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

Кафедра системного программирования

Морозов Сергей Валерьевич

Многоярусная файловая система: оптимизация архитектуры и алгоритмов управления

Курсовая работа

Научный руководитель:
д. ф.-м. н., профессор Нестеров В. М.

Санкт-Петербург
2015

Оглавление

Введение

Практически все современные компании на сегодняшний момент ежедневно генерируют большое количество данных. Очень быстро возникает проблема с хранением этих данных, а также с их последующим извлечением и использованием. Проблема хранения может быть решена разными способами. Выбор того или иного подхода к хранению данных зависит от многих причин, в том числе и от финансовых возможностей конкретной компании. Многие останавливают свой выбор на облачных хранилищах данных: малые и средние на публичных или гибридных, крупные на частных. Выбор между публичными/гибридными и частными облачными системами часто обусловлен соображениями экономии, крупным компаниям дешевле содержать собственную облачную систему хранения данных, развернутую на своих серверах.

Облачные системы хранения решают много проблем, которые ранее приходилось решать администраторам серверной в компании. Одним из преимуществ облачных систем хранения является гарантия очень высокой вероятности сохранения данных клиентов даже в случае аппаратных сбоев. К примеру, в облачных объектных системах хранения существуют такие механизмы как избыточное кодирование, сохранение нескольких копий записываемых данных на разные сервера и т. д.

Сегодня пользователи ожидают от облачного хранилища практически бесконечного запаса свободного места. Количество объектов в облачных объектных системах хранения уже достигает триллионов. Многие пользователи, в том числе приложения, ожидают от системы хранения POSIX интерфейса, чтобы иметь возможность работать с привычными абстракциями – файлами и папками. Появляется необходимость организации объектов в иерархические структуры на пространствах имен в облачном объектном хранилище. Объем метаданных пространств имен с иерархической структурой пропорционален объему хранимых в облачном объектном хранилище данных. Существует много кластерных файловых систем, скрывающих распределенную топологию размещения ресурсов хранения, предоставляя пользователю

интерфейс идентичный интерфейсу отдельно взятого сервера. Но количество свободного места для хранения данных и метаданных ограничено совокупным объемом ресурсов хранения кластера.

Многолетний опыт человечества в использовании систем хранения данных позволяет утверждать, что шаблон обращения к данным подчиняется распределению Парето [6]. В основном происходят обращения к 20% хранимых данных («горячие» данные), тогда как к остальным 80% обращения происходят редко («холодные» данные). Доступ к горячим данным должен происходить очень быстро, тогда как время получения холодных данных может быть достаточно большим. Устройства, способные обеспечить быструю скорость чтения и записи данных, такие как твердотельные накопители, стоят дорого, медленные магнитные диски – дешево. Индустрия IT давно разработала решения, позволяющие сократить стоимость систем хранения, не теряя производительности – многоярусные системы хранения. Горячие данные находятся на быстрых накопителях информации, а холодные на медленных. Расположение данных на накопителях разного класса (ярусах) определяется специальными алгоритмами.

Автором разрабатывается кластерная файловая система, способная поддерживать работу с несколькими ярусами, одним из которых является так называемый *расширенный* ярус – облачная объектная система хранения. Разрабатываемое решение будет полезно как небольшим и средним предприятиям, так и крупным. Пользователям будет предоставляться практически неограниченный объем ресурсов хранения и будет обеспечена семантика взаимодействия с файловой системой, близкая к POSIX-семантике.

В рамках курсовой работы необходимо выполнить следующие задачи:

- сделать обзор существующих аналогов;
- определить архитектуру многоярусной файловой системы;
- убедиться, что все выбранные за основу компоненты является решениями с открытым исходным кодом.

1. Определения

Файловая система – иерархическая структура на файлах. Обеспечивает простой доступ к файлам, размещенным на жестком диске, разделе диска или логическом разделе. Файловая система также осуществляет контроль доступа к файлам пользователей [7].

NAS (Network-Attached Storage) – высокопроизводительное устройство, предназначенное для обмена и хранения файлов. NAS позволяет обмениваться файлам по IP сети. NAS объединяет имеющиеся ресурсы хранения в единую систему, упрощая процесс управления ресурсами хранения. В NAS используются сетевые протоколы и протоколы файлового обмена, предоставляющие доступ к данным [7].

