Introduction to Hadoop on the SDSC Gordon Data Intensive Cluster

Mahidhar Tatineni

SDSC Summer Institute

August 06, 2013

Overview

- Hadoop framework extensively used for scalable distributed processing of large datasets.
- Typical MapReduce jobs make use of locality of data, processing data on or near the storage assets to minimize data movement. Hadoop Distributed Filesystem (HDFS) also has built in redundancy and fault tolerance.
- Map Step: The master process divides problem into smaller sub-problems, and distributes them to slave nodes as "map tasks".
- Reduce Step: Processed data from slave node, i.e. output from the map tasks, serves as input to the reduce tasks to form the ultimate output.

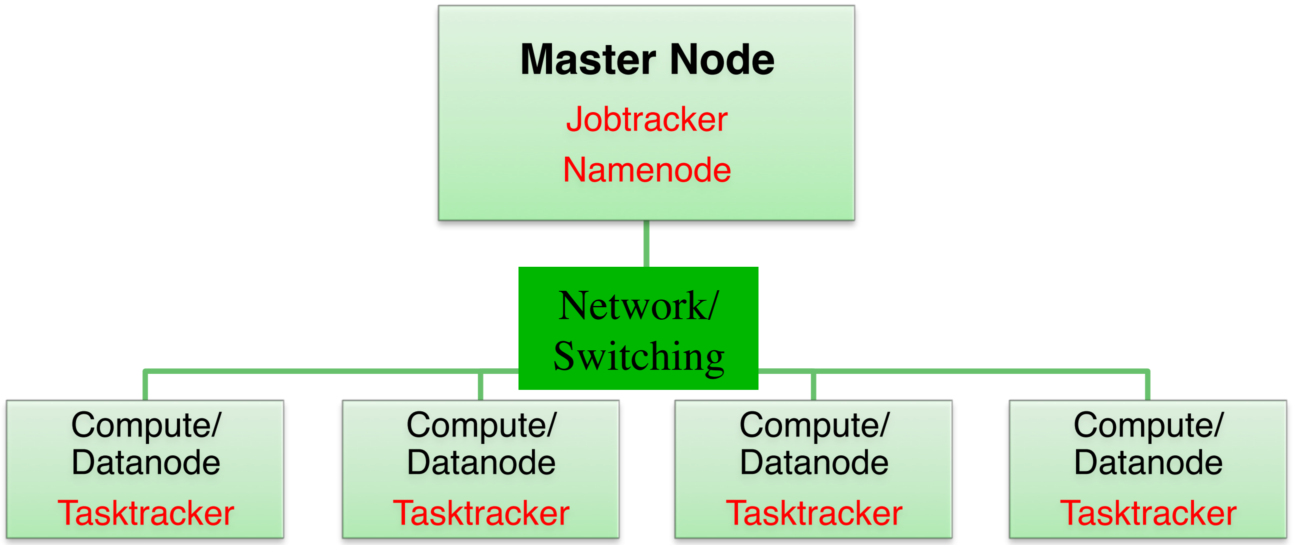
Hadoop: Application Areas

- **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

- **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.

Simple Example – 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 – 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	Task 2
<Bye, 1>	< Goodbye, 1>
<Hello, 1>	< Hadoop, 2>
<World, 2>	<Hello, 1>

- **Step 3 : Global reduce:**

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

Benefits of Gordon Architecture for Hadoop

- **Gordon architecture presents potential performance benefits with high performance storage (SSDs) and high speed network (QDR IB).**

- **Storage options viable for Hadoop on Gordon**
  - SSD via **iSCSI Extensions for RDMA** (iSER)
  - Lustre filesystem (Data Oasis), persistent storage

- **Approach to running Hadoop within the scheduler infrastructure – myHadoop (developed at SDSC).**

