Meet Hadoop!
Open Source Grid Computing

Devaraj Das

Yahoo! Inc. Bangalore & Apache Software Foundation

ddas@apache.org

Hadoop: Why?

- Need to process 100TB datasets
- On 1 node:
  – scanning @ 50MB/s = 23 days
- On 1000 node cluster:
  – scanning @ 50MB/s = 33 min
- Need Efficient, Reliable and Usable framework

Hadoop: Where?

- Batch data processing, not real-time / user facing
  - Being applied to the back-end of web search
    - Log Processing
    - Document Analysis and Indexing
    - Web Graphs and Crawling
- Highly parallel data intensive distributed applications
  - Bandwidth to data is a significant design driver
  - Number of CPUs that can be applied gates what you can do
- Very large production deployments (GRID)
  - Several clusters of 1000s of nodes
  - LOTS of data (Trillions of records, 100 TB+ data sets)

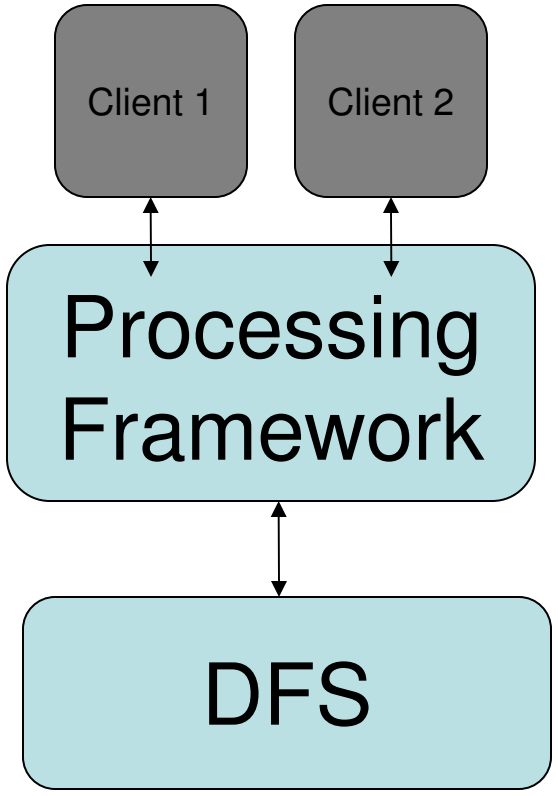
Hadoop in Open Source

- Apache Lucene sub-project, led by Doug Cutting
- Factored out of Nutch (web-search software)

- My role
  - ~1.5 years in the project
  - Contributed primarily to the MapReduce component
  - Hadoop Committer

Hadoop: Components

- **Distributed File System**
  - Modeled on GFS
- Distributed Processing Framework
  - Using **Map**/**Reduce** metaphor



hadoop

HDFS - Hadoop Distributed FS

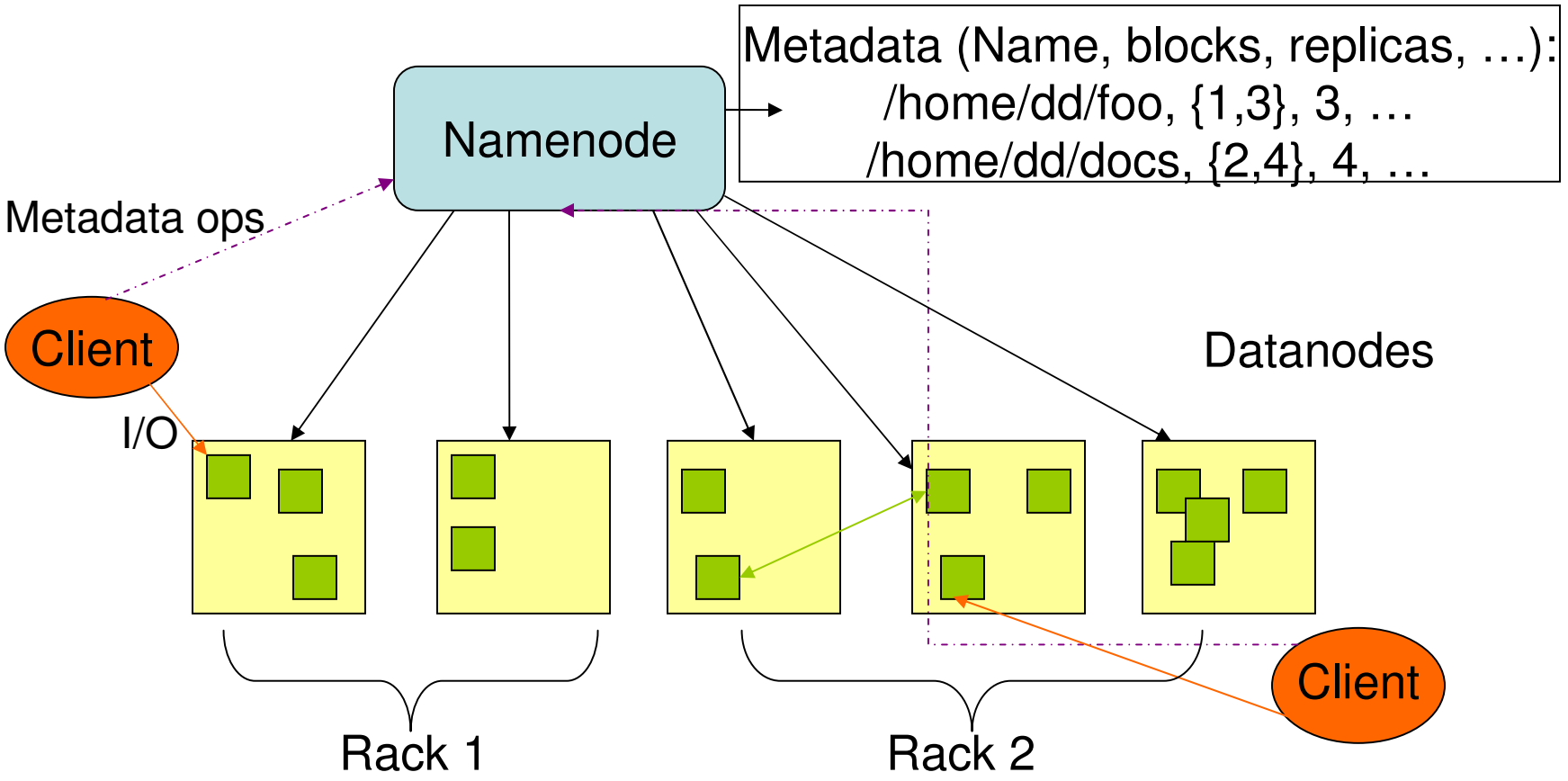
- Distributed storage system
  - Files are divided into **large** blocks and distributed across the cluster
  - Blocks replicated to handle hardware failure
  - Data placement exposed so that computes can be migrated to data

