FAQs
- TP0
  - There may be adjustment of your team composition
- PA1
  - Hadoop and Spark installation video clips are posted

Topics of Today’s Class
- Overview of the Programming Assignment 1
- Distributed Computing Models for Scalable Batch Computing
  - MapReduce

Programming Assignment 1
Hyperlink-Induced Topic Search (HITS)

This material is built based on

http://www.cs.colostate.edu/~cs535
Challenge of content-based ranking for topic search

- Assume that you are looking for "computer"
  - "computer" in the APPLE page?

- How about IBM’s web page?

- O.K… Now, Google?

Challenge of content-based ranking for topic search

- Most useful pages do not include the keyword (that the users are looking for)
  - Pages are not sufficiently descriptive!
  - Semantic mismatch
  - Search keys vs. descriptions

Ranking algorithm to find the most “authoritative” pages for the given topic

- To find the small set of the most authoritative pages that are relevant to the query

- Examples of the authoritative pages
  - For the topic, "python"
    - https://www.python.org/
  - For the information about “Colorado State University”
    - https://www.colostate.edu/

HITS (Hipertext-Induced Topic Search)

- PageRank captures simplistic view of a network

- Authority
  - A Web page with good, authoritative content on a specific topic
  - A Web page that is linked by many hubs

- Hub
  - A Web page pointing to many authoritative Web pages
  - e.g. portal pages (Yahoo)

http://www.cs.colostate.edu/~cs535
**HITS (Hypertext-Induced Topic Search)**

- A.K.A. Hubs and Authorities
  - Jon Kleinberg 1997
  - Topic search
  - Automatically determine hubs/authors

- In practice
  - Performed only on the result set (PageRank is applied on the complete set of documents)
  - Developed for the IBM Clever project
  - Used by Teoma (later Ask.com)

**Understanding Authorities and Hubs**

- Intuitive idea to find authoritative results using link analysis:
  - Not all hyperlinks are related to the conferral of authority
  - Patterns that authoritative pages have
    - Authoritative Pages share considerable overlap in the sets of pages that point to them.

**Calculating Authority/Hub scores**

Each Web page $i$ has an authority score $a_i$ and a hub score $h_i$.

We define the authority score by summing up the hub scores that point to it,

$$ a_i = \sum_j h_j A_{ji} $$

This can be written concisely as,

$$ a_i = \mathbf{h}^T \mathbf{A}_{i} $$

Similarly, we define the hub score of a Web page $i$ by summing up the authority scores $a_j$,

$$ h_i = \sum_j a_j A_{ij} $$

This can be written concisely as,

$$ h_i = \mathbf{A} \mathbf{a}_{i} $$
Hubs and Authorities

Let’s start arbitrarily from \( a^0 = 1, h^0 = 1 \), where 1 is the all-one vector.

\[
\begin{align*}
a^0 &= (1,1,1,1) \\
h^0 &= (1,1,1,1)
\end{align*}
\]

Repeating this, the sequences \( a^0, a^1, a^2, \ldots \) and \( h^0, h^1, h^2, \ldots \) converge (to limits \( x^* \) and \( y^* \)).

\[
\begin{align*}
a^1 &= (1/8, 1/8, 1/4, 1/2) \\
h^1 &= (7/22, 6/22, 5/22, 4/22)
\end{align*}
\]

After the normalization:

\[
\begin{align*}
h^1 &= (7/22, 6/22, 5/22, 4/22) \quad (\text{hub values after the first iteration})
\end{align*}
\]

Implementing Topic Search using HITS

- **Step 1.** Constructing a focused subgraph (root set)
  - Generate a root set from a text-based search engine
  - e.g. pages containing query words

- **Step 2.** Constructing a focused subgraph (base set)
  - For each page \( p \in R \)
    - Add the set of all pages \( p \) points to
    - Add the set of all pages pointing to \( p \)

http://www.cs.colostate.edu/~cs535

Spring 2020 Colorado State University
### Step 4. After the first iteration

<table>
<thead>
<tr>
<th>Node</th>
<th>Hub</th>
<th>Authority</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>3/22</td>
<td>1/8</td>
</tr>
<tr>
<td>P2</td>
<td>6/22</td>
<td>1/8</td>
</tr>
<tr>
<td>P3</td>
<td>5/22</td>
<td>2/8</td>
</tr>
<tr>
<td>P4</td>
<td>4/22</td>
<td>6/8</td>
</tr>
</tbody>
</table>

