

Introduction to Hadoop and Spark & Running on Comet and Gordon

Mahidhar Tatineni

UCSB, Feb 22, 2016



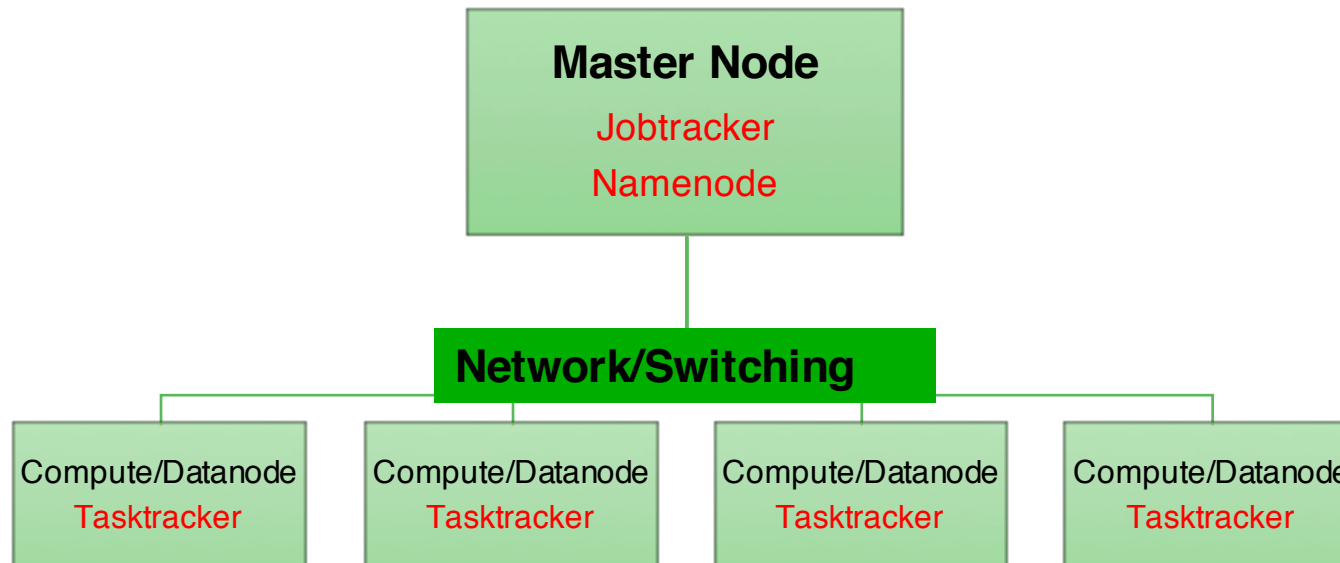
Overview

- 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.

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](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

<Hello, 1>
<World, 1>
<Bye, 1>
<World, 1>

TASK 2

< Hello, 1>
< Hadoop, 1>
< Goodbye, 1>
<Hadoop, 1>

[*http://hadoop.apache.org/docs/r0.18.3/mapred_tutorial.html](http://hadoop.apache.org/docs/r0.18.3/mapred_tutorial.html)

Simple Example (Contd)

- Step 2 : Combine on each node, sorted:

<Bye, 1>

<Hello, 1>

<World, 2>

< Goodbye, 1>

< Hadoop, 2>

<Hello, 1>

- Step 3 : Global reduce:

