Курсовой проект
Фреймворк и файловая система для распределённой обработки больших данных в рамках концепции map-reduce

Выборнов А.И.

МГТУ им. Н. Э. Баумана

art-vybor@ya.com

12 декабря 2014 г.

Обзор

Постановка задачи

Концепция map-reduce

Архитектура

Отчёт

Постановка задачи

- Анализ требований и проектирование архитектуры системы. Реализация нераспределённого map-reduce. Решение проблемы RPC.
- Реализация распределённой файловой системы. Реализация фреймворка. Тестирование на примерах.

Зачем нужен map-reduce?

- Обработка больших данных (Big Data).
  - Вычисления превосходят возможности одной машины.
  - Данные не помещаются в памяти, необходимо обращаться к диску.
  - Можно хранить много данных, но задержки и пропускная способность оборудования растут пропорционально данным.
- Удобная абстракция для построения алгоритмов обработки больших данных.
- Устойчивость к отказам.

Что такое map-reduce?

- Структура (key, value) - пара (ключ, значение).
- Программирование представляет собой определение двух функций:
  - $map : (key, value) \rightarrow [(key, value)]$
  - $reduce : (key, [value]) \rightarrow [(key, value)]$
- Между стадиями *map* и *reduce* происходит группировка и сортировка данных.

Map-reduce на примере - Поиск общих друзей

- **Задача:** Есть граф пользователей некоторого ресурса, заданный в виде строчек: «пользователь - друг1 друг2 ...». Для каждой пары пользователей найти общих друзей.
- Разберём задачу на следующих входных данных:
  - A - B C D
  - B - A C
  - C - A B D
  - D - A C

Map-reduce на примере - Поиск общих друзей

- На стадии map преобразовываем пару (пользователь, друзья) в множество пар следующим образом:
  - (A, B C D) -> (A B, B C D), (A C, B C D), (A D, B C D)
  - (B, A C) -> (A B, A C), (B C, A C)
  - (C, A B D) -> (A C, A B D), (B C, A B D), (C D, A B D)
  - (D, A C) -> (A D, A C), (C D, A C)

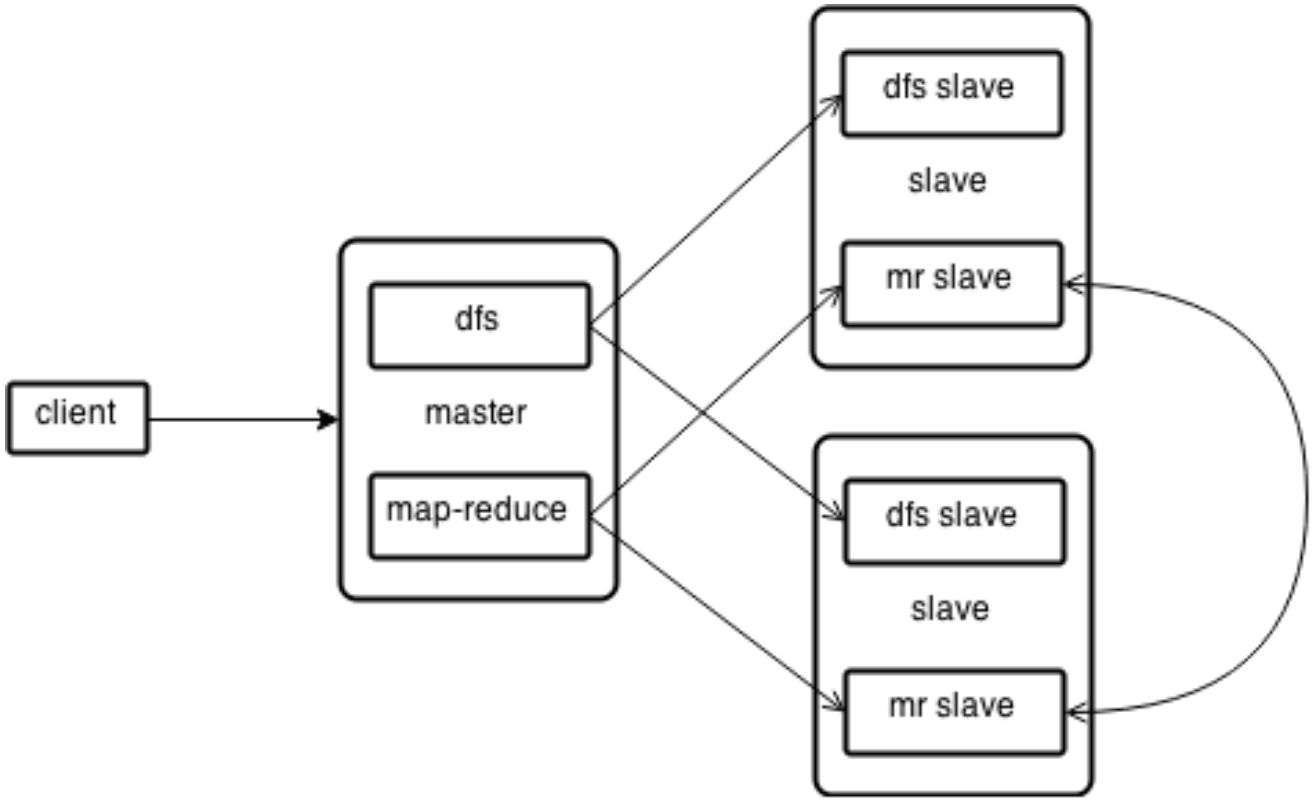
Map-reduce на примере - Поиск общих друзей

