Integrating Hadoop and Spark on USGS HPC Cluster with Magpie

RMACC

August 16, 2017

Natalya Rapstine*, Brandon Serna and Janice Gordon
U.S. Geological Survey

Roadmap

- Motivation
- Challenge
- Big data tools
- HPC infrastructure
- Magpie software
- Integrating big data tools with HPC system
- Examples

USGS

Motivation

Big Data Cluster

Big Data Cluster

Cloud = $$

Big Data Cluster

~~Cloud = $$~~

Existing HPC infrastructure:
big data cluster on demand

Challenge

Integrate

HPC systems
USGS Yeti Supercomputer

Big Data Tools
Apache Hadoop
Apache Spark

USGS

Challenge

Job/Resource management?

- **shared-disk** (storage is shared amount multiple processors)

- **shared-nothing** architecture (memory and storage are not shared)

Integrate?

HPC systems
USGS Yeti Supercomputer

Big Data Tools
Apache Hadoop
Apache Spark

USGS

Magpie

HPC systems
USGS Yeti Supercomputer

Big Data Tools
Apache Hadoop
Apache Spark

USGS

Integration Tools for Big Data on HPC Systems

Magpie	https://github.com/LLNL/magpie
myHadoop	https://github.com/glennklockwood/myhadoop/
Hadoop on Demand (HOD)	https://svn.apache.org/repos/asf/hadoop/common/tags/release-0.17.1/docs/hod.html
High-Performance Big Data (HiBD)	http://hibd.cse.ohio-state.edu/

USGS

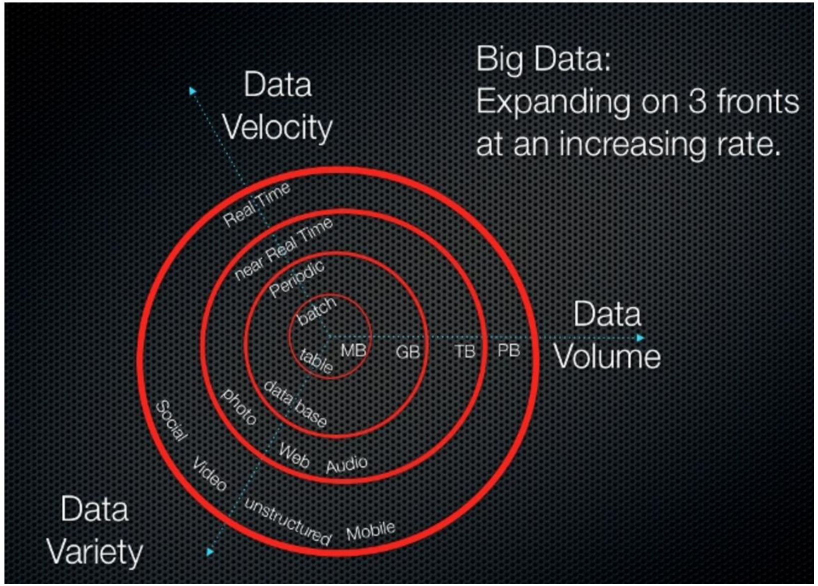
Big Data

Ha Ha

USGS

Big Data Properties

- Volume:
  - Amount of data: Terabytes to Petabytes
- Variety:
  - Types of data: structured (relational databases) and unstructured data (machine data, social network data, text, images, videos, PDF's)
- Velocity:
  - Speed of data processing: months to real time



http://www.datasciencecentral.com/forum/topics/the-3vs-that-define-big-data

USGS

Hadoop

Hadoop Architecture

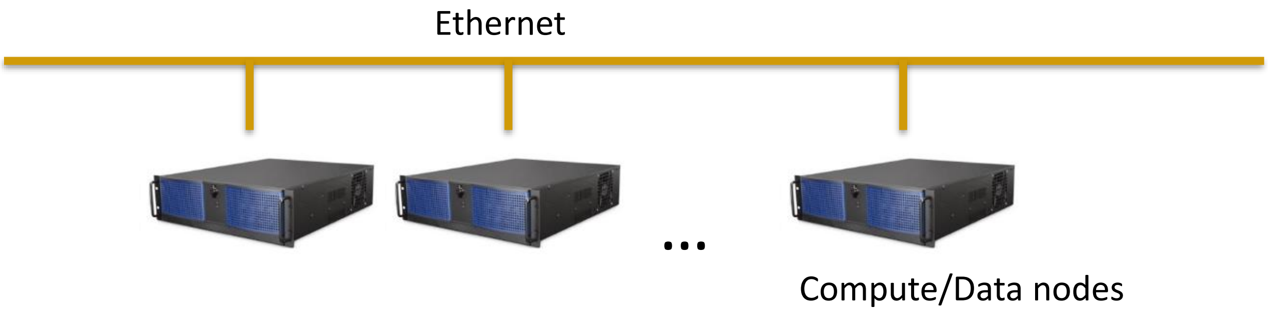
- Distributed computing for big data:
  - Hadoop Distributed File System (HDFS)
  - MapReduce programming model

USGS

HDFS

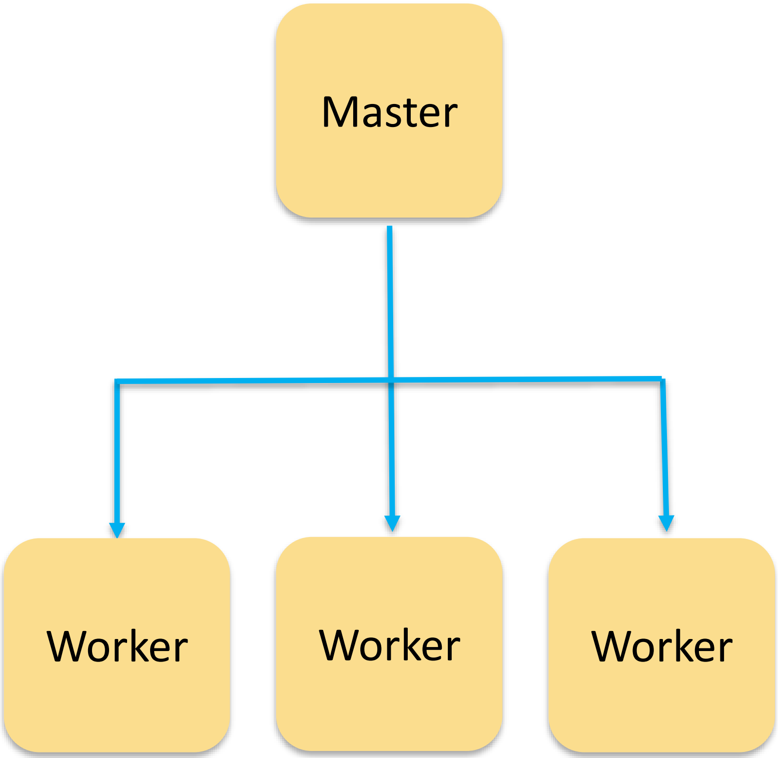
Hadoop Distributed File System:

– Compute and Data nodes are co-located
– Memory and storage are not shared among nodes



USGS

Hadoop Cluster

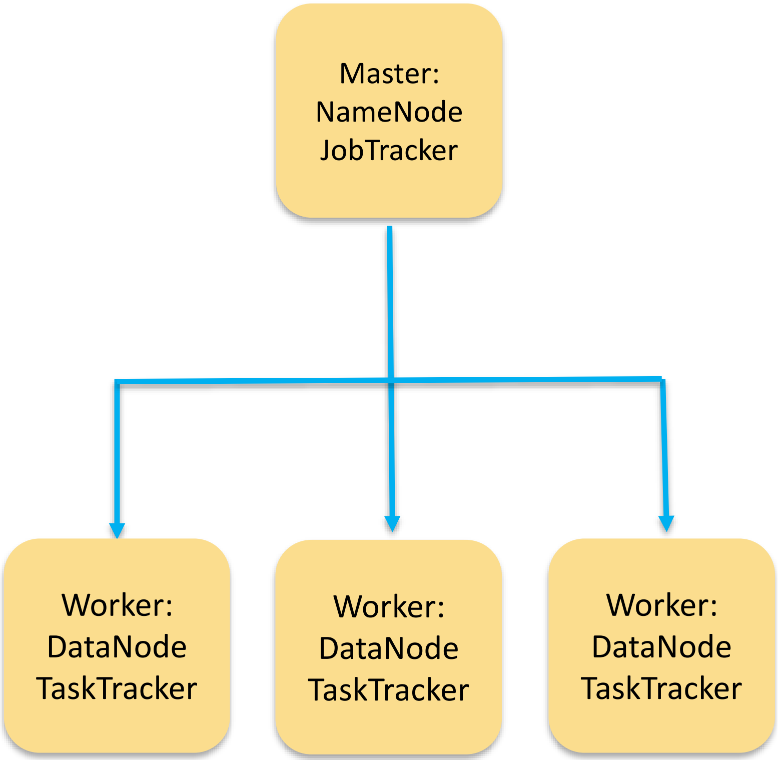


Master:
- Coordinate data storage in HDFS
- Coordinate parallel processing

Workers:
- Store data in HDFS
- Run parallel computations on that data using MapReduce

USGS

Hadoop Cluster



NameNode daemon:

- Coordinate data storage in HDFS

JobTracker daemon:

- Coordinate parallel processing

DataNode daemon:

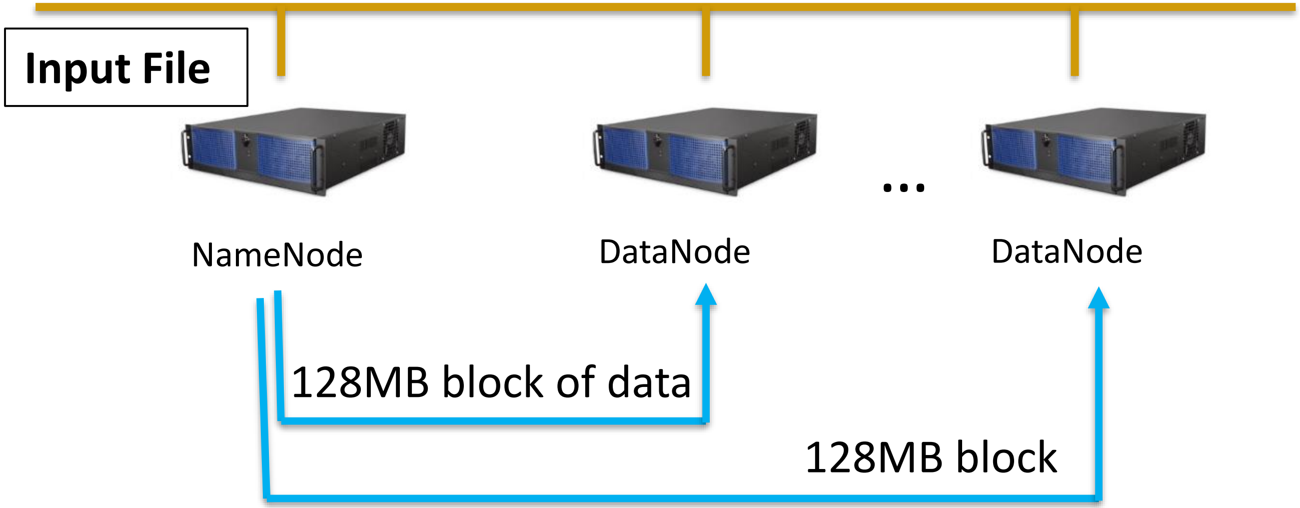
- Store data in HDFS

TaskTracker daemon:

