FAQs

• Materials covered on Wednesday (9/25)
  • You may use Storm for your term project

• Midterm
  • 10/14/2019 in class

• Term project
  • Please have a kick-off meeting with your team

Part 1. Large Scale Data Analytics

How MapReduce Works

Hadoop Small Files Problem

• Small files
  • Files are significantly smaller than the HDFS block size
  • HDFS cannot handle lots of files!
• Every file/directory and block in HDFS
  • Represented as an object in the namenode’s memory
  • 150 bytes per object (rule of thumb)
• e.g., 10 million files: each using a block: use about 3 gigabytes of memory
• Scaling up much beyond this level is a problem with current hardware
• HDFS is designed for streaming access of large files
CS435  Introduction to Big Data  
Fall 2019 Colorado State University

9/30/2019 Week 6-A  
Sangmi Lee Pallickara

Hadoop Small Files Problem

• MapReduce cannot handle lots of files!  
• Map tasks usually process a block of input at a time  
  • Due to the small files  
  • Each map task will take small input  
  • Large number of map tasks will be created  
• Extra bookkeeping overhead  
  • JVM startup overhead

9/30/2019

Hadoop Small Files Problem

• Help 1: Reusing JVM  
  • mapred.job.reuse.jvm.num.tasks  
  • MultiFileInputSplit  
  • Runs more than one split per map  
• Help 2: HAR files  
  • Hadoop Archives (Use hadoop archive command)  
  • A layered file system on top of HDFS  
  • Users can still see all the files  
  • HDFS manages smaller number of files

Hadoop Small Files Problem

• Help 3: Sequence files  
  • User the file name as a key and the file content as the value  
  • Works very well in practice

Hadoop Small Files Problem

• Help 4: User external storage system  
  • HBASE

Part 1. Large Scale Data Analytics—How MapReduce Works

Chaining Multiple MapReduce Jobs

Chaining jobs (1/2)

• Write multiple driver methods (One for each job)  
• Call the first driver method  
  • JobClient.runJob()  
• When the first job has completed  
• Call the next driver method  
• Creates a new JobConf object

Chaining jobs (2/2)

• Use org.apache.hadoop.mapred.JobControl.Job objects  
  • A Job takes a JobConf object as its constructor argument  
  • Jobs can depend on one another using  
  • x.addDependingJob(y)  
  • Job x cannot start until y has successfully completed

9/30/2019
This material is built based on,

  - Chapter 5

- http://infolab.stanford.edu/~ullman/mmds.html

What are these?

- Archie
- Veronica
- Infoseek
- Snap
- Direct Hit
- Lycos
- AltaVista
- Excite
- Yahoo
- Google

Early Search Engines

- They worked by crawling the Web and listing the terms
  - Words or other strings of characters other than white space
  - In an inverted index

- An inverted index is a data structure that makes it easy, given a term, to find (pointer to) all the places where that term occurs

Inverted index (1/2)

- Inverted index
  - For given texts,

\[ T[0] = "Colorado State University" \\
T[1] = "Colorado water source" \\
T[2] = "University of Colorado" \]

- We have the following inverted file index

\[
\begin{align*}
\text{"Colorado"}: & \{0,1,2\} \\
\text{"State"}: & \{0\} \\
\text{"source"}: & \{1\} \\
\text{"University"}: & \{0,2\} \\
\text{"water"}: & \{1\}
\end{align*}
\]

Inverted index (2/2)

- A term search for the terms, “Colorado”, “State”, and “University” would give the set

\[ \{0,1,2\} \cap \{0\} \cap \{0,2\} = \{0\} \]
Term spam

- If you were selling shirts on the Web
  - All you care about was that people would see your page
- You could add a term like “movie” to your page
  - Add thousands of times
    - Give the same color as background to the letters
- A search engine would think this page is very important one about “movie”
- You could go to the search engine and search “movie” and see the first listed page
  - Copy that page with the same color as background

PageRank

- Goals
  - Providing effective summaries for the search results
  - Ordering/Ranking results
- Simulate random Web surfers
  - Pages that would have a large number of surfers were considered more “important” than pages that would rarely be visited
- The content of a page was judged not only by the terms appearing on that page
  - But by the terms used in or near the links to that page

Definition of PageRank

- A function that assigns a real number to each page in the Web
- The higher the PageRank of a page, the more “important” it is
- There is NOT one fixed algorithm for assignment of PageRank
Example [1/5]

- Page A has links to B, C and D
- Page B has links to A and D
- Page C has a link to A
- Page D has links to B and C

Example [2/5]

- Suppose that a random surfer starts at page A
- Page B, C and D will be the next with probability 1/3
- 0 probability of being at A

Example [3/5]

- Now suppose the random surfer at B
  - B has probability of ½ of being at A, ½ of being at D and 0 of being at B or C

Example [4/5]

- Transition matrix $M$
  - What happens to random surfers after one step
  - $M$ has $n$ rows and columns ($n$ pages)
  - What is the transition matrix for this example?
Example

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

- The first column
  - A surfer at A has a 1/3 probability of next being at each of the other pages
- The second column
  - A surfer at B has a 1/2 probability of being next at A and the same for being at D

What does this matrix mean?  

- The probability distribution for the location of a random surfer
  - A column vector whose jth component is the probability that the surfer is at page j

What does this matrix mean?  

- If we surf at any of the n pages of the Web with equal probability
  - The initial vector \( v_0 \) will have 1/n for each component
  - If \( M \) is the transition matrix of the Web
    - After the first one step, the distribution of the surfer will be \( Mv_0 \)
    - After two steps, \( M(Mv_0) = M^2v_0 \) and so on

What does this matrix mean?  

