Big Data’13

Лекция IV: PageRank и распределенные вычисления на графах

Дмитрий Барашев
bigdata@barashev.net

Computer Science Center

14 марта 2013

Сегодня в программе

PageRank

PageRank и MapReduce

Pregel

Разминка

Упорядочите слова *грека, рак, река, видит* из следующего корпуса документов в порядке убывания их IDF:

1. ехал грека через река
2. видит грека в река рак
3. сунул грека рука в река
4. рак за рука грека цап

Сегодня в программе

PageRank

PageRank и MapReduce

Pregel

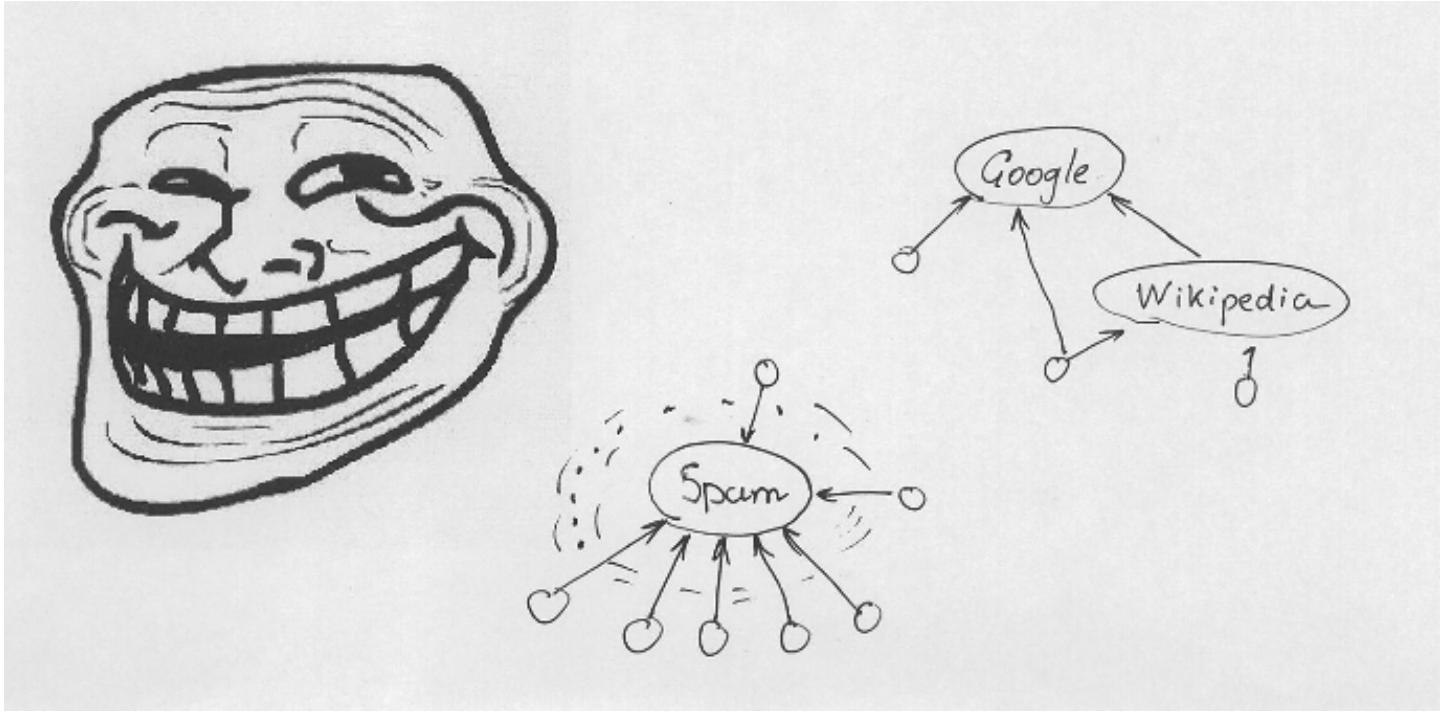
«Важность» документа

- Не все документы одинаково полезны
- Главная страница CNN или Wikipedia имеют большой вес
- А безвестный блог в ЖЖ без читателей – чуток поменьше
- Важность документа/сайта определяется многими сигналами
  - поведением пользователя на странице результатов
  - статистикой, собираемой службой аналитики
  - ссылками на документ

