Hadoop Based Data Analysis Tools on SDSC Gordon Supercomputer

Glenn Lockwood and Mahidhar Tatineni
User Services Group
San Diego Supercomputer Center

XSEDE14, Atlanta
July 14, 2014

Overview

- Hadoop framework extensively used for scalable distributed processing of large datasets. Hadoop is built to process data in orders of several hundred gigabytes to several terabytes (and even petabytes at the extreme end).

- Data sizes are much bigger than the capacities (both disk and memory) of individual nodes. Under Hadoop Distributed Filesystem (HDFS), data is split into chunks which are managed by different nodes.

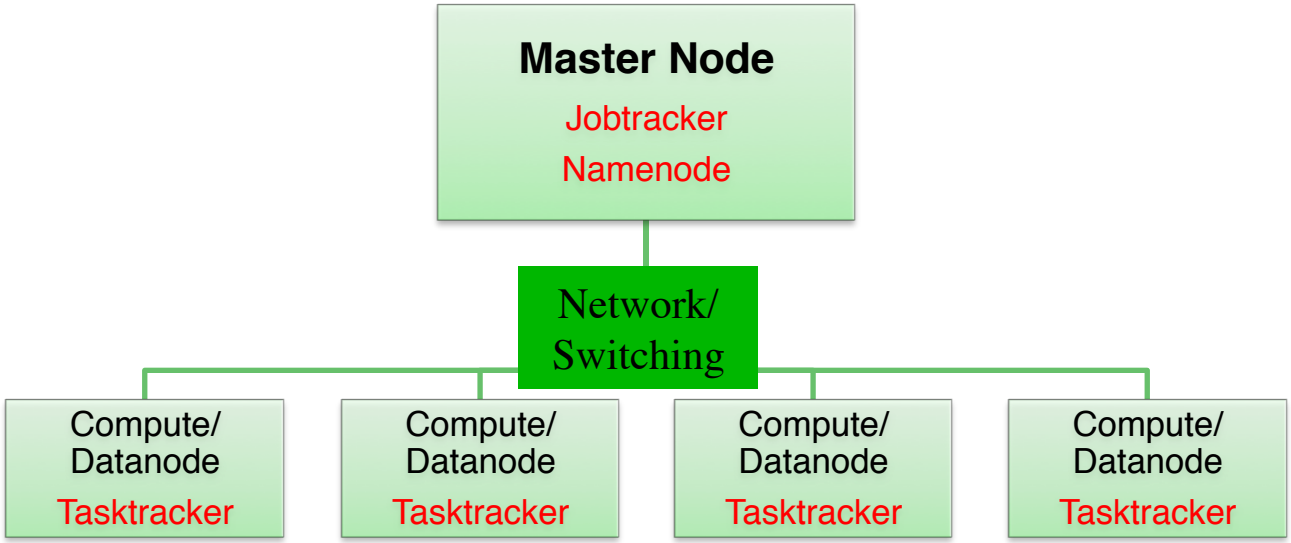
- Data chunks are replicated across several machines to provide redundancy in case of an individual node failure.

- Processing must conform to “Map-Reduce” programming model. Processes are scheduled close to location of data chunks being accessed.

Hadoop: Application Areas

- **Hadoop is widely used in data intensive analysis. Some application areas include:**
  - Log aggregation and processing
  - Video and Image analysis
  - Data mining, Machine learning
  - Indexing
  - Recommendation systems
- **Data intensive scientific applications can make use of the Hadoop MapReduce framework. Application areas include:**
  - Bioinformatics and computational biology
  - Astronomical image processing
  - Natural Language Processing
  - Geospatial data processing
- **Some Example Projects**
  - Genetic algorithms, particle swarm optimization, ant colony optimization
  - Big data for business analytics (class)
  - Hadoop for remote sensing analysis
- **Extensive list online at:**
  - http://wiki.apache.org/hadoop/PoweredBy

Hadoop Architecture



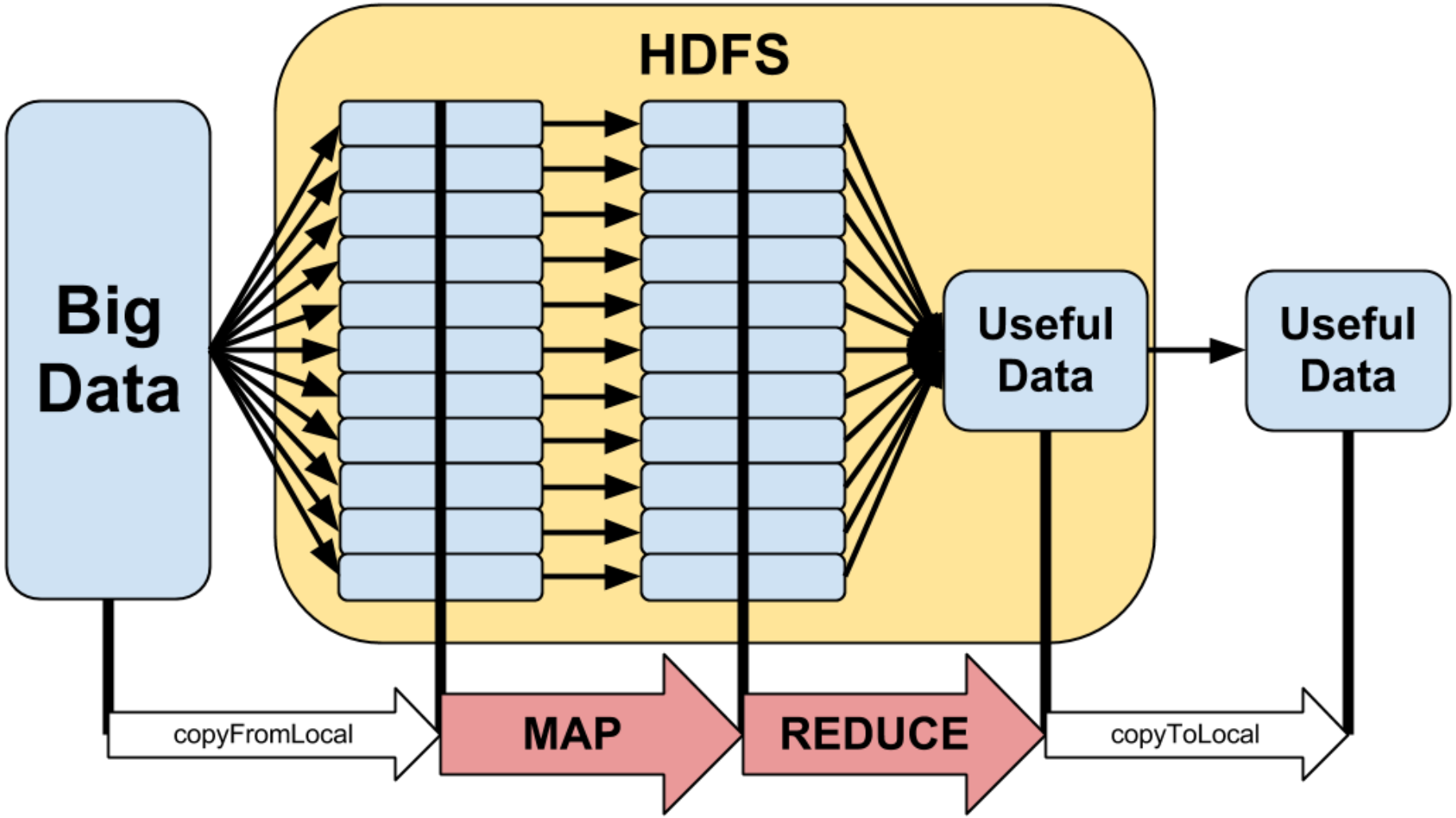
Map/Reduce Framework

- Software to enable distributed computation.
- Jobtracker schedules and manages map/ reduce tasks.
- Tasktracker does the execution of tasks on the nodes.

HDFS – Distributed Filesystem

- Metadata handled by the Namenode.
- Files are split up and stored on datanodes (typically local disk).
- Scalable and fault tolerance.
- Replication is done asynchronously.

Hadoop Workflow



Map Reduce Basics

- The MapReduce algorithm is a scalable (thousands of nodes) approach to process large volumes of data.
- The most important aspect is to restrict arbitrary sharing of data. This minimizes the communication overhead which typically constrains scalability.
- The MapReduce algorithm has two components – Mapping and Reducing.
- The first step involves taking each individual element and mapping it to an output data element using some function.
- The second reducing step involves aggregating the values from step 1 based on a reducer function.
- In the hadoop framework both these functions receive key, value pairs and output key, value pairs. A simple word count example illustrates this process in the following slides.

*Simple Example – From Apache Site**

- **Simple wordcount example.**
- **Code details:**

Functions defined

- Wordcount map class : reads file and isolates each word
- Reduce class : counts words and sums up

Call sequence

- Mapper class
- Combiner class (Reduce locally)
- Reduce class
- Output

*http://hadoop.apache.org/docs/r0.18.3/mapred_tutorial.html

Simple Example – From Apache Site*

- **Simple wordcount example. Two input files.**
- **File 1 contains: Hello World Bye World**
- **File 2 contains: Hello Hadoop Goodbye Hadoop**
- **Assuming we use two map tasks (one for each file).**

- **Step 1: Map read/parse tasks are complete. Result:**

Task 1	Task 2
<Hello, 1>	< Hello, 1>
<World, 1>	< Hadoop, 1>
<Bye, 1>	< Goodbye, 1>
<World, 1>	<Hadoop, 1>

*http://hadoop.apache.org/docs/r0.18.3/mapred_tutorial.html

Simple Example (Contd)

