Distributed Data processing in a Cloud

Rajiv Chittajallu

Yahoo! Inc

rajive@yahoo-inc.com

Software as a Service and Cloud Computing Workshop

ISEC2008, Hyderabad, India

22 February 2008

Desiderata

- **Operate scalably**
  - Petabytes of data on thousands on nodes
  - Much larger that RAM, disk I/O required
- **Operate economically**
  - Minimize $ spent on CPU cycles, disk, RAM, network
  - Lash thousands of commodity PCs into an effective compute and storage platform
- **Operate reliably**
  - In a large enough cluster something is always broken
  - Seamlessly protect user data and computations from hardware and software flakiness

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

Distributed File System

- **Fault tolerant, scalable, distributed storage system**
- **Designed to reliably store very large files across machines in a large cluster**
- **Common Namespace for the entire filesystem**
  - Distribute namespace for scalability and failover
- **Data Model**
  - Data is organized into files and directories
  - Files are divided into uniform sized blocks and distributed across cluster nodes
  - Replicate blocks to handle hardware failure
  - Checksums of data for corruption detection and recovery
  - Expose block placement so that computes can be migrated to data

Problem: seeks are expensive

- CPU & transfer speed, RAM & disk size double every 18-24 months
- Seek time nearly constant (~5%/year)
- Time to read entire drive is growing
- Moral: scalable computing must go at transfer rate

Two database paradigms: seek versus transfer

- **B-Tree (Relational Dbs)**
  - operate at seek rate, log(N) seeks/access
- **sort/merge flat files (MapReduce)**
  - operate at transfer rate, log(N) transfers/sort
- **Caveats:**
  - sort & merge is batch based
    - although possible to work around
  - other paradigms (memory, streaming, etc.)

Example: updating a terabyte DB

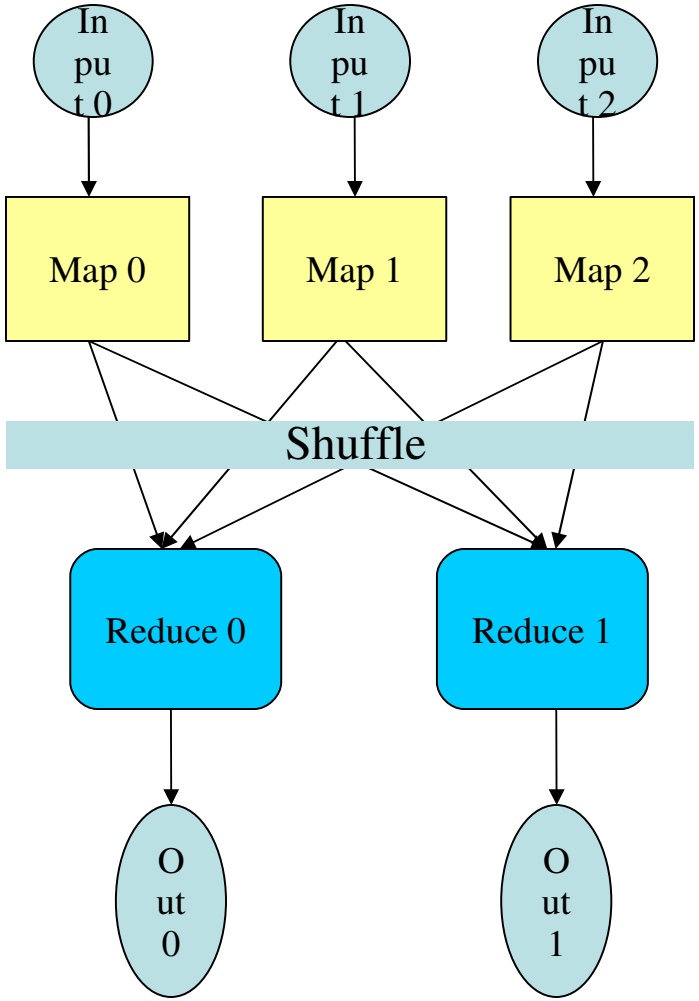
- **given:**
  - 10MB/s transfer
  - 10ms/seek
  - 100B/entry (10B entries)
  - 10kB/page (1B pages)
- **updating 1% of entries (100M) takes:**
  - 1000 days with random B-Tree updates
  - 100 days with batched B-Tree updates
  - 1 day with sort & merge

