MapReduce

Что такое MapReduce?

- Парадигма распределенных вычислений над большими объемами данных
- Придумана и впервые реализована в Google
- Используется в интернет-компаниях, при обработке результатов экспериментов на коллайдерах и много где еще

Представление данных

- В рамках парадигмы MapReduce все данные хранятся в таблицах
- Таблица — мультимножество пар (ключ, значение)
- Ключи и значения могут иметь произвольную природу

Парадигма вычислений

Все вычисления представляются в виде суперпозиции двух простых операций: map и reduce.

map : (key, value) -> [(key, value)]

reduce: (key, [value]) -> [(key, value)]

Как (примерно) это работает?

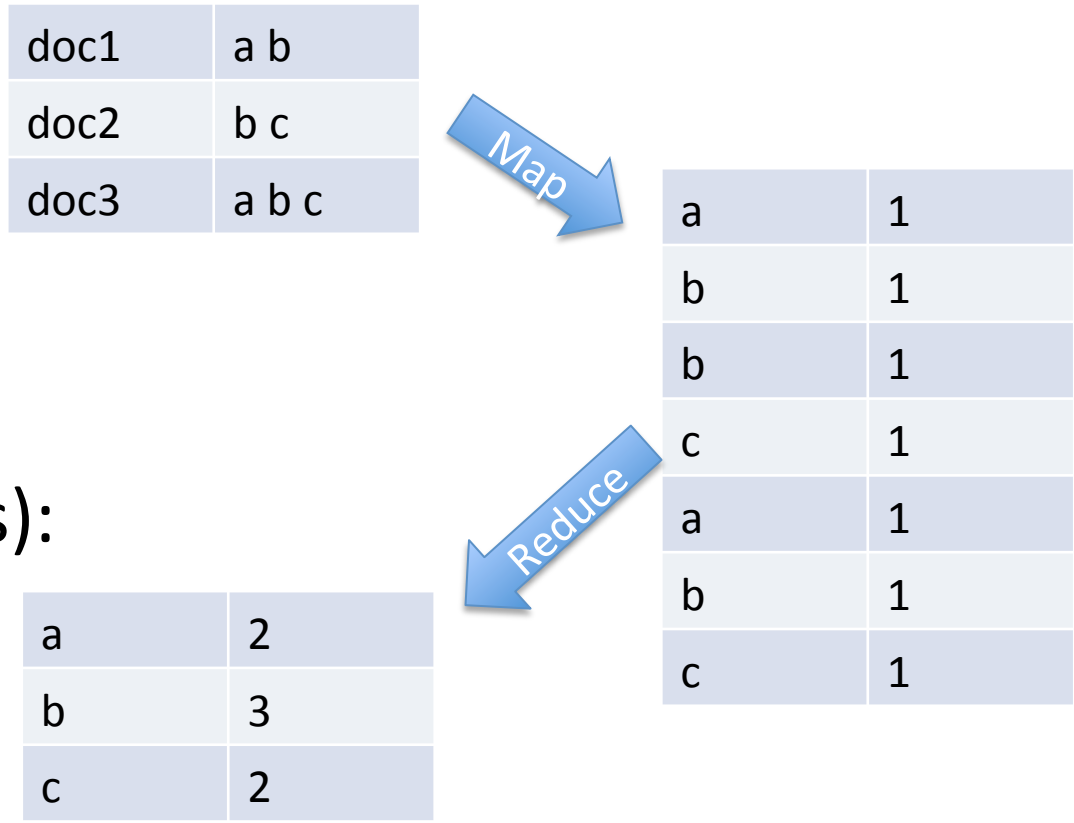
- Кусочки таблиц («чанки») лежат на различных хостах кластера
- map и reduce выполняются локально на каждом хосте, генерируя кусочки выходных таблиц
- Выходные данные операций остаются на том же хосте
- Перед reduce выполняется обмен данными между хостами, чтобы собрать на одном хосте записи с одним и тем же ключом

Простой пример: wordcount

Подсчитаем число вхождений каждого слова в большом корпусе текстов.

