

# Hadoop@Brown



**2950-T: Topics in Databases and Systems**  
**Andy Pavlo**  
**September 17, 2008**

# Outline

- MapReduce Overview
- Hadoop Overview
- Installing Hadoop
- Demos
- Advanced Topics

# Overview

- MapReduce is a software framework/programming model for parallel computations on large data sets using shared-nothing, commodity hardware clusters.
- Developers write applications that provide two basic operations on input data sets:
  - **Map**(key1, value1) -> list(key2, value2)
  - **Reduce**(key2, list(value2)) -> list(value2)
- **Key point:** By abstracting out the semantics of the distributed framework, users are not encumbered with the details of how to parallelize their programs.

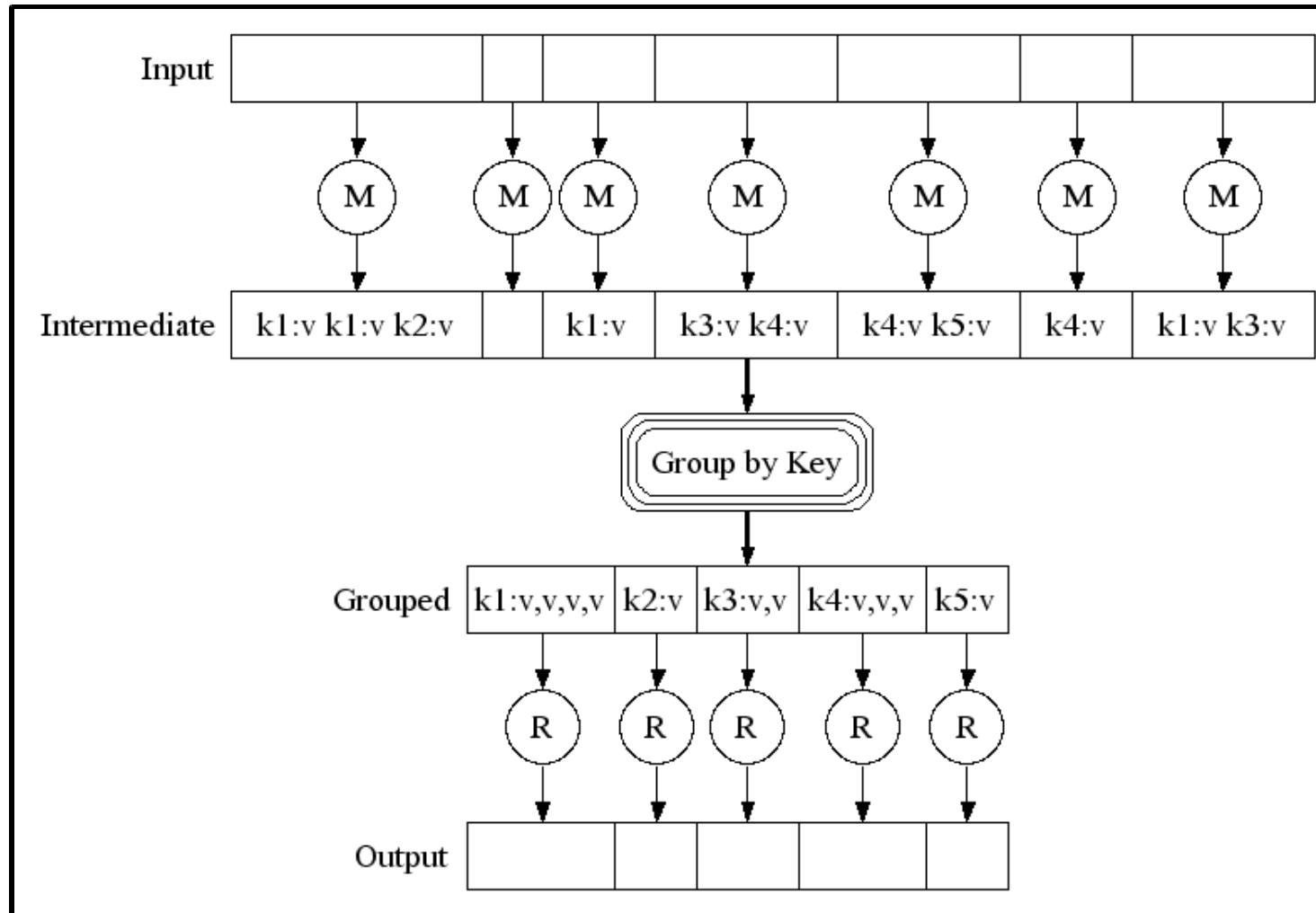
# Map

- Given a key/value pair, the Map function applies some processing function and produces a new key/value pair.
- Multiple instances of the Map are executed on multiple machines for unique subsets of the input data.
- The output of map functions are partitioned into buckets based on the output keys.
- There does not need to be a one-to-one correspondence to input key/value pairs to output key/value pairs.

# Reduce

- Like Map, the Reduce function is distributed on multiple nodes and operates on a unique subset of Map's output.
- Each Reduce task receives a key and a list of all the values that were created for that key from all the Map function executions:
  - **Shuffle:** fetch key/values outputted by all Map tasks
  - **Sort:** sort the fetched key/values into buckets
  - **Reduce:** process key/values and outputs new key/value pairs.
- The Reduce operation is optional in a MR application.