Map/Reduce: sort/merge based distributed processing

- **Best for batch-oriented processing**
- **Sort/merge is primitive**
  - Operates at transfer rate
- **Simple programming metaphor:**
  - input | map | shuffle | reduce > output
  - cat * | grep | sort | uniq -c > file
- **Pluggable user code runs in generic reusable framework**
  - A natural for log processing, great for most web search processing
  - A lot of SQL maps trivially to this construct (see PIG)
- **Distribution & reliability**
  - Handled by framework

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



Map/Reduce features

- **Fine grained Map and Reduce tasks**
  - Improved load balancing
  - Faster recovery from failed tasks
- **Locality optimizations**
  - With big data, bandwidth to data is a problem
  - Map-Reduce + DFS is a very effective solution
  - Map-Reduce queries DFS for locations of input data
  - Map tasks are scheduled local to the inputs when possible
- **Re-execution and Speculative execution**
  - In a large cluster, some nodes are always slow or flaky
  - Introduces long tails or failures in computation
  - Framework re-executes failed jobs
  - Framework runs multiple instances of last few tasks and uses the ones that finish first

Map/Reduce: pros and cons

- **Developing large scale systems is expensive, this is a shared platform**
  - Reduces development and debug time
  - Leverages common optimizations, tools etc.
- **Not always a natural fit**
  - With moderate force, many things will fit
- **Not always optimal**
  - But not far off, and often cheaper in the end

Hadoop

- **Apache Software Foundation project**
  - Framework for running applications on large clusters of commodity hardware
  - Since we've convinced Doug Cutting to split Hadoop into a separate project, Yahoo! is the main contributor of source code to the infrastructure base.
  - A search startup has adapted Hadoop to run on Amazon's EC2 and S3, and has contributed hBase, a BigTable-like extension.
    - http://hadoop.apache.org/hbase/
- **Includes**
  - HDFS - a distributed filesystem
  - Map/Reduce - offline computing engine
  - Hbase – online data access
- **Still pre-1.0, but already used by many**
  - http://wiki.apache.org/hadoop/PoweredBy
  - alpha (0.16) release available for download
- **http://lucene.apache.org/hadoop**

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

HDFS Architecture

- Master-Slave architecture
- DFS Master “Namenode”
  - Manages the filesystem namespace
  - Controls read/write access to files
  - Manages block replication
  - Checkpoints namespace and journals namespace changes for reliability
- DFS Slaves “Datanodes”
  - Serve read/write requests from clients
  - Perform replication tasks upon instruction by namenode

