**Практическое занятие номер 9**

**Пример реализации инфраструктуры в Google**

Система включает в себя около 1 миллиона недорогих серверов (почти 2 % всех серверов в мире).

Google включает в себя более 400 GFS кластеров. Один кластер может состоять из 1000 или даже 5000 компьютеров.

Десятки и сотни тысяч компьютеров получают данные из GFS кластеров, которые насчитывают более 10 петабайт дискового пространства. Суммарные пропускная способность операций записи и чтения между дата центрами может достигать 40 гигабайт в секунду

Система BigTable позволяет хранить миллиарды ссылок (URL), сотни терабайт снимков со спутников, а также настройки миллионов пользователей.

Google визуализирует свою инфраструктуру в виде трехслойного стека:

* Продукты: поиск, реклама, электронная почта, карты, видео, чат, блоги и т.п.
* Распределенная инфраструктура системы: GFS, MapReduce и BigTable
* Вычислительные платформы: множество компьютеров во множестве датацентров

GFS строилась исходя из следующим критериев:

Система строится из большого количества обыкновенного недорого оборудования, которое часто дает сбои. Должны существовать мониторинг сбоев, и возможность в случае отказа какого-либо оборудования восстановить функционирование системы.

Система должна хранить много больших файлов. Как правило, несколько миллионов файлов, каждый от 100 Мб и больше. Также часто приходится иметь дело с многогигабайтными файлами, которые также должны эффективно храниться. Маленькие файлы тоже должны храниться, но для них не оптимизируется работа системы.

Как правило, встречаются два вида чтения: чтение большого последовательного фрагмента данных и чтение маленького объема произвольных данных. При чтении большого потока данных обычным делом является запрос фрагмента размером в 1Мб и больше. Такие последовательные операции от одного клиента часто читают подряд 110 идущие куски одного и того же файла. Чтение небольшого размера данных, как правило, имеет объем в несколько килобайт. Приложения, критические по времени исполнения, должны накопить определенное количество таких запросов и отсортировать их по смещению от начала файла. Это позволит избежать при чтении блужданий вида назад-вперед.

Часто встречаются операции записи большого последовательного куска данных, который необходимо дописать в файл. Обычно, объемы данных для записи такого же порядка, что и для чтения. Записи небольших объемов, но в произвольные места файла, как правило, выполняются не эффективно.

Система должна реализовывать строго очерченную семантику параллельной работы нескольких клиентов, в случае если они одновременно пытаются дописать данные в один и тот же файл. При этом может случиться так, что поступят одновременно сотни запросов на запись в один файл. Для того чтобы справится с этим, используется атомарность операций добавления данных в файл, с некоторой синхронизацией. То есть если поступит операция на чтение, то она будет выполняться, либо до очередной операции записи, либо после.

Высокая пропускная способность является более предпочтительной, чем маленькая задержка. Так, большинство приложений в Google отдают предпочтение работе с большими объемами данных, на высокой скорости, а выполнение отдельно взятой операции чтения и записи, вообще говоря, может быть растянуто.

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

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 раделяют 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 взаимноисключающими.

**Пример реализации инфраструктуры для проекта Flickr**

Статистика

* Более четырех миллиардов запросов в день
* Примерно 35 миллионов фотографий в кэше прокси-сервера Squid
* Около двух миллионов фотографий в оперативной памяти Squid
* Всего приблизительно 470 миллионов изображений, каждое представлено в 4 или 5 размерах
* 38 тысяч запросов к memcached (12 миллионов объектов) 2 петабайта дискового пространства
* Более 400000 фотографий добавляются ежедневно

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.

Типовое оборудование для серверов:

* EMT64 под управлением RHEL 4 с 16 Gb оперативной памяти.
* 6 жестких дисков с 15000rpm, объединены в RAID-1+0.
* Размер для пользовательских метаданных достигает 12 терабайт (это не включает фотографии).
* Используются 2U корпуса.