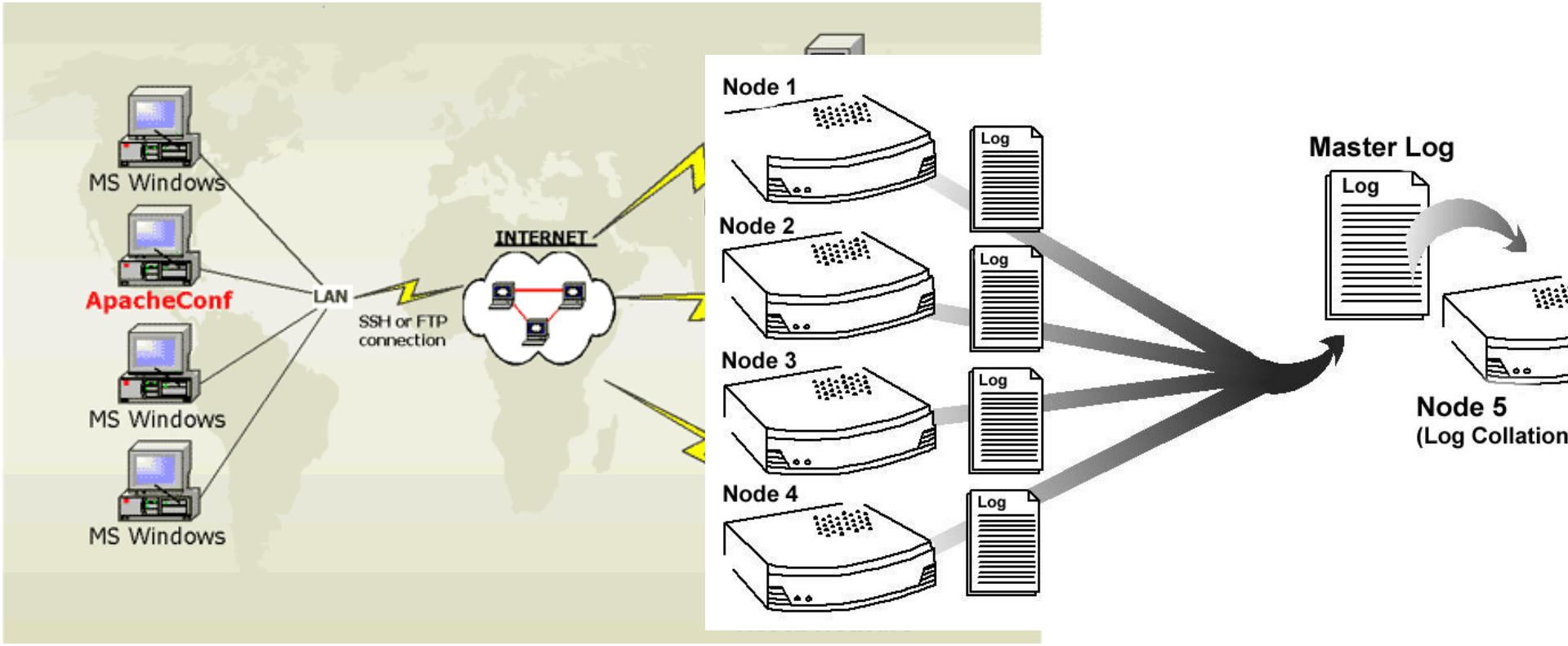
Mapreduce Programming at Comet and HW2 Log Analysis

UCSB CS240A 2016. Tao Yang

Data Analysis from Web Server Logs



Startup code and data : /home/tyang/cs240sample/log

apache1.splunk.com

apache2.splunk.com

apache3.splunk.com

Example line of the log file

66.249.64.13 - - [18/Sep/2004:11:07:48 +1000] "GET / HTTP/1.0" 200 6433 "-" "Googlebot/2.1"