Распределенная (кластерная) файловая система – файловая система, предоставляющая доступ к постоянному хранилищу, в котором множество объектов существует с момента их явного создания до момента их явного удаления, и устойчивая к отказам, как к программным, так и к аппаратным. Постоянное хранилище охватывает несколько федераций ресурсов хранения, в которых клиенты могут создавать, удалять, читать и писать файлы. В отличие от локальных файловых систем, ресурсы хранения и клиенты файловой системы распределены по сети. Распределенная файловая система должна удовлетворять свойствам *прозрачности* (пользователи имеют доступ к системе вне зависимости от места, где они выполнили вход в систему, способны выполнять операции в распределенной файловой системе как в локальной, а также не должны заботиться о возможных отказах оборудования), *отказоустойчивости* (система не должна прекращать работу в случае отказа части оборудования) и *расширяемости* (способности эффективно использовать много серверов, которые могут динамически добавляться в систему) [4].

Параллельная файловая система – тип распределенных файловых систем. Предоставляется доступ к разделяемому пространству имен. Параллельная файловая система изначально ориентирована на параллелизм, конкурентный (часто скоординированный) доступ многих кли-

ентов. Такие системы работают через высокоскоростные сети и имеют сильно оптимизированные потоки ввода-вывода, чтобы обеспечивать максимальную пропускную способность. В параллельных файловых системах происходит распределение «кусочков» файла по нескольким узлам. Предполагается, что такие файловые системы устанавливаются на корпоративные системы хранения данных [11].

Ярусность – установление иерархии на системах хранения, основывающуюся на требованиях к предоставляемому сервису, таких как производительность, непрерывность бизнеса, безопасность, стоимость и др. [12].

2. Обзор

Безусловно, «изобретать велосипед», строя свою распределенную многоярусную файловую систему с нуля, неразумно. В этом разделе будут приведены примеры распределенных файловых систем, которые рассматривались (рассматриваются), как основа для добавления многоярусности.

MooseFS[1]. Отказоустойчивая сетевая распределенная файловая система, предоставляющая пользователям общее пространство имен. Данные распределяются по нескольким физическим серверам, которые для пользователя представляются одним ресурсом. При работе со стандартными операциями с файлами, MooseFS ведет себя также, как и UNIX-подобные файловые системы. Некоторые свойства MooseFS представлены ниже:

- иерархическая структура директорий;
- хранение атрибутов POSIX-файлов;
- поддержка символьных и блоковых устройств, трубопроводов и сокетов;
- поддержка символьных и жестких ссылок;
- отказоустойчивость;
- масштабируемость (динамическое расширение объемов хранения за счет присоединения новых компьтеров или дисков).

Файловая система имеет открытый исходный код и распространяется под лицензией GPL.

CephFS[2]. Распределенная фаловая система с открытым исходным кодом, распространяемая под лицензией LGPL. CephFS динамически

[1] **http://www.moosefs.org**
[2] **http://ceph.com**

управляет распределенными метаданными с помощью *кластера метаданных (MDS)* и хранит данные и метаданные в *объектном хранилище (OSD)*.

CephFS имеет несколько интерфейсов взаимодействия с пользователями. Благодаря своей архитектуре, файловая система предоставляет семантику близкую к POSIX [2]. Поддерживается специальный механизм блокировок. Уровень согласованности данных может быть понижен выставлением специльного флага C_LAZY, который разрешает чтение файла в момент его перезаписи [4].

Файловая система CephFS обладает высокой отказоустойчивостью, а также поддерживает многоярусность. Выделяют два яруса – cache tier (для горячих данных) и storage tier (для холодных данных) [1].

GlusterFS[3]. Распределенная файловая система с открытым исходным кодом, выпускаемая под лицензией GPL. Имеет клиент-серверную архитектуру, в которой нет отдельного сервера для хранения метаданных. Данные и метаданные хранятся на нескольких устройствах, присоединенных к разным серверам. GlusterFS полностью поддерживает семантику POSIX [8].

Местонахождение файла определяется с помощью алгоритма *EHA (Elastic Hashing Algorithm)* [3], поэтому от сущности сервера метаданных в этой файловой системе отказались.

В случае отказа одного из серверов, на котором установлена GlusterFS, он автоматически удаляется из системы и операции ввода-вывода на этом узле блокируются [4]. Поддерживает несколько ярусов [9].

