PART 1. LARGE SCALE DATA ANALYTICS
WEB-SCALE LINK ANALYSIS
Sangmi Lee Pallickara
Computer Science, Colorado State University
http://www.cs.colostate.edu/~cs435

FAQs
• Matrix vs. Vector multiplication
  • Please see the updated slides
  • Marked with:
    • Version 1: fundamental Matrix – vector multiplication
    • Version 2: handling large vector
    • Version 3: handling large vector v and column size in M

Quiz 4
• Sorting large amount of numbers

PageRank
Map function
• The Map function is written to apply to one element of M
• Each Map task will operate on a chunk of the matrix M
• From each matrix element \( m_{ij} \), it produces the key-value pair \((i, m_{ij}v_j)\)
• All terms of the sum that make up the component \( x_i \) of the matrix-vector product will get the same key, i

Reduce function
• Sums all the values associated with a given key i
• The result will be a pair \((i, x_i)\)

VERSION 1: FUNDAMENTAL MATRIX vs. VECTOR MULTIPLICATION

Matrix \( M \)
Initial Vector \( v \)

Division of a matrix and vector into five stripes
The \( i \)th stripe of the matrix multiplies only components from the \( i \)th stripe of the initial vector

Results:
\[ 0.002 \times \frac{1}{n} + 0.017 \times \frac{1}{n} + 0.003 \times \frac{1}{n} + 0.010 \times \frac{1}{n} \ldots \]

\[ \text{+ (M} \ 00 \text{x v} \ 0) \text{+ (M} \ 01 \text{x v} \ 1) \text{+ (M} \ 02 \text{x v} \ 2) \text{...} \]

VERSION 2: WITH VERY LARGE \( v \)

Matrix \( M \)

\( n \) splits of \( v \)

\((n \times l)\) blocks of \( M \)

\( l \) blocks

\( k \) stripes

\( k \) splits

Reducer:
Add all of the local sums of the row \( k \), and store it in the \( k \)th element of \( v \)

Page 0: \( \frac{1}{n} \)
Page 1: \( \frac{1}{n} \)
Page 2: \( \frac{1}{n} \)
Page 3: \( \frac{1}{n} \)

Page: 0    1           2         3
======================
0.002   0.017   0.003  0.010
0.000   0.000   0.003  0.000
0.002   0.000   0.003  0.000
0.002   0.017   0.000  0.000

..           ..          …      …

VERSION 3: WITH VERY LARGE \( v \) and COLUMN in M

1. Analysis phase (first MapReduce)

Major functionality of mapper: performing a simple random sampling
Input: each record
Output: if the record is selected: \(<\text{amountOfTransaction}, \text{null}>\>
If the record is not selected: no output

Major functionality of reducer: there will be a single reducer
No particular functionality. Return the keys only
Input \(<\text{amountOfTransaction}, \text{null}>\>
Output \(<\text{amountOfTransaction}, \text{null}>\)
1.5 Separate task (non—parallel)

- Generate .seq file to specify ranges for partitioning using output from the "Analysis phase" job.

2. Sorting phase (second MapReduce)

Uses TotalOrderPartitioner (or write your own customized partitioner) (Set this in driver)
- This will be an identity Mapper
  - Input: each record, Output: <amountOfTransaction, same as the input>

Major functionality of reducer: writing the sorted values
- No particular functionality.
  - Input <amountOfTransaction, [a list of records]>
  - Output <amountOfTransaction, [a list of records]>

