Big Data Analytics with the OSU HiBD Stack at SDSC

Mahidhar Tatineni
OSU Booth Talk, SC18, Dallas

SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

Comet

“HPC for the long tail of science”

iPhone panorama photograph of 1 of 2 server rows

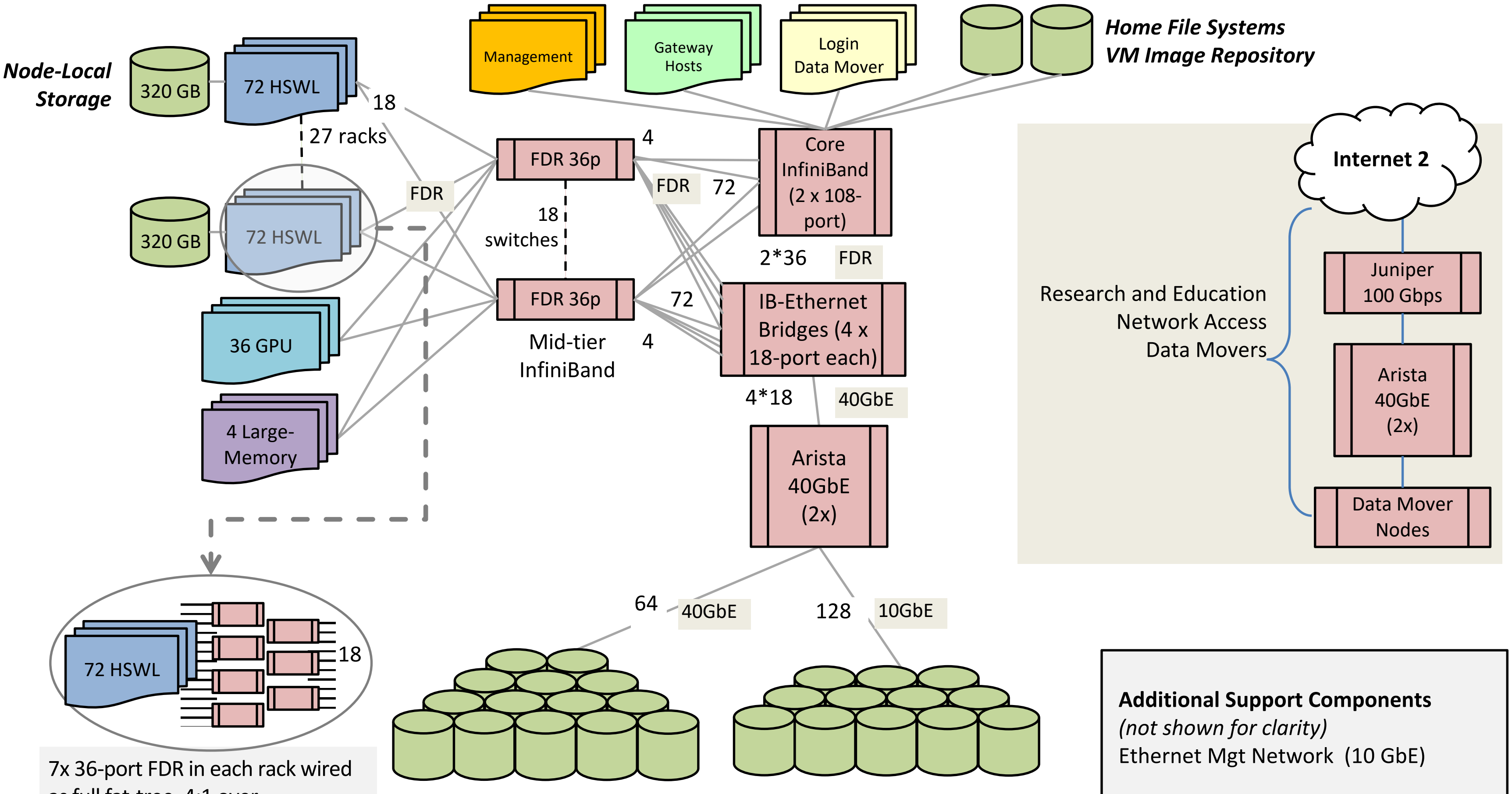
Comet: System Characteristics

- **Total peak flops ~2.1 PF**
- **Dell primary integrator**
  - Intel Haswell processors w/ AVX2
  - Mellanox FDR InfiniBand
- **1,944 standard compute nodes (46,656 cores)**
  - Dual CPUs, each 12-core, 2.5 GHz
  - 128 GB DDR4 2133 MHz DRAM
  - 2*160GB GB SSDs (local disk)
- **72 GPU nodes**
  - 36 nodes same as standard nodes *plus* Two NVIDIA K80 cards, each with dual Kepler3 GPUs
  - 36 nodes, with 4 P100 GPUs per node
- **4 large-memory nodes**
  - 1.5 TB DDR4 1866 MHz DRAM
  - Four Haswell processors/node
  - 64 cores/node

- **Hybrid fat-tree topology**
  - FDR (56 Gbps) InfiniBand
  - Rack-level (72 nodes, 1,728 cores) full bisection bandwidth
  - 4:1 oversubscription cross-rack
- **Performance Storage (Aeon)**
  - 7.6 PB, 200 GB/s; Lustre
  - Scratch & Persistent Storage segments
- **Durable Storage (Aeon)**
  - 6 PB, 100 GB/s; Lustre
  - Automatic backups of critical data
- **Home directory storage**
- **Gateway hosting nodes**
- **Virtual image repository**
- **100 Gbps external connectivity to Internet2 & ESNet**

Comet Network Architecture

InfiniBand compute, Ethernet Storage



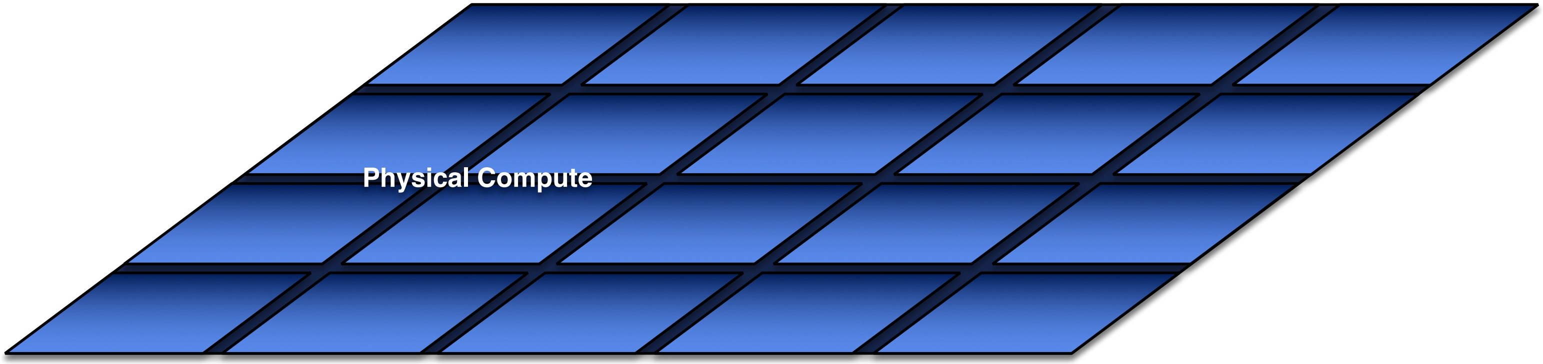
7x 36-port FDR in each rack wired as full fat-tree. 4:1 over subscription between racks.