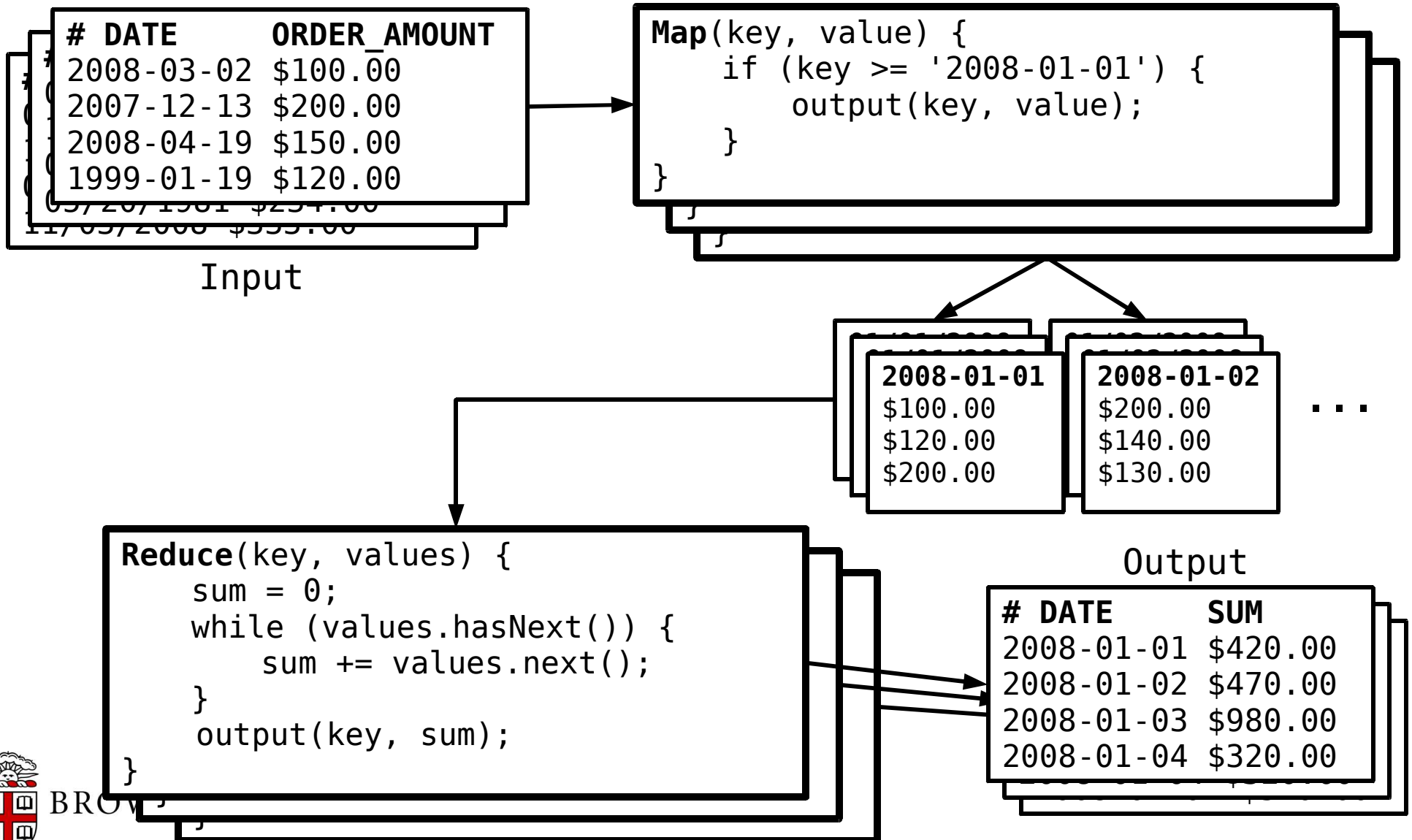
# Data Flow



Source: Brendan Melville, Lehigh University

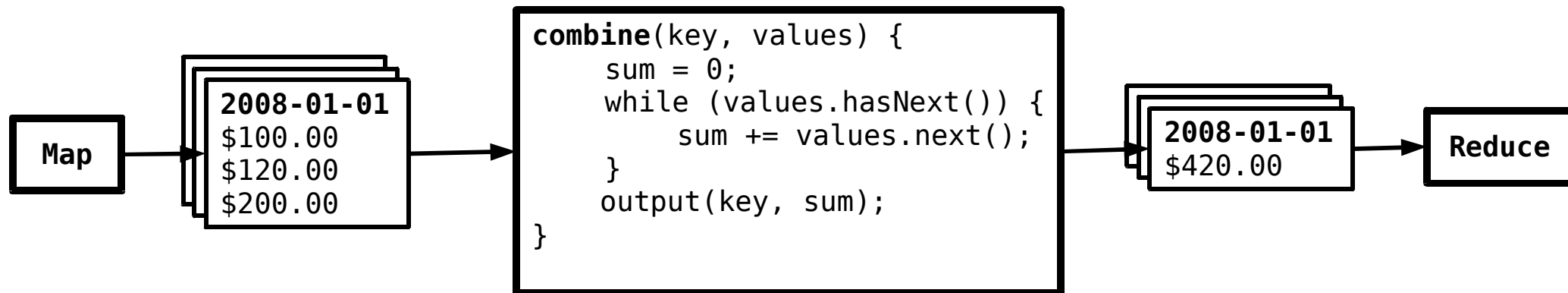
# Simple Example

- Get the total order amount for each day after 2008-01-01



# Combine

- Special intermediate Reduce task to combine duplicate keys produced from an individual Map task:
  - Map -> Combiner -> Reduce
  - Executes on the same node as its corresponding Map task.
- Order Sum Example:
  - Each Map task may produce many records for each date.
  - Rather than sending large list of values, we can perform an intermediate sum after each Map instance.



# Hadoop

- Yahoo/Apache's open-source implementation of the MapReduce model:
  - Core framework: Java
  - Native libraries: C/C++
  - MR applications can be written in any language.
- Provides many of Google's known middleware:
  - MapReduce -> Hadoop
  - GFS -> HDFS
  - BigTable -> Hbase