- Notable differences from mainstream DFS work
  - Single ‘storage + compute’ cluster vs. Separate clusters
  - Simple I/O centric API vs. Attempts at POSIX replication

HDFS Architecture

- Master-Slave Architecture
- HDFS Master “Namenode”
  - Manages all filesystem metadata
    - Transactions are logged, merged at startup
  - Controls read/write access to files
  - Manages block replication
- HDFS Slaves “Datanodes”
  - Notifies NameNode about block-IDs it has
  - Serve read/write requests from clients
  - Perform replication tasks upon instruction by namenode
  - Rack-aware

HDFS Architecture



hadoop

HDFS: Handling Failures

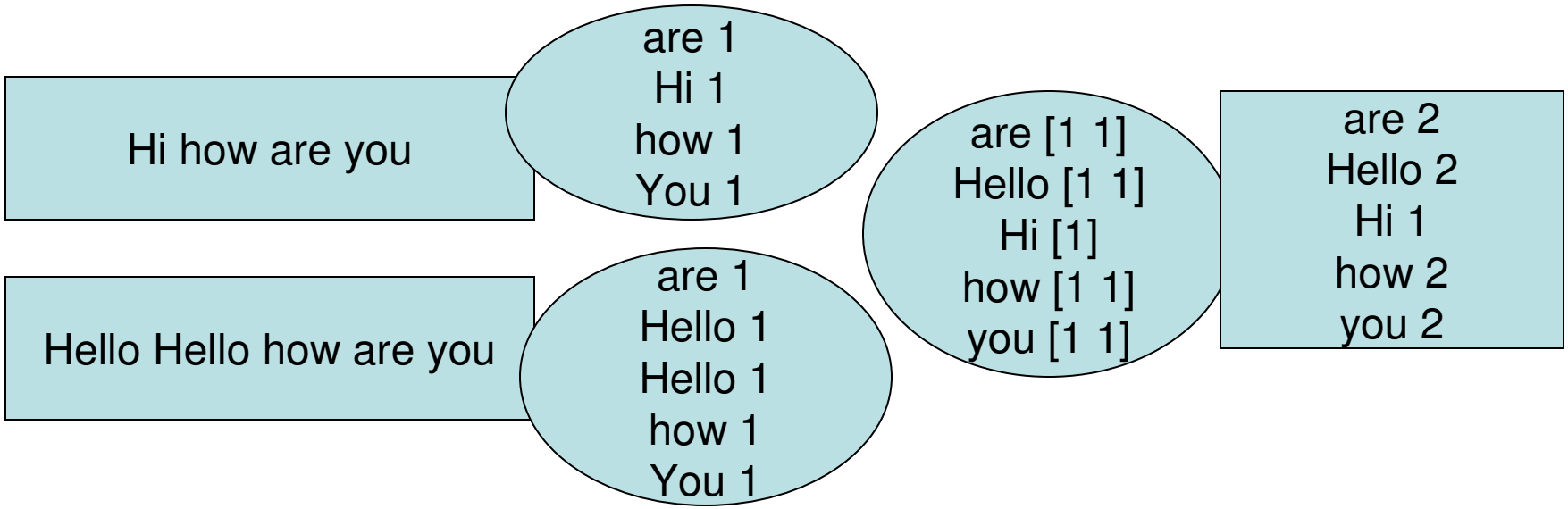
- NameNode failure
  - A single point of failure
- Secondary NameNode provides consistency semantics
  - Copies FsImage and Transaction Log from NameNode & merges them
  - Uploads new FSImage to the NameNode
- DataNode failures