OrangeFS[4]. Параллельная файловая система, являющаяся ответвлением параллельной файловой системы PVFS[5]. Предоставляет очень быстрый доступ к данных для параллельных приложений. OrangeFS отличается от своего родителя наличием функциональности, которой

[3] **https://www.gluster.org**

[4] **http://www.orangefs.org**

[5] http://www.pvfs.org

нет в PVFS. PVFS предназначена для очень больших кластеров. Следующими характеристиками обладает как OrangeFS, так и PVFS: производительность, надежность, независимость от аппаратной платформы, быстрое развертывание.

OrangeFS также обладает некоторыми достоинствами, которых нет в PVFS.

- *Операции над метаданными.* Используется взаимодействие сервер-сервер, чтобы повысить производительность выполнения операций над метаданными. В OrangeFS реализован механизм распределенных директорий [5]. Также в одном из подпроектов OrangeFS реализуется улучшенный механизм поиска по метаданным.

- *Избыточность данных.* В OrangeFS реализован конфигурируемый механизм избыточности данных, а также механизм восстановления после сбоев. Имеется возможность задавать свой уровень избыточности отдельному файлу.

- *Контроль доступа.* В OrangeFS реализована поддержка контроля доступа пользователей к файловой системе. Эта одно из основных направлений, над которыми велась работа в последних релизах.

OrangeFS поддерживает семантику, близкую к POSIX. OrangeFS является проектом с открытым исходным кодом и распространяется под лицензией LGPL.

MarFS[6]. Распределенная файловая система. Реализует с помощью одной или более POSIX файловых систем расширяемый компонент метаданных. Также реализует с помощью одного или более хранилища данных расширяемый компонент данных. В качестве хранилища данных может использоваться, например, EMC ECS ViPR[7].

MarFS поддерживает работу с несколькими ярусами, одним из которых может являться облачное объектное хранилище данных.

[6] **https://github.com/mar-file-system/marfs**

[7] http://www.emc.com/storage/ecs/index.htm

К сожалению, MarFS обладает существенными для разрабатываемой многоярусной системы хранения ограничениями. Авторами MarFS утверждается, что файловая система поддерживает семантику, близкую к POSIX. Тем не менее в документации указаны ограничения на POSIX-семантику более жесткие, чем в других рассматриваемых в обзоре файловых системах. К примеру, MarFS не производит никаких проверок в случае, когда несколько клиентов пытаются писать в один файл, также MarFS не предоставляет возможность обновления файла, находящегося в облачном хранилище.

MarFS имеет открытый исходный код и распространяется под лицензией BSD.

Сравнение файловых систем. В таблице 1 приведено сравнение рассматриваемых в обзоре файловых систем.

Таблица 1: Сравнение файловых систем. Вопросы в некоторых клетках означают отсутствие явного указания на наличие выбранной характеристики в документации к файловой системе. В большинстве систем семантика POSIX полностью не соблюдается, но в некоторых системах POSIX-семантика ослаблена сильнее, чем в остальных. Такие системы отмечены +-. Системы, послностью поддерживающие POSIX-семантику, отмечены ++.

	POSIX-совмест.	Ярусность	Отказоуст.	Лицензия
MooseFS	+	-	+	GPL
CephFS	+	+	+	LGPL
GlusterFS	++	+	+	GPL
OrangeFS	+	-(planned for 3.0 rel.)	+	LGPL
MarFS	+-	+	?	BSD

3. Архитектура многоярусной файловой системы

Архитектура будущей многоярусной системы будет зависеть от того, какую файловую систему выбрать как основу. На текущий момент автором в большей степени изучена файловая система OrangeFS.

Для начала рассмотрим общие концепции, которые будут характерны для любой реализации многоярусной файловой системы. Самый «умный» компонент – Rule Engine. Через него задаются политики синхронизации данных между ярусами с помощью специального предметно-ориентированного языка (DSL). Некоторые существующие системы содержат в себе компоненты для управления синхронизацией данных, где правила задаются с помощью DSL, но синтаксис этих языков сложен как для восприятия, так и для написания правил. Примерами таких систем являются Robinhood Policy Engine[8] и iRODS Rule Engine[9]. Еще один компонент системы – сервер метаданных. Он может быть реализован как отдельная сущность (например, простое использование etcd[10]), так и интегрирован в файловую систему. Стоит заметить, что не во всех кластерных файловых системах есть сервер метаданных. Очевидно, что у файлов появятся дополнительные атрибуты вроде obj_id, идентифицирующие расположение файла (объекта). На рисунке 1 представлен один из возможных вариантов архитектуры многоярусной файловой системы.