- Сливаем результаты полученные на стадии map, получаем список пар:
  - (A B, [B C D, A C])
  - (A C, [B C D, A B D])
  - (A D, [B C D, A C])
  - (B C, [A B D, A C])
  - (C D, [A B D, A C])

Map-reduce на примере - Поиск общих друзей

- На стадии reduce пересекаем с друг другом все элементы списка значений и получаем:
  - (A B, C)
  - (A C, B D)
  - (A D, C)
  - (B C, A)
  - (C D, A)

Архитектура распределённого map-reduce



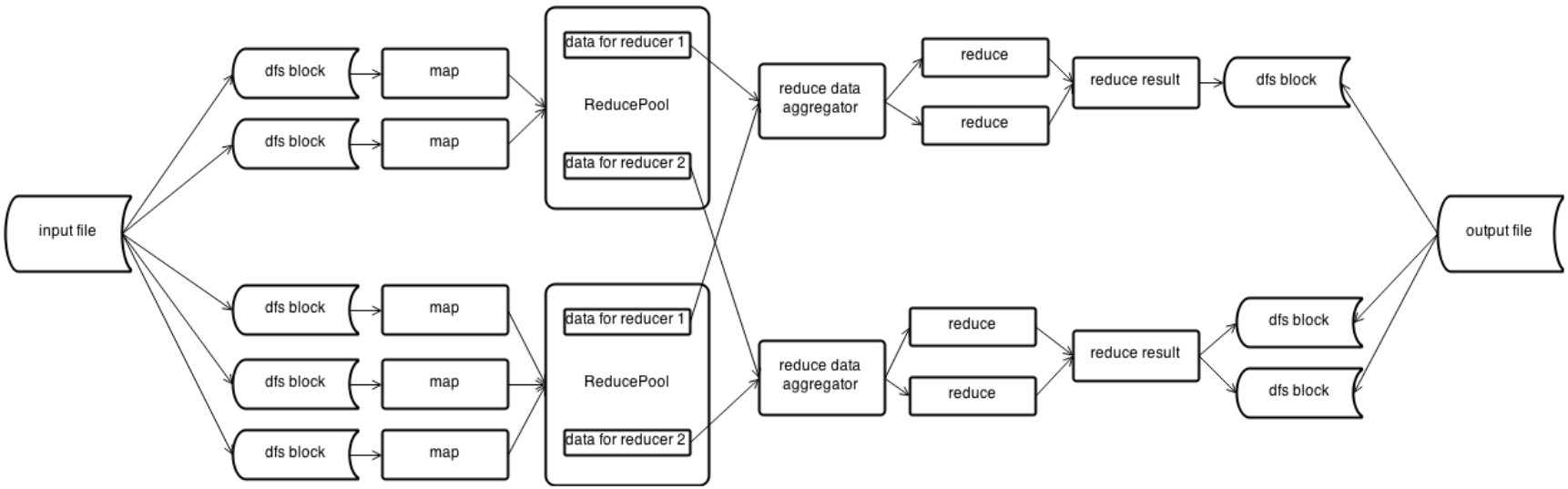
Как работает распределённая файловая система