Performance Storage
7.7 PB, 200 GB/s
32 storage servers

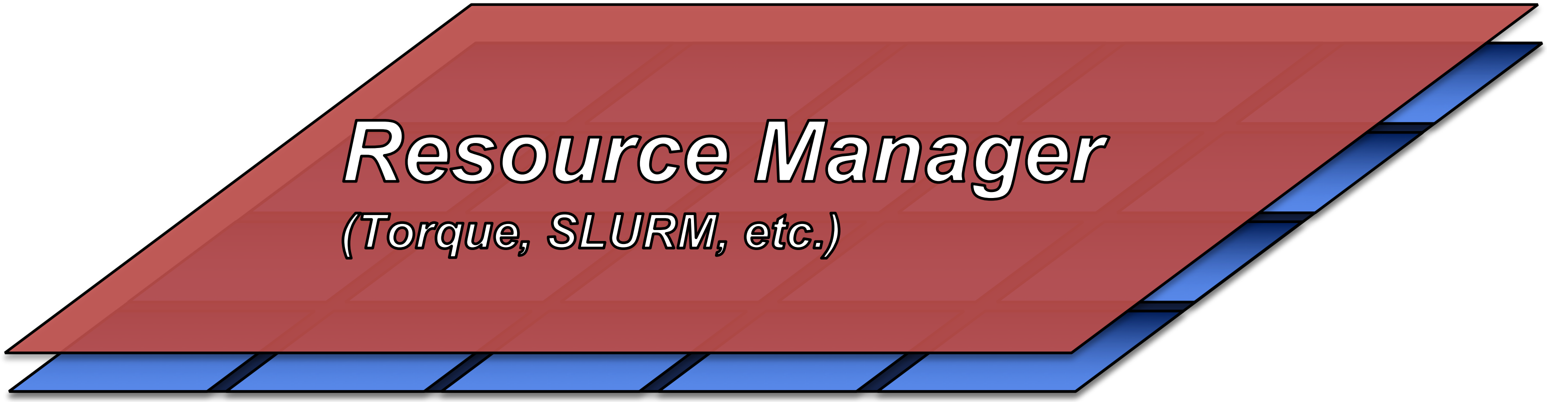
Durable Storage
6 PB, 100 GB/s
64 storage servers

Additional Support Components
(not shown for clarity)
Ethernet Mgt Network (10 GbE)

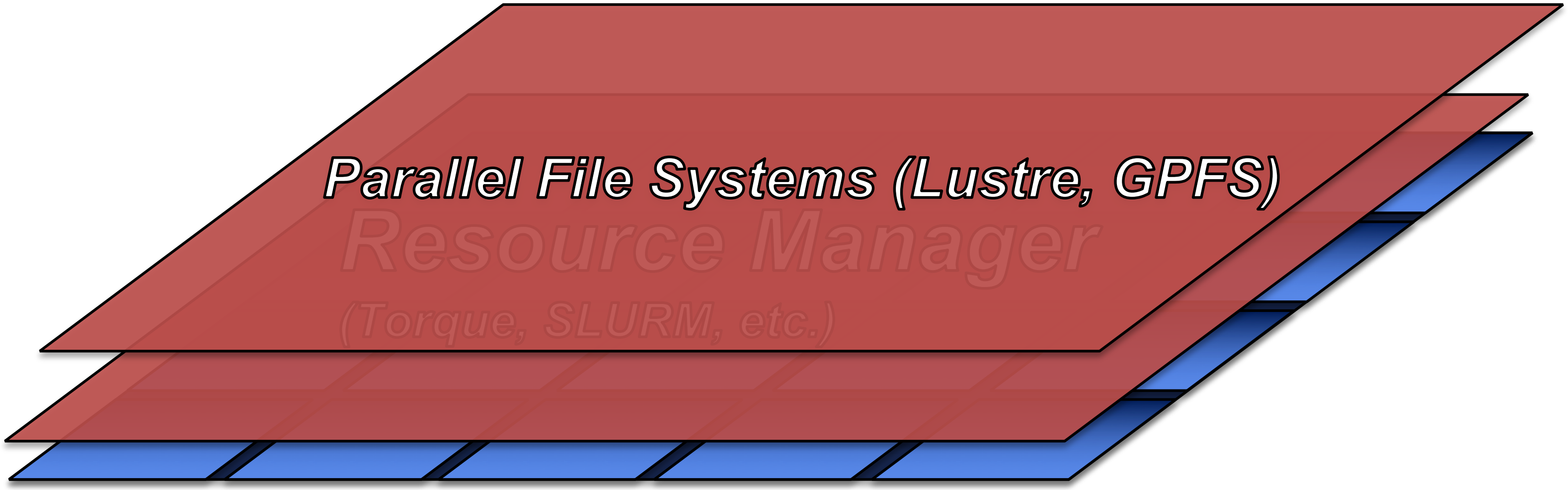
Can We Run Big Data and Deep Learning Jobs on Existing HPC Infrastructure?



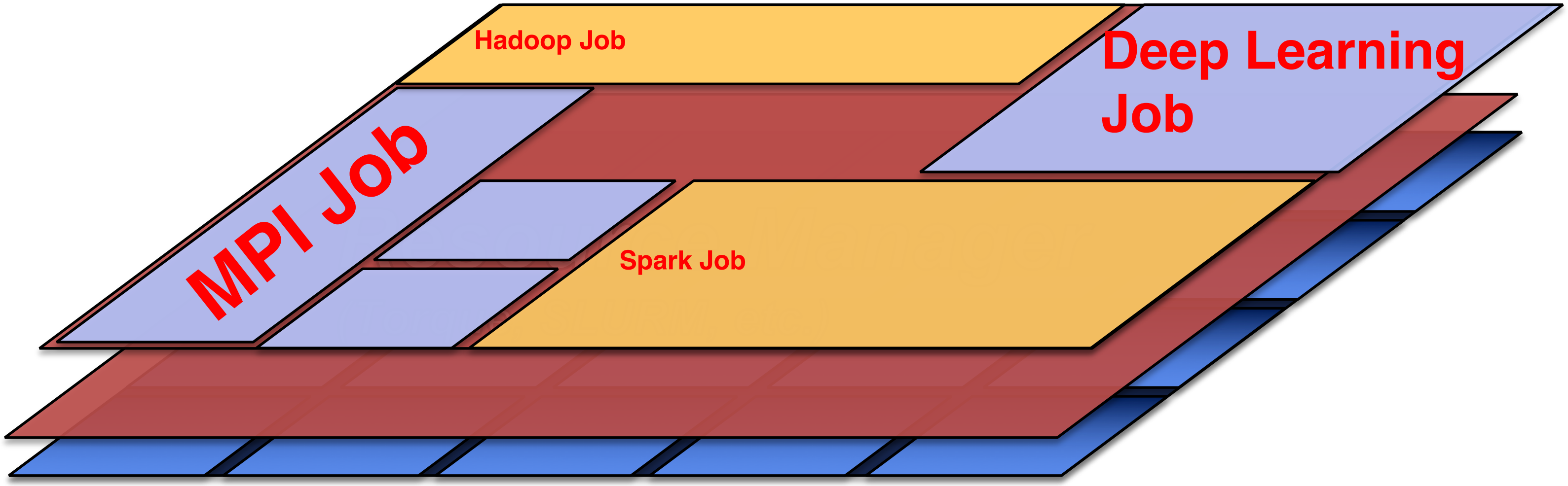
Can We Run Big Data and Deep Learning Jobs on Existing HPC Infrastructure?



Can We Run Big Data and Deep Learning Jobs on Existing HPC Infrastructure?



Can We Run Big Data and Deep Learning Jobs on Existing HPC Infrastructure?

