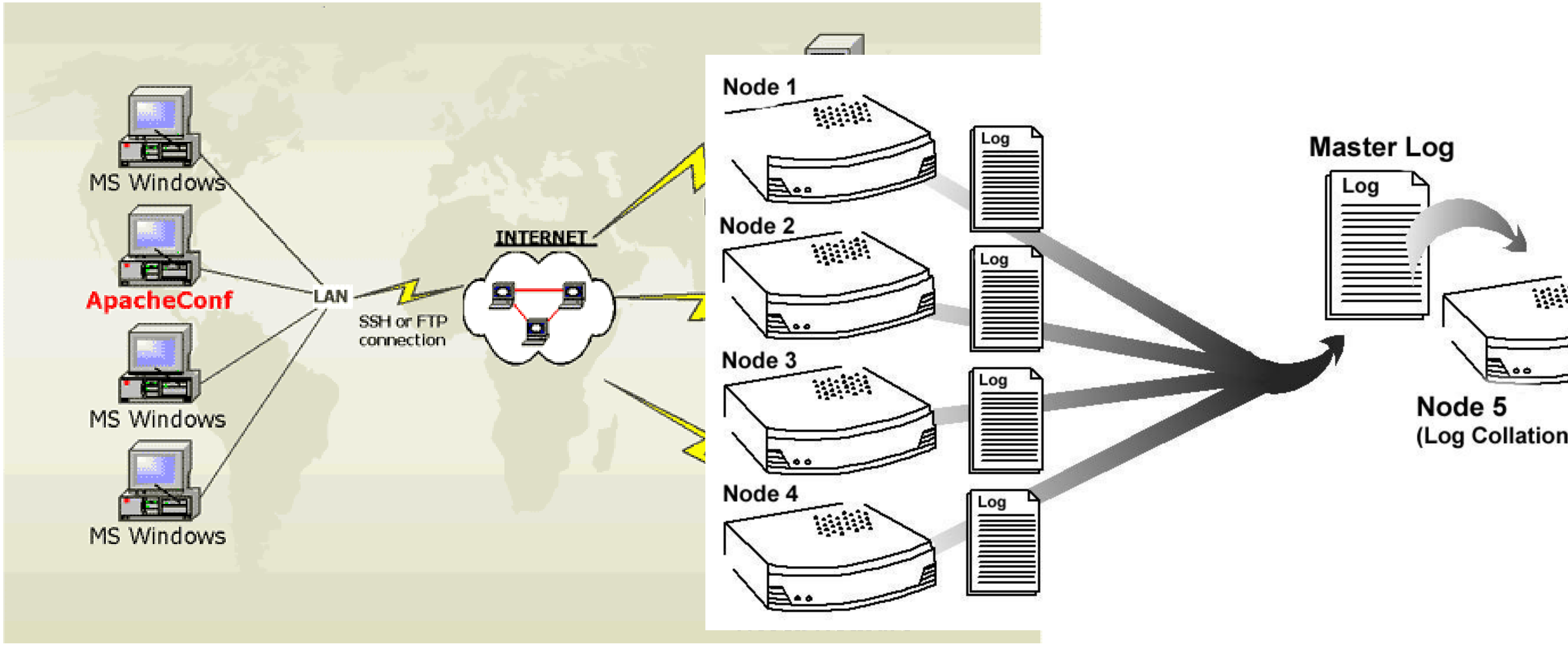
Mapreduce Programming at TSCC and HW4

UCSB CS140 2014. Tao Yang

CS140 HW4: Data Analysis from Web Server Logs



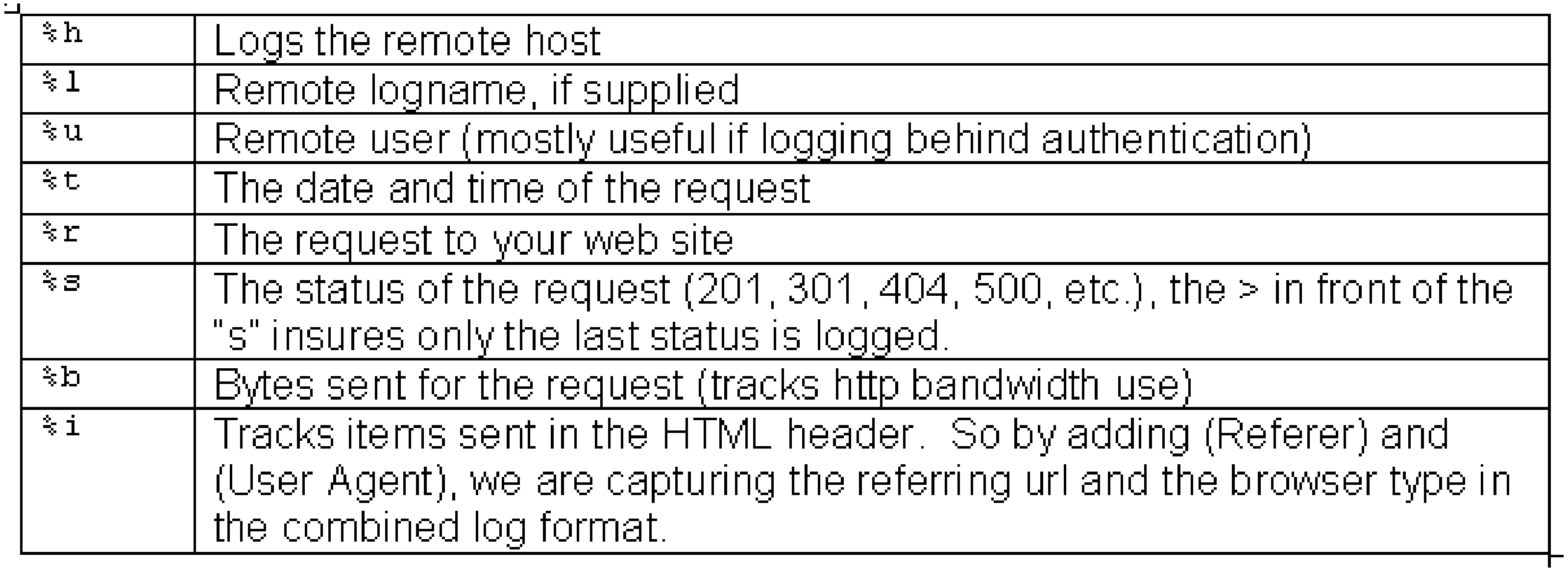
Example line of the log file

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