10.32.1.43 - - [06/Feb/2013:00:07:00] "GET /flower_store/product.screen?product_id=FL-DLH-02 HTTP/1.1" 200 10901 "http://mystore.splunk.com/flower_store/category.screen?category_id=GIFTS&JSESSIONID=SD7SL1FF9ADFF2" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.10) Gecko/20070223 CentOS/1.5.0.10-0.1.el4.centos Firefox/1.5.0.10" 4361 3217

Log Format

66.249.64.13 - - [18/Sep/2004:11:07:48 +1000]
"GET / HTTP/1.0" 200 6433 "-" "Googlebot/2.1"

%h	Logs the remote host
%l	Remote logname, if supplied
%u	Remote user (mostly useful if logging behind authentication)
%t	The date and time of the request
%r	The request to your web site
%s	The status of the request (201, 301, 404, 500, etc.), the > in front of the "s" insures only the last status is logged.
%b	Bytes sent for the request (tracks http bandwidth use)
%i	Tracks items sent in the HTML header. So by adding (Referer) and (User Agent), we are capturing the referring url and the browser type in the combined log format.

More Formal Definition of Apache Log

```
%h %l %u %t "%r" %s %b "%{Referer}i" "%{User-agent}i“
```

%h = IP address of the client (remote host) which made the request

%l = RFC 1413 identity of the client

%u = userid of the person requesting the document

%t = Time that the server finished processing the request

%r = Request line from the client in double quotes

%s = Status code that the server sends back to the client

%b = Size of the object returned to the client

Referer : where the request originated

User-agent what type of agent made the request.

http://www.the-art-of-web.com/system/logs/

Common Response Code

- 200 - OK
- 206 - Partial Content
- 301 - Moved Permanently
- 302 - Found
- 304 - Not Modified
- 401 - Unauthorised (password required)
- 403 - Forbidden
- 404 - Not Found.

LogAnalyzer.java

```
public class LogAnalyzer {
	public static void main(String[] args) throws Exception {
		Configuration conf = new Configuration();
		if (args.length != 2) {
			System.err.println("Usage: loganalyzer <in> <out>");
			System.exit(2);
		}
		Job job = new Job(conf, "analyze log");
		job.setJarByClass(LogAnalyzer.class);
		job.setMapperClass(Map.class);
		job.setReducerClass(Reduce.class);
		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(IntWritable.class);
		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job, new Path(args[1]));
		System.exit(job.waitForCompletion(true) ? 0 : 1);
	}
}
```

Map.java

```
public class Map extends Mapper<Object, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text url = new Text();
        private Pattern p = Pattern.compile("(?:GET|POST)\\s([^\\s]+)");
    @Override
        public void map(Object key, Text value, Context context)
                throws IOException, InterruptedException {
                String[] entries = value.toString().split("\r?\n");
                for (int i=0, len=entries.length; i<len; i+=1) {
                        Matcher matcher = p.matcher(entries[i]);
                        if (matcher.find()) {
                                url.set(matcher.group(1));
                                context.write(url, one);
                        }
                }
        }
}
```

Reduce.java

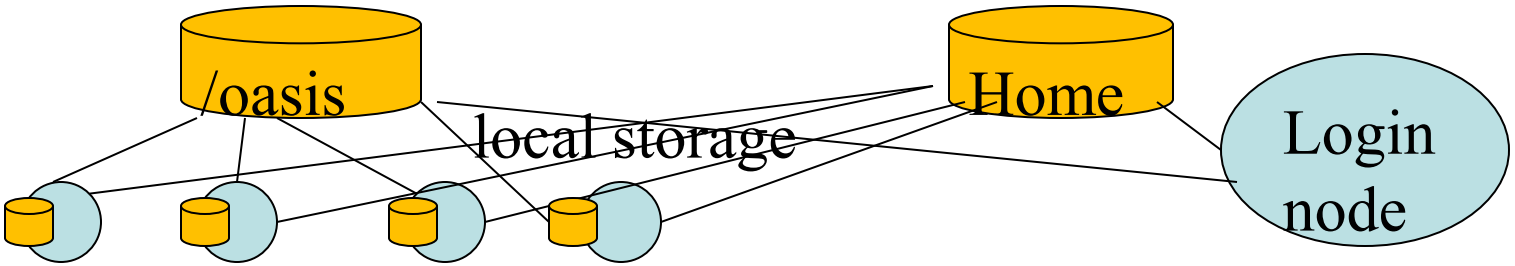
```
public class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable total = new IntWritable();

    @Override
        public void reduce(Text key, Iterable<IntWritable> values, Context
     context)
            throws IOException, InterruptedException {
            int sum = 0;
          for (IntWritable value : values) {
            sum += value.get();
          }
          total.set(sum);
          context.write(key, total);
        }
}
```