HDFS

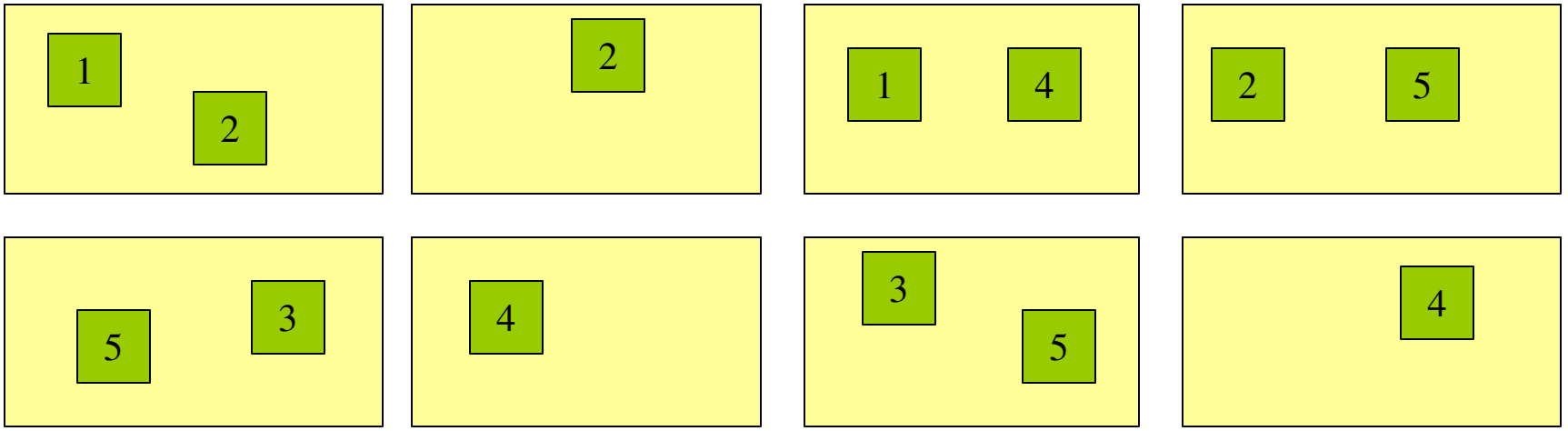
- **Notable differences from mainstream DFS work**
  - Single ‘storage + compute’ cluster vs. Separate clusters
  - Simple I/O centric API vs. Attempts at POSIX compliance
    - Not against POSIX but currently prioritizing scale and reliability

Block Placement

Namenode

name:/users/foo/myFile - copies:2, blocks:{1,3}
name:/users/bar/someData.gz, copies:3, blocks:{2,4,5}

Datanodes



HDFS API

- Most common file and directory operations supported:
  - Create, open, close, read, write, seek, tell, 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
- Support for owners, permissions (v0.16)

HDFS command line utils

```
gritgw1004:/grid/0/tmp/rajive$ ls -lt
total 1300392
-rw-r--r-- 1 rajive users  244827000 Jan 20 05:02 1.5K-alice30.txt
-rw-r--r-- 1 rajive users    8160900 Jan 20 05:02 50-alice30.txt
-rw-r--r-- 1 rajive users 1077290150 Jan 20 04:58 part-00737
gritgw1004:/grid/0/tmp/rajive$ hadoop dfs -ls
Found 1 items
/user/rajive/rand0      <dir>           2008-01-20 05:00

gritgw1004:/grid/0/tmp/rajive$ hadoop dfs -ls /user/rajive
Found 5 items
/user/rajive/alice      <dir>           2008-01-20 05:15
/user/rajive/alice-1.5k <dir>           2008-01-20 05:20
/user/rajive/rand0      <dir>           2008-01-20 05:00

gritgw1004:/grid/0/tmp/rajive$ hadoop dfs -put 50-alice30.txt /user/rajive/alice

gritgw1004:/grid/0/tmp/rajive$ hadoop dfs -ls /user/rajive/alice
Found 1 items
/user/rajive/alice/50-alice30.txt       <r 3>   8160900 2008-01-20 05:05


gritgw1004:/grid/0/tmp/rajive$ hadoop dfs -cat /user/rajive/alice/50-alice30.txt
***This is the Project Gutenberg Etext of Alice in Wonderland***
*This 30th edition should be labeled alice30.txt or alice30.zip.
***This Edition Is Being Officially Released On March 8, 1994***
**In Celebration Of The 23rd Anniversary of Project Gutenberg***
```

HDFS UI

NameNode 'grit1002.yahooresearchcluster.com:8020'

Started:	Mon Jan 07 23:40:49 UTC 2008
Version:	0.15.1, r596497
Compiled:	Tue Nov 20 01:34:59 UTC 2007 by hadoopqa

