On

“MapReduce: Simplified Data Processing on Large Clusters”[1]

Christoph Pinkel

Data Processing

visitors.txt:

Julia
Anna
Rachel
Camilla
Lorna
Data Processing

visitors.txt:
Julia
Anna
Rachel
Camilla
Lorna

Julia.txt:
1272900910
1272910733
...

Rachel.txt:
1272410730
1272810771
...

Lorna.txt:
1272410811
1272910610
...
Data Processing
Data Processing
Data Processing
Large-scale Data Processing

- Very large data sets
  - Often not in DBMS
  - Distributed file system
  - Many disks/nodes
  - Several sources
  - Heterogeneous
  - Hard to process
Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process
Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process
Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process
Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process
Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process
Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process
Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process
Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process
Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process

May 05, 2010
Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process

SELECT url, COUNT(visits) FROM log

May 05, 2010
One Size Fits…?

- Traditional DBMS mantra: “One Size Fits All”
- DBMS won’t do (not even PDBMS)
- Need custom solutions
- Often based on FS type layer
One Size Fits…?

- Traditional DBMS mantra: “One Size Fits All”
- DBMS won’t do (not even PDBMS)
- Need custom solutions
- Often based on FS type layer
One Size Fits…?

- Traditional DBMS mantra: “One Size Fits All”
- DBMS won’t do (not even PDBMS)
  - Need custom solutions
  - Often based on FS type layer
Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
  - Distribution of data
  - Data parallelism
  - Fault tolerance
  - …
Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
  - Distribution of data
  - Data parallelism
  - Fault tolerance
  - ...
Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing

- Take care of...
  - Distribution of data
  - Data parallelism
  - Fault tolerance
  - ...
Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
  - Distribution of data
  - Data parallelism
  - Fault tolerance
  - ...

- 2 TB Weblog
- SELECT url, COUNT(visits)
- Split in parts
- Hash partition on URL
- Distribute parts
- On each: sort by URL
- Count visits
- Output partial results

May 05, 2010
Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
  - Distribution of data
  - Data parallelism
  - Fault tolerance
  - ...

- 2 TB Weblog
- SELECT url, COUNT(visits)
- Split in parts
- Hash partition on URL
- Distribute parts
- On each: sort by URL
- Count visits
- Output partial results

May 05, 2010
Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
  - Distribution of data
  - Data parallelism
  - Fault tolerance
  - ...

- 2 TB Weblog
- SELECT url, COUNT(visits)
- Split in parts
  - Hash partition on URL
- Distribute parts
- On each: sort by URL
- Count visits
- Output partial results
Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
  - Distribution of data
  - Data parallelism
  - Fault tolerance
  - ...

- 2 TB Weblog
- SELECT url, COUNT(visits)
- Split in parts
- Hash partition on URL
- **Distribute parts**
- On each: sort by URL
- Count visits
- Output partial results

May 05, 2010
Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
  - Distribution of data
  - Data parallelism
  - Fault tolerance
  - ...

- 2 TB Weblog
- SELECT url, COUNT(visits)
- Split in parts
- Hash partition on URL
- Distribute parts
  - On each: sort by URL
- Count visits
- Output partial results
Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
  - Distribution of data
  - Data parallelism
  - Fault tolerance
  - ...

- 2 TB Weblog
- SELECT url, COUNT(visits)
- Split in parts
- Hash partition on URL
- Distribute parts
- On each: sort by URL
  - Count visits
- Output partial results
Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of…
  - Distribution of data
  - Data parallelism
  - Fault tolerance
  - …

- 2 TB Weblog
- SELECT url, COUNT(visits)
- Split in parts
- Hash partition on URL
- Distribute parts
- On each: sort by URL
- Count visits
- Output partial results

May 05, 2010
A Programmer’s Nightmare

May 05, 2010
A Programmer’s Nightmare

Input

May 05, 2010
A Programmer’s Nightmare

May 05, 2010
A Programmer’s Nightmare

May 05, 2010
A Programmer’s Nightmare

May 05, 2010
A Programmer’s Nightmare

Input

Results
A Programmer’s Nightmare

Input

Results

May 05, 2010
A Programmer’s Nightmare

May 05, 2010
A Programmer’s Nightmare

May 05, 2010
Custom queries take tons of custom code
Outline

- MapReduce – Back to its Cradle
- What MapReduce is and What it’s Not
- The MapReduce Framework(s)
- Strengths and Weaknesses
- Summary
Google works a lot on large Web bound data
The Google World

