MapReduce
Single-node architecture

Machine Learning, Statistics

“Classical” Data Mining
Motivation (Google example)

- 20+ billion web pages x 20KB = 400+ TB
- 1 computer reads 30-35 MB/sec from disk
  - ~4 months to read the web
- ~1,000 hard drives to store the web
- Even more to do something with the data
Commodity Clusters

- Web data sets are massive
  - Tens to hundreds of terabytes
- Cannot mine on a single server
- Standard architecture emerging:
  - Cluster of commodity Linux nodes
  - Gigabit ethernet interconnect
- How to organize computations on this architecture?
  - Mask issues such as hardware failure
Big computation – Big machines

- **Traditional big-iron box** (circa 2003)
  - 8 2GHz Xeons
  - 64GB RAM
  - 8TB disk
  - 758,000 USD

- **Prototypical Google rack** (circa 2003)
  - 176 2GHz Xeons
  - 176GB RAM
  - ~7TB disk
  - 278,000 USD

- In Aug 2006 Google had ~450,000 machines
Cluster Architecture

1 Gbps between any pair of nodes in a rack

2-10 Gbps backbone between racks

Each rack contains 16-64 nodes
Yahoo M45 cluster:

- Datacenter in a Box (DiB)
- 1000 nodes, 4000 cores, 3TB RAM, 1.5PB disk
- High bandwidth connection to Internet
- Located on Yahoo! campus
- World’s top 50 supercomputer
Large scale computing

- **Large scale computing for data mining problems on commodity hardware:**
  - PCs connected in a network
  - Process huge datasets on many computers
- **Challenges:**
  - How do you distribute computation?
  - Distributed/parallel programming is hard
  - Machines fail
- **Map-reduce** addresses all of the above
  - Google’s computational/data manipulation model
  - Elegant way to work with big data
Implications of such computing environment:

- Single machine performance does not matter
  - Add more machines

- Machines break:
  - One server may stay up 3 years (1,000 days)
  - If you have 1,000 servers, expect to lose 1/day

How can we make it easy to write distributed programs?
Idea and solution

- **Idea:**
  - Bring computation close to the data
  - Store files multiple times for reliability

- **Need:**
  - Programming model
    - Map-Reduce
  - Infrastructure – *File system*
    - Google: GFS
    - Hadoop: HDFS
**Problem:**
- If nodes fail, how to store data persistently?

**Answer:**
- **Distributed File System:**
  - Provides global file namespace
  - Google GFS; Hadoop HDFS; Kosmix KFS

**Typical usage pattern**
- Huge files (100s of GB to TB)
- Data is rarely updated in place
- Reads and appends are common
Distributed File System

- **Chunk Servers:**
  - File is split into contiguous chunks
  - Typically each chunk is 16-64MB
  - Each chunk replicated (usually 2x or 3x)
  - Try to keep replicas in different racks

- **Master node:**
  - a.k.a. Name Nodes in Hadoop’s HDFS
  - Stores metadata
  - Might be replicated

- **Client library for file access:**
  - Talks to master to find chunk servers
  - Connects directly to chunkservers to access data
Distributed File System

- Reliable distributed file system for petabyte scale
- Data kept in “chunks” spread across thousands of machines
- Each chunk *replicated* on different machines
  - Seamless recovery from disk or machine failure

Bring computation directly to the data!
Warm up: Word Count

- We have a large file of words:
  - one word per line

- Count the number of times each distinct word appears in the file

- Sample application:
  - Analyze web server logs to find popular URLs
Case 1:
- Entire file fits in memory

Case 2:
- File too large for memory, but all \(<\text{word, count}\>\) pairs fit in memory

Case 3:
- File on disk, too many distinct words to fit in memory:
  - `sort datafile | uniq -c`
Suppose we have a large corpus of documents.