# Hadoop

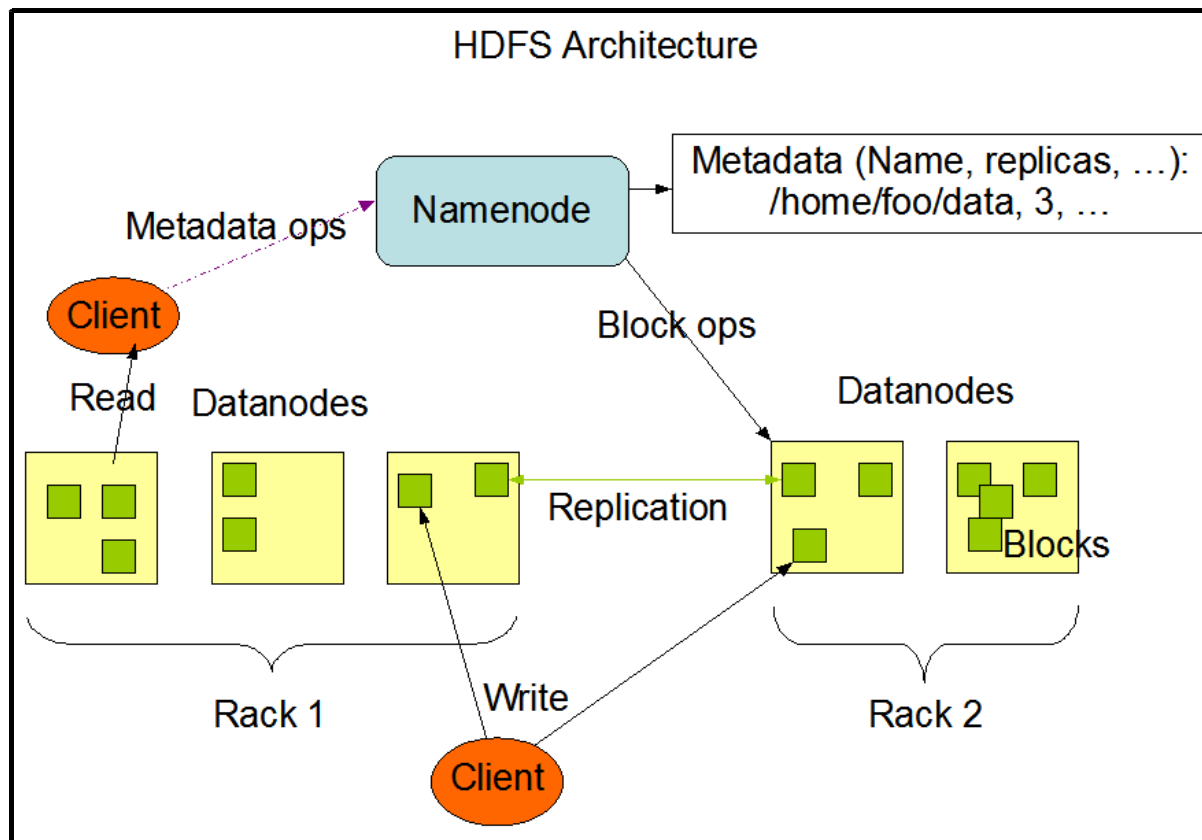
- As of July 2008, Hadoop holds the record for Jim Gray's Daytona TeraByte Sort Benchmark:
  - $10^{10}$  100-byte records in 3.48 minutes
- Notable Hadoop users:
  - Yahoo
  - Amazon
  - Last.fm
  - New York Times
  - Facebook

# Architecture

- MapReduce engine:
  - Central job tracker (master) for coordinating application execution
  - Separate task tracker on each forks off MR tasks (slaves).
- Distributed filesystem:
  - Central filesystem name node (master) handles file replication, deletion, and creation.
  - Separate data node daemons to serve files (slaves).

# HDFS

- Java user-space distributed block-level filesystem.
- Data durability is achieved by replicating blocks on multiple nodes (default = 3).
- Command-line tool for managing data.



Source: Hadoop, The Apache Software Foundation.

# Fault Tolerance

- Map/Reduce Tasks:
  - Map/Reduce tasks are pinged periodically.
  - Unresponsive tasks become eligible for re-execution.
  - Slow tasks are speculatively re-executed on available nodes.
- HDFS Name Node & Job Tracker:
  - Currently can only fail over to a backup daemons.