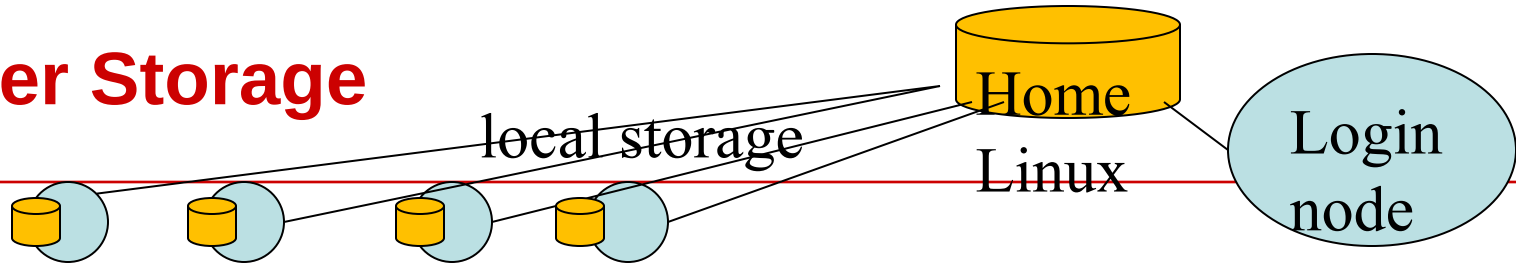
```
public class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
	private IntWritable total = new IntWritable();

	@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);
		}
}
```

