Big Data Analytics – Module 3 (MapReduce)

Managing Configuration- 3 Configuration Files

- When developing Hadoop applications, it is common to switch between running the application locally and running it on a cluster.
- In fact, you may have several clusters you work with, or you may have a local “pseudodistributed” cluster that you like to test on
- One way to accommodate these variations is to have Hadoop configuration files containing the connection settings for each cluster you run against and specify which one you are using when you run Hadoop applications or tools.
- As a matter of best practice, it’s recommended to keep these files outside Hadoop’s installation directory tree, as this makes it easy to switch between Hadoop versions without duplicating or losing settings.

Managing Configuration- 3 Configuration Files

- Let us assume the existence of a directory called **conf** that contains three configuration files: **hadoop-local.xml**, **hadoop-localhost.xml**, **and** **hadoop-cluster.xml**.
- They pack up some configuration settings.
- The **hadoop-local.xml** file contains the default Hadoop configuration for the default filesystem and the jobtracker.

Managing Configuration- 3 Configuration Files

```
<?xml version="1.0"?>
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>file:///</value>
  </property>

  <property>
    <name>mapred.job.tracker</name>
    <value>local</value>
  </property>
</configuration>
```

Managing Configuration- 3 Configuration Files

- The settings in **hadoop-localhost.xml** point to a namenode and a jobtracker both running on localhost:

```
<?xml version="1.0"?>
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://localhost/</value>
    </property>

    <property>
        <name>mapred.job.tracker</name>
        <value>localhost:8021</value>
    </property>
</configuration>
```

Managing Configuration- 3 Configuration Files

- Finally, **hadoop-cluster.xml** contains details of the cluster’s namenode and jobtracker addresses. In practice, you would name the file after the name of the cluster, rather than “cluster” as we have here:

```
<?xml version="1.0"?>
<configuration>
   <property>
          <name>fs.default.name</name>
          <value>hdfs://namenode/</value>
    </property>
   <property>
          <name>mapred.job.tracker</name>
          <value>jobtracker:8021</value>
    </property>
</configuration
```

Running on a Cluster – Steps

- Packaging a Job
- Launching a Job
- MapReduce Web UI
- Retrieving Results
- Debugging a Job

Running on a Cluster – **Packaging a Job**

- Now that we are happy with the program running on a small test dataset, we are ready to try it on the full dataset on a Hadoop cluster.
- The local job runner uses a single JVM to run a job, so as long as all the classes that your job needs are on its classpath, then things will just work.
- In a distributed setting, things are a little more complex. For a start, a job's classes must be packaged into a **job JAR file** to send to the cluster.
- Hadoop will find the job JAR automatically by searching for the JAR on the driver's classpath that contains the class set in the **setJarByClass()** method (on **JobConf** or **Job**).
- Alternatively, if you want to set an explicit JAR file by its file path, you can use the **setJar()** method.

Running on a Cluster – **Packaging a Job**

- Creating a job JAR file is conveniently achieved using a build tool such as Ant or Maven.
- The following Maven command, for example, will create a JAR file called **hadoop-examples.jar** in the project directory containing all of the compiled classes:

```
% mvn package -DskipTests
```

- If you have a single job per JAR, you can specify the main class to run in the JAR file's manifest.
- If the main class is not in the manifest, it must be specified on the command line.
- Any dependent JAR files can be packaged in a **lib** subdirectory in the job JAR file.
- Similarly, resource files can be packaged in a **classes** subdirectory.

Running on a Cluster - Packaging a Job - The Client Classpath

- The user's client-side classpath set by **hadoop jar <jar>** is made up of:
  - The job JAR file
  - Any JAR files in the ***lib*** directory of the job JAR file, and the ***classes*** directory (if present)
  - The **classpath** defined by **HADOOP_CLASSPATH**, if set
- This explains why you have to set **HADOOP_CLASSPATH** to point to dependent classes and libraries if you are running using the local job runner without a job JAR (hadoop **CLASSNAME**)

Running on a Cluster - Packaging a Job - The Task Classpath

- On a cluster(and this includes pseudodistributed mode), map and reduce tasks run in separate JVMs, and their classpaths are not controlled by **HADOOP_CLASSPATH.**
- **HADOOP_CLASSPATH** is a client-side setting and only sets the classpath for the driver JVM, which submits the job.
- Instead, the user's task classpath is comprised of the following:
  - The job JAR file
  - Any JAR files contained in the ***lib*** directory of the job JAR file, and the ***classes*** directory (if present)
  - Any files added to the distributed cache, using the **–libjars** option, or the **addFileToClassPath()** method on **Job**