May 05, 2010
The Google World
The Google World

May 05, 2010
The Google World

May 05, 2010
The Google World

uri1 dc:creator _:me.
uri1 dc:title "about".
uri2 rdf:type foaf:Document.
uri2 dc:creator _:you.
The Google World
The Google World

uri1 dc:creator _:me.
uri1 dc:title “about”.
uri2 rdf:type foaf:Document.
uri2 dc:creator _:you.
The Google Way

- Cheap commodity hardware
  - Huge number of nodes
  - Inexpensive disks
  - Commodity networking HW
  - High failure rates
- Specific needs
  - Work with very large data from the Web
- Build custom systems

Some Google Systems

- **GFS (Google File System)**[2]
  - Distributed file system

- **Bigtable**[3]
  - The structured data “special case”
  - Based on GFS

- Custom query programs
  - Originally hand-written
  - Using some libraries

- Often unstructured or semi-structured data

---


Some Google Systems

- **GFS (Google File System)**\(^2\)
  - Distributed file system

- **Bigtable**\(^3\)
  - The structured data “special case”
  - Based on GFS

- Custom query programs
  - Originally hand-written
  - Using some libraries

- Often unstructured or semi-structured data

---


\(^3\) Fay Chang et al.: “Bigtable: A Distributed Storage System for Structured Data” in *OSDI* 2006
Some Google Systems

- **GFS (Google File System)**[2]
  - Distributed file system
- **Bigtable**[3]
  - The structured data “special case”
  - Based on GFS

- Custom query programs
  - Originally hand-written
  - Using some libraries

- Often unstructured or semi-structured data

---


Some Google Systems

- GFS (Google File System)[2]
  - Distributed file system
- Bigtable[3]
  - The structured data “special case”
  - Based on GFS
- Custom query programs
  - Originally hand-written
  - Using some libraries

Often unstructured or semi-structured data

Custom queries take tons of custom code
Architectural Redundancy

May 05, 2010
Architectural Redundancy

Input

Results

May 05, 2010
Architectural Redundancy
Architectural Redundancy
Data Level Redundancy

- Takes records
  - one by one
  - key, value
- Processes records
  - Independently
- Outputs intermediate
  - 1..n per input record
  - key’, value’
Data Level Redundancy

- Takes records
  - one by one
  - key, value

- Processes records
  - Independently

- Outputs intermediate
  - 1..n per input record
  - key’, value’
Data Level Redundancy

- Takes records
  - one by one
  - key, value
- Processes records
  - Independently
- Outputs intermediate
  - 1..n per input record
  - key’, value’
Data Level Redundancy

- Takes records
  - one by one
  - key, value
- Processes records
  - Independently
- Outputs intermediate
  - 1..n per input record
  - key’, value’
Data Level Redundancy

- Takes records
  - one by one
  - key, value
- Processes records
  - Independently
- Outputs intermediate
  - 1..n per input record
  - key’, value’

May 05, 2010
Data Level Redundancy

- Takes records
  - one by one
  - key, value
- Processes records
  - Independently
- Outputs intermediate
  - 1..n per input record
  - key’, value’

- Takes intermediate
  - Groups with same key
  - key’, value’[]
- Processes records
- Outputs result
  - Per group
  - Any format
Data Level Redundancy

- Takes records
  - one by one
  - key, value
- Processes records
  - Independently
- Outputs intermediate
  - 1..n per input record
  - key’, value’

- Takes intermediate
  - Groups with same key
  - key’, value’[]
- Processes records
  - Group-wise
- Outputs result
  - Per group
  - Any format
Data Level Redundancy

- Takes records
  - one by one
  - key, value
- Processes records
  - Independently
- Outputs intermediate
  - 1..n per input record
  - key’, value’