- Run parallel computations on that data using MapReduce

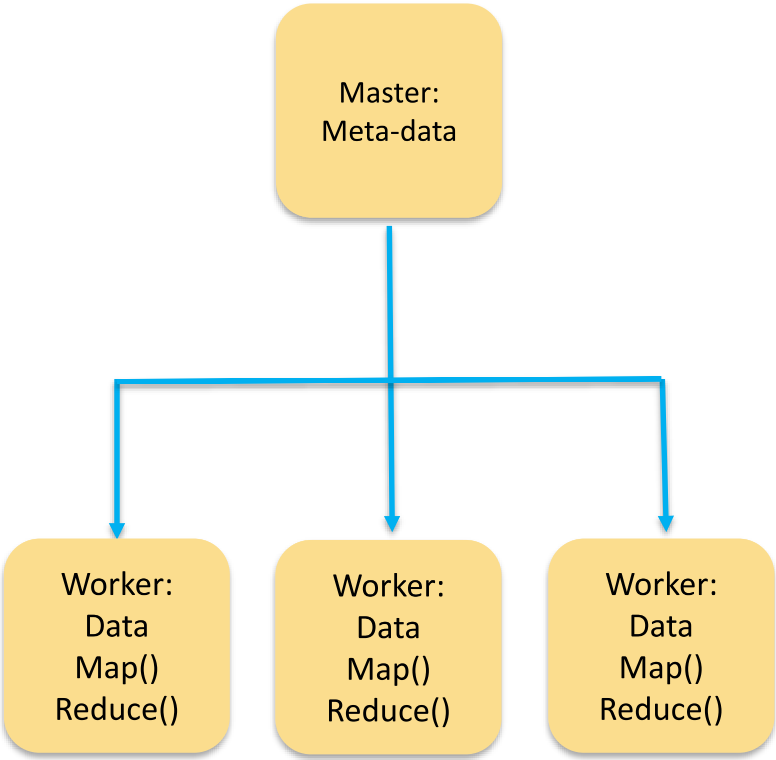
USGS

HDFS


Input File
NameNode
DataNode
...
DataNode
128MB block of data
128MB block

USGS

Hadoop Cluster



Master:

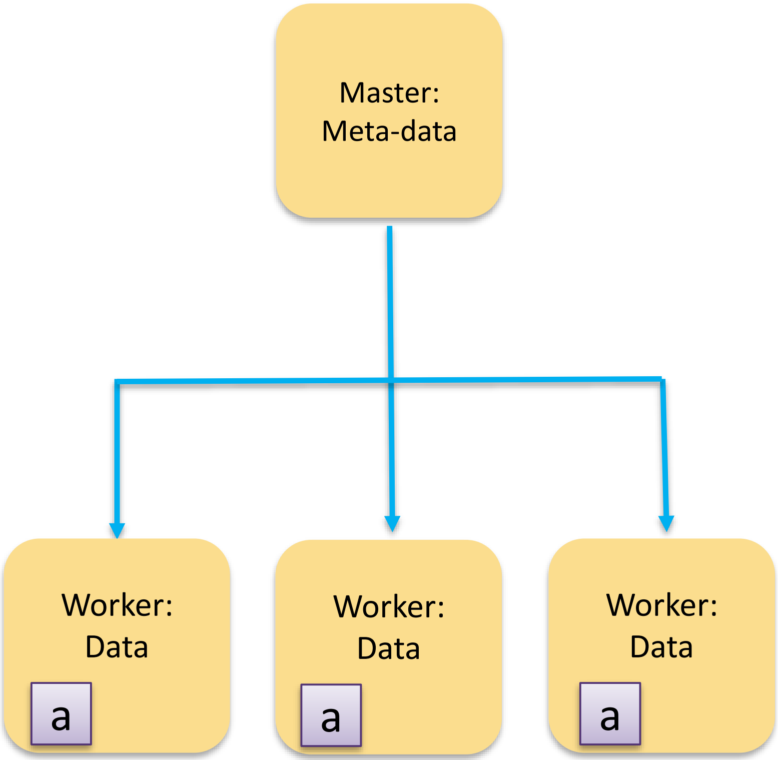
- Meta-data about distributed file blocks and location in the cluster

Workers:

- 128 MB blocks of data
- Map() and Reduce() functions on the data

USGS

Hadoop Cluster



Master:

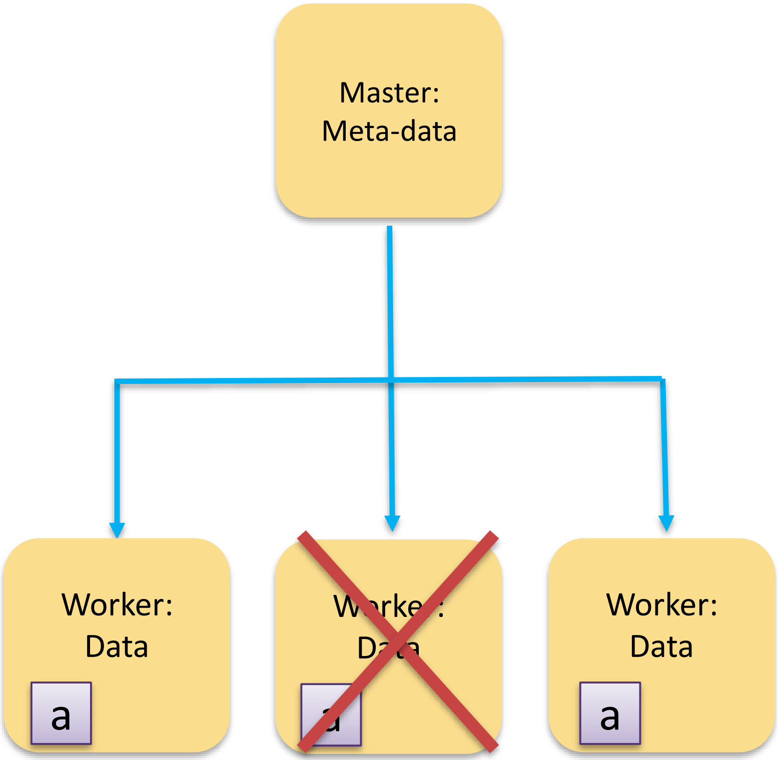
- Meta-data about distributed file blocks and location in the cluster

Data reliability:

- Data is replicated on various nodes
- Default replication factor = 3

USGS

Hadoop Cluster



JobTracker:

- Reschedule the task on a different worker node

Data reliability:

- Data is replicated on various nodes
- Default replication factor = 3

Word Count Example with MapReduce

256 MB

A long time ago in a galaxy far, far away...

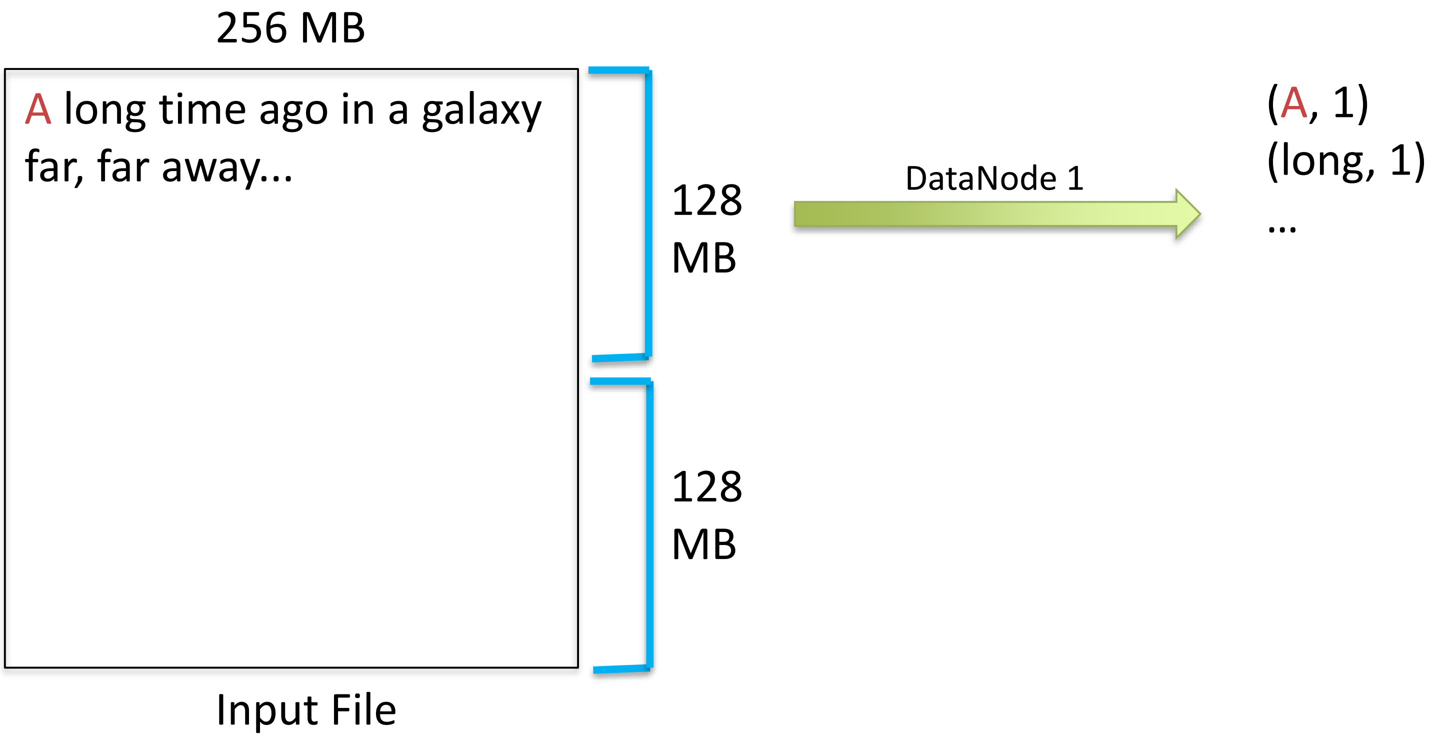
Input File

128 MB

DataNode 1

128 MB

Word Count Example with MapReduce



Hadoop MapReduce

256 MB

A long time ago in a galaxy far, far away...

Input File

128 MB

128 MB

DataNode 1

(A, 1)
(long, 1)
...

Hadoop MapReduce

256 MB

A long time ago in a galaxy far, far away…

Input File

128 MB

128 MB

DataNode 1

(A, 1)
(long, 1)
…
(far, 1)

Hadoop MapReduce

256 MB

A long time ago in a galaxy far, far away…

128 MB

128 MB

Input File

DataNode 1

(A, 1)
(long, 1)
…
(far, 2)

Hadoop MapReduce

256 MB

A long time ago in a galaxy far, far away…

Input File

128 MB

128 MB

DataNode 1

(A, 1)
(long, 1)
…
(far, 2)

Slow!

Hadoop MapReduce

256 MB

A long time ago in a galaxy far, far away…

128 MB

128 MB

Input File

DataNode 1

(A, 1)
(long, 1)
…
(far, 1)
(far, 1)

Faster

Hadoop MapReduce

256 MB

A long time ago in a galaxy far, far away....

...ashes of the Empire and will not rest until Skywalker, the last Jedi, has been destroyed.