Map(docid, text):
    for word *in* text:
        emit(word, 1)

Reduce(word, counts):
    s := *sum*(counts)
    emit(word,s)



Combiner-ы в MapReduce

- Многие реализации MapReduce также поддерживают операцию combine
- Combine, как и reduce, получает на вход записи с одним и тем же ключом, но не обязательно все
- Запускается на каждом хосте перед reduce, чтобы уменьшить объем участвующих в сетевом обмене данных
- Иногда в качестве combine можно использовать reduce (например, в wordcount)

Обучение с учителем

- Хотим обучить модель по выборке:

$$L(w) = \frac{1}{N}\sum_i L(y_i, f_w(x_i)) \to \min_w$$

- Можно попробовать решать градиентным спуском:

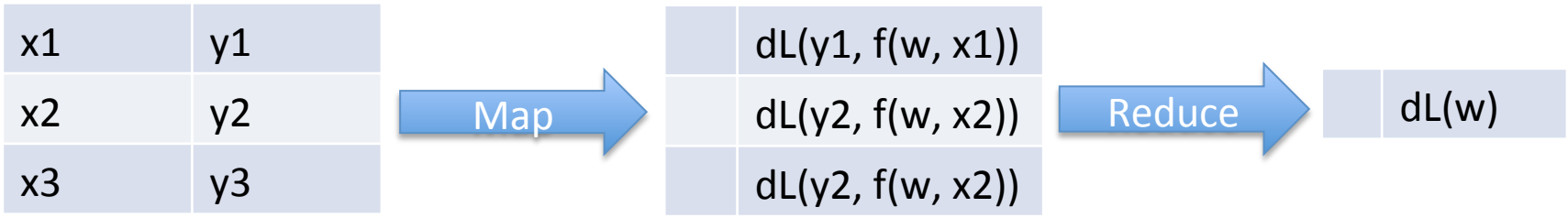
$$\nabla L(w) = \frac{1}{N}\sum_i \nabla L(y_i, f_w(x_i))$$

$$w \leftarrow w - \alpha \nabla L(w)$$

- N может быть очень большим
- Тогда каждый шаг спуска будет требовать вычисления и суммирования большого числа членов

Обучение с учителем на MapReduce

Каждый шаг градиентного спуска можно выполнить с помощью map и reduce:



Использование combine позволит распараллелить суммирование и минимизировать обмен данными

Модель PageRank

Рассмотрим следующую модель случайных блужданий по ориентированному графу:

- Начинаем из случайной вершины
- С вероятностью p из текущей вершины переходим в случайную
- С вероятностью 1-p из текущей вершины переходим в ее случайного соседа

PageRank

- Если вершины графа соответствуют веб-страницам, а ребра, — ссылкам, то такая модель приближенно описывает поведение пользователя в интернете
- PageRank-ом страницы называется вероятность оказаться в соответствующей вершине через бесконечное число времени

PageRank на MapReduce

- Будем последовательно вычислять вероятность оказаться в вершине в каждый момент времени до сходимости
- Для этого нужно знать вероятность оказаться в ее соседях в предыдущий момент времени
- Вычисления для очередного момента времени можно выполнить с помощью операций map и reduce

PageRank на MapReduce

Map(u, (u_destinations, rank)):

    emit(u, u_destinations)

    for v *in* u_destinations:

        emit(v, rank / *len*(u_destinations))

Reduce(v, (v_destinations, src_ranks)):

    rank := (1-p) * *sum*(src_ranks) + p/N

    emit(v, (v_destinations, rank))

Почему MapReduce?

- Суперкомпьютеры дорогие
- Их стоимость растет нелинейно с ростом мощности
- MapReduce может работать на кластере из «обычных» компьютеров
- Как следствие, легко масштабируется
- Отказоустойчивость достигается за счет репликации данных
- Очень простая абстракция, не нужен опыт параллельного программирования

Что почитать

- MapReduce: Simplified Data Processing on Large Clusters. *Jeffrey Dean and Sanjay Ghemawat, 2004*.
- Data-Intensive Text Processing with MapReduce. *Jimmy Lin and Chris Dyer, 2010.*
- MapReduce/Bigtable for Distributed Optimization. *Keith B. Hall, Scott Gilpin, Gideon Mann, 2010.*