



MapReduce

Ильнур Шугаев

Пример задачи

Поиск по логам

Хотим найти все
вхождения события в логи
в течении года

70 GB

Логов рекламы в день

25 TB

Логов за год

100 M/s

Скорость чтения

70

Часов

Map & Reduce¹

Большое количество програм можно выразить в виде последовательности функций map и reduce:

$$\begin{array}{lll} \text{map} & (k_1, v_1) & \rightarrow \text{list}(k_2, v_2) \\ \text{reduce} & (k_2, \text{list}(v_2)) & \rightarrow \text{list}(v_2) \end{array}$$

Данные функции можно легко распараллеливать

¹Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51:1 (2008), pp. 107–113.

Примеры

WordCount

Map Для каждого слова в документе выдает $\langle \text{word}, 1 \rangle$

Reduce Суммирует счетчики для каждого слова

Примеры

Distributed Grep

Map Ищет строки подходящие под паттерн

Reduce Id function

Примеры

Inverted Index

Map Обрабатывает документы и выдает последовательность вида $\langle \text{word}, \text{document ID} \rangle$

Reduce Для каждого слова выдает пару $\langle \text{word}, \text{list}(\text{document ID}) \rangle$

Примеры

Distributed Sort

Map Из каждой записи извлекается ключ и выдается пара $\langle \text{key}, \text{record} \rangle$

Reduce Id function

Корректность результата зависит от 1 (Partitioning function) и 2 (Ordering)

Цели MapReduce

- 1 Параллелизм
- 2 Отказоустойчивость
- 3 Распределение данных
- 4 Распределение ресурсов и планирование



Table of Contents

1. Введение

2. Реализация

- Параллелизм

- Отказоустойчивость

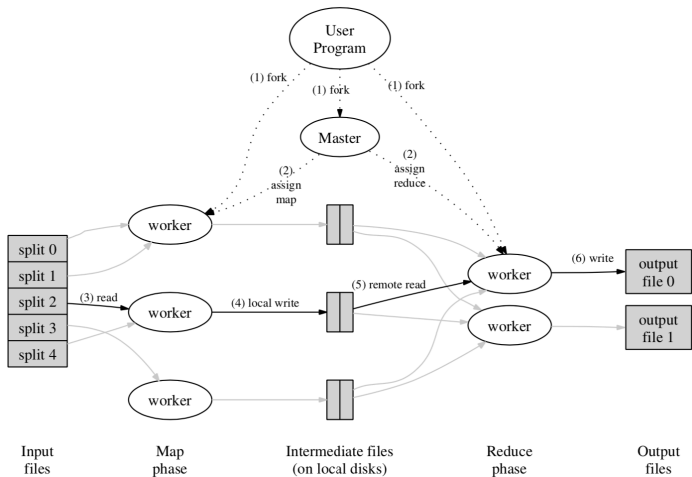
- Распределение данных

- Распределение ресурсов и планирование

3. Улучшения

4. Performance

Схема работы



Распределение задач

- M - количество map'ов, R - количество reduce'ов
- Partitioning Function example — $hash(key) \bmod R$

Падение воркеров

- При падении Map-воркера перезапускается map задача и все reduce задачи, которые зависят от данных упавшего воркера
- При падении Reduce-воркера перезапускается только reduce задача

Падение мастера

При падении мастера прерывается весь процесс MapReduce вычислений

Код к данным

Хотими минимизировать обмен данными по сети.

Решение:

- ✓ Мастер знает о том, где расположены данные, на которых нужно запустить тар задачи
- ✓ Если это возможно, то следует запускать тар задачу на машине, на которой находятся нужные данные
- ✓ В противном случае, мастер выбирает воркера, который ближе всего к данным (с точки зрения топологии сети)

Статусы воркеров

Мастер периодически спрашивает всех воркеров для уточнения статуса запущенных на них задач.

Задача может находиться в одном из следующих состояний:

- idle
- in-progress
- completed

Backup tasks

Straggler

Definition 1 (Straggler): *воркер, которому необходимо сильно больше времени для завершения его задач в сравнении с остальными воркерами*

Причины

- Медленное железо
- Данные слишком далеко
- ...

