Hadoop Deployment and Performance on Gordon Data Intensive Supercomputer

Mahidhar Tatineni, Rick Wagner, Eva Hocks, Christopher Irving, and Jerry Greenberg

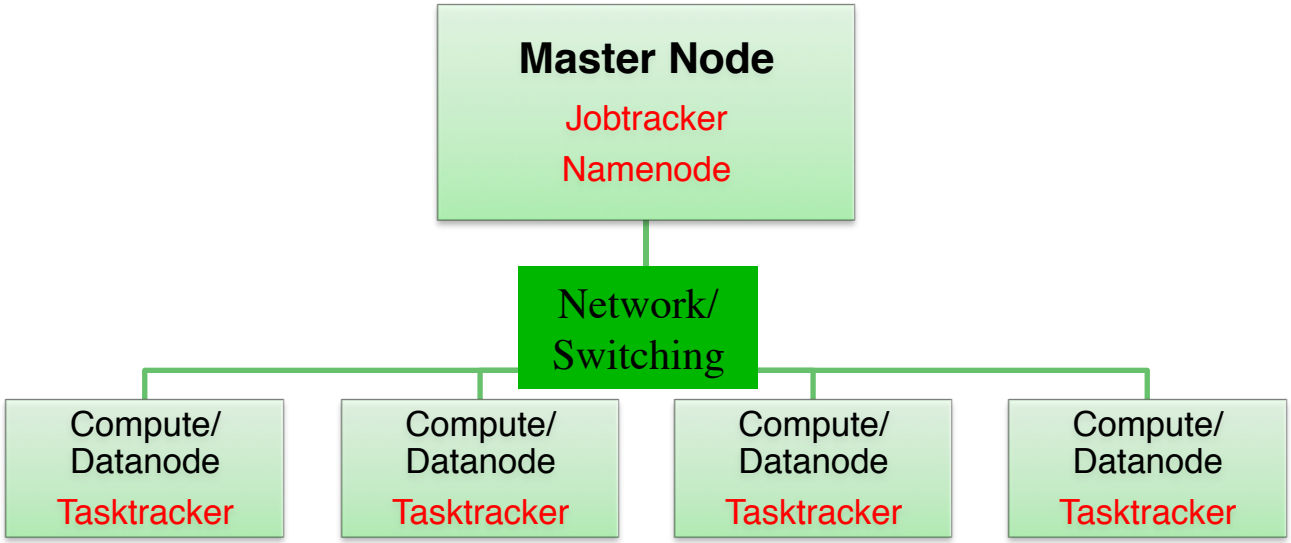
SDSC

XSEDE13, July 22-25, 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 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.

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: 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 – Application Areas on Gordon

- Research projects in humanities, data-mining algorithms, scalability of schedulers, and high-throughput dental computing.

- Predictive Analytics Center of Excellence (PACE) developing a research platform driven by an existing campus microgrid for developing large scale, predictive analytics for real-time energy management and parts of this platform use the Hadoop infrastructure on Gordon.

- UCSD Class (Some projects continued as reseach):
  - Analysis of temperature and airflow sensors on UCSD campus
  - Detection of failures in the Internet
  - Prediction of medical costs in car accidents
  - Registration of brain images
  - Deciphering RNA regulatory code
  - Analysis of election day Tweets

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.)