- Файл разбивается на блоки фиксированного размера (64мб).
- Получившиеся блоки распределяются по машинам.
- Структура фс - дерево, в котором узлы соответствуют папкам, а листья - файлам.
- Файл представляет собой список индексов.
- BlockManager - класс, раздающий новых индексы и содержащий таблицу отображения блоков в конкретную ноду.

Как работает распределённый map-reduce

- На вход подаются адреса файлов в DFS для ввода и вывода, а также файл с функциями map и reduce.
- Стадии выполнения:
  - Получение по входному файлу списка индексов в DFS.
  - Разбиение списка индексов по узлам.
  - Выполнение стадии map на узлах.
  - Распределение данных между узлами.
  - Выполнение стадии reduce на узлах.
  - Добавление результатов функций reduce в виде файла в DFS.
    - Переименование результатов.
    - Запись информации о новом файле в DFS.

Архитектура распределённого map-reduce



Как работает map на узлах

- По списку индексов получаются данные путём обращения к узлу dfs, находящемуся на этой машине.
- Для каждого индекса выполняется переданная функция map.
- Результаты каждого map в виде ассоциативного массива передаются структуре reduce pool.
- По окончании оставшиеся в reduce pool данные отправляются по узлам.

Как работает reduce на узлах

- Сливает результаты стадий map и распределяет их по узлам.
- Структура reduce pool состоит из набора ассоциативных массивов, каждый из которых соответствует одному slave узлу.
- Данные переданные в структуру reduce pool равномерно распределяются среди набора массивов.
- После добавления новых данных структура проверяет, не превышает ли один элементов набора установленного размера. При превышении установленного размера элемент отправляется на соответсвующий узел.
- На узлах, все полученные от reduce pool данные аккумулируются.

Как работает reduce на узлах

- Разбивает все данные на блоки пар, по возможности, не превышающие фиксированного размера с точность до одной пары.
- Для каждой полученный пары выполняется переданная функция reduce.
- Полученный результат разбивается на блоки равного размера (64мб).
- Блоки записываются в dfs на узел, где выполнялся reduce под псевдоименами.

Что сделано

- Разработаны архитектуры распределённых map-reduce и файловой системы.
- Реализован распределённый map-reduce.
- Для реализации RPC была выбрана связка: ZeroMQ + Marshal.
- Реализована распределённая файловая система для хранения больших данных.

Что не реализовано

- Существует узкое место при аггрегации данных перед стадией reduce (reduce pool) - возможно переполнение оперативной памяти. Исправить можно следующим образом:
  - При получении данных, они добавляются в ассоциативный массив.
  - При превышении массивом установленного размера он сортируется и выгружается на диск.
  - Перед выполнением стадии reduce необходимо отсортировать и выгрузить на диск оставшиеся в массиве данные.
  - Во время выполнения стадии reduce полученные файлы на лету сливаются и обрабатываются функцией reduce.

Что предстоит сделать

- Подготовить код распределённого map-reduce для тестов.
- Протестировать полученную систему на больших данных и проанализировать результаты.

Курсовой проект
Фреймворк и файловая система для распределённой обработки больших данных в рамках концепции map-reduce

Выборнов А.И.

МГТУ им. Н. Э. Баумана

art-vybor@ya.com

12 декабря 2014 г.