Comet Cluster

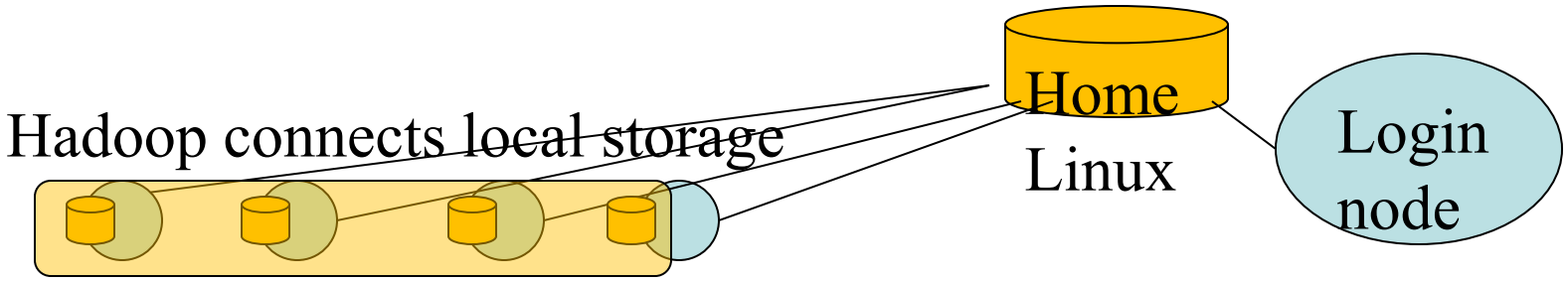
- Comet cluster has 1944 nodes and each node has 24 cores, built on two 12-core Intel Xeon E5-2680v3 2.5 GHz processors
- 128 GB memory and 320GB SSD for local scratch space.
- Attached storage: Shared 7 petabytes of 200 GB/second performance storage and 6 petabytes of 100 GB/second durable storage
  - Lustre Storage Area is a Parallel File System (PFS) called Data Oasis.
    - Users can access from

/oasis/scratch/comet/$USER/temp_project



Hadoop installation at Comet

- ***Installed in /opt/hadoop/1.2.1***
- **Configure Hadoop on-demand with myHadoop:**
  - /opt/hadoop/contrib/myHadoop/bin/myhadoop-configure.sh

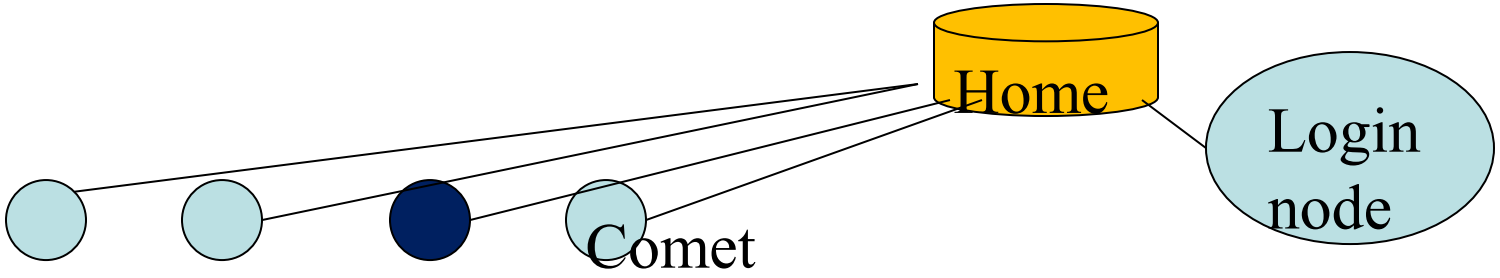


Hadoop file system is built dynamically on the nodes allocated. Deleted when the allocation is terminated.

