Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Санкт-Петербургский государственный университет»

Кафедра системного программирования

Булычев Антон Дмитриевич

Разработка децентрализованной файлообменной сети с оповещением об изменениях

Дипломная работа

Допущена к защите.

Зав. кафедрой:

д.ф.-м.н., проф. Терехов А.Н.

Научный руководитель:

инженер ЗАО «Ланит-Терком» Козлов А.П.

Рецензент:

д.ф.-м.н., проф. Терехов А.Н.

Санкт-Петербург
2015
SAINT-PETERSBURG STATE UNIVERSITY

Chair of Software Engineering

Anton Bulychev

Development of decentralised file sharing network with notification of changes

Graduation Thesis

Admitted for defence.

Head of the chair:

professor A.N. Terekhov

Scientific supervisor:

engineer, Lanit-Tercom Inc. A.P. Kozlov

Reviewer:

professor A.N. Terekhov

Saint-Petersburg
2015

Содержание

Введение

В прошлом вычислительные системы состояли из единственной программы, запущенной на единственном компьютере. Сейчас многие из них состоят из множества независимых программ, запущенных на постоянно меняющемся множестве компьютеров.

Одной из задач, возникающих при построении распределенных систем, является задача “publish — subscribe”. Она заключается в проектировании системы, состоящей из двух типов узлов: издателей и подписчиков. Издатели производят некоторую информацию, а подписчики получают эту информацию и некоторым образом используют ее. Обычно в качестве информации выступает файл.

Для решения этой задачи иногда реализуют ad hoc решения, но во многих случаях используют сетевые файловые системы или файлообменные сети.

- Сетевые файловые системы решают более сложную задачу — хранение файлов на специально выделенных серверах — одном или нескольких. При таком подходе издатель записывает файл на сетевую файловую систему, после чего подписчик может его оттуда прочитать.
- Файлообменные сети, напротив, не используют внешних серверов для хранения файлов. Передача файлов осуществляется непосредственно между узлами сети.

Файлообменные сети также делятся на два типа: централизованные и децентрализованные. В централизованной файлообменной сети один или несколько специально выделенных сервера хранят информацию о том, где можно найти тот или иной файл. В децентрализованной — эта информация распределена между всеми участниками сети.

Очень часто в распределенных промышленных системах ставятся следующие требования [5]:

- *Доступность.* Каждый издатель в любой момент времени может опубликовать файл, а подписчик — получить список опубликованных файлов и успешно загрузить файл из этого списка, если имеется доступный узел, содержащий его.
- *Согласованность в конечном счете*. При отсутствии изменений, в конечном счете, подписчик будет иметь актуальный список опубликованных файлов.
- *Устойчивость к разделению*. Расщепление системы на несколько изолированных частей не приводит к некорректной работе каждой из частей.

Выполнения свойств доступности в системе с централизацией (сетевой файловой системе или централизованной файлообменной сети) можно добиться лишь в том случае, если специально выделенных серверов несколько. Очень сложно говорить об устойчивости к разделению — если произойдет расщепление системы, при котором издатель окажется в части без выделенных серверов, то этот узел не сможет опубликовать файл.

Децентрализованные сети решают недостатки централизованных подходов, но из-за того, что информация распределена между всеми узлами сети, подписчику сложно узнать список всех опубликованных файлов в системе. Данную проблему децентрализованных файлообменных сетей можно решить путем оповещения всех узлов при публикации файлов.

1. Постановка задачи

Целью данной работы являлась разработка средства для передачи данных в промышленных распределенных сетях с большим количеством узлов, позволяющего эффективно решать задачу типа “publish - subscribe”.

Для достижения этой цели поставлены следующие задачи:

- Разработать децентрализованную файлообменную сеть с оповещением об изменениях состояний файлов
- Реализовать интерфейс к файлообменной сети
- Произвести оценку полученной файлообменной сети

2. Обзор

В этой части представлен обзор существующих решений задачи передачи файлов между компьютерами: сетевых файловых систем и файлообменных сетей. Кроме того, в конце части представлено описание протоколов оповещения.

2.1. Сетевые файловые системы

Одним из первых существующих подходов передачи файлов был подход на основе сетевых файловых систем. Сетевая файловая система — это файловая система, все данные которой хранятся на удаленном компьютере или множестве компьютеров.