Подсчет ссылок

- На любой документ можно сослаться, но на важные будет больше ссылок
- 100500 леммингов не могут ошибаться!
- Больше ссылок – выше ранг
- Это уже давно практикуется в научных публикациях

Но SEOшник не дремлет



Случайные блуждания

- Мартышка с мышкой в руках случайно щелкает по ссылкам на странице
- Если на странице ссылок нет, браузер сам телепортирует мартышку на случайную страницу
- Даже если ссылки есть, с некоторой вероятностью происходит телепорт

Случайные блуждания

- Мартышка с мышкой в руках случайно щелкает по ссылкам на странице
- Если на странице ссылок нет, браузер сам телепортирует мартышку на случайную страницу
- Даже если ссылки есть, с некоторой вероятностью происходит телепорт

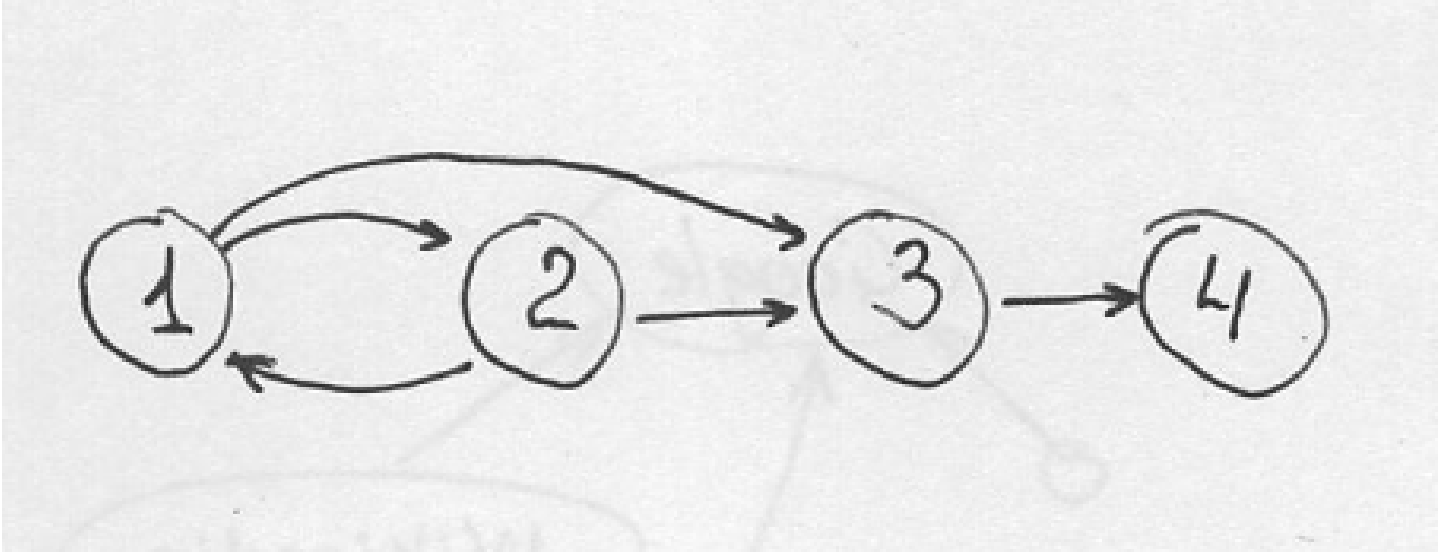
PageRank страницы – вероятность того что мартышка на ней окажется

Ряд допущений

- N страниц
- Вероятность перейти по ссылке равна $(1-\alpha)\frac{1}{|outlinks|}$
- Вероятность совершить телепорт на любую страницу равна α/N
- Для тупиковых страниц $\alpha = 1$
- Для остальных обычно $\alpha = 0.15$

Как же считать

- PageRank страницы зависит от PageRank ссылающихся страниц



Математическая модель

- Рассмотрим матрицу

$$P = (1-\alpha)A + \frac{\alpha}{N}J + \frac{(1-\alpha)}{N}d^T e$$

- Первое слагаемое: переход по ссылкам. A - матрица вероятностей переходов по ссылкам. $A_{ij} = \frac{1}{|outlinks_i|}$ если есть ссылка со страницы i на j.