Running on a Cluster - Packaging a Job - Packaging Dependencies

- Given these different ways of controlling what is on the client and task classpaths, there are corresponding options for including library dependencies for a job.
  - Unpack the libraries and repackage them in the job JAR.
  - Package the libraries in the ***lib*** directory of the job JAR. •
  - Keep the libraries separate from the job JAR, and add them to the client classpath via **HADOOP_CLASSPATH** and to the task classpath via **-libjars.**
- The last option, using the distributed cache, is simplest from a build point of view because dependencies don't need rebundling in the job JAR.
- Also, the distributed cache can mean fewer transfers of JAR files around the cluster, since files may be cached on a node between tasks.

Running on a Cluster - Launching the Job

To launch the job, we need to run the driver, specifying the cluster that we want to run the job on with the -conf option.

```
% unset HADOOP_CLASSPATH
% hadoop jar hadoop-examples.jar v3.MaxTemperatureDriver \
-conf conf/hadoop-cluster.xml input/ncdc/all max-temp
```

The waitForCompletion() method on Job launches the job and polls for progress, writing a line summarizing the map and reduce’s progress whenever either changes.

The output includes more useful information. Before the job starts, its ID is printed; this is needed whenever you want to refer to the job—in logfiles.

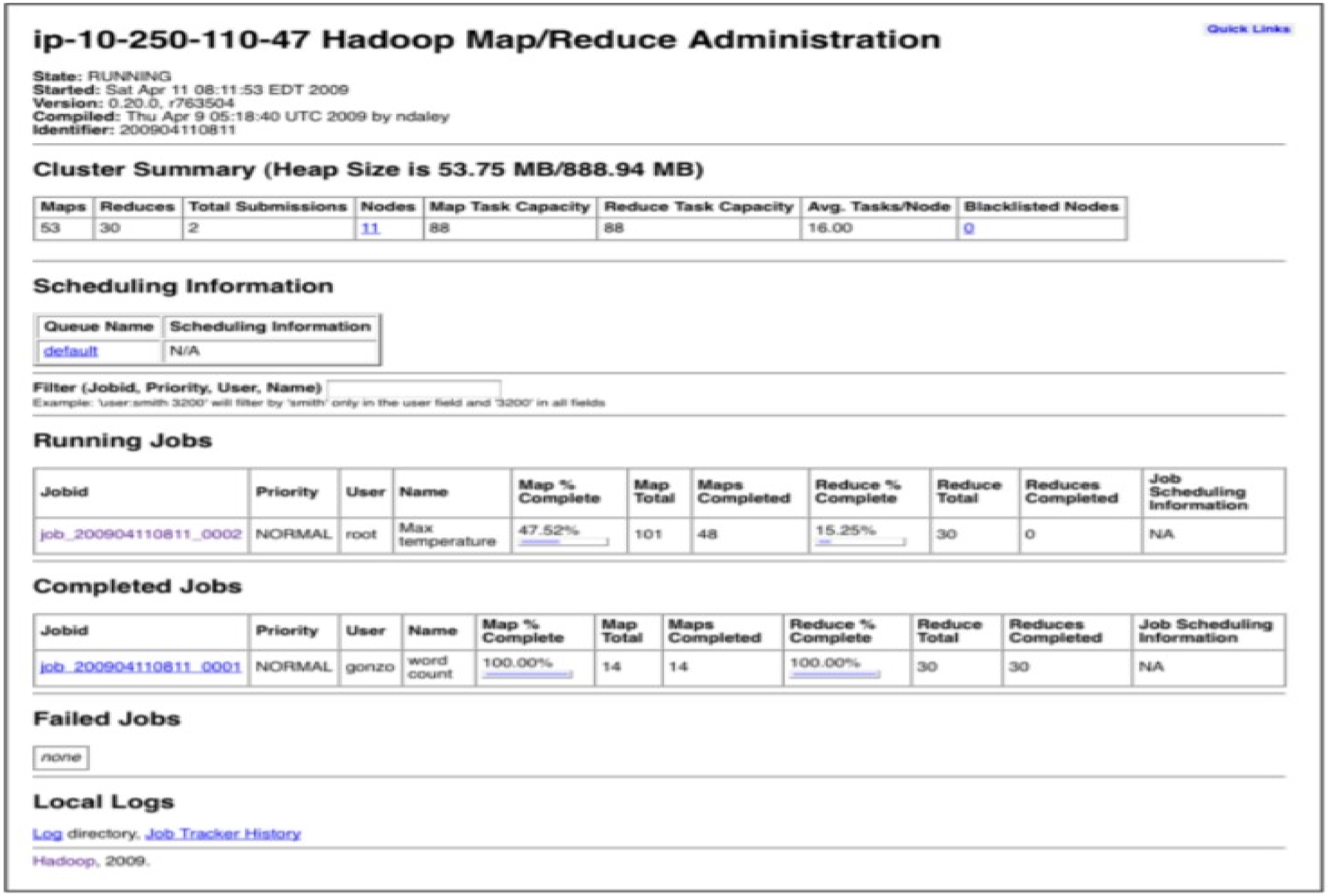
When the job is complete, its statistics (known as counters) are printed out.

