Елкина Галя, 2пг
Отчет 3.3

Примеры инфраструктурных решений, применяющихся в крупных сетевых проектах

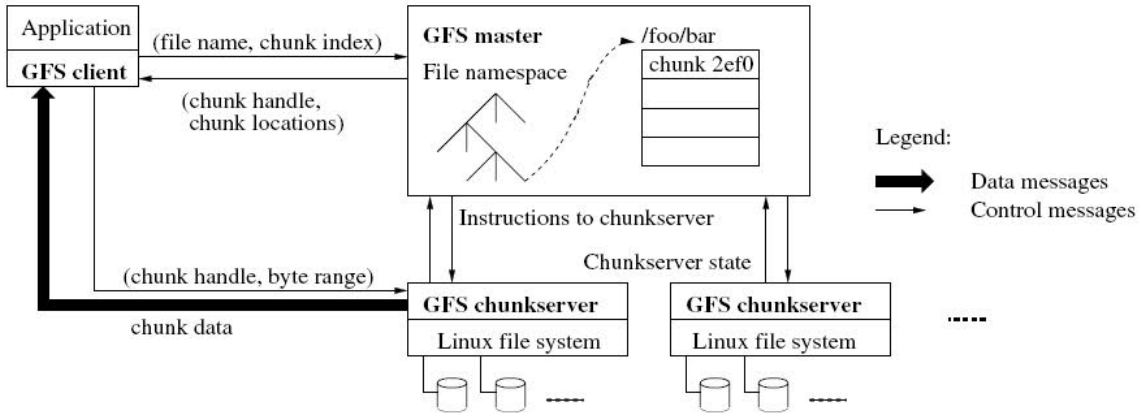
Пример реализации инфраструктуры в Google

Распределенная файловая система GFS (Google File System) — большая распределенная файловая система, способная хранить и обрабатывать огромные объемы информации.

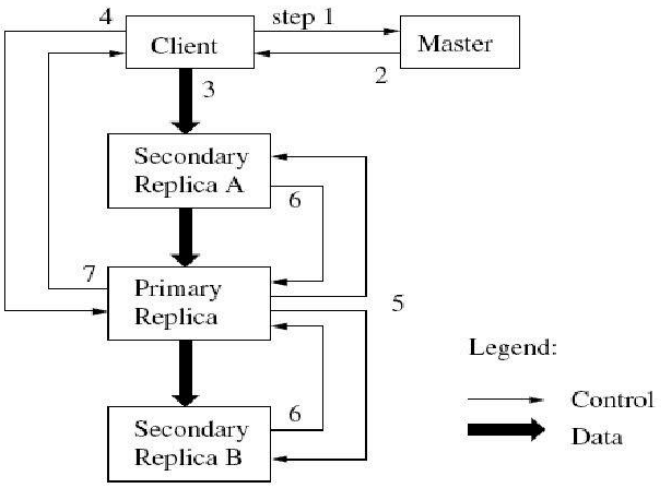
Критерии построения системы:

- мониторинг сбоев и восстановление системы;
- оптимизация работы системы;
- корректный алгоритм чтения файлов;
- оптимизация загрузки файлов;
- корректная работа в режиме совместного использования;
- высокая пропускная способность.

Архитектура GFS



Взаимодействия внутри системы



Операции, выполняемые мастером

Мастер управляет репликациями чанков: принимает решения о размещении, создает новые чанки, а также координирует различную деятельность внутри системы для сохранения чанков полностью реплицированными, балансировки нагрузки на чанк-серверы и сборки неиспользуемых ресурсов.

GFS представляет пространство имен как таблицу. Каждая вершина в этом дереве (соответствует либо абсолютному пути к файлу, либо к директории) имеет соответствующие данные для блокировки чтения и записи. Каждое операция мастера требует установления некоторых блокировок.

Кластеры GFS, являются сильно распределенными и многоуровневыми. Соединения между двумя машинами из различных стоек может проходить через один или несколько свитчей.

Политика расположения реплик старается удовлетворить следующим свойствам: максимизация надежности и доступности данных и максимизация использование сетевой пропускной способности.

Когда мастер создает чанк, ему нужно:

- новую реплику поместить на чанк-сервер с наименьшей средней загруженностью;
- ограничить число новых создаваемых чанков;
- распределить чанки среди разных стоек.

Как только число реплик падает ниже устанавливаемой пользователем величины, мастер снова реплицирует чанк. Это может случиться по нескольким причинам: чанк-сервер стал недоступным, один из дисков вышел из строя или увеличена величина, задающая число реплик. Каждому чанку, который должен реплицироваться, устанавливается приоритет, который тоже зависит от нескольких факторов. Во-первых, приоритет выше у того чанка, который имеет наименьшее число реплик. Во-вторых, чтобы увеличить надежность выполнения приложений, увеличивается приоритет у чанков, которые блокируют прогресс в работе клиента.