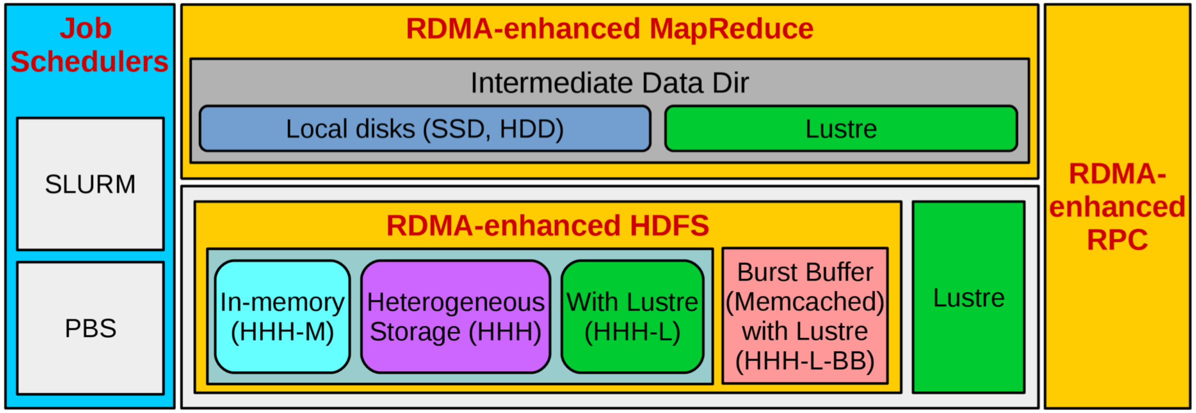


RDMA-Hadoop and RDMA-Spark

Network-Based Computing Lab (Ohio State University)

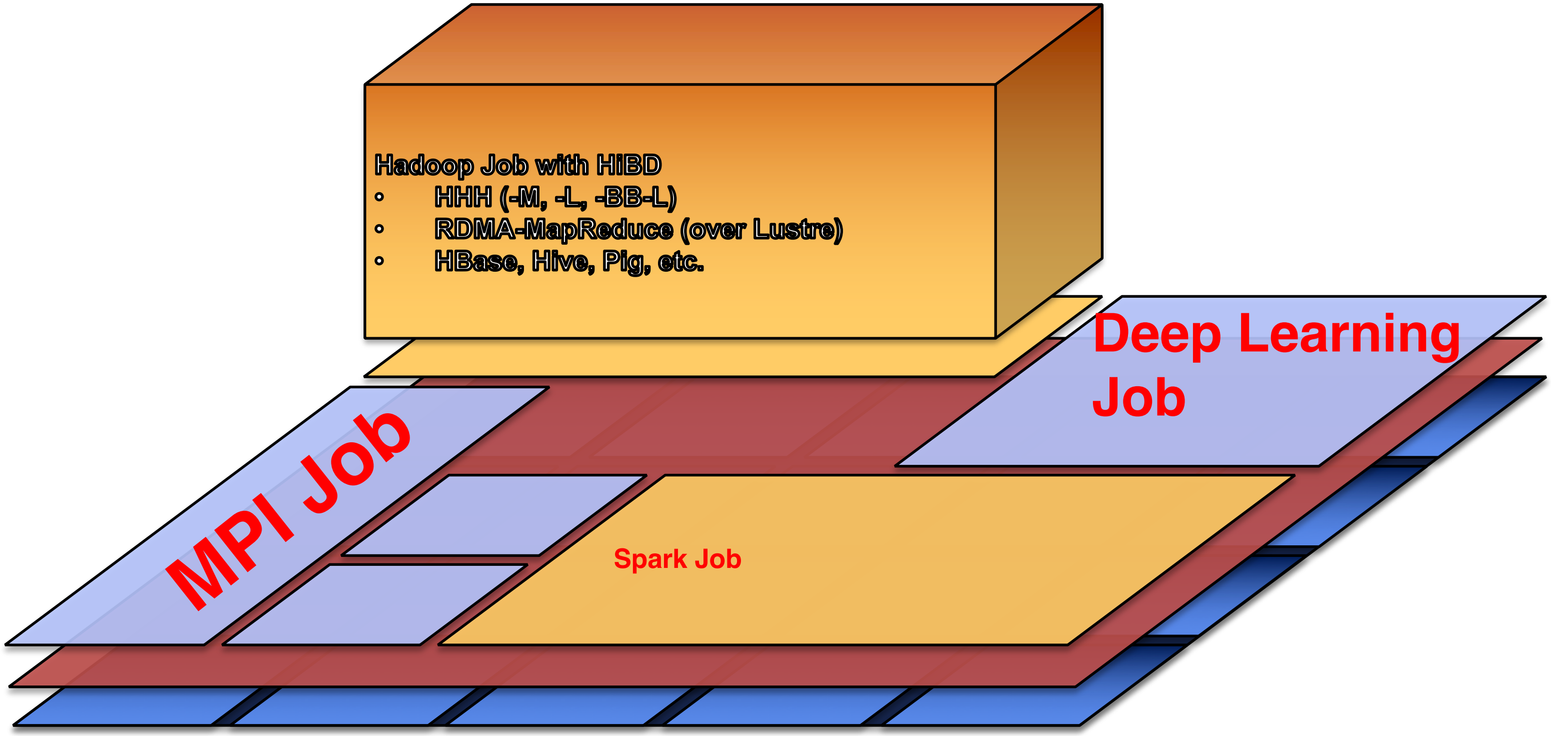
- HDFS, MapReduce, and RPC over native InfiniBand and RDMA over Converged Ethernet (RoCE).
- Based on Apache distributions of Hadoop and Spark.
- Version RDMA-Apache-Hadoop-2.x 1.3.5 (based on Apache Hadoop 2.8.0) available on Comet
- Version RDMA-Spark 0.9.5 (based on Apache Spark 2.1.0) is available on Comet.
- More details on the RDMA-Hadoop and RDMA-Spark projects at:
  - http://hibd.cse.ohio-state.edu/

Different Modes of RDMA for Apache Hadoop 2.x

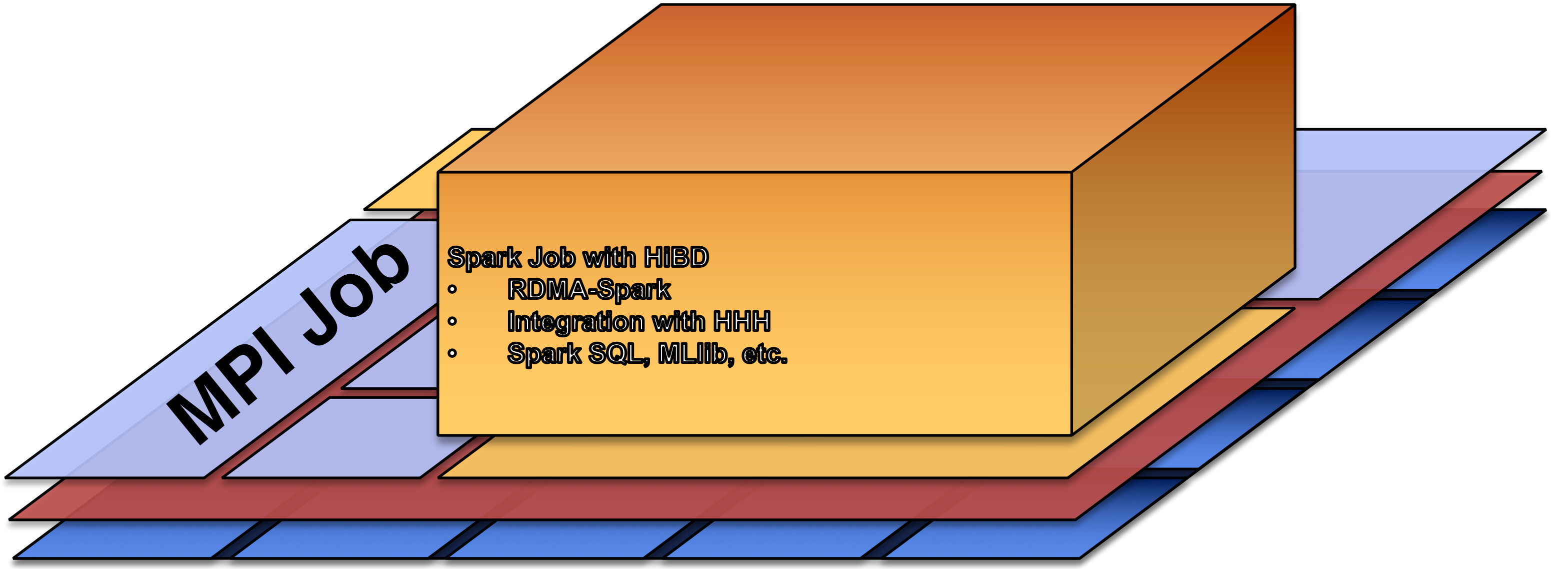


- **HHH**: Heterogeneous storage devices with hybrid replication schemes are supported in this mode of operation to have better fault-tolerance as well as performance. This mode is enabled by **default** in the package.
- **HHH-M**: A high-performance in-memory based setup has been introduced in this package that can be utilized to perform all I/O operations in-memory and obtain as much performance benefit as possible.
- **HHH-L**: With parallel file systems integrated, HHH-L mode can take advantage of the Lustre available in the cluster.
- **HHH-L-BB**: This mode deploys a Memcached-based burst buffer system to reduce the bandwidth bottleneck of shared file system access. The burst buffer design is hosted by Memcached servers, each of which has a local SSD.
- **MapReduce over Lustre, with/without local disks**: Besides, HDFS based solutions, this package also provides support to run MapReduce jobs on top of Lustre alone. Here, two different modes are introduced: with local disks and without local disks.
- **Running with Slurm and PBS**: Supports deploying RDMA for Apache Hadoop 2.x with Slurm and PBS in different running modes (HHH, HHH-M, HHH-L, and MapReduce over Lustre).