- **Step 2 : Combine on each node, sorted:**

Task 1
<Bye, 1>
<Hello, 1>
<World, 2>

Task 2
< Goodbye, 1>
< Hadoop, 2>
<Hello, 1>

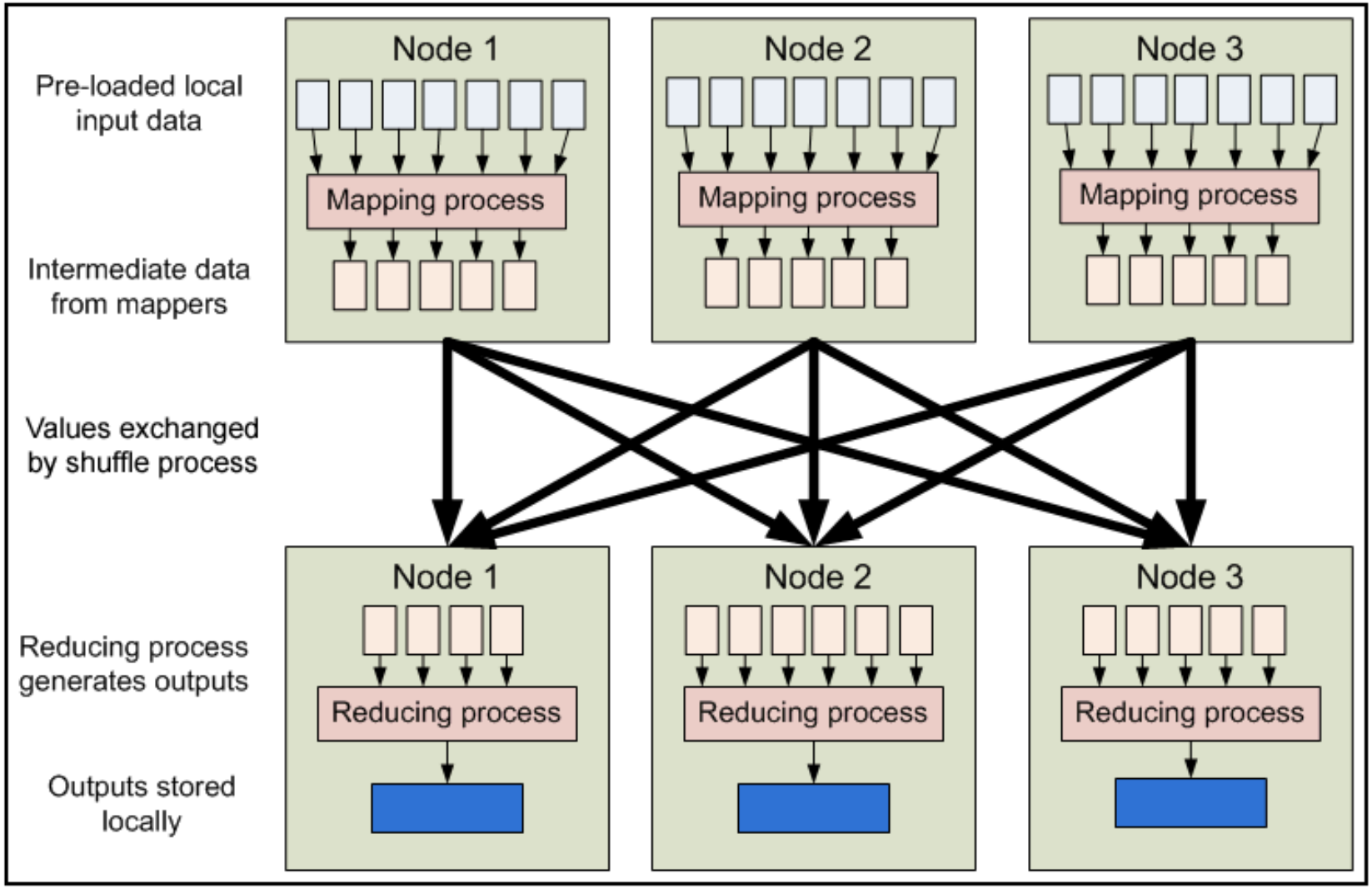
- **Step 3 : Global reduce:**

<Bye, 1>
<Goodbye, 1>
<Hadoop, 2>
<Hello, 2>
<World, 2>

Map/Reduce Execution Process

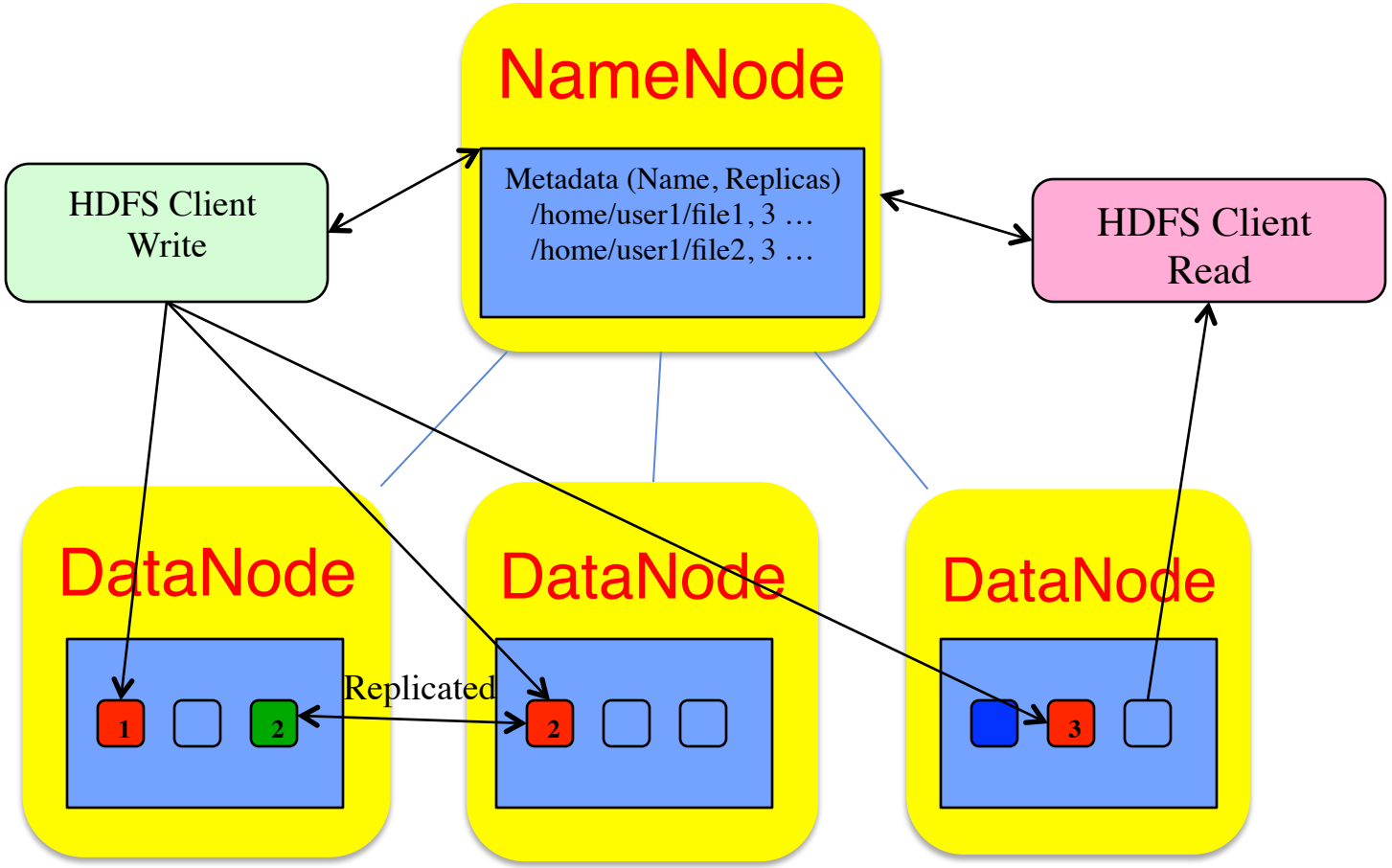
- **Components**
  - Input / Map () / Shuffle / Sort / Reduce () / Output
- **Jobtracker determines number of splits (configurable).**
- **Jobtracker selects compute nodes for tasks based on network proximity to data sources.**
- **Tasktracker on each compute node manages the tasks assigned and reports back to jobtracker when task is complete.**
- **As map tasks complete jobtracker notifies selected task trackers for reduce phase.**
- **Job is completed once reduce phase is complete.**

Hadoop MapReduce – Data pipeline



Ref: http://developer.yahoo.com/hadoop/tutorial/module4.html
"Hadoop Tutorial from Yahoo!" by Yahoo! Inc. is licensed under a Creative Commons Attribution 3.0 Unported License

HDFS Architecture



HDFS Overview

- HDFS is a block-structured filesystem. Files are split up into blocks of a fixed size (configurable) and stored across the cluster on several nodes.
- The metadata for the filesystem is stored on the NameNode. Typically the metadata info is cached in the NameNode's memory for fast access.
- The NameNode provides the list of locations (DataNodes) of blocks for a file and the clients can read directly from the location (without going through the NameNode).
- The HDFS namespace is completely separate from the local filesystems on the nodes. HDFS has its own tools to list, copy, move, and remove files.

HDFS Performance Envelope

- HDFS is optimized for large sequential streaming reads. The default block size is 64MB (in comparison typical block structured filesystems use 4-8KB, and Lustre uses ~1MB). The default replication is 3.
- Typical HDFS applications have a write-once-ready many access model. This is taken into account to simplify the data coherency model used.
- The large block size also means poor random seek times and poor performance with small sized reads/writes.
- Additionally, given the large file sizes, there is no mechanism for local caching (faster to re-read the data from HDFS).
- Given these constraints, HDFS should not be used as a general-purpose distributed filesystem for a wide application base.

HDFS Configuration

- **Configuration files are in the main hadoop config directory. The file "core-site.xml" or "hdfs-site.xml" can be used to change settings.**
- **The config directory can either be replicated across a cluster or placed in a shared filesystem (can be an issue if the cluster gets big).**
- **Configuration settings are a set of key-value pairs:**

```
<property>
        <name>property-name</name>
        <value>property-value</value>
</property>
```

- **Adding the line <final>true</final> prevents user override.**

HDFS Configuration Parameters

- **fs.default.name**

  This is the address which describes the NameNode for the cluster. For example: hdfs://gcn-4-31.ibnet0:54310

- **dfs.data.dir**

  This is the path on the local filesystem in which the DataNode stores data. Defaults to hadoop tmp location.

- **dfs.name.dir**

  This is where the NameNode metadata is stored. Again defaults to hadoop tmp location.

- **dfs.replication**

  Default replication for each block of data. Default value is 3.

- **dfs.blocksize**

  Default block size. Default value is 64MB

- **Lots of other options. See apache site:**

  http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml

HDFS: Listing Files

- **Reminder: Local filesystem commands don't work on HDFS. A simple "ls" will just end up listing your local files.**

- **Need to use HDFS commands. For example:**

```
hadoop dfs -ls /
Warning: $HADOOP_HOME is deprecated.

Found 1 items
drwxr-xr-x   - mahidhar supergroup          0 2013-11-06 15:00 /scratch
```

Copying files into HDFS

First make some directories in HDFS:

```
hadoop dfs -mkdir /user
hadoop dfs -mkdir /user/mahidhar
```

Do a local listing (in this case we have large 50+GB file):

```
ls -lt
total 50331712
-rw-r--r-- 1 mahidhar hpss 51539607552 Nov  6 14:48 test.file
```

Copy it into HDFS:

```
hadoop dfs -put test.file /user/mahidhar/

hadoop dfs -ls /user/mahidhar/
Warning: $HADOOP_HOME is deprecated.

Found 1 items
-rw-r--r--   3 mahidhar supergroup 51539607552 2013-11-06 15:10 /user/mahidhar/test.file
```

DFS Command List

Command	Description
-ls *path*	Lists contents of directory
-lsr *path*	Recursive display of contents
-du *path*	Shows disk usage in bytes
-dus *path*	Summary of disk usage
-mv *src dest*	Move files or directories within HDFS
-cp *src dest*	Copy files or directories within HDFS
-rm *path*	Removes the file or empty directory in HDFS
-rmr *path*	Recursively removes file or directory
-put *localSrc dest (Also –copyFromLocal)*	Copy file from local filesystem into HDFS

Command	Description
-get *src localDest*	Copy from HDFS to local filesystem
-getmerge *src localDest*	Copy from HDFS and merges into single local file
-cat *filename*	Display contents of HDFS file
-tail *file*	Shows the last 1KB of HDFS file on stdout
-chmod [-R]	Change file permissions in HDFS
-chown [-R]	Change ownership in HDFS
-chgrp [-R]	Change group in HDFS
-help	Returns usage info

HDFS dfsadmin options

Command	Description
hadoop dfsadmin -report	Generate status report for HDFS
hadoop dfsadmin –metasave *filename*	Metadata info is saved to file. Cannot be used for restore.
hadoop dfsadmin –safemode *options*	Move HDFS to read-only mode
hadoop dfsadmin -refreshNodes	Changing HDFS membership. Useful if nodes are being removed from cluster.
Upgrade options	Status/ finalization
hadoop dfsadmin -help	Help info

HDFS fsck

```
hadoop fsck -blocks /user/mahidhar/test.file.4GB
Warning: $HADOOP_HOME is deprecated.

FSCK started by mahidhar from /198.202.100.139 for path /user/mahidhar/test.file.4GB at Wed Nov 06 19:32:30 PST 2013
.Status: HEALTHY
 Total size:   4294967296 B
 Total dirs:   0
 Total files:  1
 Total blocks (validated): 64 (avg. block size 67108864 B)
 Minimally replicated blocks:           64 (100.0 %)
 Over-replicated blocks:  0 (0.0 %)
 Under-replicated blocks:               0 (0.0 %)
 Mis-replicated blocks:                 0 (0.0 %)
 Default replication factor:            3
 Average block replication:             3.0
 Corrupt blocks:                        0
 Missing replicas:                      0 (0.0 %)
 Number of data-nodes:                  4
 Number of racks:                       1
FSCK ended at Wed Nov 06 19:32:30 PST 2013 in 36 milliseconds


The filesystem under path '/user/mahidhar/test.file.4GB' is HEALTHY
```

HDFS – Distributed Copy

- **Useful to migrate large numbers of files**
- **Syntax: hadoop distcp *src dest***
- **Source/Destinations can be:**
  - S3 URLs
  - HDFS URL
  - Filesystem [Useful if it's a parallel filesystem like Lustre]

HDFS Architecture: Summary

- HDFS optimized for large block I/O.
- Replication(3) turned on by default.
- HDFS specific commands to manage files and directories. Local filesystem commands will not work in HDFS.
- HDFS parameters set in xml configuration files.
- DFS parameters can also be passed with commands.
- fsck function available for verification.
- Block rebalancing function available.
- Java based HDFS API can be used for access within applications. Library available for functionality in C/C++ applications.