HDFS: Data Correctness

- Use Checksums to validate data
  - Use CRC32
- File Creation
  - Client computes checksum per 512 byte
  - DataNode stores the checksum
- File access
  - Client retrieves the data and checksum from DataNode
  - If Validation fails, Client tries other replicas
- fsck

Distributed Processing

Wordcount on a huge file

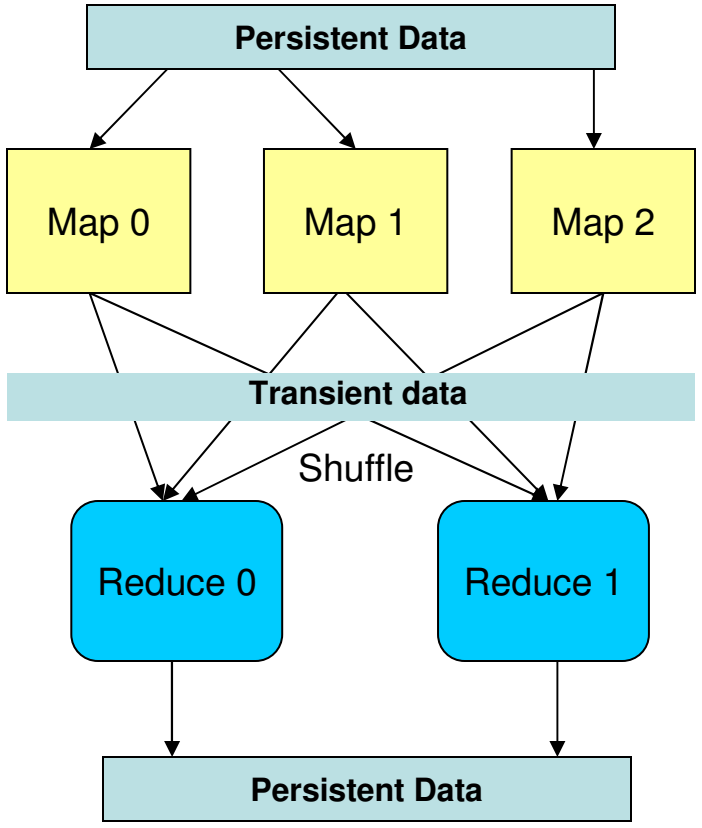


Common pattern for things like Log Processing, Statistics, Index creation, **Search Engines**!

