**GlusterFS**

**Что такое GlusterFS?**

GlusterFS — это распределенная, параллельная файловая система с возможностью защиты от сбоев. С помощью нее можно объединить множество хранилищ данных, размещенных на разных серверах (горизонтальное масштабирование) в одну сетевую файловую систему. Так же возможно объединение нескольких хранилищ одного сервера (вертикальное масштабирование). А защита от сбоев достигается с помощью различных политик дублирования данных.

GlusterFS испльзуется в основном в компаниях с большим количеством серверов для распределённого хранения данных и защиты от их потери. Он решает проблему потери данных, которые возможно не являются постоянными.

GlusterFS использует механизмы FUSE (файловая система в пользовательском пространстве) и работает поверх любых POSIX ФС, например Ext3, Ext4, XFS, Btrfs.
В качестве транспорта может использоваться Infiniband RDMA и TCP/IP.

Для работы GlusterFS не требуется отдельный сервер для хранения метаданных. Благодаря отсутствию привязки к централизованному серверу метаданных, ФС предоставляет практически неограниченную масштабируемость. Объём хранилища может измеряться петабайтами. Правда, здесь есть один нюанс. Из-за отсутствия центрального сервера метаданных, поиск файлов происходит на всех серверах и это отрицательно сказывается на производительности. По этой причине, GlusterFS наиболее эффективна для хранения больших файлов т. к. время поиска с лихвой компенсируется временем их загрузки.

Коротко об остальных возможностях ФС:

* Можно использовать любое оборудование (подойдёт даже ARM);
* Работает через стандартные протоколы или нативный клиент;
* Автоматическое обнаружение отказа отдельного хранилища (Brick Failure Detection);
* Возможно сжатие данных при передаче по сети;
* Поддерживается шифрование данных дисковых разделов на стороне сервера с использованием ключей, доступных только клиенту. При этом шифруется только содержимое файлов, имена и метаданные остаются незашифрованными. Шифрование неприменимо при использовании NFS;
* Оптимизирована для использования в качестве распределённого хранилища образов виртуальных машин;
* Может использоваться в качестве первичного хранилища в OpenStack;
* Механизм zerofill позволяет заполнять нулями новые образы виртуальных машин;

**Примечание:** FUSE - это модуль ядра в ОС UNIX и Linux, который является своего рода посредником между пользовательским пространством и пространством ядра. FUSE предоставляет интерфейс ядра (API) с помощью которого непривилегированные процессы получают доступ к низкоуровневым механизмам файловых систем. Благодаря этому, можно написать свою собственную файловую систему которая будет обладать практически всеми возможностями низкоуровневой ФС.

## Архитектура

GlusterFS имеет клиент-серверную архитектуру расширяемую за счет различных модулей (трансляторов).

Серверы здесь — это узлы хранения (Storage Node). На каждом таком узле работает служба glusterd которая предоставляет доступ клиентам к локальным файловым системам. Одна экспортируемая файловая система узла хранения называется «brick» (кирпич). Таких кирпичей на одном сервере может быть множество. Например, на одном диске или массиве может быть несколько разделов с различными файловыми системами, каждая из которых будет отдельным кирпичом. Несколько серверов объединяются в кластер или пул хранения данных (Trusted Storage Pool). В рамках пула хранения, подтома на различных узлах объединяются в логические тома (volumes) различной конфигурации.

Клиентская часть, взаимодействуя с узлами кластера, совершенно прозрачно для приложений монтирует логический том посредством механизма FUSE.
На одном компьютере допускается работа как серверной так и клиентской части.

Трансляторы

Практически вся функциональность GlusterFS реализуется с помощью специальных модулей, называемых трансляторами. Это бинарные динамические объекты каждый из которых реализует определённую функциональность. По сути, данные, записываемые и считываемые с виртуального тома, проходят через целую цепочку трансляторов, которые обрабатывают их по принципу конвейера. Разные тома могут иметь свои наборы трансляторов в зависимости от задачи. Параметры трансляторов определяется в файле конфигурации тома.

Стандартные трансляторы:

* storage – низкоуровневый транслятор, производит запись и чтение данных с локальной файловой системы;
* cluster – занимается репликацией данных и их распределением по кирпичам;
* debug – предоставляет интерфейс и статистику ошибок для отладки;
* encryption – занимается шифрованием и расшифровкой данных на лету;
* protocol – производит аутентификацию между клиентом и сервером;
* pеrformance – транслятор для оптимизации производительности(опережающее чтение (read-ahead) и запаздывающая запись (write-behind));
* system – управление ACL;
* scheduler – глобальный планировщик операций I/O;
* features – квоты, фильтры, и прочее;
* bindings – различные API.

**Сборка**

GlusterFS работает в пространстве пользователя и не требует поддержки со стороны ядра, что позволяет использовать ее практически в любых дистрибутивах. Мы использовали дистрибутив Ubuntu для сборки GlusterFS.

