту операцию и переходит к следующей.

И последний важный момент, который следуют рассмотреть — это работа таргета, который принимает запросы от другого таргета. Так как подклю-

чение одного таргета к другому ничем не отличается от подключения инициатора к таргету, то работа вся происходит на общих основаниях. Поэтому нужно уметь отделять запросы от таргетов и запросы от инициаторов, что делается следующим образом. Если в момент получения сообщения (do_recv) детектируется команда для межтаргетного взаимодействия, введеннаю выше, то данные не идут в дальнейший цикл обработки запроса, а помещаются в очередь запросов от различных таргетов. Изначально, по аналогии с уже существующей bs (backing store) системой tgt, создается новая группа нитей, которая выполняет операции от других таргетов. Эти нити берут операции из очереди, выполняют их и отправляют ответ в случае необходимости (то есть в случае операции чтения).

Также был несколько расширен уже имеющийся функционал tgt для работы с сетью (iscsi_transport), что позволило добавить важное ограничение на работу с одним таргетом только одной нити.

4.6 Результаты

Самое главное, что было получено — это прототип схемы. Это показывает, что лишь немного усложнив таргета, можно добиться лучшей масштабируемости системы и использовать дополнительный функционал виртуальных хранилищ. Но все же стоит исследовать влияние нововведений на производительность цикла взаимодействия. Среднее время работы операций записи и чтения представлены на рисунках 4.5 и 4.6 соответственно.

Вновь следует отметить, что к этим результатам не стоит относиться как к абсолютным величинам, так как тесты производились в локальной сети и с использованием всего двух физических машин, на которых использовалась система виртуальных машин (Virtual Box) и контейнеров (OpenVZ) для моделирования работы реальных таргетов и инициаторов. Для получения более точных результатов необходимо строить системы из реальных машин, лежащих за пределами локальной сети, и физических каналов связи.