hadoop

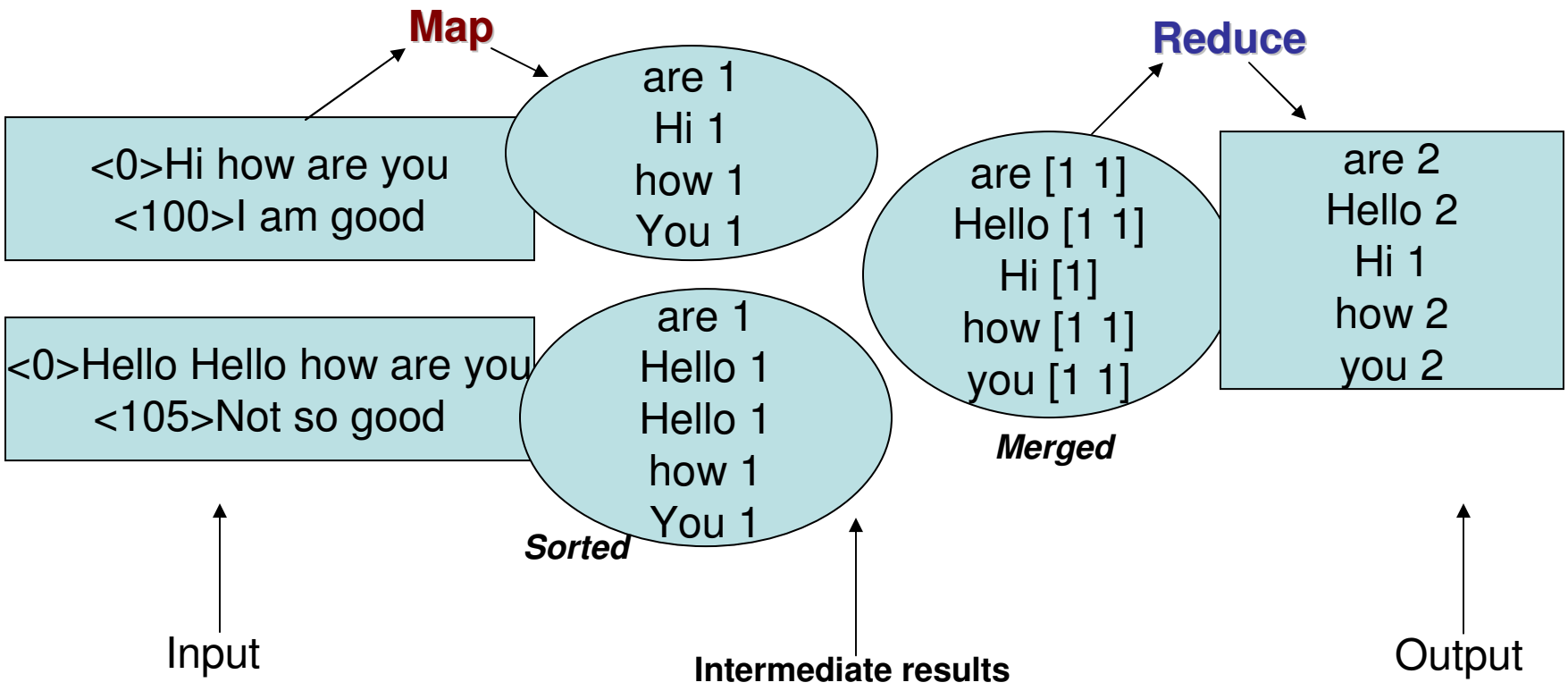
MapReduce: Data Flow

- User jobs are broken into Map tasks and Reduce tasks
- Data is sequence of *keys* and *values*
- Map Task: invokes Mapper
  - Input: *key1,value1* pair
  - Output: *key2, value2* pairs
- Reduce Task: invokes Reducer
  - Called once per a key, in sorted order
  - Input: *key2*, stream of *value2*
  - Output: *key3, value3* pairs



hadoop

Data: Stream of keys and values



wordcount

```
public void map(WritableComparable key, Writable value, OutputCollector output, Reporter reporter) throws
      IOException {

   String line = ((UTF8)value).toString();

   StringTokenizer itr = new StringTokenizer(line);

   while (itr.hasMoreTokens()) {
           word.set(itr.nextToken());
            output.collect(word, one);
   }
}


public void reduce(WritableComparable key, Iterator values, OutputCollector output,Reporter reporter) throws
      IOException {
   int sum = 0;

   while (values.hasNext()) {
      sum += ((IntWritable) values.next()).get();
    }

    output.collect(key, new IntWritable(sum));
}
```