$$0.85 \times \begin{bmatrix} & p_1 & p_2 & p_3 & p_4 \\ p_1 & 0 & 0.5 & 0.5 & 0 \\ p_2 & 0.5 & 0 & 0.5 & 0 \\ p_3 & 0 & 0 & 0 & 1 \\ p_4 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Математическая модель

- Рассмотрим матрицу

$$P = (1-\alpha)A + \frac{\alpha}{N}J + \frac{(1-\alpha)}{N}d^T e$$

- Второе слагаемое: телепортация из любой страницы. J - матрица единиц

$$0.15 \times \begin{bmatrix} & p_1 & p_2 & p_3 & p_4 \\ p_1 & 1 & 1 & 1 & 1 \\ p_2 & 1 & 1 & 1 & 1 \\ p_3 & 1 & 1 & 1 & 1 \\ p_4 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Математическая модель

- Рассмотрим матрицу

$$P = (1-\alpha)A + \frac{\alpha}{N}J + \frac{(1-\alpha)}{N}d^T e$$

- Третье слагаемое: телепортация из тупиковой страницы. d^T - вектор-столбец тупиковых страниц, e - вектор единиц

$$0.85 \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} = 0.85 \times \begin{bmatrix} & p_1 & p_2 & p_3 & p_4 \\ p_1 & 0 & 0 & 0 & 0 \\ p_2 & 0 & 0 & 0 & 0 \\ p_3 & 0 & 0 & 0 & 0 \\ p_4 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Математическая модель

- Сумма значений в каждой строке P равна 1
- P – стохастическая матрица, представляющая матрицу переходных вероятностей цепи Маркова
- Если вектор $\overrightarrow{x_0}$ - начальное распределение вероятностей пребывания на той или иной странице то $\overrightarrow{x_1} = \overrightarrow{x_0} * P$ распределение после 1-го шага, $\overrightarrow{x_2} = \overrightarrow{x_0} * P^2$ – после 2-го, $\overrightarrow{x_k} = \overrightarrow{x_0} * P^k$ – после k-го
- Известно что существует равновесное распределение $\overrightarrow{\pi}$ являющееся собственным вектором матрицы P: $\overrightarrow{\pi}P = \lambda\overrightarrow{\pi}$
- Собственный вектор можно вычислить итеративно

Сегодня в программе

PageRank

PageRank и MapReduce

Pregel

Подсчет значения PageRank

- На каждой итерации считается новый PR страницы j как сумма подарков от всех остальных страниц
- Подарки персональные и общие

Персональные подарки

- Если есть ссылка $i \to j$ то подарок от i равен

$$PR_i \times (1 - \alpha) \times A_{ij}$$

Общие подарки

- От каждой страницы i подарок в размере

$$PR_i \times \frac{\alpha}{N}$$

$$\Sigma_i PR_i \times \frac{\alpha}{N} = \frac{\alpha}{N}$$

- От каждой тупиковой страницы i подарок в размере

$$PR_i \times \frac{(1-\alpha)}{N}$$

Алгоритм вычисления

- Посчитать сумму $\Sigma_i PR_i$ для тупиковых страниц
- Каждая страница посылает персональные подарки адресатам
- Страницы суммируют персональные подарки и общие

Реализация на Map-Reduce

```
1  def mapfn(k, v):
2      docid, rank, outlinks = v.split()[:3];
3      yield docid, 0
4      if outlinks != "==":
5          dst_docids = outlinks.split(',')
6          for d in dst_docids:
7              yield d, float(rank) / len(dst_docids)
8      else:
9          for d in range(1, 5):
10             yield str(d), float(rank)/4
11
12 def reducefn(k, vs):
13     return 0.85*sum(vs) + 0.15/4
```

Map-Reduce прекрасен, но

- каждая итерация делает одно и то же
- а процессы надо запускать
- за локальностью данных следить тяжело

Существующие решения

- Использовать библиотеки для одной машины
  - не масштабируется

Существующие решения

- Использовать библиотеки для одной машины
  - не масштабируется
- Использовать библиотеку для параллельной обработки
  - их еще надо поискать