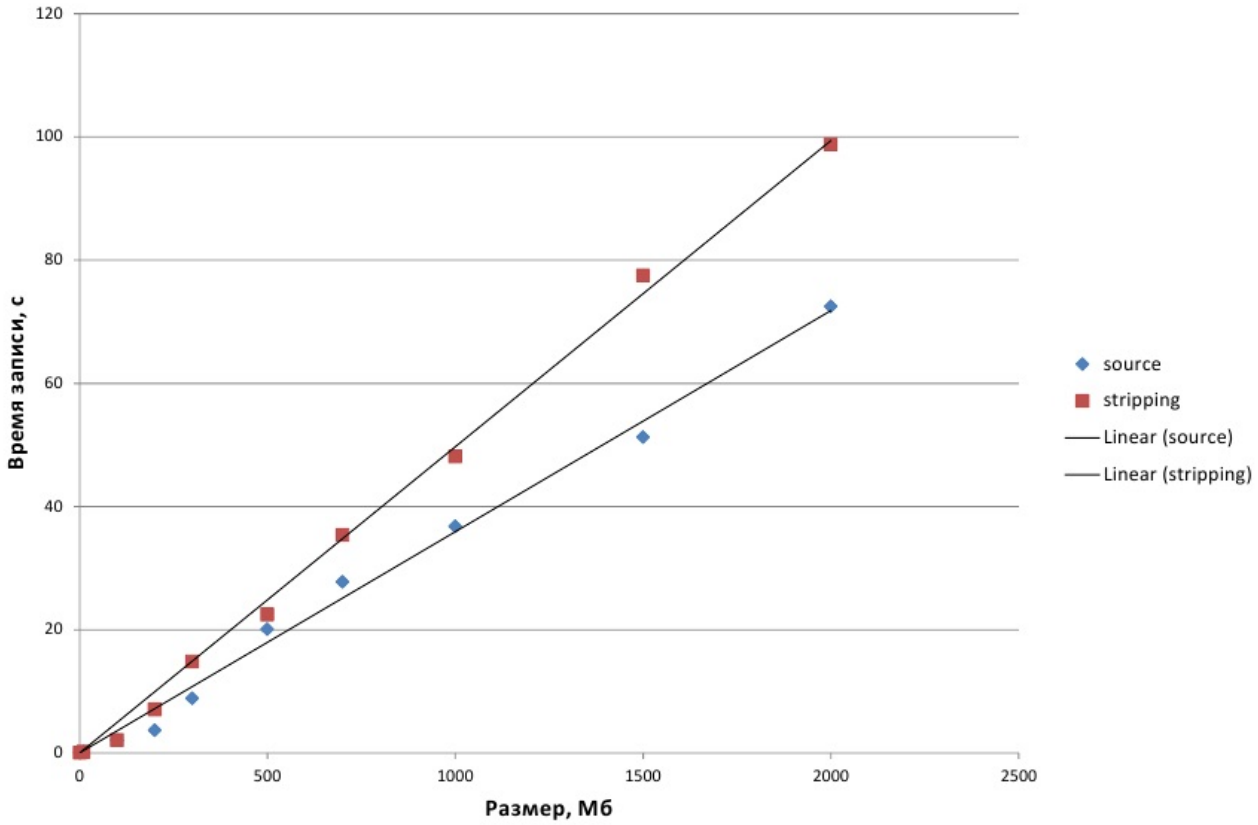

Рис. 4.5: Время выполнения операции записи

Еще одно важное отличие тестируемой системы — это использование виртуальных жестких дисков в качестве экспортируемых LUN'ов. Это сделано из-за потребности в большом количестве различных устройств, которые должны объединиться в один виртуальный LUN, и тоже является одной из основных составляющих при оценке полученных результатов.

Однако, стоит обсудить получившиеся результаты измерений. Из графиков видно, что затраты на пересылку сообщений между таргетами достаточно сильно сказываются на всей производительности системы. В современных реалиях же используются не локальные диски в качестве хранилищ, а удаленные, причем доступ к ним осуществляется по менее производительным каналам (порядка 100 Мб/с), а таргеты можно соединить, например, 10 Гб/с Ethernet кабелем. Это должно значительно уменьшить влияние времени межтаргетной пересылки по сравнению со временем доступа к хранилищу данных напрямую.