TSCC Cluster at San Diego Supercomputer Center

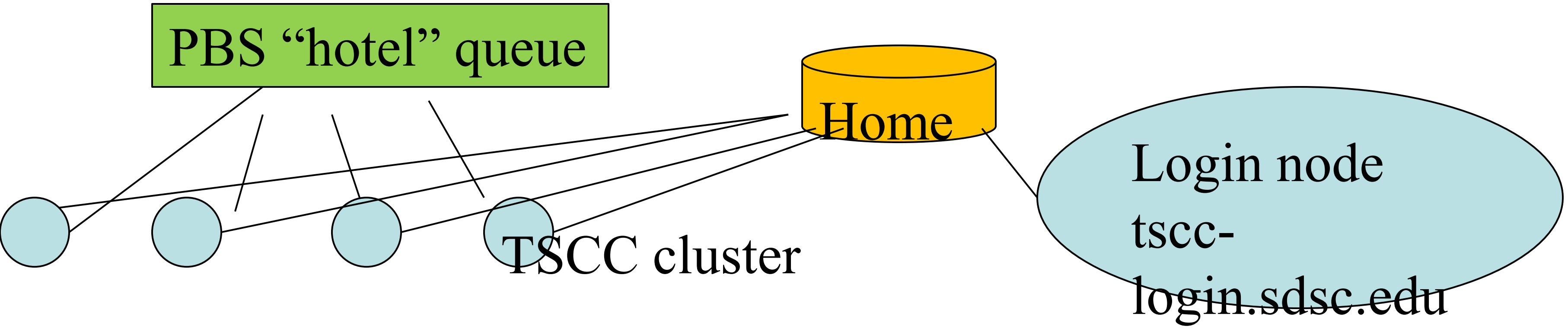
- Processors: Dual-socket, 8-core, 2.6GHz Intel Xeon E5-2670 (Sandy Bridge)
- Memory: 64GB (4GB/core)
- Local storage. 500GB onboard
- The cluster has an attached storage:
  - Lustre Storage Area is a Parallel File System (PFS) called Data Oasis. It contains at least 200TB of shared scratch space available to all users.

User Storage



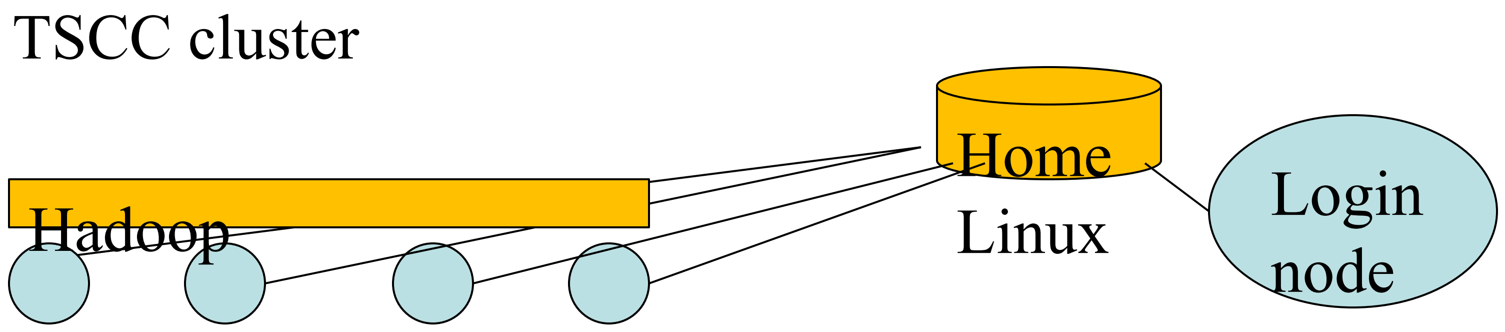
- <u>Home Area Storage</u>
  - on NFS servers using ZFS as the underlying file system. 36TB shared
  - 100GB+ per user.   E.g. /home/tyang
  - 10GbE; Delivers > 300Mb/sec to single node; > 500Mb/sec aggregate
- <u>Local Node Temporary Space</u>.
  - 6GB/core. upto 95GB on /tmp and about 285GB in   /state/partition1/$USER/$PBS_JOBID
  - RAID 1 mirror. 50Mb/sec/node;
  - No backup. Purged between job.