- Multiplying the initial vector \( v_0 \) by \( M \) a total of \( i \) times
  - The distribution of the surfer after \( i \) steps

What does this matrix mean?  

- The probability \( x_i \) that a random surfer will be at node \( i \) at the next step
  - \( x_i = \sum_{j} m_{ij} v_j \)
  - \( m_{ij} \) is the probability that a surfer at node \( j \) will move to node \( i \) at the next step
  - \( v_j \) is the probability that the surfer was at node \( j \) at the previous step

What does this matrix mean?  

- The distribution of the surfer approaches a limiting distribution \( v \) that satisfies \( v = Mv \) provided two conditions are met:
  1. The graph is strongly connected
     - It is possible to get from any node to any other node
  2. There are no dead ends
     - Nodes that have no arcs out
What does this matrix mean? [6/6]

- The limit is reached when multiplying the distribution by $M$ another time does not change the distribution
- The limiting $v$ is an eigenvector of $M$
- Since $M$ is stochastic (its columns each add up to 1), $v$ is the principle eigenvector
- Its associated eigenvalue is the largest of all eigenvalues
- For the Web, 50-75 iterations are sufficient to converge to within the error limits of double-precision arithmetic

Example

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

- Suppose we apply this process to the matrix $M$
- The initial vector $v_0$ and $v_1$ after multiplying $M$

$$v_0 = \begin{bmatrix} 0/24 \\ 1/24 \\ 1/24 \\ 1/24 \end{bmatrix}$$

$$v_1 = \begin{bmatrix} 9/24 \\ 1/24 \\ 1/24 \\ 1/24 \end{bmatrix}$$

What is the $v_2$?

Example continued

- The sequence of approximations to the limit
- We get by multiplying at each step by $M$ is

$$\begin{array}{cccc}
\frac{1}{2} & \frac{5}{24} & \frac{11}{48} & \frac{7}{32} \\
\frac{9}{24} & \frac{1}{24} & \frac{7}{32} & \frac{2}{9} \\
\frac{5}{24} & \frac{1}{24} & \frac{7}{32} & \frac{2}{9} \\
\frac{1}{24} & \frac{1}{24} & \frac{7}{32} & \frac{2}{9} \\
\end{array}$$

- This difference in probability is not noticeable
- In the real Web, there are billions of nodes of greatly varying importance
- The probability of being at a node like www.amazon.com is orders of magnitude greater than others
### Matrix-vector Multiplication by MapReduce [1/3]

• Suppose we have an $n \times n$ matrix $M$, whose element in row $i$ and column $j$ will be denoted $M_{ij}$.

• Then the matrix-vector product is the vector $x$ of length $n$, whose $i^{th}$ element $x_i$ is given by:

$$x_i = \sum_{j=1}^{n} M_{ij}v_j$$

### Matrix-vector Multiplication by MapReduce [2/3]

• If $n = 100$, we do NOT need DFS or MapReduce.

• However, if this calculation is a part of ranking Web pages ($n$ is 10M) that goes on at search engine? The vector $v$ cannot fit in main memory.

• More than 1.4B pages.

### Matrix-vector Multiplication by MapReduce [3/3]

• The matrix $M$ and the vector $v$ each will be stored in a file of the DFS(HDFS).

• Assume that row-column coordinates of each matrix element will be discoverable.
  - Either from its position in the file or explicit coordinates.

#### The 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 $(i, m_{ij})$.
• All terms of the sum that make up the component $x_i$ of the matrix-vector product will get the same key, $i$.

#### The Reduce function

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

#### If the vector $v$ cannot fit in main memory?

• It is possible that the vector $v$ is so large that it will not fit in its main memory entirely.

• We can divide the matrix into vertical stripes of equal width and divide the vector into an equal number of horizontal stripes of the same height.
  - The goal is to use enough stripes so that the portion of the vector in one stripe can fit into main memory.
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.

$0.002 \quad 0.017 \quad 0.003 \quad 0.010$
$0.000 \quad 0.000 \quad 0.001 \quad 0.000$
$0.012 \quad 0.000 \quad 0.001 \quad 0.000$
$0.001 \quad 0.001 \quad 0.120 \quad 0.000$

.. $i/n$

.. $i/n$

.. $i/n$

.. $i/n$

Results:

$0.002 \times \frac{1}{n} + 0.017 \times \frac{1}{n} + 0.003 \times \frac{1}{n} + 0.010 \times \frac{1}{n} + \ldots$
$= (M_00 \times v_0) + (M_01 \times v_1) + (M_02 \times v_2) + \ldots$

**Link Analysis**

PageRank Algorithm for the real Web

**Structure of Web (1/3)**

- Is the Web strongly connected?

**Structure of Web (2/3)**

- Tubes
  - Pages reachable from the in-component and able to reach the output-component, but unable to reach the SCC or be reached from the SCC.
- Isolated components
  - Unreachable from the large components
Anomalies from the Web structure

• These structures violate the assumptions needed for the Markov process iteration to converge to a limit
  • When a random surfer enters the out-component, they can never leave
  • Surfers starting in either the SCC or in-component are going to wind up in either the out-component or a tendril off the in-component
  • No page in the SCC or in-component winds up with any probability of a surfer being there
• Nothing in the SCC or in-component will be of any importance

Problems we need to avoid

• Dead end
  • A page that has no links out
  • Surfers reaching such a page will disappear
  • In the limit, no page that can reach a dead end can have any PageRank at all
• Spider traps
  • Groups of pages that all have outlinks but they never link to any other pages

Questions?