Using Hadoop for Webscale Computing

Ajay Anand
Yahoo!
aanand@yahoo-inc.com

Usenix 2008

Agenda

- The Problem
- Solution Approach / Introduction to Hadoop
- HDFS File System
- Map Reduce Programming
- Pig
- Hadoop implementation at Yahoo!
- Case Study: Yahoo! Webmap
- Where is Hadoop being used
- Future Directions / How you can participate

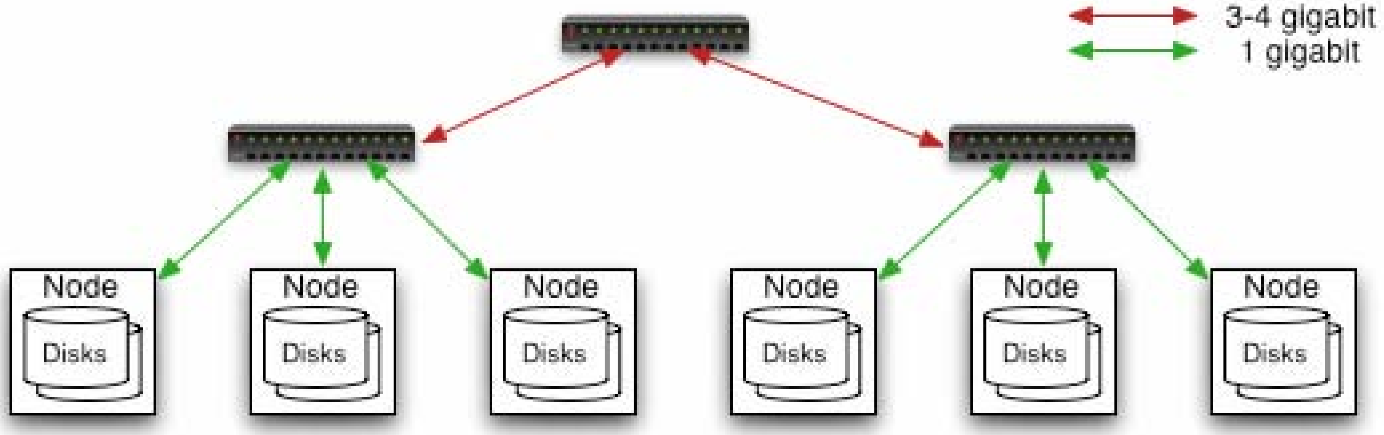
The Problem

- Need massive scalability
  - PB’s of storage, millions of files, 1000’s of nodes
- Need to do this cost effectively
  - Use commodity hardware
  - Share resources among multiple projects
  - Provide scale when needed
- Need reliable infrastructure
  - Must be able to deal with failures – hardware, software, networking
    - Failure is expected rather than exceptional
  - Transparent to applications
    - very expensive to build reliability into each application

Introduction to Hadoop

- Hadoop: Apache Top Level Project
  - Open Source
  - Written in Java
  - Started in 2005 by Doug Cutting as part of Nutch project, became Lucene sub-project in Feb 2006, became top-level project in Jan 2008
- Hadoop Core includes:
  - Distributed File System – modeled on GFS
  - Distributed Processing Framework – using Map-Reduce paradigm
- Runs on
  - Linux, Mac OS/X, Windows, and Solaris
  - Commodity hardware

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 Characteristics

- Commodity HW + Horizontal scaling
  - Add inexpensive servers with JBODS
  - Storage servers and their disks are ***not*** assumed to be highly reliable and available
- Use replication across servers to deal with unreliable storage/servers
- Metadata-data separation - simple design
  - Storage scales horizontally
  - Metadata scales vertically (today)
- Slightly Restricted file semantics
  - Focus is mostly sequential access
  - Single writers
  - No file locking features
- Support for moving computation close to data
  - i.e. servers have 2 purposes: data storage and computation

Simplicity of design

why a small team could build such a large system in the first place

Problem: bandwidth to data

- Need to process 100TB datasets
- On 1000 node cluster reading from remote storage (on LAN)
  - Scanning @ 10MB/s = 165 min
- On 1000 node cluster reading from local storage
  - Scanning @ 50-200MB/s = 33-8 min
- Moving computation is more efficient than moving data
  - Need visibility into data placement

Problem: scaling reliably is hard

- Need to store petabytes of data
  - On 1000s of nodes
  - MTBF < 1 day
  - With so many disks, nodes, switches something is always broken
- Need fault tolerant store
  - Handle hardware faults transparently and efficiently
  - Provide reasonable availability guarantees

HDFS

- Fault tolerant, scalable, distributed storage system
- Designed to reliably store very large files across machines in a large cluster
- Data Model
  - Data is organized into files and directories
  - Files are divided into large uniform sized blocks (e.g.128 MB) and distributed across cluster nodes
  - Blocks are replicated to handle hardware failure
  - Filesystem keeps checksums of data for corruption detection and recovery
  - HDFS exposes block placement so that computes can be migrated to data

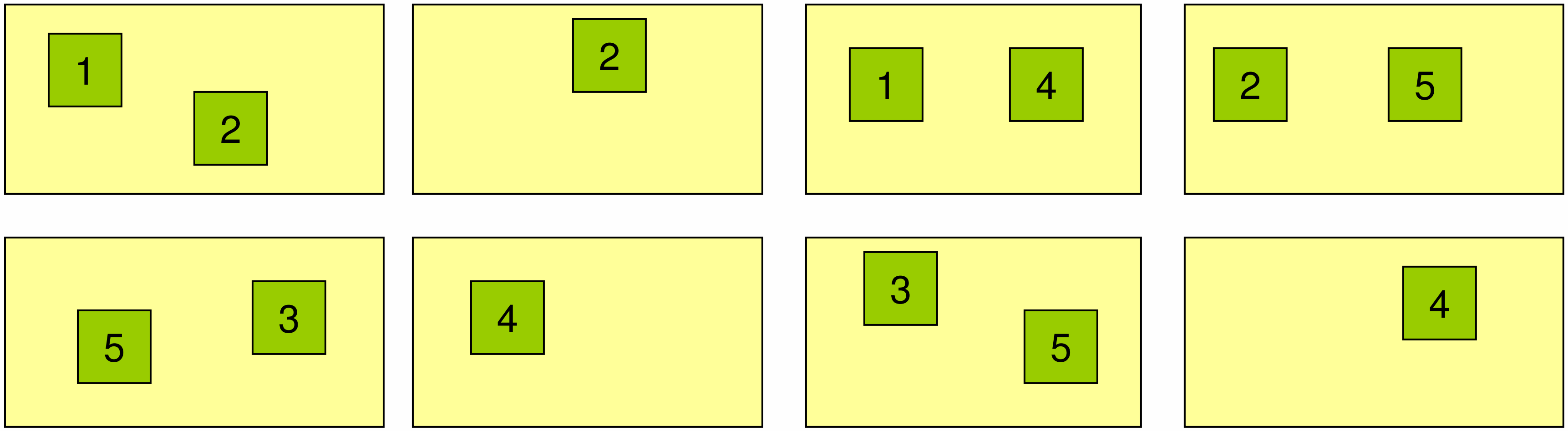
HDFS API

- Most common file and directory operations supported:
  - – Create, open, close, read, write, seek, list, delete etc.
- Files are write once and have exclusively one writer
  - Append/truncate coming soon
- Some operations peculiar to HDFS:
  - – set replication, get block locations