Browse the filesystem

Cluster Summary

Capacity	:	953.86 TB
DFS Remaining	:	713.84 TB
DFS Used	:	11.21 TB
DFS Used%	:	1.18 %
Live Nodes	:	365
Dead Nodes	:	6

Live Datanodes : 365

Node	Last Contact	Admin State	Size (TB)	Used (%)	Used (%)	Remaining (TB)	Blocks
grit1007	0	In Service	2.61	0.75		1.97	488
grit1008	0	In Service	2.61	0.27		1.98	420
grit1009	0	In Service	2.61	0.69		1.97	519
grit1010	0	In Service	2.61	0.79		1.96	531
grit1011	0	In Service	2.61	0.28		1.98	664
grit1012	1	In Service	2.61	0.71		1.97	503
grit1013	0	In Service	2.61	0.68		1.97	479
grit1014	1	In Service	2.61	12.52		1.7	3065
grit1015	0	In Service	2.61	0.18		1.98	456
grit1016	1	In Service	2.61	0.79		1.96	549
grit1017	2	In Service	2.61	0.35		1.97	655
grit1018	0	In Service	2.61	0.75		1.97	562
grit1019	1	In Service	2.61	2.04		1.94	1056
grit1020	0	In Service	2.61	0.74		1.97	497
grit1022	1	In Service	2.61	0.78		1.96	530
grit1023	1	In Service	2.61	1.98		1.94	1061
grit1024	1	In Service	2.61	0.81		1.96	558
grit1025	2	In Service	2.61	0.37		1.97	729
grit1026	2	In Service	2.61	0.8		1.96	470
grit1027	2	In Service	2.61	0.84		1.96	516
grit1028	1	In Service	2.61	0.74		1.97	467
grit1029	1	In Service	2.61	0.79		1.96	525
grit1030	1	In Service	2.61	0.72		1.97	520

HDFS UI

Contents of directory /user/rajive

Goto : /user/rajive go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time
alice	dir				2008-01-20 05:15
alice-1.5k	dir				2008-01-20 05:20
rand0	dir				2008-01-20 05:00
wcout	dir				2008-01-20 05:17

