Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра системного программирования

Максименко Дмитрий Сергеевич

Разработка серверного программного обеспечения для управления системами хранения данных

Курсовая работа

Научный руководитель:
д. ф.-м. н., профессор Терехов А. Н.

Санкт-Петербург
2020

Оглавление

Введение

Системы хранения данных - это программно-аппаратное решение для хранение данных, предоставляющее высокую скорость чтения и записи. Системы хранения данных начали распространяться более тридцати лет назад, поэтому работающие на данный момент СХД различаются между собой по архитектуре, версиям используемых протоколов и программным обеспечением, установленным непосредственно на хранящие единицы. Управление СХД происходит посредством подачи с сервера (хоста) контролирующих команд на систему хранения данных (Рис. 1) напрямую или через специальный интерфейс.

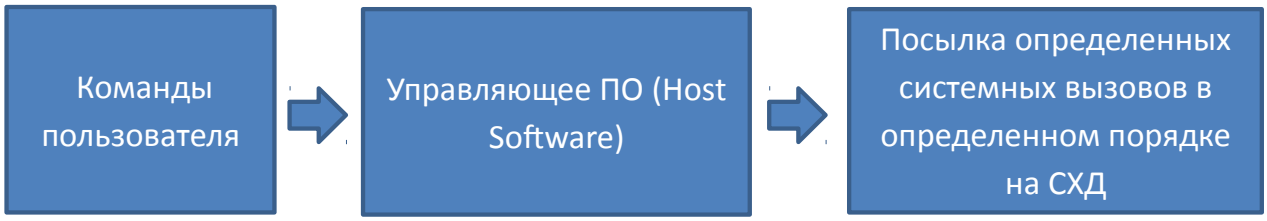

Рис. 1: Принцип преобразования команд

Ввиду разнообразия используемых решений, создание серверного программного обеспечения весьма трудоемкий процесс. Необходимо переписывать системные вызовы в зависимости от архитектуры, бизнес-логики и других параметров конкретной СХД. Для оптимизации этого процесса нашей командой ведется разработка декларативного языка программирования для описания бизнес-логики и ее подстановки в рабочий код (паралелльно разрабатывается библиотека на языке Python с аналогичным функционалом), а также эмулятора СХД для проведения тестов. Кроме того, для корректной работы СХД серверу необходима актуальная информация о структуре СХД (Рис.2), которая меняется при исполнении команд записи. Существуют разные способы представления этой структуры. В рамках данный работы мною разработан прототип плагина, позволяющего избежать избыточного запрашивания структуры.

1. Постановка задачи

Данная работа посвящена созданию набора инструментов, упрощающих разработку серверного программного обеспечения для управления системами хранения данных, а также оптимизации хранения и изменения конфигурацией СХД. Для достижения цели мною будут решены следующие задачи:

- изучить предметную область и провести обзор работающих в промышленности решений;
- создать эмулятор системы хранения данных с возможностью изменять структуру моделируемой СХД;
- придумать оптимальный способ хранения информации о струтуре СХД;
- реализовать плагин для управляющего сервера, способный хранить и динамически изменять информацию о структуре СХД, и на основе этого проверять корректность команд;

2. Обзор существующих аналогов

Существует много, как различных по структуре, используемым протоколам управления систем хранения данных, так и различных программных решений (Host software) для управления ими. Так, компания PureStorage предоставляет продукт flasharray [4] — СХД, полностью основанную на flash-памяти. А компания DELL подразделяет свои СХД на 4 части по типу доступа к данным [1]. СХД, использующие SAN (FiberChanel) [8], ISCSI [9], SAS [7] и NAS [5]. Такое разнообразие объясняет отсутствие каких-либо инструментов, которые могут быть использованы для оптимизации разработки Host Software для СХД. Прямых аналогов разрабатываемому в рамках этой работы продукту не существует, но есть различные разработки, аналогичные частям разрабатываемой платформы. Например, многие компании производители СХД, предоставляют эмуляторы СХД. Компания ALTEN предоставляет эмулятор своей СХД [3]. Он позволяет тестировать управляющее ПО с различными конфигурациями СХД, масштабировать моделируемую СХД, а также тестировать поведение при искусственно сгенерированных ошибках. Не позволяет изменять форматы управляющих команд (такие, как длина заголовка).