HDFS Architecture

Namenode (Filename, numReplicas, block-ids, …)
/users/sameerp/data/part-0, r:2, {1,3}, …
/users/sameerp/data/part-1, r:3, {2,4,5}, …

Datanodes



Functions of a NameNode

- Manages the File System Namepace
  - Maps a file name to a set of blocks
  - Maps a block to the DataNodes where it resides
- Cluster Configuration Management
- Replication Engine for Blocks
- NameNode Metadata
  - Entire metadata is in main memory
  - Types of Metadata
    - List of files
    - List of Blocks for each file
    - List of DataNodes for each block
    - File attributes, e.g. creation time, replication factor
  - Transaction log
    - Records file creations, file deletions, etc.

Block Placement

- Default is 3 replicas, but settable
- Blocks are placed
  - On same node
  - On different rack
  - On same rack
  - Others placed randomly
- Clients read from closest replica
- If the replication for a block drops below target, it is automatically replicated

Functions of a DataNode

- A Block Server
  - Stores data in the local file system (e.g. ext3)
  - Stores metadata of a block (e.g. CRC)
  - Serves data and metadata to clients
- Block Reports
  - Periodically sends a report of all existing blocks to the NameNode
- Facilitates Pipelining of Data
  - Forwards data to other specified DataNodes

Error Detection and Recovery

- Heartbeats
  - DataNodes send a heartbeat to the NameNode once every 3 seconds
  - NameNode uses heartbeats to detect DataNode failure
- Resilience to DataNode failure
  - Namenode chooses new DataNodes for new replicas
  - Balances disk usage
  - Balances communication traffic to DataNodes
- Data Correctness
  - Use checksums to validate data (CRC32)
  - Client receives data and checksum from datanode
  - If validation fails, client tries other replicas

NameNode Failure

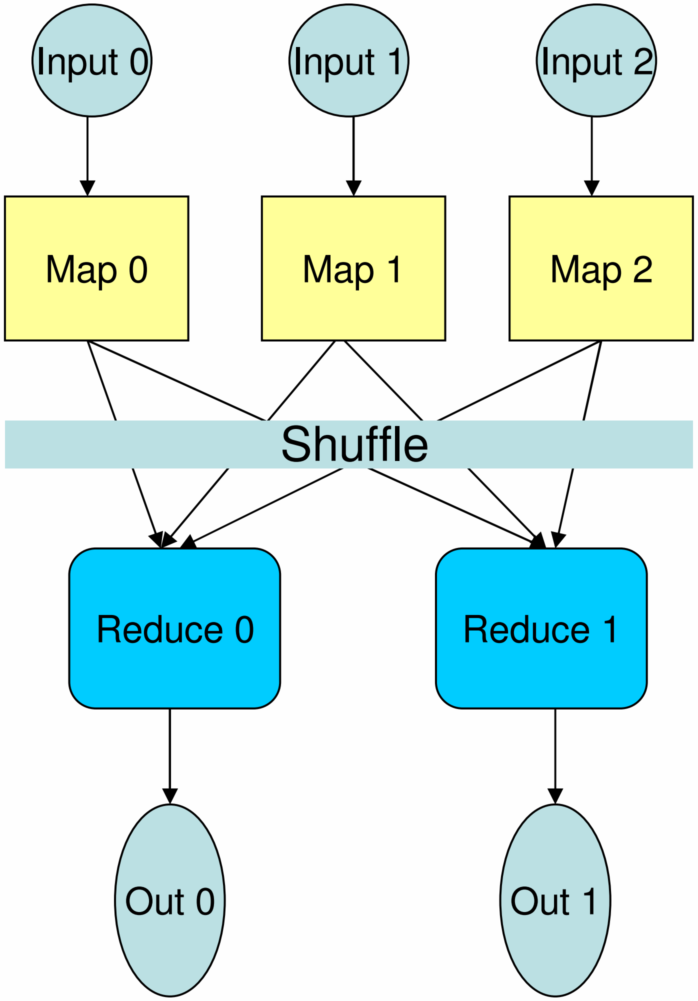
- Currently a single point of failure
- Transaction log stored in multiple directories
  - A directory on the local file system
  - A directory on a remote file system (NFS, CIFS)
- Secondary NameNode
  - Copies FSImage and Transaction Log from the Namenode to a temporary directory
  - Merges FSImage and Transaction Log into a new FSImage in the temporary directory
  - Uploads new FSImage to the NameNode
  - Transaction Log on the NameNode is purged