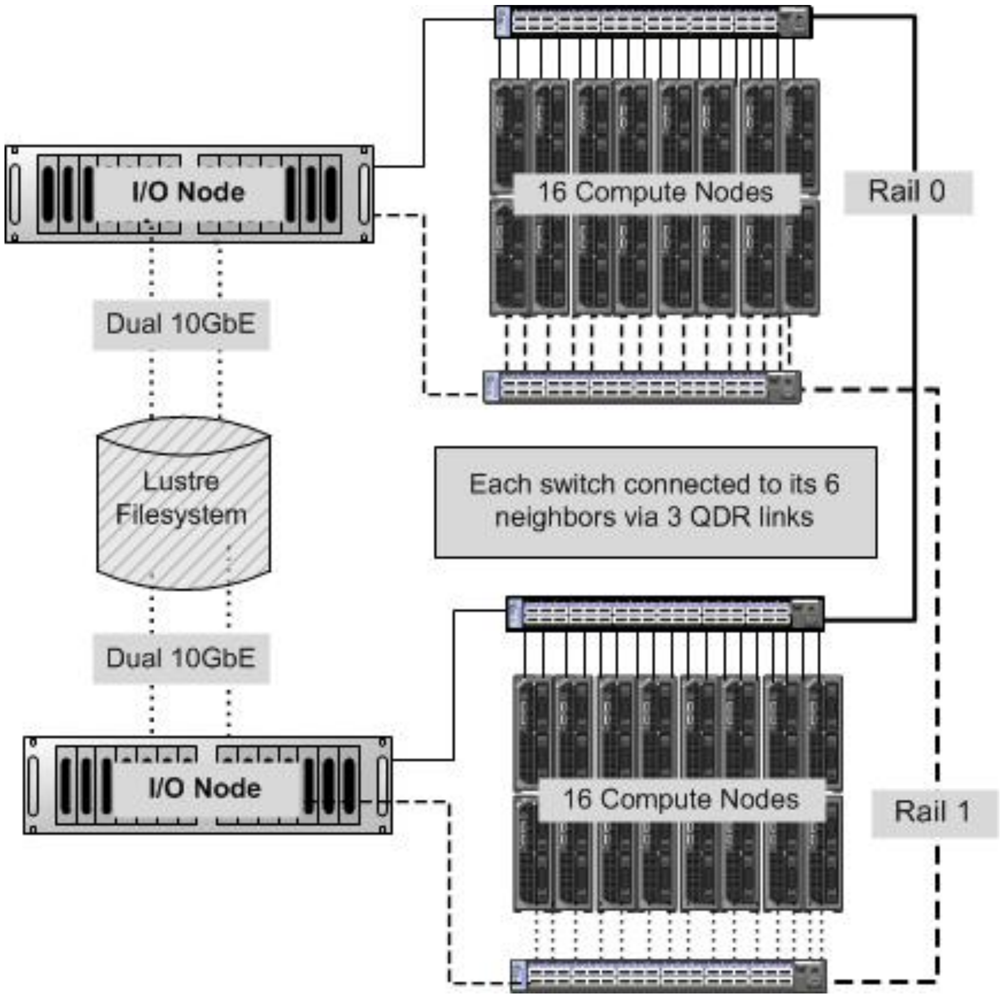
Overview (Contd.)

- **Network options**
  - 1 GigE – low performance
  - IPoIB – Can make use of high speed infiniband network.
- **Sample Results in current talk**
  - DFS – Benchmark runs done on SSD based storage.
  - TeraSort – Scaling study with sizes 100GB-1.6TB, 4-128 nodes.
- **Future/Ongoing Work for improving performance**
  - Hadoop-RDMA – Network-Based Computation Lab, OSU
  - UDA - Unstructured Data Accelerator from Mellanox

    http://www.mellanox.com/related-docs/applications/SB_Hadoop.pdf
  - Lustre for Hadoop – Can provide another persistent data option.