Input File

128 MB

DataNode 1

(A, 1)
(long, 1)
...
(far, 1)
(far, 1)

128 MB

DataNode 2

(ashes, 1)
(of, 1)
...
(destroyed, 1)

Map

DataNode 1

(A, 1)
(long, 1)
…
(far, 1)
(far, 1)

DataNode 2

(ashes, 1)
(of, 1)
…
(destroyed, 1)

USGS

Map

DataNode 1

(A, 1)
(long, 1)
…
(far, 1)
(far, 1)

DataNode 2

(ashes, 1)
(of, 1)
…
(destroyed, 1)

Combine

(A, 1)
(ashes, 1)
…
(far, 1)
(far, 1)
…

Map

DataNode 1

(A, 1)
(long, 1)
…
(far, 1)
(far, 1)

DataNode 2

(ashes, 1)
(of, 1)
…
(destroyed, 1)

Combine

(A, 1)
(ashes, 1)
…
(far, 1)
(far, 1)
…

(far, 1,1)
…

Map

DataNode 1

(A, 1)
(long, 1)
…
(far, 1)
(far, 1)

DataNode 2

(ashes, 1)
(of, 1)
…
(destroyed, 1)

Combine

Sort & Shuffle

(A, 1)
(ashes, 1)
…
(far, 1)
(far, 1)
…

(far, 1,1)
…

Map

DataNode 1

(A, 1)
(long, 1)
…
(far, 1)
(far, 1)

DataNode 2

(ashes, 1)
(of, 1)
…
(destroyed, 1)

Combine

Sort & Shuffle

(A, 1)
(ashes, 1)
…
(far, 1)
(far, 1)
…

(far, 1,1)
…

Reduce

Aggregate

(far, 2)
…

A long time ago in a galaxy far, far away.... Luke Skywalker has vanished. In his absence, the sinister FIRST ORDER has risen from the ashes of the Empire and will not rest until Skywalker, the last Jedi, has been destroyed.

256 MB

B_1 128 MB

B_2 128 MB

USGS

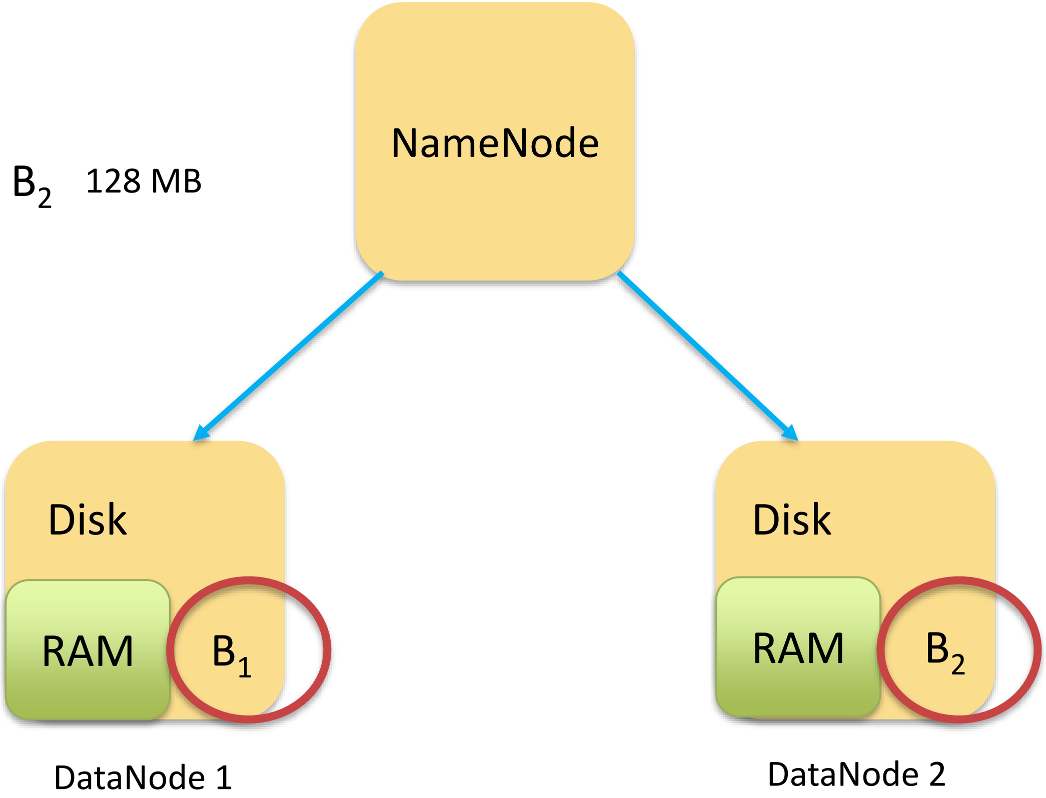
In HDFS, data resides on disk in a distributed fashion on worker nodes

A long time ago in a galaxy far, far away.... Luke Skywalker has vanished. In his absence, the sinister FIRST ORDER has risen from the ashes of the Empire and will not rest until Skywalker, the last Jedi, has been destroyed.

B_1 128 MB

B_2 128 MB

256 MB



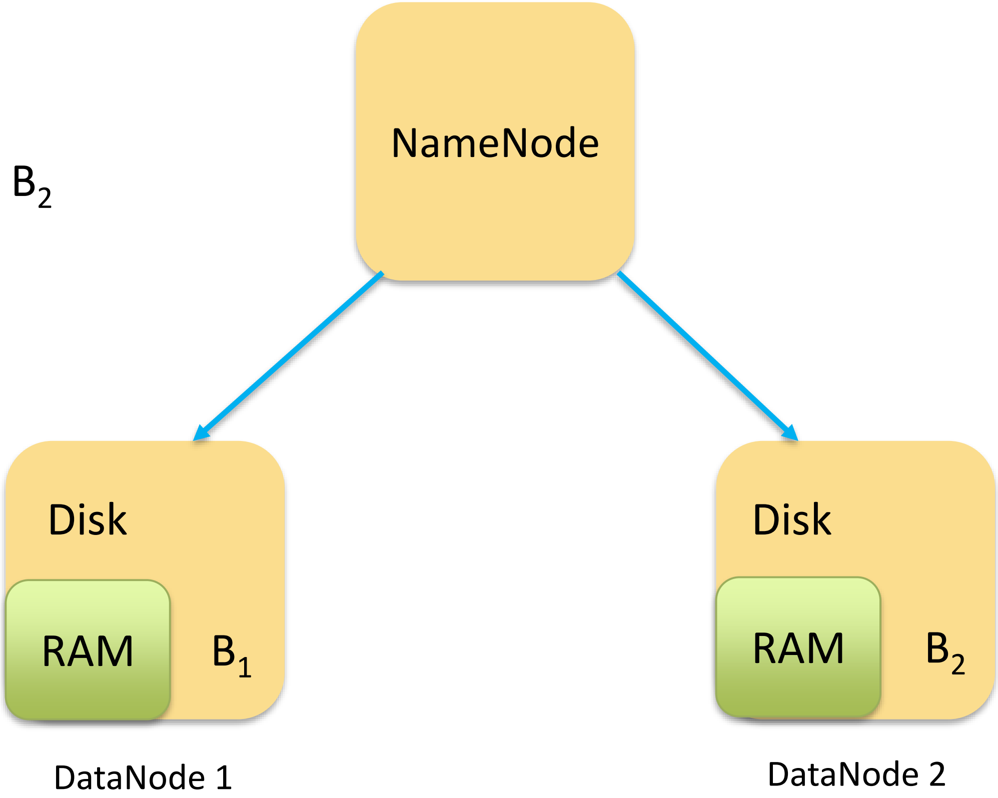
Data resides on disk on each DataNode

Processing is in memory

A long time ago in a galaxy far, far away.... Luke Skywalker has vanished. In his absence, the sinister FIRST ORDER has risen from the ashes of the Empire and will not rest until Skywalker, the last Jedi, has been destroyed.

B_1

B_2



USGS

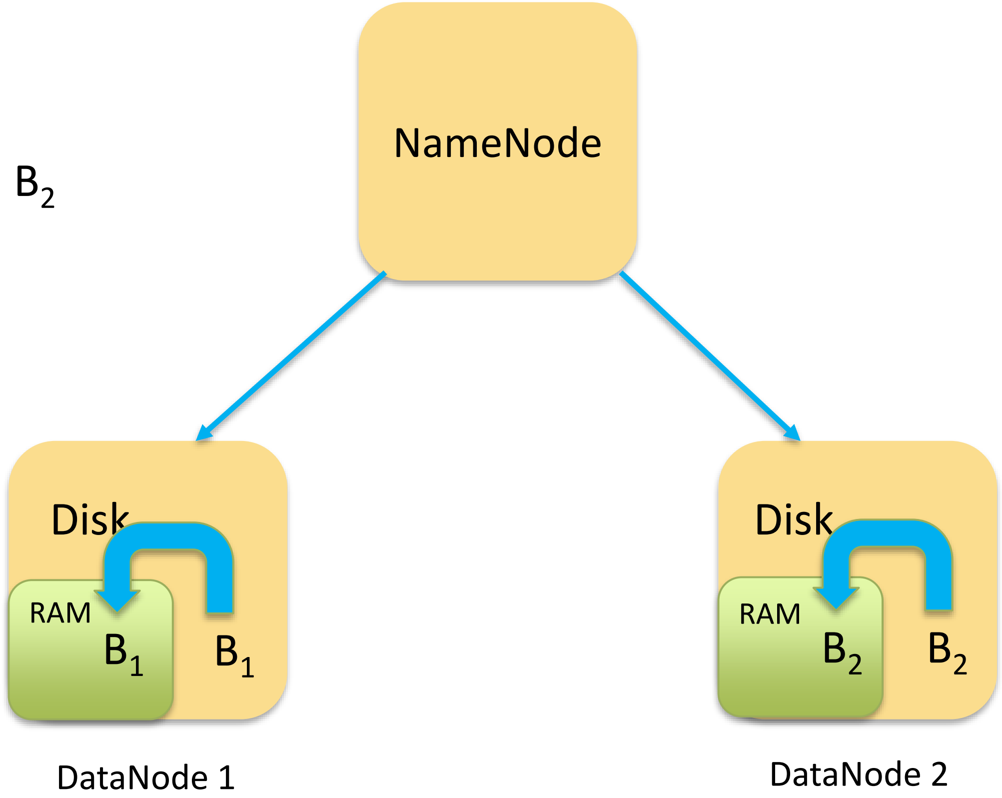
A long time ago in a galaxy far, far away.... Luke Skywalker has vanished. In his absence, the sinister FIRST ORDER has risen from the ashes of the Empire and will not rest until Skywalker, the last Jedi, has been destroyed.

B_1

B_2

Map disk I/O operations:

- Read from disk
- Data is copied to RAM



USGS

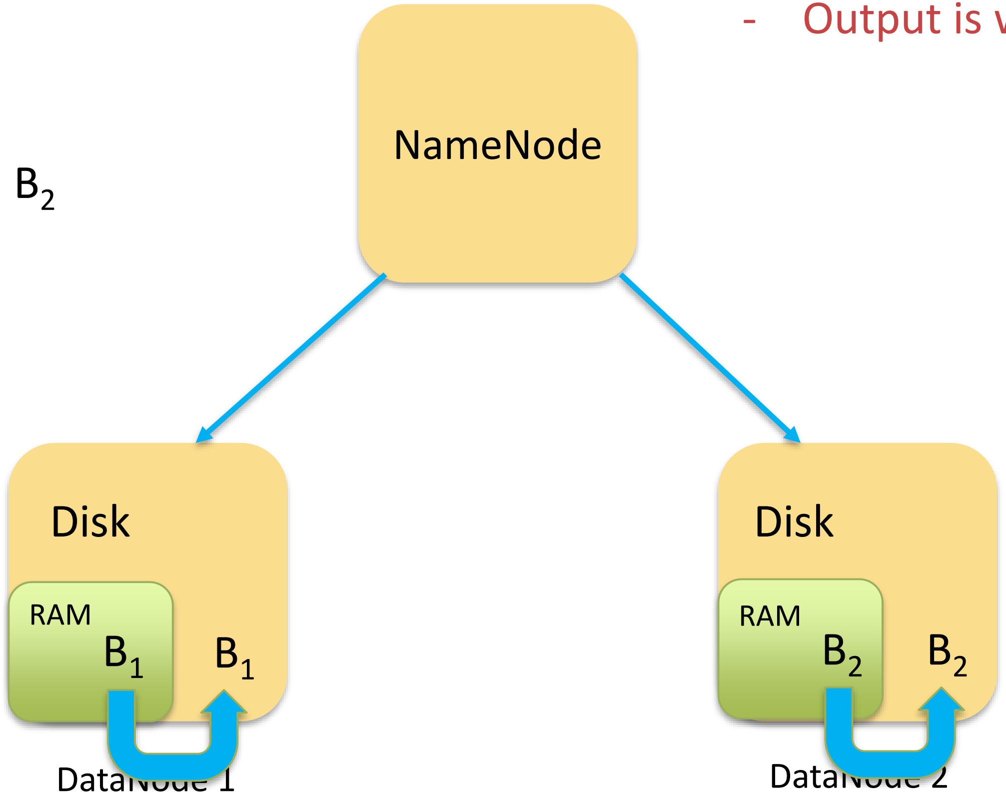
A long time ago in a galaxy far, far away.... Luke Skywalker has vanished. In his absence, the sinister FIRST ORDER has risen from the ashes of the Empire and will not rest until Skywalker, the last Jedi, has been destroyed.