Map/Reduce

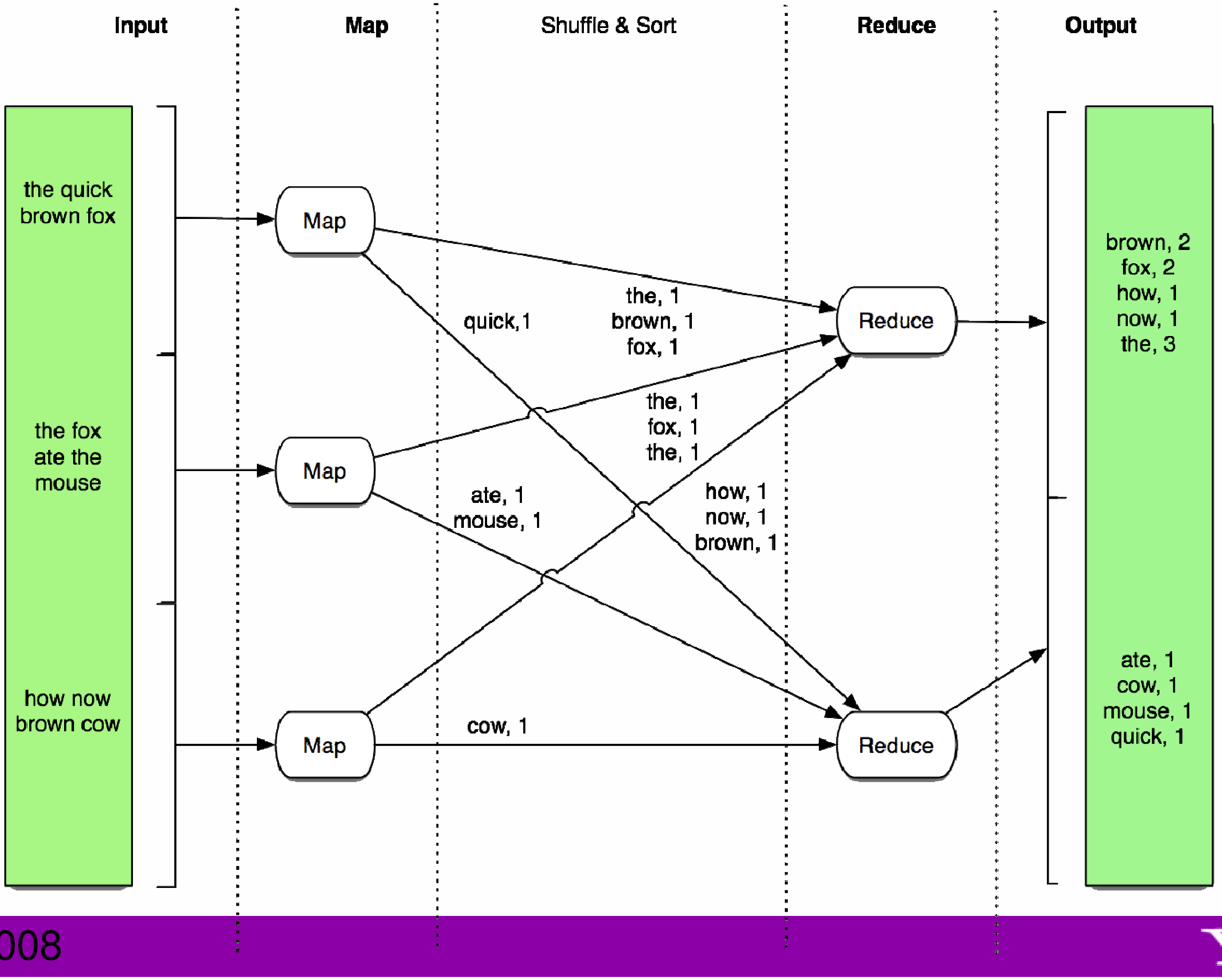
- Map/Reduce is a programming model for efficient distributed computing
- It works like a Unix pipeline:
  - cat * | grep | sort | uniq -c | cat > output
  - **Input** | **Map** | Shuffle & Sort | **Reduce** | **Output**
- Efficiency from
  - Streaming through data, reducing seeks
  - Pipelining
- Natural for
  - Log processing
  - Web index building

Map/Reduce

- **Application writer specifies**
  - – A pair of functions called *Map* and *Reduce* and a set of input files
- **Workflow**
  - – *Input* phase generates a number of *FileSplits* from input files (one per Map task)
  - – The *Map* phase executes a user function to transform input kv-pairs into a new set of kv-pairs
  - – The framework sorts & *Shuffles* the kv-pairs to output nodes
  - – The *Reduce* phase combines all kv-pairs with the same key into new kv-pairs
  - – The output phase writes the resulting pairs to files
- **All phases are distributed with many tasks doing the work**
  - – Framework handles scheduling of tasks on cluster
  - – Framework handles recovery when a node fails



Word Count Example



Map/Reduce optimizations

- Overlap of maps, shuffle, and sort
- Mapper locality
  - Map/Reduce queries HDFS for locations of input data
  - Schedule mappers close to the data.
- Fine grained Map and Reduce tasks
  - Improved load balancing
  - Faster recovery from failed tasks
- Speculative execution
  - Some nodes may be slow, causing long tails in computation
  - Run duplicates of last few tasks - pick the winners
  - Controlled by the configuration variable *mapred.speculative.execution*