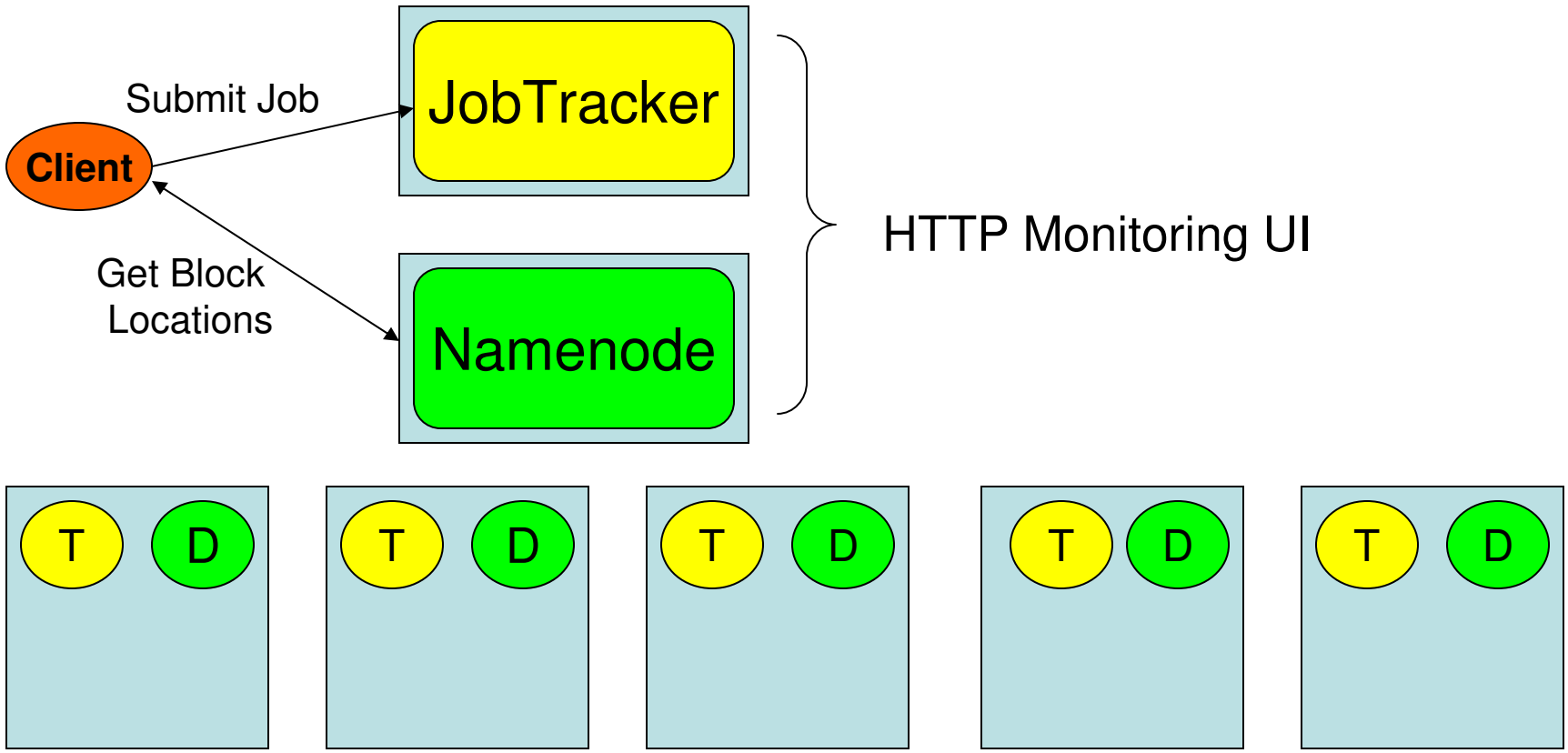
Hadoop Map-Reduce Architecture

- **Master-Slave architecture**
- **Map-Reduce Master "Jobtracker"**
  - Accepts MR jobs submitted by users
  - Assigns Map and Reduce tasks to Tasktrackers
  - Monitors task and tasktracker status, re-executes tasks upon failure
- **Map-Reduce Slaves "Tasktrackers"**
  - Run Map and Reduce tasks upon instruction from the Jobtracker
  - Manage storage and transmission of intermediate output
- **Generic Reusable Framework supporting pluggable user code**
  - Pluggable FileSystem - DFS, Kosmix, S3, ..
  - Pluggable input/output format
  - Many more

MapReduce: Client

- Define Mapper and Reducer classes and a "launching" program.
- Language support
  - Java, C++
  - Streaming model
- Special case – Maps for parallelizing only

Hadoop HDFS + MR cluster



Machines with Datanodes and Tasktrackers

Map/Reduce Optimizations

- Combiner
  - Mappers may generate duplicate keys
  - Side-effect free reducer run on mapper node
  - Minimize data size before transfer
  - Reducer is still run
- Speculative execution
  - Some nodes may be slower
  - Run duplicate task on another node
- Out-of-band data access
  - Distributed Cache

Other significant projects

- Jute
  - A way to bridge legacy data into map-reducible data
  - Versioning of data
- HOD
  - For effective cluster sharing
  - Based on Open Source Schedulers like Torque
- HBASE
  - Loosely equivalent to Google's BigTable

Scalability/Deployment

- Tested on 2000 nodes at Yahoo!
- 20TB sort on 2000 nodes takes ~2 hours
- Distributed File System in daily use:
  - 1.5 PB (replication: 3)
  - Millions of files
- Also runs on Amazon's EC2/S3

