Примеры инфраструктурных решений, применяющихся в крупных сетевых проектах

Пример реализации инфраструктуры в Google

Распределенная инфраструктура в Google состоит из 3 основных компонентов: GFS, MapReduce и BigTable.

Распределенная файловая система GFS (Google File System)

GFS - большая распределенная файловая система, способная хранить и обрабатывать огромные объемы информации.

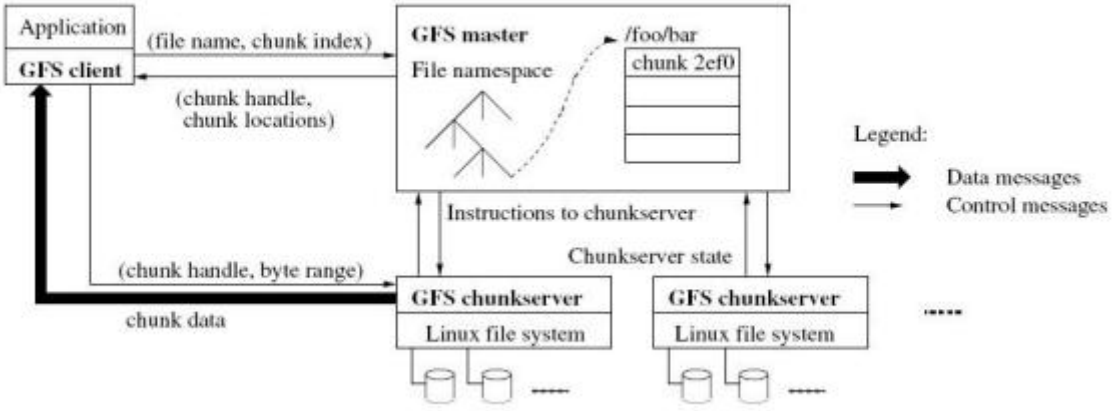
Удовлетворяет следующим критериям:

- Существует мониторинг сбоев и возможность восстановления функционирования системы в случае отказа оборудования.
- Оптимизировано хранение данных большого размера.
- Отточен алгоритм чтения файлов различного размера.
- Оптимизирована запись файлов различного размера.
- Строго очерчена семантика параллельной работы (параллельная работа клиентов).
- Высокая пропускная способность.

Организация файлов:

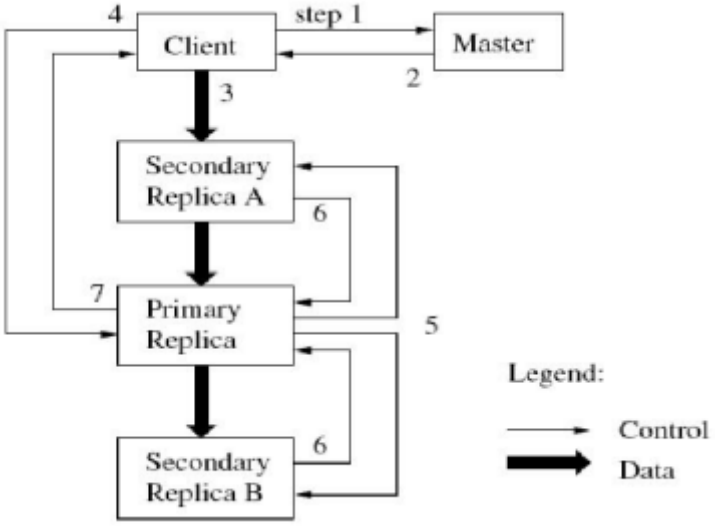
Файлы в GFS организованы иерархически, при помощи каталогов, как и в любой другой файловой системе, и идентифицируются своим путем. С файлами в GFS можно выполнять обычные операции: создание, удаление, открытие, закрытие, чтение и запись.

Архитектура GFS:



В системе существуют мастер-сервера и чанк-сервера, собственно, хранящие данные. Как правило, GFSкластер состоит из одной главной машины мастера (master) и множества машин, хранящих фрагменты файлов чанк-серверы (chunkservers).