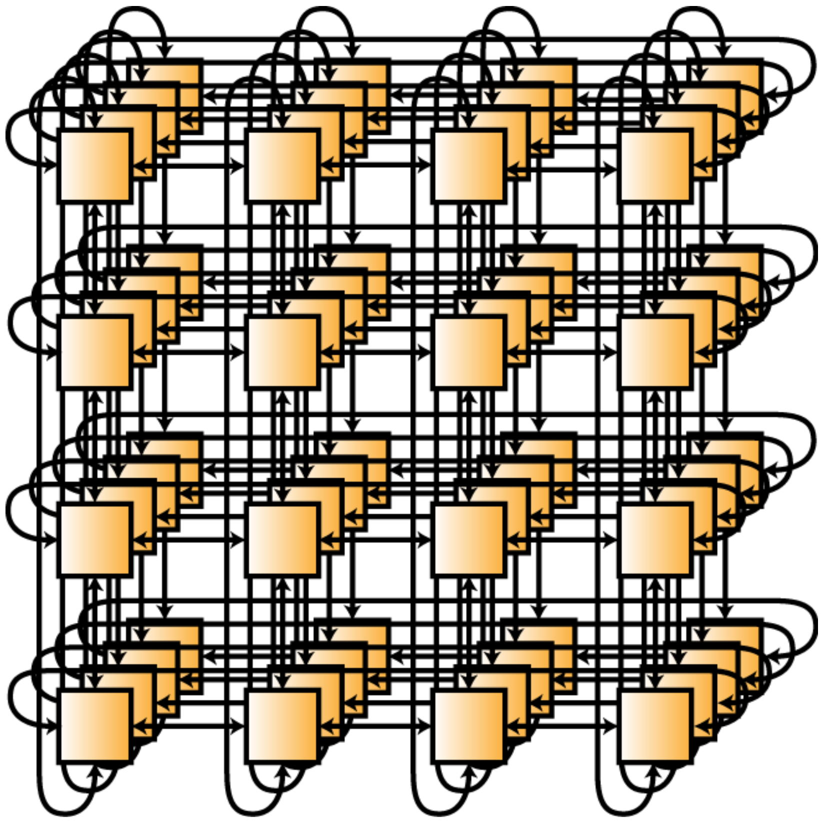
Gordon Architecture

Subrack Level Architecture



16 compute nodes per subrack

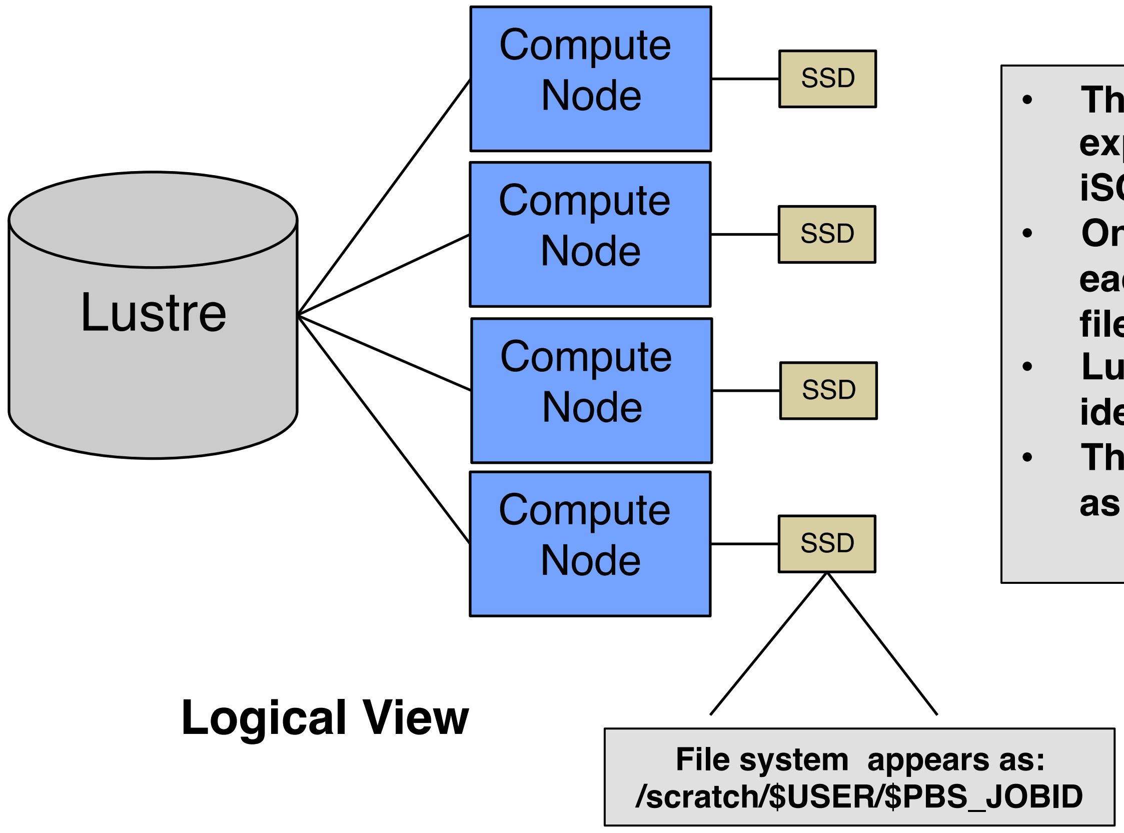
3D Torus of Switches



- Linearly expandable
- Simple wiring pattern
- Short Cables- Fiber Optic cables generally not required
- Lower Cost :40% as many switches, 25% to 50% fewer cables
- Works well for localized communication
- Fault Tolerant within the mesh with 2QoS Alternate Routing
- Fault Tolerant with Dual-Rails for all routing algorithms