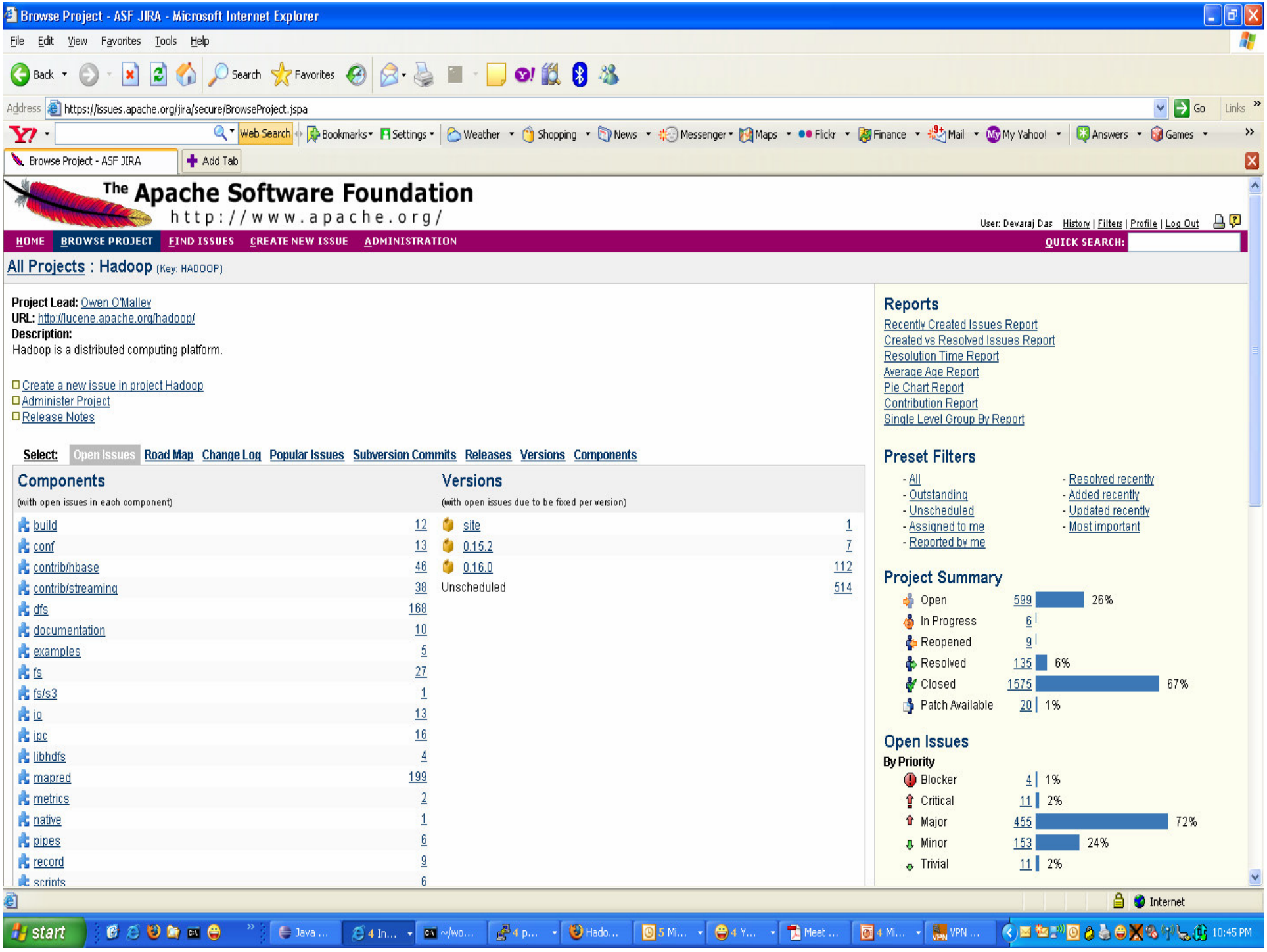
By the way
Yahoo! is one of the main contributors to Hadoop both in terms of Development as well as Adoption

Some issues addressed

- NameNode's memory usage optimizations
  - Examples - CRC, data structure optimizations
  - Switch from Threaded Server to NIO Server
- Huge transaction log
- Sort scalability/performance
- Data Compression

Things worked on

- Append/Truncate
- Support for users and permissions in HDFS
- Rack awareness in MapReduce
- 100% CPU, Network utilization
- JobTracker failover
- Checkpointing of jobs
- Debugging/Profiling apps


Browse Project - ASF JIRA - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address https://issues.apache.org/jira/secure/BrowseProject.jspa
Go Links
Web Search Bookmarks Settings Weather Shopping News Messenger Maps Flickr Finance Mail My Yahoo! Answers Games
Browse Project - ASF JIRA
Add Tab
The Apache Software Foundation
http://www.apache.org/
User: Devaraj Das History | Filters | Profile | Log Out
HOME BROWSE PROJECT FIND ISSUES CREATE NEW ISSUE ADMINISTRATION
QUICK SEARCH:
All Projects : Hadoop (Key: HADOOP)
Project Lead: Owen O'Malley
URL: http://lucene.apache.org/hadoop/
Description:
Hadoop is a distributed computing platform.
Create a new issue in project Hadoop
Administer Project
Release Notes
Select: Open Issues Road Map Change Log Popular Issues Subversion Commits Releases Versions Components
Components
(with open issues in each component)
build 12
conf 13
contrib/hbase 46
contrib/streaming 38
dfs 168
documentation 10
examples 5
fs 27
fs/s3 1
io 13
ipc 16
libhdfs 4
mapred 199
metrics 2
native 1
pipes 6
record 9
scripts 6
Versions
(with open issues due to be fixed per version)
site 1
0.15.2 7
0.16.0 112
Unscheduled 514
Reports
Recently Created Issues Report
Created vs Resolved Issues Report
Resolution Time Report
Average Age Report
Pie Chart Report
Contribution Report
Single Level Group By Report
Preset Filters
- All
- Outstanding
- Unscheduled
- Assigned to me
- Reported by me
- Resolved recently
- Added recently
- Updated recently
- Most important
Project Summary
Open 599 26%
In Progress 6
Reopened 9
Resolved 135 6%
Closed 1575 67%
Patch Available 20 1%
Open Issues
By Priority
Blocker 4 1%
Critical 11 2%
Major 455 72%
Minor 153 24%
Trivial 11 2%
Internet
start
10:45 PM

Thank You!

We need your help!