Count occurrences of words:

- `words(docs/*) | sort | uniq -c`
  - where `words` takes a file and outputs the words in it, one per a line

Captures the essence of **MapReduce**
- Great thing is it is naturally parallelizable.
Map-Reduce: Overview

- Read a lot of data
- **Map:**
  - Extract something you care about
- Shuffle and Sort
- **Reduce:**
  - Aggregate, summarize, filter or transform
- Write the result

Outline stays the same, **map** and **reduce** change to fit the problem
More specifically

- Program specifies two primary methods:
  - Map(k,v) → <k’, v’>*
  - Reduce(k’, <v’>*) → <k’, v’’>*

- All values v’ with same key k’ are reduced together and processed in v’ order
The crew of the space shuttle Endeavor recently returned to Earth as ambassadors, harbingers of a new era of space exploration. Scientists at NASA are saying that the recent assembly of the Dextre robot is the first step in a long-term space-based man/machine partnership. "The work we're doing now -- the robotics we're doing -- is what we're going to need to do to build any work station or habitat structure on the moon or Mars," said Allard Beutel.

**Big document**

**MAP:**
reads input and produces a set of key value pairs

Provided by the programmer

- (the, 1)
- (crew, 1)
- (of, 1)
- (the, 1)
- (space, 1)
- (shuttle, 1)
- (Endeavor, 1)
- (recently, 1)

**Group by key:**
Collect all pairs with same key

Obtained from the program

- (crew, 1)
- (crew, 1)
- (space, 1)
- (the, 1)
- (the, 1)
- (the, 1)
- (shuttle, 1)
- (recently, 1)

**Reduce:**
Collect all values belonging to the key and output

Provided by the programmer

- (crew, 2)
- (space, 1)
- (the, 3)
- (shuttle, 1)
- (recently, 1)

**Only sequential reads**
map(key, value):
// key: document name; value: text of document
    for each word w in value:
        emit(w, 1)

reduce(key, values):
// key: a word; value: an iterator over counts
    result = 0
    for each count v in values:
        result += v
    emit(result)
Map-Reduce: Environment

- Map-Reduce environment takes care of:
  - Partitioning the input data
  - Scheduling the program’s execution across a set of machines
  - Handling machine failures
  - Managing required inter-machine communication

- Allows programmers without a PhD in parallel and distributed systems to use large distributed clusters
**Map-Reduce: A diagram**

*Input:*
- **MAP:** reads input and produces a set of key value pairs

*Intermediate:*
- **Group by key:** Collect all pairs with same key

*Grouped:*
- **Reduce:** Collect all values belonging to the key and output

*Output:*
**Map-Reduce**

- **Programmer specifies:**
  - Map and Reduce and input files

- **Workflow:**
  - Read inputs as a set of key-value-pairs
  - **Map** transforms input kv-pairs into a new set of k'v'-pairs
  - Sorts & Shuffles the k'v'-pairs to output nodes
  - All k'v'-pairs with a given k' are sent to the same **reduce**
  - **Reduce** processes all k'v'-pairs grouped by key into new k''v''-pairs
  - Write the resulting pairs to files

- All phases are distributed with many tasks doing the work
Map-Reduce: in Parallel

Map Task 1

Map Task 2

Map Task 3

Partitioning Function

Sort and Group

Reduce Task 1

Sort and Group

Reduce Task 2
Input and final output are stored on a distributed file system:
- Scheduler tries to schedule map tasks “close” to physical storage location of input data

Intermediate results are stored on local FS of map and reduce workers

Output is often input to another map reduce task
Master data structures:

- Task status: (idle, in-progress, completed)
- Idle tasks get scheduled as workers become available
- When a map task completes, it sends the master the location and sizes of its R intermediate files, one for each reducer
- Master pushes this info to reducers

- Master pings workers periodically to detect failures
Failures

- Map worker failure
  - Map tasks completed or in-progress at worker are reset to idle
  - Reduce workers are notified when task is rescheduled on another worker
- Reduce worker failure
  - Only in-progress tasks are reset to idle
- Master failure
  - MapReduce task is aborted and client is notified
M map tasks, R reduce tasks

Rule of thumb:
- Make M and R much larger than the number of nodes in cluster
- One DFS chunk per map is common
- Improves dynamic load balancing and speeds recovery from worker failure

Usually R is smaller than M
- because output is spread across R files
Fine granularity tasks: map tasks >> machines

- Minimizes time for fault recovery
- Can pipeline shuffling with map execution
- Better dynamic load balancing
MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 00 min 18 sec
323 workers; 0 deaths