3rd dimension wrap-around not shown for clarity

Primary Storage Option for Hadoop: One SSD per Compute Node (only 4 of 16 compute nodes shown)



- The SSD drives on each I/O node are exported to the compute nodes via the iSCSI Extensions for RDMA (iSER).
- One 300 GB flash drive exported to each compute node appears as a local file system
- Lustre parallel file system is mounted identically on all nodes.
- This exported SSD storage can serve as the local storage used by HDFS.

Logical View

File system appears as:
/scratch/$USER/$PBS_JOBID

Hadoop Configuration Details

myHadoop – Integration with Gordon Scheduler

- **myHadoop* was developed at SDSC to help run Hadoop within the scope of the normal scheduler.**
- **Scripts available to use the node list from the scheduler and dynamically change Hadoop configuration files.**
  - mapred-site.xml
  - masters
  - slaves
  - core-site.xml
- **Users can make additional hadoop cluster changes if needed**
- **Nodes are exclusive to the job [does not affect the rest of the cluster].**

***myHadoop link: http://sourceforge.net/projects/myhadoop/**

Hands On Examples

- **Examples are in the hadoop directory within the SI2013 directory:**

  cd $HOME/SI2013/hadoop

- **Start with simple benchmark examples:**

  - TestDFS_2.cmd – TestDFS example to benchmark HDFS performance.

  - TeraSort_2.cmd – Sorting performance benchmark.

TestDFS Example

PBS variables part:

```
#!/bin/bash
#PBS -q normal
#PBS -N hadoop_job
#PBS -l nodes=2:ppn=1
#PBS -o hadoop_dfstest_2.out
#PBS -e hadoop_dfstest_2.err
#PBS -V
```

TestDFS example

Set up Hadoop environment variables:

```
# Set this to location of myHadoop on gordon
export MY_HADOOP_HOME="/opt/hadoop/contrib/myHadoop"

# Set this to the location of Hadoop on gordon
export HADOOP_HOME="/opt/hadoop"

#### Set this to the directory where Hadoop configs should be
generated
# Don't change the name of this variable (HADOOP_CONF_DIR) as it is
# required by Hadoop - all config files will be picked up from here
#
# Make sure that this is accessible to all nodes
export HADOOP_CONF_DIR="/home/$USER/config"
```

TestDFS Example

```
#### Set up the configuration
# Make sure number of nodes is the same as what you have requested from PBS
# usage: $MY_HADOOP_HOME/bin/configure.sh -h
echo "Set up the configurations for myHadoop"

### Create a hadoop hosts file, change to ibnet0 interfaces - DO NOT REMOVE -
sed 's/$/.ibnet0/' $PBS_NODEFILE > $PBS_O_WORKDIR/hadoophosts.txt
export PBS_NODEFILEZ=$PBS_O_WORKDIR/hadoophosts.txt

### Copy over configuration files
$MY_HADOOP_HOME/bin/configure.sh -n 2 -c $HADOOP_CONF_DIR

### Point hadoop temporary files to local scratch - DO NOT REMOVE -
sed -i 's@HADDTEMP@'$PBS_JOBID'@g' $HADOOP_CONF_DIR/hadoop-env.sh
```

TestDFS Example

```
#### Format HDFS, if this is the first time or not a persistent instance
echo "Format HDFS"
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR
namenode -format
echo
sleep 1m
#### Start the Hadoop cluster
echo "Start all Hadoop daemons"
$HADOOP_HOME/bin/start-all.sh
#$HADOOP_HOME/bin/hadoop dfsadmin -safemode leave
echo
```