- **Contribute to Hadoop's development!**

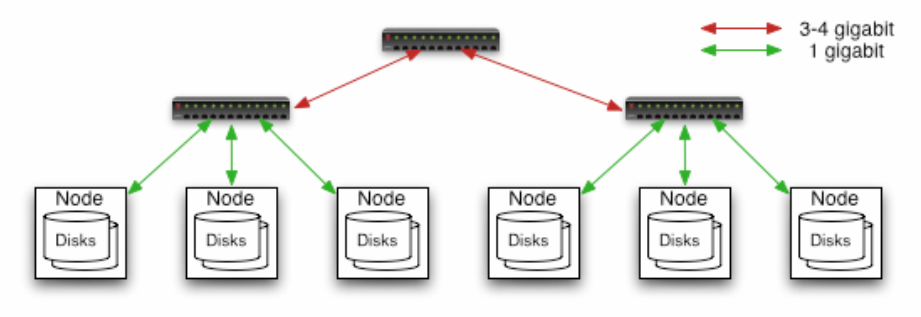
For more information:

- **http://lucene.apache.org/hadoop/**

ddas@yahoo-inc.com

Back-up slides follow

Commodity Hardware Cluster



Typically in 2 level architecture
– Nodes are commodity PCs
– 30-40 nodes/rack
– Uplink from rack is 3-4 gigabit
– Rack-internal is 1 gigabit

hadoop

HDFS: File Creation

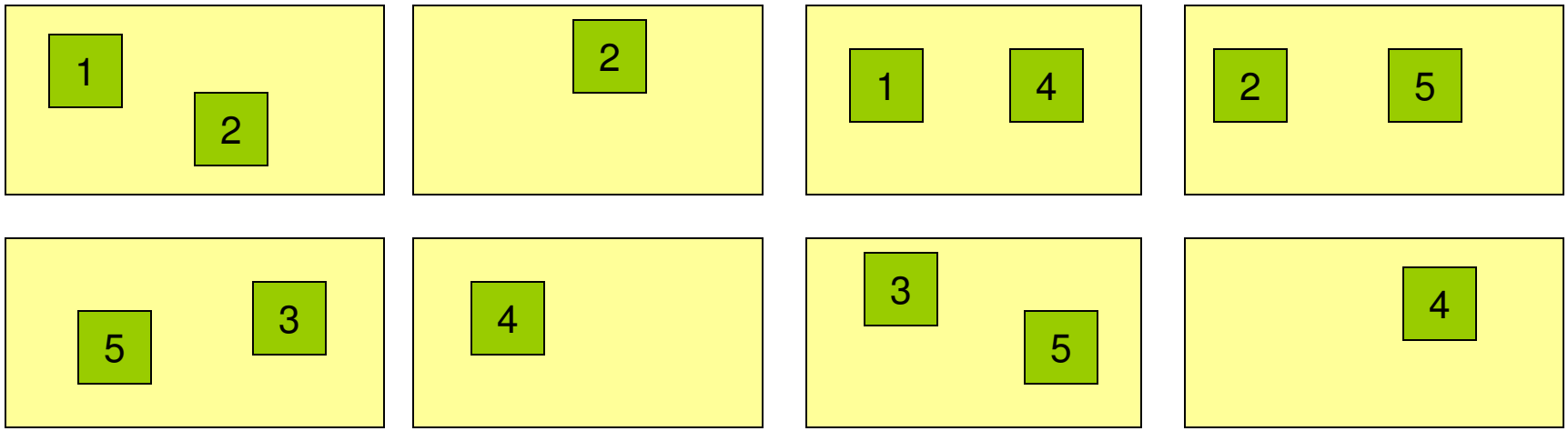
- Client retrieves a list of DataNodes on which to place replicas of a block
- Client writes block to the first DataNode
- The first DataNode forwards the data to the next DataNode in the Pipeline
- When all replicas are written, the Client moves on to the next block in file

HDFS - Master and Slaves

Namenode (the master)

name:/users/joeYahoo/myFile - copies:2, blocks:{1,3}
name:/users/bobYahoo/someData.gzip, copies:3, blocks:{2,4,5}

Datanodes (the slaves)