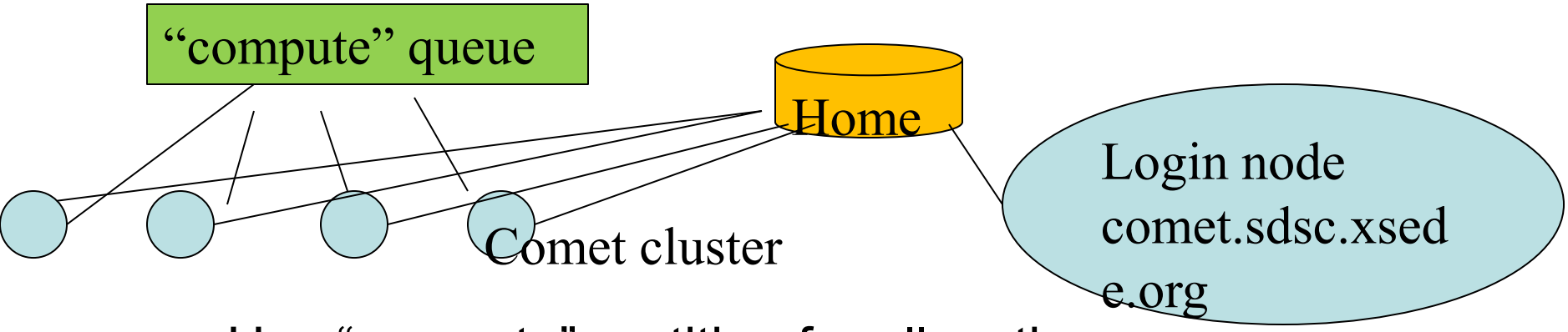
Compile the sample Java code at Comet

Java word count example is available at Comet under /home/tyang/cs240sample/mapreduce/.

- **cp –r /home/tyang/cs240sample/mapreduce .**
- **Allocate a dedicated machine for compiling**
  - /share/apps/compute/interactive/qsubi.bash -p compute --nodes=1 --ntasks-per-node=1 -t 00:
- **Change work directory to mapreduce and type make**
  - Java code is compiled under target subdirectory



How to Run a WordCount Mapreduce Job

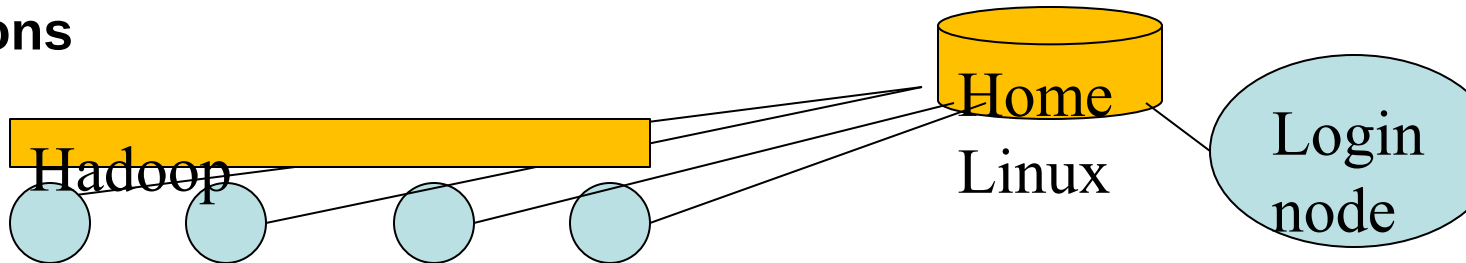


- Use “compute” partition for allocation
- Use Java word count example at Comet under /home/tyang/cs240sample/mapreduce/.
- sbatch submit-hadoop-comet.sh
  - Data input is in test.txt
  - Data output is in WC-output
- Job trace is wordcount.1569018.comet-17-14.out