Using HiBD Packages for Big Data Processing on Existing HPC Infrastructure



Using HiBD Packages for Big Data Processing on Existing HPC Infrastructure



Spark Job with HiBD
- RDMA-Spark
- Integration with HHH
- Spark SQL, MLlib, etc.

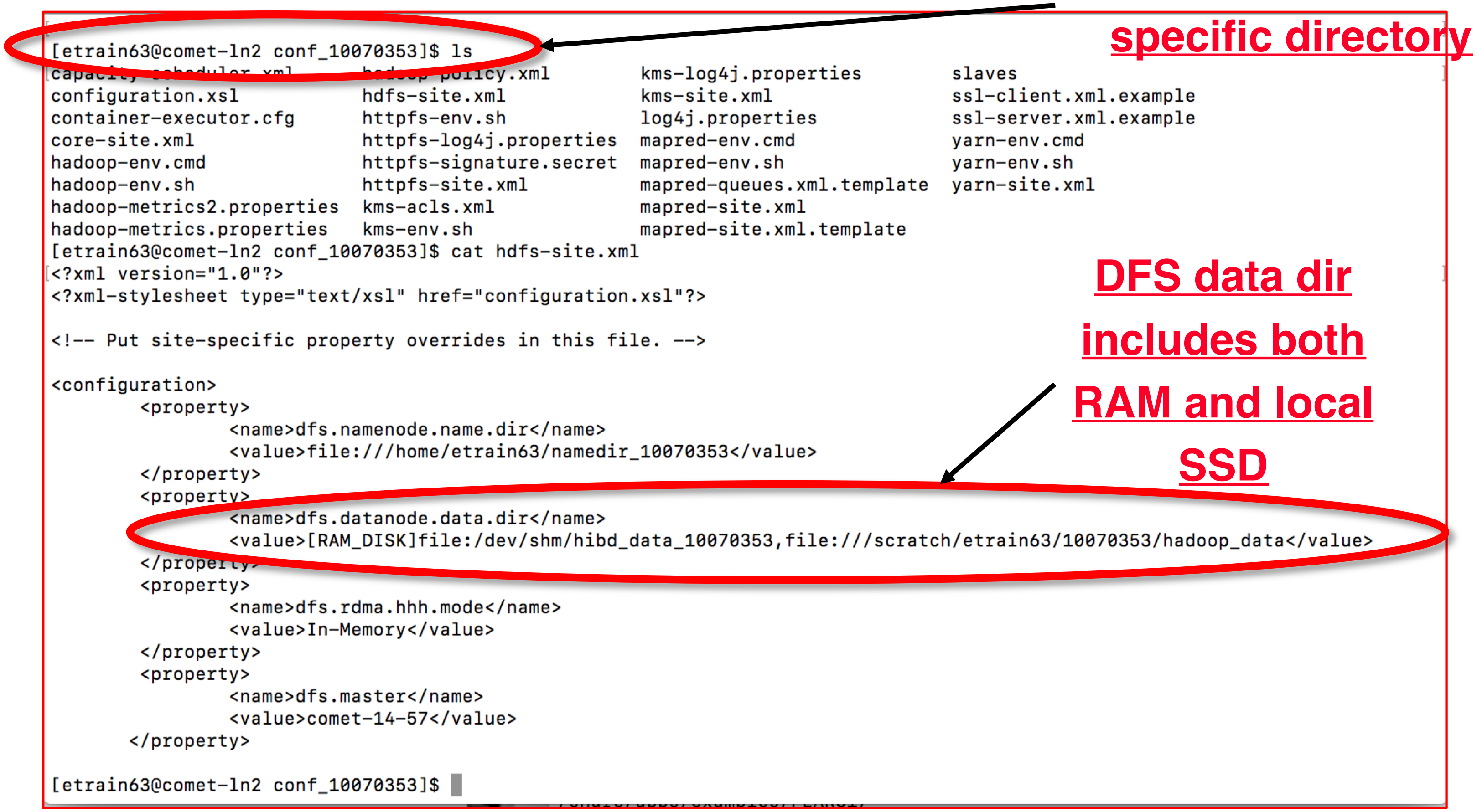
Anagram Example using HHH-M mode

```
#!/bin/bash
#SBATCH --job-name="rdmahadoopanagram"
#SBATCH --output="rdmahadoopanagram.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=24
#SBATCH -t 00:15:00

#Script request 3 nodes - one used for namenode, 2 for data nodes/processing

#Set modulepath and load RDMA Hadoop Module
export
MODULEPATH=/share/apps/compute/modulefiles/applications:$MODULEPATH
module load rdma-hadoop/2x-1.3.5
```

Anagram Example using HHH-M mode

```
#Use SLURM integrated configuration/startup script
hibd_install_configure_start.sh -s -h $HADOOP_HOME -j $JAVA_HOME -m hhh-m -r /dev/shm -S
/scratch/$USER/$SLURM_JOBID

#Commands to run ANAGRAM example
$HADOOP_HOME/bin/hdfs --config $HOME/conf_$SLURM_JOBID dfs -mkdir -p /user/$USER/input
$HADOOP_HOME/bin/hdfs --config $HOME/conf_$SLURM_JOBID dfs -put SINGLE.TXT
/user/$USER/input/SINGLE.TXT
$HADOOP_HOME/bin/hadoop --config $HOME/conf_$SLURM_JOBID jar AnagramJob.jar
/user/$USER/input/SINGLE.TXT /user/$USER/output
$HADOOP_HOME/bin/hdfs --config $HOME/conf_$SLURM_JOBID dfs -get /user/$USER/output/part*
$SLURM_WORKING_DIR

#Clean up
hibd_stop_cleanup.sh -d -h $HADOOP_HOME -m hhh-m -r /dev/shm
```

Sample configuration info



```
[etrain63@comet-ln2 conf_10070353]$ ls
capacity-scheduler.xml      hadoop-policy.xml        kms-log4j.properties        slaves
configuration.xsl           hdfs-site.xml            kms-site.xml                ssl-client.xml.example
container-executor.cfg      httpfs-env.sh            log4j.properties            ssl-server.xml.example
core-site.xml               httpfs-log4j.properties  mapred-env.cmd              yarn-env.cmd
hadoop-env.cmd              httpfs-signature.secret  mapred-env.sh               yarn-env.sh
hadoop-env.sh               httpfs-site.xml          mapred-queues.xml.template  yarn-site.xml
hadoop-metrics2.properties  kms-acls.xml             mapred-site.xml
hadoop-metrics.properties   kms-env.sh               mapred-site.xml.template
[etrain63@comet-ln2 conf_10070353]$ cat hdfs-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>file:///home/etrain63/namedir_10070353</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>[RAM_DISK]file:/dev/shm/hibd_data_10070353,file:///scratch/etrain63/10070353/hadoop_data</value>
        </property>
        <property>
                <name>dfs.rdma.hhh.mode</name>
                <value>In-Memory</value>
        </property>
        <property>
                <name>dfs.master</name>
                <value>comet-14-57</value>
       </property>

[etrain63@comet-ln2 conf_10070353]$ 
```