# Installing Hadoop

- Download and untar Hadoop:
  - Install global libraries/config files in NFS directory.
  - Use scratch space for individual nodes.
- Set environment variables:
  - **HADOOP\_HOME**: `$HOME/Programs/hadoop`
  - **HADOOP\_CONF\_DIR**: `$HADOOP_HOME/conf`
  - **HADOOP\_LOG\_DIR**: `/tmp/$USER/hadoop/logs`
  - **JAVA\_HOME**: `/usr`
- Edit node listing files:
  - `$HADOOP_CONF_DIR/masters`
  - `$HADOOP_CONF_DIR/slaves`
- Edit XML configuration files (refer to documentation):

 `$HADOOP_CONF_DIR/hadoop-site.xml`

# Installing Hadoop

- Prepare execution nodes local directories:
  - `ssh slave-node "mkdir -p $HADOOP_LOG_DIR"`
- Format HDFS:
  - `$HADOOP_HOME/bin/hadoop namenode -format`
- Start HDFS name node:
  - `ssh master-host "start-dfs.sh"`
  - `http://master-host.cs.brown.edu:50070/`
- Start job tracker:
  - `ssh master-host "start-mapred.sh"`
  - `http://master-host.cs.brown.edu:50030/`

# Writing a MapReduce Application

- Create Mapper and Reducer objects that implement the proper interfaces provided by the Hadoop library.
- Create a JobConf object and set the appropriate parameters:
  - Map/Reduce Classes
  - Input/Output Paths
  - Input/Output Data Types
- Call the run job method.

# Demo

- Get the total order amount for each day after 2008-01-01
  - Load input data into HDFS
  - Write Map/Reduce methods
  - Create execution jar
  - Deploy in Hadoop
  - View/download results
- Refinements:
  - Single output file
  - Runtime configuration parameters
  - Combine intermediate output

# Troubleshooting

- Debugging MapReduce applications:
  - Run in standalone mode.
  - Custom scripts to process failed task logs at runtime
- Blocked by HDFS Safe-Mode:
  - Decrease safe-mode wait time (`dfs.safemode.extension`)
  - Decrease replication threshold (`dfs.safemode.threshold.pct`)
- “Cannot allocate memory” errors:
  - Decrease the default JVM heap size (default is 1024MB)
  - Change `HADOOP_HEAPSIZE` parameter in `$HADOOP_CONF_DIR/hadoop-env.sh`
  - Enable Linux memory over-commit:
    - `sysctl vm.overcommit_memory`
    - `sysctl vm.overcommit_ratio`
    - Not possible on CS department machines