3. Основные понятия предметной области

Системы хранения данных являются сложным и дорогим оборудыванием. Их техническим оснащением занимаются целые компании, подходы которых имеют некоторые различия. Тем не менее для оценки прототипов, разработанных в рамках этой курсовой, достаточно лишь некоторого приближения, описывающего основные процессы в СХД. Данное приближение основано на принципах работы дисковых массивов компании DELL "EMC Symmetrix" [2].

СХД представляет собой набор из 20-30 дисковых массивов, где каждый массив содержит 30-40 тысяч устройств. Основной операцией является копирование с одного устройства на другое. Для того, чтобы произвести такое копирование, необходимо сначала создать пару между этими устройствами, указав устройство источник и устройство цель, а затем активировать копирование в этой паре. Нельзя копировать на устройство, которое уже участвует в копировании, но можно копировать с одного устройства на несколько. Именно на этапе создания пары осуществляются эти проверки.

Группа это пара из двух дисковых массивов. Один массив может быть не более чем в 255 группах. Копирование с устройства одного массива на устройство другого осуществляется в рамках какой-то из установленных групп. Группы и пары можно создавать и удалять соответствующими запросами.

У всех устройств есть внутреняя нумерация внутри массива, но лишь у нескольких внешняя. Такие устройства видны управлябщему серверу и именно на них поступает команда на соответствующий дисковый массив.

Управляющей сервер посылает команды (в этом контексте то же самое, что и запросы), состоящие из заголовков и (для некоторых команд) тела. СХД выполняет команду, если это возможно. Ответ СХД также состоит из заголовков и иногда тела. Если команду выполнить невозможно, то в заголовках возвращается информация об ошибке.

Нас не интересует, как именно СХД выполняет эти команды, но

известно поведение, которому она следует.