TestDFS Example

```
#### Run your jobs here
echo "Run some test Hadoop jobs"
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-
test-1.0.3.jar TestDFSIO -write
-nrFiles 8 -fileSize 1024 -bufferSize 1048576
sleep 30s
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-
test-1.0.3.jar TestDFSIO -read -
nrFiles 8 -fileSize 1024 -bufferSize 1048576

echo

#### Stop the Hadoop cluster
echo "Stop all Hadoop daemons"
$HADOOP_HOME/bin/stop-all.sh
echo
```

Running the TestDFS example

- **Submit the job:**

```
qsub TestDFS_2.cmd
```

- **Check the job is running (qstat)**

- **Once the job is running the hadoophosts.txt file is created. For example on a sample run:**

```
$ more hadoophosts.txt
gcn-13-11.ibnet0
gcn-13-12.ibnet0
```

Configuration Files During the Run

- **Lets check the configuration details (files in ~/config) setup dynamically:**

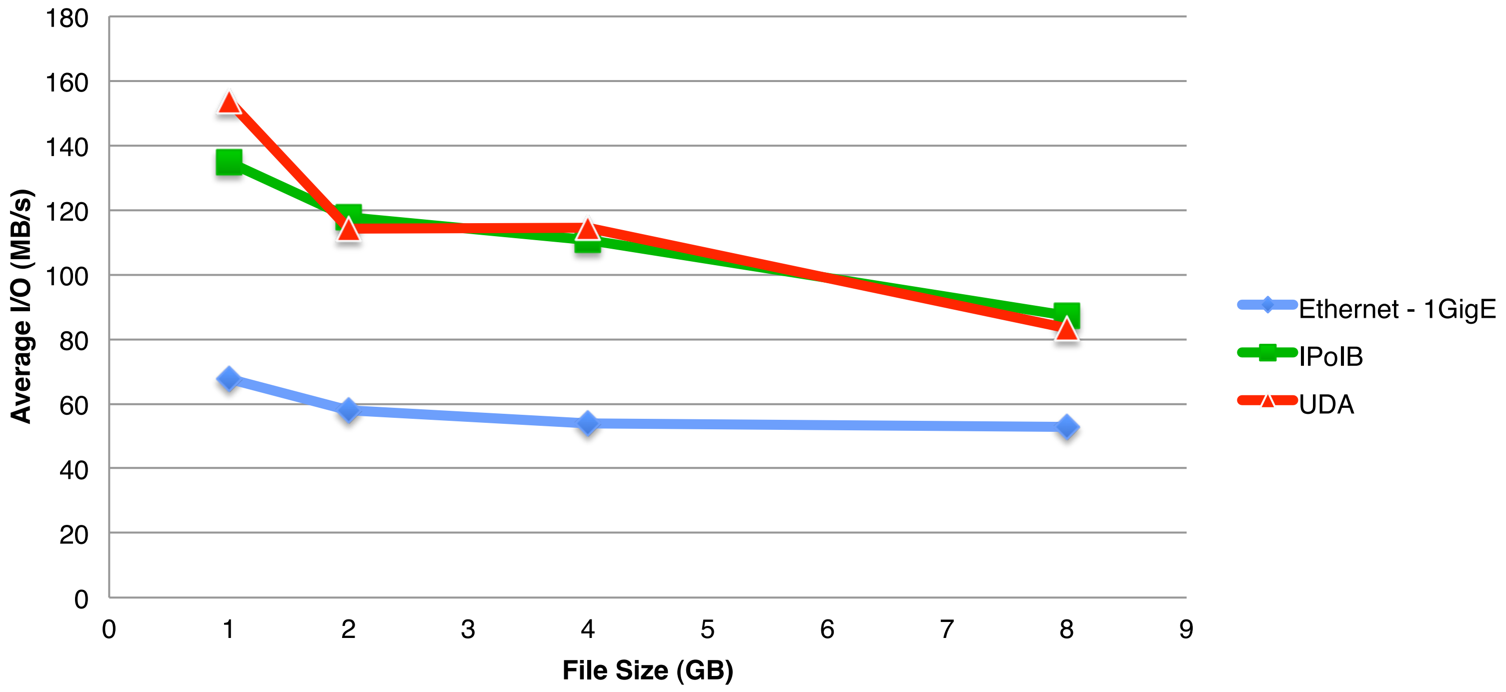
```
$ more masters
gcn-13-11.ibnet0
$ more slaves
gcn-13-11.ibnet0
gcn-13-12.ibnet0
$ grep scratch hadoop-env.sh
export HADOOP_PID_DIR=/scratch/$USER/538309.gordon-fe2.local
export TMPDIR=/scratch/$USER/538309.gordon-fe2.local
export HADOOP_LOG_DIR=/scratch/mahidhar/538309.gordon-fe2.local/hadoop-mahidhar/log
$ grep hdfs core-site.xml
  <value>hdfs://gcn-13-11.ibnet0:54310</value>
```