Running on a Cluster - The MapReduce Web UI

- Hadoop comes with a web UI for viewing information about your jobs. It is useful for following a job’s progress while it is running, as well as finding job statistics and logs after the job has completed
- You can find the UI at

http://jobtracker-host:50030/

Running on a Cluster - The MapReduce Web UI - The JobTracker page



Quick Links

ip-10-250-110-47 Hadoop Map/Reduce Administration

State: RUNNING
Started: Sat Apr 11 08:11:53 EDT 2009
Version: 0.20.0, r763504
Compiled: Thu Apr 9 05:18:40 UTC 2009 by ndaley
Identifier: 200904110811

Cluster Summary (Heap Size is 53.75 MB/888.94 MB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
53	30	2	11	88	88	16.00	0

Scheduling Information

Queue Name	Scheduling Information
default	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information
job_200904110811_0002	NORMAL	root	Max temperature	47.52%	101	48	15.25%	30	0	NA

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information
job_200904110811_0001	NORMAL	gonzo	word count	100.00%	14	14	100.00%	30	30	NA

Failed Jobs

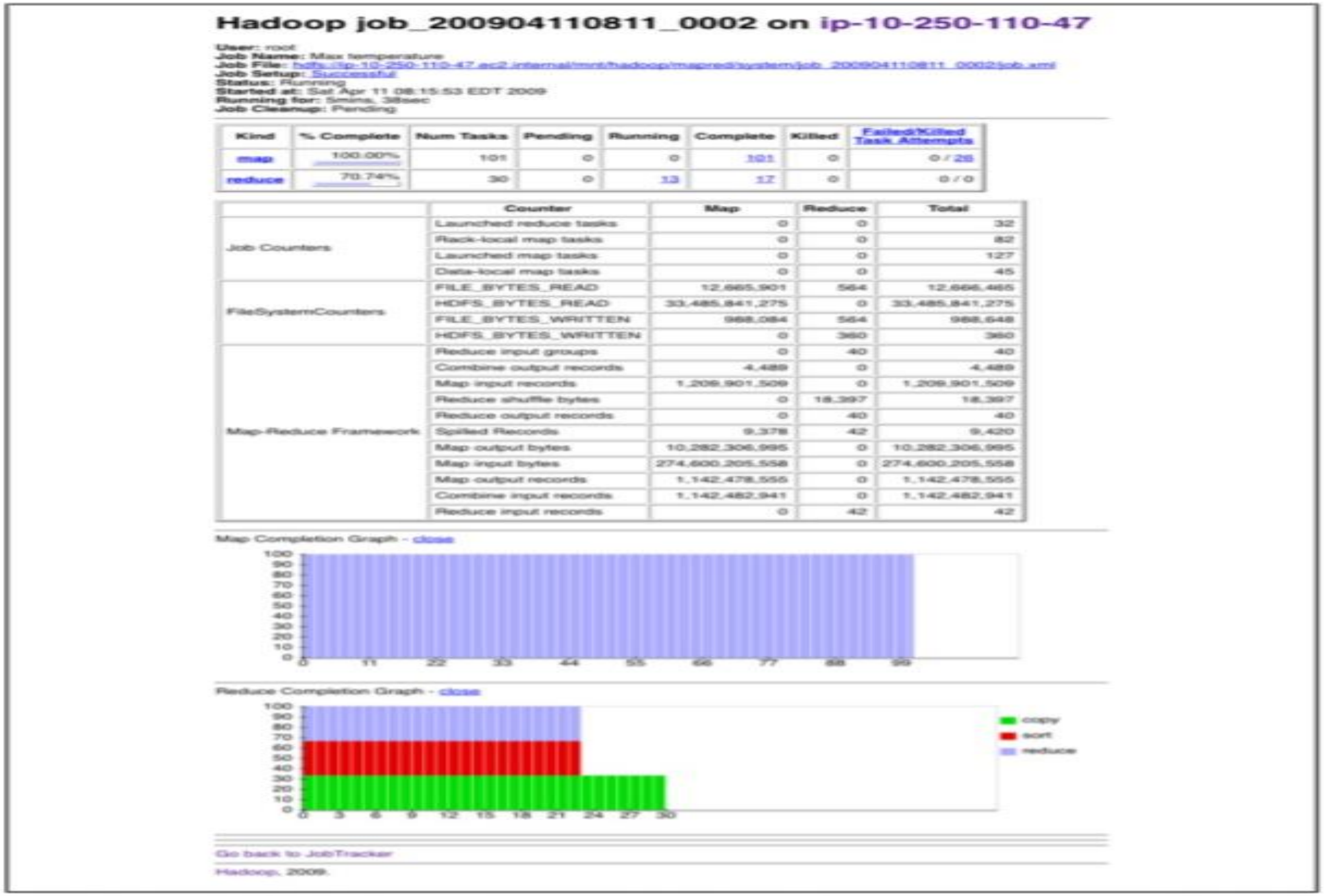
none

Local Logs

Log directory, Job Tracker History

Hadoop, 2009.

Running on a Cluster - The MapReduce Web UI - The Job page



Hadoop job_200904110811_0002 on ip-10-250-110-47