- Takes intermediate
  - Groups with same key
  - key’, value’[]
- Processes records
  - Group-wise
- Outputs result
  - Per group
  - Any format
Data Level Redundancy

- **Map**
  - Takes records
    - one by one
    - key, value
  - Processes records
    - Independently
  - Outputs intermediate
    - 1..n per input record
    - key’, value’

- **Reduce**
  - Takes intermediate
    - Groups with same key
    - key’, value’[]
  - Processes records
    - Group-wise
  - Outputs result
    - Per group
    - Any format
Outline

- MapReduce – Back to its Cradle
- What MapReduce is and What it’s Not
- The MapReduce Framework(s)
- Strengths and Weaknesses
- Summary

May 05, 2010
What is MapReduce?
What is MapReduce?

Confused?
What is MapReduce?

- It **is** a framework
  - Though some people argue that it is not

- It **is** a programming paradigm
  - Though it is not really novel and rather trivial

- It **is partially defined by** its systems
  - Though it is not Hadoop (nor Google MR)

- It **is loosely defined**
  - Even in the original paper, and ever since

May 05, 2010
What is MapReduce?

- It *is* a framework
  - Though some people argue that it is not
- It *is* a programming paradigm
  - Though it is not really novel and rather trivial
- It is partially defined by its systems
  - Though it is not Hadoop (nor Google MR)
- It is loosely defined
  - Even in the original paper, and ever since

May 05, 2010
What is MapReduce?

- It is a framework
  - Though some people argue that it is not
- It is a programming paradigm
  - Though it is not really novel and rather trivial

- It is partially defined by its systems
  - Though it is not Hadoop (nor Google MR)

- It is loosely defined
  - Even in the original paper, and ever since

May 05, 2010
What is MapReduce?

- It is a framework
  - Though some people argue that it is not
- It is a programming paradigm
  - Though it is not really novel and rather trivial
- It is partially defined by its systems
  - Though it is not Hadoop (nor Google MR)
- It is loosely defined
  - Even in the original paper, and ever since
“Inspired by…”

- **map() & reduce()** in functional programming
- \((\text{map} \ (\lambda (x) \ (* \ x \ x)) \ '(1 \ 2 \ 3)) \rightarrow '(1 \ 4 \ 9)\)
- \((\text{reduce} + 0 \ '(1 \ 2 \ 3)) \rightarrow 6\)
- Very similar concepts
“Inspired by…”

- map() & reduce() in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))` → `(1 4 9)`
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts
“Inspired by…”

- `map()` & `reduce()` in functional programming
  
  ```lisp
(map (lambda (x) (* x x)) '(1 2 3))  
→ '(1 4 9)
```

- `(reduce + 0 '(1 2 3)) → 6`

- Very similar concepts
“Inspired by…”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3)) → '(1 4 9)
- `(reduce + 0 '(1 2 3)) → 6
- Very similar concepts
“Inspired by…”

- `map()` & `reduce()` in functional programming
  - `(map (lambda (x) (* x x)) '(1 2 3))` → '(1 4 9)
  - `(reduce + 0 '(1 2 3))` → 6
  - Very similar concepts
“Inspired by…”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))` → '(1 4 9)
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts
“Inspired by…”

- `map()` & `reduce()` in functional programming
  - `(map (lambda (x) (* x x)) '(1 2 3)) → '(1 4 9)
  - `(reduce + 0 '(1 2 3)) → 6
- Very similar concepts
“Inspired by…”

- **map() & reduce()** in functional programming
- \((\text{map}\ (\lambda x)(*\ x\ x))\ (1\ 2\ 3)) \rightarrow (1\ 4\ 9)\)
- \((\text{reduce}\ +\ 0\ (1\ 2\ 3))\) \rightarrow 6
- Very similar concepts
“Inspired by…”

- map() & reduce() in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))` → '(1 4 9)
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts
“Inspired by…”

- `map()` & `reduce()` in functional programming

- `(map (lambda (x) (* x x)) '(1 2 3)) → '(1 4 9)

- `(reduce + 0 '(1 2 3)) → 6

- Very similar concepts
“Inspired by…”

- map() & reduce() in functional programming
- \( (\text{map} (\lambda x (* x x)) \ (1 \ 2 \ 3)) \rightarrow (1 \ 4 \ 9) \)
- \( (\text{reduce} + 0 \ (1 \ 2 \ 3)) \rightarrow 6 \)
- Very similar concepts

“[MapReduce] is inspired by the map and reduce primitives present in Lisp”
(Dean/Ghemawat)\(^1\)
Programming Model

Inspired by Map/Reduce in functional programming languages, such as LISP from 1960's, but not equivalent.

