RDMA Hadoop, Spark, and HBase middleware on the XSEDE Comet HPC resource.

Mahidhar Tatineni, SDSC
ECSS symposium
December 19, 2017

Collaborative project with Dr D.K. Panda’s
Network Based Computing lab at OSU
NSF Award #1447804 (OSU, Lead)
NSF Award #1447861 (SDSC)
HPC and Big Data Convergence

- Existing HPC users already have “big data” problems! Scales of simulations have been increasing with larger and higher resolution problems.
- Hardware like GPUs are optimal for segments of both HPC and Big Data workloads leading to HPC designs that can be effectively used for Big Data.
- HPC machines feature a rich set of storage and memory options - High Bandwidth Memory (HBM), SSDs, NVMe, and high performance parallel filesystems like Lustre, GPFS.

=> Opportunity for convergence of HPC and Big Data resources - both hardware and software.
RDMA Spark/Hadoop/HBase
HPC Hardware/Software+Big Data+Optimizations

• HPC architectures feature hardware designed for large scale processing with advanced processors, large and high performance parallel filesystems, advanced networks.
• RDMA technology allows processes to access remote memory locations with very low CPU load on the remote process.
• SSDs with high data reliability and performance available on several HPC machines
• Standard big data middleware and tools like Apache Hadoop, Spark, and HBase are currently not able to fully take advantage of these capabilities

⇒ Motivation to develop optimized versions of Apache Hadoop, Spark, and HBase to fully realize the benefits of the architecture.
• SDSC Comet is an excellent test-bed for such tools, with advanced processors, filesystems, SSDs, and high performance networks.
Add Data Analysis to Existing Compute Infrastructure
Add Data Analysis to Existing Compute Infrastructure

Resource Manager
(Torque, SLURM, SGE)
Add Data Analysis to Existing Compute Infrastructure
Add Data Analysis to Existing Compute Infrastructure
RDMA-Hadoop, Spark, and HBase Design

- 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.
RDMA-Hadoop, RDMA-Spark, and RDMA-HBase
Network-Based Computing Lab, Ohio State University
NSF funded project in collaboration with Dr. DK Panda

- 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 1.1.0 (based on Apache Hadoop 2.6.0)
- Version RDMA-Spark 0.9.4 (based on Apache Spark 2.1.0)
- Version RDMA-based Apache HBase 0.9.1 (RDMA-HBase) based on Apache HBase 1.1.2
- More details on the RDMA-Hadoop, RDMA-Spark, and RDMA-HBase projects at:
  - [http://hibd.cse.ohio-state.edu/](http://hibd.cse.ohio-state.edu/)
RDMA Hadoop – Scheduler Integration

• Directly integrated with Slurm/PBS
• Several running modes:
  • **HHH**: Heterogeneous storage devices with hybrid replication schemes for fault tolerance and performance.
  • **HHH-M**: A high-performance in-memory based setup.
  • **HHH-L**: Integrated with Lustre parallel filesystem.
  • **HHH-L-BB**: This mode deploys a Memcached-based burst buffer system (w/ local SSD).
  • **MapReduce over Lustre, with/without local disks**: Provides support to run MapReduce jobs on top of Lustre alone.
Sample Script: Anagram 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

...
...
...

module load rdma-hadoop/2x-1.1.0
Anagram using HHH-M mode

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

#Use SLURM integrated configuration/startup script
hibd_install_configure_start.sh -s -n ./hosts.hadoop.list -i $SLURM_JOBID -h $HADOOP_HOME -j $JAVA_HOME -m hhh-m -r /dev/shm -d /scratch/$USER/$SLURM_JOBID -t /scratch/$USER/$SLURM_JOBID/hadoop_local

#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
RDMA Hadoop: Benchmark Results

• HDFS I/O performance tested using TestDFSIO and RandomWriter.
• Hadoop performance was tested using Sort and TeraSort benchmarks.
• Performance of RDMA version compared with standard Apache version using the same hardware (Comet).
• Detailed results are available on the HiBD website (http://hibd.cse.ohio-state.edu/)
## RDMA Hadoop: Benchmark Results