<Bye, 1>

<Goodbye, 1>

<Hadoop, 2>

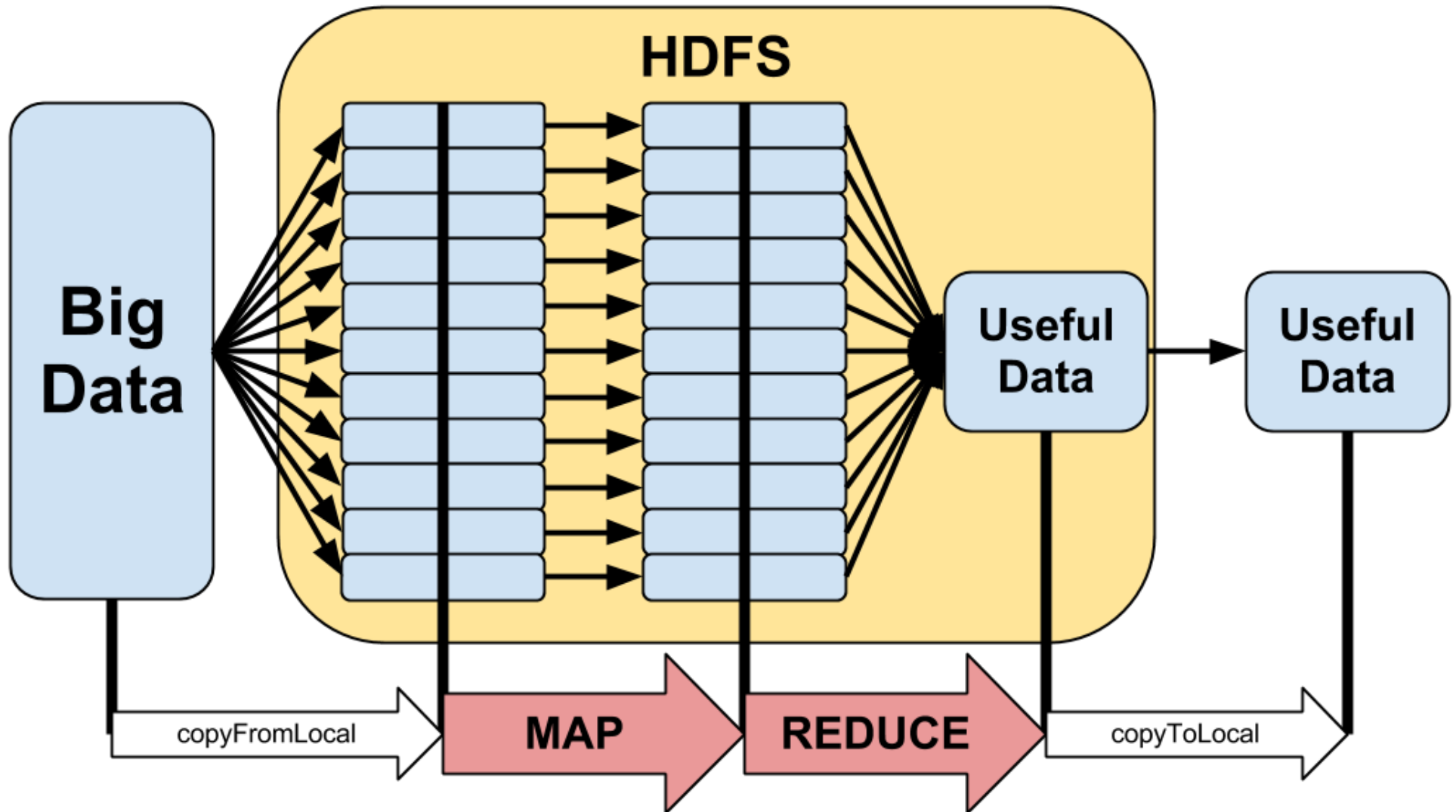
<Hello, 2>

<World, 2>

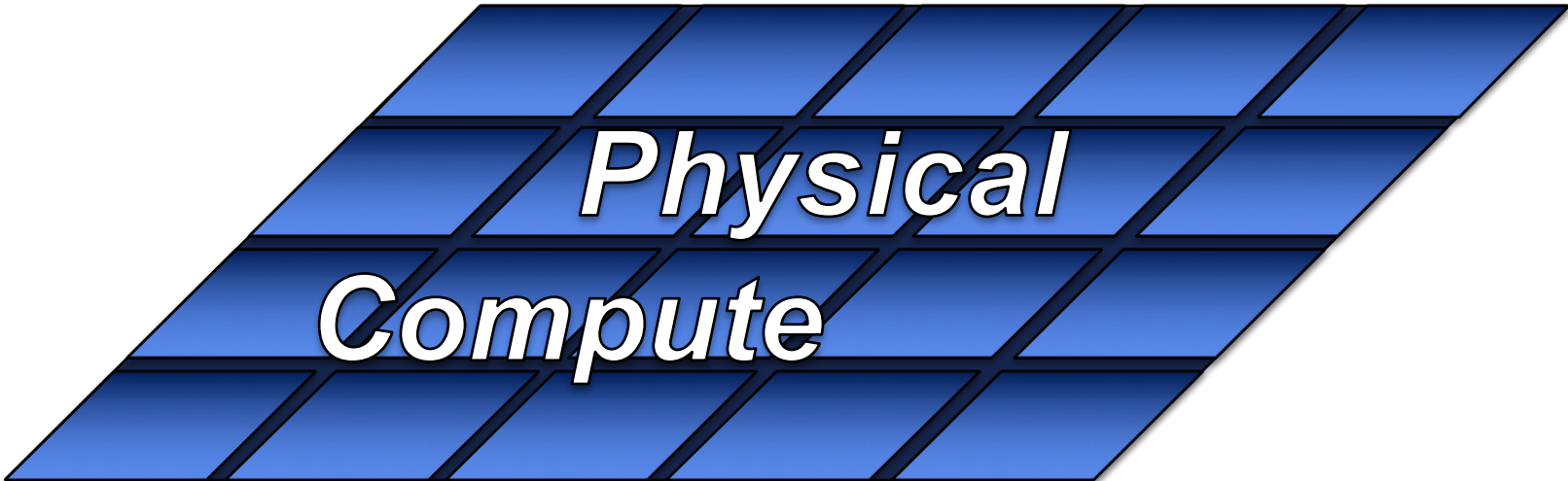