Useful Links

- **Yahoo tutorial on HDFS:**
  http://developer.yahoo.com/hadoop/tutorial/module2.html

- **Apache HDFS guide:**
  http://hadoop.apache.org/docs/r0.18.3/hdfs_design.html

- **Brad Hedlund's article on Hadoop architecture :**
  http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/

Hadoop and HPC

PROBLEM: domain scientists/researchers aren't using Hadoop

- Hadoop is commonly used in the data analytics industry but not so common in domain science academic areas.
- Java is *not* always high-performance
- Hurdles for domain scientists to learn Java, Hadoop tools.

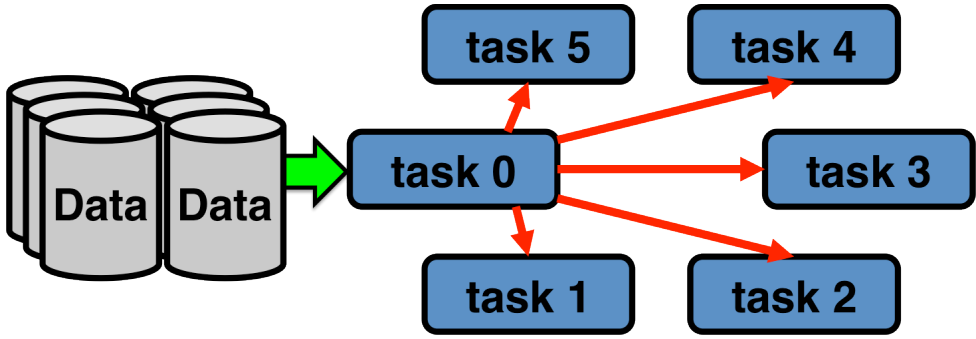
SOLUTION: make Hadoop easier for HPC users

- use existing HPC clusters and software
- use Perl/Python/C/C++/Fortran instead of Java
- make starting Hadoop as easy as possible

Compute: Traditional vs. Data-Intensive

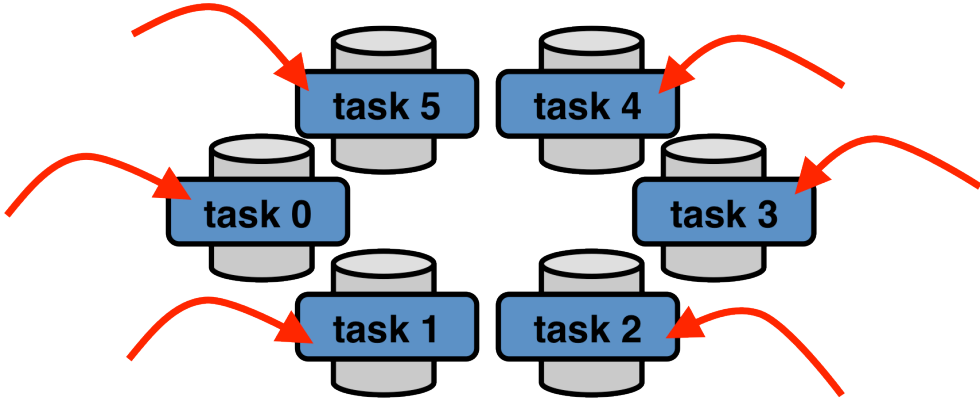
Traditional HPC

- CPU-bound problems
- Solution: OpenMP- and MPI-based parallelism



Data-Intensive

- IO-bound problems
- Solution: Map/reduce-based parallelism



Architecture for Both Workloads

PROs

- High-speed interconnect
- Complementary object storage
- Fast CPUs, RAM
- Less faulty

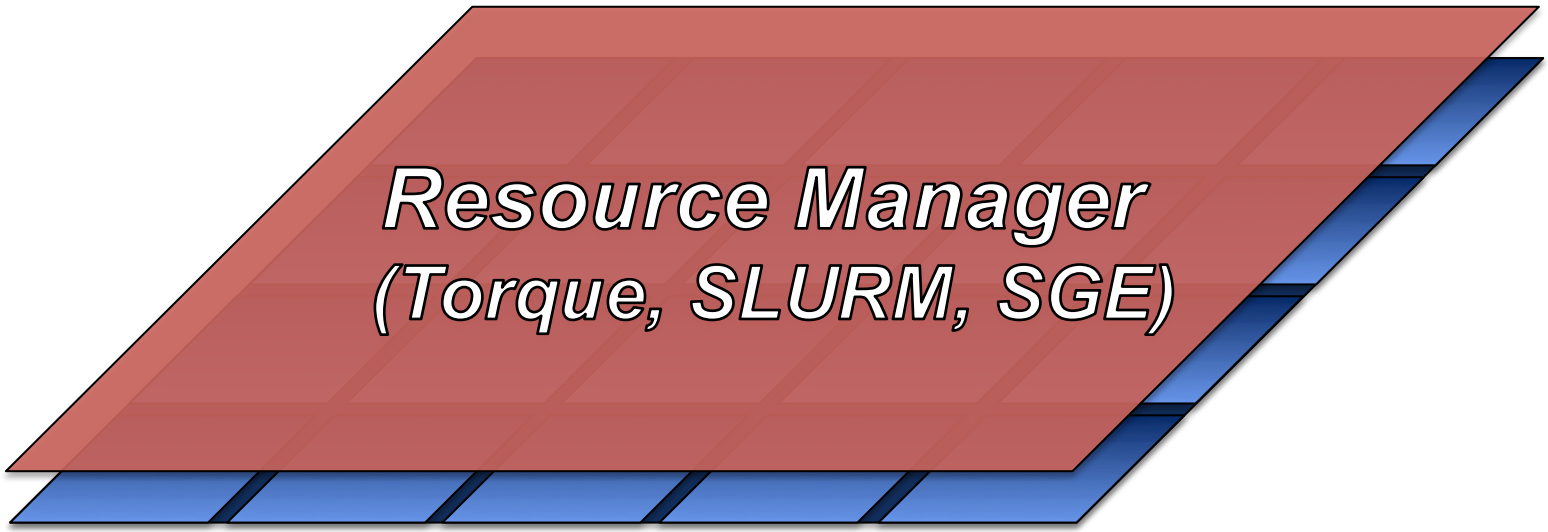
CONs

- Nodes aren't storage-rich
- Transferring data between HDFS and object storage*

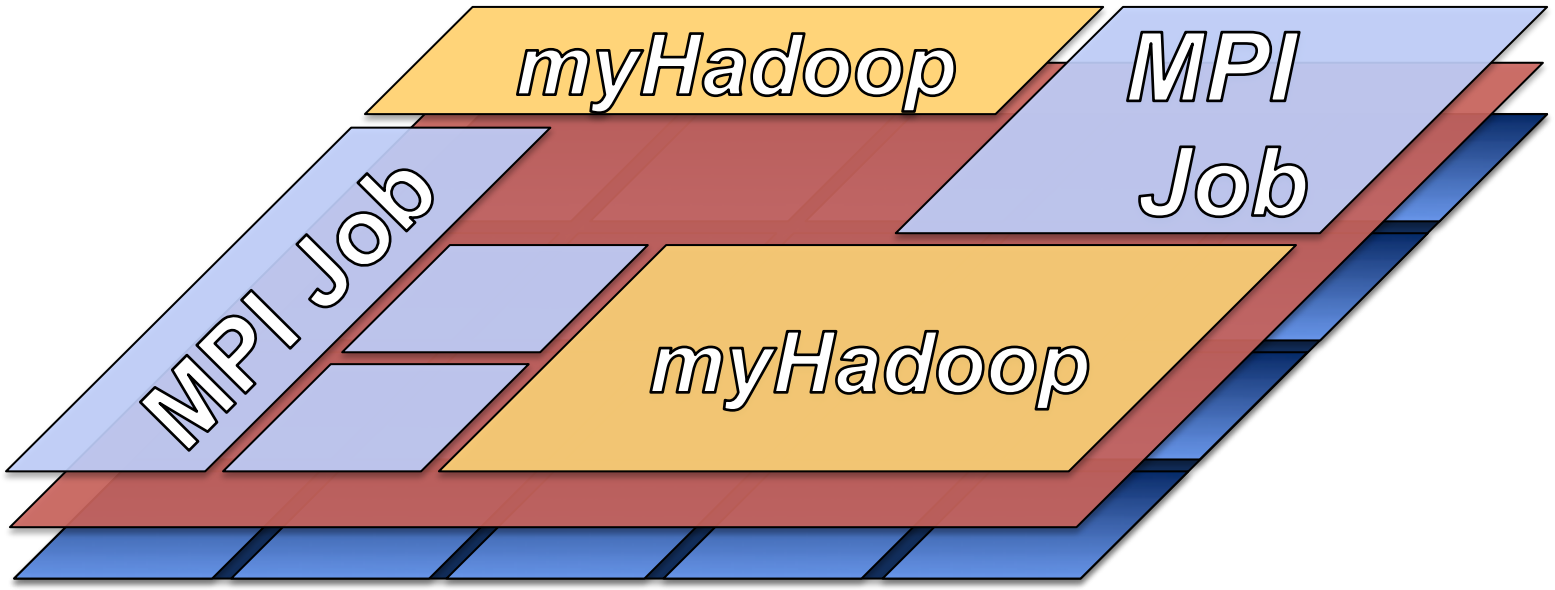
* unless using Lustre, S3, etc backends

Add Data Analysis to Existing Compute Infrastructure

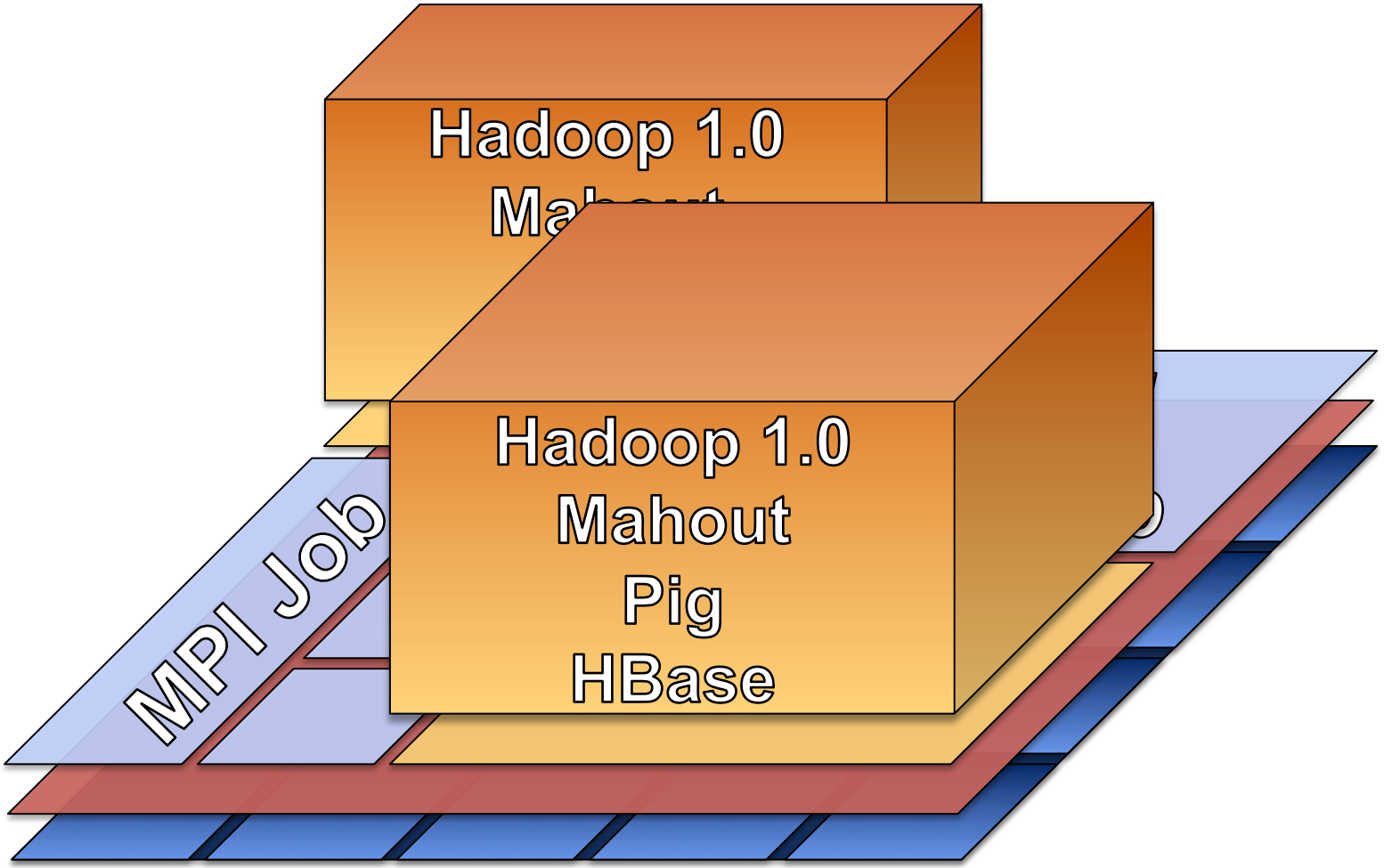
Add Data Analysis to Existing Compute Infrastructure