Sample script (submit-hadoop-comet.sh)

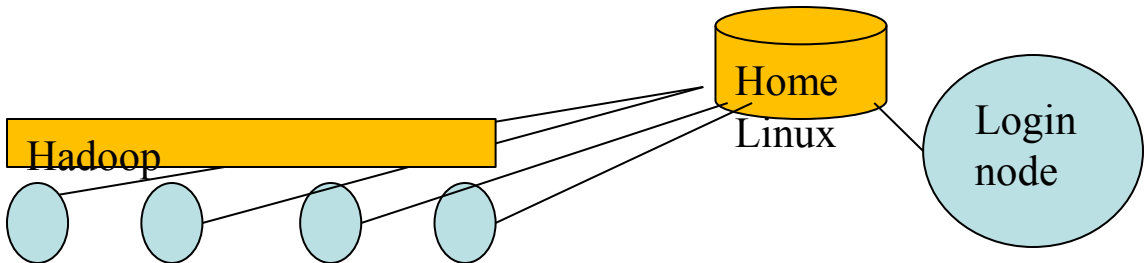
```
#!/bin/bash
#SBATCH --job-name="wordcount"
#SBATCH --output="wordcount.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=24
#SBATCH -t 00:15:00
Export HADOOP_CONF_DIR=/home/$USER/cometcluster
export WORKDIR=`pwd`
module load hadoop/1.2.1

#Use myheadoop to build a Hadoop file system on allocated nodes
myhadoop-configure.sh
#Start all demons
start-all.sh
```



Sample script

```
#make an input directory in the hadoop file system
hadoop dfs -mkdir input
#copy data from local Linux file system to the Hadoop file system
hadoop dfs -copyFromLocal $WORKDIR/test.txt input/
#Run Hadoop wordcount job
hadoop jar $WORKDIR/wordcount.jar  wordcount input/ output/
# Create a local directory WC-output to host the output data
# It does not report error even the file does not exist
rm -rf WC-out >/dev/null || true
mkdir -p WC-out
# Copy out the output data
hadoop dfs -copyToLocal output/part* WC-out
#Stop all demons and cleanup
stop-all.sh
myhadoop-cleanup.sh
```



Sample output trace wordcount.1569018.comet-17-14.out

starting namenode, logging to /scratch/tyang/1569018/logs/hadoop-tyang-namenode-comet-17-14.out

comet-17-14.ibnet: starting **datanode,** logging to /scratch/tyang/1569018/logs/hadoop-tyang-datanode-comet-17-14.sdsc.edu.out

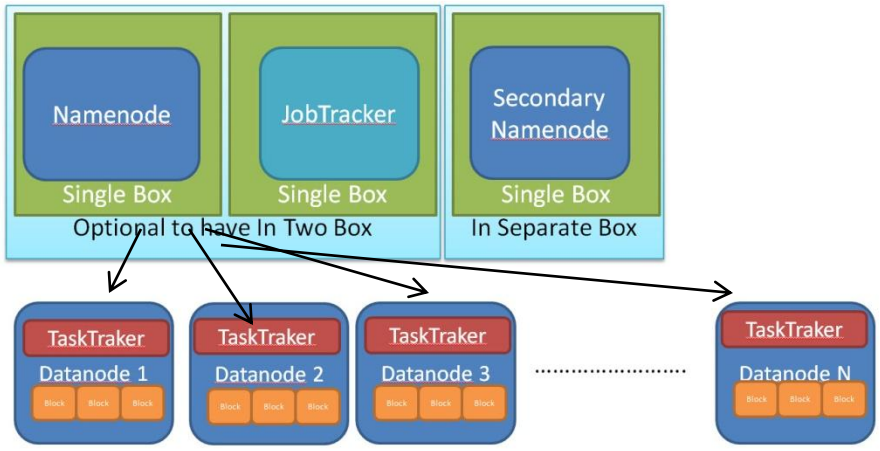
comet-17-15.ibnet: starting **datanode,** logging to /scratch/tyang/1569018/logs/hadoop-tyang-datanode-comet-17-15.sdsc.edu.out