Configuration files in job specific directory

DFS data dir includes both RAM and local SSD

Anagram Example using HHH-L mode

```
#!/bin/bash
#SBATCH --job-name="rdmahadoopanagram"
#SBATCH --output="rdmahadoopanagram.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=24
#SBATCH -t 00:15:00

#Script request 3 nodes - one used for namenode, 2 for data nodes/processing

#Set modulepath and load RDMA Hadoop Module
export
MODULEPATH=/share/apps/compute/modulefiles/applications:$MODULEPATH
module load rdma-hadoop/2x-1.3.5
```

Anagram Example using HHH-L mode

```
#Setup Lustre location for HDFS storage and set stripe.
export HDATADIR="/oasis/scratch/comet/$USER/temp_project/HDATA"
if [ ! -d "$HDATADIR" ]; then
                mkdir -p $HDATADIR
fi
lfs setstripe --stripe-size 64m $HDATADIR

#Use SLURM integrated configuration/startup script
hibd_install_configure_start.sh -s -h $HADOOP_HOME -j $JAVA_HOME -m hhh-l -l $HDATADIR -r /dev/shm -S /scratch/$USER/$SLURM_JOBID

#Commands to run ANAGRAM example
$HADOOP_HOME/bin/hdfs --config $HOME/conf_$SLURM_JOBID dfs -mkdir -p /user/$USER/input
$HADOOP_HOME/bin/hdfs --config $HOME/conf_$SLURM_JOBID dfs -put SINGLE.TXT /user/$USER/input/SINGLE.TXT
$HADOOP_HOME/bin/hadoop --config $HOME/conf_$SLURM_JOBID jar AnagramJob.jar /user/$USER/input/SINGLE.TXT /user/$USER/output
$HADOOP_HOME/bin/hdfs --config $HOME/conf_$SLURM_JOBID dfs -get /user/$USER/output/part* $S
LURM_WORKING_DIR

#Clean up
hibd_stop_cleanup.sh -d -h $HADOOP_HOME -m hhh-l -l $HDATADIR -r /dev/shm
```

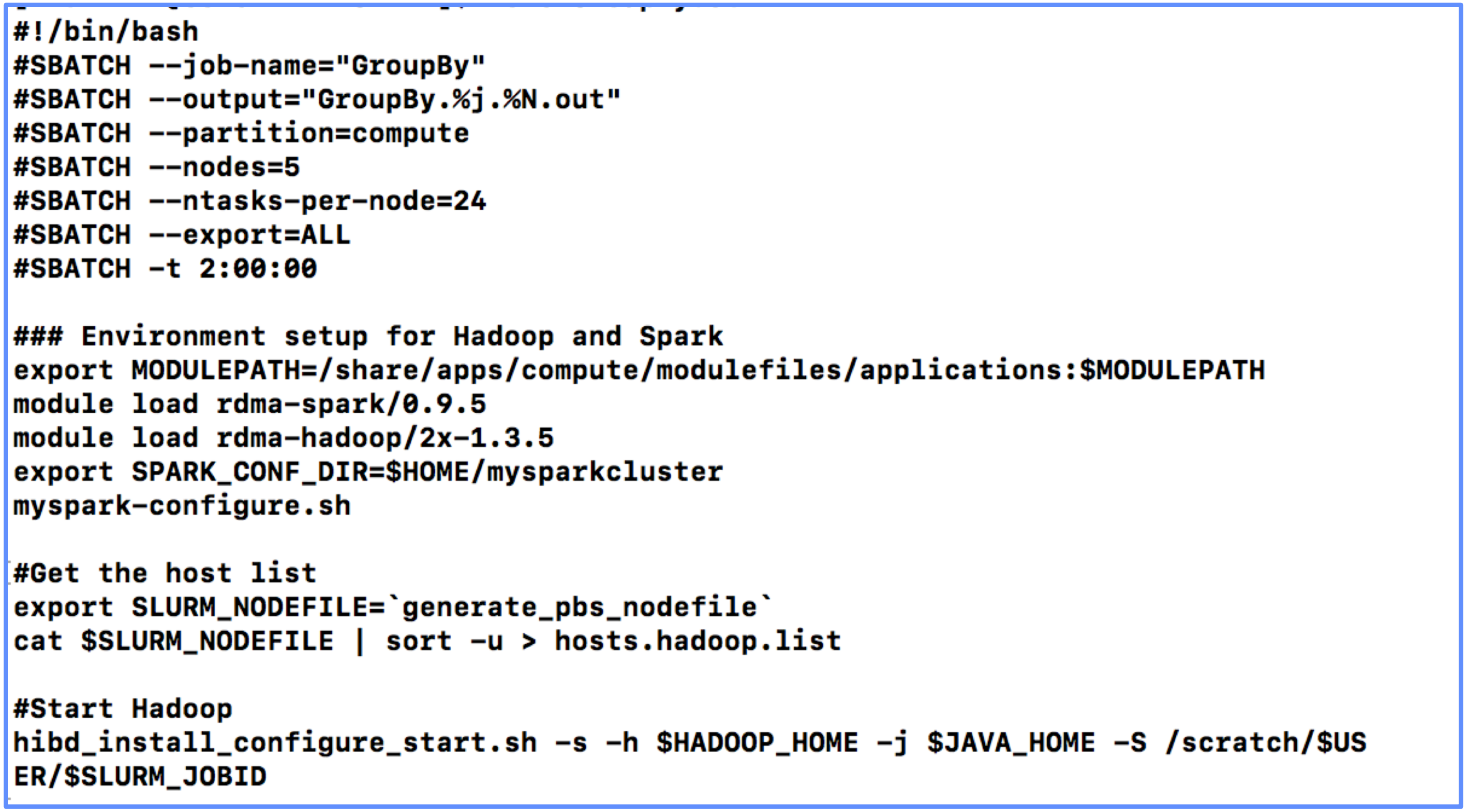
Typical RDMA Spark Configuration parameters

- **spark-env.sh**
  - export JAVA_HOME=/home/xxx/software/jdk1.7.0_79
  - export SPARK_CONF_DIR=/home/xxx/spark_2.1//conf/
  - export SPARK_LOCAL_IP=`hostname -s`.ibnet
  - export SPARK_MASTER_IP=comet-13-03.ibnet
  - export SPARK_MASTER_HOST=comet-13-03.ibnet
  - export SPARK_WORKER_MEMORY=96g
  - export SPARK_WORKER_CORES=24
  - export SPARK_WORKER_DIR=/scratch/xxx/8357275/spark//workerdir
  - export SPARK_LOCAL_DIRS=/scratch/xxx/8357275/spark//localdir
  - export SPARK_DAEMON_MEMORY=2g

Spark Configuration (Con’t)