Topics

- Large-scale Analytics 1. Web-Scale Link and Social Network Analysis

Part 1. Large Scale Data Analytics

1. Web-Scale Link Analysis and Social Network Analysis

Challenges in PageRank Algorithm

for the real Web: Examples

Example 1

- Compute the PageRank of each page assuming $\beta=0.8$

\[
\beta M (1-\beta e) = \begin{pmatrix}
\frac{1}{3} & \frac{1}{2} & \frac{1}{3} \\
\frac{1}{3} & 0 & \frac{1}{2} \\
\frac{1}{3} & \frac{1}{2} & 0
\end{pmatrix} \\
\times \begin{pmatrix}
\frac{1}{3} \\
\frac{1}{3} \\
\frac{1}{3}
\end{pmatrix} \\
\times \begin{pmatrix}
\frac{1}{3} \\
\frac{1}{3} \\
\frac{1}{3}
\end{pmatrix} \\
+ \left(1-0.8\times \frac{1}{3}\right)
\]

Example 2

- **clique**
  - Set of nodes with all possible arcs from one to another

- Suppose the Web consists of a clique of n nodes and a single additional node that is the successor of each of the n nodes in the clique

![Diagram of a clique with an additional node connected to each node in the clique]
Example 2

- Determine the PageRank of each page assuming $\beta = 0.8$
- Is there a dead end?
- Is there a spider trap?

Example 3

- Suppose that we recursively eliminate dead ends from the Web graph to solve the remaining graph
- Suppose that the graph is a chain of dead ends, headed by a node with a self-loop
- What would be the PageRank assigned to each of the nodes?
Example 3

- What is $v_0$ and $M$?

- $v_0 = [1]$, $M = [\frac{1}{3}, 1]$, PageRank of $A = 1$
- PageRank of $B = \frac{1}{3} \times 1$
- PageRank of $C = \text{PageRank of } B = \frac{1}{3}$
- PageRank of $D = \text{PageRank of } C = \frac{1}{3}$
- ...

Searching pages

- Each search engine has a secret formula that decides the order in which to show pages to the user in response to a search query consisting of one or more search terms.

- Google uses more than 250 different properties of pages.

Generating the final lists

- Selecting candidate pages
  - A page has to have at least one of the search terms in the query
  - Applying weight
  - Presence or absence of search terms in prominent places
    - e.g. headers or the links to the page itself
- Among the qualified pages, a score is computed for each
  - PageRank score

Problems in performing PageRank

- To compute the PageRank for a Web graph
  - We should perform a matrix-vector multiplication of the order of 50 times
  - Until the vector is close to unchanged at one iteration

- The transition matrix of the Web is very sparse
  - Representing it by all its elements is highly inefficient
  - We want to represent the matrix by only its nonzero elements

- We want to reduce the amount of data that must be passed from the Map tasks to Reduce tasks.
Representing Transition Matrices (1/2)

- The average Web page has about 10 out-links
  - We are analyzing a graph of 1.4 billion pages
  - Only one in 0.14 billion (140 million) entries is not 0
  - Can we list the location of the nonzero entries and their values?
  - If we use two 4-byte integers for coordinates (row#, col#) of an element
    and an 8-byte double-precision number for the probability value
  - 16-bytes per nonzero entry
  - The space needed is linear of nonzero entries

Representing Transition Matrices (2/2)

- For the Web graph
  - The value will be 1 divided by the out-degree of the page

\[
M = \begin{bmatrix}
0 & 1/2 & 0 & 0 \\
1/3 & 0 & 1/2 & 0 \\
1/3 & 1/2 & 0 & 0 \\
1/2 & 0 & 1 & 0 \\
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>Source (PR)</th>
<th>Degree</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (l)</td>
<td>3</td>
<td>B, C, D</td>
</tr>
<tr>
<td>B (m)</td>
<td>2</td>
<td>A, D</td>
</tr>
<tr>
<td>C (n)</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D (o)</td>
<td>2</td>
<td>B, C</td>
</tr>
</tbody>
</table>

Mapper generates

\((\text{key, value}) = (\text{destinations, current PR/degree})\)

E.g. for the source A, \((8, 1/3), (C, 1/3), (D, 1/3)\)
For the source B, \((8, m/2), (D, m/2)\)
For the source C, \((C, n)\)
For the source D, \((B, o/2), (C, o/2)\)