* Slide taken from tutorial by Jerry Zhao and Jelena Pjesivac-Grovic (Google Inc.): "MapReduce – The Programming Model and Practice". Tutorial held at SIGMETRICS 2009.
Map & Reduce Elsewhere

- \( (\text{map (map-udf)} '((k1,v1) (k2,v2)) ) \rightarrow '((ik1,iv1) (ik2,iv2)) \)
- \( (\text{reduce (reduce-udf)} '((ik1,iv1) ...) \rightarrow \text{result} \)

- Concept present in basically all functional programming languages
- Implemented in other languages (Python)

May 05, 2010
Map & Reduce Elsewhere

- \((\text{map (map-udf) '((k1,v1) (k2,v2))}) \rightarrow '((ik1,iv1) (ik2,iv2))\)
- \((\text{reduce (reduce-udf) '((ik1,iv1) \ldots)} \rightarrow \text{result}\)

- Concept present in basically all functional programming languages
- Implemented in other languages (Python)
Map & Reduce Elsewhere

- \( (\text{map (map-udf)} \ (\text{'(k1,v1) '(k2,v2)})) \) \rightarrow \text{'(ik1,iv1) '(ik2,iv2)}

- \( (\text{reduce (reduce-udf)} \ (\text{'(ik1,iv1) ...})) \) \rightarrow \text{result}

- Concept present in basically all functional programming languages
- Implemented in other languages (Python)
Map & Reduce Elsewhere

- (map (map-udf) '('(k1,v1) '(k2,v2)) )
  \rightarrow '('(\textcolor{red}{ik1,iv1}) '(ik2,iv2))

- (reduce (reduce-udf) '('(\textcolor{red}{ik1,iv1}) ...))
  \rightarrow \text{result}

- Concept present in basically all functional programming languages
- Implemented in other languages (Python)
Map & Reduce Elsewhere

- $(\text{map (map-udf)} \ (\text{(k1,v1) (k2,v2)})) \rightarrow \ (\text{(ik1,iv1) (ik2,iv2)})$
- $(\text{reduce (reduce-udf)} \ (\text{(ik1,iv1) ...})) \rightarrow \text{result}$

- Concept present in basically all functional programming languages
- Implemented in other languages (Python)

May 05, 2010
Semantics have been analyzed\cite{4}

- Using Haskell to model
- Comparing with map and reduce in FP

MapReduce Semantics

- Semantics have been analyzed\(^4\)
  - Using Haskell to model
  - Comparing with map and reduce in FP

---

Semantics have been analyzed\[^4\]

Using Haskell to model

Comparing with map and reduce in FP

Google’s MapReduce is essentially a special case of map/reduce in FP

---

Outline

- MapReduce – Back to its Cradle
- What MapReduce is and What it’s Not
- The MapReduce Framework(s)
- Strengths and Weaknesses
- Summary

May 05, 2010
Architectural Details

- Split 1
- Split 2
- Split 3

Worker nodes connect to intermediate output, which then connects to worker nodes for reduce phase. Output files include file 1 and file 2.
Architectural Details

split 1
split 2
split 3

worker
worker
worker

file 1
file 2

input files
map phase
intermediate output
reduce phase
output files

read
write
read

May 05, 2010
Architectural Details

input files
split 1
split 2
split 3
read
map phase
worker
worker
worker
fork
fork
assign
write
read
worker
worker
fork
assign
write
output files
file 1
file 2
user program
master
reduce phase
intermediate output
intermediate output
Data Flow & Locality

split 1
split 2
split 3

worker
worker
worker
worker

file 1
file 2

May 05, 2010
Data Flow & Locality

split 1
split 2
split 3

file 1
file 2

worker
worker
worker
worker
worker
worker
Data Flow & Locality

split 1
split 2
split 3

file 1

worker

file 2

worker

May 05, 2010
Data Flow & Locality

split 1

split 2

split 3

file 1

file 2

map

map

map

worker

Reduce [1..10]