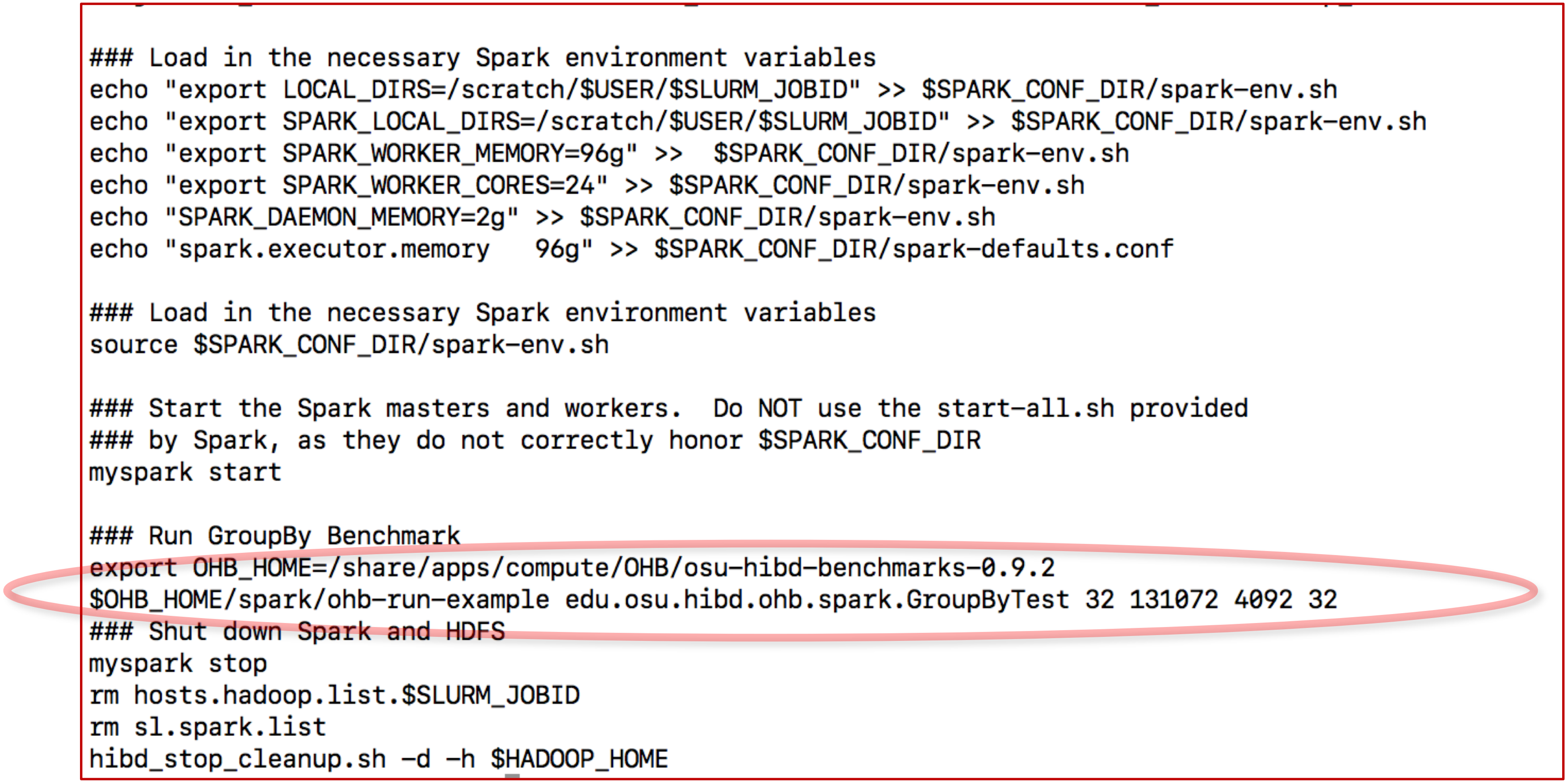
- **spark-defaults.conf**
  - spark.master   spark://comet-13-03.ibnet:7077
  - spark.executor.memory   96g
  - spark.ib.enabled true
  - hadoop.ib.enabled true
  - spark.executor.extraLibraryPath /home/xxx/spark_2.1//lib/native/Linux-amd64-64:/home/xxx/rdma-hadoop-2.x-1.3.5/lib/native
  - spark.driver.extraLibraryPath /home/xxx/spark_2.1//lib/native/Linux-amd64-64:/home/xxx/rdma-hadoop-2.x-1.3.5/lib/native

Spark submit script for GroupBy Benchmark



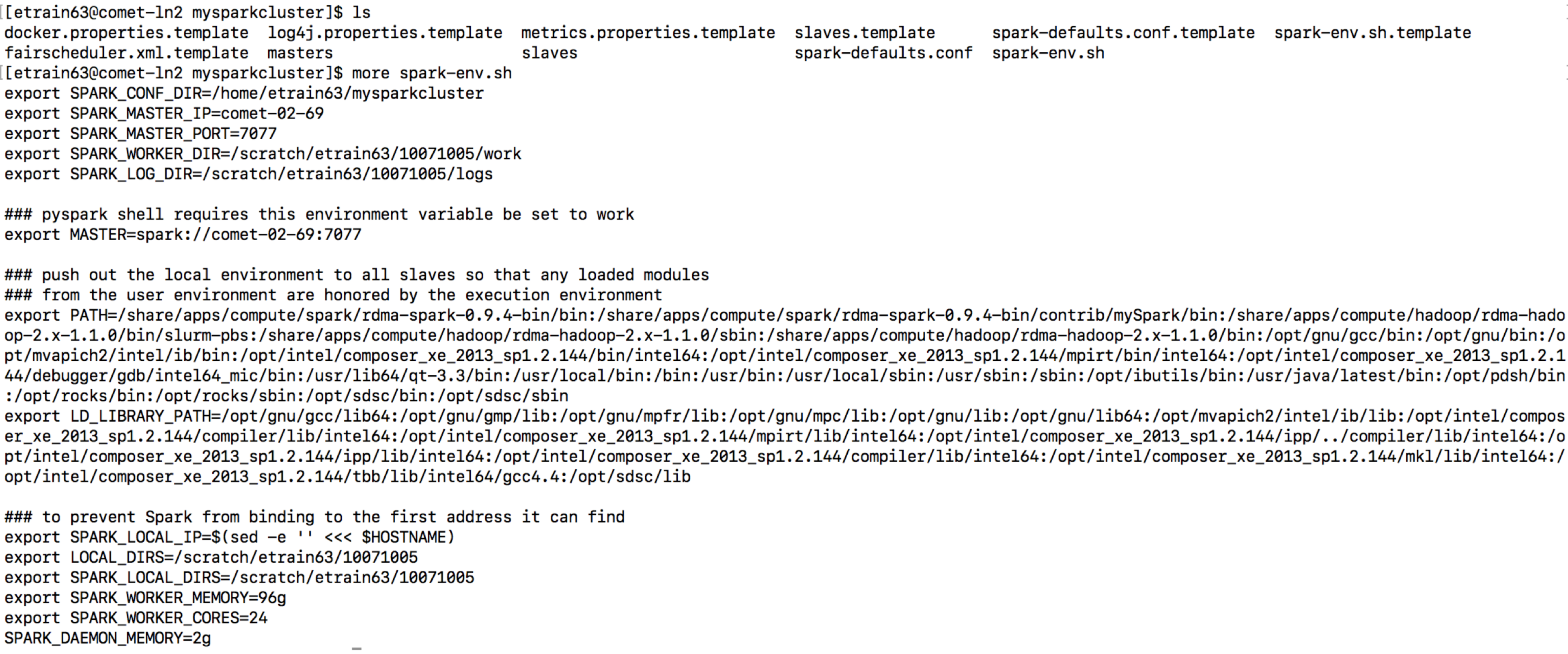
```
#!/bin/bash
#SBATCH --job-name="GroupBy"
#SBATCH --output="GroupBy.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=5
#SBATCH --ntasks-per-node=24
#SBATCH --export=ALL
#SBATCH -t 2:00:00

### Environment setup for Hadoop and Spark
export MODULEPATH=/share/apps/compute/modulefiles/applications:$MODULEPATH
module load rdma-spark/0.9.5
module load rdma-hadoop/2x-1.3.5
export SPARK_CONF_DIR=$HOME/mysparkcluster
myspark-configure.sh

#Get the host list
export SLURM_NODEFILE=`generate_pbs_nodefile`
cat $SLURM_NODEFILE | sort -u > hosts.hadoop.list

#Start Hadoop
hibd_install_configure_start.sh -s -h $HADOOP_HOME -j $JAVA_HOME -S /scratch/$US
ER/$SLURM_JOBID
```

Spark submit script for GroupBy Benchmark (Continued)