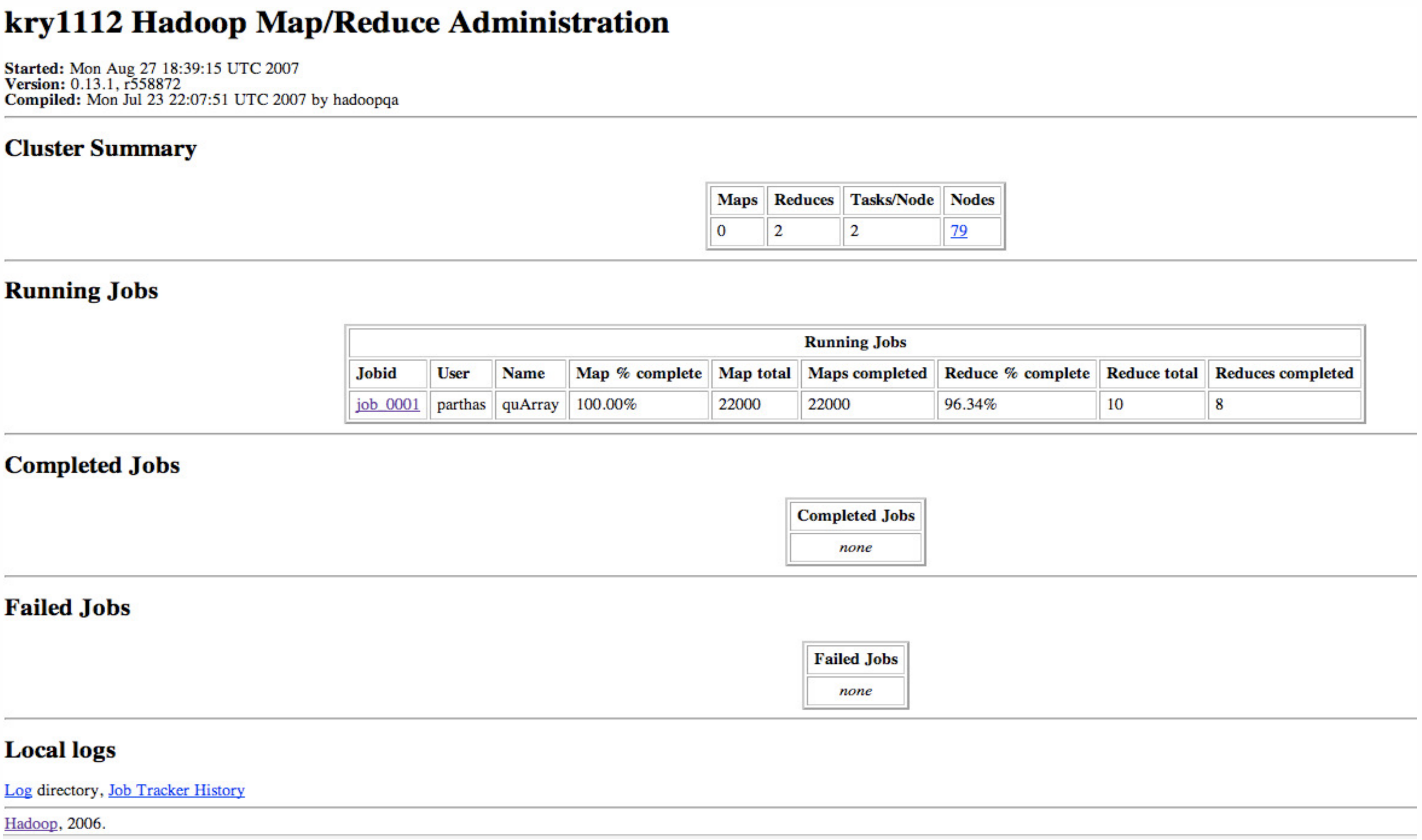
Compression

- Compressing the outputs and intermediate data will often yield huge performance gains
  - Can be specified via a configuration file or set programatically
  - Set *mapred.output.compress* to *true* to compress job output
  - Set *mapred.compress.map.output* to *true* to compress map outputs
- Compression Types *(mapred(.map)?.output.compression.type)*
  - "block" - Group of keys and values are compressed together
  - "record" - Each value is compressed individually
  - Block compression is almost always best
- Compression Codecs *(mapred(.map)?.output.compression.codec)*
  - Default (zlib) - slower, but more compression
  - LZO - faster, but less compression

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

Jobtracker front page



kry1112 Hadoop Map/Reduce Administration

Started: Mon Aug 27 18:39:15 UTC 2007
Version: 0.13.1, r558872
Compiled: Mon Jul 23 22:07:51 UTC 2007 by hadoopqa

Cluster Summary

Maps	Reduces	Tasks/Node	Nodes
0	2	2	79

Running Jobs

Running Jobs								
Jobid	**User**	**Name**	**Map % complete**	**Map total**	**Maps completed**	**Reduce % complete**	**Reduce total**	**Reduces completed**
job_0001	parthas	quArray	100.00%	22000	22000	96.34%	10	8

Completed Jobs

Completed Jobs
none

Failed Jobs

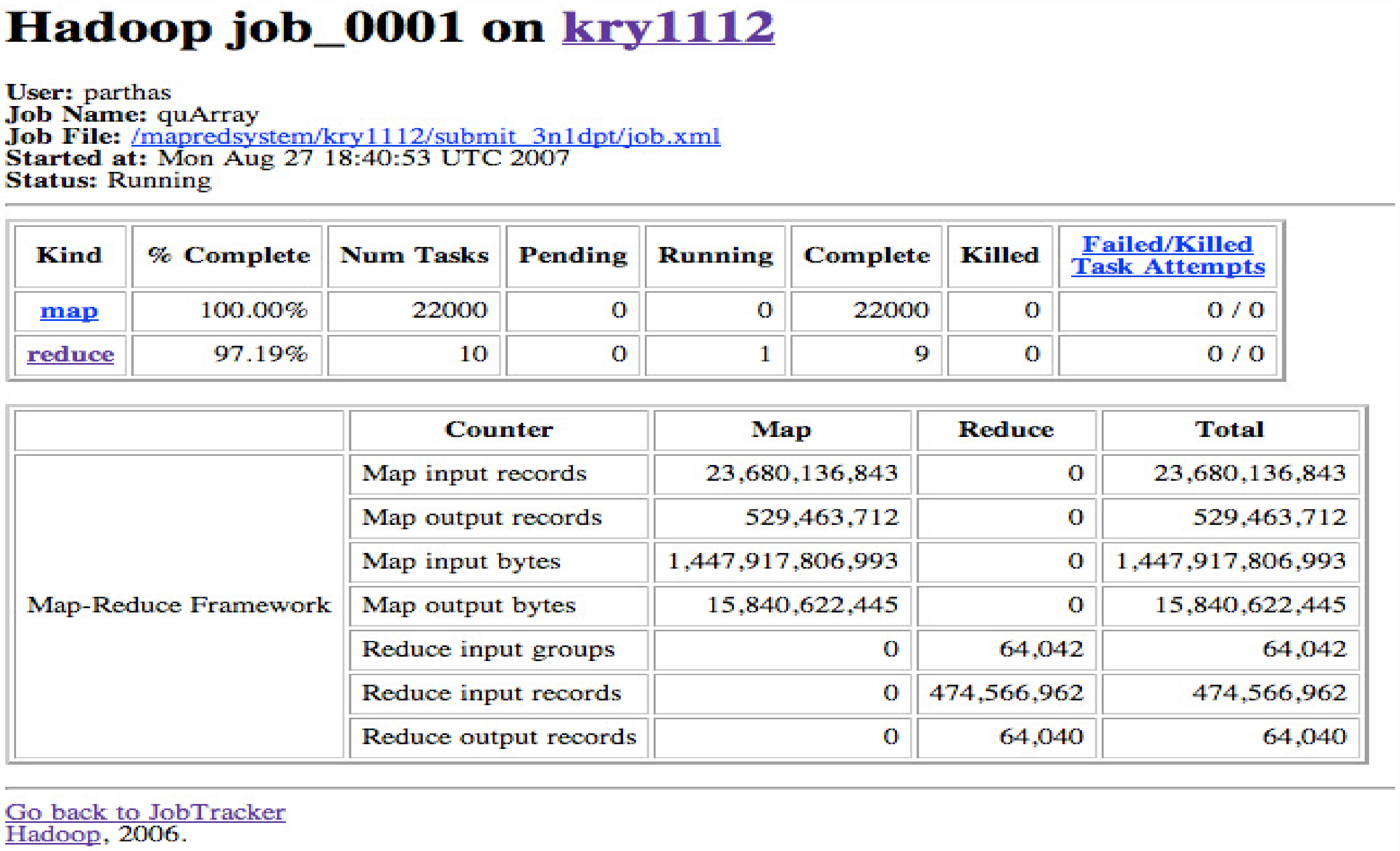
Failed Jobs
none

Local logs

Log directory, Job Tracker History

Hadoop, 2006.