Add Data Analysis to Existing Compute Infrastructure



Add Data Analysis to Existing Compute Infrastructure



myHadoop – 3-step Install (not reqd. for today's workshop)

1. **Download Apache Hadoop 1.x and myHadoop 0.30**

```
$ wget http://apache.cs.utah.edu/hadoop/common/
   hadoop-1.2.1/hadoop-1.2.1-bin.tar.gz
$ wget http://users.sdsc.edu/~glockwood/files/
   myhadoop-0.30.tar.gz
```

2. **Unpack both Hadoop and myHadoop**

```
$ tar zxvf hadoop-1.2.1-bin.tar.gz
$ tar zxvf myhadoop-0.30.tar.gz
```

3. **Apply myHadoop patch to Hadoop**

```
$ cd hadoop-1.2.1/conf
$ patch < ../myhadoop-0.30/myhadoop-1.2.1.patch
```

myHadoop – 3-step Cluster

1. Set a few environment variables

```
# sets HADOOP_HOME, JAVA_HOME, and PATH
$ module load hadoop
$ export HADOOP_CONF_DIR=$HOME/mycluster.conf
```

2. Run myhadoop-configure.sh to set up Hadoop

```
$ myhadoop-configure.sh -s /scratch/$USER/$PBS_JOBID
```

3. Start cluster with Hadoop's `start-all.sh`

```
$ start-all.sh
```

Advanced Features - Useability

- **System-wide default configurations**
  - `myhadoop-0.30/conf/myhadoop.conf`
  - `MH_SCRATCH_DIR` – specify location of node-local storage for all users
  - `MH_IPOIB_TRANSFORM` – specify regex to transform node hostnames into IP over InfiniBand hostnames
- **Users can remain totally ignorant of scratch disks and InfiniBand**
- **Literally define `HADOOP_CONF_DIR` and run `myhadoop-configure.sh` with no parameters – myHadoop figures out everything else**

Advanced Features - Useability

- **Parallel filesystem support**
  - HDFS on Lustre via myHadoop persistent mode (`-p`)
  - Direct Lustre support (IDH)
  - No performance loss at smaller scales for HDFS on Lustre
- **Resource managers supported in unified framework:**
  - Torque 2.x and 4.x – Tested on SDSC Gordon
  - SLURM 2.6 – Tested on TACC Stampede
  - Grid Engine
  - Can support LSF, PBSpro, Condor easily (need testbeds)

Hadoop-RDMA

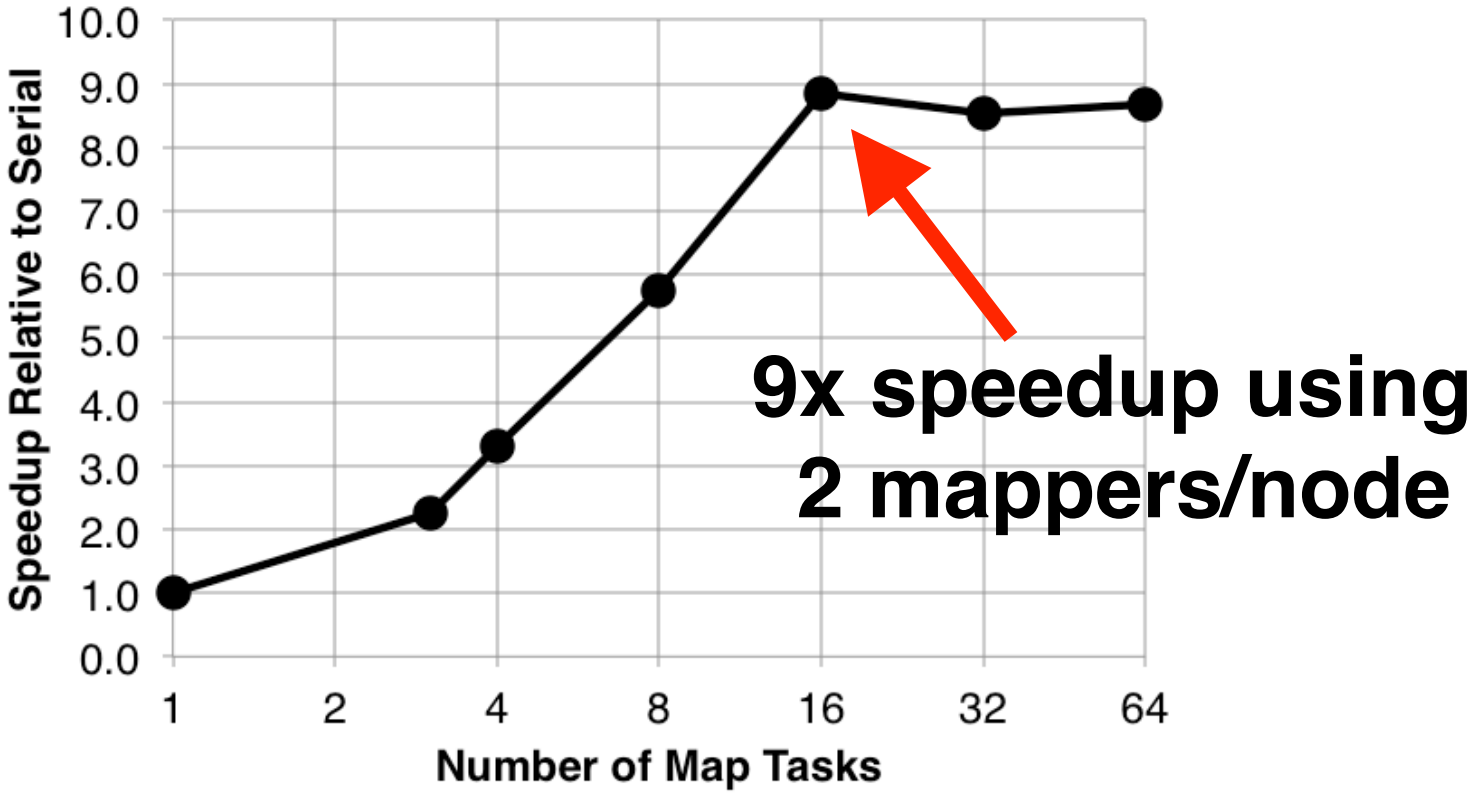
Network-Based Computing Lab (Ohio State University)

- **HDFS, MapReduce, and RPC over native InfiniBand and RDMA over Converged Ethernet (RoCE).**
- **Based on Apache Hadoop 1.2.1**
- **Can use myHadoop to partially set up configuration. Need to add to some of the configuration files.**
- **Location on Gordon:**
  - **/home/diag/opt/hadoop-rdma/hadoop-rdma-0.9.9**
- **More details :**
  - **http://hadoop-rdma.cse.ohio-state.edu/**

Data-Intensive Performance Scaling

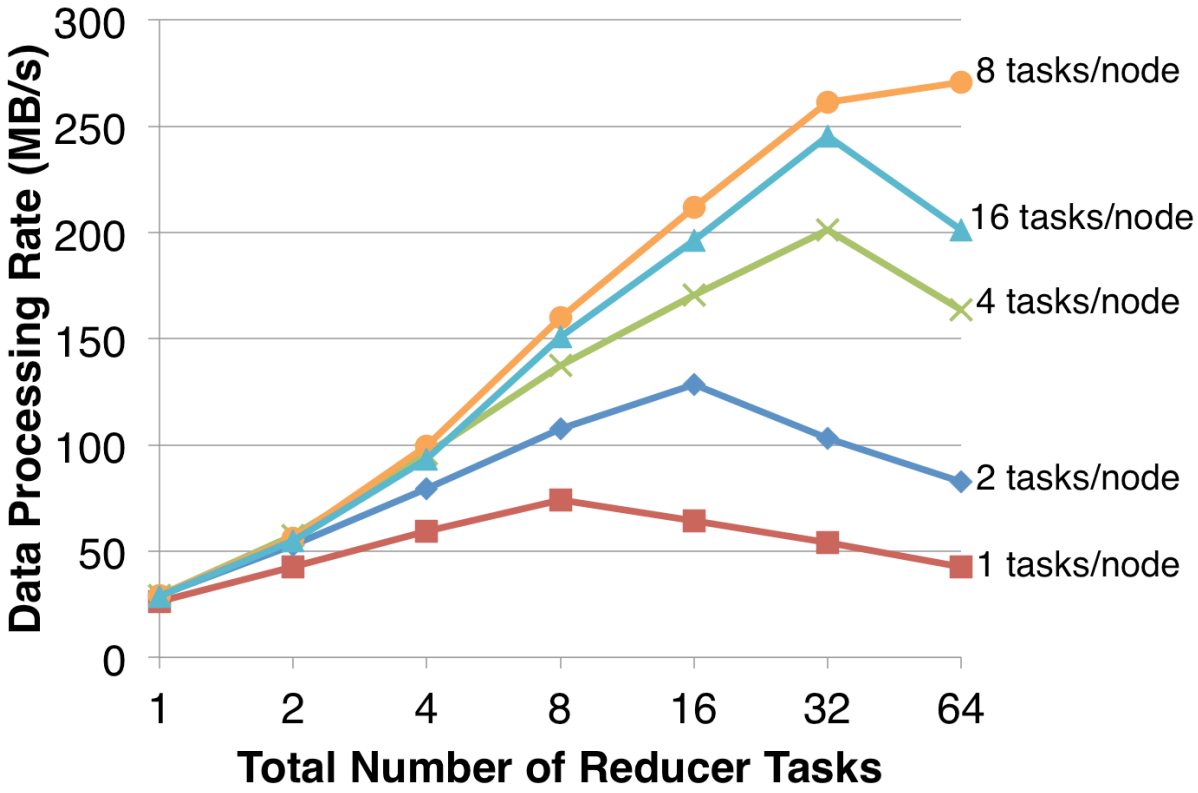
8-node Hadoop cluster on Gordon

8 GB VCF (Variant Calling Format) file



Data-Intensive Performance Scaling

7.7 GB of chemical evolution data, 270 MB/s processing rate



Hadoop Installation & Configuration

USING GORDON

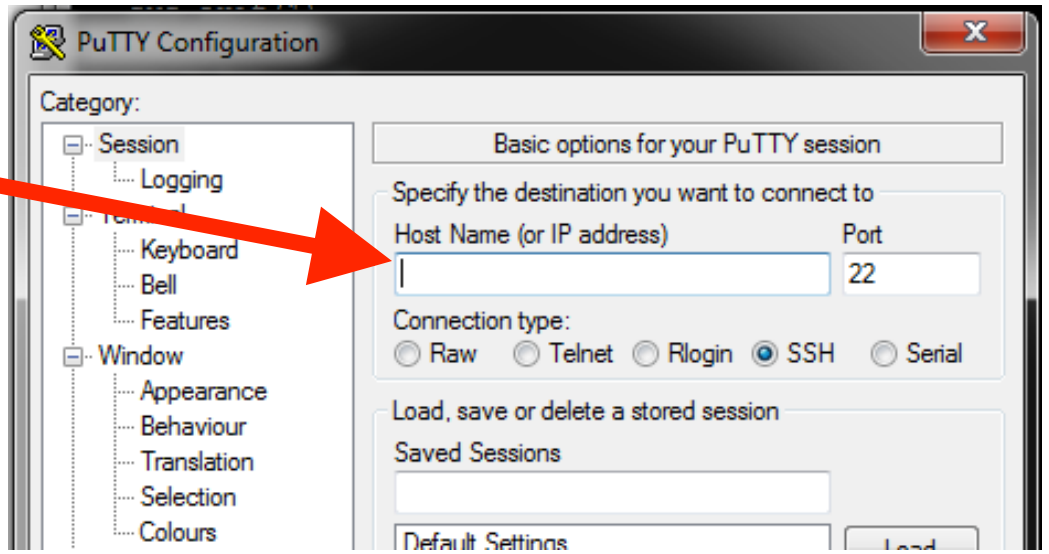
Logging into Gordon

Mac/Linux:

```
ssh etrainXY@gordon.sdsc.edu
```

Windows (PuTTY):

```
gordon.sdsc.edu
```



Example Locations

- **Today's examples are located in the training account home directories:**
  - ~/XSEDE14