Reduce [11..20]

May 05, 2010
Combining

- Combiner instead of starting reducer early
- "Mini-reducer" in each map task
- Requires associative, cumulative reducer
- Might also reduce network traffic
Combining

- Combiner instead of starting reducer early
- “Mini-reducer” in each map task
- Requires associative, cumulative reducer
- Might also reduce network traffic

May 05, 2010
Combining

- Combiner instead of starting reducer early
- “Mini-reducer” in each map task
- Requires associative, cumulative reducer
- Might also reduce network traffic
Combining

- Combiner instead of starting reducer early
- "Mini-reducer" in each map task
- Requires associative, cumulative reducer
- Might also reduce network traffic
Combining

- Combiner instead of starting reducer early
- “Mini-reducer” in each map task
- Requires associative, cumulative reducer
- Might also reduce network traffic

➢ Early aggregation

May 05, 2010
Cluster Farming

- Balancing
  - Break job in small tasks
  - Schedule tasks as workers report idle

- Backup tasks
  - Scope with “stragglers” (slow workers)
  - “Speculative execution”
Cluster Farming

- **Balancing**
  - Break job in small tasks
  - Schedule tasks as workers report idle

- **Backup tasks**
  - Scope with “stragglers” (slow workers)
  - “Speculative execution”
Fault Tolerance

- Task failures
  - Just redo task (tasks are small)
  - Potentially on different machine

- Worker failures
  - Reallocate running tasks
  - Don’t schedule on worker anymore
  - What happens with intermediate output on that worker? (potentially re-schedule all)
Fault Tolerance

- **Task failures**
  - Just redo task (tasks are small)
  - Potentially on different machine

- **Worker failures**
  - Reallocate running tasks
  - Don’t schedule on worker anymore
  - What happens with intermediate output on that worker? (potentially re-schedule all)
Fault Tolerance Semantics

- Tasks are individual maps or reduces
  - Atomicity of operations
- Data level parallelism
  - Operations don’t interact
- Operations supposed to be deterministic
  - Repeated executions cause same output
- Side effect freeness
  - Generally no side effects (some exceptions)
Fault Tolerance Semantics

- Tasks are individual maps or reduces
  - Atomicity of operations
- Data level parallelism
  - Operations don’t interact
- Operations supposed to be deterministic
  - Repeated executions cause same output
- Side effect freeness
  - Generally no side effects (some exceptions)
Fault Tolerance Semantics

- Tasks are individual maps or reduces
  - Atomicity of operations
- Data level parallelism
  - Operations don’t interact
- Operations supposed to be deterministic
  - Repeated executions cause same output
- Side effect freeness
  - Generally no side effects (some exceptions)
Fault Tolerance Semantics

- Tasks are individual maps or reduces
  - Atomicity of operations
- Data level parallelism
  - Operations don’t interact
- Operations supposed to be deterministic
  - Repeated executions cause same output
- Side effect freeness
  - Generally no side effects (some exceptions)
Fault Tolerance Semantics

- Tasks are individual maps or reduces
  - Atomicity of operations
- Data level parallelism
  - Operations don’t interact
- Operations supposed to be deterministic
  - Repeated executions cause same output
- Side effect freeness
  - Generally no side effects (some exceptions)

- FT measures lead to same overall output
Implementations

- Google MapReduce
  - The original proposal, Google only

- Apache Hadoop
  - Open Source, used in academia

- Microsoft Dryad
  - Microsoft only, not exactly MapReduce

- Sector/Sphere\[5\]
  - Research prototype, not exactly MapReduce

---

\[5\] Robert Grossman and Yunhong Gu: “Data Mining Using High Performance Data Clouds: Experimental Studies Using Sector and Sphere” in *KDD 2008*
Implementations

