No Cell phones in the class.
If you need to use a laptop, please sit in the back row.
I will ask you to turn off your laptop if it seems to be distracting to others.

No Cell phones in the class.
If you need to use a laptop, please sit in the back row.
I will ask you to turn off your laptop if it seems to be distracting to others.

PART 1. LARGE SCALE DATA ANALYTICS

Sangmi Lee Pallickara
Computer Science, Colorado State University
http://www.cs.colostate.edu/~cs435

No Cell phones in the class.
If you need to use a laptop, please sit in the back row.
I will ask you to turn off your laptop if it seems to be distracting to others.

FAQs

• PA1 has been posted
  • September 18, 5:00PM via Canvas
  • Individual submission (No team submission)

• Team for term project
  • 3-4 members
  • Aug. 30, 5:00PM via Canvas
  • Looking for team members? Post your advertisement on Piazza!

Topics

• Introduction to MapReduce
• MapReduce Design Pattern I. Summarization Patterns

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
NCDC data example

- A national climate data center record
- Find the maximum temperature of a year (1900 ~ 1999)

```
0057 332130 # USAF weather station identifier
99999 # WBAN weather station identifier
19500101 # observation date
0300 # observation time
+51317 # latitude (degrees x 1000)
+028783 # longitude (degrees x 1000)
FM -12
+0171 # elevation (meters)
V020
52 # wind direction (degrees)
1 # quality code
```

The first entries for 1990

```
ls raw/1990 | head
```

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/*
do
  echo -ne " basename $ year .gz " \```
  basename $ year .gz

  gunzip -c $ year | awk '{ temp = substr( $0, 88, 5) + 0;
  q = substr( $0, 93, 1);
  if (temp != 9999 && q ~ /[01459]/ && temp > max)
    max = temp } END { print max }'

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 246
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!)*

**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
    - Types of input/output key-values
    - The map function
    - The reduce function

**Visualizing the way the MapReduce works (1/3)**

Sample lines of input data

```
00470199898999919501517004... 99999998 00001 +99999999999...
00480199898999919501517004... 99999998 00022 +99999999999...
00490199898999919501517004... 99999998 00011 +99999999999...
00401264909991949032412004... 99999998 00011 +99999999999...
```

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

```
[(0, 00470199898999919501517004... 99999998 00001 +99999999999...)],
[(106, 00480199898999919501517004... 99999998 00022 +99999999999...)],
[(212, 00490199898999919501517004... 99999998 00011 +99999999999...)],
```

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

```
(1949, 111),
(1950, 22),
(1950, -11)
```

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

```
[1949, 111],
[1950, 22]
```

This is the final output

```
(1949, 111),
(1950, 22)
```

**Visualizing the way the MapReduce works (3/3)**

```
Visualizing the way the MapReduce works (3/3)
```

**Comparison with other systems**

- **MPI vs. MapReduce**
  - MapReduce tries to collocate the data with the compute node
  - Data access is fast
- **Volunteer computing vs. MapReduce**
  - SETI@Home
    - Using donated CPU time
MapReduce Example 3

Dataset
• Network communication log file

• There are two attributes
  • From and To
  • (IP1, IP2) - There was a communication from IP1 to IP2

Retrieve records with “220.4.5.20” as a source IP

• Check each record and provide as output only those records containing “220.4.5.20” as a source IP

Retrieve records with “220.4.5.20” as the source

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
Why Design Patterns?

- Tools for solving problems
  - Reusable and providing a general framework
  - Developers can spend less time figuring out how she/he is going to solve the problem

Summarization Patterns

- Design patterns that produce a top-level, summarized view of your data
  - Helps you to glean insights not available from looking at a localized set of records alone
  - Numerical summarizations
    - Max/Min, Count, Average, Median, Standard Deviation
  - Inverted index

Design Pattern 1: Summarization Patterns

Numerical Summarizations

- Pattern Description
  - General pattern for calculating aggregate statistical values over data
- Intent
  - Group records together by a key field and calculate a numerical aggregate per group to get a top-level view of the larger data set
- Applicability
  - Numerical summarizations should be used when both of the followings are true
    - Numerical data or counting
    - The data can be grouped by specific fields

Numerical Summarizations: Introduction

- Calculating the Min, Max and Count of Values per user

Example 1. Minimum, Maximum and Count

1. Design your Map function
   - input, output, functionality
   - For each record, Map function produces the key-value pair (key, value)
2. Design your Reduce function
   - input, output, functionality
   - Returns (key, value) as the result
**Example 1. Minimum, Maximum and Count**

Calculating the Min, Max and Count of per user

- **Reducer Code**

```java
public static class MinMaxCountReducer extends Reducer<
Text, MinMaxCountTuple, Text, MinMaxCountTuple>

public void reduce(Text key, Iterable<
MinMaxCountTuple> values, Context context)

result = new MinMaxCountTuple();

for (MinMaxCountTuple val : values)

result.setMax(val.getMax());

result.setMin(val.getMin());

context.write(key, result);
```

**Example 2. Average**

Calculating an average of Values per user

- **Reducer Code --continued**

```java
// Grab the "UserID" since it is what we are grouping by
User user = transformXmlToMap(userId.toString());

// Write out the key and the average comment length
context.write(user, outTuple);
```