<table>
<thead>
<tr>
<th>Type</th>
<th>Shards</th>
<th>Done</th>
<th>Active</th>
<th>Input(MB)</th>
<th>Done(MB)</th>
<th>Output(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>13853</td>
<td>0</td>
<td>323</td>
<td>878934.6</td>
<td>1314.4</td>
<td>717.0</td>
</tr>
<tr>
<td>Shuffle</td>
<td>500</td>
<td>0</td>
<td>323</td>
<td>717.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Reduce</td>
<td>500</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Counters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapped (MB/s)</td>
<td>72.5</td>
</tr>
<tr>
<td>Shuffle (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Output (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>doc-index-hits</td>
<td>145825686</td>
</tr>
<tr>
<td>docs-indexed</td>
<td>506631</td>
</tr>
<tr>
<td>dups-in-index-merge</td>
<td>0</td>
</tr>
<tr>
<td>mr-operator-calls</td>
<td>508192</td>
</tr>
<tr>
<td>mr-operator-outputs</td>
<td>506631</td>
</tr>
</tbody>
</table>
MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 05 min 07 sec
1707 workers; 1 deaths

<table>
<thead>
<tr>
<th>Type</th>
<th>Shards</th>
<th>Done</th>
<th>Active</th>
<th>Input(MB)</th>
<th>Done(MB)</th>
<th>Output(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>13853</td>
<td>1857</td>
<td>1707</td>
<td>878934.6</td>
<td>191995.8</td>
<td>113936.6</td>
</tr>
<tr>
<td>Shuffle</td>
<td>500</td>
<td>0</td>
<td>500</td>
<td>113936.6</td>
<td>57113.7</td>
<td>57113.7</td>
</tr>
<tr>
<td>Reduce</td>
<td>500</td>
<td>0</td>
<td>0</td>
<td>57113.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Counters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapped (MB/s)</td>
<td>699.1</td>
</tr>
<tr>
<td>Shuffle (MB/s)</td>
<td>349.5</td>
</tr>
<tr>
<td>Output (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>doc-index-hits</td>
<td>5004411944</td>
</tr>
<tr>
<td>docs-indexed</td>
<td>17290135</td>
</tr>
<tr>
<td>dups-in-index-merge</td>
<td>0</td>
</tr>
<tr>
<td>mr-operator-calls</td>
<td>17331371</td>
</tr>
<tr>
<td>mr-operator-outputs</td>
<td>17290135</td>
</tr>
</tbody>
</table>
MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 10 min 18 sec
1707 workers; 1 deaths

<table>
<thead>
<tr>
<th>Type</th>
<th>Shards</th>
<th>Done</th>
<th>Active</th>
<th>Input(MB)</th>
<th>Done(MB)</th>
<th>Output(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>13853</td>
<td>5354</td>
<td>1707</td>
<td>878934.6</td>
<td>406020.1</td>
<td>241058.2</td>
</tr>
<tr>
<td>Shuffle</td>
<td>500</td>
<td>0</td>
<td>500</td>
<td>241058.2</td>
<td>196362.5</td>
<td>196362.5</td>
</tr>
<tr>
<td>Reduce</td>
<td>500</td>
<td>0</td>
<td>0</td>
<td>196362.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Counters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapped (MB/s)</td>
<td>704.4</td>
</tr>
<tr>
<td>Shuffle (MB/s)</td>
<td>371.9</td>
</tr>
<tr>
<td>Output (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>doc-index-hits</td>
<td>5000364228</td>
</tr>
<tr>
<td>docs-indexed</td>
<td>17300709</td>
</tr>
<tr>
<td>dups-in-index-merge</td>
<td>0</td>
</tr>
<tr>
<td>mr-operator-calls</td>
<td>17342493</td>
</tr>
<tr>
<td>mr-operator-outputs</td>
<td>17300709</td>
</tr>
</tbody>
</table>
MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 15 min 31 sec
1707 workers; 1 deaths