При использовании данного подхода издатель записывает файл на сетевую файловую систему. Затем подписчики при обращении к сетевой файловой системе видят опубликованный файл и, при желании, могут его загрузить.

Сетевые файловые системы решают более сложную задачу — хранение файлов. Это проявляется в следующем: если издатель по какой-либо причине станет недоступным, например выйдет из строя, то файл останется доступным для подписчиков. Такое свойство системы иногда является важным.

Сетевые файловые системы, состоящие из одного компьютера, являются ненадежными: в случае выхода компьютера из строя система перестает функционировать, то есть невозможно ни получить список существующих файлов, ни прочитать их, ни записать новые.

Сетевые файловые системы, состоящие из нескольких компьютеров, обладают большей надежностью и доступностью, но требуют больше ресурсов. Достигается это за счет избыточности: информация дублируется на нескольких

узлах, поэтому при выходе из строя нескольких из них, информация останется доступной. Стоит отметить, что подобные системы сложны в настройке и эксплуатации.

Примерами сетевых файловых систем являются: FTP[6], NFS[14], Hadoop[10].

2.2. Файлообменные сети

Вторым походом решения задачи передачи файлов является использование файлообменных сетей. В отличии от сетевых файловых систем, в файлообменных сетях нет специально выделенных узлов, хранящих файлы. Хранением файлов занимаются непосредственно участники обмена файлов. Процесс выглядит следующим образом: издатель публикует файл, после чего подписчики могут загрузить этот файл у него. Затем, когда подписчик загрузил файл, он сам становится источником загрузки для других подписчиков.

Так как сами узлы сети хранят и дают возможность загрузить с них файлы, такая система может содержать большой объем данных. И благодаря этой возможности узлы могут загружать различные части файлов с нескольких источников одновременно, что увеличивает скорость передачи данных.

2.2.1. Централизованные файлообменные сети

Организация данной разновидности файлообменных сетей выглядит следующим образом. Выделяют специальный сервис, который хранит информацию о местоположении каждого файла. Когда один из узлов хочет сообщить о наличии у него файла, он оставляет запись на сервисе с метаинформацией файла (название, размер) и своим сетевым адресом. То есть в отличии от сетевых файловых систем происходит размещение не самого файла, а данных о нем и его расположении. После размещения информации, она попадает в список размещенных файлов, который может загрузить каждый узел

при обращении на сервис. Когда подписчику необходимо загрузить файл из полученного списка, он обращается на сервис, чтобы узнать адреса узлов, содержащих этот файл, после чего происходит загрузка файла напрямую с этих узлов.

В простейшем случае сервис может состоять из одного компьютера. К достоинствам такого подхода можно отнести сравнительную простоту программирования и небольшое время его настройки. Недостатком же является наличие единой точки отказа.

Этот недостаток устраняется при использовании не одного, а нескольких компьютеров, объединенных в единый сервис. Однако такой подход требует больше ресурсов, времени разработки и настройки.

К общим достоинствам централизованного подхода можно отнести небольшой объём служебной информации, передаваемой и хранящейся на сервисе. К общим недостаткам — отсутствие устойчивости к разделению.

Ярким примером такой сети является Napster [13] — файлообменная сеть, действовавшая с 1999 года по 2001 год. Сервис позволял легко обмениваться музыкальными файлами с другими людьми, что привело к обвинениям в нарушении авторских прав со стороны музыкальной отрасли. Сервис был остановлен по решению суда. Примерами также еще являются DC [4] и BitTorrent [2] (если не используется расширение с DHT [3]).

2.2.2. Децентрализованные файлообменные сети

Децентрализованные файлообменные сети отличаются от централизованных отсутствием специального сервиса для хранения сетевых адресов узлов, содержащих тот или иной файл. Информация об адресах распределена между всеми узлами сети таким образом, что по ней можно

осуществить быстрый поиск. Подобная функциональность реализуется с помощью распределенной хеш-таблицы.

Распределённая хеш-таблица — это хеш-таблица, содержимое которой распределено между узлами системы. Таким образом, каждый узел хранит лишь ее часть.