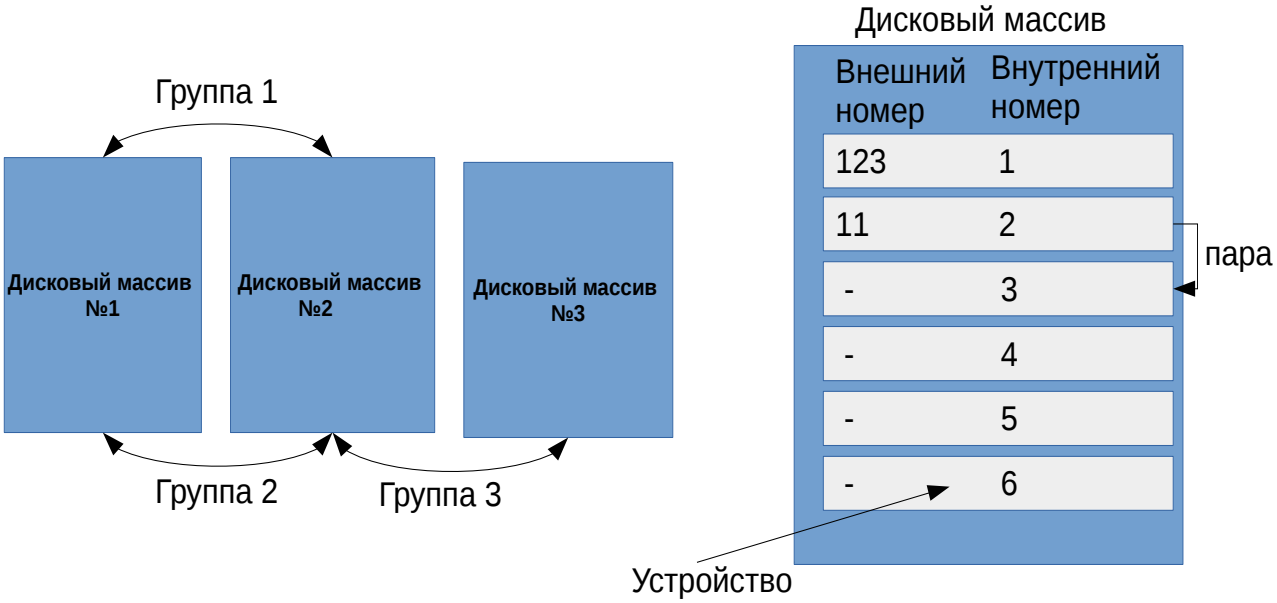

Рис. 2: Основные элементы структуры СХД

4. Эмулятор

4.1. Протокол

Одним из ключевых факторов, определяющих работу системы хранения данных или ее эмулятора, является протокол взаимодействия управляющего сервера и непосредственно системой хранения данных. Протокол включает в себя строго регламентированные команды, а также ответы на них и возможные ошибки. В рамках работы был разработан соответствующий протокол [4]. Он включает в себя 12 команд (Рис. 3), 34 возможные ошибки. Все описанные запросы составляют необходимый минимум для проверки работы гипотез, которые в будущем будут проверяться с использованием данного протокола. Кроме того, протокол не противоречит логике работы реальных СХД компании DELL, что говорит о том, что программы, работающие с использованием разработанного протокола, могут быть перенесены на реальную аппаратуру.

Ссылка на протокол: https://drive.google.com/file/d/1HAMVQAheQ2hKb3L

header	flags	start	cnt	cmd	fmt
30000002	00000000	00000000	00000000	0003	0000

Рис.3: Состав команды

4.2. Свойства

Эмулятор системы хранения данных, разработанный мною, представляет собой многопоточный сервер и клиентскую библиотеку, написанные на языке C++. Взаимодействие с сервером осуществляется через TCP сокет, и состоит из записи со стороны клиентской части в сокет определенной команды (Рис.3) и получения ответа на эту команду от сервера, также через запись в сокет.

Эмулятор реализует такую же структуру, как и СХД, и изменяет

ее в зависимости от поступающих команд. Внутреннее представление структуры:

- СХД (набор дисков с быстрым взаимодействием между ними. На практике, как правило, диски, физически находящиеся рядом);
- диски;
- треки дисков;
- пары между дисками (локальные и удалённые);
- группы между дисковыми массивами;

На данный момент существует возможность инициализировать эмулируемую структуру из текстового файла, что позволяет работать со структурой любых размеров, а также сохранять различные по свойствам структуры. Кроме того, написана программа на языке python, заполняющая соответствующий текстовый файл случайной структурой с указанными параметрами.

Для соотвествия поведения эмулятора и рельной аппаратуры, в эмуляторе предусмотрен файл с параметрами, характеризующими специальные простои, моделирующие задержку при выполнении соответствующих действий реальной СХД. Написанная мной клиентская часть может собираться в статическую библиотеку, что облегчает ее подключение в проект. Клиентская часть предоставляет функцию sendmessage (Рис. 4), которая принимает параметрами: команду, ip и порт эмулятора, что позволяет работать с эмуляторами удаленно. Таким образом от пользователя библиотеки скрыты все подробности взаимодействия по сети, такие как установка соединения, кодирования команды и прочее.