<table>
<thead>
<tr>
<th>Type</th>
<th>Shards</th>
<th>Done</th>
<th>Active</th>
<th>Input(MB)</th>
<th>Done(MB)</th>
<th>Output(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>13853</td>
<td>8841</td>
<td>1707</td>
<td>878934.6</td>
<td>621608.5</td>
<td>369459.8</td>
</tr>
<tr>
<td>Shuffle</td>
<td>500</td>
<td>0</td>
<td>500</td>
<td>369459.8</td>
<td>326986.8</td>
<td>326986.8</td>
</tr>
<tr>
<td>Reduce</td>
<td>500</td>
<td>0</td>
<td>0</td>
<td>326986.8</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Counters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapped (MB/s)</td>
<td>706.5</td>
</tr>
<tr>
<td>Shuffle (MB/s)</td>
<td>419.2</td>
</tr>
<tr>
<td>Output (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>doc-index-hits</td>
<td>4982870667</td>
</tr>
<tr>
<td>docs-indexed</td>
<td>17229926</td>
</tr>
<tr>
<td>dups-in-index-merge</td>
<td>0</td>
</tr>
<tr>
<td>mr-operator-calls</td>
<td>17272056</td>
</tr>
<tr>
<td>mr-operator-outputs</td>
<td>17229926</td>
</tr>
</tbody>
</table>
MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 29 min 45 sec
1707 workers; 1 deaths

<table>
<thead>
<tr>
<th>Type</th>
<th>Shards</th>
<th>Done</th>
<th>Active</th>
<th>Input(MB)</th>
<th>Done(MB)</th>
<th>Output(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>13853</td>
<td>13853</td>
<td>0</td>
<td>878934.6</td>
<td>878934.6</td>
<td>523499.2</td>
</tr>
<tr>
<td>Shuffle</td>
<td>500</td>
<td>195</td>
<td>305</td>
<td>523499.2</td>
<td>523389.6</td>
<td>523389.6</td>
</tr>
<tr>
<td>Reduce</td>
<td>500</td>
<td>0</td>
<td>195</td>
<td>523389.6</td>
<td>2685.2</td>
<td>2742.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Counters</th>
<th>Variable</th>
<th>Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mapped (MB/s)</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>Shuffle (MB/s)</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>Output (MB/s)</td>
<td>45.7</td>
</tr>
<tr>
<td></td>
<td>doc-index-hits</td>
<td>2313178</td>
</tr>
<tr>
<td></td>
<td>docs-indexed</td>
<td>7936</td>
</tr>
<tr>
<td></td>
<td>dups-in-index-merge</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>mr-merge-calls</td>
<td>1954105</td>
</tr>
<tr>
<td></td>
<td>mr-merge-outputs</td>
<td>1954105</td>
</tr>
</tbody>
</table>
MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 31 min 34 sec
1707 workers; 1 deaths

<table>
<thead>
<tr>
<th>Type</th>
<th>Shards</th>
<th>Done</th>
<th>Active</th>
<th>Input(MB)</th>
<th>Done(MB)</th>
<th>Output(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>13853</td>
<td>13853</td>
<td>0</td>
<td>878934.6</td>
<td>878934.6</td>
<td>523499.2</td>
</tr>
<tr>
<td>Shuffle</td>
<td>500</td>
<td>500</td>
<td>0</td>
<td>523499.2</td>
<td>523499.5</td>
<td>523499.5</td>
</tr>
<tr>
<td>Reduce</td>
<td>500</td>
<td>0</td>
<td>500</td>
<td>523499.5</td>
<td>133837.8</td>
<td>136929.6</td>
</tr>
</tbody>
</table>

Counts

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapped (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Shuffle (MB/s)</td>
<td>0.1</td>
</tr>
<tr>
<td>Output (MB/s)</td>
<td>1238.8</td>
</tr>
<tr>
<td>doc-index-hits</td>
<td>0.10</td>
</tr>
<tr>
<td>docs-indexed</td>
<td>0</td>
</tr>
<tr>
<td>dups-in-index-merge</td>
<td>0</td>
</tr>
<tr>
<td>mr-merge-calls</td>
<td>51738599</td>
</tr>
<tr>
<td>mr-merge-outputs</td>
<td>51738599</td>
</tr>
</tbody>
</table>
MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 33 min 22 sec
1707 workers; 1 deaths

<table>
<thead>
<tr>
<th>Type</th>
<th>Shards</th>
<th>Done</th>
<th>Active</th>
<th>Input(MB)</th>
<th>Done(MB)</th>
<th>Output(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>13853</td>
<td>13853</td>
<td>0</td>
<td>878934.6</td>
<td>878934.6</td>
<td>523499.2</td>
</tr>
<tr>
<td>Shuffle</td>
<td>500</td>
<td>500</td>
<td>0</td>
<td>523499.2</td>
<td>523499.5</td>
<td>523499.5</td>
</tr>
<tr>
<td>Reduce</td>
<td>500</td>
<td>0</td>
<td>500</td>
<td>523499.5</td>
<td>263283.3</td>
<td>269351.2</td>
</tr>
</tbody>
</table>