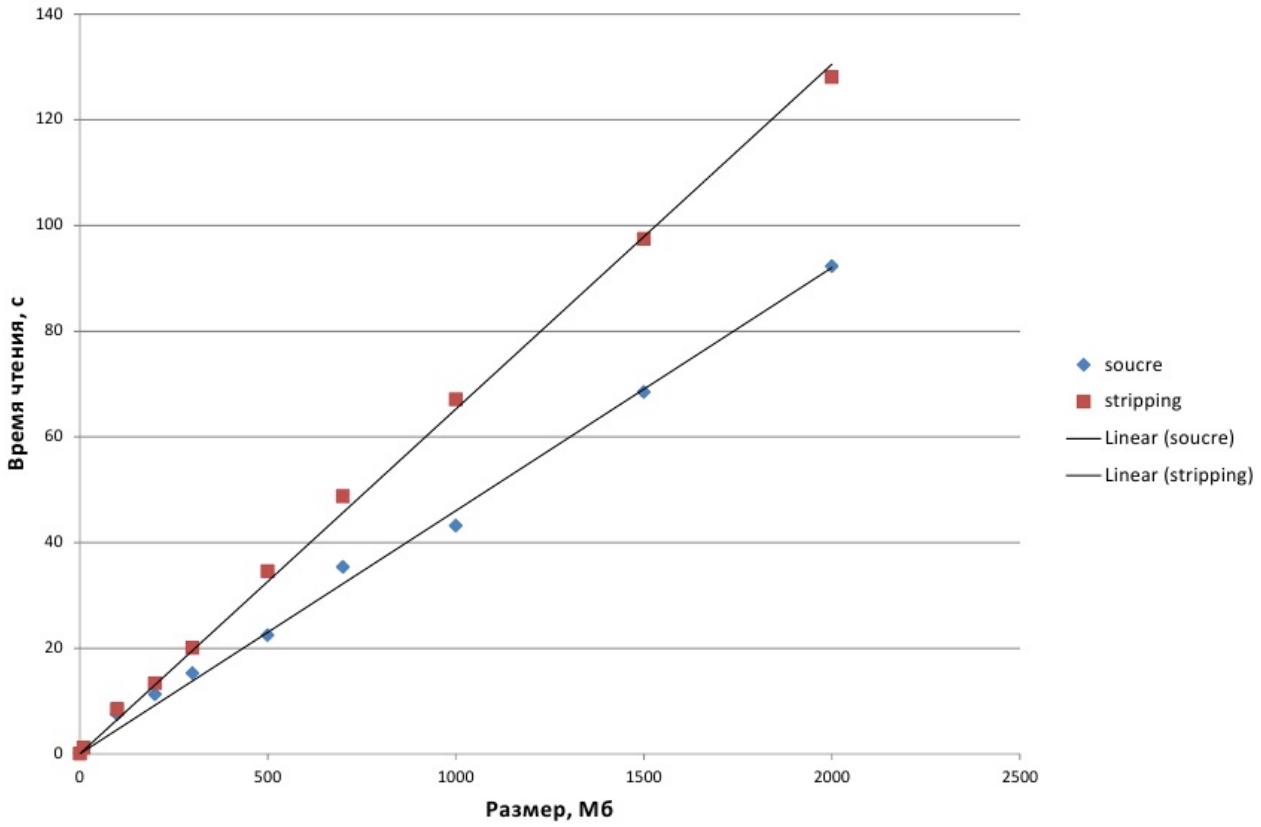

Рис. 4.6: Время выполнения операции чтения

Также стоит отметить еще одну интересную особенность: среднее время операции чтения значительно выше (порядка 30%) среднего времени на запись. Это объясняется тем, что операция на запись происходит только в одну сторону (данные инициатором отправляются на таргет, там они при необходимости разделяются и записываются на диски), а при чтении есть и обратный путь (посылается запрос, данные считываются и отправляются обратно, а все это время инициатор находится в ожидании).

Глава 5

Заключение

Таким образом, в работе была исследована проблема параллельного доступа к одному дисковому устройству (LUN'у). А именно, была исследована возможность использования multipath технологии в таких наиболее популярных инициаторах как Linux, Windows и VMware Initiator'ы и получены сравнительные результаты увеличения производительности взаимодействия.

Также было предложено новое решение для поставленной проблемы, основанное на использовании нескольких связанных таргетов и stripping технологии для создания одного виртуального LUN'а. Для этого метода также были посчитаны и проанализированы времена выполнения основных операций при работе с распределенным хранилищем.

В дальнейшем планируется попытаться использовать более продвинутые RAID технологии для решения проблемы отказоустойчивости, которая проявляется при использовании stripping'а, а также провести тестирование на реальных системах, что позволит точно определить, в каких случаях предложенные решения имеет смысл использовать при организации системы распределенных хранилищ.

Литература

[1] Julian Satran, K Meth, C Sapuntzakis, Mallikarjun Chadalapaka, and E Zeidner. Iscsi. Technical report, RFC 3720, April, 2004.

[2] Yaron Klein. Storage virtualization with iscsi protocol. Technical report, Internet Draft, XP-015030964, 2000.

[3] Kalman Z Meth and Julian Satran. Features of the iscsi protocol. *IEEE Communications Magazine*, 41(8):72–75, 2003.

[4] Kalman Z Meth and Julian Satran. Design of the iscsi protocol. In *Mass Storage Systems and Technologies, 2003.(MSST 2003). Proceedings. 20th IEEE/11th NASA Goddard Conference on*, pages 116–122. IEEE, 2003.

[5] Mallikarjun Chadalapaka. iscsi state diagrams. *Networked Storage Architecture, NSSO, Rev 0.7*, 2002.

[6] Stephen Aiken, Dirk Grunwald, Andrew R Pleszkun, and Jesse Willeke. A performance analysis of the iscsi protocol. In *Mass Storage Systems and Technologies, 2003.(MSST 2003). Proceedings. 20th IEEE/11th NASA Goddard Conference on*, pages 123–134. IEEE, 2003.

[7] Abhijeet Joglekar, Michael E Kounavis, and Frank L Berry. A scalable and high performance software iscsi implementation. In *FAST*, volume 5, pages 267–280, 2005.

[8] Shruti Kulkarni and KN Narasimha Murthy. A survey of optimizing the performance of iscsi.

[9] W Tody Ariefianto, Yudha Purwanto, and Hendra Wiratama. Storage area network based-on internet small computer standard interface optimization using internet protocol multipathing. In *Information and Communication Technology (ICoICT), 2013 International Conference of*, pages 303–307. IEEE, 2013.

[10] Bill Dawkins and Arnold Jones. Common raid disk data format specification. Technical report, Technical report, SNIA Technical Position, 2007.