User: root
Job Name: Max temperature
Job File: hdfs://ip-10-250-110-47.ec2.internal/mnt/hadoop/mapred/system/job_200904110811_0002/job.xml
Job Setup: Successful
Status: Running
Started at: Sat Apr 11 08:15:53 EDT 2009
Running for: 5mins, 38sec
Job Cleanup: Pending

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	101	0	0	101	0	0 / 26
reduce	70.74%	30	0	13	17	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	Launched reduce tasks	0	0	32
	Rack-local map tasks	0	0	82
	Launched map tasks	0	0	127
	Data-local map tasks	0	0	45
FileSystemCounters	FILE_BYTES_READ	12,665,901	564	12,666,465
	HDFS_BYTES_READ	33,485,841,275	0	33,485,841,275
	FILE_BYTES_WRITTEN	988,084	564	988,648
	HDFS_BYTES_WRITTEN	0	360	360
Map-Reduce Framework	Reduce input groups	0	40	40
	Combine output records	4,489	0	4,489
	Map input records	1,209,901,509	0	1,209,901,509
	Reduce shuffle bytes	0	18,397	18,397
	Reduce output records	0	40	40
	Spilled Records	9,378	42	9,420
	Map output bytes	10,282,306,995	0	10,282,306,995
	Map input bytes	274,600,205,558	0	274,600,205,558
	Map output records	1,142,478,555	0	1,142,478,555
	Combine input records	1,142,482,941	0	1,142,482,941
	Reduce input records	0	42	42

Map Completion Graph - close

Reduce Completion Graph - close

Go back to JobTracker

Hadoop, 2009.