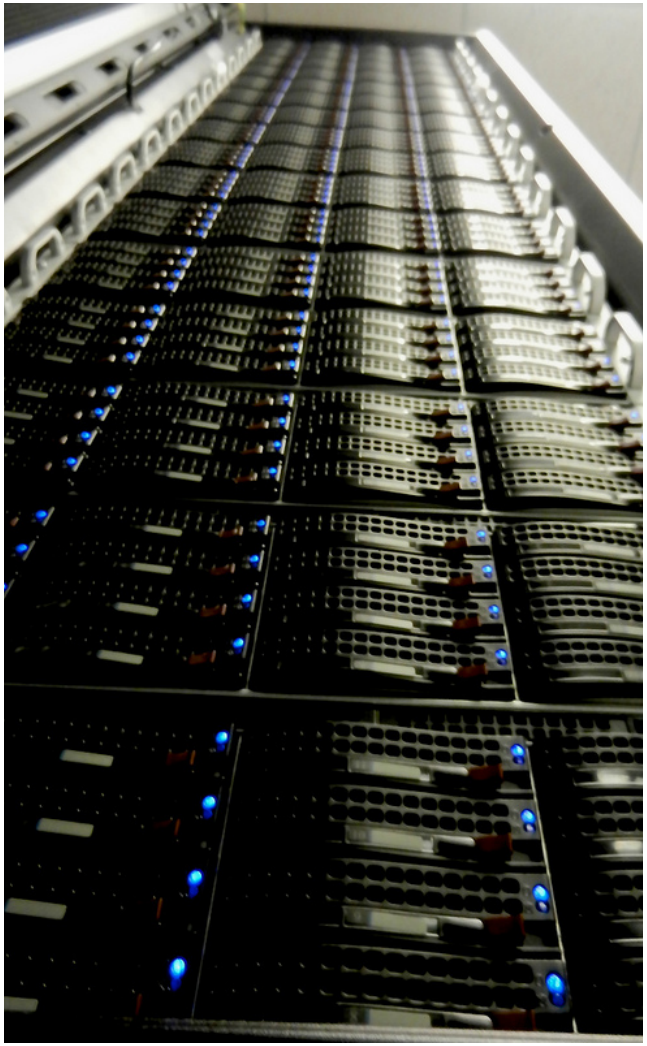
- **Following subdirectories should be visible:**

  ls ~/XSEDE14

  ANAGRAM GUTENBERG mahout Pig2 streaming

What is Gordon?

- **1024 compute nodes, 64 IO nodes**
  - 16 cores of Intel Xeon E5-2670
  - 64 GB of DDR3 DRAM
  - 300 GB enterprise SSD
- **QDR InfiniBand**
  - 40 Gbps QDR4X (cf. 10 Gb Ethernet)
  - two rails
  - 3D torus topology
- **Torque batch system**



Request Nodes with Torque

```
Rocks 6.1 (Emerald Boa)
Profile built 13:17 19-Aug-2013

Kickstarted 13:49 19-Aug-2013

                          WELCOME TO
 _____________________________   ________              ________
__  ___/__  __ \_  ___/_  ____/   __  ____/_________________  /_____________
____ \__  / / /___ \_  /        _  /  __ _  __ \_  ___/  __  /_  __ \_  __ \
___/ /_  /_/ /___/ // /___      / /_/ / / /_/ /  /   / /_/ / / /_/ /  / / /
/____/ /_____/ /____/ \____/     \____/  \____//_/    \__,_/  \____//_/ /_/

*************************************************************************

gordon-ln1:~$ qsub -I -l
nodes=2:ppn=16:native:flash,walltime=4:00:00 -q normal
qsub: waiting for job 1225662.gordon-fe2.local to start
qsub: job 1225662.gordon-fe2.local ready

gcn-3-15:~$
```

Request Nodes with Torque

```
qsub -I -l nodes=2:ppn=16:native:flash,walltime=4:00:00
   -q normal -v Catalina_maxhops=None,QOS=0
```

`-I`	request Interactive access to your nodes
`-l nodes=2`	request two nodes
`ppn=16`	request 16 processors per node
`native`	request native nodes (not vSMP VMs)
`flash`	request nodes with SSDs
`walltime=4:00:00`	reserve these nodes for four hours
`-q normal`	not requesting any special queues

Use the `idev` alias so you don't have to memorize this!

Hadoop Installation & Configuration

GENERAL INSTALLATION AND CONFIGURATION

Installing Hadoop on Linux

1. Download Hadoop tarball from Apache

(http://www.apache.org/dyn/closer.cgi/hadoop/common/)

```
$ wget http://apache.cs.utah.edu/hadoop/common/
hadoop-1.2.1/hadoop-1.2.1-bin.tar.gz
...
Length: 38096663 (36M) [application/x-gzip]
Saving to: “hadoop-1.2.1-bin.tar.gz.2”

85% [=================================>      ] 32,631,817  6.22M/s  eta 2s
```

2. Unpack tarball

```
$ tar zxvf hadoop-1.2.1-bin.tar.gz
hadoop-1.2.1/
...
hadoop-1.2.1/src/contrib/ec2/bin/list-hadoop-clusters
hadoop-1.2.1/src/contrib/ec2/bin/terminate-hadoop-cluster
```

Configure Your First Hadoop Cluster (on Gordon)

Only necessary once per qsub

```
gcn-3-15:~$ myhadoop-configure.sh
myHadoop: Keeping HADOOP_HOME=/home/diag/opt/hadoop/hadoop-1.0.4 from user environment
myHadoop: Setting MH_IPOIB_TRANSFORM='s/$/.ibnet0/' from myhadoop.conf
myHadoop: Setting MH_SCRATCH_DIR=/scratch/$USER/$PBS_JOBID from myhadoop.conf
myHadoop: Keeping HADOOP_CONF_DIR=/home/glock/mycluster from user environment
myHadoop: Using HADOOP_HOME=/home/diag/opt/hadoop/hadoop-1.0.4
myHadoop: Using MH_SCRATCH_DIR=/scratch/glock/1225034.gordon-fe2.local
myHadoop: Using JAVA_HOME=/usr/java/latest
myHadoop: Generating Hadoop configuration in directory in /home/glock/mycluster...
myHadoop: Designating gcn-3-15.ibnet0 as master node
myHadoop: The following nodes will be slaves:
gcn-3-15.ibnet0
gcn-3-24.ibnet0
...
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at gcn-3-15.sdsc.edu/198.202.100.78
************************************************************/
```

myHadoop

- **Interface between batch system (qsub) and Hadoop**
- **`myhadoop-configure.sh` is *not* a part of Hadoop**
  - gets node names from batch environment
  - configures InfiniBand support
  - generates Hadoop config files
  - formats namenode so HDFS is ready to go

Configuring Hadoop

- **$HADOOP_CONF_DIR *is* the cluster**
  - all of your cluster's global configurations
  - all of your cluster's nodes
  - all of your cluster's node configurations**
- **`echo $HADOOP_CONF_DIR` to see your cluster's configuration location**
- **`cd $HADOOP_CONF_DIR`**

```
$ ls
capacity-scheduler.xml      hadoop-policy.xml      myhadoop.conf
configuration.xsl           hdfs-site.xml          slaves
core-site.xml               log4j.properties       ssl-client.xml.example
fair-scheduler.xml          mapred-queue-acls.xml  ssl-server.xml.example
hadoop-env.sh               mapred-site.xml        taskcontroller.cfg
hadoop-metrics2.properties  masters
```

core-site.xml

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/scratch/glock/1225034.gordon-fe2.local/tmp</value>
  <description>A base for other temporary directories.</description>
</property>

<property>
  <name>fs.default.name</name>
  <value>hdfs://gcn-3-15.ibnet0:54310</value>
  <description>The name of the default file system. </description>
</property>
```

core-site.xml

```
<property>
  <name>io.file.buffer.size</name>
  <value>131072</value>
  <description>The size of buffer for use in sequence files. The
     size of this buffer should probably be a multiple of hardware
     page size (4096 on Intel x86), and it determines how much data
     is buffered during read and write operations.
  </description>
</property>
```

More parameters and their default values:

http://hadoop.apache.org/docs/r1.0.4/core-default.html

hdfs-site.xml

```
<property>
  <name>dfs.name.dir</name>
  <value>/scratch/glock/1225034.gordon-fe2.local/namenode_data
         </value>
  <final>true</final>
</property>

<property>
  <name>dfs.data.dir</name>
  <value>/scratch/glock/1225034.gordon-fe2.local/hdfs_data</value>
  <final>true</final>
</property>
```

hdfs-site.xml

```
<property>
  <name>dfs.replication</name>
  <value>3</value>
</property>

<property>
  <name>dfs.block.size</name>
  <value>67108864</value>
</property>
```

More parameters and their default values:

http://hadoop.apache.org/docs/r1.0.4/hdfs-default.html

mapred-site.xml

```
<property>
  <name>mapred.job.tracker</name>
  <value>gcn-3-15.ibnet0:54311</value>
  <description>The host and port that the MapReduce job tracker runs
     at.  If "local", then jobs are run in-process as a single map
     and reduce task.
  </description>
</property>

<property>
  <name>mapred.local.dir</name>
  <value>/scratch/glock/1225034.gordon-fe2.local/mapred_scratch
     </value>
  <final>true</final>
</property>
```

mapred-site.xml

```
<property>
  <name>io.sort.mb</name>
  <value>650</value>
  <description>Higher memory-limit while sorting data.</description>
</property>

<property>
  <name>mapred.map.tasks</name>
  <value>4</value>
  <description>The default number of map tasks per job.</description>
</property>

<property>
  <name>mapred.reduce.tasks</name>
  <value>4</value>
  <description>The default number of reduce tasks per job. </description>
</property>
```

mapred-site.xml

```
<property>
  <name>mapred.tasktracker.map.tasks.maximum</name>
  <value>8</value>
  <description>The maximum number of map tasks that will be run
     simultaneously by a task tracker.</description>
</property>

<property>
  <name>mapred.tasktracker.reduce.tasks.maximum</name>
  <value>8</value>
  <description>The maximum number of reduce tasks that will be run
     simultaneously by a task tracker.</description>
</property>
```

More parameters and their default values:

http://hadoop.apache.org/docs/r1.0.4/mapred-default.html

hadoop-env.sh

- **Establishes environment variables for all Hadoop components**
- **Essentials:**
  - `HADOOP_LOG_DIR` – location of Hadoop logs
  - `HADOOP_PID_DIR` – location of Hadoop PID files
  - `JAVA_HOME` – location of Java that Hadoop should use
- **Other common additions**
  - `LD_LIBRARY_PATH` - for mappers/reducers
  - `HADOOP_CLASSPATH` - for mappers/reducers
  - `_JAVA_OPTIONS` - to hack global Java options

Hadoop Installation & Configuration

HANDS-ON CONFIGURATION

masters / slaves

- **Used by `start-all.sh` and `stop-all.sh` *Hadoop control scripts***