Test Output

- **Check the PBS output and error files for the results.**
- **Sample snippets:**

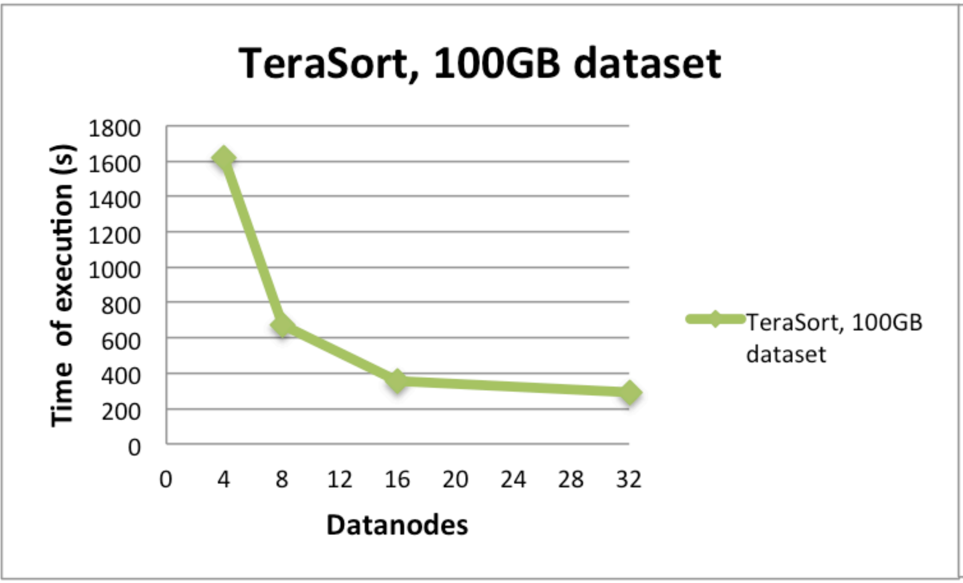
```
13/01/30 18:56:00 INFO fs.TestDFSIO:        Number of files: 8
13/01/30 18:56:00 INFO fs.TestDFSIO: Total MBytes processed: 8192
13/01/30 18:56:00 INFO fs.TestDFSIO:      Throughput mb/sec: 41.651837012782316
13/01/30 18:56:00 INFO fs.TestDFSIO: Average IO rate mb/sec: 93.37788391113281
13/01/30 18:56:00 INFO fs.TestDFSIO:  IO rate std deviation: 58.587153756958564
13/01/30 18:56:00 INFO fs.TestDFSIO:     Test exec time sec: 99.554
13/01/30 18:56:00 INFO fs.TestDFSIO:
…
….
13/01/30 18:57:16 INFO fs.TestDFSIO:        Number of files: 8
13/01/30 18:57:16 INFO fs.TestDFSIO: Total MBytes processed: 8192
13/01/30 18:57:16 INFO fs.TestDFSIO:      Throughput mb/sec: 207.49746707193515
13/01/30 18:57:16 INFO fs.TestDFSIO: Average IO rate mb/sec: 207.5077362060547
13/01/30 18:57:16 INFO fs.TestDFSIO:  IO rate std deviation: 1.4632078247537383
13/01/30 18:57:16 INFO fs.TestDFSIO:     Test exec time sec: 41.365
```