Job counters



Hadoop job_0001 on kry1112

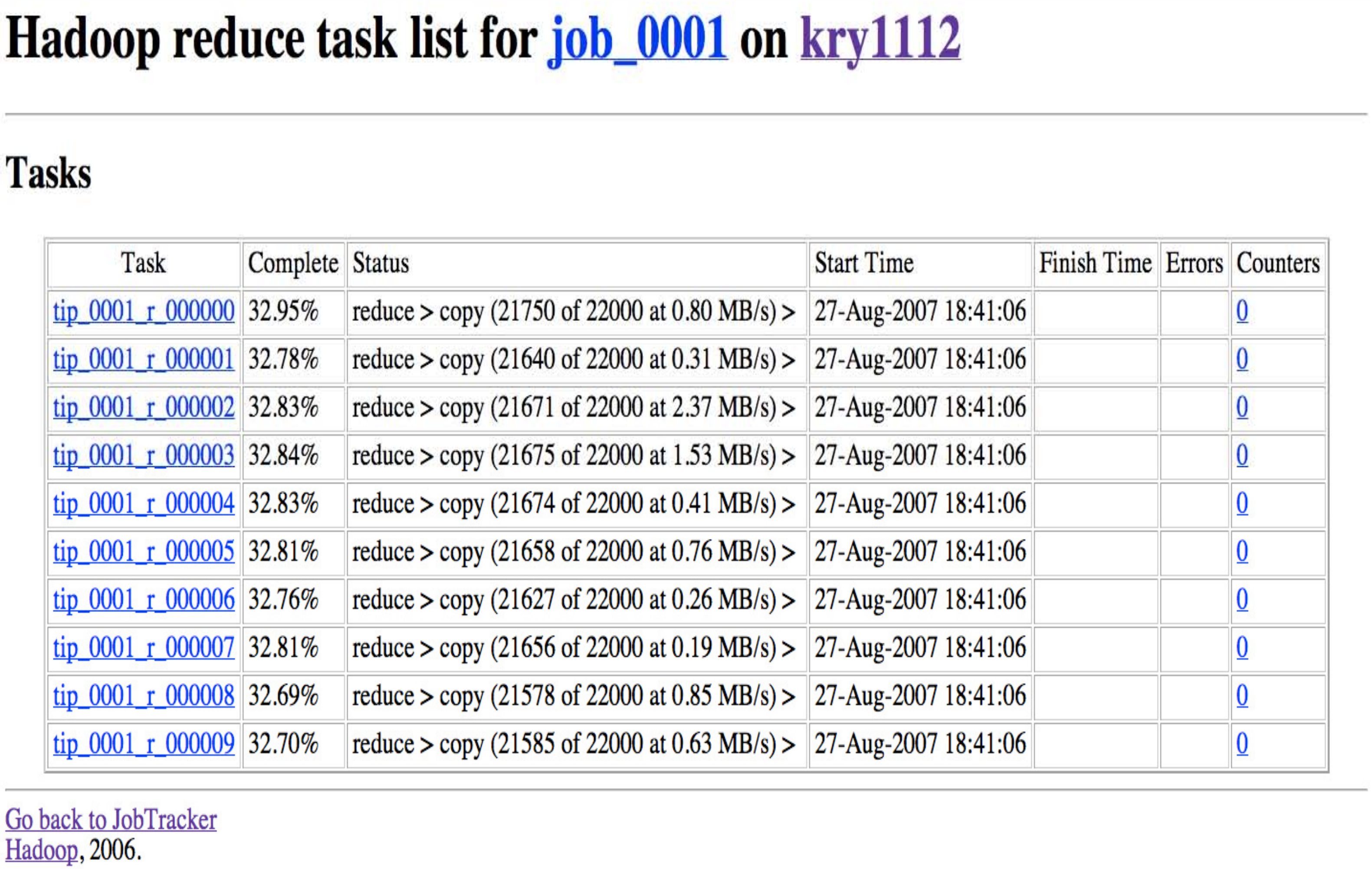
User: parthas
Job Name: quArray
Job File: /mapredsystem/kry1112/submit_3n1dpt/job.xml
Started at: Mon Aug 27 18:40:53 UTC 2007
Status: Running

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	22000	0	0	22000	0	0 / 0
reduce	97.19%	10	0	1	9	0	0 / 0

	Counter	Map	Reduce	Total
Map-Reduce Framework	Map input records	23,680,136,843	0	23,680,136,843
	Map output records	529,463,712	0	529,463,712
	Map input bytes	1,447,917,806,993	0	1,447,917,806,993
	Map output bytes	15,840,622,445	0	15,840,622,445
	Reduce input groups	0	64,042	64,042
	Reduce input records	0	474,566,962	474,566,962
	Reduce output records	0	64,040	64,040

Go back to JobTracker
Hadoop, 2006.

Task status



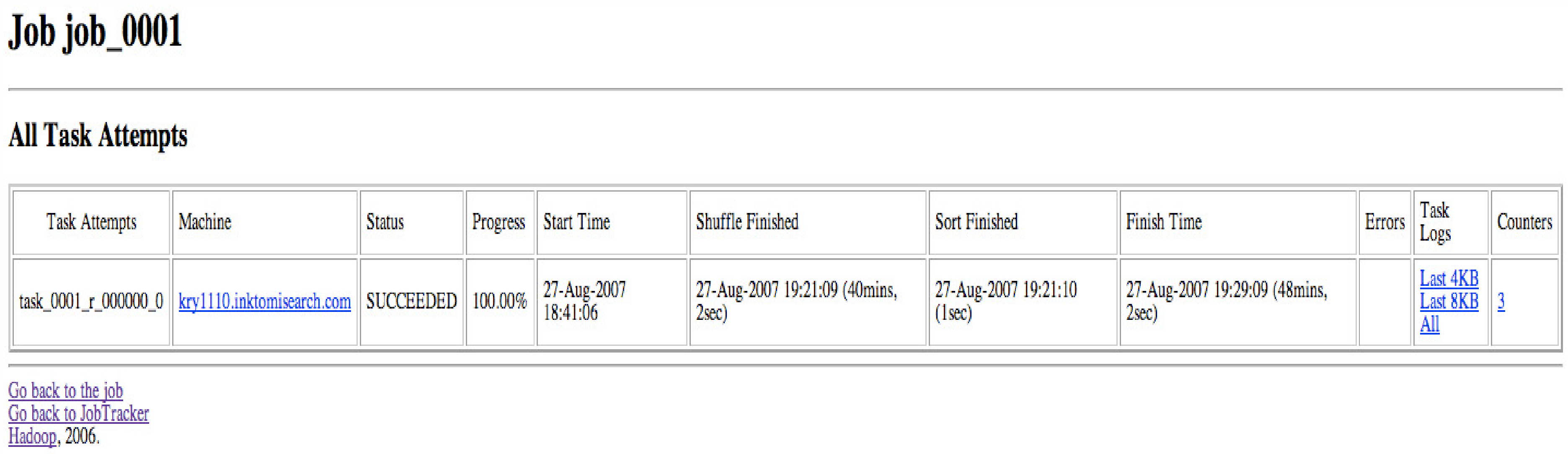
Hadoop reduce task list for job_0001 on kry1112

