# Housekeeping

## Quiz:

<table>
<thead>
<tr>
<th>Stat</th>
<th>Individual</th>
<th>Team</th>
<th>Lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>16.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>4.73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stat</td>
<td>Individual</td>
<td>Team</td>
<td>Lift</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>Mean</td>
<td>16.94</td>
<td>21.85</td>
<td>4.91</td>
</tr>
<tr>
<td>Median</td>
<td>17</td>
<td>21.5</td>
<td>4.25</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>4.73</td>
<td>4.37</td>
<td>4.97</td>
</tr>
<tr>
<td>Max</td>
<td>27</td>
<td>29</td>
<td>14</td>
</tr>
<tr>
<td>Min</td>
<td>10</td>
<td>13</td>
<td>-7</td>
</tr>
</tbody>
</table>
Housekeeping

Lab 4:

Test Cases are now correct

Remote MongoDB connection

"server": "ambari-head.csc.calpoly.edu"
Cal Poly VPN

Robot Password Changes
Housekeeping

Lab 4:

Test Cases are now correct

Remote MongoDB connection

“server”: “ambari-head.csc.calpoly.edu”
Cal Poly VPN

Robot Password Changes
MapReduce
Motivation: The Google Example

The World Wide Web:
Motivation: The Google Example

The World Wide Web:

Motivation: The Google Example

The World Wide Web

{“Cal”, “Poly”, “San”, “Luis”, “Obispo”, “university”}...


{“Covid-19”, “Newsom”, “beach”, “stay-at-home”}...

{“students”, “university”, “on-line”, “classes”, “sleep”}...
Motivation: The Google Example

The Inverted Index

“university”

“Covid-19”

“Luis”

“Obispo”

“beach”

“sleep”
Motivation: The Google Example

The Inverted Index

"university" → ... → ... → ... → "sleep"
Motivation: The Google Example

The Inverted Index

"university"  ➔  ...  ➔  BUT HOW?

"Covid-19"  ➔  ...  ➔  Distributed

"Luis"  ➔  ...  ➔  (Petabyte scale index)

"Obispo"  ➔  ...  ➔  Fast

"beach"  ➔  ...  ➔  Simple to write

"sleep"  ➔  ...
MapReduce

Jeffrey Dean, Sanjay Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*

Noticed that a lot of code of distributed computing kept doing same “types” of things.

Writing distributed code is hard

Proposed a level of abstraction
Data

<key, value> pairs
Data Processing

<key, value> pairs

All distributed computing reduced to three types of operations

Map: from <key, value> → <key1, value1>

Shuffle: collect keys

Reduce: from <key, [value1, value2,..,valueN] → <key1, value1>
Data Processing

<key, value> pairs

All distributed computing reduced to three types of operations

**Map**: from <key, value> → <key1, value1>

**Shuffle**: collect keys (most always the same)

**Reduce**: from <key, [value1, value2, .., valueN] → <key1, value1>
MapReduce

Write a Map() and Reduce() transformations of data
- Simple code

Build a distributed computing framework that does the rest
MapReduce: Inverted Index

Map(key, value):  //key=url, value= **bag of words**
  for word in value do
    emit(word, key)
  end for

Reduce(key, values): //key=**word**, values= [url1,...,urln]
  return(key, values)
More Formally: Map()

\[ \text{Map: } K \times V \rightarrow K' \times V' \]

\( K, K' \) -- universes of keys

\( V, V' \) -- universes of values (can be compound)

Transformation
More Formally: Map()

Map: \( K \times V \rightarrow \{ K' \times V' \} \)

\( K, K' \) -- universes of keys

\( V, V' \) -- universes of values (can be compound)

Transformation
More Formally: Map()

Map: $K \times V \rightarrow \{K' \times V\}$

$K, K'$ -- universes of keys

$V, V'$ -- universes of values (can be compound)

emit() instead of return()

Transformation
More Formally: Map()

Map: \( K \times V \rightarrow \{K' \times V\} \)

```plaintext
map(key, value): //value - bag of words
    for word in value:
        emit(word, 1)
    end for
```
More Formally: Reduce()

Reduce:  $K \times (V)^* \rightarrow (V)^*$

Reduce  $K \times (V)^* \rightarrow K \times (V)^*$

Aggregation
More Formally: Reduce()

\[ \text{Map: } K \times (V)^* \rightarrow (V)^* \]

\[ \text{Map: } K \times (V)^* \rightarrow K \times (V)^* \]

reduce(key, value):    //value - [1,1,1,...,1]
    count := 0
    for x in value:
        count := count+x
    end for
    emit(key, count)
Map-Shuffle-Reduce
Map-Shuffle-Reduce

Mappers
Map-Shuffle-Reduce

Mappers
Map-Shuffle-Reduce

Mappers
Map-Shuffle-Reduce

Mappers
Map-Shuffle-Reduce

Mappers
Map-Shuffle-Reduce

Mappers
Map-Shuffle-Reduce

Mappers

Shuffle
Map-Shuffle-Reduce

Mappers

Shuffle

Reducers
Map-Shuffle-Reduce

Mappers

Shuffle

Reducers
Map-Shuffle-Reduce

Mappers

Shuffle

Reducers
Map-Shuffle-Reduce

Mappers

Shuffle

Reducers