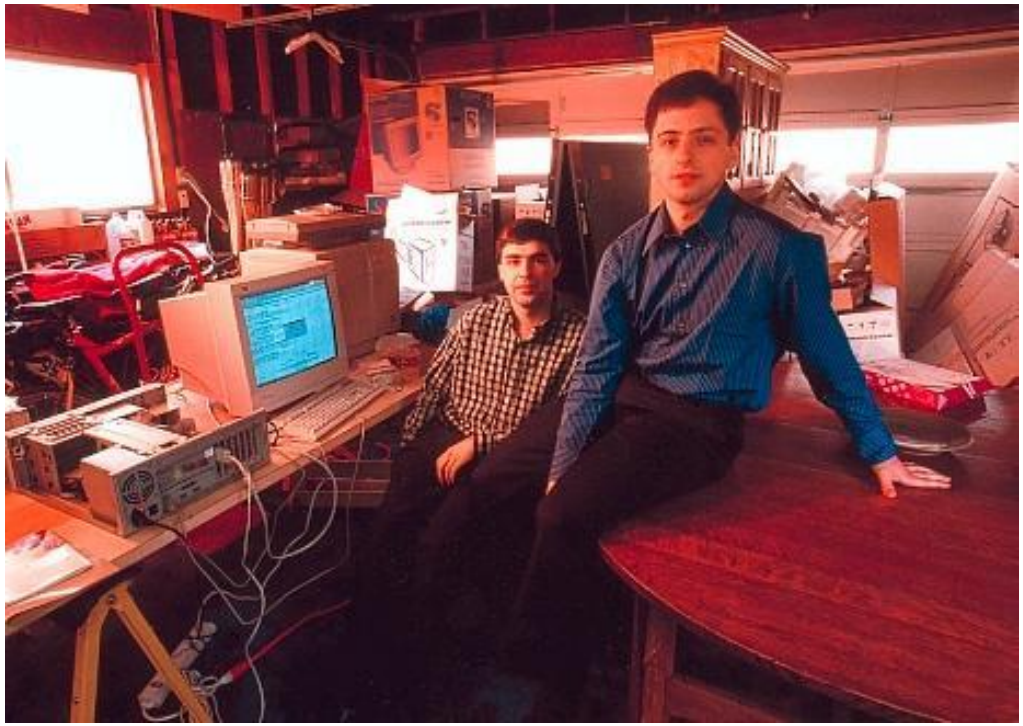
Технологии распределённой обработки данных компании Google