Рассмотрим файловую систему OrangeFS. Она имеет несколько значимых преимуществ: данные и метаданные равномерно распределяются по всем узлами, по сравнению с другими распределенными системами объем метаданных на файл меньше, в следующих версиях OrangeFS официально планируется добавить поддержку нескольких ярусов, а значит в будущем разработка автора может быть интегрирована и в исходный код OrangeFS.

[8]https://github.com/cea-hpc/robinhood/wiki

[9]http://wiki.irods.org/index.php/Rule_Engine, https://docs.irods.org/master/manual/rule_language

[10]https://github.com/coreos/etcd

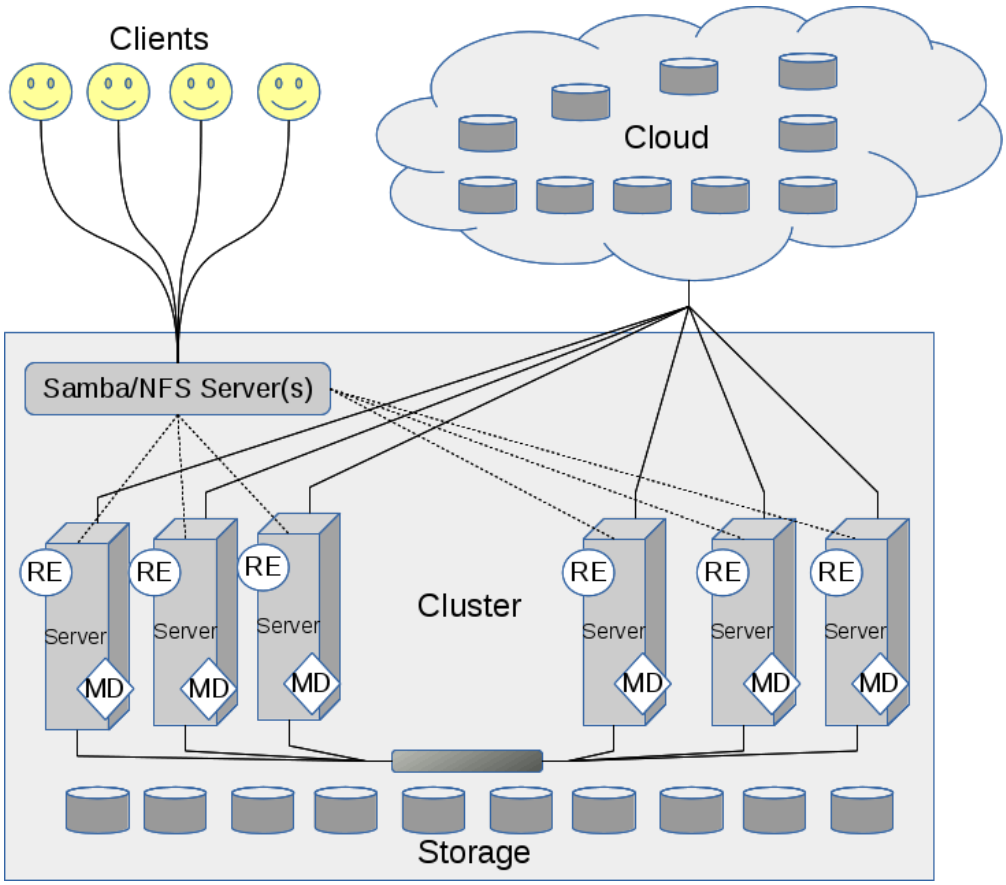

Рис. 1: Архитектура многоярусной файловой системы. Распределенная файловая система установлена на серверах. На серверах же реализован распределенный Rule Engine (RE). Хранилище метаданных (MD) может быть реализовано отдельно, а может являться уже существующим компонентом файловой системы-родителя. Клиенты работают с Samba/NFS директориями, которые привязаны к директориям многоярусной файловой системы. Горячие данных хранятся в локальном хранилище (Storage) на т. н. *оперативном* ярусе, а холодные в облачном объектном хранилище (Cloud) на т. н. *расширенном* ярусе.