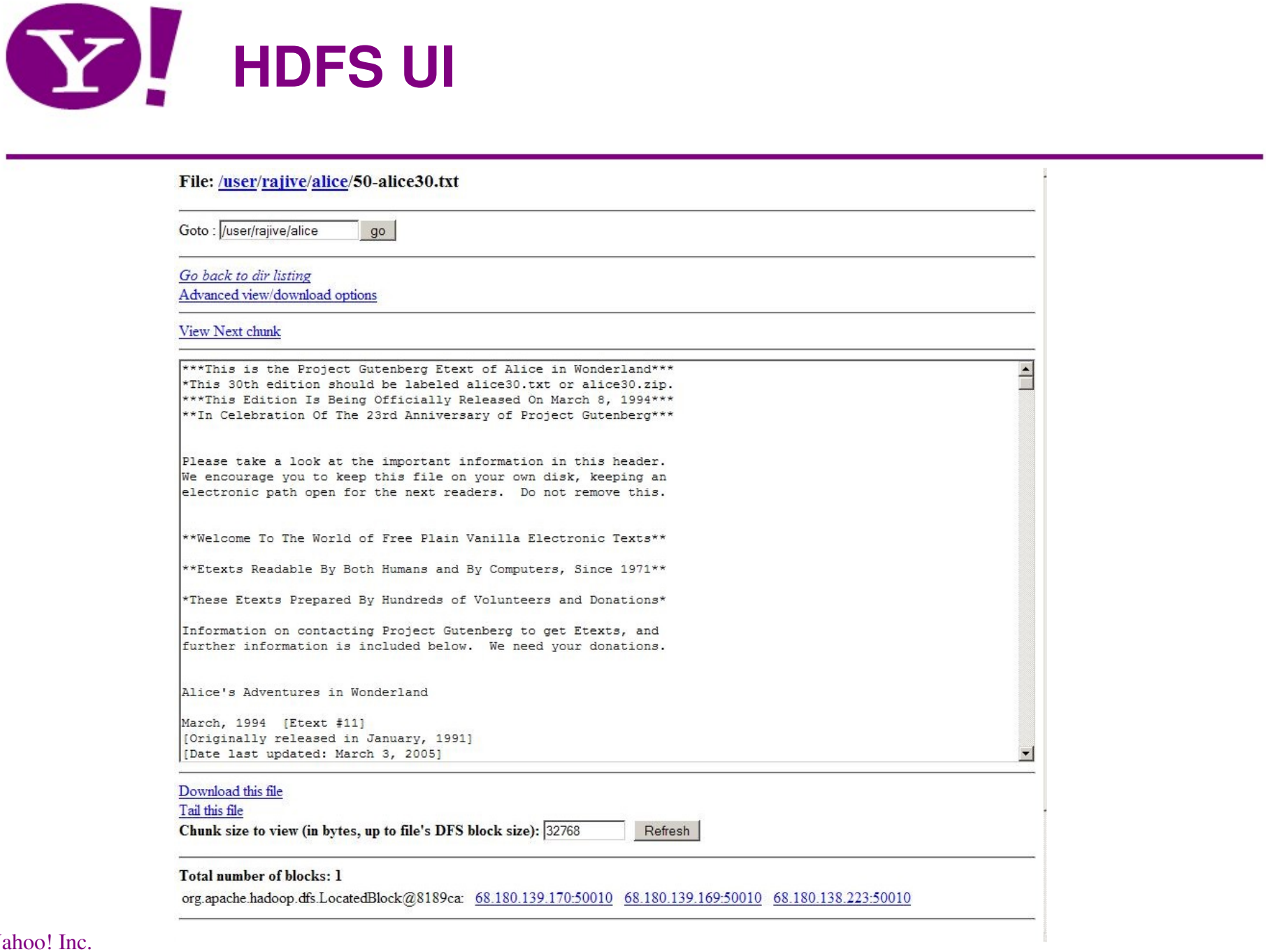
Go back to DFS home

Local logs

Log directory

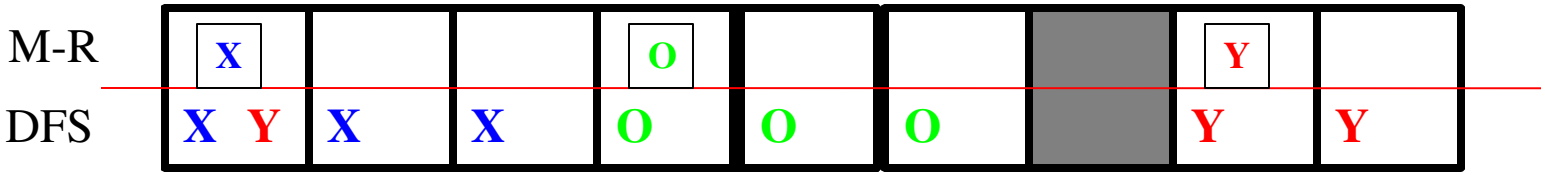
Hadoop, 2007.