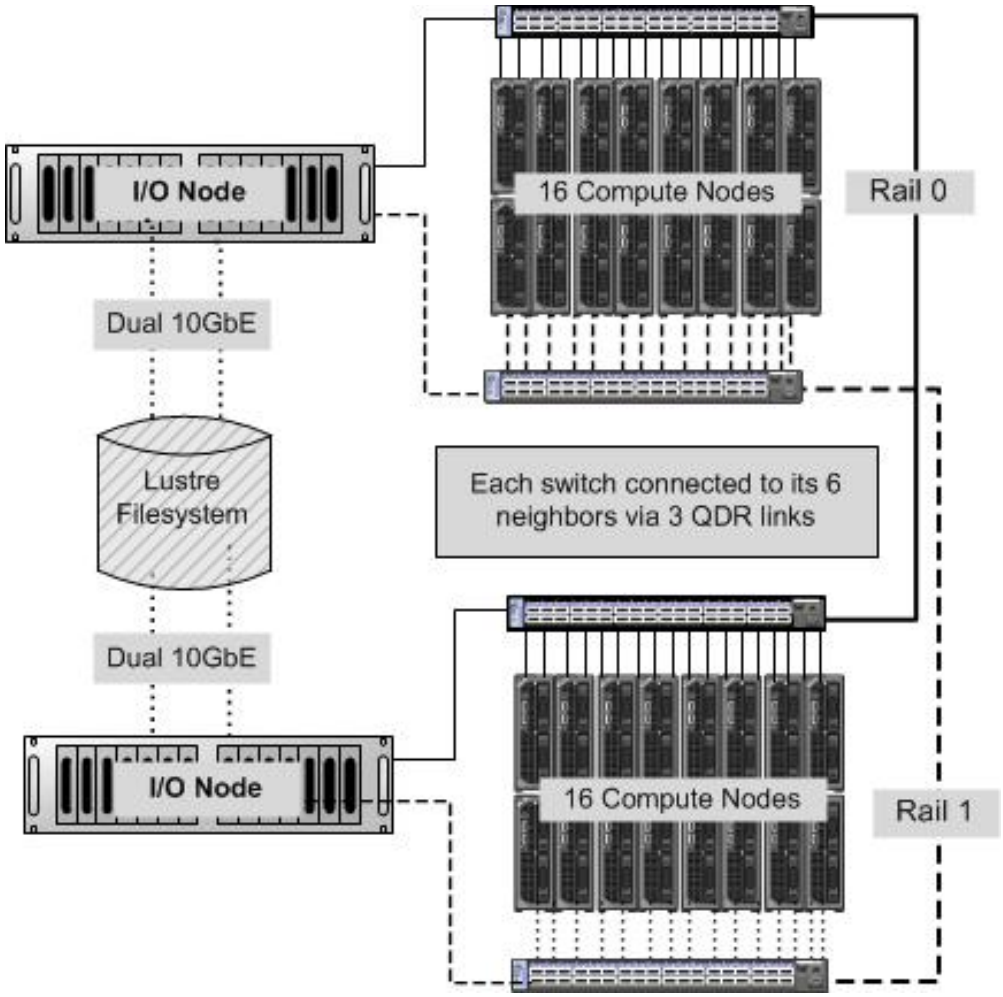
- **Network options**
  - 1 GigE – low performance
  - IPoIB – Can make use of high speed infiniband network.
- **Sample Results in current talk**
  - DFS – Benchmark runs done on SSD based storage.
  - TeraSort – Scaling study with sizes 100GB-1.6TB, 4-128 nodes.
- **Future/Ongoing Work for improving performance**
  - HADOOP-RDMA – Network-Based Computing Lab, Ohio State University
  - 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

Gordon System Specification

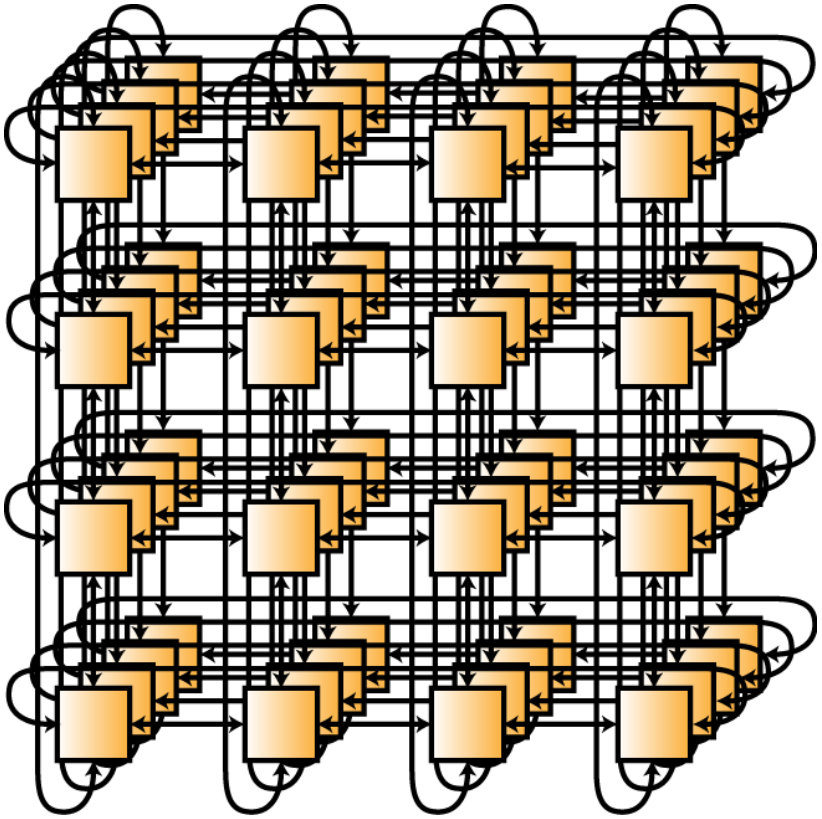
Intel Sandy Bridge Compute node	
Sockets	2
Cores	16
Clock speed	2.6
DIMM slots per socket	4
DRAM capacity	64 GB
Intel Flash I/O Node	
NAND flash SSD drives	16
SSD capacity per drive/Capacity per node/total	300 GB / 4.8 TB / 300 TB
Flash bandwidth per drive (read/write) Flash bandwidth per node (write/read)	270 MB/s / 210 MB/s 4.3 /3.3 GB/s
SMP Super-Node	
Compute nodes	32
I/O nodes	2
Addressable DRAM	2 TB
Addressable memory including flash	12TB
Gordon	
Compute Nodes	1,024
Total compute cores	16,384
Peak performance	341TF
Aggregate memory	64 TB
InfiniBand Interconnect	
Aggregate torus BW	9.2 TB/s
Type	Dual-Rail QDR InfiniBand
Link Bandwidth	8 GB/s (bidirectional)
Latency (min-max)	1.25 μs – 2.5 μs
Disk I/O Subsystem	
Total storage	4.5 PB (raw)
I/O bandwidth	100 GB/s
File system	Lustre

