> 45% of all Hadoop tutorials count words. 25% count sentences. 20% are about paragraphs. 10% are log parsers. The remainder are helpful.
>
> @jandersen

1 Задача

В индустрии часто возникают задачи, связанные с анализом данных, представленных в виде графа. Одна из таких задач – это поиск собственного вектора, соответствующего наибольшему собственному числу матрицы смежности графа A. Она лежит в основе алгоритмов ссылочного ранжирования страниц Интернет, алгоритма вычисления индекса цитируемости научных работ, ранжирования пользователей социальных сетей и т.п.

Задача решается численно с помощью метода степенных итераций или его разновидностей (степенной метод со сдвигом, с отсечением и др). На первом шаге метода выбирается начальное приближение – вектор $\vec{x_0}$. На практике можно просто взять единичный вектор.

На каждом k-м шаге алгоритм вычисляет $\vec{x_{k+1}} = \frac{A\vec{x_k}}{||A\vec{x_k}||}$. Можно показать, что метод сходится со скоростью $|\frac{\lambda_2}{\lambda_1}|$, где λ_1 – наибольшее собственное число, а λ_2 – следующее за ним.

В качестве критерия остановки алгоритма можно использовать, например, следующий: $|\vec{x_{k+1}} - \vec{x_k}| < \epsilon$, где ϵ – заданная точность.

1.1 Параллелизация

Для того, чтобы реализовать метод степенных итераций на больших объёмах данных, необходима параллелизация вычислений. Технология MapReduce и её открытая реализация Hadoop являются одним из наиболее удобных инструментов.

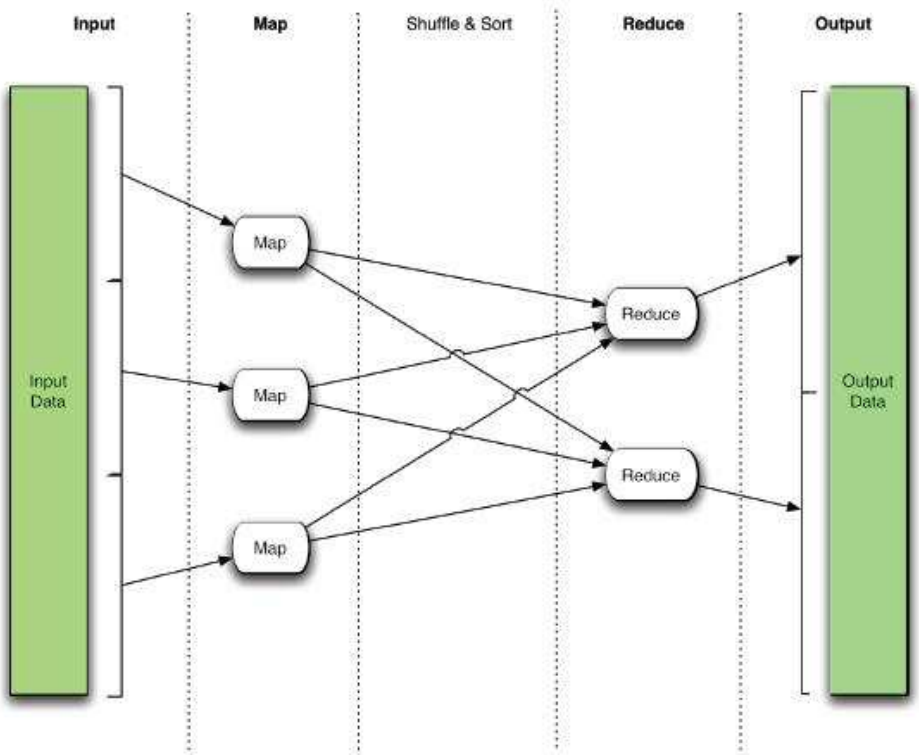
Задача программиста Hadoop – представить алгоритм в виде пары функций:

- функцию *mapper*, которая преобразовывает входные пары ключ-значение в пары промежуточных результатов в том же формате:

```
mapper (k1, v2) -> list<(k2, v2)>
```

- функцию *reducer*, которая получает пары из ключей и списков промежуточных значений, соотвествующих этому ключу. Результатом функции должен быть список значений:

```
reducer (k2, list<v2>) -> list<v3>
```



Этапы Hadoop

Обработка данных на Hadoop состоит из последовательных этапов:

1. Input – чтение исходных данных с распределённой файловой системы HDFS. На этом этапе пользователь может указать, как интерпретировать входную информацию. Hadoop поддерживает несколько стандартных форматов и позволяет определять собственные. Примеры форматов:
   (a) $TextInputFormat$ – данные в виде текстового файла. Ключ – номер строки в файле, значение – сама строка.
   (b) $KeyValueTextInputFormat$ – данные в виде текстового файла. Ключ – часть строки до символа-разделителя (обычно табуляция), значение – остальная часть строки.

2. Map – обработка входных данных с помощью функции *mapper*.

3. Shuffle & sort – промежуточный этап между Map и Reduce. Данные перераспределяются между узлами кластера и группируются по значению промежуточного ключа.

4. Reduce – обработка промежуточных данных с помощью функции *reducer*.

5. Output – запись результатов на HDFS. Как и на этапе Input, пользователь может задавать формат сохранения результатов.

"Родной" язык Hadoop – это Java. Однако система позволяет запускать алгоритмы, реализованные на любых языках, с помощью интерфейса Streaming[1].

Идея Streaming заключается в том, что Hadoop использует стандартный ввод/вывод (stdin/stdout) для обмена данными с пользовательскими программами.

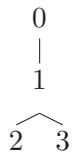
2 Задание

1. Используя шаблоны *mapper.fs* и *reducer.fs*, напишите решение для задачи подсчёта слов. В качестве тестового примера возьмите файл *lorem.txt*

   Для отладки решения можно использовать командную строку:

   ```
   cat lorem.txt | mono mapper.exe | sort | mono reducer.exe
   ```

2. Используя шаблоны, напишите алгоритм вычисления степенной итерации для поиска собственного вектора матрицы. В качестве входных данных возьмите файл *ba_100_10.txt*. Файл описывает случайный граф из ста узлов, сгенерированный по модели Барабаси–Альберта в следующем формате:


   ```
   0    1
   1    0 2 3
   2    1
   3    1
   ```

   (a) Проверьте вычисление одной итерации на локальной машине.

   (b) Запустите вычисление на тестовом кластере Hadoop, взяв в качестве входных данных файл *ba_40k_120.txt* – случайный граф на 40000 узлов.

[1] http://hadoop.apache.org/common/docs/current/streaming.html

3. Напишите решение для задачи поиска десяти наиболее “важных” вершин в графе, используя в качестве меры важности “eigenvector centrality”.

   Степенные итерации придётся вычислять последовательно, но каждую из итераций можно распараллелить. Для лабораторной достаточно вычислить фиксированное число итераций. На практике обычно используют связывание задач средствами Hadoop[2].

   (a) Проверьте вычисление одной итерации на локальной машине, взяв в качестве входных данных файл $ws\_100\_10.txt$ – случайный граф из ста узлов по модели Воттса-Строгатца.

   (b) Запустите вычисление на тестовом кластере на графе из 10000 узлов из файла $ws\_10k\_10.txt$.

[2] http://developer.yahoo.com/hadoop/tutorial/module4.html#chaining