B_1

B_2

Map disk I/O operations:

- Read from disk
- Data is copied to RAM
- Output is written back to disk



Map

Sort & Shuffle

DataNode 1

(A, 1)
(long, 1)
…
(far, 1)
(far, 1)

DataNode 2

(ashes, 1)
(of, 1)
…
(destroyed, 1)

Combine

(A, 1)
(ashes, 1)
…
(far, 1)
(far, 1)
…

Data transfer over network

Map | Sort & Shuffle | Reduce

DataNode 1

(A, 1)
(long, 1)
…
(far, 1)
(far, 1)

DataNode 2

(ashes, 1)
(of, 1)
…
(destroyed, 1)

Combine

Data transfer over network

(A, 1)
(ashes, 1)
…
(far, 1)
(far, 1)
…

(far, 1,1)
…

Aggregate

(far, 2)
…

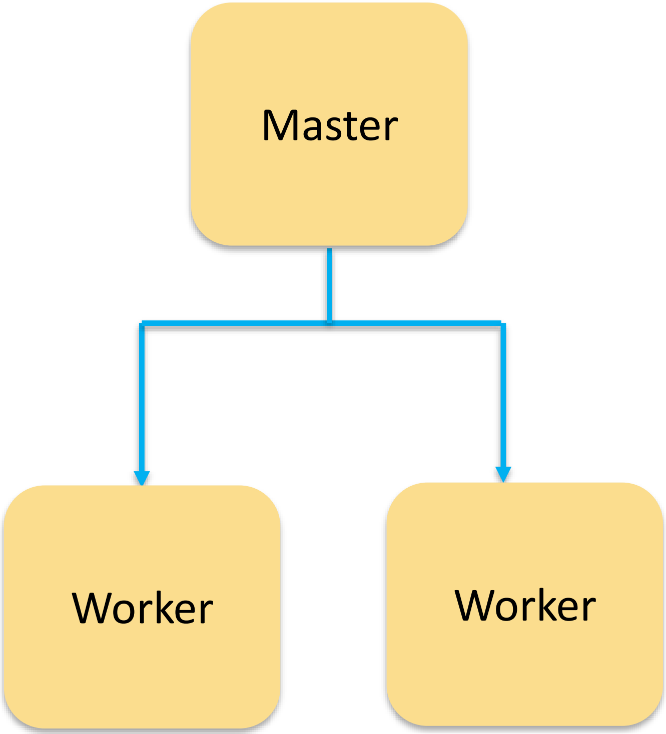
Disk I/O operations in Hadoop MapReduce programs are a major performance bottleneck

USGS

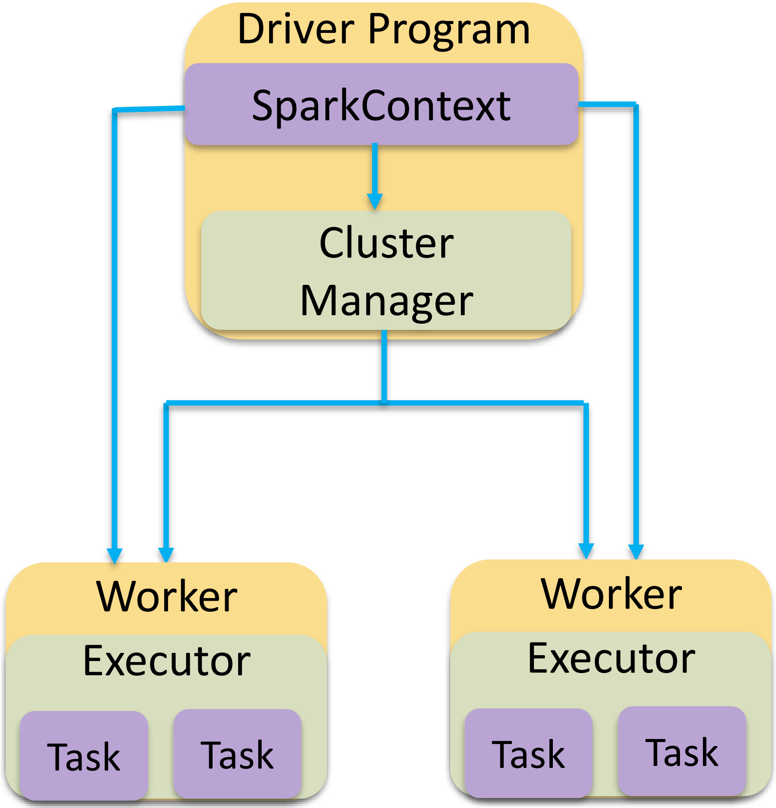
Spark

Spark Architecture

- Master/worker model
- Resilient Distributed Datasets (RDD)
  - Solves MapReduce performance bottleneck
  - Spark programs are faster than Hadoop MapReduce programs
- PySpark = Python API for Spark



Spark Cluster



Master:

Spark Driver Program:

- hosts **SparkContext** that coordinates workers and schedules tasks on executors
- SparkContext connects to the **Cluster Manger** which allocates resources

Workers:

- Executors store data and run computations

USGS

Initializing Spark

- In Python:

```
from pyspark import SparkContext, SparkConf
conf = SparkConf().setAppName(appName).setMaster(master)
sc = SparkContext(conf=conf)
```

https://spark.apache.org/docs/2.0.2/programming-guide.html

Initializing Spark

- In Python:

```
from pyspark import SparkContext, SparkConf
conf = SparkConf().setAppName(appName).setMaster(master)
sc = SparkContext(conf=conf)
```

SparkConf() - Contains information about your application

Initializing Spark

- In Python:

```
from pyspark import SparkContext, SparkConf
conf = SparkConf().setAppName(appName).setMaster(master)
sc = SparkContext(conf=conf)
```

SparkContext()

- Main entry for Spark functionality
- Every Spark program has one of these objects
- Connection to a Spark cluster
- Used to create RDDs, broadcast variables, etc.

https://spark.apache.org/docs/2.0.2/api/java/org/apache/spark/SparkContext.html

Spark RDD

- RDD’s:
  - Resilient: data is replicated across nodes (by default, 3 times)
  - Immutable: new RDD is created after each transformation
  - All computations are done in memory
  - Minimize disk I/O (unlike Hadoop MapReduce)

Spark RDD

- Resilient Distributed Datasets (RDD):
  - Collection of elements to run parallel operations on
- To create RDD:
  1. `SparkContext`'s `parallelize()` method on existing iterable or collection in your program

```
data = [1, 2, 3, 4, 5]
distData = sc.parallelize(data)
```

USGS

Spark RDD

- Operate on the distributed dataset in parallel
- For example, to add up elements of the list:

```
data = [1, 2, 3, 4, 5]
distData = sc.parallelize(data)
dataSum = distData.reduce(lambda a, b: a+b)
```

- Can use anonymous functions in Python on RDD
- For example, `reduce()` function returns a single value by combining elements via a supplied function
- RDD’s are immutable

USGS

Spark RDD

```
data = [1, 2, 3, 4, 5]
distData = sc.parallelize(data)
dataSum = distData.reduce(lambda a, b: a+b)
```

- When `reduce()` is run, Spark breaks the computation into tasks to run across multiple nodes and each node returns the answer to the driver program on the master node

Spark RDD

```
data = [1, 2, 3, 4, 5]
distData = sc.parallelize(data, 2)
dataSum = distData.reduce(lambda a, b: a+b)
```

- Optionally, pass the number of partitions as a second argument to `parallelize()`
- Spark will run one task per partition

Spark RDD

2. Load data from HDFS, Lustre or local file system

```
distFile = sc.textFile(“data.txt”)
```

`textFile()`

- Method to read a text file from HDFS
- Returns an RDD of strings
- Loads in memory

USGS

Spark RDD

2. Load data from HDFS, Lustre or local file system

```
distFile = sc.textFile("data.txt")
```

- Operate on the distributed dataset in parallel
- For example, to add up the lengths of all the lines in the text file:

```
sumRDD = distFile.map(lambda s: len(s)).reduce(lambda a, b: a + b)
```

- `map()` function applies an operation to each item in a list
- `reduce()` function returns a single value by combining elements via a supplied function

USGS

Spark RDD Example

```
number = sc.textFile("Numbers.txt")
filter1 = number.filter(lambda x: x < 10)
```

384 MB

1, 3, 44, 78, …
….,

23, 36, 5, 2, 1, …
…,

3, 7, 55, ….
…

Numbers.txt

USGS

Spark RDD Example

```
number = sc.textFile("Numbers.txt")
filter1 = number.filter(lambda x: x < 10)
```