Subrack Level Architecture



16 compute nodes per subrack

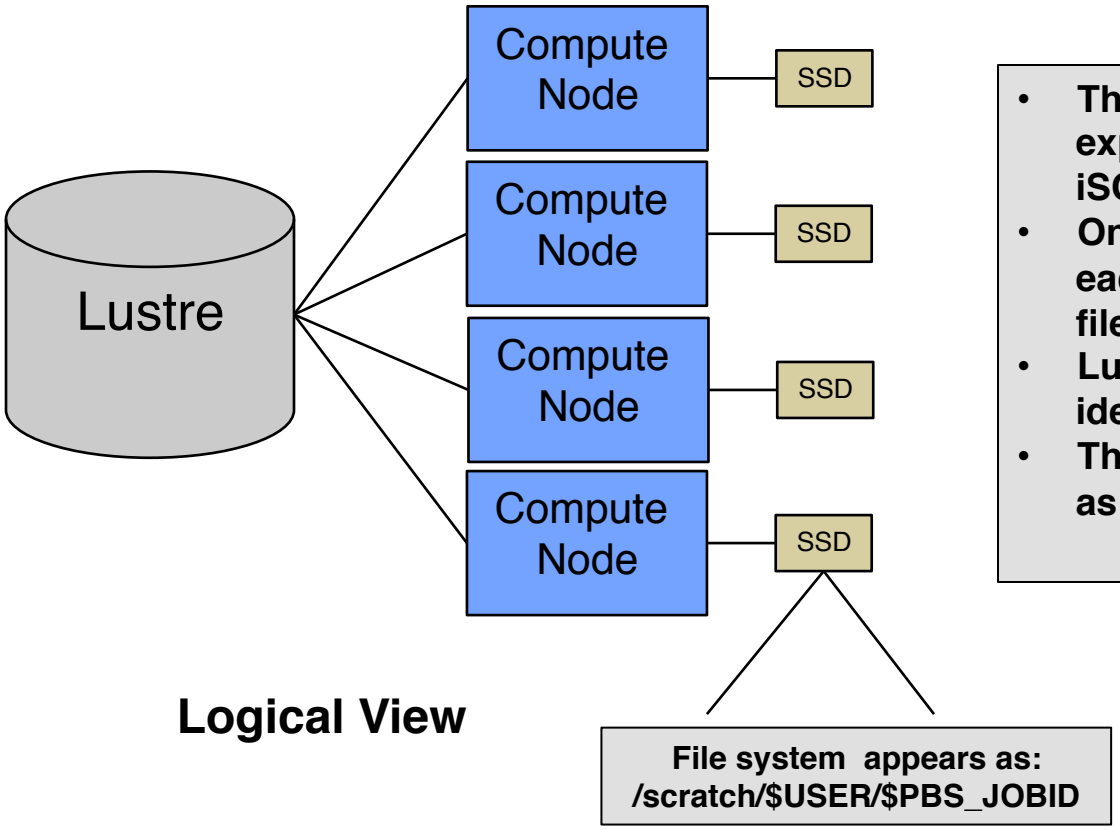
3D Torus of Switches



- 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.**

Hadoop Configuration

Dynamic Provisioning Using myHadoop

myHadoop – Integration with Gordon Scheduler

- **myHadoop* was developed at SDSC (primary developer: Sriram Krishnan) 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/**