```
$ cat masters
gcn-3-15.ibnet0

$ cat slaves
gcn-3-15.ibnet0
gcn-3-24.ibnet0
```

- **`masters` – secondary namenodes**
- **`slaves` – datanodes + tasktrackers**
- **namenode and jobtracker are launched on localhost**

Launch Your First Hadoop Cluster (on Gordon)

```
$ start-all.sh
starting namenode, logging to /scratch/glock/1225034.gordon-fe2.local/logs/hadoop-gloc
gcn-3-15.ibnet0: starting datanode, logging to /scratch/glock/1225034.gordon-fe2.loca
gcn-3-24.ibnet0: starting datanode, logging to /scratch/glock/1225034.gordon-fe2.loca
gcn-3-15.ibnet0: starting secondarynamenode, logging to /scratch/glock/1225034.gordon
starting jobtracker, logging to /scratch/glock/1225034.gordon-fe2.local/logs/hadoop-gl
gcn-3-15.ibnet0: starting tasktracker, logging to /scratch/glock/1225034.gordon-fe2.l
gcn-3-24.ibnet0: starting tasktracker, logging to /scratch/glock/1225034.gordon-fe2.l
```

- **Verify that namenode and datanodes came up:**

```
$ hadoop dfsadmin -report
Configured Capacity: 599845068800 (558.65 GB)
Present Capacity: 599776509982 (558.59 GB)
DFS Remaining: 599776493568 (558.59 GB)
DFS Used: 16414 (16.03 KB)
DFS Used%: 0%
Under replicated blocks: 1
Blocks with corrupt replicas: 0
...
```

Try it Yourself: Change Replication

1. Shut down cluster to change configuration

```
$ stop-all.sh
gcn-3-15.ibnet0: stopping tasktracker
gcn-3-24.ibnet0: stopping tasktracker
stopping namenode
gcn-3-15.ibnet0: stopping datanode
gcn-3-24.ibnet0: stopping datanode
gcn-3-24.ibnet0: stopping secondarynamenode
```

2. Change `dfs.replication` setting in `hdfs-site.xml` by adding the following:

```
<property>
  <name>dfs.replication</name>
  <value>2</value>
</property>
```

3. Restart (`start-all.sh`) and re-check

Try it Yourself: Change Replication

Are the under-replicated blocks gone?

What does this mean for actual runtime behavior?

Run Some Simple Commands

Get a small set of data (a bunch of classic texts) and copy into HDFS:

```
$ ls –lh gutenberg.txt
-rw-r--r-- 1 glock sdsc 407M Mar 16 14:09 gutenberg.txt

$ hadoop dfs -mkdir data

$ hadoop dfs -put gutenberg.txt data/

$ hadoop dfs -ls data
Found 1 items
-rw-r--r--   3 glock supergroup  426138189 2014-03-16 14:21 /user/glock/data/gutenberg.txt
```

Run Your First Map/Reduce Job

```
$ hadoop jar $HADOOP_HOME/hadoop-examples-1.2.1.jar
    wordcount data wordcount-output
14/03/16 14:24:18 INFO input.FileInputFormat: Total input paths to process : 1
14/03/16 14:24:18 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/03/16 14:24:18 WARN snappy.LoadSnappy: Snappy native library not loaded
14/03/16 14:24:18 INFO mapred.JobClient: Running job: job_201403161202_0010
14/03/16 14:24:19 INFO mapred.JobClient:  map 0% reduce 0%
14/03/16 14:24:35 INFO mapred.JobClient:  map 15% reduce 0%
...
14/03/16 14:25:23 INFO mapred.JobClient:  map 100% reduce 28%
14/03/16 14:25:32 INFO mapred.JobClient:  map 100% reduce 100%
14/03/16 14:25:37 INFO mapred.JobClient: Job complete: job_201403161202_0010
14/03/16 14:25:37 INFO mapred.JobClient: Counters: 29
14/03/16 14:25:37 INFO mapred.JobClient:   Job Counters
14/03/16 14:25:37 INFO mapred.JobClient:     Launched reduce tasks=1
...
14/03/16 14:25:37 INFO mapred.JobClient:     Launched map tasks=7
14/03/16 14:25:37 INFO mapred.JobClient:     Data-local map tasks=7
```

Running Hadoop Jobs

```
hadoop jar $HADOOP_HOME/hadoop-examples-1.2.1.jar
   wordcount data wordcount-output
```

`hadoop jar`	launch a map/reduce job
`$HADOOP_HOME/hadoop-examples-1.2.1.jar`	map/reduce application jar
`wordcount`	command-line argument for jarfile
`data`	location of input data (file or directory)
`wordcount-output`	location to dump job output

Check Job Output

```
$ hadoop dfs -ls wordcount-output
Found 3 items
... glock supergroup ... /user/glock/wordcount-output/_SUCCESS
... glock supergroup ... /user/glock/wordcount-output/_logs
... glock supergroup ... /user/glock/wordcount-output/part-r-00000
... glock supergroup ... /user/glock/wordcount-output/part-r-00001
...
```

- `_SUCCESS` – if exists, job finished successfully
- `_logs` – directory containing job history
- `part-r-00000` – output of reducer #0
- `part-r-00001` – output of reducer #1
- `part-r-0000`*n* – output of *n*th reducer

Check Job Output

```
$ hadoop job -history wordcount-output

Hadoop job: 0010_1395005058622_glock
=====================================
...
Submitted At: 16-Mar-2014 14:24:18
Launched At: 16-Mar-2014 14:24:18 (0sec)
Finished At: 16-Mar-2014 14:25:36 (1mins, 18sec)
...
Task Summary
============================
Kind    Total   Successful  Failed  Killed  StartTime   FinishTime

Setup   1   1       0   0   16-Mar-2014 14:24:18    16-Mar-2014 14:24:23 (4sec)
Map     7   7       0   0   16-Mar-2014 14:24:24    16-Mar-2014 14:25:14 (49sec)
Reduce  1   1       0   0   16-Mar-2014 14:24:48    16-Mar-2014 14:25:29 (40sec)
Cleanup 1   1       0   0   16-Mar-2014 14:25:30    16-Mar-2014 14:25:35 (4sec)

============================
```

Retrieve Job Output

```
$ hadoop dfs -get wordcount-output/part* .

$ ls -lrtph
...
-rw-r--r-- 1 glock sdsc  24M Mar 16 14:31 part-r-00000

$ hadoop dfs -rmr wordcount-output
Deleted hdfs://gcn-3-15.ibnet0:54310/user/glock/wordcount-output

$ hadoop dfs -ls
Found 1 items
... glock supergroup ... /user/glock/data
```

Silly First Example

1. How many mappers were used?
2. How many reducers were used?
3. If Map and Reduce phases are real calculations, and Setup and Cleanup are "job overhead," what fraction of time was spent on overhead?

User-Level Configuration Changes

- **Bigger HDFS block size**
- **More reducers**

```
$ hadoop dfs -Ddfs.block.size=134217728 –put
    ./gutenberg.txt data/gutenberg-128M.txt

$ hadoop fsck -block /user/glock/data/gutenberg-128M.txt
...
Total blocks (validated):    4 (avg. block size 106534547 B)
...

$ hadoop jar $HADOOP_HOME/hadoop-examples-1.2.1.jar
    wordcount -Dmapred.reduce.tasks=4
    data/gutenberg-128M.txt output-128M
```

User-Level Configuration Changes

1. How many mappers were used before and after specifying the user-level tweaks? What caused this change?
2. How much speedup did these tweaks give? Any ideas why?
3. What happens if you use `mapred.reduce.tasks=8` or `=16`? Any ideas why?
4. `hadoop dfs -ls output-128M` –How many output files are there now? Why?

TestDFSIO

- Test to verify HDFS read/write performance
- Generates its own input via `-write` test
- Make 8 files, each 1024 MB large:

```
$ hadoop jar $HADOOP_HOME/hadoop-test-1.2.1.jar TestDFSIO
    -write -nrFiles 8 -fileSize 1024
14/03/16 16:17:20 INFO fs.TestDFSIO: nrFiles = 8
14/03/16 16:17:20 INFO fs.TestDFSIO: fileSize (MB) = 1024
14/03/16 16:17:20 INFO fs.TestDFSIO: bufferSize = 1000000
14/03/16 16:17:20 INFO fs.TestDFSIO: creating control file: 1024 mega bytes, 8 files
14/03/16 16:17:20 INFO fs.TestDFSIO: created control files for: 8 files
...
14/03/16 16:19:04 INFO fs.TestDFSIO: ----- TestDFSIO ----- : write
14/03/16 16:19:04 INFO fs.TestDFSIO:            Date & time: Sun Mar 16 16:19:04 PDT 2014
14/03/16 16:19:04 INFO fs.TestDFSIO:        Number of files: 8
14/03/16 16:19:04 INFO fs.TestDFSIO: Total MBytes processed: 8192
14/03/16 16:19:04 INFO fs.TestDFSIO:      Throughput mb/sec: 39.83815748521631
14/03/16 16:19:04 INFO fs.TestDFSIO: Average IO rate mb/sec: 139.90382385253906
14/03/16 16:19:04 INFO fs.TestDFSIO:  IO rate std deviation: 102.63743717054572
14/03/16 16:19:04 INFO fs.TestDFSIO:     Test exec time sec: 103.64
```

TestDFSIO

- Test the read performance
- use `-read` instead of `-write`:
- Why such a big performance difference?

```
$ hadoop jar $HADOOP_HOME/hadoop-test-1.2.1.jar TestDFSIO
    -read -nrFiles 8 -fileSize 1024
14/03/16 16:19:59 INFO fs.TestDFSIO: nrFiles = 8
14/03/16 16:19:59 INFO fs.TestDFSIO: fileSize (MB) = 1024
14/03/16 16:19:59 INFO fs.TestDFSIO: bufferSize = 1000000
14/03/16 16:19:59 INFO fs.TestDFSIO: creating control file: 1024 mega bytes, 8 files
14/03/16 16:20:00 INFO fs.TestDFSIO: created control files for: 8 files
...
14/03/16 16:20:40 INFO fs.TestDFSIO: ----- TestDFSIO ----- : read
14/03/16 16:20:40 INFO fs.TestDFSIO:            Date & time: Sun Mar 16 16:20:40 PDT 2014
14/03/16 16:20:40 INFO fs.TestDFSIO:        Number of files: 8
14/03/16 16:20:40 INFO fs.TestDFSIO: Total MBytes processed: 8192
14/03/16 16:20:40 INFO fs.TestDFSIO:      Throughput mb/sec: 276.53254118282473
14/03/16 16:20:40 INFO fs.TestDFSIO: Average IO rate mb/sec: 280.89404296875
14/03/16 16:20:40 INFO fs.TestDFSIO:  IO rate std deviation: 30.524924236925283
14/03/16 16:20:40 INFO fs.TestDFSIO:     Test exec time sec: 40.418
```

HDFS Read/Write Performance

1. **What part of the HDFS architecture might explain the difference in how long it takes to *read* 8x1 GB files vs. *write* 8x1 GB files?**
2. **What configuration have we covered that might alter this performance gap?**
3. **Adjust this parameter, restart the cluster, and re-try. How close can you get the read/write performance gap?**

TeraSort – Heavy-Duty Map/Reduce Job

Standard benchmark to assess performance of Hadoop's MapReduce and HDFS components

- `teragen` – **Generate *a lot* of random input data**
- `terasort` – **Sort teragen's output**
- `teravalidate` – **Verify that terasort's output is sorted**

Running TeraGen

```
hadoop jar $HADOOP_HOME/hadoop-examples-1.2.1.jar
   teragen 100000000 terasort-input
```

`teragen`	launch the teragen application
`100000000`	how many 100-byte records to generate (100,000,000 ~ 9.3 GB)
`terasort-input`	location to store output data to be fed into terasort

Running TeraSort

```
$ hadoop jar $HADOOP_HOME/hadoop-examples-1.2.1.jar teragen
    100000000 terasort-input
Generating 100000000 using 2 maps with step of 50000000
...