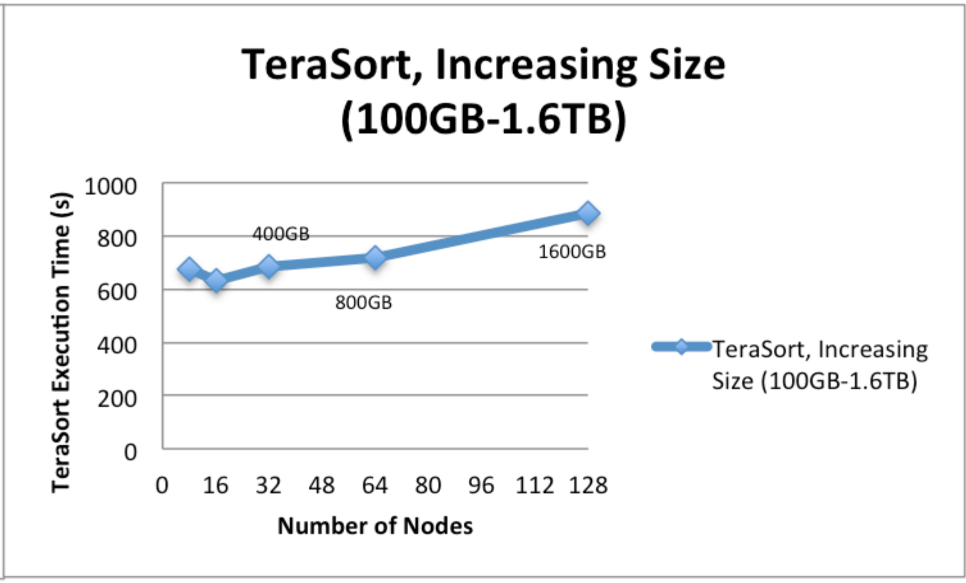
Hadoop on Gordon – TestDFSIO Results



Results from the TestDFSIO benchmark. Runs were conducted using 4 datanodes and 2 map tasks. Average write I/O rates are presented.

Hadoop on Gordon – TeraSort Results





(a) (b)

Results from the TeraSort benchmark. Data generated using teragen and is then sorted. Two sets or runs were performed using IPoIB, SSD storage – (a) Fixed dataset size of 100GB, number of nodes varied from 4 to 32, (b) Increasing dataset size with number of nodes.

Hadoop Parameters

- Several parameters can be tuned at the system level, in general configuration (namenode and jobtracker server threads), HDFS configuration (blocksize) , MapReduce configuration (number of tasks, input streams, buffer sizes etc).

- Good summary in:

http://software.intel.com/sites/default/files/m/f/4/3/2/f/31124-Optimizing_Hadoop_2010_final.pdf

Using Hadoop Framework

- **Hadoop framework developed in Java, with Java API available to developers.**
- **Map/Reduce applications can use Java or have several other options:**
  - C++ API using Hadoop Pipes
  - Python – Using Hadoop Streaming, Dumbo, Pydoop
- **Hadoop Streaming**
  - Allows users to create and run map/reduce jobs with custom executables and scripts as map and reduce tasks.