HDFS UI

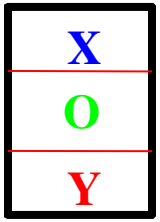

File: /user/rajive/alice/50-alice30.txt
Goto : /user/rajive/alice go
Go back to dir listing
Advanced view/download options
View Next chunk
This is the Project Gutenberg Etext of Alice in Wonderland
*This 30th edition should be labeled alice30.txt or alice30.zip.
This Edition Is Being Officially Released On March 8, 1994
In Celebration Of The 23rd Anniversary of Project Gutenberg*
Please take a look at the important information in this header.
We encourage you to keep this file on your own disk, keeping an
electronic path open for the next readers. Do not remove this.
Welcome To The World of Free Plain Vanilla Electronic Texts
Etexts Readable By Both Humans and By Computers, Since 1971
These Etexts Prepared By Hundreds of Volunteers and Donations
Information on contacting Project Gutenberg to get Etexts, and
further information is included below. We need your donations.
Alice's Adventures in Wonderland
March, 1994 [Etext #11]
[Originally released in January, 1991]
[Date last updated: March 3, 2005]
Download this file
Tail this file
Chunk size to view (in bytes, up to file's DFS block size): 32768 Refresh
Total number of blocks: 1
org.apache.hadoop.dfs.LocatedBlock@8189ca: 68.180.139.170:50010 68.180.139.169:50010 68.180.138.223:50010

Hadoop: Two Services in One

Cluster Nodes run both DFS and M-R
(taking computation to the data)



Input File
(128MB blocks)



HOD (Hadoop on Demand)

- **Map/Reduce is just one programming model**
- **Hadoop is not a resource manager or scheduler**
  - Most sites already have a deployed solution
- **HOD**
  - Bridge between Hadoop and resource managers
  - Currently supports Torque
  - Part of contrib in Hadoop 0.16 release
  - http://hadoop.apache.org/core/docs/current/hod.html

HOD: Provisioning Hadoop

- **Hadoop is submitted like any other job**
- **User specifies number of nodes desired**
- **HOD deals with allocation and setup**
  - Allocates requested nodes
  - Brings up Map/Reduce and (optionally) HDFS daemons
- **User submits Map/Reduce jobs**

HOD Benefits

- **Effective usage of the grid**
  - No need to do ‘social scheduling’
  - No need for static node allocation
- **Automated setup for Hadoop**
  - Users / Ops no longer need to know where and how to bring up daemons

Running Jobs

```
gritgw1004:/grid/0/tmp/rajive$ hod -m 5
HDFS UI on grit1002.yahooresearchcluster.com:50070
Mapred UI on grit1278.yahooresearchcluster.com:55118

Hadoop config file in: /grid/0/kryptonite/hod/tmp/hod-15575-tmp/hadoop-
   site.xml

allocation information:

      1 job tracker node
      4 task tracker nodes
      5 nodes in total


[hod] (rajive) >>
```