Tasks

Task	Complete	Status	Start Time	Finish Time	Errors	Counters
tip_0001_r_000000	32.95%	reduce > copy (21750 of 22000 at 0.80 MB/s) >	27-Aug-2007 18:41:06			0
tip_0001_r_000001	32.78%	reduce > copy (21640 of 22000 at 0.31 MB/s) >	27-Aug-2007 18:41:06			0
tip_0001_r_000002	32.83%	reduce > copy (21671 of 22000 at 2.37 MB/s) >	27-Aug-2007 18:41:06			0
tip_0001_r_000003	32.84%	reduce > copy (21675 of 22000 at 1.53 MB/s) >	27-Aug-2007 18:41:06			0
tip_0001_r_000004	32.83%	reduce > copy (21674 of 22000 at 0.41 MB/s) >	27-Aug-2007 18:41:06			0
tip_0001_r_000005	32.81%	reduce > copy (21658 of 22000 at 0.76 MB/s) >	27-Aug-2007 18:41:06			0
tip_0001_r_000006	32.76%	reduce > copy (21627 of 22000 at 0.26 MB/s) >	27-Aug-2007 18:41:06			0
tip_0001_r_000007	32.81%	reduce > copy (21656 of 22000 at 0.19 MB/s) >	27-Aug-2007 18:41:06			0
tip_0001_r_000008	32.69%	reduce > copy (21578 of 22000 at 0.85 MB/s) >	27-Aug-2007 18:41:06			0
tip_0001_r_000009	32.70%	reduce > copy (21585 of 22000 at 0.63 MB/s) >	27-Aug-2007 18:41:06			0

Go back to JobTracker
Hadoop, 2006.