Reducer calculates

Add values per node

E.g.
For B: \((B, l/3), (B, o/2)\)
For C: \((C, l/3), (C, n), (C, o/2)\)

\(v_{\text{current}} = <m/2, (l/3+o/2), (l/3+n+o/2), (l/3+m/2)>\)

PageRank Iteration Using MapReduce

- One iteration of the PageRank algorithm involves,
  \(v' = \beta Mv + (1 - \beta)e / n\)

  - First round of MapReduce
    - Calculate \(Mv\) and store the result to \(v'\)
  
  - Second round of MapReduce
    - For each component, multiply \(\beta\) and add \((1 - \beta) / n\)

PageRank Iteration Using MapReduce

- If \(n\) is small enough that each Map task can store the full
  vector \(v\) in main memory
  - And \(v'\)

  - For the Web, \(v\) is much too large to fit in main memory
  - We need striping
    - \(M\) into vertical stripes and break \(v\) into corresponding horizontal strips

Part 1. Large Scale Data Analytics

1. Web-Scale Link Analysis and Social Network Analysis

Link spam
Architecture of a Spam Farm

• **Spam Farm**
  - A collection of pages whose purpose is to increase the PageRank of a certain page or pages

• From the point of view of the spammer, the Web is divided into two parts
  - **Inaccessible pages**
    - The pages that the spammer cannot affect
  - **Accessible pages**
    - Those pages that, while they are not controlled by the spammer, can be affected by the spammer

The Web from the point of view of the link spammer

Understanding Spam Farm (1/2)

• Setting the links to the target page
  - Without link from outside, the spam farm is not useful
  - e.g. Blogs or news papers
    - Comments like "I agree. Please see my article at www.mySpamFarm.com"

Understanding Spam Farm (2/2)

• There is one page \( t \), the target page
  - Spammer attempts to place as much PageRank as possible

• There are a large number of \( m \) supporting pages
  - Accumulate the portion of the PageRank that is distributed equally to all pages
  - The fraction \( 1/\beta \) of the PageRank that represents surfers going to a random page
  - Prevent the PageRank of \( t \) from being lost
  - Note that all of the supporting pages links only to \( t \)

Analysis of a Spam Farm (1/6)

• A taxation parameter \( \beta \)
  - The fraction of a page's PageRank that gets distributed to its successors at the next round

• Let there be,
  - \( n \) pages on the Web in total
  - A target page \( t \)
  - \( m \) supporting pages

Analysis of a Spam Farm (2/6)

• Let \( x \) be the amount of PageRank contributed by the accessible pages
  - \( x \) is the sum over all accessible page \( p \) with a link to \( t \) of the PageRank of \( p \) times \( \beta \) divided by the number of successors of \( p \)

• Finally, let \( y \) be the unknown PageRank of \( t \)
Analysis of a Spam Farm (3/6)

- The PageRank of each supporting page
  - $\beta y/m + (1 - \beta)/n$
- First term represents the contribution from $t$ where $\beta y$ is distributed to $t$'s successors
- Second term is the supporting page's share of the fraction $1 - \beta$ of the PageRank that is divided equally among all pages on the Web

Analysis of a Spam Farm (4/6)

- PageRank of $y$ of target page $t$ is $(1)+(2)+(3)$
  1. Contribution $x$ from outside
     - $x$ is the sum over all accessible page $p$ with a link to $t$, of the PageRank of $p$ times $\beta$ divided by the number of successors of $p$
  2. $\beta$ times the PageRank of every supporting page $\beta m y/m + (1 - \beta)/n$
  3. $(1 - \beta)/n$, the share of the fraction $1 - \beta$ of the PageRank that belongs to $t$

This amount is negligible

Analysis of a Spam Farm (5/6)