OrangeFS поддерживает POSIX семантику на уровне двух интерфейсов: ядерного модуля и direct интерфейса [10]. Ядерный модуль позволяет использовать файловую систему OrangeFS в родном для Linux виде, как монтированную файловую систему, где можно использовать стандартные Linux-утилиты: *ls*, *cp*, *rm* и т. п. Direct интерфейс позволяет работать с файловой системой OrangeFS в обход ядра Linux. Интерфейс обладает похожей на POSIX семантикой.

Правильнее всего было бы внедрить поддержку нескольких ярусов в серверную реализацию OrangeFS (компонент pvfs2-server). Эта зада-

ча весьма сложная и велика вероятность внести ошибки в исходный код, который уже хорошо отлажен. Но, в случае выбора OrangeFS в качестве системы-родителя, рано или поздно это сделать придется. Но также можно рассматривать возможность внесение изменений лишь в клиентскую часть реализации OrangeFS (компонент pvfs2-client), чтобы адаптировать pvfs2-client к работе с Rule Engine, а, соответственно, к работе с несколькими ярусами (если файла нет на оперативном ярусе, обратиться к облачному хранилищу, т.е. к расширенному ярусу).

Еще одной, привлекательной на первый взгляд для исследования распределенной файловой системой, является CephFS, во многом благодаря встроенной поддержке многоярусности. Если файловая система CephFS будет взята в качестве основы для многоярусной файловой системы, то изменений придется вносить меньше. Необходимо будет представить облачное хранилище, как storage tier (в терминологии CephFS). Вероятно потребуется расширять существующие политики, определяющие алгоритмы синхронизации между ярусами. Вопрос соответствия файловой системы CephFS требованиям, предъявляемым к многоярусной файловой системе, требует дополнительного исследования.

Заключение

В рамках курсовой работы были выполены следующие задачи:

- сделан обзор существующих распределенных файловых систем, в том числе поддерживающих многоярусность;
- определены несколько вариантов архитектуры многоярусной файловой системы;
- все рассмотренные в работе файловые системы являются решениями с открытым исходным кодом.

В будущем предполагается реализовать один из предложенных вариантов архитектуры многоярусной файловой системы, а в дальнейшем углубиться в оптимизицию созданного Rule Engine.

Список литературы

[1] Ceph Storage. Cache tiering. — URL: http://docs.ceph.com/docs/master/rados/operations/cache-tiering/ (online; accessed: 22.12.2015).

[2] CephFS. Differences from POSIX. — URL: http://docs.ceph.com/docs/hammer/dev/differences-from-posix/ (online; accessed: 22.12.2015).

[3] Cloud Storage for Modern Data Center. An Introduction to Gluster Architecture. — Version 3.1.x.

[4] Depardon Benjamin, Le Mahec Gaël, Séguin Cyril. Analysis of Six Distributed File Systems. — 2013.

[5] Distributed Directories. — URL: https://www-01.ibm.com/support/knowledgecenter/ssw_ibm_i_61/rzahy/rzahydistdir.htm (online; accessed: 22.12.2015).

[6] Dror Feitelson. Workload Modeling for Computer Systems Performance Evaluation. — 2011.

[7] EMC Education Services. Information Storage and Management: Storing, Managing and Protecting Digital Information in Classic, Virtualized and Cloud Environments / Ed. by Somasundaram Gnanasundaram, Alok Shrivastava. — Second edition. — John Wiley & Sons, Inc., 2012. — P. 528.

[8] GlusterFS General FAQ. — URL: http://www.gluster.org/community/documentation/index.php/GlusterFS_General_FAQ (online; accessed: 22.12.2015).

[9] GlusterFS. Tier. — URL: http://gluster.readthedocs.org/en/release-3.7.0/Features/tier/ (online; accessed: 22.12.2015).

[10] OrangeFS Installation Instructions. — Version 2.9.

[11] Parallel File Systems. — URL: http://www.cs.iit.edu/~iraicu/teaching/CS554-F13/lecture17-pfs-sam-lang.pdf (online; accessed: 22.12.2015).

[12] Tiered File System without Tiers. — URL: http://www.snia.org/sites/default/education/tutorials/2011/spring/file/ShepardLaura_Tiered_FileSystem_Without_Tiers_FINAL.pdf (online; accessed: 21.12.2015).