Ranks
- Hub: P1 > P2 > P3 > P4
- Authority: P1 = P2 < P3 < P4

**Normalization**
- Original paper: using squares sum (to 1)
- You can use sum (to 1)
- \( \text{value} = \text{value} / (\text{sum of all values}) \)

### Step N. Convergence of scores
- Repeat the calculation (step 4) until the scores converge
- You should specify your threshold (maximum number of N)

---

### 3. Distributed Computing Models for Scalable Batch Computing

**Section 1. MapReduce**

- **Introduction to MapReduce**
- ![Hadoop](image)
- ![MapReduce Design Patterns](image)

---

This material is developed based on:
- Download this chapter from the CS435 schedule page
- MapReduce Design Patterns, Donald Miner and Adam Shook, O'Reilly, 2013

---

http://www.cs.colostate.edu/~cs535

---

Spring 2020 Colorado State University
What is MapReduce?

MapReduce is inspired by the concepts of map and reduce in Lisp.

"Modern" MapReduce
- Developed within Google as a mechanism for processing large amounts of raw data.
- Crawled documents or web request logs
- Distributes these data across thousands of machines
- Same computations are performed on each CPU with different dataset

MapReduce provides an abstraction that allows engineers to perform simple computations while hiding the details of parallelization, data distribution, load balancing and fault tolerance.

MapReduce
- MapReduce maps input key/value pairs to a set of intermediate key/value pairs
  - Maps are the individual tasks that transform input records into intermediate records
  - The transformed intermediate records do not need to be of the same type as the input records
  - A given input pair may map to zero or many output pairs
- The Hadoop MapReduce framework spawns one map task for each InputSplit generated by the InputFormat for the job

Reducer
- Reducer reduces a set of intermediate values which share a key to a smaller set of values
- Reducer has 3 primary phases
  - Shuffle, sort and reduce
  - Shuffle
    - Input to the reducer is the sorted output of the mappers
    - The framework fetches the relevant partition of the output of all the mappers via HTTP
  - Sort
    - The framework groups input to the reducer by keys

MapReduce Example 1

http://www.cs.colostate.edu/~cs535
Example 1: NCDC data example
- A national climate data center record
- Find the maximum temperature of a year (1900 ~ 1999)

```
0057 # WMAF weather station identifier
9999 # WBAN weather station identifier
19500101 # observation date
0300 # observation time (4)
+ 51317 # latitude (degrees x 1000)
+ 028783 # longitude (degrees x 1000)
FM-12 # elevation (meters)
9999 # WBAN
020 # wind direction (degrees)
1 # quality code
```

Analyzing the data with Unix Tools (1/2)
- A program for finding the maximum recorded temperature by year from NCDC weather records

```
#!/usr/bin/env bash
for year in all/*
done
```

Analyzing the data with Unix Tools (2/2)
- The script loops through the compressed year files
- Printing the year
- Processing each file using awk
  - Extracts two fields
  - Air temperature and the quality code
- Check if it is greater than the maximum value seen so far

```
% ./max_temperature.sh
1901 317
1902 244
1903 289
1904 256
1905 283
```

Results?
- The complete run for the century took 42 minutes
- To speed up the processing
  - We need to run parts of the program in parallel
  - Process different years in different processes
  - What will be the problems?

Challenges
- Dividing the work into equal-size pieces
  - Data size per year?
- Combining the results from independent processes
  - Combining results and sorting by year?
- You are still limited by the processing capacity of a single machine (the worst one!)

http://www.cs.colostate.edu/~cs535
Map and Reduce

- MapReduce works by breaking the processing into two phases
  - The map phase
  - The reduce phase
- Each phase has key-value pairs as input and output
- Programmers should specify
  1. Types of input/output key-values
  2. The map function
  3. The reduce function
- Optional components
  1. Combiner
  2. Partitioner
  3. InputFormat/OutputFormat

Visualizing the way the MapReduce works (1/3)

Sample lines of input data

These lines are presented to the map function as the key-value pairs

The keys are the line offsets within the file (optional).