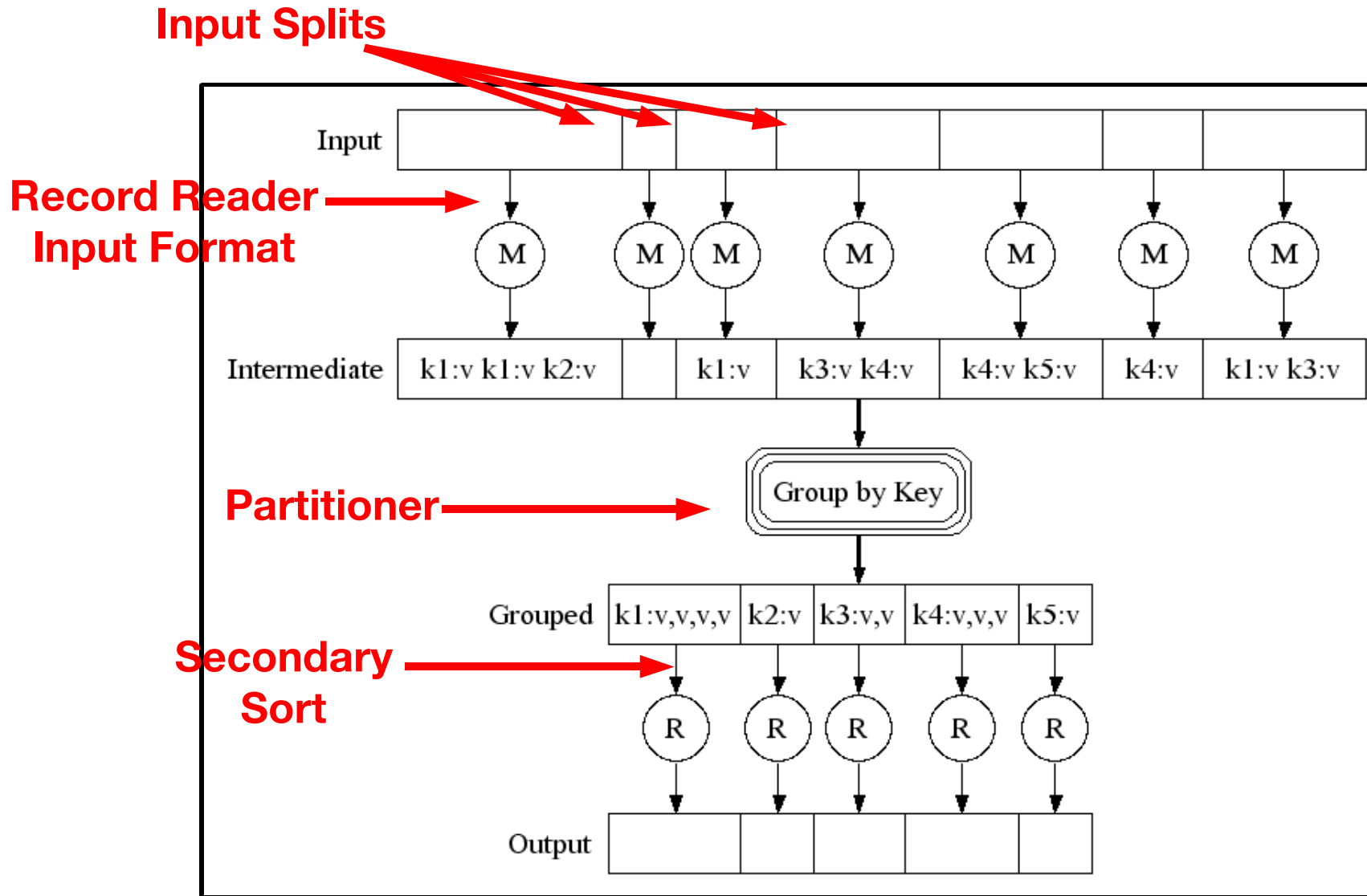
# Advanced Topics

- Input File Formats
- Custom Partitioning/Splitting
- Compression
- Optimizations
- Additional techniques and technologies

# Input File Formats

- The InputFileFormat of a job defines how the input data files will be interpreted by Hadoop, partitioned, and transmitted to Map tasks.
- TextInputFiles (default):
  - Plain text, human-readable, Text type only
  - Keys => line #, Value => line contents (split by newline)
- SequenceFile:
  - Flat files, binary key/value pairs
  - Used as intermediate output from Map tasks
- KeyValueTextInputFormat:
  - Plain text, human readable, parameterized types
  - Key/Value pairs are each line split by tab/delimiter byte

# Data Flow



Source: Brendan Melville, Lehigh University

# Custom Input Handlers

- **InputSplit/RecordReader:**
  - Splits input data into logical partitions and feeds records to the Map task.
  - Example: Map task generates data. Use InputSplit to tell how many tasks and how many records each task should create.
- **InputFormat:**
  - Split the records in input data files using a custom scheme.
  - Example: Single-line of data, key=>10-bytes value=>90-bytes
  - Example: Split value into multiple attributes
- **Partitioner:**
  - Default is to use `Object.hashCode()`, may not always be the best
  - Example: Divide Map output by ranges so that Reducer output files are in a globally sorted order.