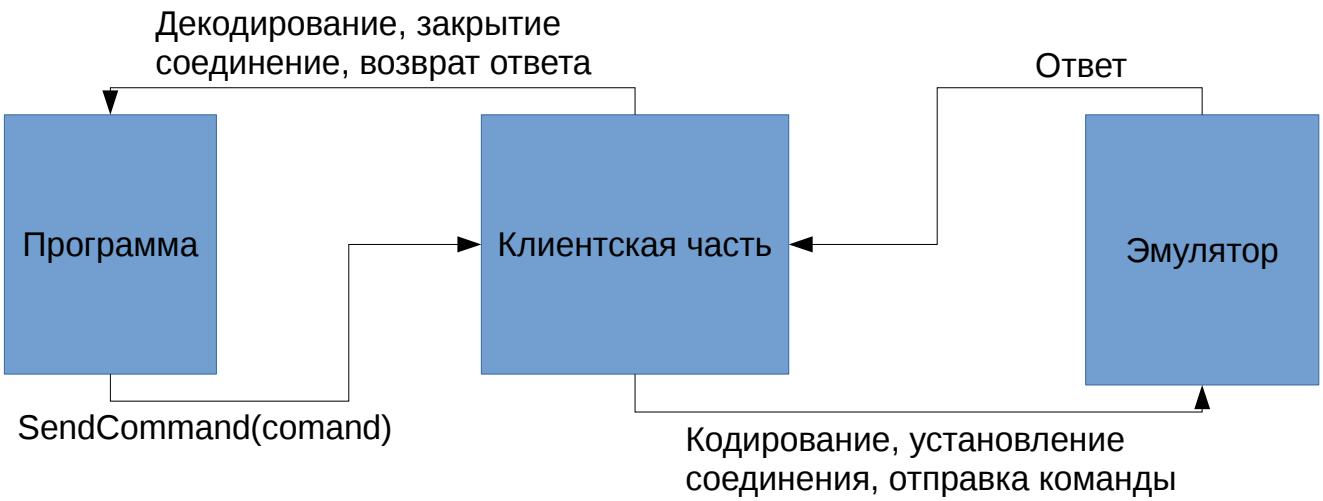

Рис 4: Принцип работы клиентской части

4.3. Применимость

Работа эмулятора состоит из следубщих этапов:

1. чтение и декодирование команды;
2. переход от команды-строки к команде-структуре;
3. проверка команды на исполнимость;
4. если команда выполнима, то передача параметров запроса классам, отвечающим за структуру;
5. изменение структуры согласно запросу;

6. при необходимости: сбор данных для ответа;

7. переход от ответу-структуры к ответу-строке;

8. кодирование и запись ответа;

Четкое разделение на эти этапы позволяет легко модернизировать эмулятор, расширяя функционал, добавляя новые детали структуры.

Наличие клиентской части позволяет любому программисту быстро начать работу с эмулятором, в том числе дистанционно.

На данный момент эмулятор уже использовался мною и еще двумя людьми, проводящими исследование в данной области. Запланированные далее исследования также будут восстребованы эмулятором такого рода.

5. Плагин для управляющего сервера

5.1. Требования

Сервер, управляющий системой хранения данных, занимается преобразованием пользовательских запросов в набор команд для СХД. Для корректной работы серверу необходима информация о структуре СХД (Рис. 2). Для получения полной структуры серверу необходимо передать сотни команд. Некоторые команды (например, создание пары между устройствами) меняют структуру СХД. В рамках курсовой работы реализована программа, которая может быть установлена на сервер. Программа оптимизирует работу сервера за счет проверки запросов на валидность, что осуществялется в два этапа.

Первый этап состоит в том, что программа генерирует и передает серверному программному обеспечению список запросов, необходимых для определения структуры СХД. На основе полученных ответов генерируется структура, которая затем поддерживается актуальной.

Второй этап состоит непосредственно в проверке запросов (Рис. 5). За него отвечает функция checkrequest, которая получает на вход запрос, который сервер собирается сделать к СХД. Затем проверяется, может ли запрос быть выполнен. Если запрос может быть выполнен, то функция возвращает ноль и изменяет структуру согласно этому запросу. Гаранитруется, что если серверное ПО передало запрос в плагин, то при получении нуля, запрос будет выполнен. Если запрос не может быть выполнен, то возвращется код ошибки согласно протоколу. Если плагин не может определить, может ли быть произведен запрос, то он возвращает минус один и список запросов, которые должны быть выполнены, чтобы дать точный ответ. При этом считается, что серверное ПО не будет выполнять запрос, валидность которого не определена.