Распределенные хеш-таблицы в децентрализованных файлообменных сетях применяются следующим образом. Пусть у каждого файла есть его идентификатор, который обычно вычисляется как хеш-функция от его названия или содержимого. Если узел с адресом A хочет опубликовать файл с идентификатором k, он записывает в распределенную хеш-таблицу пару $k \rightarrow A$. Когда узел хочет узнать, на каких узлах расположен файл с определенным идентификатором, он извлекает из распределенной хеш-таблицы все пары с этим ключом, равным идентификатору файла. Адресами узлов, на которых содержится этот файл, будут значения этих пар.

Для функционирования децентрализованных файлообменные сетей не требуется специально выделенных серверов. Стоит отметить, что объём передаваемой служебной информации в таких файлообменных сетях больше, чем в централизованных, но зато их надёжность гораздо выше, так как в них отсутствуют единые точки отказа и они устойчивы к разделению сети. Одним из примеров децентрализованной файлообменной сети является Kad [12].

Так как данные распределены между всеми узлами сети, каждому узлу сложно получить список всех опубликованных файлов в сети, что является существенным недостатком данного подхода.

2.3. Протоколы оповещения

Одним из недостатков децентрализованных файлообменных сетей была названа невозможность получения списка опубликованных файлов. Этот изъян можно исправить, используя различные методы оповещения. Ниже будут представлены наивный подход и подход, основанный на gossip-протоколе [9].

2.3.1. Наивный подход

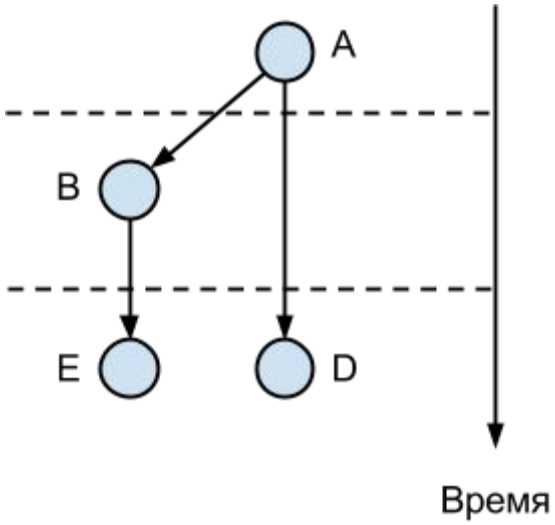
Наивный подход заключается в том, что каждый подписчик периодически опрашивает каждого издателя об изменениях, которые тот совершил с момента предыдущего опроса. Этот подход очень прост в реализации. Также он надежен и имеет имеет предсказуемое время распространения изменений и фиксированную нагрузку на сеть. Пусть P — количество издателей в системе, S — подписчиков и r — количество запросов в секунду, которые делает каждый подписчик. Тогда время распространения изменения будет равно $\frac{P}{r}$ секунд, а нагрузка на сеть будет равна Sr запросов в секунду. Такой подход не поддается масштабированию: обе характеристики зависят линейно от количества узлов.

2.3.2. Gossip-протокол

В подходе, основанном на gossip-протоколе, все узлы участвуют в распространении изменений. Данный подход является вероятностным, а его идея заключается в том, что каждый узел с некоторой периодичностью выбирает случайные узлы и обменивается с ними информацией.

Например, пусть есть множество узлов, обозначенных латинскими буквами, и узел A хочет распространить некоторую информацию. Для этого он выбирает случайно один из доступных ему узлов — пусть это будет узел B — и сообщает ему эту информацию. Через некоторое время уже два узла, A и B, случайно выбирают узлы, которым передадут информацию, — пусть это будут

D и *E*, соответственно. Таким образом, в силу случайного выбора узлов, информация распространится до всех доступных узлов.



Такой подход передачи информации надежен: скажем, если узел *E* по каким-либо причинам не получил сообщения от *B*, то он с большой вероятностью скоро получил бы информацию от какого-нибудь другого узла.

Количество узлов, ответственных за осуществление обмена информацией, можно менять. Увеличивая их число, увеличивается скорость распространения. Обычно количество таких узлов равняется 2.

В каждый момент времени узлы могут иметь различную и несогласованную между собой информацию, но, при отсутствии изменений, через некоторое время она станет согласованной на всех узлах. Математическое ожидание времени распространения обновления пропорционально логарифму от количества узлов системы — даже при выходе узлов из строя и потери сообщений в сети.