Взаимодействия внутри системы:



Операции, выполняемые мастером:

- Управление репликациями чанков, принятие решений о размещении и создание новых чанков.
- Координация деятельности для поддержания полной репликации чанков, балансировки нагрузки и сборки неиспользуемых ресурсов.
- Управление логическим представлением пространства имен как таблицы метаданных.
- Установка блокировок чтения-записи для операций мастера, включая блокировки на уровнях директорий и файлов.
- Определение политики расположения реплик для максимизации надежности и доступности данных, а также оптимизация использования сетевой пропускной способности.
- Выбор расположения реплик при создании нового чанка с учетом загруженности дисков и ограничения числа создаваемых чанков на каждом чанк-сервере.
- Распределение чанков между разными стойками для повышения надежности.
- Репликация чанков происходит при падении числа реплик ниже установленного значения.
- Установка приоритетов для чанков, учитывая количество реплик и блокирование прогресса работы клиента.
- Копирование чанков с высоким приоритетом для поддержания репликации, основываясь на выборе чанков с наибольшим приоритетом.
- Балансировка реплик для выравнивания загруженности дисков и обеспечения надежности системы.
- Удаление реплик с наименьшим свободным местом на жестких дисках чанк-сервера.

- Сборка мусора, включая удаление скрытых файлов, помеченных на удаление, и определение чанков, которые можно удалить для оптимизации использования пространства.
- Регулярное сканирование пространства имен файлов и чанков для удаления отмеченных на удаление данных, с учетом заданного интервала времени.
- Переименование файлов при удалении, добавление времени удаления и скрытие от пользователя до окончательного удаления мастером.
- Восстановление удаленных данных, гибкая балансировка нагрузки при удалении и восстановление системы в случае сбоев

Устойчивость к сбоям и диагностика ошибок в GFS:

GFS поддерживает систему в рабочем виде при помощи двух простых стратегий: *быстрое восстановление* и *репликация*.

- Быстрое восстановление включает перезагрузку машины с минимальной заминкой.
- Репликация чанков происходит при недоступности или повреждении реплик, выявленных с использованием контрольных сумм.

Репликация мастера:

- Реплируются лог операций и контрольные точки мастера.
- Лог операций записывается перед любым изменением файлов, обеспечивая целостность данных.
- При проблемах с жестким диском или другой важной инфраструктурой мастера, запускается новый мастер.

Поддержание целостности данных:

- Каждый чанк разбивается на блоки, каждому из которых соответствует 32-битная контрольная сумма.
- Контрольные суммы хранятся в памяти и периодически сохраняются в лог.
- Чанк-сервера самостоятельно проверяют целостность данных с использованием контрольных сумм.
- При обнаружении несовпадения контрольных сумм, чанк-сервер возвращает ошибку, и мастер решает проблему.

Верификация контрольных сумм при чтении и записи:

- Перед чтением данных чанк-сервер проверяет контрольные суммы блоков.
- При обнаружении несовпадения возвращается ошибка, и мастер принимает меры по восстановлению.

- При записи новые контрольные суммы записываются, а верификация осуществляется при попытке чтения.

Добавление новых данных:

- При добавлении новых данных верификация контрольных сумм не выполняется.
- При записи сравниваются только первый и последний блоки, поскольку часть данных на этих блоках не перезаписывается.

Организация работы с данными при помощи MapReduce

MapReduce является программной моделью и соответствующей реализацией обработки и генерации больших наборов данных.

- Реализация, автоматически распараллеливающая и адаптирующая программы для выполнения на распределенных кластерах.

Преимущества:

- Эффективный способ распределения задач между множеством компьютеров.
- Обработка сбоев в работе.
- Работа с различными типами смежных приложений.
- Вычисления автоматически приближаются к источнику ввода-вывода

Использует три типа серверов:

- Master - управляет заданиями.
- Map - обрабатывает входные данные и записывает промежуточные файлы.
- Reduce - сокращает промежуточные файлы.