$ hadoop jar $HADOOP_HOME/hadoop-examples-1.2.1.jar
    terasort terasort-input terasort-output
14/03/16 16:43:00 INFO terasort.TeraSort: starting
14/03/16 16:43:00 INFO mapred.FileInputFormat: Total input
    paths to process : 2
...
```

TeraSort Default Behavior

Default settings are suboptimal:

- `teragen` – use 2 maps, so TeraSort input split between 2 files
- `terasort` – use 1 reducer so reduction is serial

Try optimizing TeraSort:

- Double `dfs.block.size` to 134217728 (128MB)
- Increase `mapred.tasktracker.map.tasks.maximum` and `mapred.tasktracker.reduce.tasks.maximum` to increase concurrency
- Launch with more reducers (`-Dmapred.reduce.tasks=XX`)

Summary of Commands

```
### Request two nodes for four hours on Gordon
$ qsub -I -l nodes=2:ppn=16:native:flash,walltime=4:00:00 -q normal
### Configure $HADOOP_CONF_DIR on Gordon
$ myhadoop-configure.sh
### Hadoop control script to start all nodes
$ start-all.sh
### Verify HDFS is online
$ hadoop dfsadmin -report
### Copy file to HDFS
$ hadoop dfs -put somelocalfile hdfsdir/
### View file information on HDFS
$ hadoop fsck -block hdfsdir/somelocalfile
### Run a map/reduce job
$ hadoop jar somejarfile.jar -option1 -option2
### View job info after it completes
$ hadoop job -history hdfsdir/outputdir
### Shut down all Hadoop nodes
$ stop-all.sh
### Copy logfiles back from nodes
$ myhadoop-cleanup.sh
```

Anagram Example

- **Source:**

https://code.google.com/p/hadoop-map-reduce-examples/wiki/Anagram_Example

- **Uses Map-Reduce approach to process a file with a list of words, and identify all the anagrams in the file**

- **Code is written in Java. Example has already been compiled and the resulting jar file is in the example directory.**

Anagram – Map Class (Detail)

- String word = value.toString();

Convert the word to a string

- char[] wordChars = word.toCharArray();

Assign to character array

- Arrays.sort(wordChars);

Sort the array of characters

- String sortedWord = new String(wordChars);

Create new string with sorted characters

- sortedText.set(sortedWord);
  orginalText.set(word);
  outputCollector.collect(sortedText, orginalText);

Prepare and output the sorted text string (serves as the key), and the original test string.

Anagram – Map Class (Detail)

- Consider file with list of words : alpha, hills, shill, truck
- Alphabetically sorted words : aahlp, hills, hills, ckrtu
- Hence after the Map Step is done, the following key pairs would be generated:

(aahlp,alpha)

(hills,hills)

(hills,shill)

(ckrtu,truck)

Anagram Example – Reducer (Detail)

- while(anagramValues.hasNext())

```
{
    Text anagam = anagramValues.next();
    output = output + anagam.toString() + "~";
}
```

Iterate over all the values for a key. hasNext() is a Java method that allows you to do this.

We are also creating an output string which has all the words separated with ~.

- StringTokenizer outputTokenizer = new StringTokenizer(output,"~");

StringTokenizer class allows you to store this delimited string and has functions that allow you to count the number of tokens.

- if(outputTokenizer.countTokens()>=2)

```
{
    output = output.replace("~", ",");
    outputKey.set(anagramKey.toString());
    outputValue.set(output);
    results.collect(outputKey, outputValue);
}
```

We output the anagram key and the word lists if the number of tokens is >=2 (i.e. we have an anagram pair).

Anagram Reducer Class (Detail)

- For our example set, the input to the Reducers is:

  (aahlp,alpha)
  (hills,hills)
  (hills,shill)
  (ckrtu,truck)

- The only key with #tokens >=2 is <hills>.

- Hence, the Reducer output will be:

  hills hills, shill,

Anagram Example – Submit Script

```
#!/bin/bash
#PBS -N Anagram
#PBS -q normal
#PBS -l nodes=2:ppn=16:native
#PBS -l walltime=01:00:00
#PBS -m e
#PBS -M youremail@xyz.com
#PBS -V
#PBS -v Catalina_maxhops=None
cd $PBS_O_WORKDIR
myhadoop-configure.sh
start-all.sh
hadoop dfs -mkdir input
hadoop dfs -copyFromLocal $PBS_O_WORKDIR/SINGLE.TXT input/
hadoop jar $PBS_O_WORKDIR/AnagramJob.jar input/SINGLE.TXT output
hadoop dfs -copyToLocal output/part* $PBS_O_WORKDIR
stop-all.sh
myhadoop-cleanup.sh
```

Anagram Example – Sample Output

```
cat part-00000
...
aabcdelmnu          manducable,ambulanced,
aabcdeorrsst        broadcasters,rebroadcasts,
aabcdeorrst         rebroadcast,broadcaster,
aabcdkrsw           drawbacks,backwards,
aabcdkrw            drawback,backward,
aabceeehlnsst       teachableness,cheatableness,
aabceeelnnrsstu     uncreatableness,untraceableness,
aabceeelrrt         recreatable,retraceable,
aabceehlt           cheatable,teachable,
aabceellr           lacerable,clearable,
aabceelnrtu         uncreatable,untraceable,
aabceelorrrstv      vertebrosacral,sacrovertebral,
...
...
```

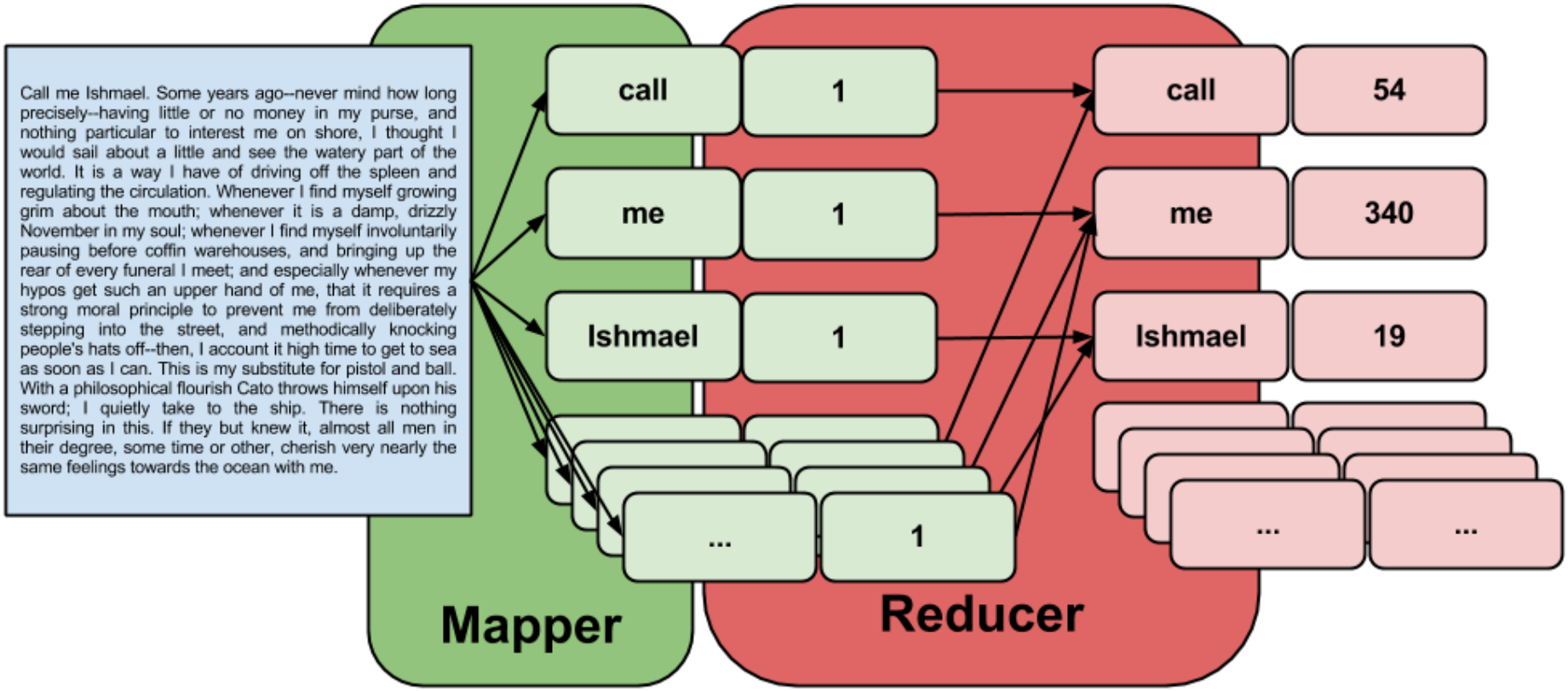
Hadoop Streaming

Programming Hadoop without Java

Hadoop and Python

- **Hadoop streaming w/ Python mappers/reducers**
  - portable
  - most difficult (or least difficult) to use
  - you are the glue between Python and Hadoop
- **mrjob (or others: hadoopy, dumbo, etc)**
  - comprehensive integration
  - Python interface to Hadoop streaming
  - Analogous interface libraries exist in R, Perl
  - Can interface directly with Amazon

Wordcount Example



Hadoop Streaming with Python

- **"Simplest" (most portable) method**
- **Uses raw Python, Hadoop – you are the glue**

```
cat input.txt | mapper.py | sort | reducer.py > output.txt
```

provide these two scripts; Hadoop does the rest

- **generalizable to any language you want** (Perl, R, etc)

Spin Up a Cluster

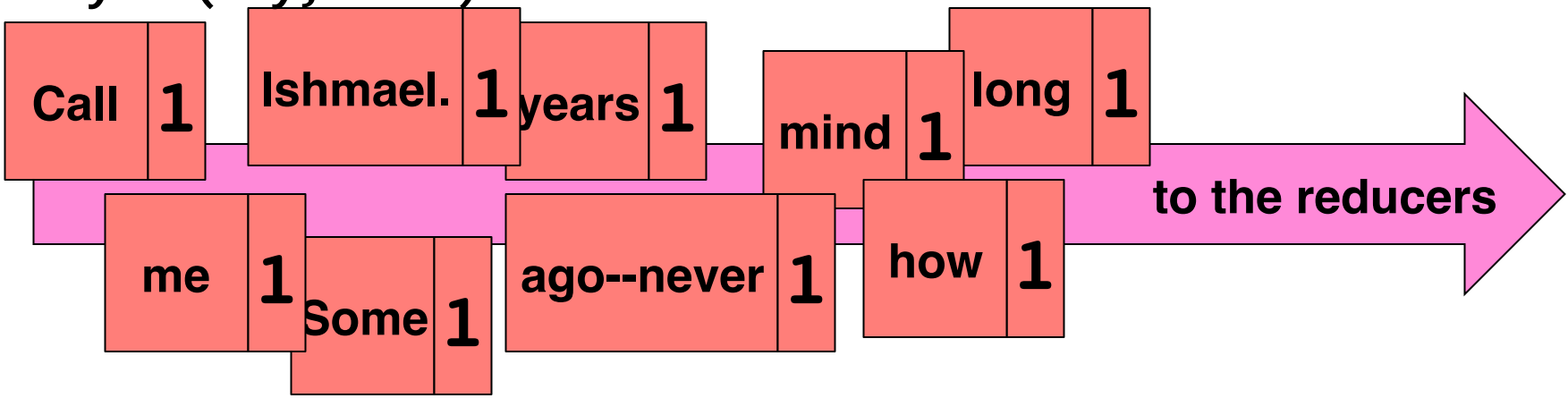
```
$ idev
          qsub: waiting for job 1228020.gordon-fe2.local to start
          qsub: job 1228020.gordon-fe2.local ready

$ myhadoop-configure.sh
   myHadoop: Keeping HADOOP_HOME=/home/diag/opt/hadoop/hadoop-1.0.4 from user environment
          myHadoop: Setting MH_IPOIB_TRANSFORM='s/$/.ibnet0/' from myhadoop.conf
                                    ...