Существуют различные типы gossip-протоколов. В приведенном примере использовался push gossip-протокол — каждый узел “проталкивает” информацию в другие узлы. Другим видом gossip-протокола является pull — каждый узел выбирает случайные узлы и спрашивает у них новую информацию.

Очевидно, что когда узлов, владеющих информацией мало, то при push gossip-протоколе узлы, владеющие информацией, с большей вероятностью будут выбирать узлы, которые не владеют ей, и информация будет быстро распространяться. Подобным свойством обладает и pull gossip-протокол, когда количество владеющих информацией велико.

Существует push-pull gossip-протокол. Он сочетает преимущества двух предыдущих типов. Каждый узел выбирает случайные узлы и “синхронизируется” с ними — они в двустороннем режиме обмениваются между собой актуальной информацией.

Gossip-протокол часто используют в распределенных системах для того, чтобы каждый узел:

- знал какие узлы есть в системе и какие появляются новые
- знал о том, когда узел перестает быть доступным и когда становится снова доступным

Основываясь на этих сведениях, узлы распределенной системы могут принимать те или иные решения в зависимости от решаемой задачи.

3. Описание решения

В разделе “Обзор” были рассмотрены существующие решения задачи обмена файлами между различными компьютерами. Ранее описанные проблемы сетевых файловых систем и централизованных файлообменных сетей можно решить с помощью дополнительных ресурсов: например, выделения новых компьютеров для хранения информации. Подобный подход сопряжен также с трудностью настройки и поддержки такой системы.

Децентрализованные файлообменные сети не обладают подобными проблемами, но они не имеют часто требуемой функциональности — получения списка опубликованных файлов. Для решения этой проблемы в дипломной работе было разработано решение, использующее gossip-протокол для оповещения об изменениях. Добавление оповещения в децентрализованные сети позволяет устранить их существенный недостаток, сохраняя при этом преимущества.

3.1. Файловое дерево и виды оповещений

В дипломной работе было решено организовать файлы в виде файлового дерева. Такая иерархичная структура удобна для создания, чтения, удаления и изменения информации в ней. Это файловое дерево допускает все операции, доступные в файловой системе. Основные из них:

- создание файла
- удаление файла
- редактирование файла
- создание директории
- удаление директории

Решение, описываемое в работе, является модификацией децентрализованных файлообменных сетей. Модификация заключается в том, что к системе добавляется компонент, служащий для оповещения участников сети в случае совершения изменений в файловой системе одним из узлов. Так например, в случае создания файла, файл публикуется в файлообменной сети и соответствующие изменения в файловой системе распространяются среди всех узлов.

Удаление файла происходит сложнее: сначала файл удаляется из файлового дерева, и соответствующие изменения распространяются через компонент. Из файлообменной сети файл удаляется с некоторой задержкой, давая тем самым завершить загрузку тем узлам, которую ее начали.

Изменение файла эквивалентно атомарному удалению и созданию нового файла. Создание и удаление директорий никак не затрагивают файлообменную сеть — только изменения файлового дерева распространяются через новый компонент.

Реализация компонента оповещения об изменениях файлового дерева будет рассмотрена позже.

3.2. Таблица маршрутизации

Прежде чем рассматривать реализацию компонента оповещения будет рассмотрено понятие таблицы маршрутизации. Как было сказано в разделе “Обзор”, децентрализованные файлообменные сети используют в своих реализациях распределенные хеш-таблицы. В качестве ключей хеш-таблицы используются идентификаторы файлов, а в качестве значения — адрес узла, содержащего файл. Такая структура помогает быстро находить файлы в сети.

В дипломной работе было расширено использование распределенной хеш-таблицы. В качестве значения по-прежнему выступает адрес узла, а в

качестве ключа идентификатор сущности, расположенной на этом узле. В качестве сущности может выступать файл, но позже будут рассмотрены и иные сущности.