# Compression

- All files in HDFS can be compressed to speed up data transfer rates:
  - Block-level compression
  - Record-level compression
  - Only SequenceFiles can be compressed
- Hadoop makes JNI calls into native implementations of various compression codecs (e.g., zlib, gzip, lzo)
- Data throughput vs. CPU performance

# Optimizations

- How many map tasks?
  - The number of maps is usually determined by the total number of blocks of the input files.
  - Overhead of job start-up vs. operational costs
- Distributed cache
  - Deploy static data files and libraries to all nodes before map tasks begin executing.
- Rack awareness
  - Allows HDFS to store the majority of a particular block's replicas on the same rack as the machine where it was created.
  - Tasks will try to fetch blocks from data nodes in the same rack

# Additional Hadoop Techniques

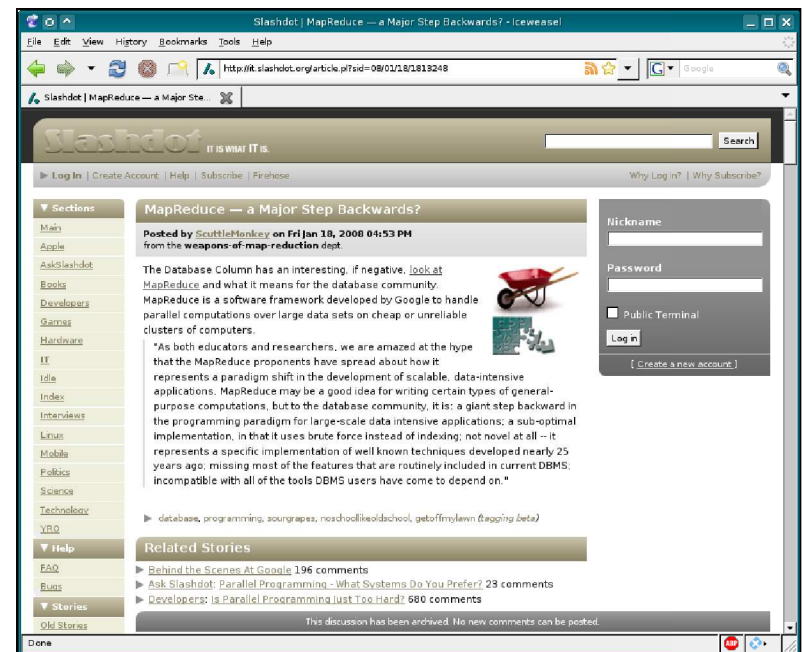
- Hadoop Streaming:
  - Utility for executing Map/Reduce jobs using any executable:
  - `hadoop jar $HADOOP_HOME/hadoop-streaming.jar \`
    - `-input myInputDirs \`
    - `-output myOutputDir \`
    - `-mapper /bin/cat \`
    - `-reducer /bin/wc`
- IBM's Hadoop Eclipse Plugin:
  - Wizards for creating MapReduce classes.
  - Built-in views of Hadoop servers.

# Additional Hadoop Technologies

- Hbase:
  - Open-source BigTable operation built on top of Hadoop/HDFS
  - Hbase was broken out from the Hadoop project this Spring
  - Does not support SQL!
- Pig/Pig Latin:
  - Dataflow programming environment/language for generating large-scale processing data flows.
  - Supports relational-algebra operations (join, filter, project) and the traditional functional-programming operations (map, reduce)
  - The Pig Latin code gets translated into Hadoop Map/Reduce operations.

# Final Thoughts

- Hadoop is simple and effective framework for processing large amounts of data:
  - Easy to install and deploy applications.
- Some universities are teaching Hadoop to students in first year CS courses. Google is providing cluster time.
- Hadoop vs. DBMS:
  - Some argue that people are using MapReduce to perform tasks that are best suited by a traditional database system.
  - Jury is still out...



MapReduce: A major step backwards  
January 18th, 2008