Backup tasks

Straggler. Решение проблемы

- Запуск дополнительных дублирующих задач, для задач, которые в состоянии in-progress, в ситуации, когда большинство задач уже в статусе completed
- Задача считается завершенной, когда она сама или дублирующая успешно завершаются

Table of Contents

1. Введение

2. Реализация

3. Улучшения

- Partitioning functions

- Ordering guarantees

- Combiner function

- Skipping bad records

4. Performance

Partitioning functions

- Функция $hash(key) \bmod R$ создает слишком большие ограничения
- Пользователь может определить функцию партицирования самостоятельно

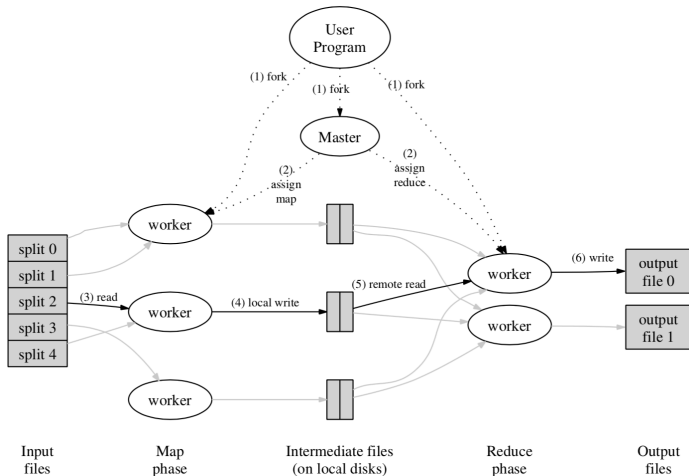
Ordering guarantees

1. В результате выполнения map задачи, на локальном диске воркера образуется R партиций данных
2. Гарантируется, что данные на вход reduce задаче будут передаваться в порядке неубывания ключей

Если `reduce = Id function`, то после завершения всех reduce задач, мы получаем частично упорядоченный файл из R партиций, где данные внутри каждой партиции отсортированы

Ordering guarantees

Общая схема



Ordering guarantees

Distributed Sort

Несколько вариантов

- Использовать только один reduce воркер ($R = 1$)
- Использовать свою Partitioning function, которая разбивает данные на партии по ключу, а не по хэшу от ключа

Combiner function

Проблема. Word count

- Частоты слов распределены по Zipf Law²
- Небольшое количество reduce задач получают большие списки на вход

¹https://en.wikipedia.org/wiki/Zipf%27s_law

Combiner function

Решение. Word count

Пользователь может задать Combiner функцию, которая на стороне map воркера будет делать предсчет

Skipping bad records

Проблема

Если в данных есть записи, к которым код map/reduce функций не готов, то весь процесс завершится с ошибкой

Skipping bad records

Решение

1. Можно попросить MapReduce запоминать записи, на который детерминировано появляются ошибки
2. Когда задача будет перезапущена она может успешно пропустить проблемные записи

Table of Contents

1. Введение

2. Реализация

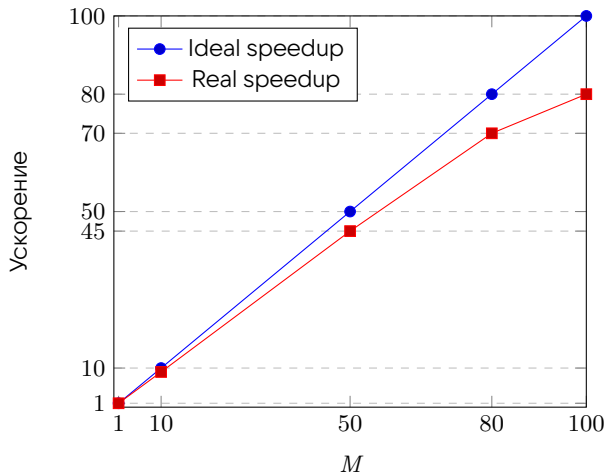
3. Улучшения






4. Performance

Пример

Поиск по логам

Ускорение в зависимости от числа Map воркеров



-  Chu, Cheng-Tao et al. "Map-reduce for machine learning on multicore". In: *Advances in neural information processing systems*. 2007, pp. 281–288.
-  Dean, Jeffrey and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pp. 107–113.
-  Lin, Jimmy. "Mapreduce is good enough? if all you have is a hammer, throw away everything that's not a nail!" In: *Big Data* 1.1 (2013), pp. 28–37.
-  Lin, Jimmy and Chris Dyer. "Data-intensive text processing with MapReduce". In: *Synthesis Lectures on Human Language Technologies* 3.1 (2010), pp. 1–177.
-  White, Tom. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.



Вопросы?

Ильнур Шугаев