Данное расширение необходимо для быстрого поиска не только файлов, но и других сущностей. Поэтому мы будем называть распределенную хеш-таблицу таблицей маршрутизации. Также мы будем писать “узел A добавляет запись E в таблицу маршрутизации”, имея в виду, что узел A записывает в распределенную хеш-таблицу запись “E → адрес узла A”. Запись “найти E в таблице маршрутизации” будет значить извлечь значения всех записей из распределенной хеш-таблицы с ключем E.

3.3. Gossip-протокол для оповещения об изменениях

В разделе “Обзор” были описаны подходы для оповещения об изменениях. В дипломной работе для реализации компонента оповещения был выбран подход, использующий push-pull gossip-протокол, так как он хорошо масштабируется, что важно в решаемой задаче.

Используется gossip-протокол следующим образом. Каждый узел a хранит счетчик V_a — версию локальных данных. Производя операцию с файловым деревом, узел увеличивает свой счетчик и, используя gossip-протокол, сообщает новое значение счетчика остальным узлам в сети.

Может показаться, что для получения самого изменения достаточно обратиться к самому узлу. Но такой подход не надежен, так как этот узел может стать недоступен прежде, чем было получено изменение. Поэтому рассмотрим решение с использованием таблицы маршрутизации. Пусть узел a производит операцию и получает версию v. Он запоминает операцию, которую совершил, и делает запись в таблице маршрутизации $\{a : v\}$. После этого другие узлы, получив уведомление через gossip-протокол, ищут $\{a : v\}$ в таблице

маршрутизации и загружают с полученных узлов информацию о совершенной операции. После чего они сами добавляют запись $\{a : v\}$ в таблицу маршрутизации. Таким образом обеспечивается следующее свойство: если доступен хотя бы один узел с изменением, все узлы его получат.

3.4. Seed-модификация в gossip-протоколе

Как было отмечено раньше, время распространения изменений в gossip-протоколе пропорционально логарифму от количества узлов сети. Это время можно уменьшить в большинстве случаев за счет введения элемента “централизации” в gossip-протокол.

Рассмотрим предназначение seed-списка. Когда в систему нужно подключить новый узел, он должен вступить в контакт с узлом, уже находящимся в системе. Обычно в качестве таких узлов выбирают (для надежности) несколько узлов, которые с меньшей вероятностью могут перестать быть доступными. Администраторы на этапе конфигурации системы прописывают всем узлам одинаковый список адресов таких узлов. Этот список и называется seed-списком. Таким образом, новый узел не сможет вступить в систему, если все узлы из seed-списка будут недоступны. Подобное решение обладает высокой надежностью.

В дипломной работе предлагается использовать этот список не только для подключения новых узлов в сеть, но и для ускорения распространения информации. Для этого каждый узел для распространения выбирает не два случайных узла в системе, а выбирает один случайный узел из seed-списка и один случайный среди всех.

Выбор узла из seed-списка обусловлен тем, что при реализации данного подхода изменения распространяются преимущественно следующим образом: узел передает информацию seed-узлу, затем seed-узлы распространяют

информацию между собой, после чего остальные узлы, обратившись к seed-узлам, получают эту информацию. А за счет выбора случайного узла среди всех такой подход сохраняет элементы децентрализованности.

Стоит отметить следующее: если количество узлов в системе N, то без seed-модификации нагрузка на каждый узел была в среднем равна 2. В случае seed-модификации, где количество seed-узлов равно S, нагрузка на seed-узел будет равна в среднем $1 + \frac{N-1}{S}$. При этом общая нагрузка на сеть не возрастает.

3.5. Интерфейс к файлообменной сети

В дипломной работе в качестве интерфейса к сети был выбран интерфейс файловой системы. Это удобный интерфейс для операций с файловым деревом, в которое организованы все опубликованные файлы в сети. Такой подход позволяет избежать создания специальных клиентских библиотек для приложений на различных языках программирования. Также сторонние приложения или приложения, находящиеся на поддержке, могут без каких-либо изменений использовать файлообменную сеть.

Для большинства ОС известны следующие способы реализации файловой системы:

- реализация модуля ядра ОС
- реализация интерфейсов FUSE [7]

Модуль ядра ОС — это объект, содержащий код, расширяющий функциональность ОС. Для загрузки модуля в ядро требуются соответствующие привилегии. Подобный подход не является кроссплатформенным, поэтому для каждой операционной системы придется реализовывать свой модуль.