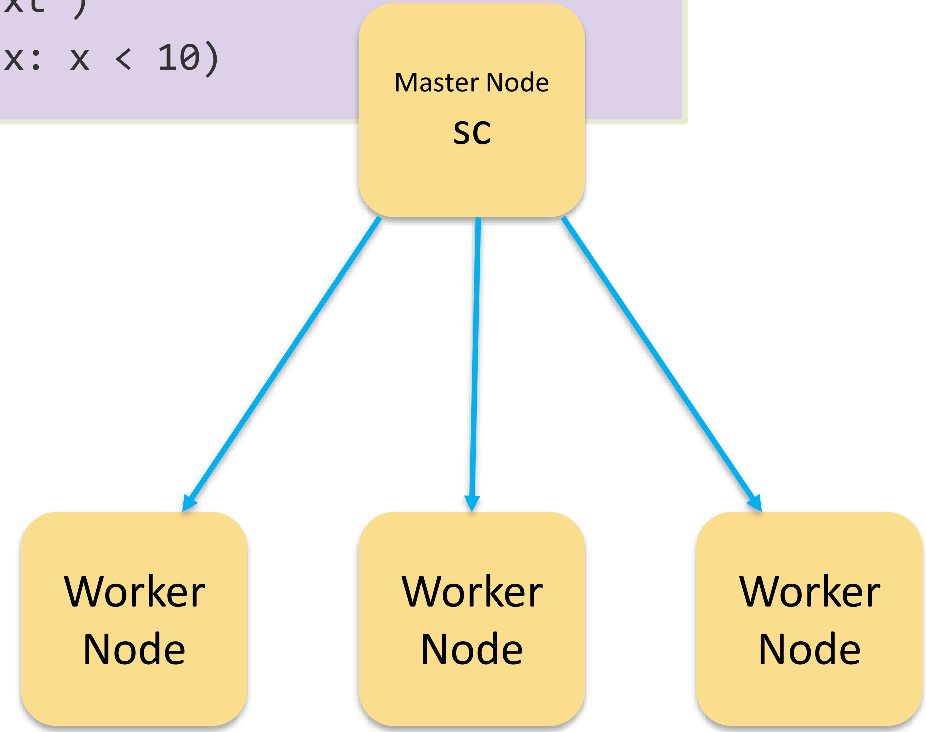
384 MB

1, 3, 44, 78, …
….,

23, 36, 5, 2, 1, …
…,

3, 7, 55, ….
…

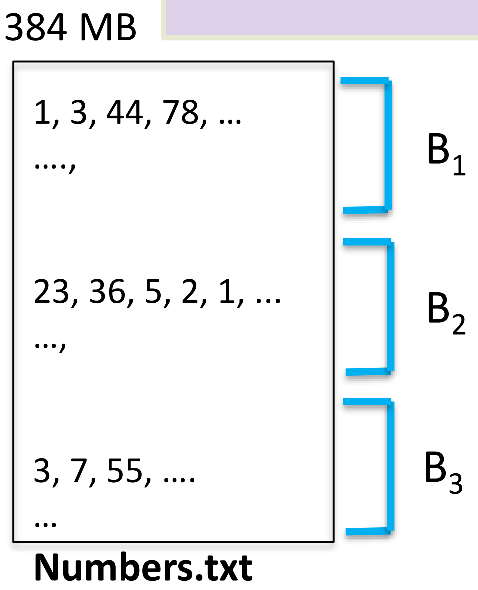
Numbers.txt

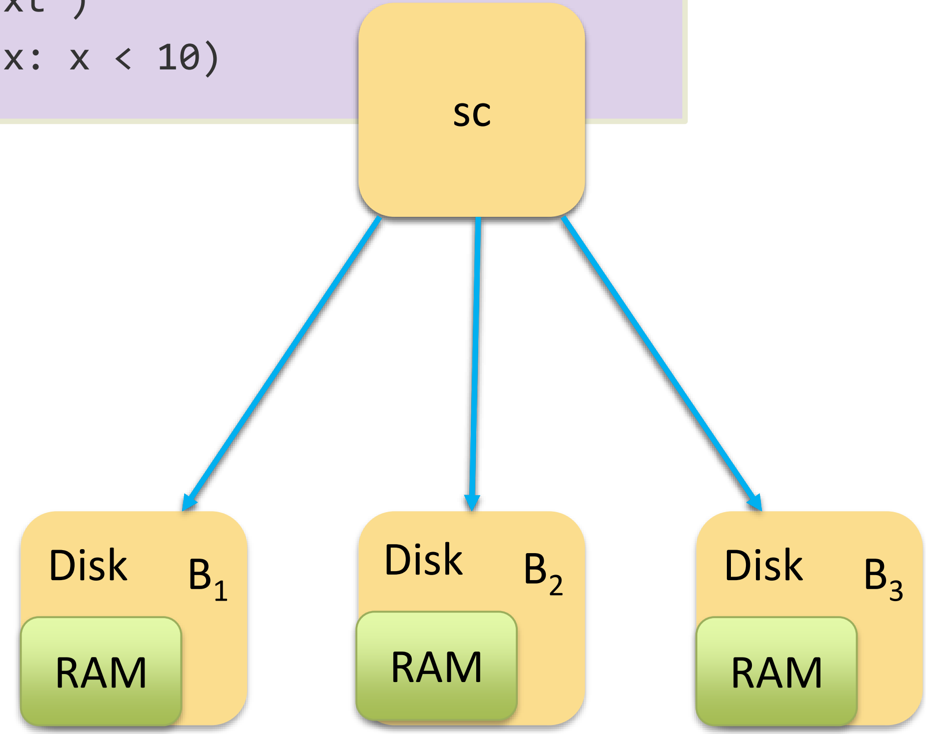


USGS

Spark RDD Example

```
number = sc.textFile("Numbers.txt")
filter1 = number.filter(lambda x: x < 10)
```

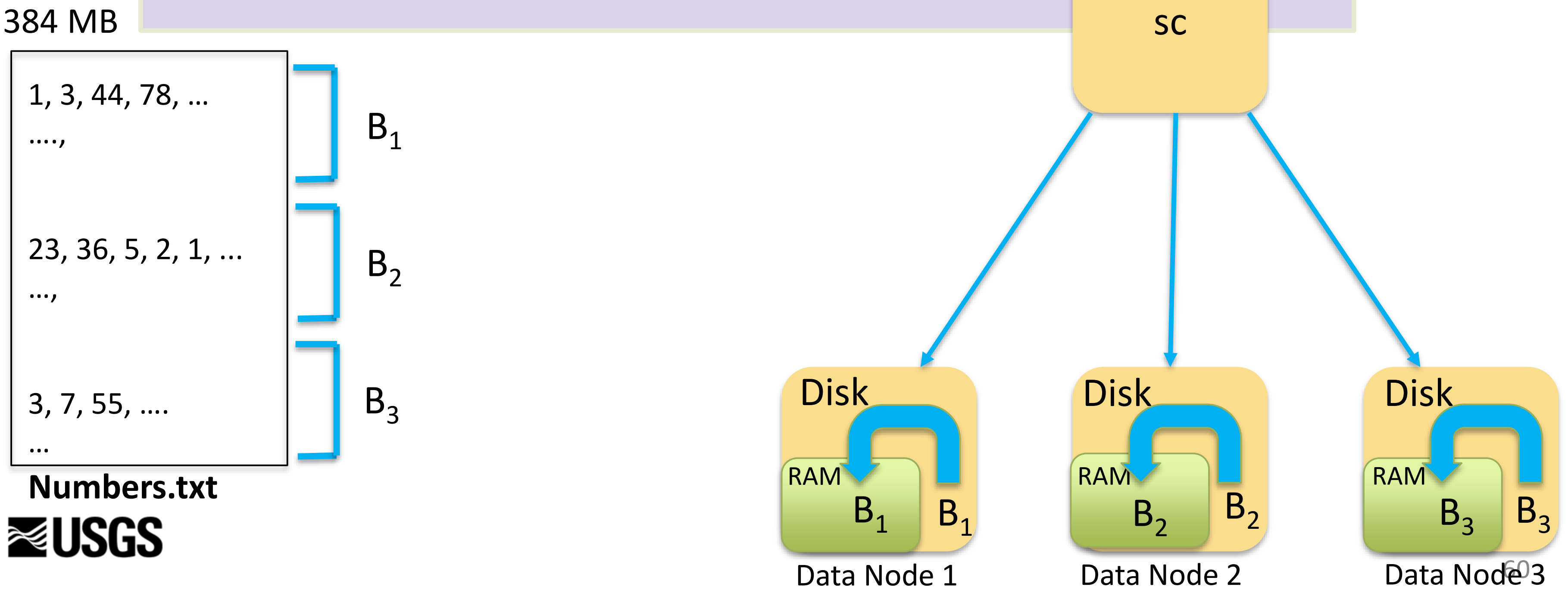




USGS

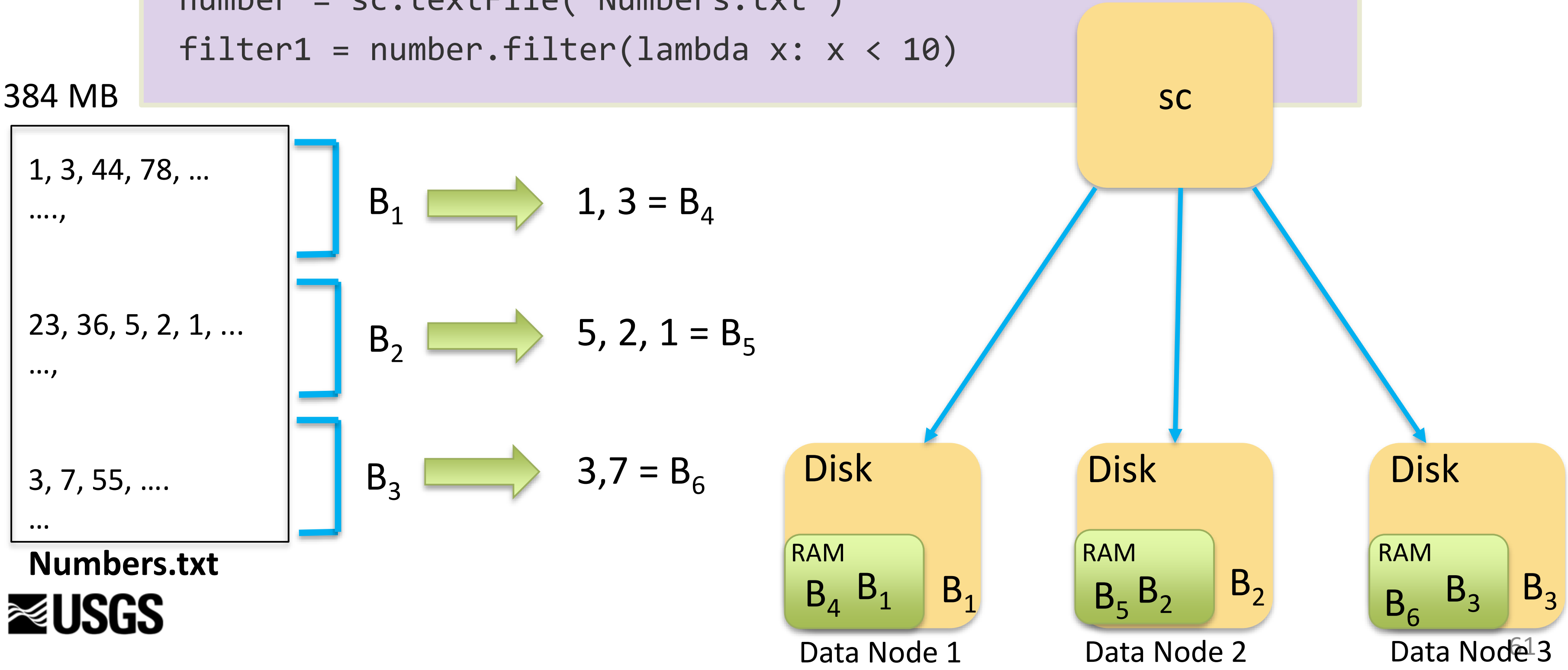
Spark RDD Example

```
number = sc.textFile("Numbers.txt")
filter1 = number.filter(lambda x: x < 10)
```



Spark RDD Example

```
number = sc.textFile("Numbers.txt")
filter1 = number.filter(lambda x: x < 10)
```



USGS

Spark RDD

- RDD’s:
  - Resilient: data is replicated across nodes (by default, 3 times)
  - Immutable: new RDD is created after each transformation
  - All computations are done in memory
  - Minimize disk I/O (unlike Hadoop MapReduce)

USGS

USGS Supercomputer

USGS

Yeti Supercomputer

- Traditional HPC cluster:

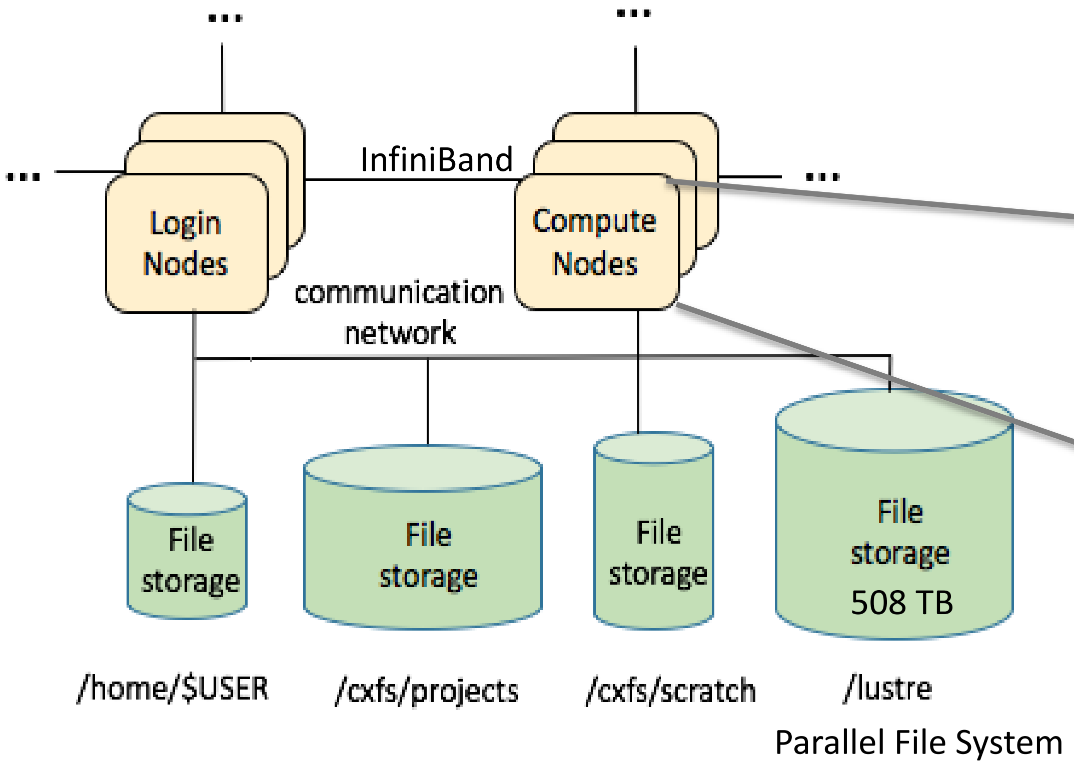
High performance, high cost computers

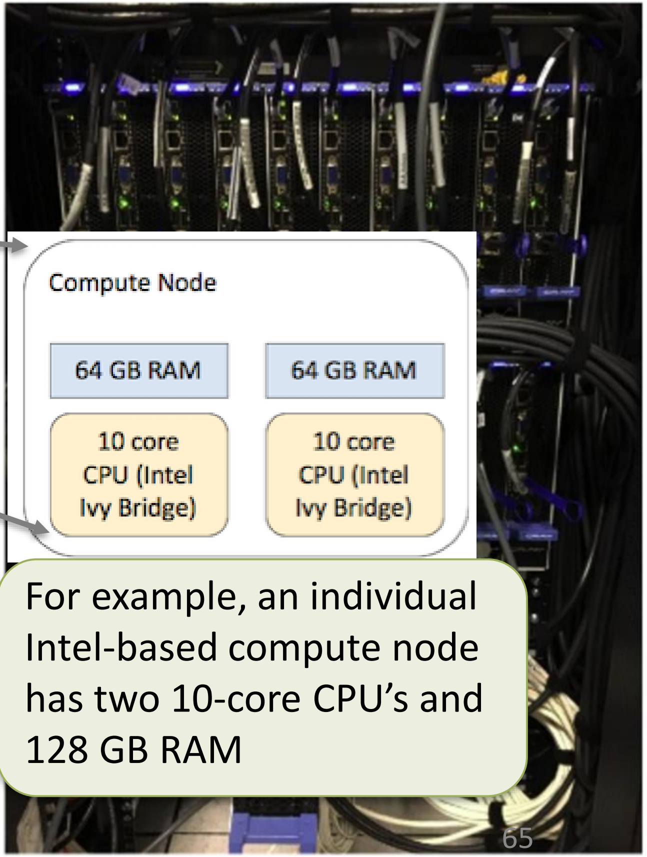
- 3,728 CPU cores
- 14 GPU’s and 6 Xeon Phi’s

147 compute nodes are organized into shared and distributed memory partitions (normal, UV, large, long)

USGS

HPC System





For example, an individual Intel-based compute node has two 10-core CPU's and 128 GB RAM

USGS

Resources are shared by many users

SLURM

- HPC resources (cores and memory) are shared among many users
- SLURM schedules jobs and manages resources
  - Batch job submission
  - Maximizes cluster utilization
  - MPI would typically be used to manage communications within the parallel program

USGS

SLURM Batch Script

```
> sbatch –A training –p normal –t 01:00:00 –N 2 –n 10
do_work.slurm
Submitted batch job 2724129
```

Or include the same flags in your batch script:

```
#SBATCH –A training
#SBATCH –p normal
#SBATCH –t 01:00:00
#SBATCH –N 2
#SBATCH –n 10
```

–A Account code

–p Partition (normal, UV, large, long)

–t Wall time (d–hh:mm:ss)

–N Number of nodes

–n Number of tasks

USGS

Magpie

Magpie

- Open-source software from Lawrence Livermore National Laboratory: https://github.com/LLNL/magpie
- **Purpose**: run Big Data software in traditional HPC environments
- **Support**:
  - **SLURM**, Moab, Torque and LSF
  - Lustre parallel file system
  - **Hadoop, Spark**, Hbase, Storm, Pig, Mahout, Phoenix, Kafka, Tachyon, Zeppelin, and Zookeeper

Magpie Workflow

1. Submit a Magpie batch script to allocate nodes on Yeti cluster using SLURM scheduler/resource manager.

   magpie.sbatch-srun-hadoop
   magpie.sbatch-srun-spark

```
sbatch ./magpie.sbatch-srun-hadoop
```

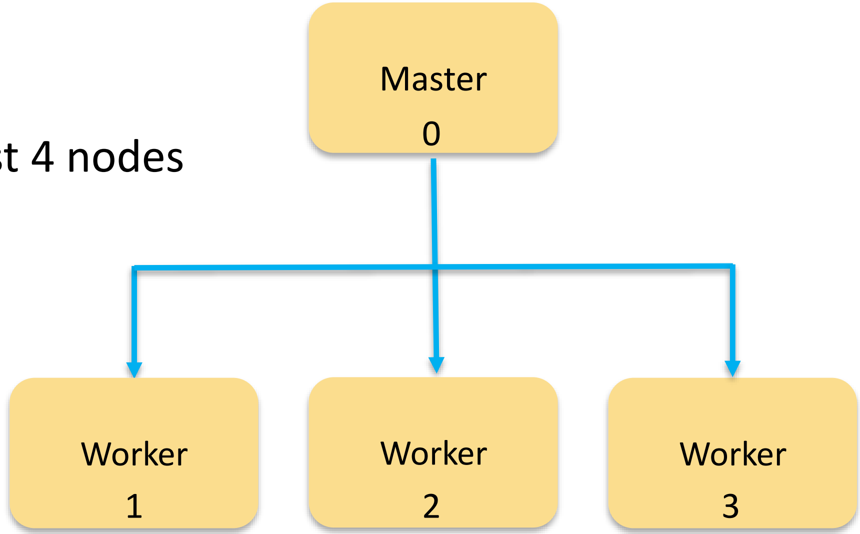
Yeti has 147 compute nodes:

0	1	…	145	146

Magpie Workflow

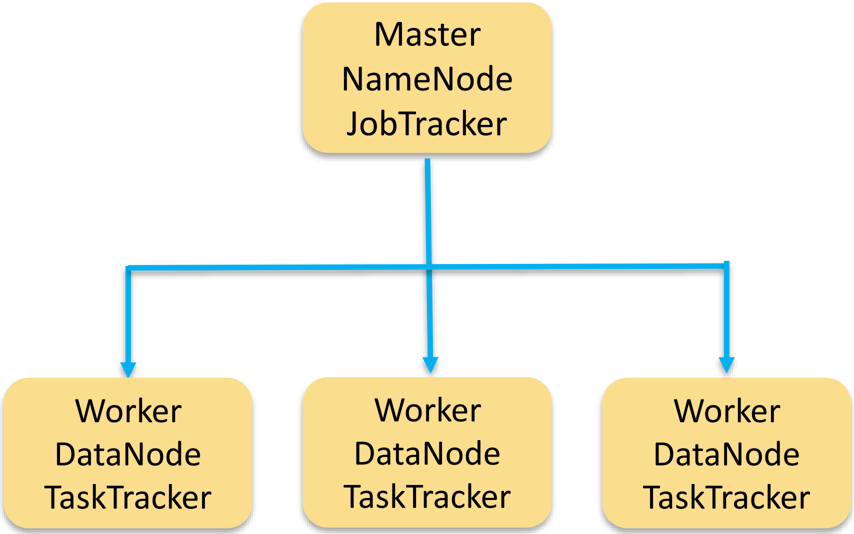
2. The batch script creates configuration files for all appropriate projects (Hadoop/Spark). The configuration files will be setup so the rank 0 node is the "master". All compute nodes will have configuration files created that point to the node designated as the master server.

In the script, request 4 nodes



Magpie Workflow

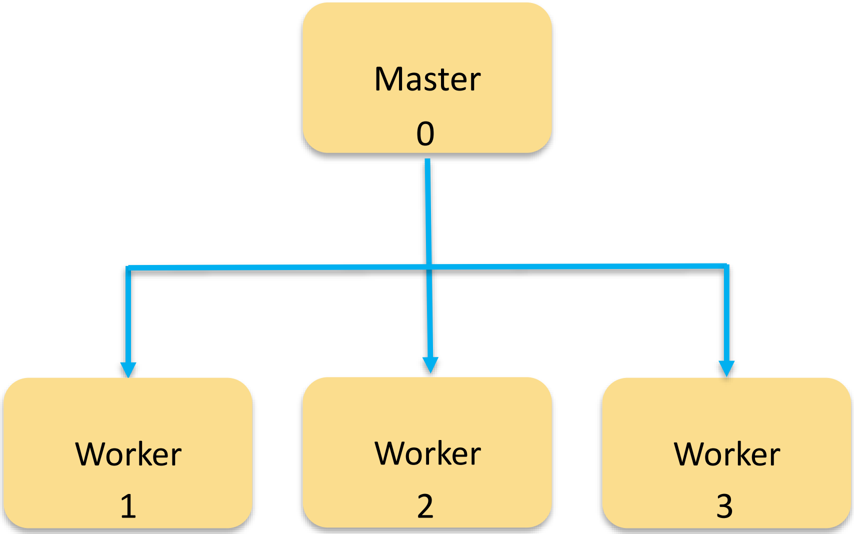
3. Launch daemons on all nodes. The rank 0 node will run master daemons, such as the Hadoop NameNode and JobTracker. All remaining nodes will run appropriate slave daemons, such as the Hadoop DataNodes and TaskTracker.



Magpie Workflow

4. To interact with a mini big data cluster, log into the master node or run a Magpie script to execute a big data calculation instead.

ssh n0

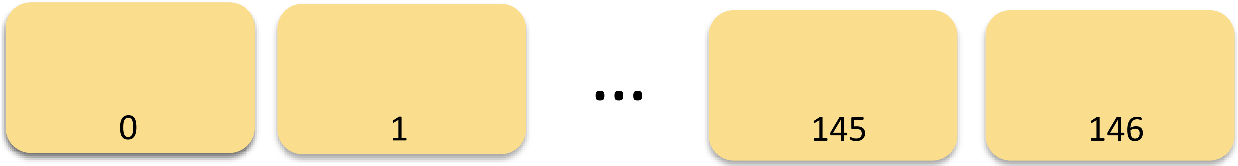


USGS

Magpie Workflow

5. When the job completes or allocation times out, Magpie will cleanup the job by tearing down daemons (plus some additional cleanup work to make re-execution cleaner and faster).