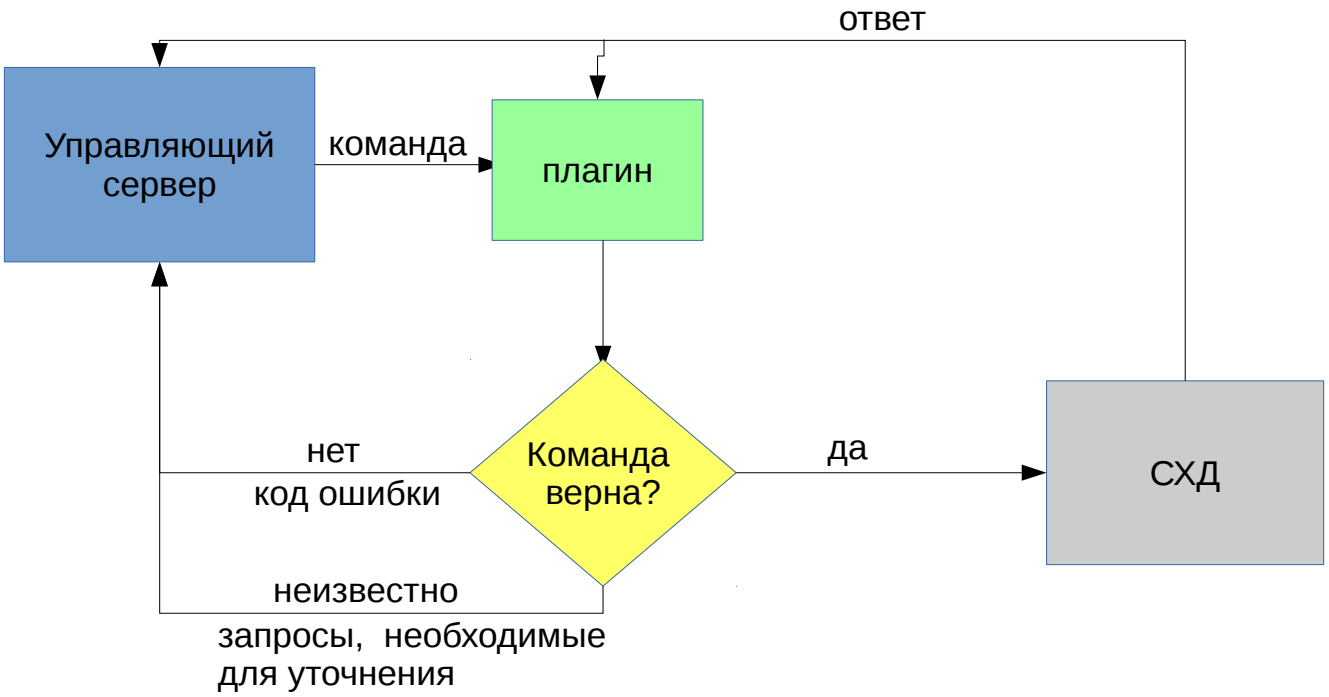

Рис 5: Принцип работы плагина

5.2. Синхронизация

Описанный ранее плагин является не более, чем прототипом, тем не менее крайне важным является продумывание решений проблем, которые могут возникать при попытке создания прикладных программ на основе данного прототипа. Одной из таких проблем является проблема синхронизации. Она состоит в том, что на реальной аппаратуре могут возникать ошибки, которые невозможно предсказать извне. Такие ошибки могут привести к несоотвествию структуры, сохраненной в плагине и структуры реальной СХД, что черевато серьезными простоями оборудывания, потому как при продумывании запросов сервер будет получать ”зеленый свет” от плагина и посылать запрос, который

не будет выполнен СХД.