---

**Mapper Code**

```java
public static class MinMaxCountMapper extends Mapper<Object, Text,
MinMaxCountTuple, Text, MinMaxCountTuple>

public void map(Object key, Text value, Context context)

MinMaxCountTuple result = new MinMaxCountTuple();

Map<String, Integer> parsed = transformXmlToMap(value.toString());

if (parsed.get("userid") != null) {

int userId = Integer.parseInt(parsed.get("userid"));

context.write(userId, outUserId);

for (String line : parsed.keySet()) {

for (String value : parsed.get(line).split(" ")) {

if (parsed.get(line).get("value") != null && !parsed.get(line).get("value").equals("ERROR")) {

result.setCount(val.getCount());

result.setMin(val.getMin());

result.setMax(val.getMax());

context.write(line, outTuple);
}
}

result.setMin(input); // Assume we have developed a class, parse.

result.setMax(input); // Grab the minimum and maximum date values to the value
result.setCount(input); // Set our user ID as the output key
context.write(userId, outUserId);
```

---

**Numerical Summarization**

**Example 2. Average**

Calculating an average of Values per user

- **Map Function**

For each record, Map function produces the key-value pair [key, value].

- **Reduce Function**

For each shuffled pair (key, [a list of values]) returns [key, value] as the result.

---

**Reducer Code**

```java
public static class MinMaxCountReducer extends Reducer<Text,
MinMaxCountTuple, Text, MinMaxCountTuple> {

public void reduce(Text key, Iterable<
MinMaxCountTuple> values, Context context) throws
IOException, InterruptedException {

MinMaxCountTuple result = new MinMaxCountTuple();

for (MinMaxCountTuple val : values) {

result.setCount(val.getCount());

result.setMin(val.getMin());

result.setMax(val.getMax());

context.write(key, result);
```
Design Pattern 1: Summarization Patterns
Combiner Functions

Example: Find the maximum temperature

- First map produces
  - (1950, 0)
  - (1950, 20)
  - (1950, 10)
- Second map produces
  - (1950, 25)
  - (1950, 15)
- Input to the reduce function
  - (1950, [0, 20, 10, 25, 15])
- Output
  - (1950, 25)

If a combiner finds the maximum temperature for each map output:

- First map produces
  - (1950, 0)
  - (1950, 20)
  - (1950, 10) → (1950, 20)
- Second map produces
  - (1950, 25)
  - (1950, 15) → (1950, 25)
- Input to the reduce function
  - (1950, [0, 20, 10, 25, 15]) → (1950, [20, 25])
- Output

Numerical Summarization
Example 2. Average

- Can we use a combiner function for finding average value?

Map function

1. Design your Map function
   - input, output, functionality

2. Design your Combiner function
   - input, output, functionality

3. Design your Reduce function
   - input, output, functionality

For each record, Map function produces the key-value pair (key, value).

Reduce function

For the shuffled pairs (key, [list of values]) returns (key, value) as the result.
Numerical Summarization

Example 2. Average

- Calculating the Average of Values per user

<table>
<thead>
<tr>
<th>USER-ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Value-1</td>
</tr>
<tr>
<td>B</td>
<td>Value-2</td>
</tr>
<tr>
<td>C</td>
<td>Value-3</td>
</tr>
<tr>
<td>D</td>
<td>Value-4</td>
</tr>
<tr>
<td>E</td>
<td>Value-5</td>
</tr>
<tr>
<td>F</td>
<td>Value-6</td>
</tr>
<tr>
<td>G</td>
<td>Value-7</td>
</tr>
<tr>
<td>H</td>
<td>Value-8</td>
</tr>
<tr>
<td>I</td>
<td>Value-9</td>
</tr>
<tr>
<td>J</td>
<td>Value-10</td>
</tr>
</tbody>
</table>

Reduce function

For the shuffled pairs (userID, [a list of local-average-values, local-count])
Returns (userID, global-average-value) as the result

Combiner function
Calculation local average
Returns (count, [local-average-value, local-count]) as the input

Inverted index of StackOverflow links to Wikipedia

Adding StackOverflow links to each Wikipedia page that is reference in a StackOverflow comment

Structure

Inverted index of StackOverflow links to Wikipedia

Map function
For each record, Map function produces the key-value pair (key, value)

Reduce function
For the shuffled pair (key, [a list of values]), returns [key, value] as the result
Inverted index of StackOverflow links to Wikipedia

Map function
- If there is any link included in "comment" emit <link, commentID>

Reduce function
- All the commentIDs that referred same wikipedia page will be grouped together

Mapper Code

```java
public static class WikipediaExtractor extends Mapper<Object, Text, Text, Text> {
    private Text link = new Text();
    private Text outkey = new Text();
    public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {
        Map<String, String> parsed = MRDPUtils.transformXmlToMap(value.toString());
        // Grab the necessary XML attributes
        String txt = parsed.get("Body");
        String posttype = parsed.get("PostTypeId");
        String row_id = parsed.get("Id");

        // if the body is null, or the post is a question (1), skip
        if (txt == null || (posttype != null && posttype.equals("1"))) { return; }

        // Unescape the HTML because the SO data is escaped.
        txt = StringEscapeUtils.unescapeHtml(txt.toLowerCase());
        link.set(getWikipediaURL(txt));
        outkey.set(row_id);
        context.write(link, outkey);
    }
}
```

Questions?