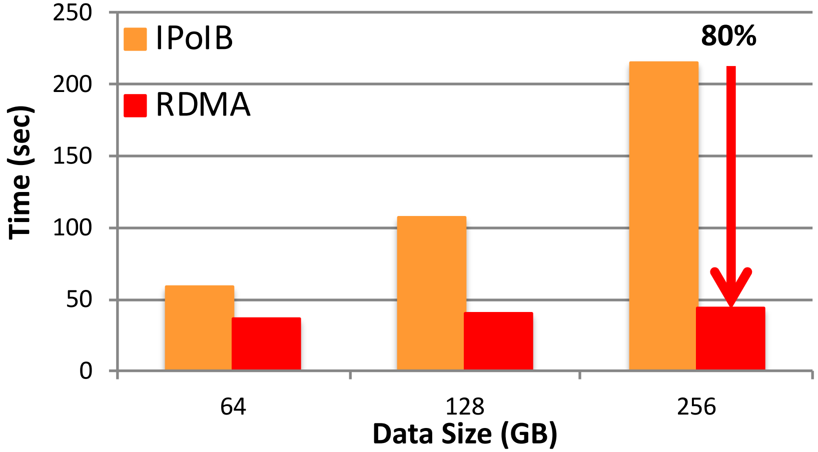
```
### Load in the necessary Spark environment variables
echo "export LOCAL_DIRS=/scratch/$USER/$SLURM_JOBID" >> $SPARK_CONF_DIR/spark-env.sh
echo "export SPARK_LOCAL_DIRS=/scratch/$USER/$SLURM_JOBID" >> $SPARK_CONF_DIR/spark-env.sh
echo "export SPARK_WORKER_MEMORY=96g" >>  $SPARK_CONF_DIR/spark-env.sh
echo "export SPARK_WORKER_CORES=24" >> $SPARK_CONF_DIR/spark-env.sh
echo "SPARK_DAEMON_MEMORY=2g" >> $SPARK_CONF_DIR/spark-env.sh
echo "spark.executor.memory   96g" >> $SPARK_CONF_DIR/spark-defaults.conf

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

### Start the Spark masters and workers.  Do NOT use the start-all.sh provided
### by Spark, as they do not correctly honor $SPARK_CONF_DIR
myspark start

### Run GroupBy Benchmark
export OHB_HOME=/share/apps/compute/OHB/osu-hibd-benchmarks-0.9.2
$OHB_HOME/spark/ohb-run-example edu.osu.hibd.ohb.spark.GroupByTest 32 131072 4092 32
### Shut down Spark and HDFS
myspark stop
rm hosts.hadoop.list.$SLURM_JOBID
rm sl.spark.list
hibd_stop_cleanup.sh -d -h $HADOOP_HOME
```

Example configuration file generation for Spark

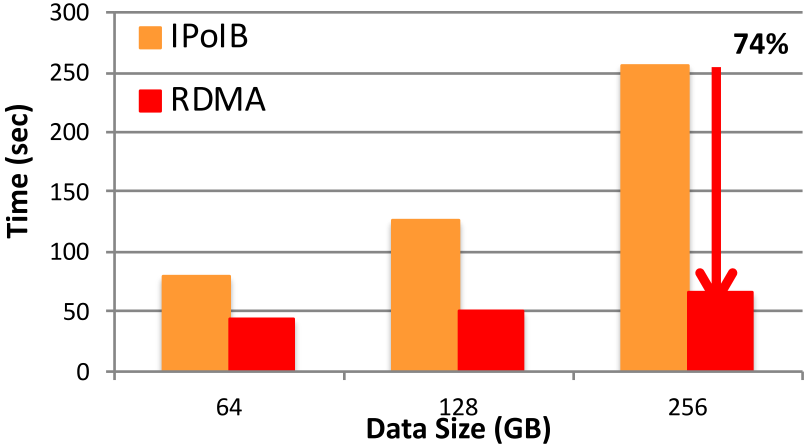


```
[etrain63@comet-ln2 mysparkcluster]$ ls
docker.properties.template  log4j.properties.template  metrics.properties.template  slaves.template      spark-defaults.conf.template  spark-env.sh.template
fairscheduler.xml.template  masters                    slaves                       spark-defaults.conf  spark-env.sh
[etrain63@comet-ln2 mysparkcluster]$ more spark-env.sh
export SPARK_CONF_DIR=/home/etrain63/mysparkcluster
export SPARK_MASTER_IP=comet-02-69
export SPARK_MASTER_PORT=7077
export SPARK_WORKER_DIR=/scratch/etrain63/10071005/work
export SPARK_LOG_DIR=/scratch/etrain63/10071005/logs

### pyspark shell requires this environment variable be set to work
export MASTER=spark://comet-02-69:7077

### push out the local environment to all slaves so that any loaded modules
### from the user environment are honored by the execution environment
export PATH=/share/apps/compute/spark/rdma-spark-0.9.4-bin/bin:/share/apps/compute/spark/rdma-spark-0.9.4-bin/contrib/mySpark/bin:/share/apps/compute/hadoop/rdma-hado
op-2.x-1.1.0/bin/slurm-pbs:/share/apps/compute/hadoop/rdma-hadoop-2.x-1.1.0/sbin:/share/apps/compute/hadoop/rdma-hadoop-2.x-1.1.0/bin:/opt/gnu/gcc/bin:/opt/gnu/bin:/o
pt/mvapich2/intel/ib/bin:/opt/intel/composer_xe_2013_sp1.2.144/bin/intel64:/opt/intel/composer_xe_2013_sp1.2.144/mpirt/bin/intel64:/opt/intel/composer_xe_2013_sp1.2.1
44/debugger/gdb/intel64_mic/bin:/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/opt/ibutils/bin:/usr/java/latest/bin:/opt/pdsh/bin
:/opt/rocks/bin:/opt/rocks/sbin:/opt/sdsc/bin:/opt/sdsc/sbin
export LD_LIBRARY_PATH=/opt/gnu/gcc/lib64:/opt/gnu/gmp/lib:/opt/gnu/mpfr/lib:/opt/gnu/mpc/lib:/opt/gnu/lib:/opt/gnu/lib64:/opt/mvapich2/intel/ib/lib:/opt/intel/compos
er_xe_2013_sp1.2.144/compiler/lib/intel64:/opt/intel/composer_xe_2013_sp1.2.144/mpirt/lib/intel64:/opt/intel/composer_xe_2013_sp1.2.144/ipp/../compiler/lib/intel64:/o
pt/intel/composer_xe_2013_sp1.2.144/ipp/lib/intel64:/opt/intel/composer_xe_2013_sp1.2.144/compiler/lib/intel64:/opt/intel/composer_xe_2013_sp1.2.144/mkl/lib/intel64:/
opt/intel/composer_xe_2013_sp1.2.144/tbb/lib/intel64/gcc4.4:/opt/sdsc/lib

### to prevent Spark from binding to the first address it can find
export SPARK_LOCAL_IP=$(sed -e '' <<< $HOSTNAME)
export LOCAL_DIRS=/scratch/etrain63/10071005
export SPARK_LOCAL_DIRS=/scratch/etrain63/10071005
export SPARK_WORKER_MEMORY=96g
export SPARK_WORKER_CORES=24
SPARK_DAEMON_MEMORY=2g
```

Performance Evaluation on SDSC Comet – SortBy/GroupBy



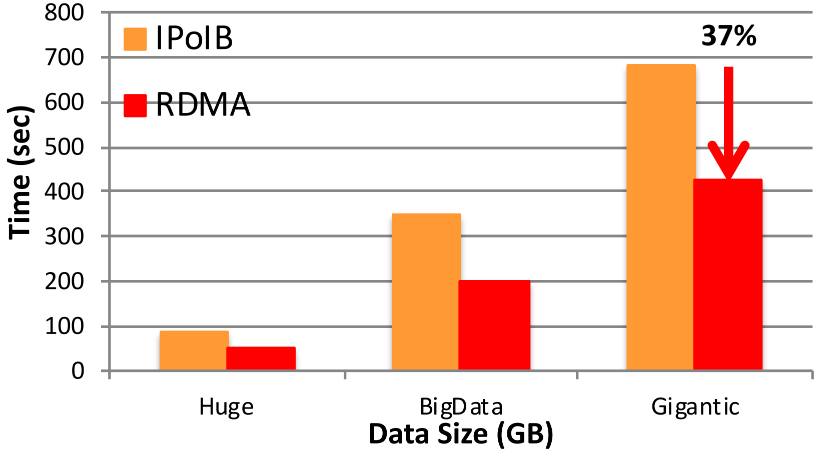
64 Worker Nodes, 1536 cores, SortByTest Total Time