How to Run a Parallel Job



- Use "hotel" PBS queue
- Execute a job in one of two modes
  - Interactive
    - qsub -I -l nodes=2:ppn=1 -l walltime=00:15:00
  - qsub job-script-file
    - qsub   shell-script-file
- Java word counting example is available at TSCC under /home/tyang/wc1.
  - Execute using wordcount5.sh

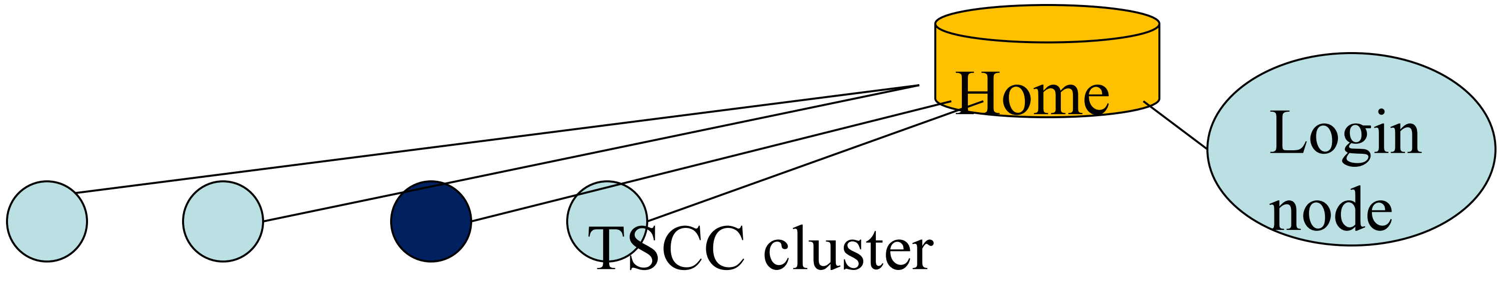
How to Execute Log Processing Sample



- ssh tscc-login.sdsc.edu -l tyang
- cd log
- Debugging mode:
  - Allocate 2 nodes interactively using
    - qsub -I -l nodes=2:ppn=1 -l walltime=00:15:00
  - Execute a script to create Hadoop file system, and run the log processing job.
    - sh log.sh
  - Type: exit