Порядок действий:
1. Мы клонировали репозиторий: [https://github.com/gluster/glusterfs.git](https://vk.com/away.php?to=https://github.com/gluster/glusterfs.git)

2. Установили все необходимые компоненты и библиотеки, необходимые для сборки проекта:

$ sudo apt-get install make automake autoconf libtool flex bison pkg-config libssl-dev libxml2-dev python-dev libaio-dev libibverbs-dev librdmacm-dev libreadline-dev liblvm2-dev libglib2.0-dev liburcu-dev libcmocka-dev

3. Выполнив следующую команду в папке GlusterFS, мы сгенерировали скрипт конфигурации:

$ ./autogen.sh

4.Когда скрипт конфигурации сгенерировался, то мы запустили его, чтобы сгенерировать все MakeFile-ы:

$ ./configure

5. Для сборки и установки мы выполнили 2 команды:

$ make

$ sudo make install

6. Выполнив финальную команду, мы запустили программу GlusterFS:

# glusterd

**Руководство по использованию:**

Прежде всего необходимо открыть порты FireWall-а для корректной работы GlusterFS:

$ iptables –l INPUT –m state—state NEW –m tcp-p tcp—dport 24007:24011 –j ACCEPT

$ iptables –l INPUT –m state—state NEW –m tcp-p tcp—dport 111 –j ACCEPT

$ iptables –l INPUT –m state—state NEW –m udp-p udp—dport 111 –j ACCEPT

После проделанных команд уже можно запускать службу GlusterFS:

$ service glusterd start

Теперь мы можем уже работать в самой программе. Первое, что нужно начать делать – настроить сервер(ы). Для этого мы выбираем 1 из наших будущих серверов или тот, который у нас будет, и создаём его в качестве Master Server-a. Также нужно создать Pool или Pool-ы устройств хранения данных и зайти как корневой пользователь в Master Server-e, выполнив затем следующие команды:

$ gluster peer probe <Вводим IP-адрес>

$ gluster peer status

После ввода первой строки должно вывестись «peer probe: success», если вы ввели

корректный IP-адрес. После ввода второй строки, в консоли будет показано, сколько серверов подключено с клиентами и готовы работать с клиентами в данный момент.

Теперь, когда связь налажена, клиент может создать Volume («Виртуальный том»). Клиент может видеть виртуальный том во всех серверах и может производить различные операции:

Distributed - распределенный том. Тип тома, при котором данные распределяются равномерно (в произвольном порядке) по всем под томам. Например первый файл будет записан на первый сервер а второй файл — на третий. Тома такого типа очень хорошо и легко масштабируются, но никак не защищены средствами GlusterFS. Надежность Distributed тома необходимо обеспечить отдельно на аппаратном или программном уровне. В случае выхода из строя сервера или его дисков, данные находящиеся на нем будут не доступны. Самое интересное в этой схеме, что совершенно непредсказуемо, какие именно данные будут потеряны.

**Replicated Volume** - («Реплицированный виртуальный том») ещё одна опция, которая позволяет теперь каждому серверу запоминать только половину полной информации, что даёт выгоду в безопасности, если есть, например, 4 сервера, то при крахе 1, клиент будет уверен, что данные не потеряются, но зато скорость теперь значительно медленнее, чем в случае с опцией Distributed Volume.

**Striped Volume** - том с чередованием. Наиболее производительный и одновременно самый ненадежный тип. Все поступающие данные разбиваются на части, и параллельно пишутся на разные подтома на разных серверах. При считывании данные в обратном порядке собираются и отдаются клиенту. В результате выхода из строя одного сервера или его диска, том приходит в негодность до восстановления сбойного узла.

Distributed Striped — распределение с чередованием. Здесь как и в случае с Distributed-томом данные распределяются между разными серверами при этом они еще разбиваются на части между несколькими Striped-томами(См. рис. 1).

Distributed Replicated — то же, что и Distributed Striped, только вместо чередования будет использоваться репликация (мм. рис. 2). Этот вариант предоставляет такую же масштабируемость как и простой Distributed том но при этом обладает повышенной надежностью за которую придется платить вдвое большим количеством серверов/дисков. Такая конфигурация рекомендуется разработчиками для высокопроизводительных сред, с повышенными требованиям к надежности.

**Лицензия и ограничения на варианты использования**

GlusterFS имеет лицензию GNU General Public License, которая предоставляет пользователю права копировать, модифицировать и распространять (в том числе на коммерческой основе) программы, а также гарантировать, что и пользователи всех производных программ получат вышеперечисленные права.

**Структура проекта**

Папка glusterfsd – основные файлы программы.

Папка libglusterfs – основные библиотеки.

В остальных папках находятся дополнительные файлы

**Интерфейс**

Интерфейса как такого нет, пользователь выбирает действия программы, с помощью аргументов командной строки.

**Особенности кода**

Код очень хорошо защищен от вылетов, т.к. при выполнении каждой функции выполняются проверки внутри нее, + после ее выполнения задается значение переменной ret.

**Плюсы кода:**

1) Многие функции хорошо закомментированы.

2) Многие переменные и функции имеют хорошие (понятные) называния.

3) В программе хорошо ловятся ошибки.

4)Присутствуют целые блоки кода с пояснениями, что будет делаться дальше во всём коде и объяснение, что происходит при разных значениях ret, например:

success: 0 when successfully parsed

failure: 1 when failed to parse one more more lines

-1 when other critical errors (dlopen () etc)

**Минусы кода:**

1) Большое количество использования goto, что немного запутывает программу.

2) В некоторых названиях функций и переменных используются неочевидные сокращения, что затрудняет перевод.

**Странности и особенности кода:**

1) При объявлении некоторых функций разработчики придерживаются странного код стайла:

int

print\_exports\_file (const char \*exports\_file)

2) Интерфейсы сервера и клиента реализованы в одной программе.

**Изменения в коде:**

В качестве изменения кода мы попробовали изменить некоторые сообщения об ошибках, например «Check that glusterfs daemon is running».

**Выводы по изучению проекта:**

Самый полезный навык, который мы приобрели за время выполнения лабораторной – это умение разбираться в структуре и коде больших проектов, так же мы узнали что такое распределенные файловые системы, как и где они используются. Так же мы научились пользоваться утилитой grep для нахождения файов в определенной папке по его содержимого.

Разбор основной функции проекта находится в файле “Функция main”.

**Используемая литература:**

“Анализ программного кода на примере Open Source”, Скотт Мейерс.