Map-Reduce and the Shuffle

Yoav Freund / Big Data Analytics
The contract
The contract

1. All elements with the same key go to the same reducer.
The contract

1. All elements with the same key go to the same reducer.
2. The input to each reducer is sorted by key.
Logical data flow

Figure 2-1. MapReduce logical data flow
mapper locality
Task with no reducers
Task with a single reducer

Figure 2-3. MapReduce data flow with a single reduce task
Task with two reducers

Figure 2-4. MapReduce data flow with multiple reduce tasks
Shuffle and Sort

Figure 6-6. Shuffle and sort in MapReduce
Performance improvements
Performance improvements

• Compression - reduce bandwidth
Performance improvements

- Compression - reduce bandwidth
- Combiners - Start performing the reduction on the mapper machine.
Performance improvements

- Compression - reduce bandwidth
- Combiners - Start performing the reduction on the mapper machine.
  - Reduces communication bandwidth
Performance improvements

• Compression - reduce bandwidth

• Combiners - Start performing the reduction on the mapper machine.
  • Reduces communication bandwidth
  • Supported by: max, sum, ....
Performance improvements

• Compression - reduce bandwidth

• Combiners - Start performing the reduction on the mapper machine.
  • Reduces communication bandwidth
    • Supported by: max, sum, ....
    • Not supported by: average, median, ....
Examples
Examples

• Input: table of temperature as function of time (minute) and location (10,000 locations)
Examples

• Input: table of temperature as function of time (minute) and location (10,000 locations)

• Each file contains one day of measurements for a particular location.
Examples

• Input: table of temperature as function of time (minute) and location (10,000 locations)

• Each file contains one day of measurements for a particular location.

• Tasks:
Examples

• Input: table of temperature as function of time (minute) and location (10,000 locations)

• Each file contains one day of measurements for a particular location.

• Tasks:
  • Find the largest daily swing (max-min) for a set of locations.
Examples

• Input: table of temperature as function of time (minute) and location (10,000 locations)

• Each file contains one day of measurements for a particular location.

• Tasks:
  • Find the largest daily swing (max-min) for a set of locations.
  • Find the average and empirical std for the max temp in each location.
Examples

• Input: table of temperature as function of time (minute) and location (10,000 locations)

• Each file contains one day of measurements for a particular location.

• Tasks:
  • Find the largest daily swing (max-min) for a set of locations.
  • Find the average and empirical std for the max temp in each location.
  • For each pair of locations that are within 10 miles of each other find the average of the daily abs-distance.
HBase

- When you have a small number of large files, HDFS is fine.
HBase

- When you have a small number of large files, HDFS is fine.
- When the number of files > 1,000 then managing file names and the directory hierarchy becomes cumbersome and SLOW
HBase

- When you have a small number of large files, HDFS is fine.
- When the number of files > 1,000 then managing file names and the directory hierarchy becomes cumbersome and SLOW
- HBase is a distributed column-oriented database that sits on top of HDFS.
HBase

- When you have a small number of large files, HDFS is fine.
- When the number of files > 1,000 then managing file names and the directory hierarchy becomes cumbersome and SLOW.
- HBase is a distributed column-oriented database that sits on top of HDFS.
- Good for analytics - scanning many items. Not good for transactions.
Column Oriented DB
Logical Schema

Figure 1-2. The Hush schema expressed as an ERD
### HBase Schema

#### Table: shorturl

<table>
<thead>
<tr>
<th>Row Key:</th>
<th>shortId</th>
</tr>
</thead>
</table>

**Family:**
- data: Columns: url, refShortId, userId, clicks
- stats-daily: [ttl: 7 days]
  Columns: YYYYMMDD, YYYYMMDD\*x00\<country-code>
- stats-weekly: [ttl: 4 weeks]
  Columns: YYYYWW, YYYYWW\*x00\<country-code>
- stats-monthly: [ttl: 12 months]
  Columns: YYYYMM, YYYYMM\*x00\<country-code>

**Table: url**

<table>
<thead>
<tr>
<th>Row Key:</th>
<th>MD5(url)</th>
</tr>
</thead>
</table>

**Family:**
- data: [compressed]
  Columns: refShortId, title, description
- content: [compressed]
  Columns: raw

**Table: user-shorturl**

<table>
<thead>
<tr>
<th>Row Key:</th>
<th>username*x00shortId</th>
</tr>
</thead>
</table>

**Family:**
- data: Columns: timestamp

**Table: user**

<table>
<thead>
<tr>
<th>Row Key:</th>
<th>username</th>
</tr>
</thead>
</table>

**Family:**
- data: Columns: credentials, roles, firstname, lastname, email

*Figure 1-3. The Hush schema in HBase*
Column Families

<table>
<thead>
<tr>
<th></th>
<th>column A (int)</th>
<th>column B (varchar)</th>
<th>column C (boolean)</th>
<th>column D (date)</th>
</tr>
</thead>
<tbody>
<tr>
<td>row A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>row B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>row C</td>
<td></td>
<td></td>
<td><strong>NULL?</strong></td>
<td></td>
</tr>
<tr>
<td>row D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 1-4. Rows and columns in HBase*
Regions - Partitioning the rows

Figure 1-7. Rows grouped in regions and served by different servers
Happybase
Happybase

- A Python Library for interacting with HBase.
Happybase

- A Python Library for interacting with HBase.
- Allows high-level interaction - short scripts
Happybase

- A Python Library for interacting with HBase.
- Allows high-level interaction - short scripts
- A script for loading a CSV file into hbase:
Happybase

- A Python Library for interacting with HBase.
- Allows high-level interaction - short scripts
- A script for loading a CSV file into hbase:
  - cd $CSD181/hadoop/hbase_scripts/
Happybase

- A Python Library for interacting with HBase.
- Allows high-level interaction - short scripts
- A script for loading a CSV file into hbase:
  - cd $CSD181/hadoop/hbase_scripts/
  - csv2hbase.py roster.csv New_Table
Todo:
Todo:

- Start loading your data into hdfs or hbase.
Todo:

• Start loading your data into hdfs or hbase.
• Describe your project in your project page
Todo:

• Start loading your data into hdfs or hbase.
• Describe your project in your project page
• Finalize teams
Todo:

- Start loading your data into hdfs or hbase.
- Describe your project in your project page
- Finalize teams
- Schedule a time to chat with me.