Map/Reduce Execution Process

- **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 Workflow



Add Data Analysis to Existing Compute Infrastructure

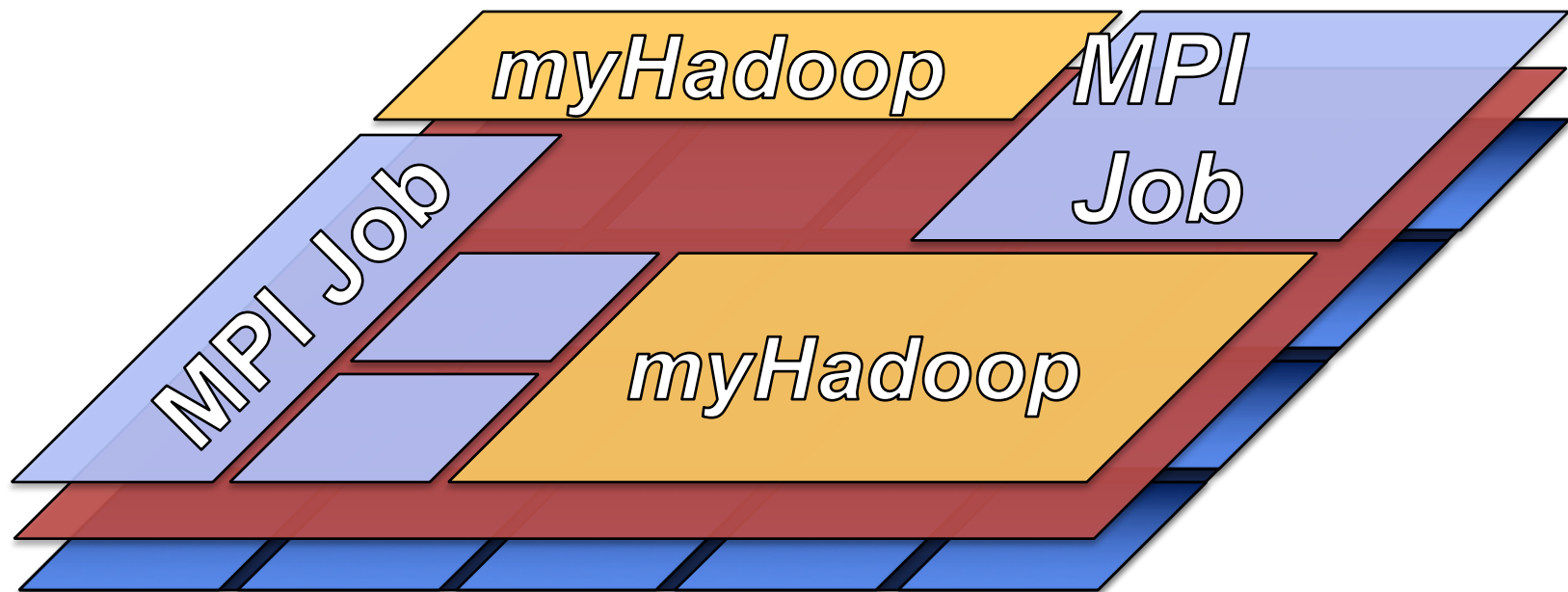


*Physical
Compute*

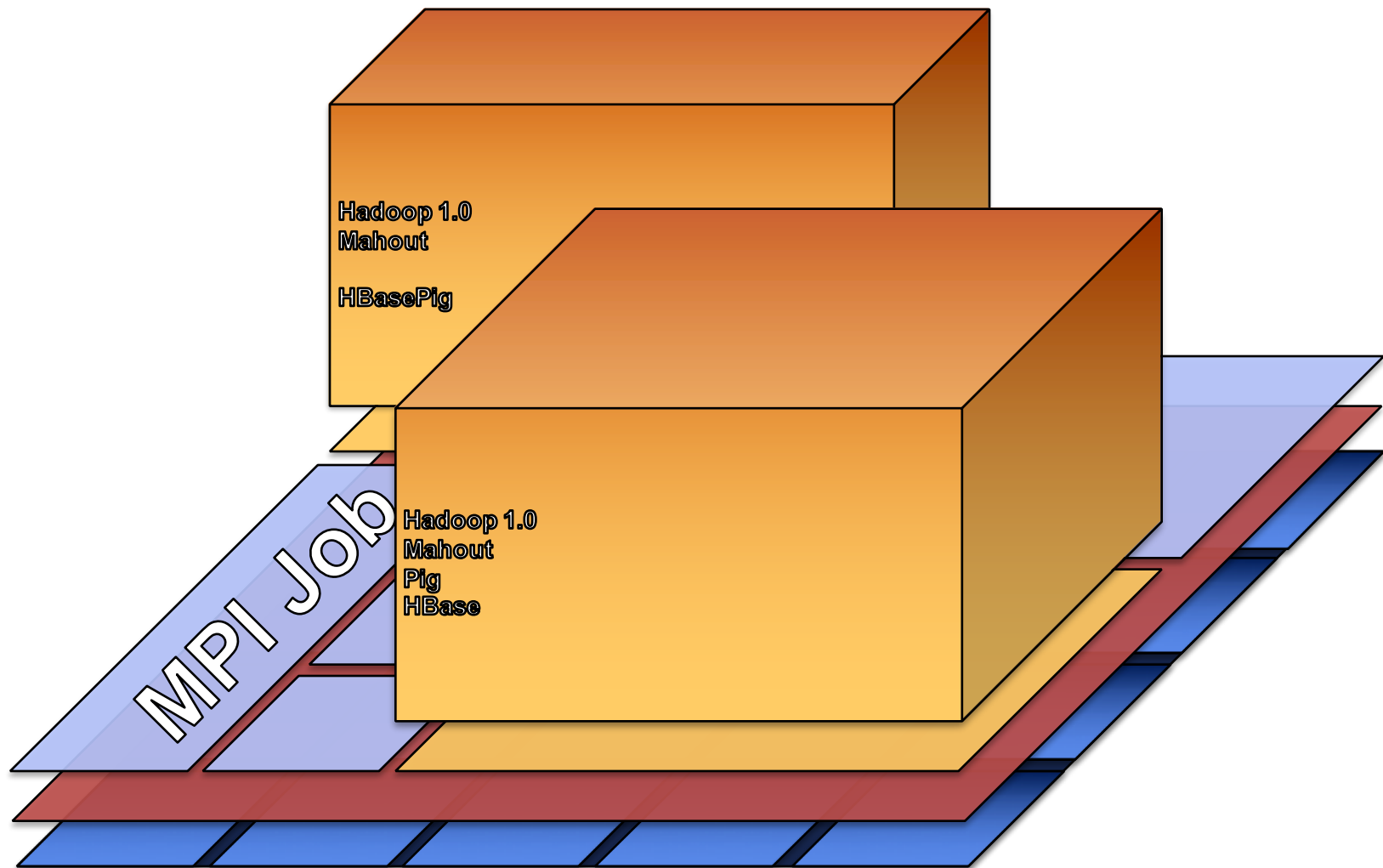
Add Data Analysis to Existing Compute Infrastructure



