# Паттерн «Map-Reduce»

Александр Смирнов

10.03.2020

# Введение

- Техника для обработки больших данных
- map
  - применяет функцию к каждому элементу последовательности и возвращает итератор с результатами
  - `list(map(str.upper, ['one', 'two', 'list', '', 'dict']))`
    `# ['ONE', 'TWO', 'LIST', '', 'DICT']`
  - `list(map(lambda x: x + 1, [0, 2, 5]))`
    `# [1, 3, 6]`
- reduce
  - последовательно применяет функцию к элементам списка, возвращает единичное значение
  - `reduce(lambda x, y: x + y, [1, 5, 9])`
    `# 15`
  - `reduce(lambda x, y: x if (x > y) else y, [1, 5, 9])`
    `# 9`

# Пример

- Вернуть самую длинную строку из списка строк
- На больших данных работает медленно
- Вертикальное масшабирование
  - Внедрение более качественного и быстрого оборудования
  - Увеличение размера данных
- Горизонтальное масшабирование
  - Разработаем код так, чтобы он мог работать параллельно
  - Станет быстрее, когда добавим процессоров

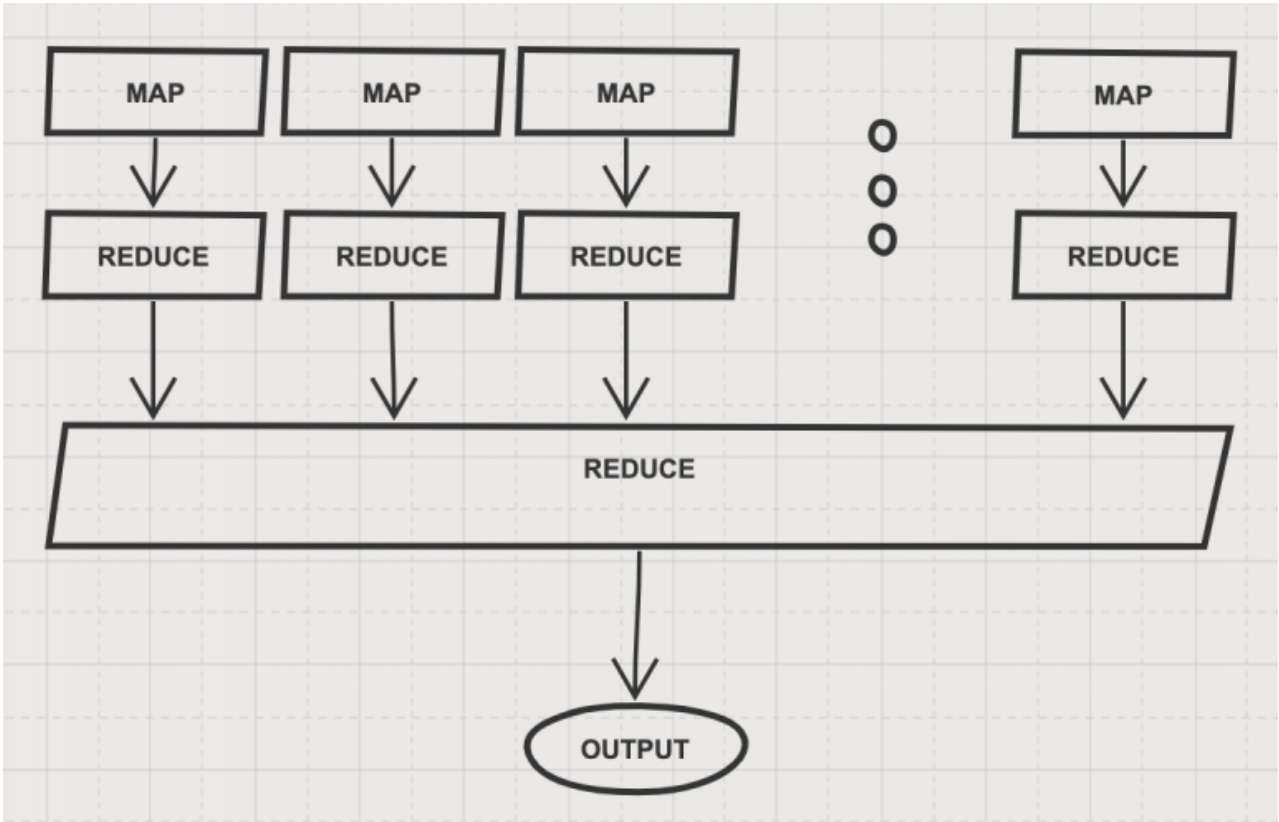
# Решение (1)

- Разбить код на множество блоков
- Выполнить функцию поиска самой длинной строки для каждого блока параллельно
- Найти самую длинную строку среди выходных данных всех блоков

# Решение (2)

- Разбили код на два этапа
  - Вычисляем длину всех строк
  - Выбираем максимальное значение
- Дадим шагам имена
  - mapper
  - reducer
- Разобъём входные данные на куски
- Распараллелим

# Архитектура




- Каждый блок обрабатывает входные данные и reduce-ит их
- Результаты reduce-ов reduce-ятся

# Особенности

- Масштабируемость
  - Если больше данных, то просто добавляем больше процессоров без изменения кода
- Универсальность
  - Поддерживается широкий спект задач, просто меняем функционал наших map и reduce
- Большие данные
  - Разбиение на фрагметы неэффективно, поэтому будем хранить данные в виде кусков изначально
  - Hadoop Distributed File System