Drilling down



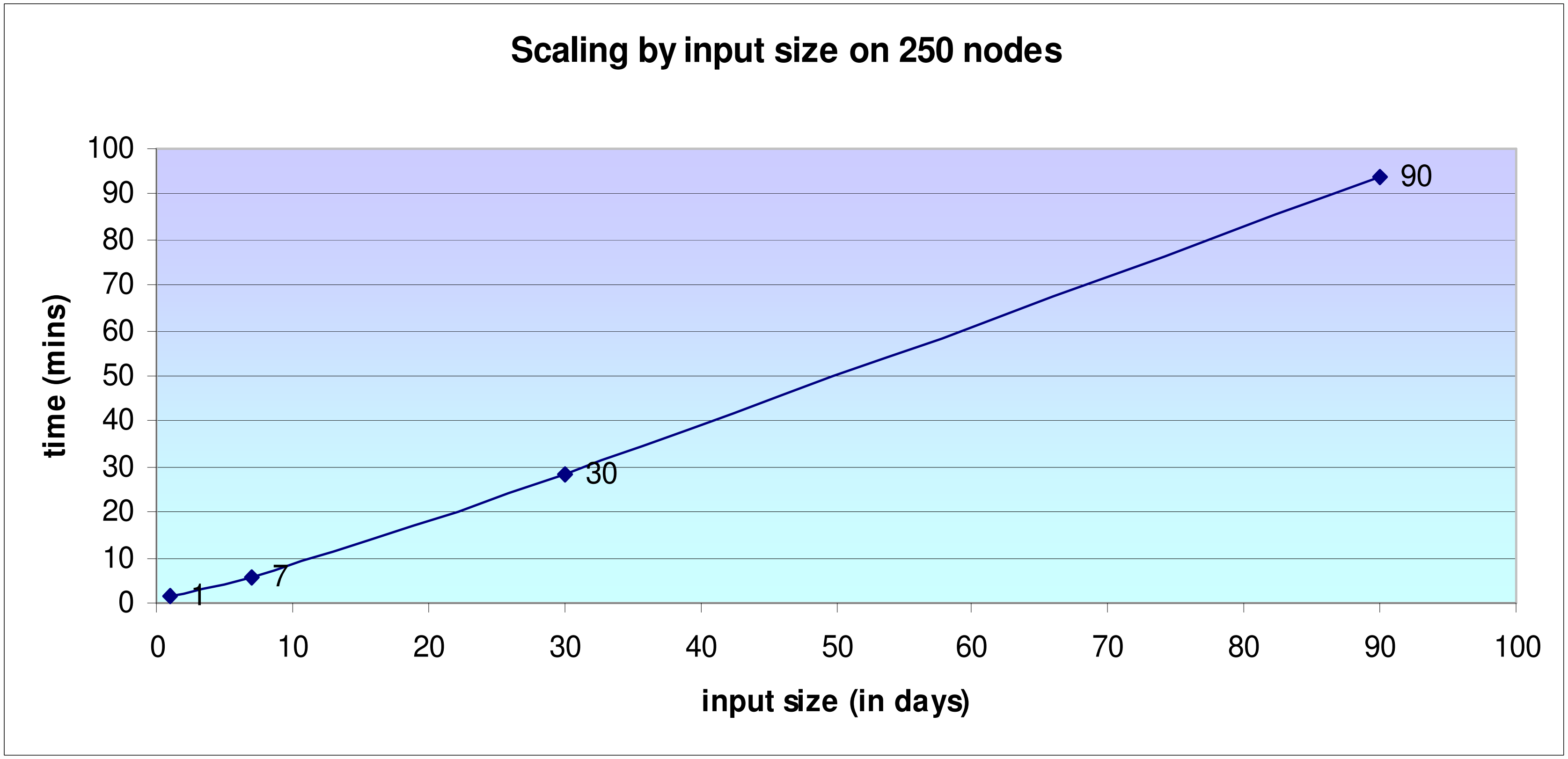
Job job_0001

All Task Attempts

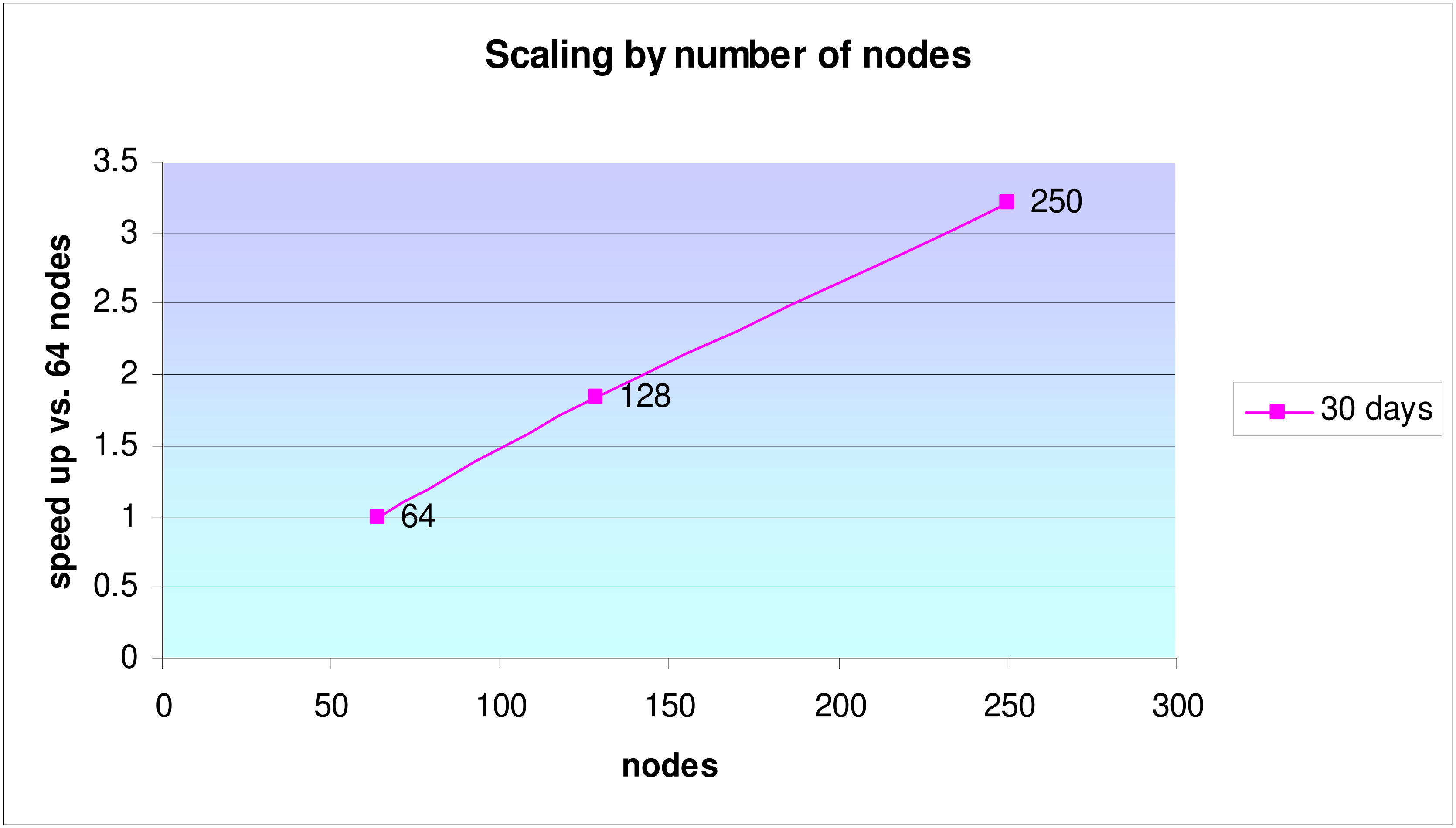
Task Attempts	Machine	Status	Progress	Start Time	Shuffle Finished	Sort Finished	Finish Time	Errors	Task Logs	Counters
task_0001_r_000000_0	kry1110.inktomisearch.com	SUCCEEDED	100.00%	27-Aug-2007 18:41:06	27-Aug-2007 19:21:09 (40mins, 2sec)	27-Aug-2007 19:21:10 (1sec)	27-Aug-2007 19:29:09 (48mins, 2sec)		Last 4KB Last 8KB All	3

Go back to the job
Go back to JobTracker
Hadoop, 2006.

Scaling by input size (on 250 nodes)



Scaling by increasing number of nodes (30 days input)



Queuing and Scheduling

- Hadoop does not have an advanced scheduling system
  - MapReduce JobTracker manages one or more jobs running within a set of machines
  - Works well for “dedicated” applications, but does not work so well for shared resources
- Hadoop on Demand (HOD)
  - Bridge between Hadoop and resource managers, such as Torque and Condor
  - Virtual private JobTracker clusters
  - Job isolation
    - Users create clusters of the size they need
    - Submit jobs to their private JobTracker
  - Disadvantages:
    - Lose data locality
    - Increased complexity
    - Lose a node for private JobTracker
    - Single reducer doesn’t free unused nodes: ~30% efficiency loss

Pig

- Pig: Apache incubator project initiated by Yahoo!
- Pig Latin: High level dataflow language that generates Map/Reduce jobs
- Simpler for users
  - High-level, extensible data processing primitives
- Comparing Pig and Map-Reduce
  - Map-Reduce welds together 3 primitives:
    - Process records -> create groups -> process groups
  - Using Pig:

```
a = FOREACH input GENERATE flatten(Map(*));
b = GROUP a BY $0;
c = FOREACH b GENERATE Reduce(*)
```

Grid Computing at Yahoo!

- Drivers
  - 500M unique users per month
  - Billions of interesting events per day
  - *“Data analysis is the inner-loop at Yahoo!”*
- Yahoo! Grid Vision and Focus
  - On-demand, shared access to vast pool of resources
  - Support for massively parallel execution (1000s of processors)
  - Data Intensive Super Computing (DISC)
  - Centrally provisioned and managed
  - Service-oriented, elastic
- What We’re Not
  - Not “Grid” in the sense of scientific community (Globus, etc)
  - Not focused on public or 3rd-party utility (Amazon EC2/S3, etc)

Yahoo! / Apache Grid Ecosystem

- Open Source Stack
  - Commitment to Open Source Development
  - Y! is Apache Platinum Sponsor
- Hadoop
  - Distributed File System
  - MapReduce Framework
  - Dynamic Cluster Manager (HOD)
- Pig
  - Parallel Programming Language and Runtime
- Zookeeper
  - High-Availability Directory and Configuration Service
- Simon
  - Cluster and Application Monitoring

The Apache Software Foundation
http://www.apache.org/

hadoop

Yahoo! Grid Services

- Operate multiple Grid clusters within Yahoo!
- 10,000s nodes, 100,000s cores, TBs RAM, PBs disk
- Support internal user community
  - Account management, training, etc
- Manage data needs
  - Ingest TBs per day
- Deploy and manage software stack
- 24x7 support

Case Study: Yahoo! Webmap

- What's a WebMap?
  - Gigantic table of information about every web site, page and link Yahoo knows about
  - Directed graph of the web
  - Various aggregated views (sites, domains, etc)
  - Various algorithms for ranking, duplicate detection, region classification, spam detection, etc.
- Why port to Hadoop?
  - Leverage scalability, load balancing and resilience of Hadoop infrastructure
  - Reduce management overhead
  - Provide access to many researchers
  - Focus on application vs infrastructure
  - Leverage open source, rapidly improving platform

Webmap Results

- 33% time savings over previous similarly sized cluster
- Largest job:
  - 100,000+ maps, ~10,000 reduces
  - ~70 hours runtime
  - ~300 TB shuffling
  - ~200 TB compressed output
- Over 10,000 cores in system
- Reduced operational cost
- Simplified access to researchers
- Many opportunities for further improvement

Who else is using Hadoop?