FUSE (Filesystem in Userspace) — модуль для ядер UNIX-подобных операционных систем, позволяющий пользователям без привилегий создавать

их собственные файловые системы. Код файловой системы запускается в режиме пользователя, в то время как FUSE является мостом для интерфейсов ядра. В данном подходе требуется реализовать интерфейсы FUSE, которые не будут зависеть от платформы.

FUSE уступает в производительности модулю ядра, так как является лишь интерфейсом взаимодействия с виртуальной файловой системой, в то время как модуль ядра может взаимодействовать с ней напрямую. Но несмотря на это, он позволяет более простым способом реализовать файловую систему, так как обладает следующими свойствами:

- является кроссплатформенным и не требует изменения кода ядра ОС
- исполняется в пользовательском режиме и является безопасным для ОС
- позволяет легко отлаживать программы

Поэтому в работе было отдано предпочтение FUSE.

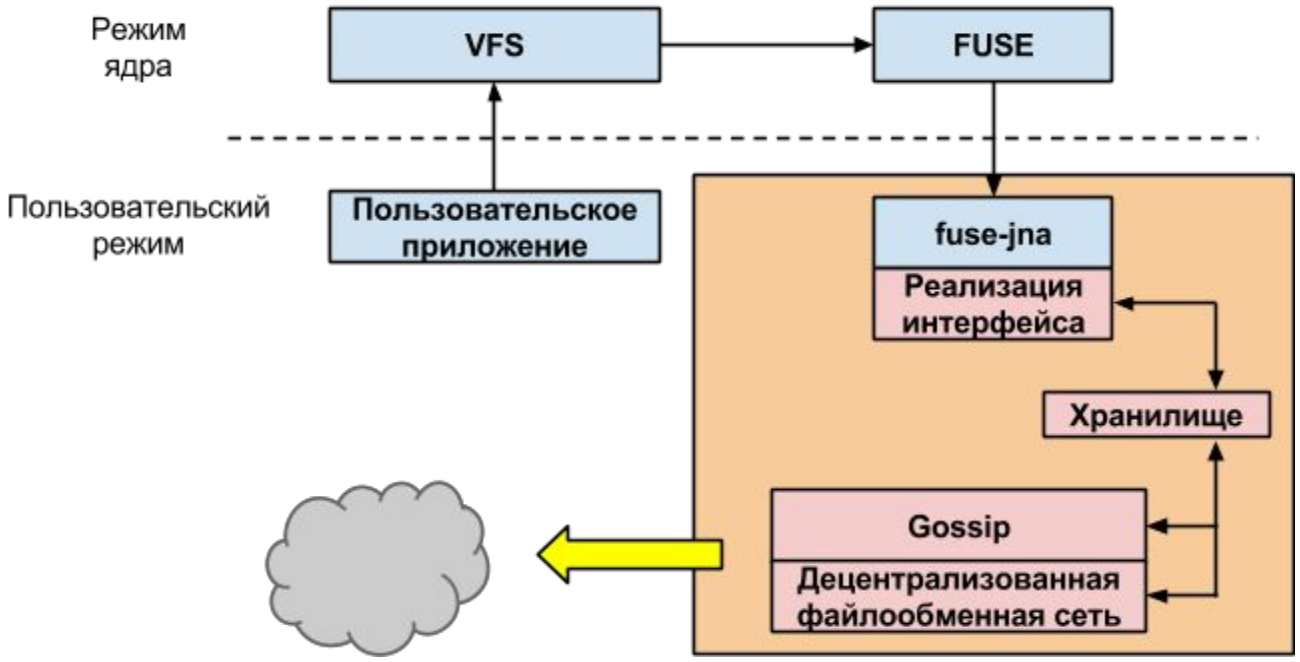
3.6. Реализация

Решение, описанное в дипломной работе, было реализовано на языке программирования Scala [15]. Это функциональный язык программирования, компилируемый в Java [11] байт-код и исполняемый на JVM. Он обладает выразительными средствами, позволяющими существенно уменьшить объем кода и ускорить процесс разработки приложений.

Реализация была осуществлена с использованием фреймворка Akka[1]. Этот фреймворк реализует Actor модель, позволяющую просто и безопасно писать многопоточные приложения. Также эта модель упрощает разбиение приложения на отдельные модули, что также ускоряет разработку и позволяет тестировать каждый модуль в отдельности.