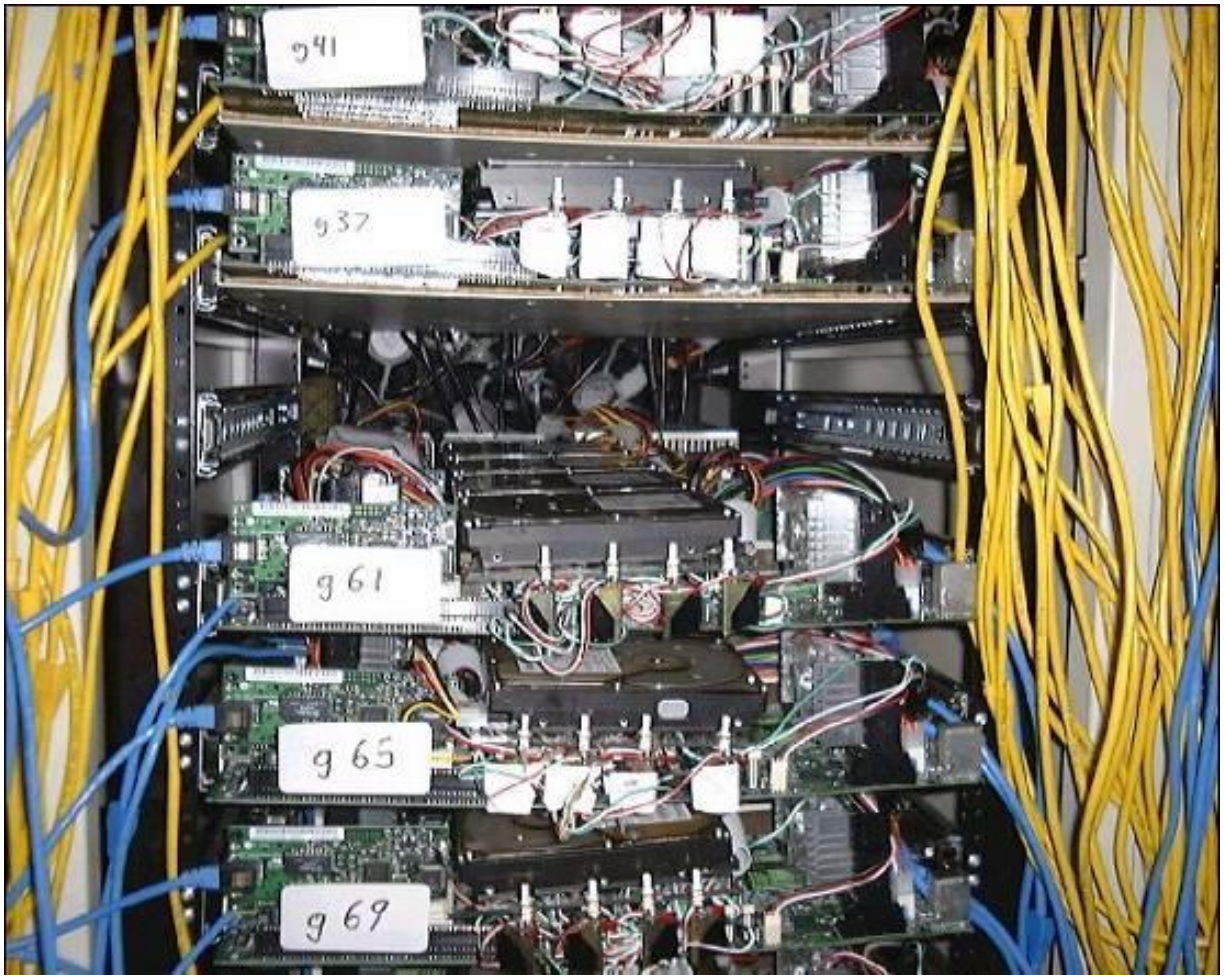


Начало истории Google

- 1996 г. - Ларри Пейдж (Larry Page) и Сергей Брин в Стэнфордский университет. Начало разработки собственно поисковой системы. Основная задача – упорядочение (ранжирование) документов по релевантности. В основу разработанного алгоритма была положена работа по ранжированию научных статей.
- 1997 г. – алгоритм Page Rank. Ранг web-страницы обпределяется принцип взвешенного голосования.
- 1998 г. - Google запущен на сервере Стэнфордского университета, и его можно найти по адресу google.Stanford.edu.

Начало истории Google. 1997 г.

Начало истории Google. 1999 г.



Начало истории Google. 2000 г.

Начало истории Google. 2001 г.

Главная цель компании Google

Организовать всю информацию в мире и сделать ее доступной.

При попытке достичь данной цели возникают три основные задачи:

1. Организация обработки информации при постоянном увеличении её объемов.
2. Возможность обеспечить информацией всех желающих при постоянном увеличении числа пользователей.
3. Повышение качества организуемой информации и предоставляемых услуг.