$ start-all.sh
   starting namenode, logging to /scratch/train18/1228020.gordon-fe2.local/logs/hadoop-train18-
                              namenode-gcn-13-74.sdsc.edu.out
  gcn-13-74.ibnet0: starting datanode, logging to /scratch/train18/1228020.gordon-fe2.local/logs/
                          hadoop-train18-datanode-gcn-13-74.sdsc.edu.out

$ cd PYTHON/streaming
$ ls
    README   mapper.py   pg2701.txt   pg996.txt   reducer.py
```

What One Mapper Does

line = Call me Ishmael. Some years ago—never mind how long

keys = Call | me | Ishmael. | Some | years | ago--never | mind | how | long

emit.keyval(key,value) ...



Wordcount: Hadoop streaming mapper

```
#!/usr/bin/env python

import sys

for line in sys.stdin:
    line = line.strip()
    keys = line.split()
     for key in keys:
          value = 1
print( '%s\t%d' % (key, value) )
```

Reducer Loop Logic

For each key/value pair...

- **If this key is the same as the previous key,**
  - add this key's value to our running total.
- **Otherwise,**
  - print out the previous key's name and the running total,
  - reset our running total to 0,
  - add this key's value to the running total, and
  - "this key" is now considered the "previous key"

Wordcount: Streaming Reducer (1/2)

```
#!/usr/bin/env python

import sys

last_key = None
running_total = 0

for input_line in sys.stdin:
 input_line = input_line.strip()
this_key, value = input_line.split("\t", 1)
 value = int(value)

(to be continued...)
```

Wordcount: Streaming Reducer (2/2)

```
            if last_key == this_key:
                running_total += value
                        else:
                       if last_key:
        print( "%s\t%d" % (last_key, running_total) )
                      running_total = value
                       last_key = this_key

            if last_key == this_key:
    print( "%s\t%d" % (last_key, running_total) )
```

Testing Mappers/Reducers

-

```
$ head -n100 pg2701.txt | ./mapper.py |
          sort | ./reducer.py
                  ...
              with  5
              word, 1
           world.      1
       www.gutenberg.org 1
              you   3
              The   1
```

Launching Hadoop Streaming

```
hadoop jar
  $HADOOP_HOME/contrib/streaming/hadoop-streaming-1.2.1.jar
  -D mapred.reduce.tasks=2
  -mapper "$PWD/mapper.py"
  -reducer "$PWD/reducer.py"
  -input mobydick.txt
  -output output
```

`.../hadoop-streaming-1.2.1.jar`	Hadoop Streaming jarfile
`-mapper $PWD/mapper.py`	Mapper executable
`-reducer $PWD/reducer.py`	Reducer executable
`-input mobydick.txt`	location (on hdfs) of job input
`-output output`	location (on hdfs) of output dir

Tools/Applications w/ Hadoop

- **Several open source projects utilizing Hadoop infrastructure. Examples:**
  - HIVE – A data warehouse infrastructure providing data summarization and querying.
  - Hbase – Scalable distributed database
  - PIG – High-level data-flow and execution framework
  - Elephantdb – Distributed database specializing in exporting key/value data from Hadoop.
- **Some of these projects have been tried on Gordon by users.**

Apache Mahout

- **Apache project to implement scalable machine learning algorithms.**
- **Algorithms implemented:**
  - Collaborative Filtering
  - User and Item based recommenders
  - K-Means, Fuzzy K-Means clustering
  - Mean Shift clustering
  - Dirichlet process clustering
  - Latent Dirichlet Allocation
  - Singular value decomposition
  - Parallel Frequent Pattern mining
  - Complementary Naive Bayes classifier
  - Random forest decision tree based classifier

Mahout Example – Recommendation Mining

- Collaborative Filtering algorithms aim to solve the prediction problem where the task is to estimate the preference of a user towards an item which he/she has not yet seen.

- Itembased Collaborative Filtering estimates a user's preference towards an item by looking at his/her preferences towards similar items.

- Mahout has two Map/Reduce jobs to support Itembased Collaborative Filtering

  - ItemSimiliarityJob – Computes similarity values based on input preference data
  - RecommenderJob - Uses input preference data to determine recommended itemIDs and scores. Can compute Top – N recommendations for user for example.

- Collaborative Filtering is very popular (for example used by Google, Amazon in their recommendations)

Mahout – Recommendation Mining

- Once the preference data is collected, a user-item-matrix is created.
- Task is to predict missing entries based on the known data. Estimate a user's preference for a particular item based on their preferences to similar items.
- Example:

	Item 1	Item 2	Item 3
User 1	4	3	6
User 2	5	2	?
User 3	3	2	1

- More details and links to informative presentations at:
  https://cwiki.apache.org/confluence/display/MAHOUT/Itembased+Collaborative+Filtering

Mahout – Hands on Example

- Recommendation Job example : Mahout.cmd
- qsub Mahout.cmd to submit the job.
- Script also illustrates use of a custom hadoop configuration file. Customized info in mapred-site.xml.stub file which is copied into the configuration used.

Mahout – Hands On Example

- The input file is a comma separated list of userIDs, itemIDs, and preference scores (test.data). As mentioned in earlier slide, all users need not score all items. In this case there are 5 users and 4 items. The users.txt file provides the list of users for whom we need recommendations.

- The output from the Mahout recommender job is a file with userIDs and recommendations (with scores).

- The show_reco.py file uses the output and combines with user, item info from another input file (items.txt) to produce details recommendation info. This segment runs outside of the hadoop framework.

Mahout – Sample Output

User ID : 2
Rated Items

Item 2, rating=2

Item 3, rating=5

Item 4, rating=3

Recommended Items

Item 1, score=3.142857

PIG

- **PIG - high level programming on top of Hadoop map/reduce**
- **Sql-like data flow operations**
- **Essentially, key,values abstracted to fields and more complex data strutures**

Hot off the press on Gordon!

- **Hadoop2 (v2.2.0) installed and available via a module.**
- **Apache Spark installed and available via a module. Configured to run with Hadoop2.**
- **myHadoop extended to handle Hadoop2/YARN, and Spark.**

Summary

- Hadoop framework extensively used for scalable distributed processing of large datasets. Several tools available that leverage the framework.
- HDFS provides scalable filesystem tailored for accelerating MapReduce performance.
- Hadoop Streaming can be used to write mapper/reduce functions in any language.
- Data warehouse (HIVE), scalable distributed database (HBASE), and execution framework (PIG) available.
- Mahout w/ Hadoop provides scalable machine learning tools.

PIG (Hands On)

- **Exercises:**
- **Log in to gordon**

```
>
>qsub -I -q normal -lnodes=2:ppn=1:native,walltime=2:00:00 -A ######
>cd Pig2
>
> ps axu lgrep "p4rod”

> source Setup_pigtwt.cmd
> source Setup_Data4pig.cmd

> ps axu lgrep "p4rod”
>pig

grunt>
```

PIG Hands On - Output

> ls
Cleanup_pigtwt.cmd Run_pigtwtsrc.qsub hadoophosts.txt pig_wds_grp_cnt.txt
Doscript_pigtwt.cmd SetupData4pig.cmd pig_1383786713814.log rawtw_dtmsg.csv
GetDataFrompig.cmd Setup_pigtwt.cmd pig_out_mon_day_cnt.txt tw_script1.pig

Setup_4twt

gcn-14-21.ibnet0: starting tasktracker, logging to /scratch/p4rodrig/997799.gordon-fe2.local/hadoop-p4rodrig/log/hadoop-p4rodrig-tasktracker-gcn-14-21.sdsc.edu.out

Warning: $HADOOP_HOME is deprecated

Set up Data

2013-11-06 17:11:54,217 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://gcn-14-17.ibnet0:54310

2013-11-06 17:11:54,727 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to map-reduce job tracker at: gcn-14-17.ibnet0:54311

PIG (Hands On)

```
#watch out for   _=_   keep spacing around equal sign

raw = LOAD 'pigdata2load.txt' USING PigStorage(',') AS
(mon_day,year,hour,msg,place,retwtd,uid,flers_cnt,fling_cnt);
tmp = LIMIT raw 5;
DUMP tmp;

describe raw

rawstr = LOAD 'pigdata2load.txt' USING PigStorage(',') AS
(mon_day:chararray,year:int,hour:chararray,msg:chararray,place:chararray,retwted:char
array,uid:int,flers_cnt:int,fling_cnt:int);

describe rawstr
```

PIG Hands On (output)

1 dump tmp

INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1

(Apr 03,2012,09:11:56,havent you signe the petition yet hrd alkhawaja in his 55 day of hungerstrike well send it to obama httpstco8ww2sfzx bahrain,Null,False,283111461,185,Null)

(Apr 08,2012,07:02:08,rt azmoderate uc chuckgrassley pres obama has clearly demonstrated his intelligence you on the other hand look sound like you shou ,Null,False,128728124,904,Null)

(Apr 12,2012,13:02:44,rt dcgretch

2 describe raw

raw: {mon_day: bytearray,year: bytearray,hour: bytearray,msg: bytearray,place: bytearray,retwtd: bytearray,uid: bytearray,flers_cnt: bytearray,fling_cnt: bytearray}

3 describe rawstr

rawstr: {mon_day: chararray,year: int,hour: chararray,msg: chararray,place: chararray,retwted: chararray,uid: int,flers_cnt: int,fling_cnt: int}

PIG (Hands On)

```
rawbymon_day = GROUP rawstr BY mon_day;

--  mon_day_cnt = FOREACH rawbymon_day GENERATE group,COUNT(*);
-- return error about cast

mon_day_cnt = FOREACH rawbymon_day GENERATE group,COUNT(rawstr);

tmpc      = LIMIT mon_day_cnt 5;
DUMP tmpc;
STORE mon_day_cnt INTO 'pig_out_mon_day_cnt.txt';
```

PIG Hands On - output

INFO org.apache.pig.tools.pigstats.SimplePigStats - Script Statistics:

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
1.0.4 0.12.0 p4rodrig 2013-11-06 17:19:50 2013-11-06 17:20:35 GROUP_BY

Success!

Job Stats (time in seconds):

JobId Maps Reduces MaxMapTime MinMapTIme AvgMapTime MedianMapTime MaxReduceTime MinReduceTime AvgReduceTime MedianReducetime Alias Feature Outputs

job_201311061709_0005 1 1 6 6 6 6 15 15 1515 mon_day_cnt,rawbymon_day,rawstr GROUP_BY,COMBINER hdfs://gcn-14-17.ibnet0:54310/user/p4rodrig/pig_out_mon_day_cnt.txt,

Input(s):

Successfully read 109 records (13996 bytes) from: "hdfs://gcn-14-17.ibnet0:54310/user/p4rodrig/pigdata2load.txt"

PIG (Hands On)

```
raw_msg = FOREACH rawstr {mwds=TOKENIZE(msg); GENERATE $0,$1,$2,mwds;};
tmpd= LIMIT raw_msg 5;
DUMP tmpd;

raw_msgf = FOREACH rawstr {mwds=TOKENIZE(msg); GENERATE $0,$1,$2,flatten(mwds);};
tmpd = LIMIT raw_msg 5;
DUMP tmpd;
```

PIG Hands On - output

First dump

(Apr 03,2012,09:11:56,{(havent),(you),(signe),(the),(petition),(yet),(hrd),(alkhawaja),(in),(his),(55),(day),(of),(hungerstrike),(well),(send),(it),(to),(obama),(httpstco8ww2sfzx),(bahrain)})

(Apr 08,2012,07:02:08,{(rt),(azmoderate),(uc),(chuckgrassley),(pres),(obama),(has),(clearly),(demonstrated),(his),(intelligence),(you),(on),(the),(other),(hand),(look),(sound),(like),(you),(shou)})

Second dump

(Apr 03,2012,09:11:56,{(havent),(you),(signe),(the),(petition),(yet),(hrd),(alkhawaja),(in),(his),(55),(day),(of),(hungerstrike),(well),(send),(it),(to),(obama),(httpstco8ww2sfzx),(bahrain)})

(Apr 08,2012,07:02:08,{(rt),(azmoderate),(uc),(chuckgrassley),(pres),(obama),(has),(clearly),(demonstrated),(his),(intelligence),(you),(on),(the),(other),(hand),(look),(sound),(like),(you),(shou)})