- Google MapReduce
  - The original proposal, Google only

- Apache Hadoop
  - Open Source, used in academia

- Microsoft Dryad
  - Microsoft only, not exactly MapReduce

- Sector/Sphere\(^5\)
  - Research prototype, not exactly MapReduce

[\(^5\) Robert Grossman and Yunhong Gu: “Data Mining Using High Performance Data Clouds: Experimental Studies Using Sector and Sphere” in *KDD 2008*]
Implementations

- Google MapReduce
  - The original proposal, Google only
- Apache Hadoop
  - Open Source, used in academia
- Microsoft Dryad
  - Microsoft only, not exactly MapReduce
- Sector/Sphere\[5\]
  - Research prototype, not exactly MapReduce

\[5\] Robert Grossman and Yunhong Gu: “Data Mining Using High Performance Data Clouds: Experimental Studies Using Sector and Sphere” in *KDD 2008*
Implementations

- Google MapReduce
  - The original proposal, Google only
- Apache Hadoop
  - Open Source, used in academia
- Microsoft Dryad
  - Microsoft only, not exactly MapReduce
- Sector/Sphere\[^{5}\]
  - Research prototype, not exactly MapReduce

\[^{5}\] Robert Grossman and Yunhong Gu: “Data Mining Using High Performance Data Clouds: Experimental Studies Using Sector and Sphere” in *KDD 2008*
# Implementations

<table>
<thead>
<tr>
<th></th>
<th>Google MR</th>
<th>Hadoop</th>
<th>Dryad</th>
<th>Sector</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Availability</strong></td>
<td>Proprietary</td>
<td>Open Source</td>
<td>Proprietary</td>
<td>Open Source</td>
</tr>
<tr>
<td><strong>Used by</strong></td>
<td>Google</td>
<td>Research, Yahoo!, Facebook, Amazon (EC2!)</td>
<td>Microsoft</td>
<td>Research</td>
</tr>
<tr>
<td><strong>Implemented</strong></td>
<td>C++</td>
<td>Java</td>
<td>C++</td>
<td>C++</td>
</tr>
<tr>
<td><strong>Designed for</strong></td>
<td>Data center</td>
<td>Data center</td>
<td>Data center</td>
<td>Several data centers</td>
</tr>
</tbody>
</table>

May 05, 2010
Hadoop Terminology

May 05, 2010
Hadoop Terminology

- user program
- master
- split 1
- split 2
- split 3
- worker
- file 1
- file 2

May 05, 2010
Hadoop Terminology

- **split 1**
- **split 2**
- **split 3**

- **worker**
- **worker**
- **worker**
- **worker**

- **user program**
- **job tracker**

- **node**
- **node**
- **node**
- **node**

- **file 1**
- **file 2**

May 05, 2010
MapReduce in Use

- **Facebook**
  - 600 nodes cluster for warehouse
  - 2 PB, growing by 15 TB per day
  - Daily analyses, concurrent ad-hoc queries

- **Google**
  - Aug 2004: ~ 30,000 jobs, 217 machine years
  - Sep 2007: ~ 2 million jobs, 11,081 machine years

- **Yahoo!**
  - Using for web services
  - Won TeraSort contest in 2008 with Hadoop cluster

- **Amazon**
  - MapReduce on EC2

*May 05, 2010*
MapReduce in Use

- **Facebook**
  - 600 nodes cluster for warehouse
  - 2 PB, growing by 15 TB per day
  - Daily analyses, concurrent ad-hoc queries

- **Google**
  - Aug 2004: ~30,000 jobs, 217 machine years
  - Sep 2007: ~2 million jobs, 11,081 machine years

- **Yahoo!**
  - Using for web services
  - Won TeraSort contest in 2008 with Hadoop cluster

- **Amazon**
  - MapReduce on EC2

May 05, 2010
MapReduce in Use

- **Facebook**
  - 600 nodes cluster for warehouse
  - 2 PB, growing by 15 TB per day
  - Daily analyses, concurrent ad-hoc queries