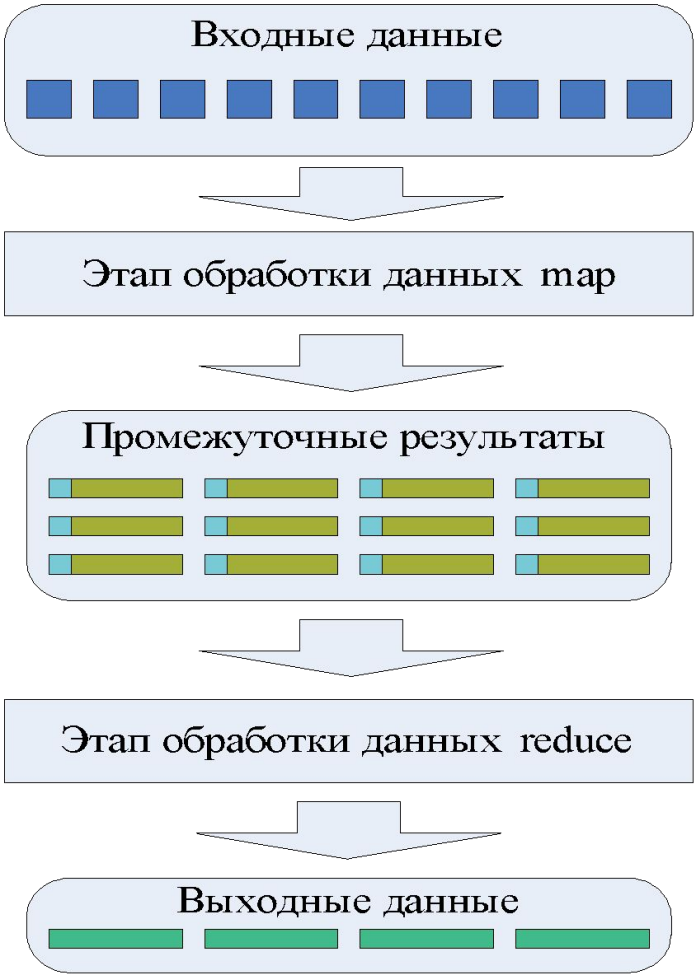
Составляющие системы распределённой обработки данных Google

- Map Reduce – технология распределённой обработки данных и модель параллельного программирования.
- Chubby – высоконадежная распредёленная служба синхронизации и хранения большого числа маленьких файлов.
- Google File System (GFS) – специализированная распределённая файловая система.
- BigTable – специализированная распределённая база данных.

Модель программирования MapReduce

- разработана на основе современных представлений о проектировании программных интерфейсов;
- учитывает необходимость поддержки разработанных программ и минимизации объема кода;
- освоение модели не требует глубоких навыков в области параллельного программирования;
- все особенности реализации алгоритма, связанные распределением вычислений скрыты от пользователя программного интерфейса, который должен заботиться только о решении задачи.

Обработка данных в модели программирования MapReduce



Программист определяет две функции:

map () - формирование множества промежуточных результатов типа «ключ-значение».

reduce() - объединение результатов для каждого значения ключа.

Модель программирования MapReduce

- Ограничения по использованию модели программирования

Пример решения задачи в модели программирования MapReduce

Исходные данные задачи:

Я два раза не повторяю! Не повторяю …

Подготовка данных для передачи на MapReduce-обработку:

```
char inputData [] = {  'Я', ' ', 'д', 'в',
'а', ' ', 'р', 'а', 'з', 'а', ' ', 'н',
'е', ' ', 'п', 'о', 'в', 'т', 'о', 'р',
'я', 'ю', '!', ' ', 'Н', 'е ', ' ',  'п',
'о', 'в', 'т', 'о', 'р', 'я', 'ю', ' ',
'.', '.', '.' };
```

Пример решения задачи в модели программирования MapReduce

Функция map():

```
list< pair< char, int > > map ( char inputChar )
{
   list< pair< char, int > > result;
   result.push_back( pair< char, int >(inputChar, 1 ));
   return result;
}
```

Промежуточный результат обработки:

```
{  ('Я', 1), (' ', 1), ('д', 1), ('в', 1), ('а', 1),
   (' ', 1), ('р', 1), ('а', 1), ('з', 1), ('а', 1),
   (' ', 1), ('н', 1), ('е', 1), (' ', 1), ('п', 1),
   ('о', 1), ('в', 1), ('т', 1), ('о', 1), ('р', 1),
   ('я', 1), ('ю', 1), ('!', 1), (' ', 1), ('Н', 1),
   ('е', 1), (' ', 1), ('п', 1), ('о', 1), ('в', 1),
   ('т', 1), ('о', 1), ('р', 1), ('я', 1), ('ю', 1),
   (' ', 1), ('.', 1), ('.', 1), ('.', 1) }
```