Yeti has 147 compute nodes:



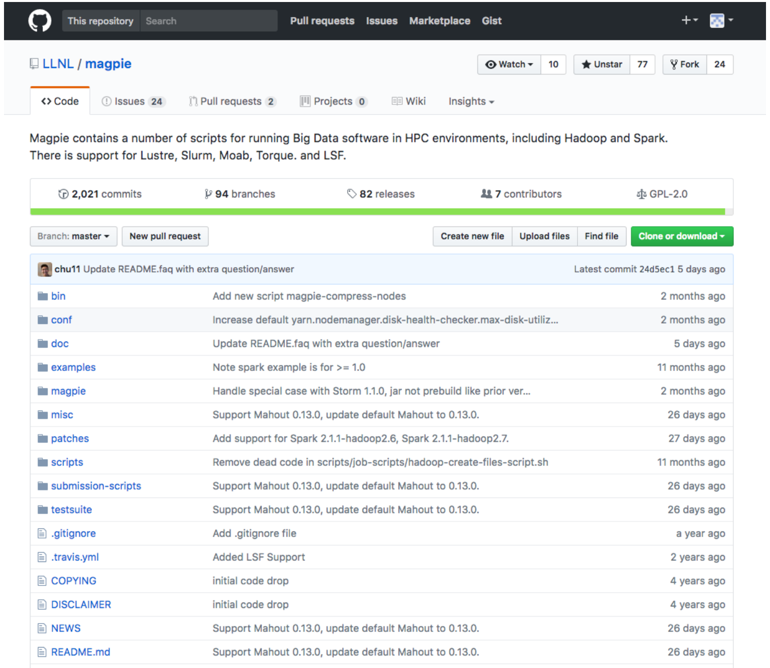
USGS

Integrating Big Data tools on top of USGS HPC cluster

USGS

Installing Hadoop and Spark on Yeti

```
git clone https://github.com/LLNL/magpie.git
```



[Magpie GitHub]

USGS

Installing Hadoop and Spark on Yeti

```
cd magpie/misc
vi magpie-apache-download-and-setup.sh
```

- Script to download the latest Hadoop and Spark versions
- Also available: other Apache software projects (Hbase, Pig, Mahout, Zookeeper, Storm, Phoenix, Kafka, Zeppelin)

Installing Hadoop and Spark on Yeti

```
HADOOP_DOWNLOAD="N"
HBASE_DOWNLOAD="N"
PIG_DOWNLOAD="N"
MAHOUT_DOWNLOAD="N"
ZOOKEEPER_DOWNLOAD="N"
SPARK_DOWNLOAD="N"
STORM_DOWNLOAD="N"
PHOENIX_DOWNLOAD="N"
KAFKA_DOWNLOAD="N"
ZEPPELIN_DOWNLOAD="N”

PRESET_LAUNCH_SCRIPT_CONFIGS="N"
```

USGS

Installing Hadoop and Spark on Yeti

```
HADOOP_DOWNLOAD="Y"
HBASE_DOWNLOAD="N"
PIG_DOWNLOAD="N"
MAHOUT_DOWNLOAD="N"
ZOOKEEPER_DOWNLOAD="N"
SPARK_DOWNLOAD="Y"
STORM_DOWNLOAD="N"
PHOENIX_DOWNLOAD="N"
KAFKA_DOWNLOAD="N"
ZEPPELIN_DOWNLOAD="N”

PRESET_LAUNCH_SCRIPT_CONFIGS="Y"
```

In the script, change "N" to "Y" for Hadoop and Spark (or other Apache software)

Set PRESET_LAUNCH_SCRIPT_CONFIGS to "Y"

USGS

Installing Hadoop and Spark on Yeti

```
HADOOP_PACKAGE="hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz”
SPARK_PACKAGE="spark/spark-2.0.2/spark-2.0.2-bin-hadoop2.7.tgz"
```

- Hadoop version installed is 2.7.3 (25 August, 2016)
- Other Hadoop releases available:
    - 3.0.0-alpha3 (26 May, 2017)
    - 2.8.0 (22 March, 2017)
    - 2.6.5 (8 October, 2016)
- Spark version installed is 2.0.2 (14 November, 2016)
    - 2.1.1 (2 May, 2017)
    - 2.1.0 (28 December, 2016)
    - 2.0.1 (3 October 2016)
    - 2.0.0 (26 July 2016)

USGS

Installing Hadoop and Spark on Yeti

```
mkdir bigdata
sh magpie-apache-download-and-setup.sh
```

- Make directory where to install Hadoop and Spark
- Run the script to download Apache projects for Magpie

Installing Hadoop and Spark on Yeti

```
ls ~/bigdata
```

```
hadoop-2.7.3/ hadoop-2.7.3.tar.gz spark-2.0.2-bin-hadoop2.7/ spark-2.0.2-bin-hadoop2.7.tgz
```

USGS

Running Hadoop on Yeti

```
cd ~/magpie/submission-scripts/script-sbatch-srun
vi magpie.sbatch-srun-hadoop
```

- Script to run Hadoop on Yeti
- Set SLURM flags to utilize Yeti resources
- Other scripts available for Moab, Torque, and LSF schedulers

SLURM Flags in Magpie Script

```
#SBATCH –A training
#SBATCH –p normal
#SBATCH –t 01:00:00
#SBATCH –N 2
#SBATCH --job-name=hadoop_test
#SBATCH --output="hadoop-%j.out"
#SBATCH --ntasks-per-node=1
#SBATCH –exclusive
#SBATCH --mail-type=ALL
#SBATCH --mail-user=youremail@usgs.gov
```

Startup and Shutdown Times

```
export MAGPIE_STARTUP_TIME=30
export MAGPIE_SHUTDOWN_TIME=30
```

- The sum of startup and shutdown times (in minutes) should be less than SLURM flag #SBATCH --time=1:00:00 above

USGS

Java Environment Module

```
module load java/jdk-1.8.0_45
```

- Hadoop is written in Java and requires Java Virtual Machine (JVM) to run
- "module load" loads Java module at runtime
- Sets JAVA_HOME path

Hadoop Job Configurations

```
export HADOOP_MODE="script"
export HADOOP_SCRIPT_PATH="${MAGPIE_SCRIPTS_HOME}/examples/hadoop-example-job-script"
export HADOOP_SCRIPT_ARGS=""
```

- Job type "script" executes a specified script with provided arguments in HADOOP_SCRIPT_ARGS
- Set the path to the script and provide arguments to the script
- Other job types (used to debug/test):
  - "interactive"
  - "launch"
  - "setuponly"

HDFS over Lustre

```
export HADOOP_FILESYSTEM_MODE="hdfsoverlustre"
export HADOOP_HDFSOVERLUSTRE_PATH="/lustre/projects/css/csas/${USER}/hdfsoverlustre/"
export HADOOP_HDFS_BLOCKSIZE=134217728
```

- Sets up HDFS over Lustre file system
- Default HDFS Block size is set to 128 MB (134217728 bytes)

USGS

Hadoop Example

- TeraGen and TeraSort with Hadoop 2.0
- Standard benchmark of MapReduce applications
  - TeraGen generates input dataset – one terabyte of randomly distributed data
  - TeraSort sorts the input data

Hadoop TeraGen and TeraSort

```
vi ~/magpie/examples/hadoop-example-job-script
```

```
command="bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-$HADOOP_VERSION.jar teragen 50000000 terasort-example-job-teragen"
command="bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-$HADOOP_VERSION.jar terasort -Dmapred.reduce.tasks=1 terasort-example-job-teragen terasort-example-job-sort"
```

Submit Job

```
cd ~/magpie/submission-scripts/script-sbatch-srun/
sbatch ./magpie.sbatch-srun-hadoop
```

- Submit the script to SLURM to be executed with "`sbatch`" command

Known Issues

- Job output file has the following exception:

```
*******************************************************

Waiting 30 more seconds for services to come up
Waiting 30 more seconds for services to come up
Waiting 30 more seconds for services to come up
Waiting 30 more seconds for services to come up
...
```

- This exception means that the NameNode is in safe mode
- Need to exit safe mode before the Hadoop program can run

Exit Safe Mode

```
cd ~/magpie/magpie/run
vi magpie-run-project-hadoop
```

- Modify the script to turn off the safe mode

Exit Safe Mode

```
# Ensure namenode isn't in safe mode.
# We do not use "-safemode wait", b/c we want to inform the user
# as we're waiting.
        if [ ${HADOOP_MODE} != "setuponly" ] \
         && Magpie_hadoop_filesystem_mode_is_hdfs_type
     then
         if [ "${hdfs_was_setup}" == "1" ]
         then
             command="${hadoopcmdprefix}/${dfsadminscript} dfsadmin -
safemode leave"
             echo "Running $command" >&2
             $command
            # Return 0 if service up, 1 if not
```

Submit Job

```
cd ~/magpie/submission-scripts/script-sbatch-srun
sbatch ./magpie.sbatch-srun-hadoop
```

- Submit the script to SLURM to be executed with "`sbatch`" command
- Run on 2 nodes (40 CPU cores)

```
JOBID      PART     NAME           TIME ST    TIME_LEFT     USER CPUS  NODES NODELIST(REASON)
2347453    norm hadoop_t           0:06  R      2:59:54 nrapstin   40      2 n3-[88,98]
```

Output File

Should see the following in the output file:

```
*******************************************************
Running bin/hdfs dfsadmin -safemode leave
Safe mode is OFF
starting historyserver, logging to
/tmp/nrapstine/hadoop/hadoop_test/1664986/log/mapred-nrapstine-
historyserver-n3-107.out
*******************************************************
* Executing script /home/nrapstine/magpie/examples/hadoop-example-job-
script
*******************************************************
```

USGS

Output File

```
* Magpie General Job Info
*
* Job Nodelist: n3-[88,98]
* Job Nodecount: 2
* Job Timelimit in Minutes: 180
* Job Name: hadoop_test
* Job ID: 2347439
*
*******************************************************
Starting hadoop
```

USGS

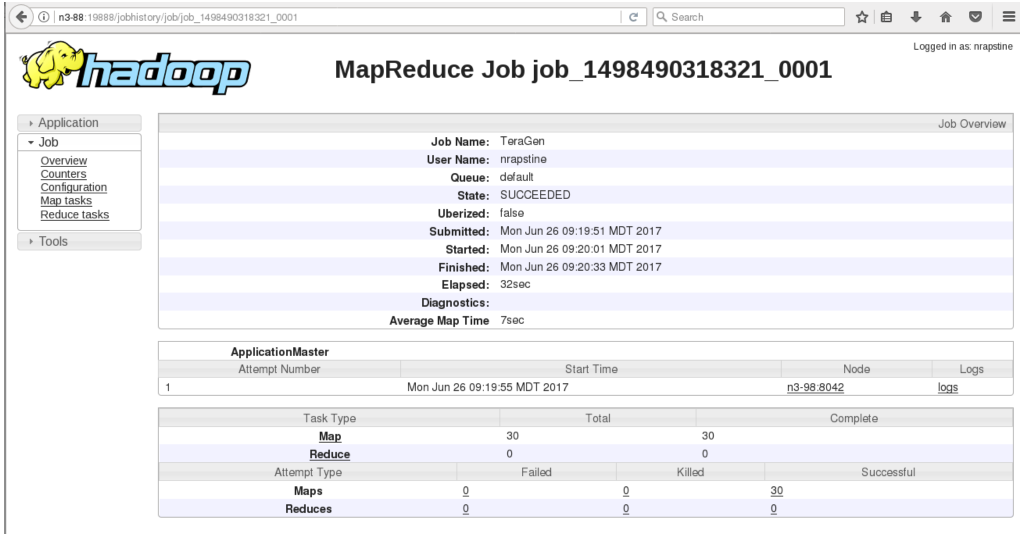
Output File

```
Running bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-
2.7.3.jar teragen 50000000 terasort-example-job-teragen
17/01/05 08:29:02 INFO client.RMProxy: Connecting to ResourceManager at
n3-114/10.10.3.114:8032
17/01/05 08:29:04 INFO terasort.TeraSort: Generating 50000000 using 30
17/01/05 08:29:05 INFO mapreduce.JobSubmitter: number of splits:30
17/01/05 08:29:05 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1483630091065_0001 17/01/05 08:29:06 INFO impl.YarnClientImpl:
Submitted application application_1483630091065_0001
```