У данной проблемы существует несколько решений, выбор между которыми во многом связан с параметрами аппаратуры, тем не менее в рамках работы было реализовано два из них: первый заключается в установке в плагине таймаутов, связанных с частями структур, относящихся к конкретным дисковым массивам и запуске этих таймеров после первой команды, изменяющей структуру в данном массиве. По истечении таймера структура, относящаяся к этому массиву, будет полностью перезапрошена.

Второй способ состоит в реализации специальной хэш-функции, которая принимает параметром структуру дискового массива. По таймауту плагин запрашивает хэши всех дисковых массивов СХД, а также вычисляет их на своей копии структуру, потом хэши сверяются, и при несовпадении хоть одного из них, вся структура перезапрашивается.

5.3. Представление структуры

Сервер может посылать к СХД сотни запросов в секунду, поэтому крайне важным является то, чтобы проверка запроса плагином не занимала много времени. При этом системы хранения данных могут состоять из десятков дисковых массивов с десятками тысяч устройств в каждом, что может породить сотни тысяч пар копирования и сотни групп. Наиболее частыми являются запросы создания, удаленная и активации локального и удаленного копирования. Это учитывалось при разработке методов хранения структуры. Структура может быть поделена на две части: часть, которая не меняется с течением времени и изменяющаяся часть. К первой относятся множества дисковых массивов и устройств в них. Ко второй — группы и пары. Основные действия, которые необходимо выполнить плагину при проверке запроса создания/удаления/активации копирования:

1. вычисление дискового массива, которому предназначен запрос, по номеру устройства, на который запрос отправлен (если массив не указан явно);

2. проверка статуса устройства, который является целью копирования;

3. проверки статуса устройства, с которого происходит копирование;

4. прочие проверки;

5. если запрос корректен, изменение структуры соответствующим образом;

Заметим, что при появлении ошибки на некотором шаге, дальнейшие шаги не выполняются.

Данные, необходимые для проведения шага 1 относятся к статическим данным, поэтому для их хранения был выбран отсортированный по номерам устройств массив. Это обеспечивает логарифмическую зависимость скорости поиска от количества устройств.

На 2 и 3 шаге используются как статические, так и динамические данные. Для оптимизации работы было использовано то, что не меняются номера устройств и их принадлежность дисковым массивам. Таким образом, для каждого массива создаются сбалансированные деревья с листьями соответствующими устройствам (внутренний номер устройства является ключом в дереве). В каждом листе содержиться вся информация, касающаяся этого диска. Таким образом, фактически сложность и поиска, и изменения статуса устройства логорифмически зависят от количества устройста в дисковом массиве.

Построение таких деревьев долгий процесс, но оно происходит только во время инициализации или синхроназации, то есть крайне редко. Изменение статусов устройств не влияет на деревья, а влияет только на поля структур. Сами структуры содержат два поля (при необходимости могут быть расширены технической информацией о диске, такой как вместимость).

- статус устройства, как цели копирования. Здесь находится ноль, если устройство не является целью копирования. Номер устройства, с которого происходит копирование, в противном случае.

- статус устройства, как источника копирования. Здесь находится указатель на список, содержащий подробную информацию о копировании, которое происходит с этого устройства (к ней относится группа, в которой идет копирование, активность копирования и тд).

Напомним, что общее количество устройств в СХД не превышает двух миллионов, таким образом, эта часть структуры помещается в несколько десятков мегабайт. Кроме того, к важным частям структуры можно отнести списки групп. У каждого дискового массива он свой. Так как каждый массив может участвовать только в 255 группах, эти списки не занимают много памяти, и поиск в них сравнительно быстрый.

Представленная схема хранения по максимуму использует статичность некоторых частей структуры, на реальных данных укладываясь в 100 МБ. При этом скорость обработки запросов достигает 30000 запросов в секунду. Тесты проводились на персональном компьютеры, данные на промышленной аппаратуре могут отличаться.