Пример решения задачи в модели программирования MapReduce

Функция reduce():

```
pair< char, int > reduce ( char key,
                           list< int > subproduct )
{
   pair< char, int > result ( key, 0 );
   for ( int i = 0; i < subproduct.size(); ++i )
      result.second += subproduct[ i ];
   return result;
}
```

Конечный результат обработки:

```
{  ('Я', 1), (' ', 7), ('д', 1), ('в', 3), ('а', 3),
('р', 3), ('з', 1), ('н', 2), ('е', 2), ('п', 2),
('о', 4), ('т', 2), ('я', 2), ('ю', 2),   ('!', 1),
('.', 3) }
```

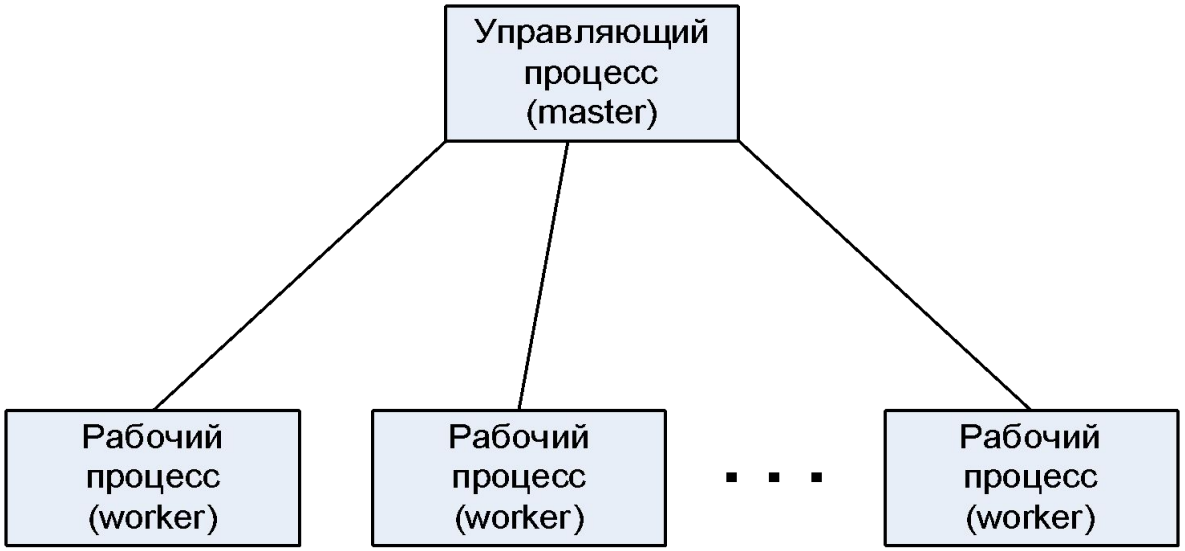
Технология распределённой обработки данных MapReduce

Аппаратная платформа - большое число недорогих серверов из массовых комплектующих:

- двухпроцессорные x86 ЭВМ, оперативная память 4-8 Гб;
- среда передачи данных - Gigabit Ethernet;
- внешняя память - недорогие жёсткие диски с интерфейсом IDE.