TeraGen

- Track job progress in a browser:

```
17/06/26 08:19:26 INFO mapreduce.Job: The url to track the job: http://n3-88:19888/jobhistory/job/job_1498490318321_0001/
```



MapReduce Job

```
17/01/05 08:29:06 INFO mapreduce.Job: Running job: job_1483630091065_0001
17/01/05 08:29:18 INFO mapreduce.Job: Job job_1483630091065_0001 running
in uber mode : false
17/01/05 08:29:18 INFO mapreduce.Job: map 0% reduce 0%
17/01/05 08:29:27 INFO mapreduce.Job: map 7% reduce 0%
17/01/05 08:29:28 INFO mapreduce.Job: map 10% reduce 0%
...
17/01/05 08:29:48 INFO mapreduce.Job: map 100% reduce 0%
17/01/05 08:29:49 INFO mapreduce.Job: Job job_1483630091065_0001 completed
successfully
17/01/05 08:29:49 INFO mapreduce.Job: Counters: 31
...
```

USGS

MapReduce Job

```
Running bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar terasort -Dmapred.reduce.tasks=1 terasort-example-job-teragen terasort-example-job-sort
17/01/05 08:29:51 INFO terasort.TeraSort: starting
17/01/05 08:29:54 INFO input.FileInputFormat: Total input paths to process : 30
...
17/01/05 08:30:07 INFO mapreduce.Job: map 0% reduce 0%
...
17/01/05 08:33:36 INFO mapreduce.Job: map 100% reduce 100%
17/01/05 08:33:38 INFO mapreduce.Job: Job job_1483630091065_0002 completed successfully
17/01/05 08:33:39 INFO mapreduce.Job: Counters: 49
```

MapReduce Job

n3-88:19888/jobhistory/job/job_1498494242252_0002

Logged in as: nrapstine

MapReduce Job job_1498494242252_0002

- Application
- Job
  - Overview
  - Counters
  - Configuration
  - Map tasks
  - Reduce tasks
- Tools

	Job Overview
Job Name:	TeraSort
User Name:	nrapstine
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Mon Jun 26 10:26:15 MDT 2017
Started:	Mon Jun 26 10:26:23 MDT 2017
Finished:	Mon Jun 26 10:31:28 MDT 2017
Elapsed:	5mins, 5sec
Diagnostics:	
Average Map Time	35sec
Average Shuffle Time	2mins, 22sec
Average Merge Time	33sec
Average Reduce Time	1mins, 45sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Mon Jun 26 10:26:18 MDT 2017	n3-98:8042	logs

Task Type	Total	Complete
Map	60	60
Reduce	1	1

Attempt Type	Failed	Killed	Successful
Maps	0	0	60
Reduces	0	0	1

Access HDFS Directly

```
ssh n0
export JAVA_HOME="/cxfs/projects/root/opt/java/java8/jdk1.8.0_45/"
export HADOOP_HOME="/home/nrapstine/bigdata/hadoop-2.7.3"
export HADOOP_CONF_DIR="/tmp/nrapstine/hadoop/hadoop_test/1665465/conf"
```

- If SLURM allocated node n0 (as seen in the output file) for your job, you can interact with the Hadoop file system on that node
- Need to export environmental variables when accessing the node directly

USGS

Access HDFS Directly

- On that node, interact with HDFS:

```
$HADOOP_HOME/bin/hdfs dfs ...
```

- To launch jobs:

```
$HADOOP_HOME/bin/hadoop jar ...
```

USGS

Other Hadoop Benchmarks

```
cd ~/bigdata/hadoop-2.7.3/share/hadoop/mapreduce
```

Other benchmarks in `hadoop-mapreduce-examples-2.7.3.jar`:

- aggregatewordcount: An aggregate based MapReduce program that counts the words in the input files.
- aggregatewordhist: An Aggregate based MapReduce program that computes the histogram of the words in the input files.
- bbp: A MapReduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
- dbcount: An example job that counts the number of page views from a database.

…

Running Spark on Yeti

```
cd ~/magpie/submission-scripts/script-sbatch-srun
vi magpie.sbatch-srun-spark
```

- Script to run Spark on Yeti
- Similar to Hadoop modifications, in the script, set SLURM flags, path to Java, HDFS, and path to Magpie submission script to run

USGS

Spark Example

- SparkPi program
- Python program that computes π approximation
- spark-submit command launces application on a Spark cluster

```
vi ~/magpie/examples/spark-example-job-script
```

```
command="bin/spark-submit --class org.apache.spark.examples.SparkPi examples/jars/spark-examples_2.11-2.0.2.jar"
```

USGS

Submit Job

```
cd ~/magpie/submission-scripts/script-sbatch-srun/
sbatch ./magpie.sbatch-srun-spark
```

- Submit the script to SLURM with “sbatch”

USGS

Missing Jar Error

- Check the output file “spark-<jobid>.out”
- If you see the following error:

```
Local jar ~/bigdata/spark-2.0.2-bin-hadoop2.7/lib/spark-examples-1.0.0-hadoop2.2.0.jar does not exist, skipping.
```

- Spark examples jar is actually in `~/bigdata/spark-2.0.2-bin-hadoop2.7/examples/jars/spark-examples_2.11-2.0.2.jar`

Edit Job Script

- Need to modify example job script

```
cd ~/magpie/examples

vi spark-example-job-script
```

- Comment out older versions of Spark
- Point to the location of spark-examples jar

```
command="bin/spark-submit --class org.apache.spark.examples.SparkPi examples/jars/spark-examples_2.11-2.0.2.jar"
```

USGS

Submit Job

```
sbatch ./magpie.sbatch-srun-spark
```

- Submits the script to SLURM to be executed

Examples

K-means Clustering with Spark

- KDD example:
  - Using Spark to analyze KDD Cup of 1999 data mining competition
  - **Goal**: build a network intrusion detector to detect “bad” connections
  - Data: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
- Batch Mode
  - Submit Magpie batch script to SLURM
  - Runs KDD K-means Python program
- Interactive Mode
  - PySpark shell

KDD.py Program

- K-means clustering program
- Input parameter: max number of clusters k

```
if __name__ == "__main__":
    max_k = 150
    data_file ="/cxfs/projects/css/csasl/arc/tusk/data/kddcup.data"
    sc = SparkContext(conf=conf)
```

USGS

Spark Execution Script

```
cd ~/magpie/examples
cp spark-example-job-script kdd-job-script
vi kdd-job-script
```

- Copy a sample execution script
- Change the script to run KDDCup99.py script with spark-submit

```
cd ${SPARK_HOME}
command="./bin/spark-submit /cxfs/projects/css/csasl/arc/tusk/scripts/KDDCup99.py"
```

USGS

SLURM Submission Script

```
cd ~/magpie/submission-scripts/script-sbatch-srun/
cp magpie.sbatch-srun-spark kdd-spark
vi kdd-spark
```

- Copy a sample submission script
- Point to the `kdd-job-script` to be executed

USGS

SLURM Submission Script

```
#SBATCH --nodes=10
#SBATCH --output="kdd-spark-%j.out"
#SBATCH --time=03:00:00
#SBATCH --job-name=kdd-spark
#SBATCH --mail-type=ALL
#SBATCH --mail-user=<your_email>@usgs.gov

export MAGPIE_SCRIPT_PATH="${MAGPIE_SCRIPTS_HOME}/examples/kdd_job_script"
```

USGS

Submit Job

```
sbatch ./kdd-spark
```

- Submits the script to SLURM to be executed on 10 nodes (200 CPU cores)

Interactive PySpark Shell

- Create Python 2.7 environment

```
module load python/anaconda2
conda create --name py27 python=2.7
pip install pandas scikit-learn scipy numpy
```

- Launch an interactive PySpark shell:

```
/home/nrapstine/bigdata/spark-2.0.2-bin-hadoop2.7/bin/pyspark
```

USGS

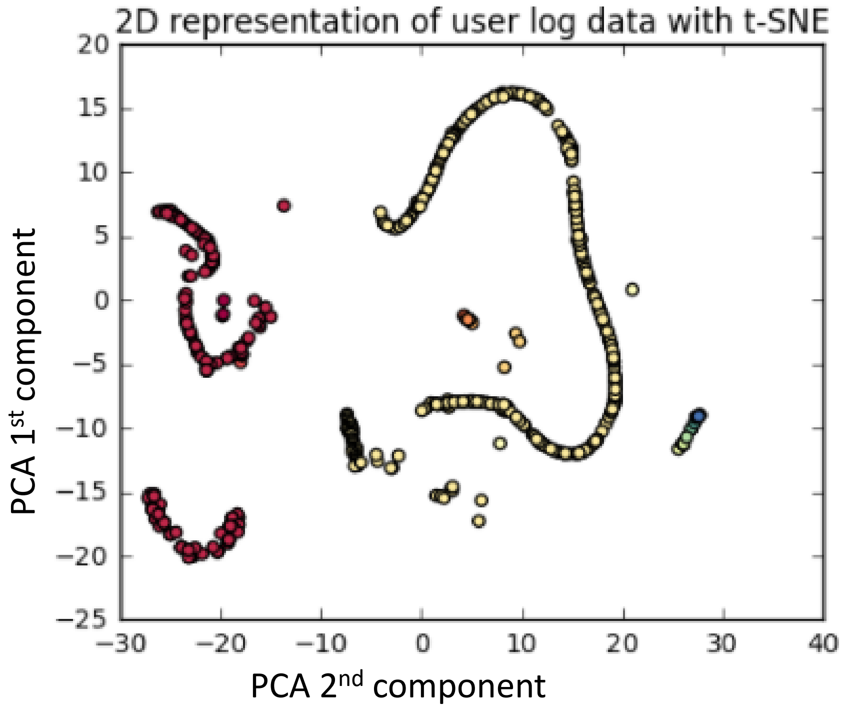
```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.0.2
      /_/

Using Python version 2.7.13 (default, Dec 20 2016 23:09:15)
SparkSession available as 'spark'.
>>> import pandas
>>> from time import time
>>> col_names = ["duration","protocol_type","service","flag","src_bytes",
"dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins",
"logged_in","num_compromised","root_shell","su_attempted","num_root",
"num_file_creations","num_shells","num_access_files","num_outbound_cmds",
"is_host_login","is_guest_login","count","srv_count","serror_rate",
"srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate",
"diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
"dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
"dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
"dst_host_rerror_rate","dst_host_srv_rerror_rate","label"]
```

```
>>> print "Predicted in {} seconds".format(round(tt,3))
Predicted in 629.934 seconds
>>> from sklearn.metrics import accuracy_score
>>> acc = accuracy_score(pred, labels_test)
>>> print "R squared is {}.".format(round(acc,4))
R squared is 0.9253.
>>> from sklearn.cluster import KMeans
>>> k = 30
>>> km = KMeans(n_clusters = k)
>>> t0 = time()
>>> km.fit(features)
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=30, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
>>> tt = time()-t0
>>> print "Clustered in {} seconds".format(round(tt,3))
Clustered in 90.119 seconds
```

Yeti Daily Logs Analysis

Yeti user behavior analysis:

- Yeti daily logs record every job ever submitted on the cluster
- Features: number of nodes, cores, partition, required memory, elapsed time, time limit and job state
- **Goal**: cluster Yeti users



USGS

Magpie

HPC systems
USGS Yeti Supercomputer

Big Data Tools
Apache Hadoop
Apache Spark

USGS

APACHE hadoop

APACHE Spark

Summary

- Install big data tools on already existing HPC infrastructure without the need for costly cloud computing services
- Open source Magpie software is successfully used to integrate Hadoop and Spark on USGS HPC system
- Spark programs can be written in Python and run with the HPC scheduler SLURM to process data efficiently

nrapstine@usgs.gov