Другие функции мастера:

- балансирование реплик;
- сборка мусора;
- сканирование пространства имен файлов;
- сканирование пространства имен чанков;

Устойчивость к сбоям и диагностика ошибок

Рабочее состояние системы поддерживается двумя способами:

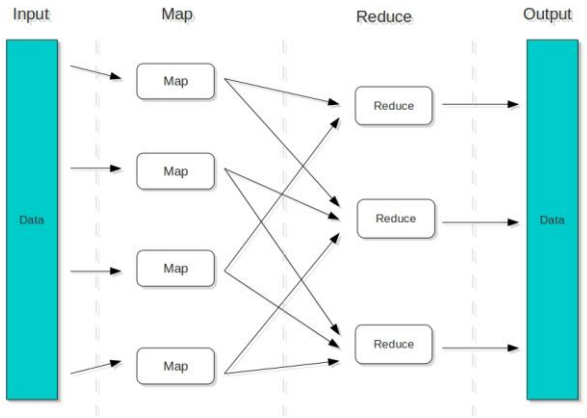
- Быстрое восстановление — это, фактически, перезагрузка машины.
- А также репликация чанков мастером. Поврежденные чанки определяется при помощи вычисления контрольных сумм.

Также есть репликация мастера. В случае небольших неполадок мастер может перезагрузиться. В случае отказа старого мастера, запускается новый, являющийся тенью старого, а не точной копией. То есть он выполняет функции старого мастера, пока тот не «починится».

Важной частью системы является возможность поддерживать целостность данных. Система может восстановить данные с помощью репликаций, но для этого необходимо понять испортились ли данные. Поэтому перед тем, как считать данные чанк-сервер проверяет контрольные суммы блоков чанка. И если они не совпадают, то на чанк-сервер возвращает ошибку машине и с ней уже работает мастер.

Организация работы с данными при помощи MapReduce

MapReduce является программной моделью и соответствующей реализацией обработки и генерации больших наборов данных. Программы, написанные в таком функциональном стиле, автоматически распараллеливаются и адаптируются для выполнения на обширных кластерах. Система берет на себя детали разбиения входных данных на части, составления расписания выполнения программ на различных компьютерах, управления ошибками, и организации необходимой коммуникации между компьютерами.



Преимущества использования MapReduce:

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

MapReduce использует три типа серверов:

- Master: назначают задания остальным типам серверов, а также следят за процессом их выполнения.
- Map: принимают входные данные от пользователей и обрабатывают их, результаты записываются в промежуточные файлы.
- Reduce: принимают промежуточные файлы от Map-серверов и сокращают их указанным выше способом.

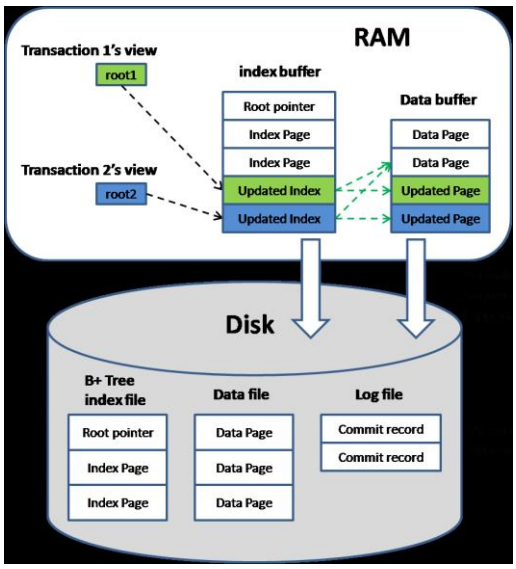
Последовательность выполняемых действий выглядела бы следующим образом: GFS → Map → перемешивание → Reduce → запись результатов обратно в GFS.

Транспортировка данных между серверами происходит в сжатом виде. Идея заключается в том, что ограничивающим фактором является пропускная способность канала и ввода-вывода, что делает резонным потратить часть процессорного времени на компрессию и декомпрессию данных.

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, а вовсе не реляционную базу данных и, как следствие, не поддерживает SQL-запросы и операции типа Join. Она предоставляет механизм просмотра данных для получения доступа к структурированным данным по имеющемуся ключу.

С помощью контролирования своих низкоуровневых систем хранения данных, Google получает больше возможностей по управлению и модификации их системой.