Counters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapped (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Shuffle (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Output (MB/s)</td>
<td>1225.1</td>
</tr>
<tr>
<td>docs-indexed</td>
<td>0</td>
</tr>
<tr>
<td>docs-indexed</td>
<td>0</td>
</tr>
<tr>
<td>dups-in-indexed</td>
<td>0</td>
</tr>
<tr>
<td>mr-merge-calls</td>
<td>51842100</td>
</tr>
<tr>
<td>mr-merge-outputs</td>
<td>51842100</td>
</tr>
</tbody>
</table>
MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 35 min 08 sec

1707 workers; 1 deaths

<table>
<thead>
<tr>
<th>Type</th>
<th>Shards</th>
<th>Done</th>
<th>Active</th>
<th>Input(MB)</th>
<th>Done(MB)</th>
<th>Output(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>13853</td>
<td>13853</td>
<td>0</td>
<td>878934.6</td>
<td>878934.6</td>
<td>523499.2</td>
</tr>
<tr>
<td>Shuffle</td>
<td>500</td>
<td>500</td>
<td>0</td>
<td>523499.2</td>
<td>523499.5</td>
<td>523499.5</td>
</tr>
<tr>
<td>Reduce</td>
<td>500</td>
<td>0</td>
<td>500</td>
<td>523499.5</td>
<td>390447.6</td>
<td>399457.2</td>
</tr>
</tbody>
</table>

Counters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapped (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Shuffle (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Output (MB/s)</td>
<td>1222.0</td>
</tr>
<tr>
<td>doc-index-hits</td>
<td>0.10</td>
</tr>
<tr>
<td>docs-indexed</td>
<td>0.0</td>
</tr>
<tr>
<td>dups-in-index-merge</td>
<td>0.0</td>
</tr>
<tr>
<td>mr-merge-calls</td>
<td>51640600</td>
</tr>
<tr>
<td>mr-merge-outputs</td>
<td>51640600</td>
</tr>
</tbody>
</table>
MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 38 min 56 sec
1707 workers; 1 deaths

<table>
<thead>
<tr>
<th>Type</th>
<th>Shards</th>
<th>Done</th>
<th>Active</th>
<th>Input(MB)</th>
<th>Done(MB)</th>
<th>Output(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>13853</td>
<td>13853</td>
<td>0</td>
<td>878934.6</td>
<td>878934.6</td>
<td>523499.2</td>
</tr>
<tr>
<td>Shuffle</td>
<td>500</td>
<td>500</td>
<td>0</td>
<td>523499.2</td>
<td>519781.8</td>
<td>519781.8</td>
</tr>
<tr>
<td>Reduce</td>
<td>500</td>
<td>498</td>
<td>2</td>
<td>519781.8</td>
<td>519394.7</td>
<td>519440.7</td>
</tr>
</tbody>
</table>

Counters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapped (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Shuffle (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Output (MB/s)</td>
<td>9.4</td>
</tr>
<tr>
<td>doc-index-hits</td>
<td>0</td>
</tr>
<tr>
<td>docs-indexed</td>
<td>0</td>
</tr>
<tr>
<td>dups-in-index-merge</td>
<td>0</td>
</tr>
<tr>
<td>mr-merge-calls</td>
<td>394792</td>
</tr>
<tr>
<td>mr-merge-outputs</td>
<td>394792</td>
</tr>
</tbody>
</table>
MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 40 min 43 sec

1707 workers; 1 deaths

<table>
<thead>
<tr>
<th>Type</th>
<th>Shards</th>
<th>Done</th>
<th>Active</th>
<th>Input(MB)</th>
<th>Done(MB)</th>
<th>Output(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>13853</td>
<td>13853</td>
<td>0</td>
<td>878934.6</td>
<td>878934.6</td>
<td>523499.2</td>
</tr>
<tr>
<td>Shuffle</td>
<td>500</td>
<td>500</td>
<td>0</td>
<td>523499.2</td>
<td>519774.3</td>
<td>519774.3</td>
</tr>
<tr>
<td>Reduce</td>
<td>500</td>
<td>499</td>
<td>1</td>
<td>519774.3</td>
<td>519735.2</td>
<td>519764.0</td>
</tr>
</tbody>
</table>