Add Data Analysis to Existing Compute Infrastructure



Add Data Analysis to Existing Compute Infrastructure



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 – Tested on SDSC Gordon
 - SLURM – Tested on SDSC Comet, TACC Stampede
 - Grid Engine
 - Can support LSF, PBSpro, Condor easily (need testbeds)

Interactive Hadoop Cluster on Gordon

```
### 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 hdfsdire/
### View file information on HDFS
$ hadoop fsck -block hdfsdire/somelocalfile
### Run a map/reduce job
$ hadoop jar somejarfile.jar -option1 -option2
### View job info after it completes
$ hadoop job -history hdfsdire/outputdir
### Shut down all Hadoop nodes
$ stop-all.sh
### Copy logfiles back from nodes
$ myhadoop-cleanup.sh
```

Interactive Hadoop Cluster on Comet

```
### Request two nodes for 20 minutes on Comet
$ /share/apps/compute/interactive/qsubi.bash -p compute --nodes=2 --ntasks-per-node=24 -t 00:20:00 --export=ALL
### 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 hdfsdire/
### View file information on HDFS
$ hadoop fsck -block hdfsdire/somelocalfile
### Run a map/reduce job
$ hadoop jar somejarfile.jar -option1 -option2
### View job info after it completes
$ hadoop job -history hdfsdire/outputdir
### Shut down all Hadoop nodes
$ stop-all.sh
### Copy logfiles back from nodes
$ myhadoop-cleanup.sh
```