Технология MapReduce:

- Map отображает один набор данных в другой, создавая пары ключ/значение.
- Reduce суммирует результаты, возвращая финальный результат.

Для оптимизации пропускной способности канала и ввода-вывода транспортировка данных между серверами происходит в сжатом виде.

Хранение структурированных данных в BigTable

BigTable является крупномасштабной, устойчивой к потенциальным ошибкам, самоуправляемой системой, которая может включать в себя терабайты памяти и петабайты данных, а также управлять миллионами операций чтения и записи в секунду. BigTable представляет собой распределенный механизм хэширования, построенный поверх GFS.

Особенности:

- Не реляционная база данных, отсутствие поддержки SQL и операций типа Join.
- Предоставляет механизм просмотра данных по ключу для доступа к структурированным данным.

Структура данных в BigTable:

- Каждый блок данных хранится в ячейке, доступ к которой по ключу строки/столбца и временной метке.
- Строки могут храниться в одной или нескольких таблицах.
- Таблицы организованы в виде последовательности блоков по 64 килобайта в формате SSTable.

Типы серверов в BigTable:

- Master: распределяет таблицы по Tablet-серверам, следит за их расположением.
- Tablet: обрабатывает запросы чтения/записи, разделяет таблицы при необходимости.
- Lock: обеспечивает распределенный сервис ограничения одновременного доступа.

Оптимизация хранения данных:

- Локальная группировка для физического хранения связанных данных вместе.
- Кэширование таблиц в оперативной памяти серверов для оптимизации доступа к данным.

Пример реализации инфраструктуры для проекта Flickr

На проекте Flickr используется практически только свободное программное обеспечение.

Использующиеся программные компоненты:

- GNU/Linux (RedHat);
- СУБД MySQL;
- Web-сервер Apache;
- Скрипты программной логики, написанные на языке PHP и Perl;
- Средства сегментирования (Shards);
- Memcached для кэширования часто востребованного контента;
- Squid в качестве обратного прокси-сервера для html-страниц и изображений;
- Шаблонизатор Smarty PEAR для парсинга e-mail и XML;
- ImageMagick для обработки изображений;
- SystemImager для развертывания элементов конфигурации;

- Ganglia для мониторинга распределенных систем;
- Subcon для хранения важных системных конфигурационных файлов в SVN-репозитории для легкого развертывания на машины в кластере;
- Cvsup для распространения и обновления коллекций файлов по сети.

Системная архитектура:

Обработка запросов:

- Входные запросы поступают на дублированные контроллеры приложений Brocade ServerIron ADX.
- Коммутация приложений и балансировка трафика осуществляются на основе виртуальных ферм серверов.

Выбор сервера:

- Коммутатор приложений выбирает "лучший" сервер на основе Real-Time Health и требуемой производительности.
- Интеллектуальное распределение загрузки для всех доступных ресурсов.

Управление сессиями:

- Обслуживание сессий ведется последовательно.
- Записи о пользователях создаются в таблице, каждая сессия назначается определенному серверу.
- Синхронизация таблиц сессий между коммутаторами.

Хранение данных:

- Центральная база данных включает таблицу пользователей, за исключением фотографий.
- Фотографии хранятся в системе хранения данных.

Сегментация:

- Реализованы федеративные принципы сегментации данных.
- Глобальное кольцо, аналогичное DNS, для определения места хранения данных.
- Логика, реализованная в виде PHP скриптов, поддерживает соединение с сегментом и целостность данных.
- Каждый сервер в рамках сегмента в нормальном состоянии нагружен ровно на половину.

Организация резервного копирования данных реализована с помощью процесса ibbackup, который выполняется регулярно посредством cron daemon'a, причем на каждом сегменте он настроен на разное время. Каждую ночь делается снимок со всего кластера баз данных.