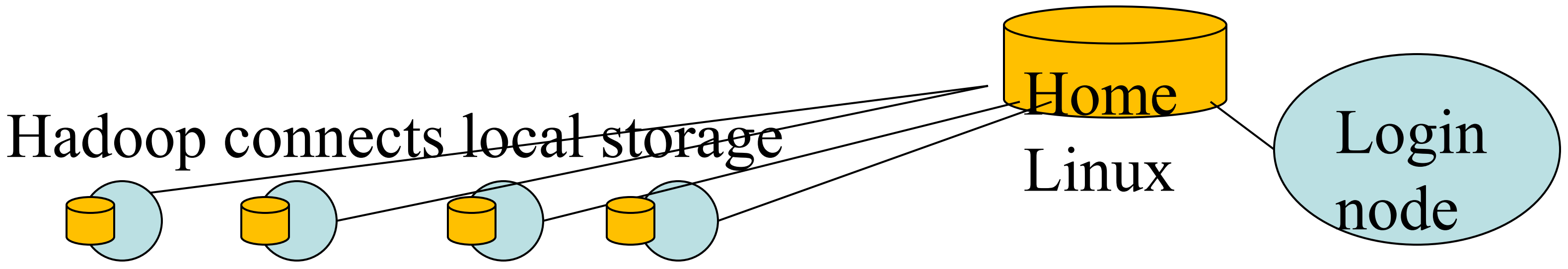
Compile the sample log code at TSCC

- **Copy code/data from /home/tyang/log to your own directory.**
- **Allocate a machine for compiling**
  - qsub -I -l nodes=1:ppn=1 -l walltime=00:15:00
- **Change directory to log and type make**
  - Java code is compiled to produce loganalyzer.jar



Hadoop installation at TSC

- ***Installed in /opt/hadoop/***
  - *Management script is called myhadoop*
  - *Only accessible from the computing nodes.*
- **Configure Hadoop on-demand with myHadoop:**
  - Request nodes using PBS
    - *For example, #PBS –l nodes=2:ppn=1*
  - Configure (transient mode. Use local temporary storage)
    - $MY_HADOOP_HOME/bin/pbs-configure.sh -n 2 –c $HADOOP_CONF_DIR



Shell Commands for Hadoop File System

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

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

The head of sample script (log.sh)

- #!/bin/bash

- #PBS -q hotel
- #PBS -N LogSample
- #PBS -l nodes=2:ppn=1
- #PBS -o user.out
- #PBS -e user.err
- #PBS -l walltime=00:10:00
- #PBS -A your-account-name
- #PBS -V
- #PBS -M your-email@cs.ucsb.edu
- #PBS -m abe

Sample script log.sh (Continue)

- **Setup environment variables properly**

  export MY_HADOOP_HOME="/opt/hadoop/contrib/myHadoop“

  export HADOOP_HOME="/opt/hadoop/"

  export HADOOP_CONF_DIR="/home/tyang/log/ConfDir“

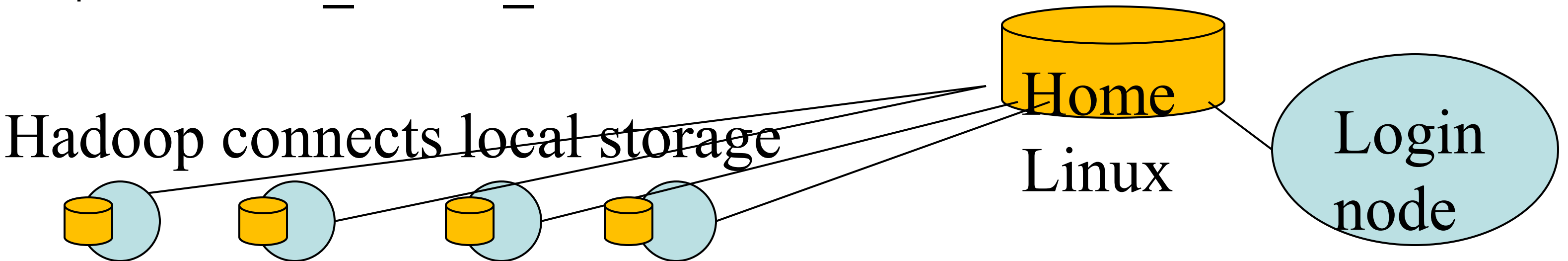
  export HADOOP_DATA_DIR="/state/partition1/$USER/$PBS_JOBID/data“

  export HADOOP_LOG_DIR="/state/partition1/$USER/$PBS_JOBID/log“

- **Set up the configurations for myHadoop**

Create a configuration directory for Hadoop based on machines allocated during qsub.