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>Configuration</th>
<th>Max. Benefit over Apache Hadoop using IPoIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestDFSIO Throughput</td>
<td>RDMA Hadoop HHH-M, 16 Data Nodes</td>
<td>1.82x</td>
</tr>
<tr>
<td>RandomWriter</td>
<td>RDMA Hadoop HHH, 16 Data Nodes</td>
<td>1.32x</td>
</tr>
<tr>
<td>Sort</td>
<td>RDMA Hadoop HHH, 16 Data Nodes</td>
<td>1.48x</td>
</tr>
<tr>
<td>TeraSort</td>
<td>RDMA Hadoop HHH, 16 Data Nodes</td>
<td>1.23x</td>
</tr>
</tbody>
</table>
#!/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:$MODULEPATH
module load rdma-spark/0.9.4
module load rdma-hadoop/2x-1.1.0
export SPARK_CONF_DIR=$HOME/mysparkcluster
myspark-configure.sh

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

#Start Hadoop
hibd_install_configure_start.sh -s -n ./hosts.hadoop.list.$SLURM_JOBID -i $SLURM_JOBID -h $HADOOP_HOME -e -j $JAVA_HOME -d /scratch/$USER/$SLURM_JOBID -t /scratch/$USER/$SLURM_JOBID/hadoop_local
Spark submit script for GroupBy Benchmark (Continued)

```bash
### 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
```
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-x86_64:/home/xxx/rdma-hadoop-2.x-1.1.0/lib/native
  - spark.driver.extraLibraryPath /home/xxx/spark_2.1/lib/native/Linux-x86_64:/home/xxx/rdma-hadoop-2.x-1.1.0/lib/native
## RDMA Spark: Benchmark Results

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>Configuration</th>
<th>Max. Benefit over Apache Hadoop using IPoIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupBy</td>
<td>RDMA Spark 64 nodes, 1536 maps/reduces</td>
<td>1.57x</td>
</tr>
<tr>
<td>SortBy</td>
<td>RDMA Spark 64 nodes, 1536 maps/reduces</td>
<td>1.8x</td>
</tr>
<tr>
<td>HiBench TeraSort</td>
<td>RDMA Spark 64 nodes, 1536 cores</td>
<td>1.22x</td>
</tr>
<tr>
<td>HiBench PageRank</td>
<td>RDMA Spark 64 nodes, 1536 cores</td>
<td>1.46x</td>
</tr>
</tbody>
</table>
RDMA Spark, HDFS : SQL-Aggregation

RDMA-enhanced Heterogenous HDFS design HHH as the underlying filesystem that stores the Hive tables. Spark is run in Standalone mode. SSD is used for Spark Local and Work data. The RDMA-IB design with HHH improves the job execution time of SQL-Select by 14% - 22% compared to IPoIB (56Gbps).
RDMA-Spark: Applications

• **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.

---

RDMA-Spark: Applications

Execution times (s) for Kira SE benchmark using 65 GB dataset, 48 cores.
RDMA Spark: Applications

- Application in Social Sciences: Topic modeling using big data middleware and tools*.
- Latent Dirichlet Allocation (LDA) for unsupervised analysis of large document collections.
- Computational complexity increases as the volume of data increases.
- RDMA Spark enabled simulation of largest test cases.

*See 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
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*
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
RDMA-Spark: BigDL Test Case

- BigDL is a distributed deep learning library for Apache Spark.
- This test case uses BigDL to train and test a Visual Geometry Group (VGG)-like network on CIFAR-10 data.

![Graph showing One Epoch Time (sec) vs. Number of cores for IPoIB and RDMA]
RDMA-Hbase: Yahoo Cloud Serving Benchmark (YCSB)

These experiments are run with 4 Regionservers and up to 16 client nodes, each running 1-8 HBase client instances. Each client uses 1 thread for communication and performs 25000 operations on the HBase cluster. RDMA-IB design improves the throughput for Workload A, B, and C by up to 145.51%, 164.82%, and 259.41% over IPoIB (56Gbps), respectively.

Ref: http://hibd.cse.ohio-state.edu/performance/ycsb/
Summary

- Convergence of large scale high performance computing (HPC) resources and Big Data architectures can lead to performance improvements.

- RDMA-Hadoop, Spark, HBase from HiBD group at OSU leverages existing HPC hardware, keeps compliance with standard distributions, and provides significant performance benefits.

- Easy integration with Slurm scheduler on Comet - either direct scripts (like RDMA-Hadoop) or via mySpark.

- Application testing with RDMA enabled frameworks show good performance on Comet.