- Still pre-1.0, but already used by many: http://wiki.apache.org/hadoop/PoweredBy
- Some examples from this site:
  - A9.com – Amazon
    - We build Amazon's product search indices using the streaming API and pre-existing C++, Perl, and Python tools.
    - We process millions of sessions daily for analytics, using both the Java and streaming APIs.
    - Our clusters vary from 1 to 100 nodes. .
  - Facebook
    - We use Hadoop to store copies of internal log and dimension data sources and use it as a source for reporting/analytics and machine learning.
    - Currently have a 320 machine cluster with 2,560 cores and about 1.3 PB raw storage. Each (commodity) node has 8 cores and 4 TB of storage.
    - We are heavy users of both streaming as well as the Java apis. We have built a higher level data warehousing framework using these features called Hive (see the JIRA ticket). We have also written a read-only FUSE implementation over hdfs.
  - Fox Interactive Media
  - Google University Initiative
  - IBM
  - Joost
  - Last.fm
  - Mahout
  - The New York Times
  - PARC
  - Powerset
  - Veoh
  - Yahoo!
  - Multiple Universities

Running on Amazon EC2/S3

- Amazon sells cluster services
  - EC2: priced per cpu hour
  - S3: priced per GB month
- Hadoop supports:
  - EC2: cluster management scripts included
  - S3: file system implementation included
- Tested on 400 node cluster
- Combination used by several startups

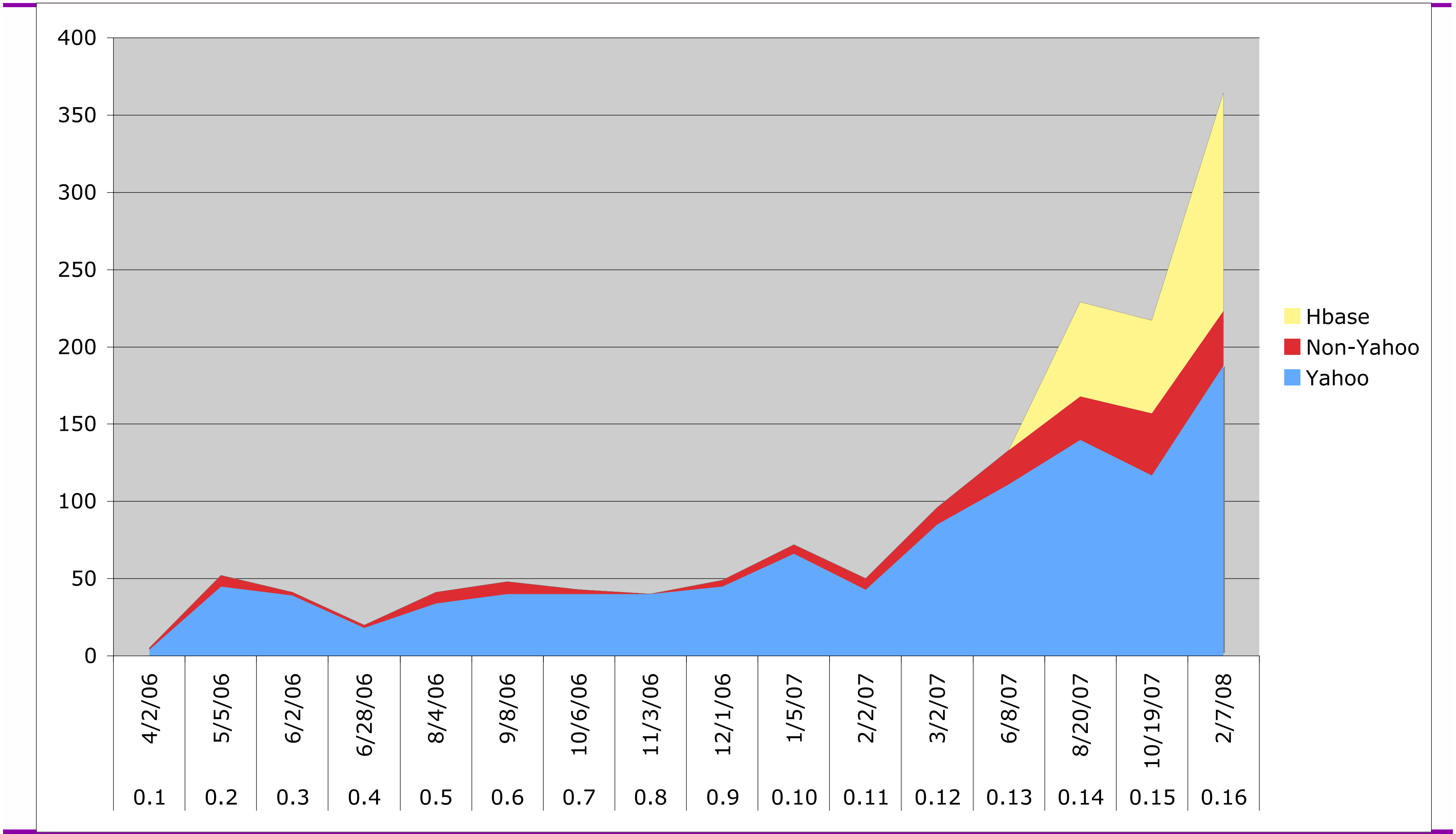
M45 Program -- Open Academic Clusters

- Collaboration with Major Research Universities
  - Foster open research
  - Focus on large-scale, highly parallel computing
- Seed Facility: Datacenter in a Box (DiB)
  - 500 nodes, 4000 cores, 3TB RAM, 1.5PB disk
  - High bandwidth connection to Internet
  - Located on Yahoo! corporate campus
- Runs Yahoo! / Apache Grid Stack
- Carnegie Mellon University is Initial Partner
- Public Announcement 11/12/07

Subprojects

- Pig (initiated by Yahoo!)
  - Programming language and runtime for data analysis
- Hbase (initiated by Powerset)
  - Table storage for semi-structured data
- Zookeeper (initiated by Yahoo!)
  - Coordinating distributed systems
- Hive (initiated by Facebook, coming soon)
  - SQL-like query language and metastore
- Mahout
  - Machine learning algorithms

Tracking patches per release



Join the Apache Hadoop Community

- Hosted Hadoop summit in March 08
  - Registrants from over 100 organizations
- Hadoop is now in universities in several continents
  - Yahoo! initiatives in US, India
    - M45 Program
    - Initiative with Tata / CRL in India
  - IBM / Google university initiative
- http://wiki.apache.org/hadoop/ProjectSuggestions
  - Ideas for folks who want to get started
- http://hadoop/apache.org - the main Apache site
  - Mailing lists, the code, documentation and more
- http://wiki.apache.org/hadoop/PoweredBy
  - A list of users, please add yourself!

Questions?

Using Hadoop for Webscale Computing

Ajay Anand
Yahoo!
aanand@yahoo-inc.com

Usenix 2008