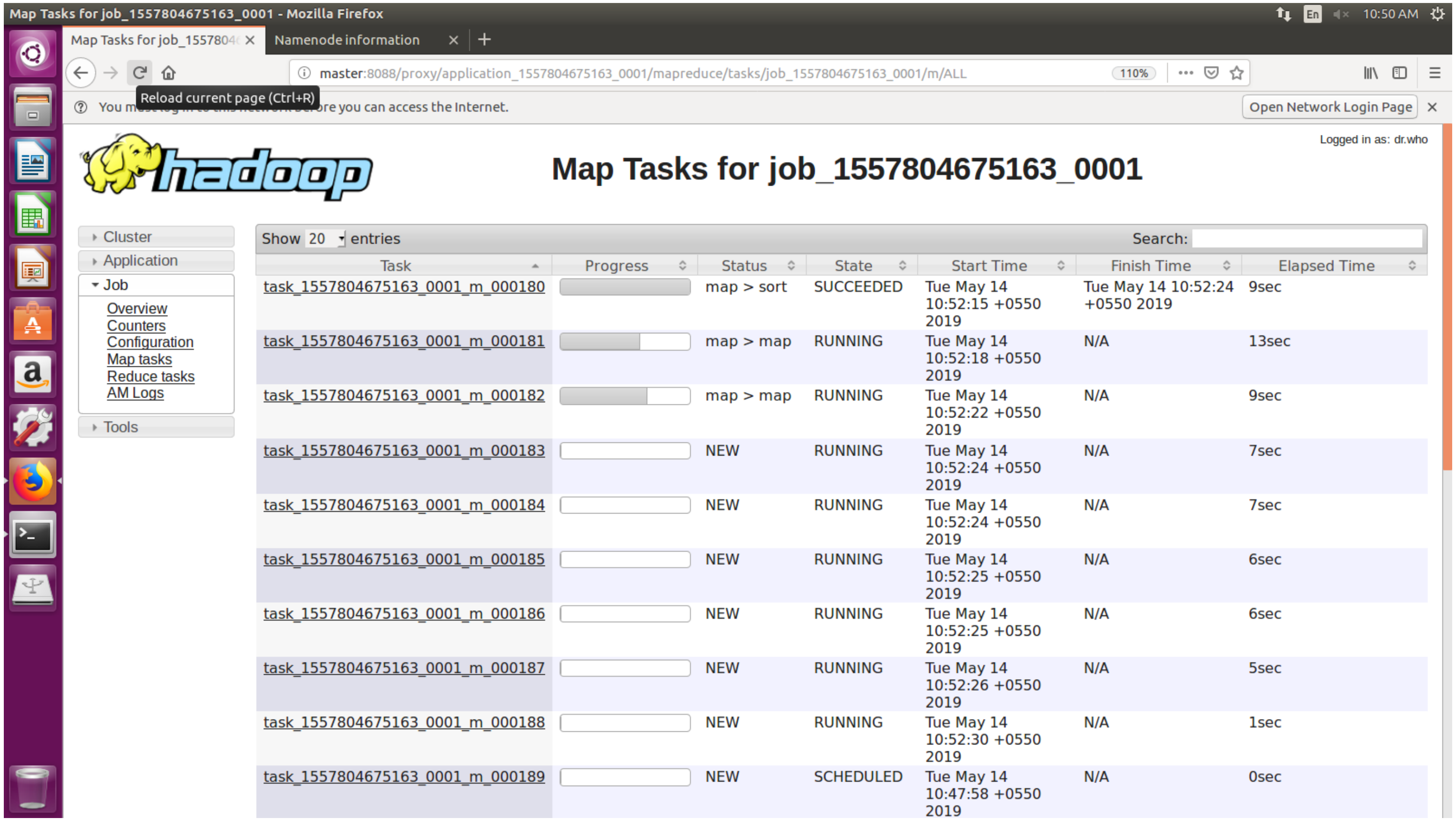
Debugging a Job

- We can use a debug statement to log to standard error, in conjunction with a message to update the task's status message to prompt us to look in the error log. The web UI makes this easy.
- Ex: We can also create a custom counter to count the total number of records with unreasonable temperatures in the whole dataset.
- This gives us valuable information about how to deal with the condition.
- We add our debugging to the mapper, as opposed to the reducer, as we want to find out what source data causing the abnormal output.

The tasks page

- The job page has a number of links for viewing the tasks in a job in more detail.
- For example, by clicking on the “map” link, you are brought to a page that lists information for all of the map tasks on one page.
- You can also see just the completed tasks

Screenshot of the tasks page



Map Tasks for job_1557804675163_0001 - Mozilla Firefox

master:8088/proxy/application_1557804675163_0001/mapreduce/tasks/job_1557804675163_0001/m/ALL

Logged in as: dr.who

Map Tasks for job_1557804675163_0001

Show 20 entries Search:

Task	Progress	Status	State	Start Time	Finish Time	Elapsed Time
task_1557804675163_0001_m_000180		map > sort	SUCCEEDED	Tue May 14 10:52:15 +0550 2019	Tue May 14 10:52:24 +0550 2019	9sec
task_1557804675163_0001_m_000181		map > map	RUNNING	Tue May 14 10:52:18 +0550 2019	N/A	13sec
task_1557804675163_0001_m_000182		map > map	RUNNING	Tue May 14 10:52:22 +0550 2019	N/A	9sec
task_1557804675163_0001_m_000183		NEW	RUNNING	Tue May 14 10:52:24 +0550 2019	N/A	7sec
task_1557804675163_0001_m_000184		NEW	RUNNING	Tue May 14 10:52:24 +0550 2019	N/A	7sec
task_1557804675163_0001_m_000185		NEW	RUNNING	Tue May 14 10:52:25 +0550 2019	N/A	6sec
task_1557804675163_0001_m_000186		NEW	RUNNING	Tue May 14 10:52:25 +0550 2019	N/A	6sec
task_1557804675163_0001_m_000187		NEW	RUNNING	Tue May 14 10:52:26 +0550 2019	N/A	5sec
task_1557804675163_0001_m_000188		NEW	RUNNING	Tue May 14 10:52:30 +0550 2019	N/A	1sec
task_1557804675163_0001_m_000189		NEW	SCHEDULED	Tue May 14 10:47:58 +0550 2019	N/A	0sec