- From (1) and (2),
  
  \[
  \begin{align*}
  y &= x + \beta m y/m + (1 - \beta)/n \\
  y &= x / (1 - \beta) + c m/n \\
  \text{Where} \\
  c &= \beta (1 - \beta) / (1 - \beta^2) = \beta / (1 + \beta)
  \end{align*}
  \]

Analysis of a Spam Farm (6/6)

- If we choose $\beta = 0.85$, then $1/(1 - \beta^2) = 3.6$
- $c = \beta (1 + \beta) = 0.46$
  
- The structure has amplified the external PageRank contribution by 360%
- Also, it obtained an amount of PageRank that is 46% of the fraction of the Web, $m/n$, that is in the spam farm

Part 1. Large Scale Data Analytics

1. Web-Scale Link Analysis and Social Network Analysis

   Combatting Link Spam

- Detecting and eliminating link spam have been critical for search engines
  - Just as it was critical to eliminate term spam in the previous decade
- Detecting particular structures
  - Spam farm
    - One page links to a very large number of pages
    - Each of which links back to it
Combatting Link Spam

- Modifying PageRank to lower the rank of link-spam pages automatically
  - TrustRank
  - Spam mass

TrustRank

- TrustRank is a topic-sensitive PageRank
  - “topic” is a set of pages believed to be trustworthy (not spam)
- Develop a suitable teleport set of trustworthy pages
  - Let humans examine a set of pages and decide which of them are trustworthy
  - Pick a domain whose membership is controlled
    - University pages
    - .mil, or .gov

Calculating TrustRank (1/2)

- Then the topic-sensitive PageRank for $S$ is the limit of the iteration,
  \[ v' = \beta M v + (1 - \beta) e_S / |S| \]
- $M$ is the transition matrix of the Web, and $|S|$ is the size of set $S$

Calculating TrustRank (2/2)

- Suppose we use $\beta = 0.8$, and our trust rank is represented by the teleport set (trustworthy pages) $S = \{B, D\}$
- $B$ and $D$ get a higher PageRank than before

Spam Mass

- Measures the fraction of its PageRank that comes from spam for each page
- For an arbitrary page $p$,
  - PageRank $r$
    - Computing ordinary PageRank
    - TrustRank $t$
      - Computing the TrustRank based on some teleport set of trustworthy pages
    - The spam mass
      - $\text{spam mass} \leq A - B$
Example

• Suppose that both the PageRank and TrustRank were computed
• Teleport set was page B and D
  • Which nodes are not the link spams?
  • Is there any link spam?

<table>
<thead>
<tr>
<th>Web Page</th>
<th>PageRank</th>
<th>TrustRank</th>
<th>SpamMass</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3/9</td>
<td>54/210</td>
<td>0.229</td>
</tr>
<tr>
<td>B</td>
<td>2/9</td>
<td>59/210</td>
<td>-0.264</td>
</tr>
<tr>
<td>C</td>
<td>2/9</td>
<td>38/210</td>
<td>0.186</td>
</tr>
<tr>
<td>D</td>
<td>2/9</td>
<td>59/210</td>
<td>-0.264</td>
</tr>
</tbody>
</table>

Example

• Suppose that both the PageRank and TrustRank were computed
• Teleport set was page B and D
  • Which nodes are not the link spams?
  • B and D
  • C has lower chance to be the link spam compared to A

<table>
<thead>
<tr>
<th>Web Page</th>
<th>PageRank</th>
<th>TrustRank</th>
<th>SpamMass</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3/9</td>
<td>54/210</td>
<td>0.229</td>
</tr>
<tr>
<td>B</td>
<td>2/9</td>
<td>59/210</td>
<td>-0.264</td>
</tr>
<tr>
<td>C</td>
<td>2/9</td>
<td>38/210</td>
<td>0.186</td>
</tr>
<tr>
<td>D</td>
<td>2/9</td>
<td>59/210</td>
<td>-0.264</td>
</tr>
</tbody>
</table>

Questions?