Map-Reduce в домашних условиях

Что такое Map-Reduce

- Парадигма программирования для обработки больших объемов данных
- В промышленном программировании появилась благодаря компании Google
- Сейчас используется многими компаниями для обработки логов, лайков, нахождения похожих объектов, и.т.д

Содержание

- Map-Reduce с точки зрения программиста, прикладного и системного
- Map-Reduce в домашних условиях
- Map-Reduce в облачной инфраструктуре

Логическая модель

- Какой-то набор исходных данных
  - веб страницы, логи
- Три стадии: Map (маппинг, разбиение), Shuffle (сортировка), Reduce (сборка, свертка)
- На стадии Map входные данные преобразовываются в пары (ключ, значение)
- На стадии Shuffle значения с одинаковым ключом группируются в единый список
- На стадии Reduce обрабатывается список значений с одинаковым ключом.

Модель для прикладного программиста

- Функция map(input) → [(key, value)]
- Функция reduce(key, [value]) → result
- A shuffle пусть просто будет

Что в плюсе?

- Простая модель программирования
- Если задача "хорошая", то можно поделить входные данные на части и запустить параллельно несколько map и reduce процессов
- Код функций при этом не меняется
- Данных стало в два раза больше? Ок, увеличим число map-reduce процессов вдвое

Что в минусе?

- Не все задачи так хорошо распараллеливаются
- Как дирижировать работой десятков map-reduce процессов?
- Кто же, черт возьми, напишет shuffle?
- И где взять столько процессорных ядер?

Взгляд системного программиста

- Реализовать сортировку!
- Предоставить прикладнику способ прочитать всевозможные входные данные (реализации input reader'ов)
- Реализовать диспетчера, рассылающего задания мапперам, координирующего сортировку и свертку
- А в больших кластерах есть еще тысяча мелочей:
  - автоматически стартовать map-reduce процессы
  - обрабатывать сбои оборудования (они происходят постоянно)

Как-то все сложно

- Map-Reduce в больших системах -- сложный инфраструктурный компонент
- Неужели нельзя как-нибудь попроще? У нас вот тут четыре компа и они не ломаются так часто.

Mincemeat: poor man's mapreduce

- 450 строчек на питоне (диспетчер и сортировка)
- код функций mapreduce -- несколько строчек
- любой компьютер в любой момент может стать частью кластера, достаточно знать IP диспетчера и пароль

Прикладная задача

- Раздобыли часть корпуса Book n-grams: 2-граммы начинающиеся с букв "gr"
- ~1G текста
- Хотим посчитать все начальные буквенные униграммы и биграммы в слове после слова "great"
  - great britain
  - great wall
  - great work

  "b": 1 "w":2 "br": 1, "wa": 1, "wo": 2

Кто справится быстрее всех?!

Забег первый

- На вход самому первому мапперу будет подаваться одна строчка из файла

```
Great persons_NOUN      1768    1       1
```

- Маппер будет выплевывать единичку для униграммы и биграммы если строчка начинается с "great"
- Поехали!

Забег первый: результаты

- Разминочный корпус состоит из 4M текста
- Один маппер справился с 4M за минуту с небольшим
- Как-то медленно, не?
- Эталонная программа безо всякого map-reduce справилась за полсекунды

Эээээээ.....

Забег второй

- Давайте будем давать мапперу на вход все содержимое файла
- Маппер будет суммировать число найденных униграмм и биграмм и выплевывать на выход все полученные пары
- 1 секунда -- уже лучше!
- Как будет на 1G?
  - Эталонная программа прошла за минуту с небольшим
  - Один mapreduce процесс справился за полторы минуты
  - Значит надежда есть

Забег третий: два mapreduce

- Добавим еще один mapreduce на соседней машине
- Погнали!
- Внезапно стало хуже...
- Но два mapreduce процесса на одной двухядерной машине почти не проиграли эталону
- Когда процессов больше чем ядер, существенного прироста не наблюдается

Забег четвертый: равные условия

- Пересылать десятки мегабайтов по сети дольше чем читать с диска
- Пусть данные будут на каждой машине!
- Входом для маппера будет имя файла, а содержимое пусть читает сам

Результат:

- один mapreduce почти не отстал от эталона
- вдвоем на разных машинах уже обогнали
- втроем на трех ядрах пришли в два раза быстрее

Локальность данных

- Хорошо когда данные и код живут рядом. На одной машине
- Копировать все данные неразумно
- Возможные решения
  - распределенные файловые системы

Распределенная ФС

- Файл хранится на нескольких разных машинах
- Диспетчер ФС хранит метаданные и знает какой файл где искать
- Если mapreduce процессы запускаются на DFS узлах то диспетчер mapreduce имеет возможность давать им локальные задания
- Известны проприетарные GFS (Google) и S3 (Amazon), открытые GlusterFS для Linux (на C) и Hadoop FS (Java)

Равноудаленность данных

- Весь интернет на одной машине не поместится все равно
- (No)SQL СУБД может хранить данные и быть входом для мапперов
  - да и выходом для reducer'ов тоже
- Известные NoSQL СУБД: MongoDB, Cassandra, HBase, DynamoDB... тыщи их

Движемся в облако

- Amazon EC2
  - Можно вручную запустить задачи на нескольких instance'ax
  - Можно воспользоваться фреймворком Elastic Map Reduce -- проще и дешевле
- Google App Engine
  - специальной поддержки map-reduce нет, механизма явного запуска нового instance'а
  - приложение автоматически масштабируется под нагрузку (запускаются новые instance'ы)
  - есть общее хранилище

GAE

- GAE Python + библиотека appengine-mapreduce
- Хранилищем данных выступает GAE datastore (bigtable) или blobstore
- Поддерживаются разные форматы входных и выходных данных
- Можно объединять несколько mapreduce'ов в конвейер

GAE: демо map-reduce

- Демо-приложение "map reduce made easy" позволяет относительно быстро начать
- Код выглядит не настолько easy как в mincemeat, но суть не меняется: функции map и reduce
- Входными данными может служить содержимое ZIP архива в blobstore. В библиотеке есть соответствующие реализации input reader'ов

GAE: тестирование

- Тестировать map-reduce можно на локальном сервере
- Можно развернуть боевой сервер и потестировать на небольшом объеме
- Помните о том, что бесплатная квота ресурсов на GAE довольно скромна

GAE: недостатки

- Не совсем контролируемая масштабируемость
- Отсутствие локальности данных

Amazon EC2 и Hadoop

- В EC2 можно практически сделать виртуальный датацентр, контролируя количество и характеристики запущенных instance'ов
- Можно развернуть Apache Hadoop -- промышленную реализацию Map-Reduce (место рождения -- Yahoo). В Hadoop входят map-reduce framework, распределенная ФС, NoSQL хранилище, высокоуровневые языки обработки данных