hadoop

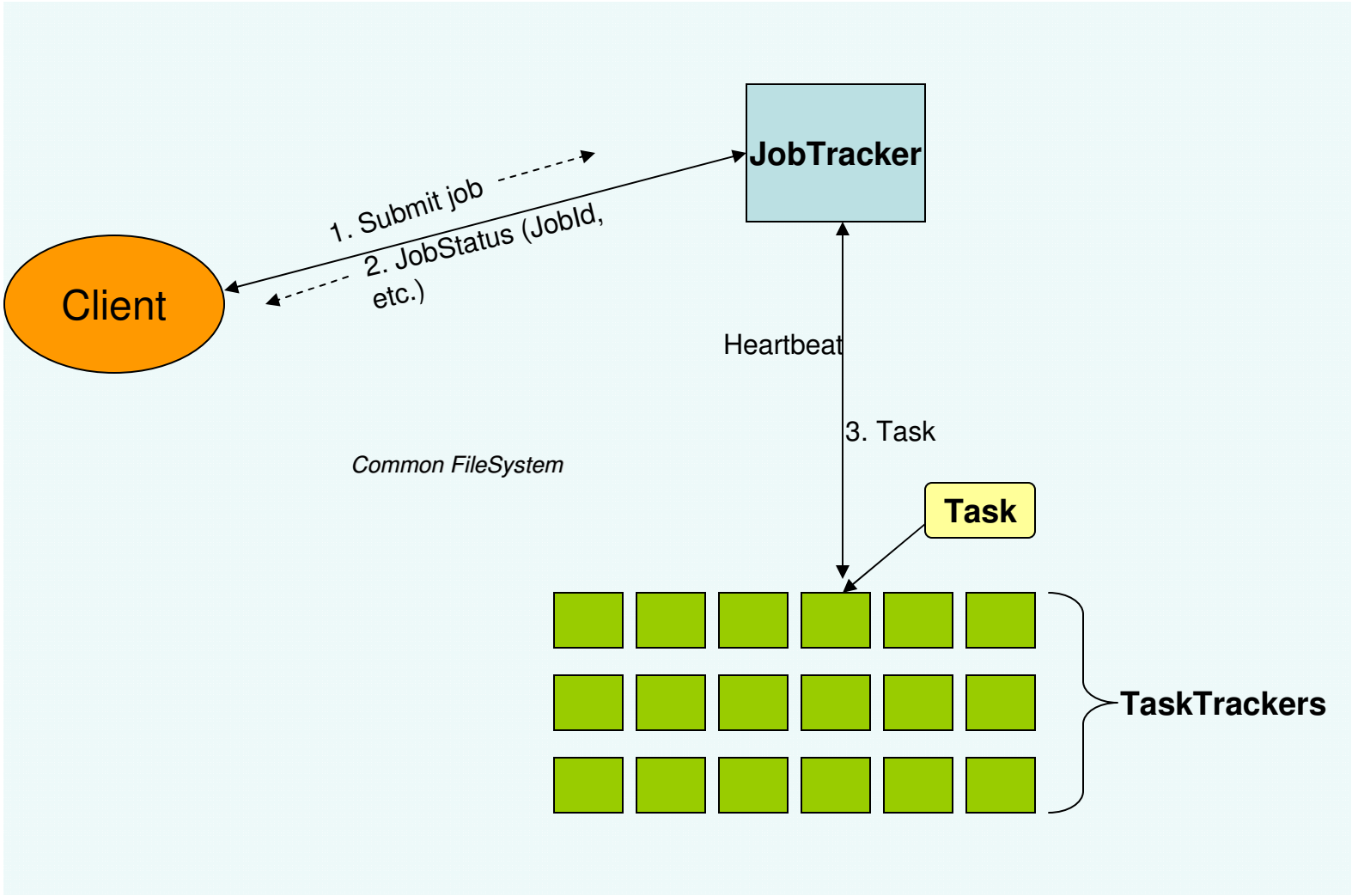
MapReduce: Client

- User defined
  - InputFormat, RecordReader, InputSplit
  - OutputFormat, RecordWriter
  - #Reducers
  - Sort order & Partitioner
  - Host of other configs ….
- Launching Program
  - Gets the FileSystem (pluggable)
  - Creates an InputSplit array (#maps = #splits)
  - Creates a JobConf to define a job
  - Submits JobConf, InputSplits & Application binaries to JobTracker and waits for completion

MapReduce: JobTracker & TaskTracker

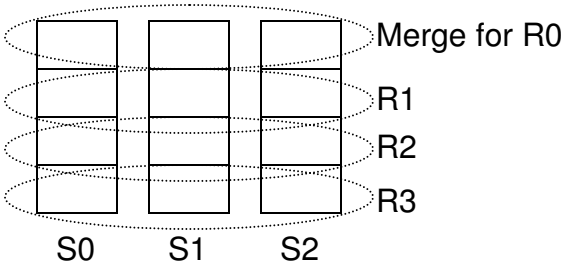
- JobTracker
  - Handles all jobs
  - Makes all scheduling decisions
  - Breaks job into tasks, queues up
  - Schedules tasks on nodes close to data
    - Location Info comes from InputSplit
  - Monitors tasks
  - Kills and restarts tasks if they fail/hang/disappear
- TaskTracker
  - Asks for new tasks, executes, monitors, reports status

MapReduce: Architecture



MapReduce: Data Path

- Map
  - Framework allows pluggable Sort
  - In-memory buffer
  - Sort & Spill
  - Merge across spills



- Shuffle
  - Based on task completion events, http based
- Reduce
  - Buffers in ramfs
  - Periodically merges & spills
  - Does a final (multi-level) merge of the on-disk files
  - Starts reducer method invocations
- **Final Result is always sorted – Distributed MergeSort**

MapReduce: Scalability/Performance

- Sort scalability/performance
- Merge performance
- Disk IO/Seek issues
- MapReduce+HDFS scalability issues
- Network issues in Shuffle of transient data
- JobTracker/TaskTracker scalability/performance
- Data Compression
- Debugging the framework issues!!!