Hadoop Streaming

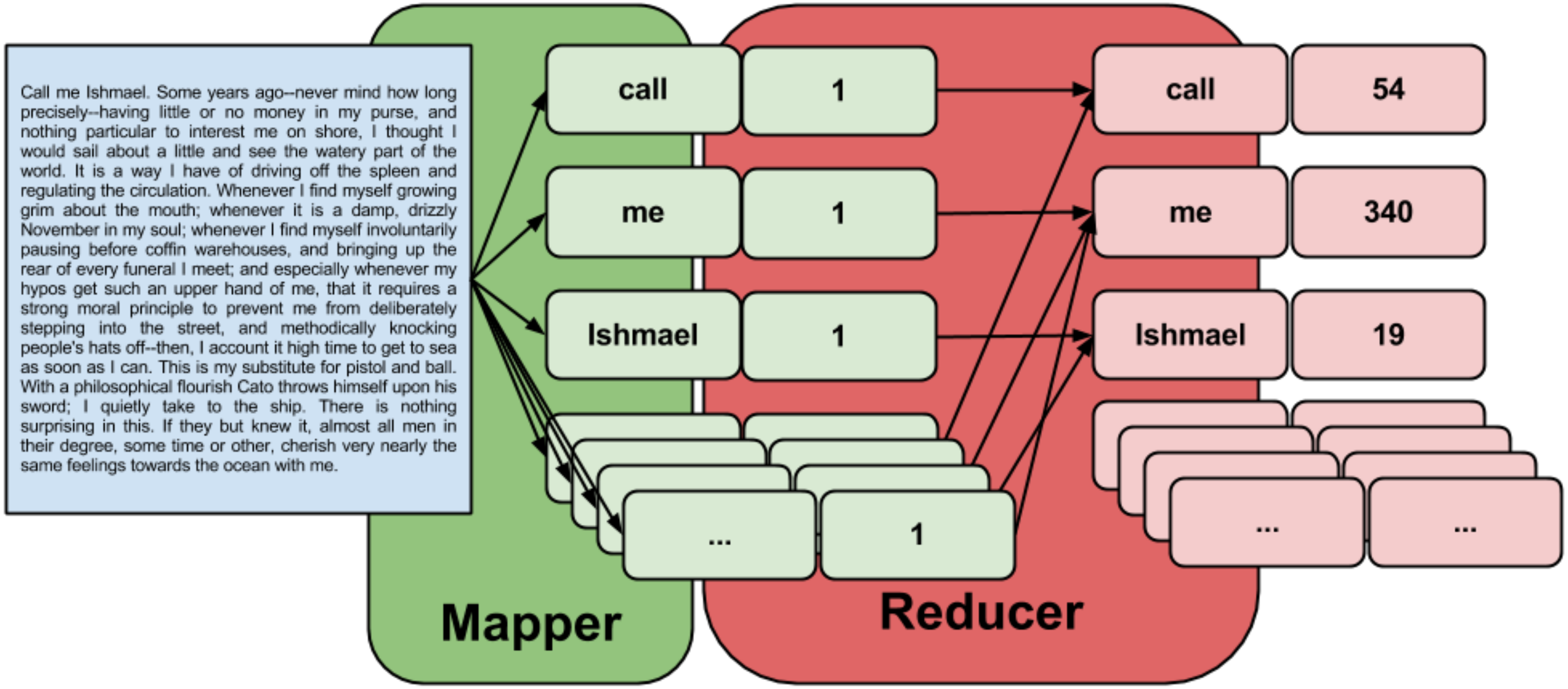
- Write mapper and reducer functions in any language
- Hadoop passes records and key/values via stdin/ stdout pipes:

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

provide these two scripts; Hadoop does the rest

Don't have to know Java--can adapt existing scripts, libraries

Hadoop Streaming – Word Count


Call me Ishmael. Some years ago--never mind how long precisely--having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off--then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the ship. There is nothing surprising in this. If they but knew it, almost all men in their degree, some time or other, cherish very nearly the same feelings towards the ocean with me.

Word Count – mapper.py

```
#!/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) )
```

Word Count - Reducer

- All key/value pairs are sorted before being presented to the reducers
- All key/value pairs sharing the same key are sent to the same reducer

call	1
call	1
me	1
me	1
me	1
Ishmael	1

```
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"
```

Word Count – reducer.py (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)
...
```

Word Count – reducer.py (2/2)

```
for input_line in sys.stdin:
    ...
    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) )
```

/home/glock/tutorials/hpchadoop/wordcount.py/reducer.py

Word Count – Input Data

```
$ wget http://www.gutenberg.org/cache/epub/2701/
   pg2701.txt
```

(or just `cp /home/glock/tutorials/hpchadoop/wordcount.py/pg2701.txt`)

Test your mappers/reducers!

```
$ head -n1000 | ./mapper.py | sort | ./reducer.py
...
young    4
your     16
yourself     3
zephyr   1
```

Word Count – Job Launch

```
# Move input data into HDFS
$ hadoop dfs -mkdir wordcount
$ hadoop dfs -copyFromLocal ./pg2701.txt wordcount/
   mobydick.txt

# Job launch!
$ hadoop \
   jar /opt/hadoop/contrib/streaming/hadoop-
      streaming-1.0.3.jar \
   -mapper "python $PWD/mapper.py" \
   -reducer "python $PWD/reducer.py" \
   -input "wordcount/mobydick.txt"   \
   -output "wordcount/output"
```

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 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
------------------------
```

Summary

- Hadoop framework extensively used for scalable distributed processing of large datasets. Several tools available that leverage the framework.
- Hadoop can be dynamically configured and run on Gordon using myHadoop framework.
- High speed Infiniband network allows for significant gains in performance using SSDs.
- TeraSort benchmark shows good performance and scaling on Gordon.
- Hadoop Streaming can be used to write mapper/reduce functions in any language.
- Mahout w/ Hadoop provides scalable machine learning tools.