В BigTable тоже используется три типа серверов:

- Master: распределяют таблицы по Tablet-серверам, а также следят за расположением таблиц и перераспределяют задания в случае необходимости.
- Tablet: обрабатывают запросы чтения/записи для таблиц. Они разделяют таблицы, когда те превышают лимит размера (обычно 100-200 мегабайт). Когда такой сервер прекращает функционирование по каким-либо причинам, 100 других серверов берут на себя по одной таблице и система продолжает работать как будто ничего не произошло.
- Lock: формируют распределенный сервис ограничения одновременного доступа. Операции открытия таблицы для записи, анализа Master-сервером или проверки доступа должны быть взаимоисключающими.

Пример реализации инфраструктуры для проекта Flickr

Flickr является мировым лидером среди сайтов размещения фотографий.

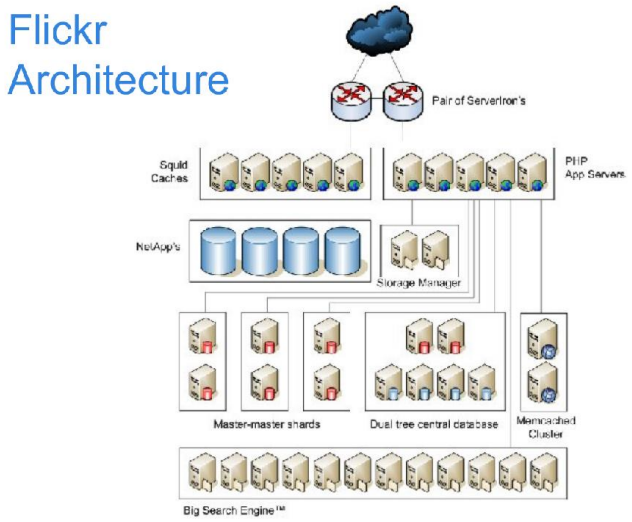
Использующиеся программные компоненты

Примечательно, что на проекте Flickr используется практически только свободное программное обеспечение.

- Платформа GNU/Linux (RedHat)

- СУБД MySQL
- Web-сервер Apache
- Скрипты программной логики, написанные на языке PHP и Perl
- Средства сегментирования (Shards) (прим.: разбиение системы на части, обслуживающие каждая свою группу пользователей; называть можно было по-разному, но давайте остановимся на этом варианте перевода)
- 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



Входные запросы поступают на сдублированные контроллеры приложений Brocade ServerIron ADX. Они обеспечивают коммутацию приложений и балансировку трафика, основываясь на принципе виртуальных ферм серверов.

Коммутатор приложений получает все клиентские запросы. Выбор “лучшего” сервера производится на основании механизма Real-Time Health и наличия требуемой производительности. Последовательно повышается коэффициент использования для всех серверов. Интеллектуальное распределение загрузки осуществляется для всех доступных ресурсов. Метод конфигурируется и выбирается пользователем. Обеспечивается защита серверной фермы от атак и от неправильной эксплуатации. Клиенты подсоединяются к серверам приложений используя виртуальный IP (VIP). VIP адреса настраиваются на коммутаторе приложений. Коммутатор приложений осуществляет трансляцию адресов после выбора нужного сервера, причем сами адреса серверов скрыты.

Обслуживание сессий ведется согласно последовательности. Запись о каждой пользовательской сессии создается в таблице. Каждая сессия назначается определенному серверу. Все сообщения в рамках сессии посылаются к одному серверу. Таблицы сессий синхронизируются между двумя коммутаторами. За счет дублирования коммутаторов нет простоя сервиса, когда коммутатор вышел из строя: второй коммутатор обнаруживает отказ и начинает обслуживать сессии пользователей.

Центральная база данных включает в себя таблицу пользователей. Все, за исключением фотографий, хранится в базе данных. Фотографии хранятся в системе хранения данных.

В основе масштабируемости лежит репликация. Для поиска по определенной части базы данных создается отдельная копия этого фрагмента. Активная репликация производится по принципу мастер-мастер. Автоматическое инкрементирование идентификационных номеров используется для поддержания системы в режиме одновременной активности обоих серверов в паре.

В системе заложены федеративные принципы сегментации: "Мои данные хранятся на моем сегменте, но запись о Вашем комментарии хранится на Вашем сегменте". При этом реализуется глобальное кольцо, принцип работы которого схож с DNS: "Необходимо знать куда Вы хотите пойти и кто контролирует то место, куда Вы собираетесь пойти". Логика, реализованная в виде PHP скриптов, устанавливает соединение с сегментом и поддерживает целостность данных.

Каждый сервер в рамках одного сегмента в обычном 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.