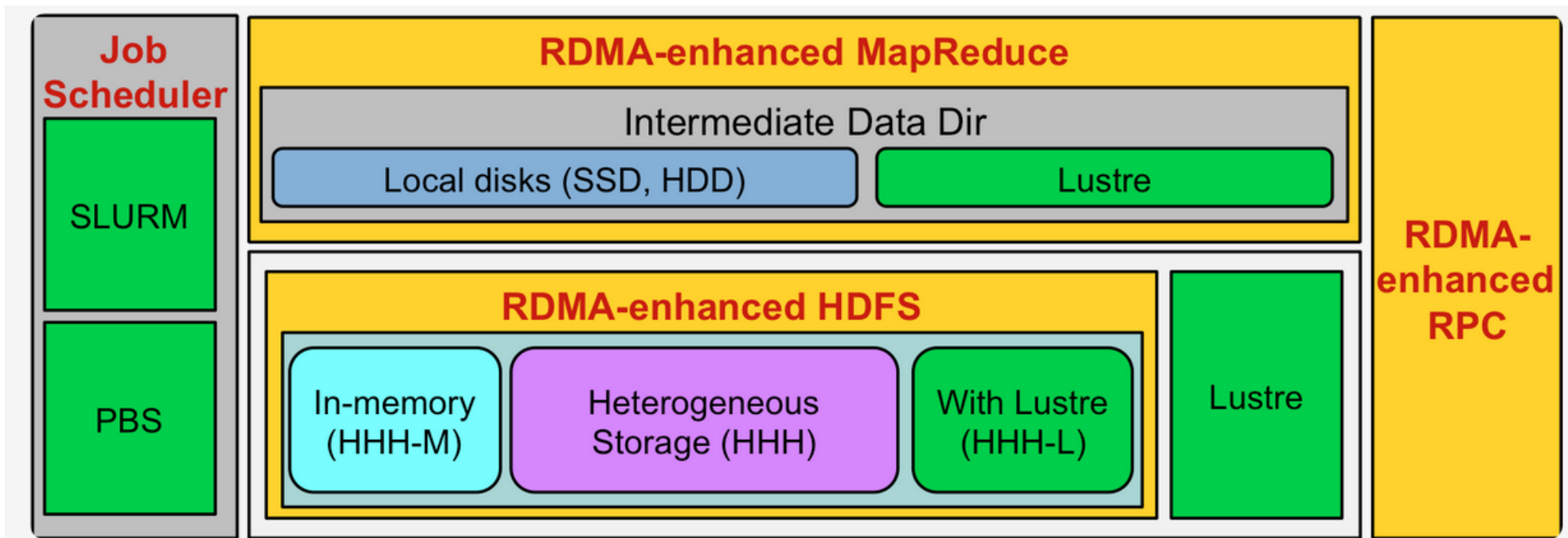
Hadoop-RDMA

Network-Based Computing Lab (Ohio State University)

- **HDFS, MapReduce, and RPC over native InfiniBand and RDMA over Converged Ethernet (RoCE).**
- **Based on Apache Hadoop.**
- **Can use myHadoop to partially set up configuration. Need to add to some of the configuration files.**
- **Version **RDMA-Apache-Hadoop-2.x 0.9.7** available on Comet:**
 - ***/share/apps/compute/hadoop***
- **More details :**
 - **<http://hibd.cse.ohio-state.edu/>**