  - $MY_HADOOP_HOME/bin/pbs-configure.sh -n 2 -c $HADOOP_CONF_DIR



Sample script log.sh (Continue)

- **Format HDFS**
  - $HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR namenode –format
  - More hadoop shell command: http://hadoop.apache.org/docs/stable/file_system_shell.html

  http://hadoop.apache.org/docs/stable/commands_manual.html

- **Start daemons in all nodes for Hadoop**
  - $HADOOP_HOME/bin/start-all.sh

- **If you type “jps”, you will see the following demon processes:**

  NameNode (master), SecondaryNameNode, Datanode (hadoop),JobTracker, TaskTracker

Script log.sh (Continue)

- **Copy data to HDFS**
  - $HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -copyFromLocal ~/log/templog1 input/a
- **Run log analysis job**
  - time $HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR jar loganalyzer.jar LogAnalyzer input output
  - Use /home/username/log/Loganalyzer.jar during batch submission
  - Successful running information displayed.

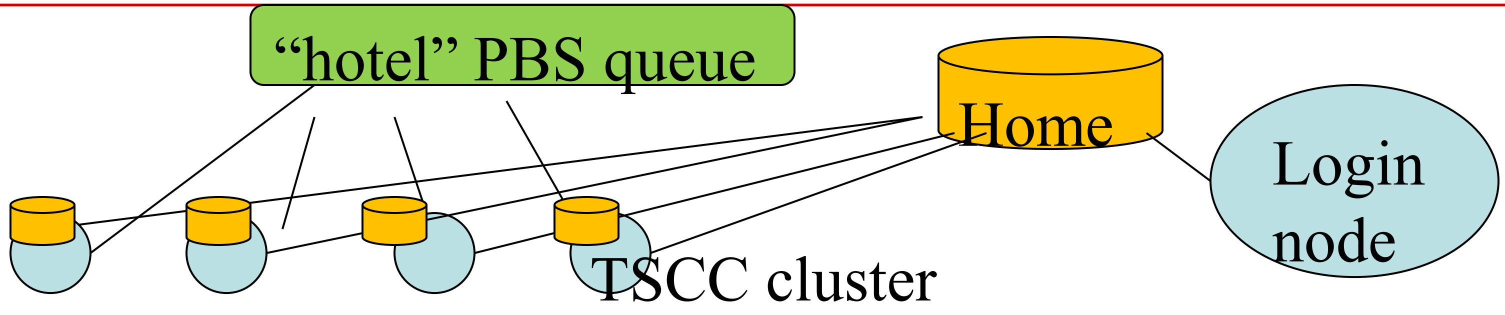
```
14/02/04 12:26:57 INFO mapred.JobClient:  map 0% reduce 0%
14/02/04 12:27:02 INFO mapred.JobClient:  map 100% reduce 0%
14/02/04 12:27:09 INFO mapred.JobClient:  map 100% reduce 33%
14/02/04 12:27:11 INFO mapred.JobClient:  map 100% reduce 100%
```

Script log.sh (Continue)

- **Copy out the output results**
  - $HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR dfs -copyToLocal output ~/log/output

- **Stop all Hadoop daemons and cleanup**
  - $HADOOP_HOME/bin/stop-all.sh
  - $MY_HADOOP_HOME/bin/pbs-cleanup.sh -n 2

Node allocation and Hadoop consistency



- **Node allocation through PBS**
  - The processors per node (ppn) are set to 1.
  - For example, qsub -I -l nodes=2:ppn=1 -l walltime=00:10:00
- **Consistency in dynamic Hadoop configuration**:
  - "-n" option is set consistently in commands
    - $MY_HADOOP_HOME/bin/pbs-configure.sh
    - $MY_HADOOP_HOME/bin/pbs-cleanup.sh

Job execution commands

- **Account balance**
  - gbalance -u username
  - Charge formula:

  #CPUs x #nodes x wall-clock-time.
- **You receive emails on job starting and completion**
  - qstat -a and showq -u $USER
    - examining the current state of your jobs.
  - In qstat, Q means queued, R means running, and C means complete
- **Delete a job**
  - qdel jobnumber
  - jobnumber was assigned by qsub.