Running Jobs

```
[hod] (rajive) >> run jar /grid/0/hadoop/current/hadoop-examples.jar wordcount
    /user/rajive/alice-1.5k /user/rajive/wcout2
08/01/20 05:21:26 WARN mapred.JobConf: Deprecated resource 'mapred-default.xml' is being loaded, please discontinue its
    usage!
08/01/20 05:21:27 INFO mapred.FileInputFormat: Total input paths to process : 1
08/01/20 05:21:30 INFO mapred.JobClient: Running job: job_200801200511_0002
08/01/20 05:21:31 INFO mapred.JobClient:  map 0% reduce 0%
08/01/20 05:21:38 INFO mapred.JobClient:  map 3% reduce 0%
08/01/20 05:21:42 INFO mapred.JobClient:  map 12% reduce 0%
08/01/20 05:21:48 INFO mapred.JobClient:  map 20% reduce 0%
08/01/20 05:22:12 INFO mapred.JobClient:  map 27% reduce 0%
08/01/20 05:22:18 INFO mapred.JobClient:  map 37% reduce 0%
08/01/20 05:22:21 INFO mapred.JobClient:  map 41% reduce 0%
08/01/20 05:22:41 INFO mapred.JobClient:  map 45% reduce 0%
08/01/20 05:22:48 INFO mapred.JobClient:  map 54% reduce 0%
08/01/20 05:22:51 INFO mapred.JobClient:  map 59% reduce 0%
08/01/20 05:22:59 INFO mapred.JobClient:  map 62% reduce 0%
08/01/20 05:23:19 INFO mapred.JobClient:  map 71% reduce 0%
08/01/20 05:23:22 INFO mapred.JobClient:  map 76% reduce 0%
08/01/20 05:23:29 INFO mapred.JobClient:  map 83% reduce 0%
08/01/20 05:23:49 INFO mapred.JobClient:  map 88% reduce 0%
08/01/20 05:23:52 INFO mapred.JobClient:  map 93% reduce 0%
08/01/20 05:23:59 INFO mapred.JobClient:  map 100% reduce 0%
08/01/20 05:24:19 INFO mapred.JobClient:  map 100% reduce 100%
08/01/20 05:24:20 INFO mapred.JobClient: Job complete: job_200801200511_0002
08/01/20 05:24:20 INFO mapred.JobClient: Counters: 11
08/01/20 05:24:20 INFO mapred.JobClient:   Job Counters
08/01/20 05:24:20 INFO mapred.JobClient:     Launched map tasks=2
08/01/20 05:24:20 INFO mapred.JobClient:     Launched reduce tasks=1
08/01/20 05:24:20 INFO mapred.JobClient:   Map-Reduce Framework
08/01/20 05:24:20 INFO mapred.JobClient:     Map input records=5779500
08/01/20 05:24:20 INFO mapred.JobClient:     Map output records=42300000
08/01/20 05:24:20 INFO mapred.JobClient:     Map input bytes=244827000
08/01/20 05:24:20 INFO mapred.JobClient:     Map output bytes=398698500
08/01/20 05:24:20 INFO mapred.JobClient:     Combine input records=42300000
08/01/20 05:24:20 INFO mapred.JobClient:     Combine output records=59080
08/01/20 05:24:20 INFO mapred.JobClient:     Reduce input groups=5908
08/01/20 05:24:20 INFO mapred.JobClient:     Reduce input records=59080
08/01/20 05:24:20 INFO mapred.JobClient:     Reduce output records=5908

[hod] (rajive) >>
```

JobTracker UI

grit1278 Hadoop Map/Reduce Administration

State: RUNNING
Started: Sun Jan 20 05:11:29 UTC 2008
Version: 0.15.1, r596497
Compiled: Tue Nov 20 01:34:59 UTC 2007 by hadoopqa
Identifier: 200801200511

Cluster Summary

Maps	Reduces	Tasks/Node	Total Submissions	Nodes
2	0	2	1	4

Running Jobs

Running Jobs								
Jobid	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed
job_200801200511_0001	rajive	wordcount	0.00%	2	0	0.00%	1	0

Completed Jobs

Completed Jobs
none

Failed Jobs

Failed Jobs
none

Local logs

Log directory, Job Tracker History

Hadoop, 2007.

JobTracker UI

Hadoop job_200801200511_0002 on grit1278

User: rajive
Job Name: wordcount
Job File: /mapredsystem/grit1278.yahooresearchcluster.com/job_200801200511_0002/job.xml
Status: Succeeded
Started at: Sun Jan 20 05:21:30 UTC 2008
Finished at: Sun Jan 20 05:24:19 UTC 2008
Finished in: 2mins, 48sec

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	2	0	0	2	0	0 / 0
reduce	100.00%	1	0	0	1	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	Launched map tasks	0	0	2
	Launched reduce tasks	0	0	1
Map-Reduce Framework	Map input records	5,779,500	0	5,779,500
	Map output records	42,300,000	0	42,300,000
	Map input bytes	244,827,000	0	244,827,000
	Map output bytes	398,698,500	0	398,698,500
	Combine input records	42,300,000	0	42,300,000
	Combine output records	59,080	0	59,080
	Reduce input groups	0	5,908	5,908
	Reduce input records	0	59,080	59,080
	Reduce output records	0	5,908	5,908

Thank you

- Questions?

- Hadoop: http://hadoop.apache.org
- Blog http://developer.yahoo.com/blogs/hadoop
- This presentation: http://public.yahoo.com/rajive/isec2008.pdf
- email: *rajive@yahoo-inc.com*