Механизм FUSE разрабатывался для использования его в приложениях, написанных на C/C++. Поэтому в JVM он доступен только через native-интерфейсы. В дипломной работе была использована библиотека fuse-jna[8]. Это Java библиотека, предоставляющая абстрактный класс, в котором реализованы интерфейсы взаимодействия с FUSE.

Архитектура системы выглядит следующим образом (розовым цветом выделены те части, которые реализованы в данной дипломной работе):



Когда пользовательское приложение производит операцию с файловой системой, соответствующая операции попадают в модуль ядра FUSE через виртуальную файловую систему. Затем модуль ядра FUSE осуществляет обратный вызов приложения для выполнения операции.

В системе был реализован интерфейс, необходимый для работы fuse-jna, и модуль, реализующий gossip-протокол. Также была реализована классическая децентрализованная файлообменная сеть: распределенная хеш-таблица и механизм для загрузки файлов с узлов.

Центральной частью системы является хранилище. Оно содержит в себе файлы, которые были записаны или загружены узлом, и файловое дерево с его историей изменений. Все данные обязательно сохраняются на жестком диске. Также часть из них находится в оперативной памяти: например, файловое дерево или блоки файлов, к которым недавно обращались.

4. Оценка решения

В этом разделе будет произведено сравнение следующих методов оповещения об изменениях:

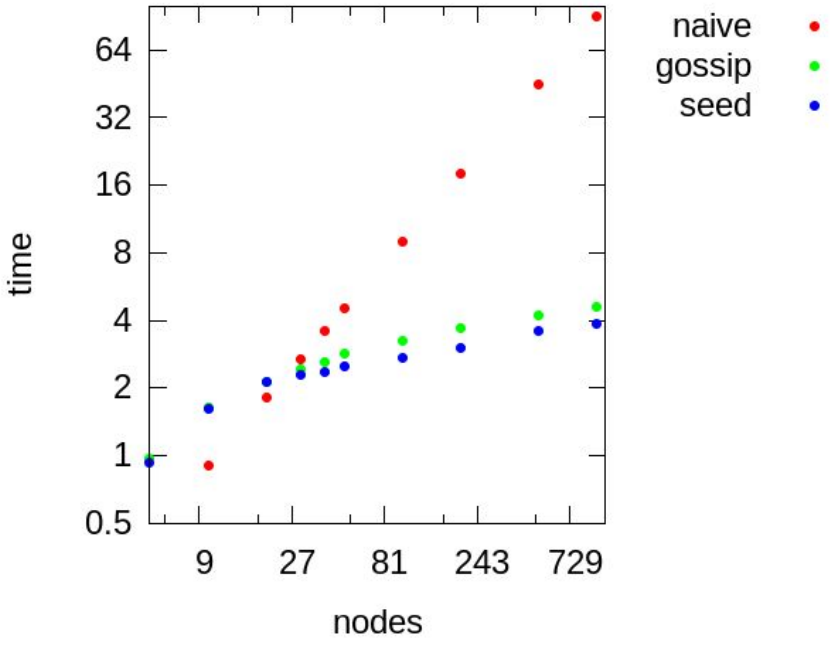
- наивного подхода, основанного на периодическом опросе подписчиками всех издателей
- gossip-протокол без модификаций
- gossip-протокол с seed-модификацией

Моделировалась система со следующими характеристиками:

- N — количество узлов
- 20% издателей
- 80% подписчиков

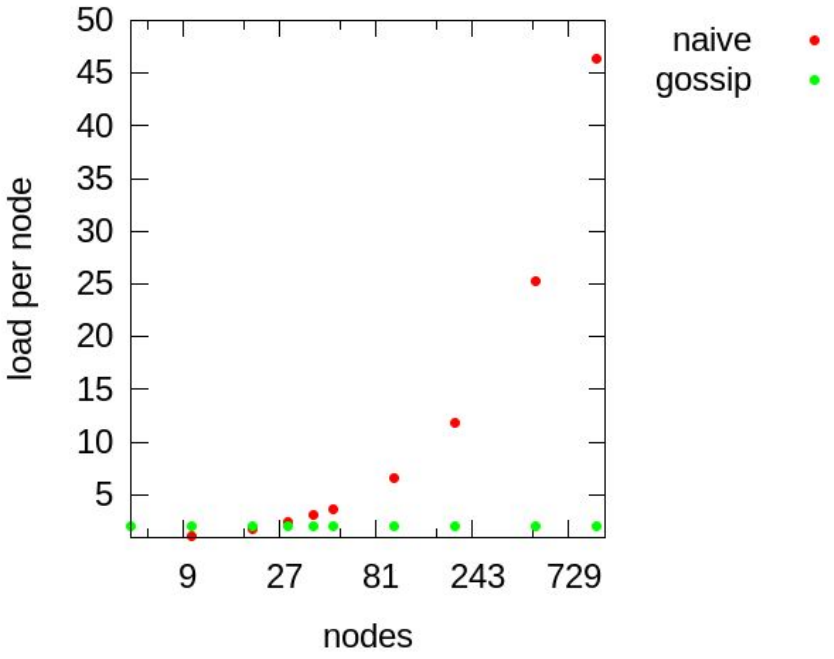
Количество seed-узлов в seed-модификации было выбрано равным $max\ \left\{\frac{N}{40},\ 3\right\}$. Такие параметры позволяют обеспечить нагрузку на seed-узел в среднем не больше 40 запросов в секунду и понизить вероятность того, что все seed-узлы выйдут из строя.

Сначала для наивного подхода была зафиксирована нагрузка на сеть, равная нагрузке на сеть, осуществляемой gossip-протоколами, а именно $2N$. В результате было получено следующее время распространения изменений в системе в зависимости от количества узлов.



При небольшом количестве узлов наивный алгоритм ведет себя лучше алгоритмов, использующих gossip-протокол, но с их ростом начинает деградировать. Seed-модификации сокращает время распространения изменений по сравнению с gossip-протоколом без модификации. Так например, разница во времени распространении при 1000 узлах состовляет 0.7 секунды.

Затем для наивного подхода было зафиксировано время распространения изменений, равное времени распространения в gossip-протоколах. Были получены следующие результаты нагрузки на сеть в зависимости от количества узлов в сети. Нагрузка считалась как общая нагрузка на сеть, деленная на количество узлов.



Результат получается аналогичным — с ростом количества узлов наивный алгоритм увеличивает нагрузку на сеть. Так например нагрузка при 1000 узлах равна 46 запросам, в то время как у gossip-протокола нагрузка равна 2.

Данные результаты показывают невозможность масштабирования наивного подхода и превосходство использования gossip-протокола в случае большого количества узлов.

Заключение

В рамках данной дипломной работы была разработана децентрализованная файлообменная сеть с оповещением об изменениях, позволяющая решать эффективно задачу типа “publish - subscribe”. Был разработан интерфейс к сети в виде файловой системы, позволивший работать с файлообменной сетью через стандартный интерфейс. Была произведена оценка, показавшая, что данное решение эффективно горизонтально масштабируется.

Список литературы

1. Akka. — URL: http://akka.io/
2. BitTorrent. — URL: http://www.bittorrent.org/beps/bep_0000.html
3. BitTorrent Enhancement Proposal: DHT. — URL: http://www.bittorrent.org/beps/bep_0005.html
4. Direct Connect. — URL: http://en.wikipedia.org/wiki/Direct_Connect_(protocol)
5. Eventual consistency. — URL: http://en.wikipedia.org/wiki/Eventual_consistency
6. File Transfer Protocol RFC. — URL: http://www.ietf.org/rfc/rfc959.txt
7. FUSE. — URL: http://fuse.sourceforge.net/
8. fuse-jna. — URL: https://github.com/EtiennePerot/fuse-jna
9. Gossip. — URL: http://en.wikipedia.org/wiki/Gossip_protocol
10. Hadoop Distributed File System . — URL: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
11. Java. — URL: http://www.oracle.com/technetwork/java/index.html
12. Kad. — URL: http://en.wikipedia.org/wiki/Kad_network
13. Napster. — URL: http://en.wikipedia.org/wiki/Napster
14. Network File System RFC. — URL: http://www.ietf.org/rfc/rfc3530.txt
15. Scala. — URL: http://www.scala-lang.org/