Hadoop on Gordon – Storage Options

- **Primary storage option is to use SSDs which are available via iSCSI Extensions for RDMA (iSER) on each compute node.**

- **Prolog creates job specific SSD filesystem location which can be used for HDFS storage. myHadoop framework leveraged to do this dynamically.**

- **Lustre storage is available for persistent option. DFS benchmark results show very good performance comparable to SSD storage.**

Hadoop on Gordon – Network Options

- **All Gordon compute nodes are dual QDR Infiniband connected. Additionally, a 1GigE connection is available.**

- **For a standard Apache Hadoop install IPoIB using one of the default QDR links is the best network option. myHadoop setup uses the ib0 addresses in dynamic configuration set up.**

- **Mellanox's Unstructured Data Accelerator (UDA) is another option providing the middleware interface directly to high speed IB link. We are currently testing this on Gordon.**

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
```

Configuration Files During a Typical Run

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

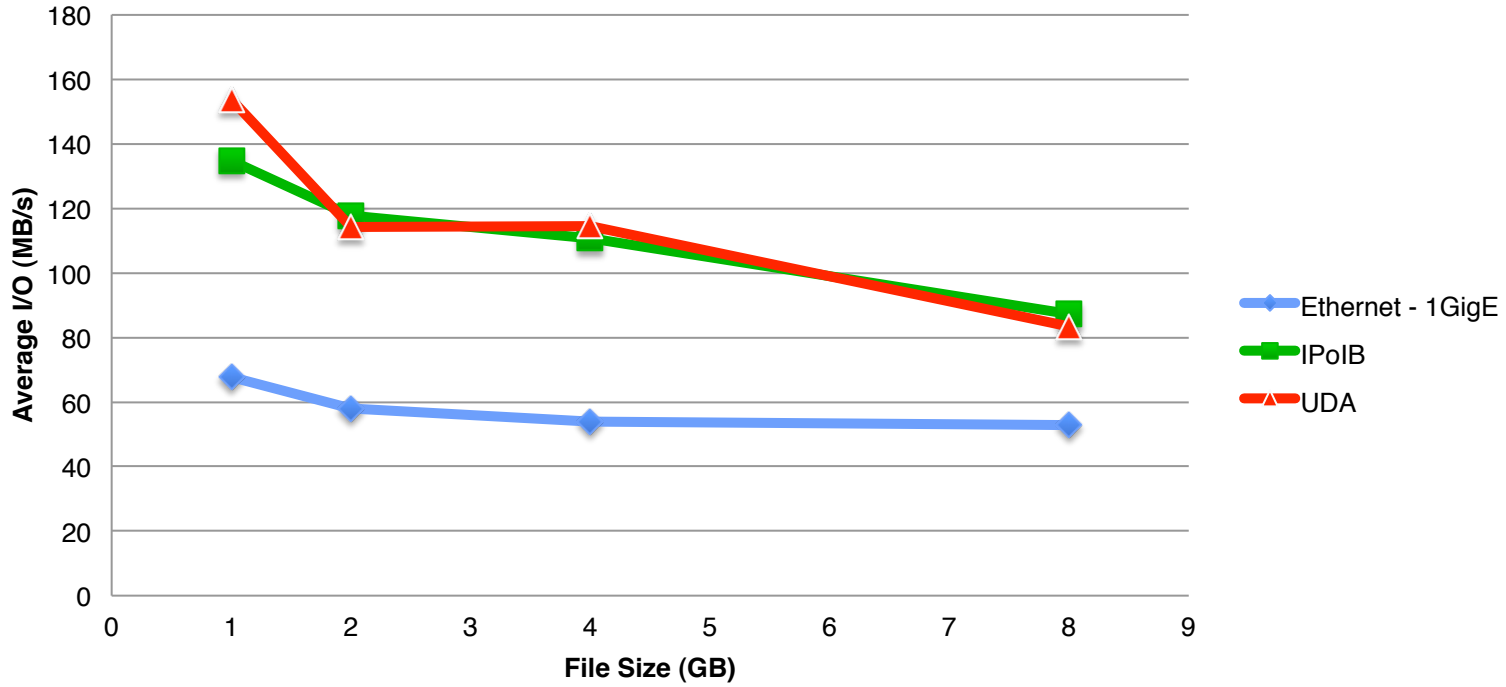
```
$ more masters
gcn-13-11.ibnet0
$ more slaves
gcn-13-11.ibnet0
gcn-13-12.ibnet0
$ grep scratch hadoop-env.sh
export HADOOP_PID_DIR=/scratch/$USER/538309.gordon-fe2.local
export TMPDIR=/scratch/$USER/538309.gordon-fe2.local
export HADOOP_LOG_DIR=/scratch/mahidhar/538309.gordon-fe2.local/hadoop-mahidhar/log
$ 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:**