Counters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapped (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Shuffle (MB/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Output (MB/s)</td>
<td>1.9</td>
</tr>
<tr>
<td>doc-index-hits</td>
<td>0</td>
</tr>
<tr>
<td>docs-indexed</td>
<td>0</td>
</tr>
<tr>
<td>dups-in-index-merge</td>
<td>0</td>
</tr>
<tr>
<td>mr-merge-calls</td>
<td>73442</td>
</tr>
<tr>
<td>mr-merge-outs</td>
<td>73442</td>
</tr>
</tbody>
</table>
Problems suited for Map-Reduce

- Want to simulate disease spreading in a network
  - **Input**
    - Each line: node id, virus parameters
  - **Map**
    - Reads a line of input and simulate the virus
    - Output: triplets (node id, virus id, hit time)
  - **Reduce**
    - Collect the node IDs and see which nodes are most vulnerable
Statistical machine translation:
  - Need to count number of times every 5-word sequence occurs in a large corpus of documents

Easy with MapReduce:
  - Map:
    - Extract (5-word sequence, count) from document
  - Reduce:
    - Combine counts
Suppose we have a large web corpus

Look at the metadata file
- Lines of the form (URL, size, date, ...)

For each host, find the total number of bytes
- i.e., the sum of the page sizes for all URLs from that host

Other examples:
- Link analysis and graph processing
- Machine Learning algorithms
Implementations

- Google
  - Not available outside Google
- Hadoop
  - An open-source implementation in Java
  - Uses HDFS for stable storage
- Aster Data
  - Cluster-optimized SQL Database that also implements MapReduce
Cloud Computing

- Ability to rent computing by the hour
  - Additional services e.g., persistent storage

- Amazon’s “Elastic Compute Cloud” (EC2)

- Aster Data and Hadoop can both be run on EC2

- For CS345 (offered next quarter) Amazon will provide free access for the class
Problem:
- Slow workers significantly lengthen the job completion time:
  - Other jobs on the machine
  - Bad disks
  - Weird things

Solution:
- Near end of phase, spawn backup copies of tasks
  - Whichever one finishes first “wins”

Effect:
- Dramatically shortens job completion time
Refinements: Backup tasks

- Backup tasks reduce job time
- System deals with failures
Often a map task will produce many pairs of the form \((k,v_1), (k,v_2), \ldots\) for the same key \(k\)
- E.g., popular words in Word Count

Can save network time by **pre-aggregating at mapper**:
- \(\text{combine}(k_1, \text{list}(v_1)) \rightarrow v_2\)
- Usually same as the reduce function

Works only if reduce function is commutative and associative
Refinements: Partition Function

- Inputs to map tasks are created by contiguous splits of input file
- Reduce needs to ensure that records with the same intermediate key end up at the same worker
- System uses a default partition function:
  - hash(key) mod R
- Sometimes useful to override:
  - E.g., hash(hostname(URL)) mod R ensures URLs from a host end up in the same output file
Reading

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

Resources

• **Hadoop Wiki**
  - Introduction
  - Getting Started
  - Map/Reduce Overview
    • [http://wiki.apache.org/lucene-hadoop/HadoopMapReduce](http://wiki.apache.org/lucene-hadoop/HadoopMapReduce)
    • [http://wiki.apache.org/lucene-hadoop/HadoopMapRedClasses](http://wiki.apache.org/lucene-hadoop/HadoopMapRedClasses)
  - Eclipse Environment

• **Javadoc**
  - [http://lucene.apache.org/hadoop/docs/api/](http://lucene.apache.org/hadoop/docs/api/)

Jure Leskovec, Stanford C246: Mining Massive Datasets
Releases from Apache download mirrors

- http://www.apache.org/dyn/closer.cgi/lucene/hadoop/

Nightly builds of source


Source code from subversion

Further reading

- Programming model inspired by functional language primitives
- Partitioning/shuffling similar to many large-scale sorting systems
  - NOW-Sort ['97]
- Re-execution for fault tolerance
  - BAD-FS ['04] and TACC ['97]
- Locality optimization has parallels with Active Disks/Diamond work
  - Active Disks ['01], Diamond ['04]
- Backup tasks similar to Eager Scheduling in Charlotte system
  - Charlotte ['96]
- Dynamic load balancing solves similar problem as River's distributed queues
  - River ['99]