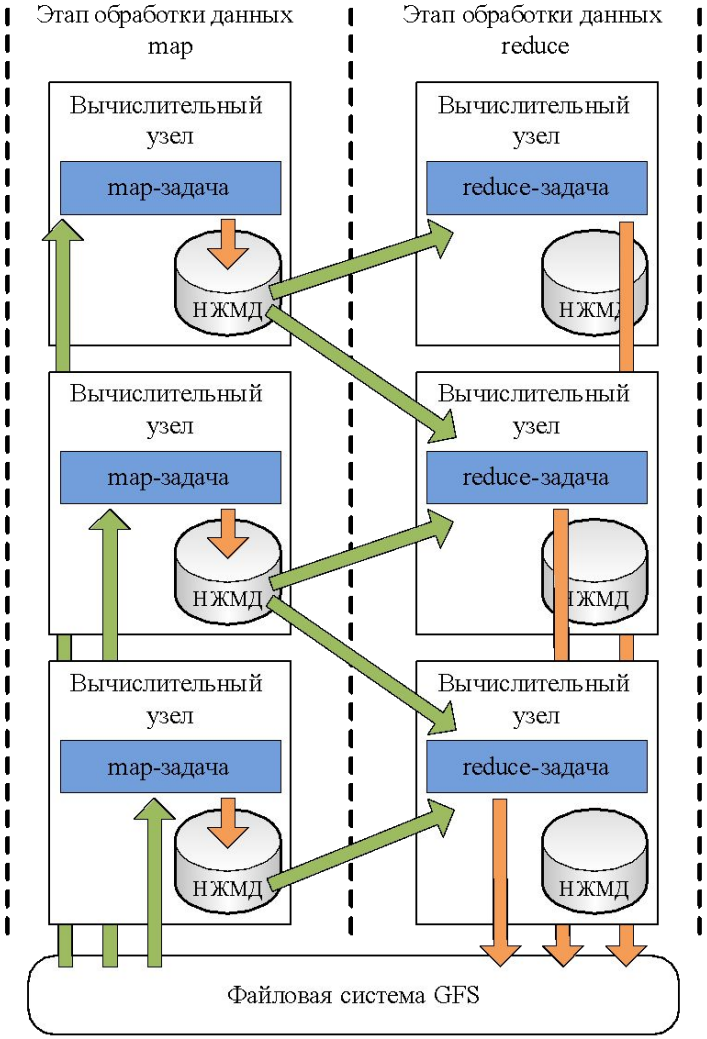
Операционная система Linux.

Архитектура MapReduce кластера



- Выбор управляющего процесса, отслеживание состояния процессов и их синхронизация осуществляется с помощью службы Chubby.
- Управляющий процесс не осуществляет обработку данных.
- На одном вычислительном узле могут быть одновременно запущены процессы MapReduce, GFS и BigTable, – как правило так и происходит.

Процесс обработки данных по технологии MapReduce



- Разбиение входных данных на части (16-64 Мбайт).
- Запуск копий службы MapReduce на всех машинах кластера выбор управляющего процесса.
- Запуск map-задач на рабочих процессах.
- Разбиение результатов выполнения map-задач на части и сохранение на локальных носителях вычислительных узлов в отсортированном виде.
- Запуск reduce-задач на рабочих процессах.
- Сбор данных для выполнения reduce-задач с локальных носителей вычислительных узлов.
- Сохранение результатов обработки в распределённой файловой системе GFS.

Обеспечение отказоустойчивости в технологии MapReduce

- Отслеживание состояния рабочих процессов управляющим процессом и выявление фактов отказа вычислительных узлов.
- Выполненные map-задачи в случае отказа вычислительного узла перезапускаются.
- Выполненные reduce-задачи в случае отказа вычислительного узла не перезапускаются, - данные уже сохранены в GFS.
- Выполнявшиеся и map-, и reduce-задачи перезапускаются.
- Отказ управляющего процесса определяется с помощью механизма исключающей блокировки с периодическим продлением сессии реализованного в службе Chubby.

Оптимизация процесса обработки данных в технологии MapReduce

- Запуск map-задач на тех машинах, где размещены входные данные для этих задач.
- Локальная редукция данных - использование дополнительной функции combiner() для объединения промежуточных результатов обработки сформированных одной map-задачей.
- Помощь отстающему процессу – запуск дополнительных задач для данных, обработка которых задерживается по каким-либо причинам.
- Пропуск (игнорирование) некорректно обрабатываемых пар.