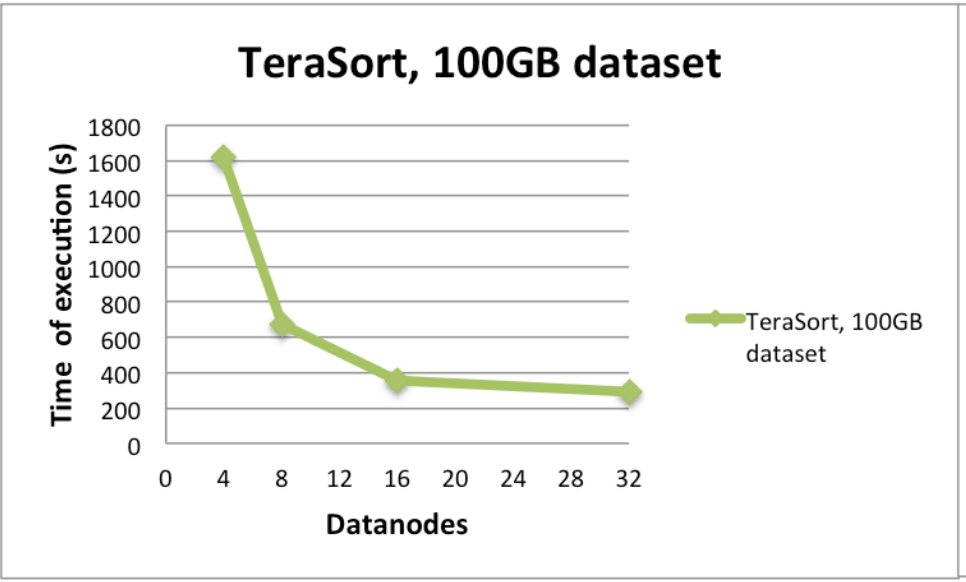
```
13/01/30 18:56:00 INFO fs.TestDFSIO:       Number of files: 8
13/01/30 18:56:00 INFO fs.TestDFSIO: Total MBytes processed: 8192
13/01/30 18:56:00 INFO fs.TestDFSIO:     Throughput mb/sec: 41.651837012782316
13/01/30 18:56:00 INFO fs.TestDFSIO: Average IO rate mb/sec: 93.37788391113281
13/01/30 18:56:00 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

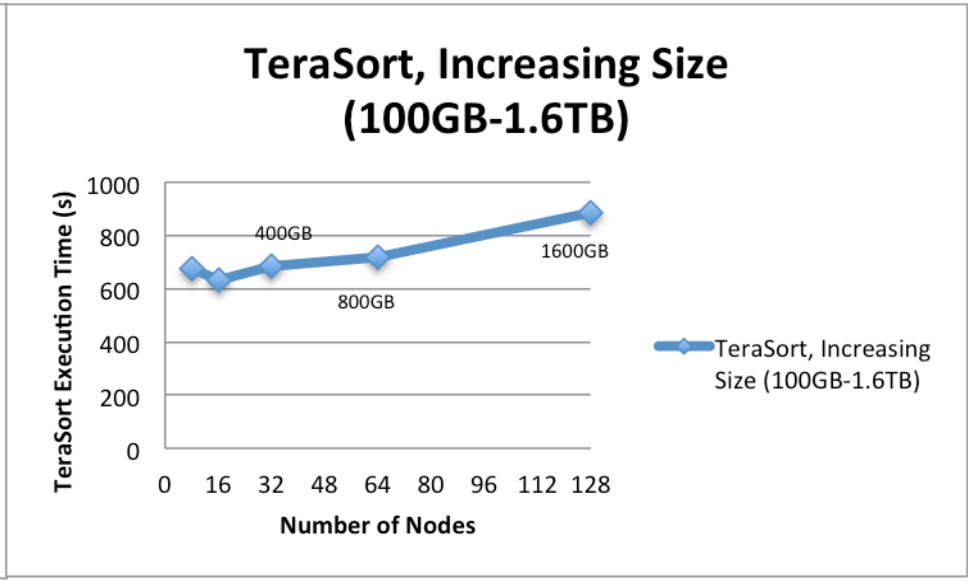


Results from the TestDFSIO benchmark. Runs were conducted using 4 datanodes and 2 map tasks. Average write I/O rates are presented.

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 – Distributed Copy

- **Dynamic cluster set up => requires good performance for copying data between lustre and HDFS filesystems.**

- **Copy performance benchmarked with varying node counts and map tasks.**

- **HDFS set up with default replication (=3) and with no replication (=1).**

Hadoop – Distributed Copy Results Default Replication (3)

Table 1. Performance results for bulk copy between parallel lustre filesystem and HDFS. Tests were conducted using varying number of nodes and map tasks for transferring 32 files with a total size of 512GB.

Nodes	#Map Tasks	Copy Rate (MB/s)
8	2	183.64
16	2	220.57
16	4	276.09
16	8	387.51
32	8	618.99
32	16	779.03

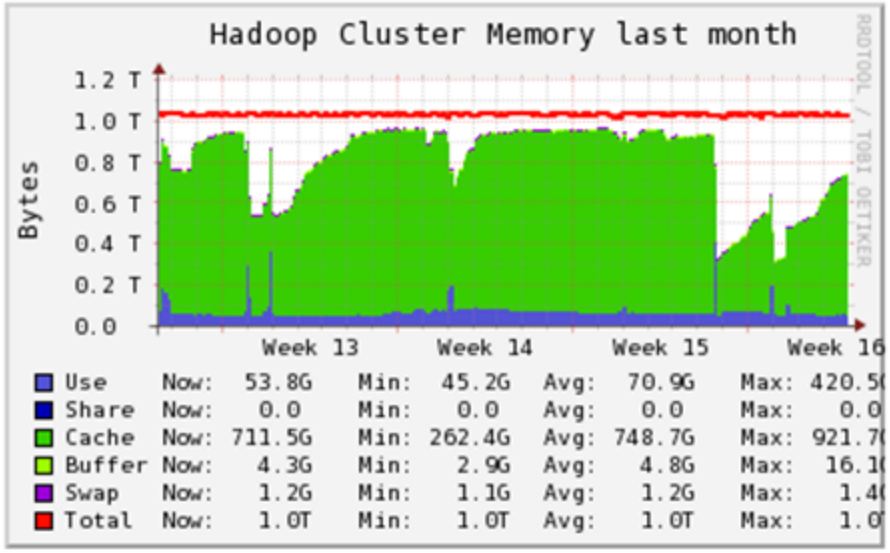
Hadoop – Distributed Copy Results Replication=1

- With replication=1, we get close to 3GB/s with distributed copy. As discussed, the architecture is scalable and this is seen in the results.

Nodes	Map Tasks	Effective Copy Rate (MB/s)
8	4	530.66
8	8	1018.03
16	8	793.17
16	16	1814.15
32	32	2995.93

Persistent Hadoop Cluster – IO Node + 16 compute nodes

- **Persistent Hadoop cluster set up for UCSD class. Set up including ganglia based monitoring.**
- **Tools used included – HIVE, Hbase, PIG, Rhadoop, Mahout, and Spark.**



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 – Realistic Examples

- Postprocessing results from FORTRAN analysis for millions of records

- Analyzing millions of variations within genomes
  - need to use (and install) special parsing libraries
  - need to use newer version of Python
  - input file “not exactly” Hadoop ready
  - Hadoop output “not exactly” what we want

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

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. Distributed copy performance allows for rapid data movement.
- High speed Infiniband network allows for significant gains in performance using SSDs.
- TeraSort benchmark shows good performance and scaling on Gordon.
- Future improvements/tests on Gordon include using HADOOP-RDMA, Mellanox UDA, and by using lustre filesystem for storage.