5.4. Альтернативные подходы

Важной задачей является дать оценку разработанному прототипу плагина, сравнить этот подход с существующим и альтернативными. Существующий сейчас подход заключается в частичном запросе конфигурации после каждого запроса, изменющего структуру СХД. Описанный плагин позволяет фактически свести эти запросы к запросам инициализации и синхронизации, что может уменьшить количетсво запросов на порядки (в зависимости от частоты проведения синхронизации). Но работа плагина требует затрат оперативной памяти (до 100 мб) и некоторой части работы времени процессора (размер части зависит от характеристик аппаратуры). Кроме того, некоторые сложности могут возникнуть при интеграции плагина в планировщик запросов, работающий на управляющем сервере. Таким образом, вопрос о рациональности использования плагина не может быть закрыт без инфор-

мации об аппаратуре.

К альтернативным подходам можно отнести плагин, идея которого в том, чтобы хранить структуру, поделенную на некоторые сектора, и помечать информацию о них, как недействительную, при отправке запроса, изменяющего структуру в этом секторе. При следующем запросе, опирающемся на этот сектор, необходимо будет запросить только информацию об этом секторе. Данный подход не реализован на данный момент, но планы по его реализации есть. Когда он будет реализован, появится возможность сравнить его с плагином, описанным в этой работе.

6. Инструментарий

При создании эмулятора и плагина для сервера был использован язык программирования C++, так как он сочитает в себе возможности объектно-ориентированного программирования, что необходимо для создания удобного в понимании и модификации эмултора и достаточный контроль памяти, чтобы создавать экономичные по памяти структуры, что необходимо при разработке плагина.

Для обнаружения утечек памяти, а также замера расходуемой памяти и контроля работы многопоточного эмулятора использовался набор инструментов valgrind[6].

Для организации сетевого взаимодействия использовалась библиотека сокетов ”sys/socket.h”.

Заключение

- предметная область изучена, проведен обзор работающих в промышленности решений;
- создан эмулятор системы хранения данных с возможностью изменять структуру моделируемой СХД;
- придуман оптимальный способ хранения информации о струтуре СХД;
- реализован плагин для управляющего сервера, способный хранить и динамически изменять информацию о структуре СХД, и на основе этого проверять корректность команд;

Ссылка на репозиторий: https://github.com/Dmitree-Max/NAS_emulator

Список литературы

[1] DELL. Data Storage Systems. — URL: http://dellexpert.ru/vybor-system-hranenya-dannyh-dell-ibm-lenovo-cxd.html (дата обращения: 11.02.2020).

[2] EMC DELL. EMC Symmetrix product guide. — URL: https://www.dellemc.com/en-us/collaterals/unauth/technical-guides-support-information/products/storage-4/docu47591.pdf (дата обращения: 19.04.2020).

[3] Inc Calsoft. Storage emulators. — URL: https://calsoftinc.com/accelerator-ips/emulators/ (дата обращения: 18.12.2019).

[4] PureSoftware. FlashArray. — URL: https://www.purestorage.com/ru/products/flasharray-c.html (дата обращения: 26.11.2019).

[5] QNAP. NAS. — URL: https://www.qnap.com/en/support/con_show.php?cid=11 (дата обращения: 18.02.2020).

[6] Software Open Source / Free. Valgrind. — URL: https://valgrind.org/ (дата обращения: 17.04.2020).

[7] T10. SAS documentation. — URL: https://www.t10.org/ (дата обращения: 10.02.2020).

[8] T11. T11 - Fibre Channel Interfaces. — URL: https://standards.incits.org/apps/group_public/workgroup.php?wg_abbrev=t11/ (дата обращения: 05.02.2020).

[9] Ubuntu. ISCSI documentation. — URL: https://ubuntu.com/server/docs/service-iscsi (дата обращения: 15.02.2020).