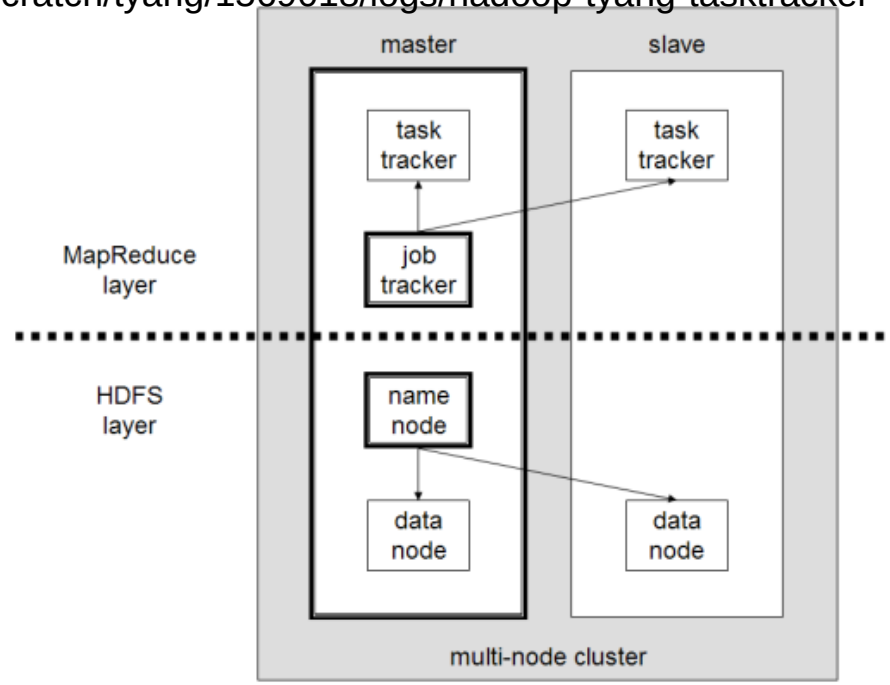
comet-17-14.ibnet: starting **secondarynamenode,** logging to /scratch/tyang/1569018/logs/hadoop-tyang-secondarynamenode-comet-17-14.sdsc.edu.out

starting **jobtracker**, logging to /scratch/tyang/1569018/logs/hadoop-tyang-jobtracker-comet-17-14.out

comet-17-14.ibnet: starting **tasktracker,** logging to /scratch/tyang/1569018/logs/hadoop-tyang-tasktracker-comet-17-14.sdsc.edu.out

comet-17-15.ibnet: starting **tasktracker,** logging to /scratch/tyang/1569018/logs/hadoop-tyang-tasktracker-comet-17-15.sdsc.edu.out