Существующие решения

- Использовать библиотеки для одной машины
  - не масштабируется
- Использовать библиотеку для параллельной обработки
  - их еще надо поискать
- Написать свою инфраструктуру для своей задачи
  - лень

Существующие решения

- Использовать библиотеки для одной машины
  - не масштабируется
- Использовать библиотеку для параллельной обработки
  - их еще надо поискать
- Написать свою инфраструктуру для своей задачи
  - лень
- Как насчет SQL?

```
1 SELECT dst_docid, 0.85 * SUM(p.rank / p.outlinks) +
      0.15/(SELECT COUNT(*) FROM Pages) as rank
2 FROM Pages p JOIN Links l ON (l.src_docid = p.id)
3 GROUP BY l.dst_docid
```

  - дороговато может выйти

хочется Map-Reduce заточенный на графовые алгоритмы

Сегодня в программе

PageRank

PageRank и MapReduce

Pregel

Pregel – это река

Инфраструктура для алгоритмов на больших графах

- Идея та же: программист пишет код, который что-то вычисляет в вершине и посылает сообщения другим вершинам
- Инфраструктура заботится о координации, масштабировании, пересылке сообщений, восстановлении после сбоев
- Процесс итеративный
- Надо обработать быстрее или больше данных – добавляем машин

Действующие лица

- Структуры, хранящие данные о вершинах и исходящих дугах. Лежат в хранилище (GFS, СУБД)
- Бинарник с пользовательским кодом, выполняющийся на машинах в кластере
- Мастер – один из бинарников, координирующий действия остальных

Состояния вершины

- Активное: работа еще не закончена или есть входящие сообщения
- Пассивное: входящих сообщений нет, пользовательский код считает что работа закончена
- Пользовательское значение: какая-то изменяемая структура

Коммуникация между вершинами

- Вершины могут посылать и принимать сообщения
- Сообщение – это какая-то пользовательская структура или запрос изменения топологии