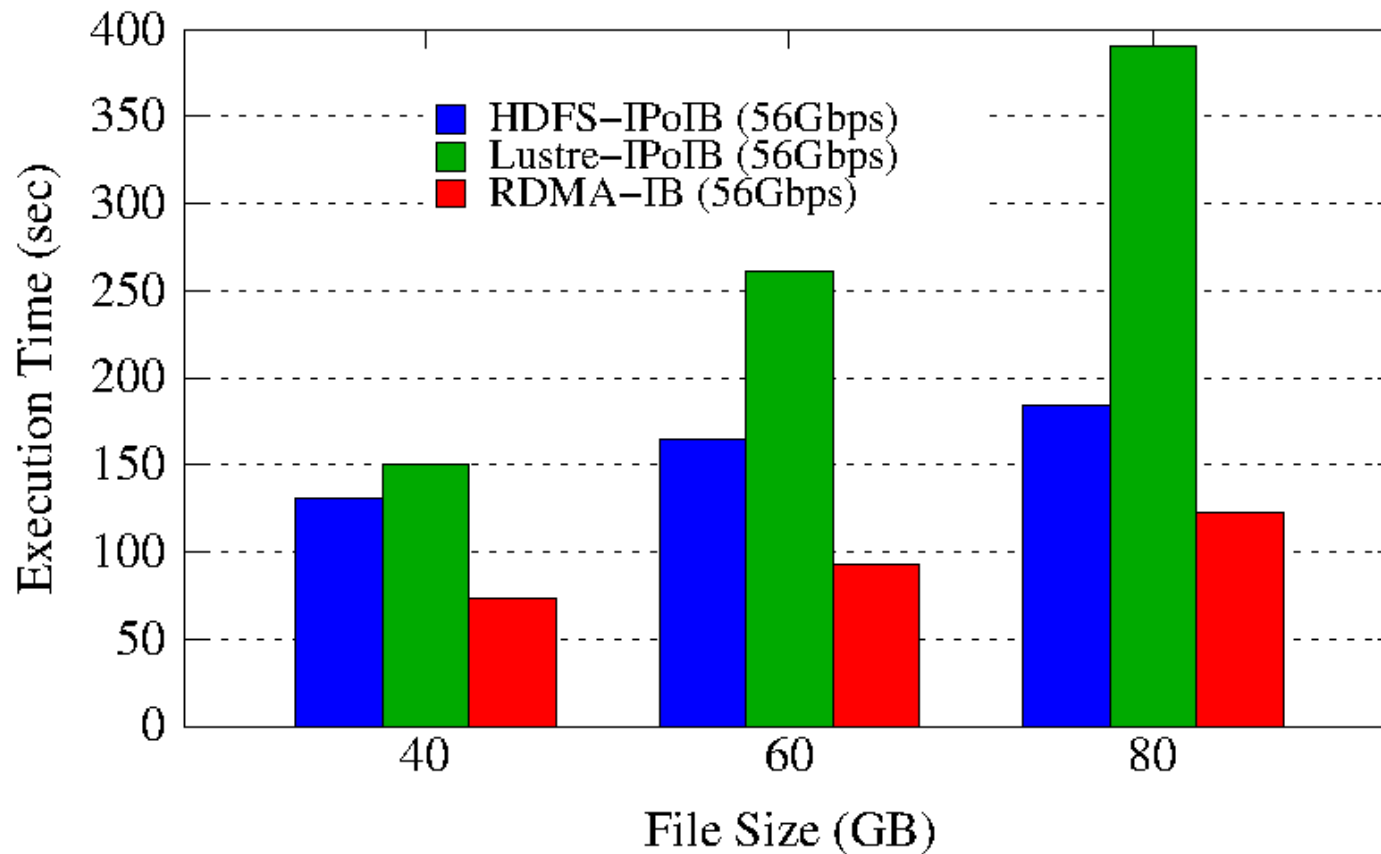
RDMA-Hadoop

- Exploit performance on modern clusters with RDMA-enabled interconnects for Big Data applications.
- Hybrid design with in-memory and heterogeneous storage (HDD, SSDs, Lustre).
- Keep compliance with standard distributions from Apache.



Sort*

HHH-L mode on Comet



*Reference results from Dr. D.K. Panda's HiBD group at OSU

<http://hibd.cse.ohio-state.edu/performance/withlustre/>

Beyond the Map Reduce Paradigm

- ***What if Application doesn't fit or is not efficient in Map Reduce Paradigm?***
 - *Interactive data exploration*
 - *Iterative data processing*

Apache Spark

- ***Advanced directed acyclic graph (DAG) execution engine***
- ***Supports cyclic data flow*** - easier to handle iterative graph algorithms, machine learning, and stream processing
- ***In-memory computing***
 - Spark keeps track of data produced during the operations
 - Allows for storing working datasets in memory (with graceful spill over to disk).
 - Data can be shared across DAGS, between iterations, and for reuse.
- ***Java, Scala, Python, R***
- ***Existing optimized libraries*** - graph analytics, machine learning, streaming applications

Spark on Comet and Gordon

- *myHadoop framework can handle Spark cluster configuration and startup.*
- *Spark can use HDFS for storage or directly use parallel filesystems (Lustre based) on Comet and Gordon.*
- *Communication over IPolB network.*

Sample Spark script

```
#!/bin/bash
#SBATCH --job-name="graphx-demo"
#SBATCH --output="graphx-demo.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodelist=2
#SBATCH --ntasks-per-node=24
#SBATCH --export=ALL
#SBATCH -t 00:30:00

### Environment setup for Hadoop and Spark
module load spark
export PATH=/opt/hadoop/2.6.0/sbin:$PATH
export HADOOP_CONF_DIR=$HOME/mycluster.conf
export WORKDIR=`pwd`
```

myhadoop-configure.sh

Sample Spark script (Contd.)

```
### Start HDFS. Starting YARN isn't necessary since Spark will be running in  
### standalone mode on our cluster.
```

start-dfs.sh

```
### Load in the necessary Spark environment variables  
source $HADOOP_CONF_DIR/spark/spark-env.sh
```

myspark start

```
### Copy the data into HDFS
```

```
hdfs dfs -mkdir -p /user/$USER
```

```
hdfs dfs -put $WORKDIR/facebook_combined.txt /user/$USER/
```

```
run-example org.apache.spark.examples.graphx.LiveJournalPageRank facebook_combined.txt --  
numEPart=8
```

```
### Shut down Spark and HDFS
```

myspark stop

stop-dfs.sh

```
### Clean up
```

myhadoop-cleanup.sh

Thanks!
Questions: Email mahidhar@sdsc.edu