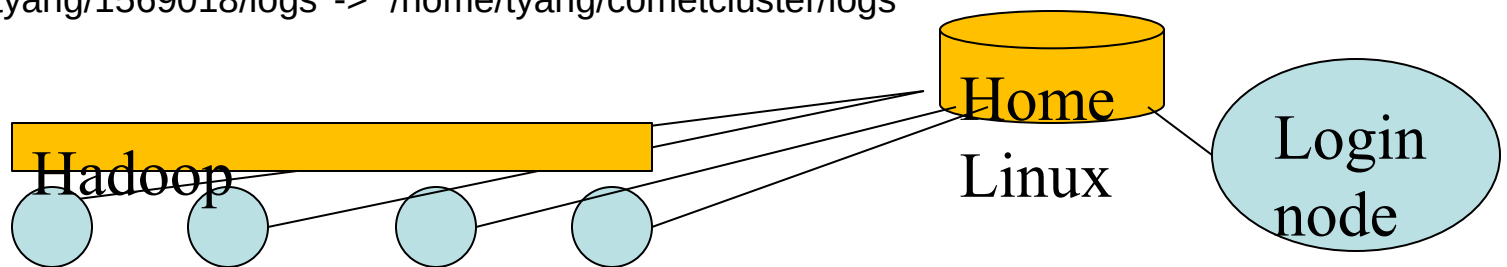




Sample output trace wordcount.1569018.comet-17-14.out

16/01/31 17:43:44 INFO input.FileInputFormat: Total input paths to process : 1
16/01/31 17:43:44 INFO util.NativeCodeLoader: Loaded the native-hadoop library
16/01/31 17:43:44 WARN snappy.LoadSnappy: Snappy native library not loaded
16/01/31 17:43:44 INFO mapred.JobClient: Running job: job_201601311743_0001
16/01/31 17:43:45 INFO mapred.JobClient: **map 0% reduce 0%**
16/01/31 17:43:49 INFO mapred.JobClient: **map 100% reduce 0%**
16/01/31 17:43:56 INFO mapred.JobClient: map 100% reduce 33%
16/01/31 17:43:57 INFO mapred.JobClient: **map 100% reduce 100%**
16/01/31 17:43:57 INFO mapred.JobClient: **Job complete: job_201601311743_0001**

comet-17-14.ibnet: stopping tasktracker
comet-17-15.ibnet: stopping tasktracker
stopping namenode
comet-17-14.ibnet: stopping datanode
comet-17-15.ibnet: stopping datanode
comet-17-14.ibnet: stopping secondarynamenode
Copying Hadoop logs back to /home/tyang/cometcluster/logs...
`/scratch/tyang/1569018/logs' -> `/home/tyang/cometcluster/logs'



Sample input and output

$ cat test.txt

how are you today 3 4 mapreduce program
1 2 3 test send
how are you mapreduce
1 send test USA california new

$ cat WC-out/part-r-00000

1 2
2 1
3 2
4 1
USA 1
are 2
california 1
how 2
mapreduce 2
new 1
program 1
send 2
test 2
today 1
you 2

Shell Commands for Hadoop File System

- **Mkdir, ls, cat, cp**
  - hadoop dfs -mkdir /user/deepak/dir1
  - hadoop dfs -ls /user/deepak
  - hadoop dfs -cat /usr/deepak/file.txt
  - hadoop dfs -cp /user/deepak/dir1/abc.txt /user/deepak/dir2
- **Copy data from the local file system to HDF**
  - hadoop dfs -copyFromLocal <src:localFileSystem> <dest:Hdfs>
  - Ex: hadoop dfs –copyFromLocal /home/hduser/def.txt /user/deepak/dir1
- **Copy data from HDF to local**
  - hadoop dfs -copyToLocal <src:Hdfs> <dest:localFileSystem>

http://www.bigdataplanet.info/2013/10/All-Hadoop-Shell-Commands-you-need-Hadoop-Tutorial-Part-5.html

Notes

- **Java process listing “jps”, shows the following demons**

  NameNode (master), SecondaryNameNode, Datanode (hadoop),JobTracker, TaskTracker
- **To check the status of your job**

  squeue -u username
- **To cancel a submitted job**

  scancel job-id
- You have to request *all* 24 cores on the nodes. Hadoop is java based and any memory limits start causing problems. Also, in the compute partition you are charged for the whole node anyway.

Notes

- Your script should delete the outout directory if you want to rerun and copy out data to that directory. Otherwise the Hadoop copy back fails because the file already exists.

  The current script forces to remove "WC-output".
- If you are running several Mapreduce jobs simultaneously, please make sure you choose different locations for for the configuration files. Basically change the line:

```
export HADOOP_CONF_DIR=/home/$USER/cometcluster
```

to point to different directories for each run. Otherwise the configuration from different jobs will overwrite in the same directory and cause problems.