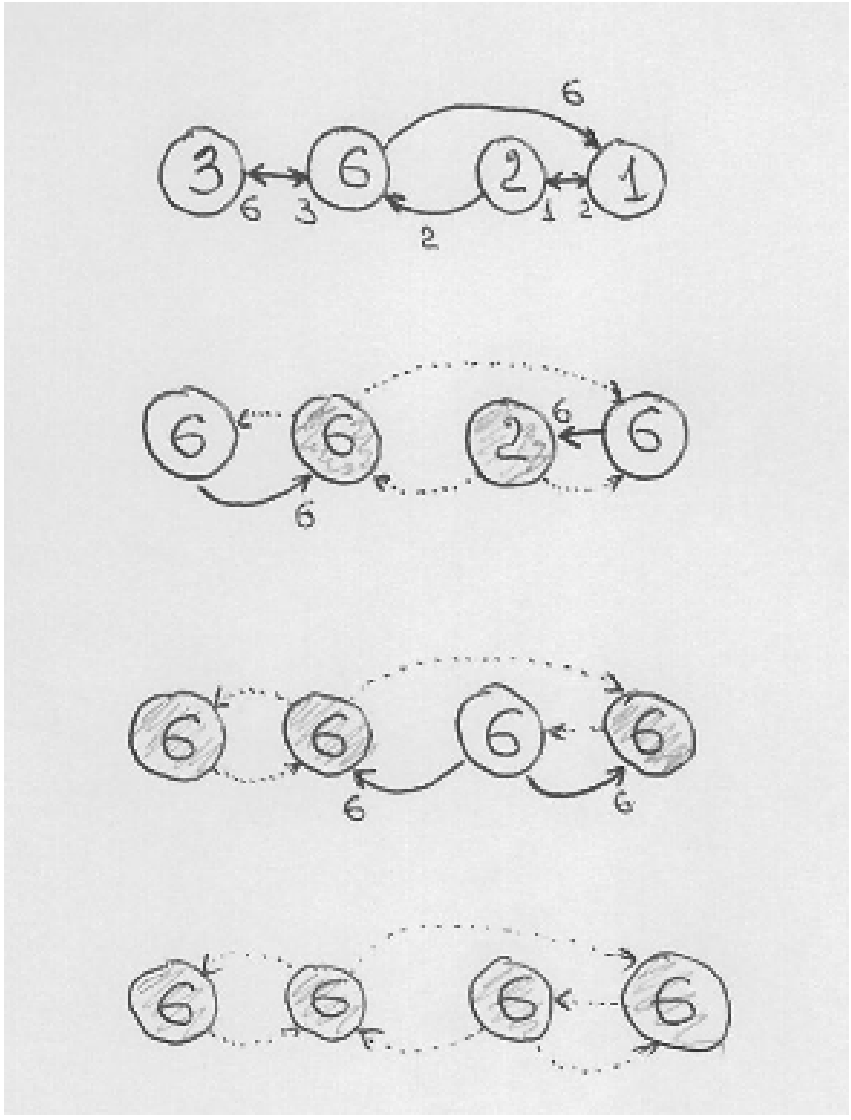
Пошаговая стратегия

1. Граф делится на фрагменты
2. Бинарники стартуют и читают свои фрагменты (каждому достается много)
3. Мастер приказывает выполнить итерацию (*superstep*).
4. Рабочие процессы сообщают мастеру о результатах: сколько вершин будет активно на следующей итерации
5. Мастер может приказать рабочим сохранить их текущее состояние и входящие сообщения (checkpoint)
6. Если все вершины остановлены то работа завершена
7. `GOTO 3`

Во время итерации

1. Выполняет функцию `compute()` для каждой вершины из фрагмента и предназначенных ей сообщений из предыдущей итерации
2. Посылает и принимает сообщения текущей итерации
3. Изменяет значение в вершине и исходящих дугах
4. «Деактивирует» вершины, где работа закончена

Вычисление максимального значения



Дополнительные возможности

- Комбайнеры: если сообщения ассоциативные и коммутативные то перед отсылкой можно произвести локальную редукцию
- Агрегаторы
  - вершина предоставляет датчик
  - рабочий процесс аггрегирует значения датчиков по своим вершинам
  - мастер аггрегирует значения по рабочим процессам и предоставляет результат всем вершинам

Восстановление после сбоев

- Мастер пингует рабочие процессы
- Если процесс помирает, его фрагменты передаются другим
- Все читают состояние, сохраненное из последней контрольной точки
- Несколько последних итераций, возможно, будут повторены

Вычисление PageRank в Pregel

```
1  class PageRankVertex(Vertex):
2    def compute(self):
3      if self.superstep < 50:
4        self.value = 0.15 / num_vertices + 0.85*sum(
5          [pagerank for (vertex,pagerank) in self.
               incoming_messages])
6        outgoing_pagerank = self.value / len(self.
             out_vertices)
7        self.outgoing_messages =
8          [(vertex,outgoing_pagerank) for vertex in self.
               out_vertices]
9        else:
10         self.active = False
```

Другие задачи

- Кратчайшие пути
- Остовное дерево
- Кластеризации
- roll your own...

Дерево кратчайших путей

```
1  class SSSPVertex(Vertex):
2    def update(self):
3      mindist = 0 if self.is_source() else float(”inf”)
4      for (vertex,dist) in self.incoming_messages:
5        if mindist > dist:
6          mindist = dist
7        if mindist < self.value:
8          self.value = mindist
9          self.outgoing_messages = [(vertex, mindist + 1)
10           for vertex in self.out_vertices]
11       else:
12         self.active = True if mindist == float(”inf”)
               else False
```

Занавес

- PageRank – значимость вершины графа в зависимости от ее расположения в топологии
- Вероятность находиться в вершине при случайных блужданиях
- Может вычисляться последовательностью map-reduce процессов
- Существуют более эффективные реализации графовых алгоритмов (Pregel, Giraph, Hama)

Эта презентация сверстана в

PAPEERIA

LaTeX в вашем браузере

alpha.papeeria.com

Литература I

- Jonathan Cohen.
  Graph twiddling in a mapreduce world.
  Computing in Science & Engineering, 11(4):29–41, 2009.
- Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski.
  Pregel: a system for large-scale graph processing.
  In *Proceedings of the 2010 international conference on Management of data*, pages 135–146. ACM, 2010.
- Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd.
  The pagerank citation ranking: bringing order to the web.
  1999.