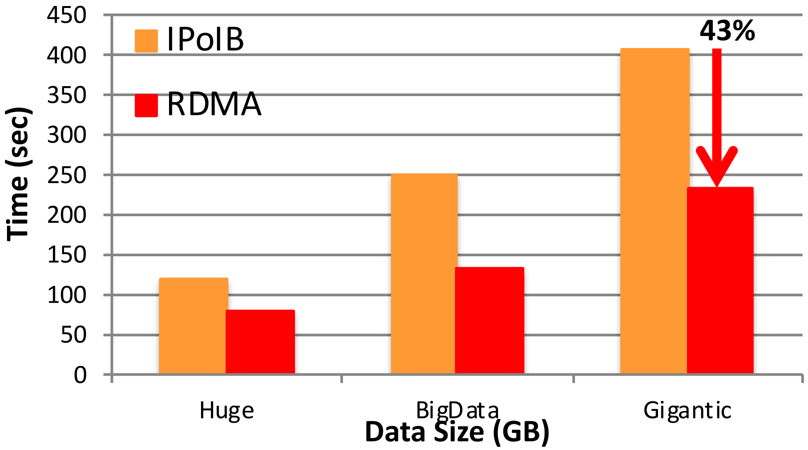
64 Worker Nodes, 1536 cores, GroupByTest Total Time

- **InfiniBand FDR, SSD, 64 Worker Nodes, 1536 Cores, (1536M 1536R)**
- **RDMA-based design for Spark 1.5.1**
- **RDMA vs. IPoIB with 1536 concurrent tasks, single SSD per node.**
  - SortBy: Total time reduced by up to 80% over IPoIB (56Gbps)
  - GroupBy: Total time reduced by up to 74% over IPoIB (56Gbps)

Performance Evaluation on SDSC Comet – HiBench PageRank



32 Worker Nodes, 768 cores, PageRank Total Time

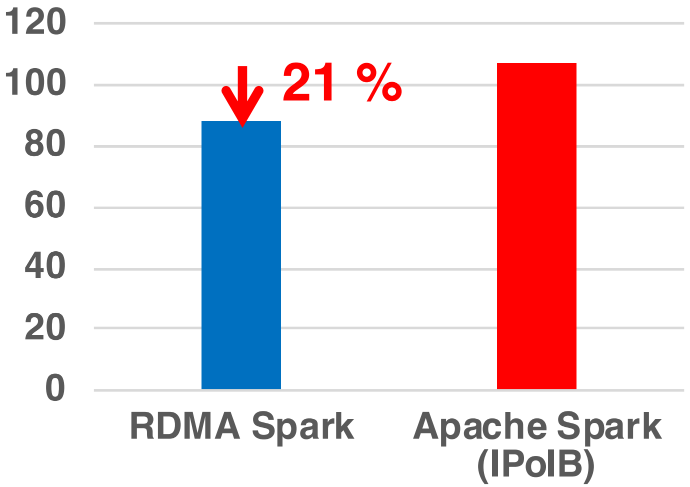


64 Worker Nodes, 1536 cores, PageRank Total Time

- **InfiniBand FDR, SSD, 32/64 Worker Nodes, 768/1536 Cores, (768/1536M 768/1536R)**
- **RDMA-based design for Spark 1.5.1**
- **RDMA vs. IPoIB with 768/1536 concurrent tasks, single SSD per node.**
  - 32 nodes/768 cores: Total time reduced by 37% over IPoIB (56Gbps)
  - 64 nodes/1536 cores: Total time reduced by 43% over IPoIB (56Gbps)

Performance Evaluation on SDSC Comet: Astronomy Application

- **Kira Toolkit[1]:** Distributed astronomy image processing toolkit implemented using Apache Spark.
- Source extractor application, using a 65GB dataset from the SDSS DR2 survey that comprises 11,150 image files.
- Compare RDMA Spark performance with the standard apache implementation using IPoIB.

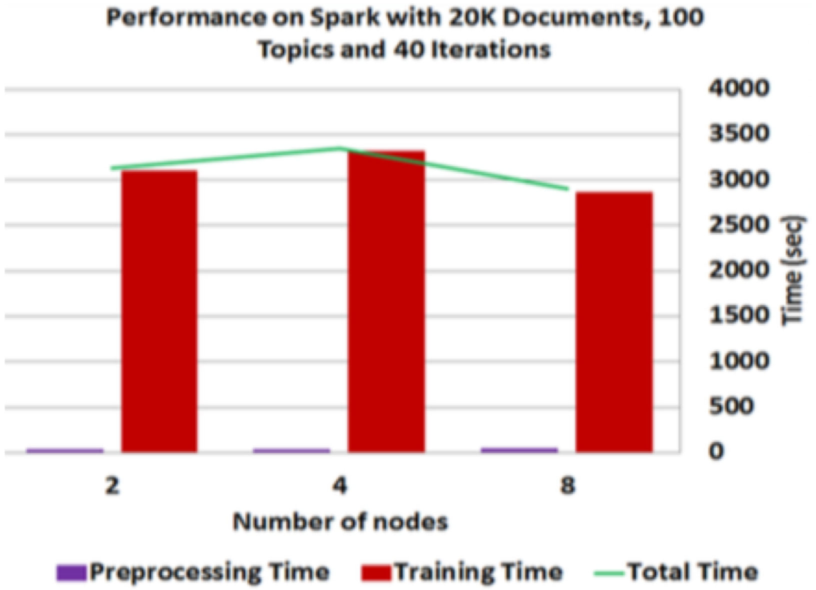


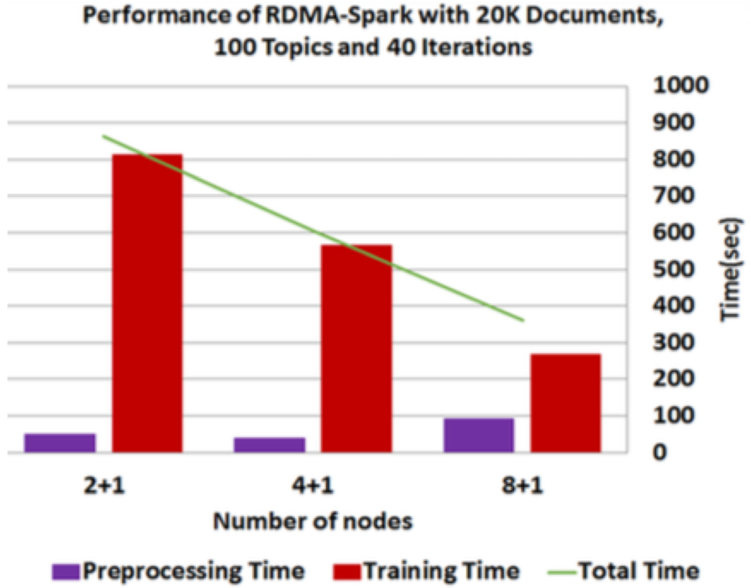
Execution times (sec) for Kira SE benchmark using 65 GB dataset, 48 cores.

[1] Z. Zhang, K. Barbary, F. A. Nothaft, E.R. Sparks, M.J. Franklin, D.A. Patterson, S. Perlmutter. Scientific Computing meets Big Data Technology: An Astronomy Use Case. *CoRR, vol: abs/1507.03325*, Aug 2015

M. Tatineni, X. Lu, D. J. Choi, A. Majumdar, and D. K. Panda, Experiences and Benefits of Running RDMA Hadoop and Spark on SDSC Comet, XSEDE'16, July 2016

RDMA Spark: Topic Modeling Application





***Reference: XSEDE16 Poster: Investigating Topic Models for Big Data Analysis in Social Science Domain Nitin Sukhija, Nicole Brown, Paul Rodriguez, Mahidhar Tatineni, and Mark Van Moer**

Current Work

- **Several Genomics tools have Spark integrated versions and work is in progress to enable them on Comet.**
  - HAIL - open-source, scalable framework for exploring and analyzing genetic data
  - GATK - variant discovery and genotyping toolkit
  - ADAM – uses Apache Spark to parallelize and scale several genomic data analysis tools including Avocado, Cannoli (BWA, Bowtie2, Freebayes), Gnocchi, Lime, and Mango (viz library).

- **RDMA-HBase** – already extensively tested by OSU HiBD group. Move to production availability on Comet soon.