- **Google**
  - Aug 2004: ~ 30,000 jobs, 217 machine years
  - Sep 2007: ~ 2 million jobs, 11,081 machine years

- **Yahoo!**
  - Using for web services
  - Won TeraSort contest in 2008 with Hadoop cluster

- **Amazon**
  - MapReduce on EC2
MapReduce in Use

- **Facebook**
  - 600 nodes cluster for warehouse
  - 2 PB, growing by 15 TB per day
  - Daily analyses, concurrent ad-hoc queries

- **Google**
  - Aug 2004: ~ 30,000 jobs, 217 machine years
  - Sep 2007: ~ 2 million jobs, 11,081 machine years

- **Yahoo!**
  - Using for web services
  - Won TeraSort contest in 2008 with Hadoop cluster

- **Amazon**
  - MapReduce on EC2
Outline

- MapReduce – Back to its Cradle
- What MapReduce is and What it’s Not
- The MapReduce Framework(s)
- Strengths and Weaknesses
- Summary
Clever Recombination

- map & reduce from functional programming
- Applied for distributed systems
- Simple, intuitive interface

May 05, 2010
Clever Recombination

- map & reduce from functional programming
- Applied for distributed systems
- Simple, intuitive interface

Highly useful system for large-scale data processing needs
Impact

- Inspired a lot of scientific publications
  - Extending the model or framework
  - Trying to combine with other techniques

- Impact on Industry
  - Solves actual problems
  - Used by many companies
Impact

- Inspired a lot of scientific publications
  - Extending the model or framework
  - Trying to combine with other techniques

Impact on Industry
- Solves actual problems
- Used by many companies
Not Really New

- Programming model is not new
  - Functional programming

- Distributed systems are not novel
  - Well…

- Fault tolerance, balancing, etc.
  - Studied in various fields (especially DS)
Not Really New

- Programming model is not new
  - Functional programming
- Distributed systems are not novel
  - Well…
- Fault tolerance, balancing, etc.
  - Studied in various fields (especially DS)
Not Really New

- Programming model is not new
  - Functional programming
- Distributed systems are not novel
  - Well…
  - Fault tolerance, balancing, etc.
    - Studied in various fields (especially DS)
Not Really New

- Programming model is not new
  - Functional programming
- Distributed systems are not novel
  - Well…
- Fault tolerance, balancing, etc.
  - Studied in various fields (especially DS)

Could be considered pure engineering
Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact
Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

May 05, 2010
Summary

 It is hard to process very large datasets
 Even harder with non-homogeneous data
 Need massive parallelism
 Hard to implement case-by-case
 MapReduce: parallelization framework
 Uses FP concepts
 Simple and elegant solution
 Huge impact
Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

May 05, 2010
Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- **MapReduce:** parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

May 05, 2010
Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
  - Uses FP concepts
  - Simple and elegant solution
  - Huge impact
Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact
Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact
Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

Thank you! Questions?

May 05, 2010
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of jobs</td>
<td>29,000</td>
<td>171,000</td>
<td>2,217,000</td>
</tr>
<tr>
<td>Avg. runtime [sec]</td>
<td>634</td>
<td>874</td>
<td>395</td>
</tr>
<tr>
<td>Total machine years</td>
<td>217</td>
<td>2,002</td>
<td>11,081</td>
</tr>
<tr>
<td>Map input [TB]</td>
<td>3,288</td>
<td>52,254</td>
<td>403,152</td>
</tr>
<tr>
<td>Intermediate (map) output [TB]</td>
<td>758</td>
<td>6,743</td>
<td>34,774</td>
</tr>
<tr>
<td>Final (reduce) output [TB]</td>
<td>193</td>
<td>2,970</td>
<td>14,018</td>
</tr>
<tr>
<td>Machines per job [avg]</td>
<td>157</td>
<td>268</td>
<td>394</td>
</tr>
<tr>
<td>Unique mappers</td>
<td>395</td>
<td>1958</td>
<td>4083</td>
</tr>
<tr>
<td>Unique reducers</td>
<td>269</td>
<td>1208</td>
<td>2418</td>
</tr>
</tbody>
</table>