Visualizing the way the MapReduce works (2/3)

The map function extracts the year and the air temperature and emit them as its output

This output key-value pairs will be sorted (by key) and grouped by key
Values passed to each reducer are NOT sorted
Our reduce function will see the following input:

This is the final output

Visualizing the way the MapReduce works (3/3)

Reduce function iterates through the list and pick up the maximum reading

Example 2: WordCount

- For text files stored under /usr/joe/wordcount/input, count the number of occurrences of each word
- How do files and directory look?

```
$ hadoop dfs -ls /usr/joe/wordcount/input/
/usr/joe/wordcount/input/file01
/usr/joe/wordcount/input/file02
$ hadoop dfs -cat /usr/joe/wordcount/input/file01
hello World, New World!
$ hadoop dfs -cat /usr/joe/wordcount/input/file02
Hello Hadoop, Goodbye to Hadoop.
```

MapReduce Example 2

http://www.cs.colostate.edu/~cs535

Spring 2020 Colorado State University
Example 2: WordCount

- Run the MapReduce application

```bash
$ bin/hadoop jar /usr/joe/wordcount.jar org.myorg.WordCount
```

```bash
$ bin/hadoop dfs -cat /usr/joe/wordcount/output/part-00000
```

Bye 1
Goodbye 1
World: 1
World: 1
hadoop: 1
Hello 2
World! 1
Hadoop, 1
to 1

MapReduce data flow with a single reducer
Answer

- Assume that all the InstagramUserIDs are unique.
- (1) Mapper
  Input: <dummy key (e.g. file offset number), a line of the input file>
  Functionality: Tokenize the string and calculate the age
  Output: <age, InstagramUserID, number_of_followers>
- (2) Reducer
  Input: <age, a list of [InstagramUserID, number_of_followers]>
  Functionality: Scan the list of values and identify top 10 Instagram users with highest number of followers
  Output: <age, a list of users>

Better Answer: Top-N design pattern

- Assume that all the ClientIDs are unique.
- (1) Mapper
  Input: <dummy key (e.g. file offset number), a line of the input file>
  Functionality: Create multiple data structures (HashSet:<LocalTop10> to store the local top 10 Instagram users’ information (localID and number of followers) per age)
  Output: <age, InstagramUserID, number_of_followers>
- (2) Reducer
  Input: <age, a list of [localTop10]>
  Functionality: Scan the list of values and identify top 10 users with highest number of followers
  Output: <age, a list of users>
- This approach will reduce the communication within your MR cluster significantly.
Better Answer: Top-N design pattern: More Info

- A map function can generate 0 or more outputs.
- setup() and cleanup() are called for each Mapper and Reducer "only once". So, if there are 20 mappers running (10,000 inputs each), the setup/cleanup will be called only 20 times.

- Example:
  
  ```java
  public void run(Context context) throws IOException, InterruptedException {
    setup(context);
    try { while (context.nextKey()) {
      reduce(context.getCurrentKey(), context.getValues(), context);
    } }
    finally { cleanup(context); }
  }
  ```

Comparison with other systems

- MPI vs. MapReduce
  - MapReduce tries to colocate the data with the compute node
  - Data access is fast
    - Data is local!
- Volunteer computing vs. MapReduce
  - SETI@home
    - Using donated CPU time

What are the differences between MapReduce vs. SETI@home?

Data locality optimization

- Hadoop tries to run the map task on a node where the input data resides in HDFS
  - Minimizes usage of cluster bandwidth
- If all replication nodes are running other map tasks
  - The job scheduler will look for a free map slot on a node in the same rack

Data movement in Map tasks

- The process by which the system performs the sort and transfers the map outputs to the reducers as inputs
  - MapReduce guarantees that the input to every reducer is sorted by key

http://www.cs.colostate.edu/~cs535
Combiner functions
- Minimize data transferred between map and reduce tasks
- Users can specify a **combiner function**
  - To be run on the map output
  - To replace the map output with the combiner output

Combiner example
- Example (from the previous max temperature example)
  - The first map produced:
    - (1950, 0), (1950, 20), (1950, 10)
  - The second map produced:
    - (1950, 25), (1950, 15)
  - The reduce function is called with a list of all the values:
    - (1950, [0, 20, 10, 25, 15])
  - Output will be:
    - (1950, 25)
  - We may express the function as:
    - $\max(0, 20, 10, 25, 15)$
    - $= \max(\max(0, 20, 10), \max(25, 15))$
    - $= \max(20, 25) = 25$

Combiner function
- Run a **local** reducer over Map output
- Reduce the amount of data shuffled between the mappers and the reducers
- Combiner cannot replace the reduce function
- Why?

Questions?