Running Hadoop Programs on cslvm Cluster

Overview

This document contains information that is specific to running Hadoop programs on our local cluster.

CSLVM Hadoop Cluster

We have a Hadoop installation running on `cslvm31`, `cslvm32` and `cslvm33` machines.

To successfully run Hadoop jobs on the cluster, you need to know the following.

Location of Hadoop Binaries. Hadoop is intalled in `/usr/local/hadoop`. The `hadoop`, `hdfs`, and `yarn` binaries are located in `/usr/local/hadoop/bin`.

You should edit your `.bashrc` file on `cslvm31` (`cslvm32`, `cslvm33`) and add the following command:

```
export PATH=$PATH:/usr/local/hadoop/bin
```

Our `Hadoop` installation is version 2.6.

Java Hadoop Package. Core Hadoop library is available in the `hadoop-core-1.2.1.jar` jar file. The file can be downloaded from the course web page:

> http://users.csc.calpoly.edu/~dekhtyar/369-Winter2016/code/hadoop-core-1.2.1.jar

Compiling Hadoop code. To compile a Hadoop job located in file `myJob.java`, place the `hadoop-core-1.2.1.jar` file in the directory where `myJob.java` is located and run the following command from that directory:

```
$ javac -cp hadoop-core-1.2.1.jar myJob.java
```

Note: To stop prolifiration of the `hadoop-core-1.2.1.jar` I recommend creating a ∼`userid/jars` directory, placing the jar file there, and adding the `jars` directory to the classpath.

Making a Jar of your Hadoop Job. Hadoop allows running `.class` files only on local intallations. Our Hadoop is installed as a three-node cluster. On such installations, all Hadoop jobs must be packaged as jars.

To create a jar for your Hadoop job, run the following command (after compiling the code):

```
$ jar cvf myJob.jar *.class
```

Alternatively, you can create a simple `manifest.txt` file with the content:

```
Main-Class: myJob
```

and create a jar file using the command:

```
$ jar cvf myJob.jar manifest.txt *.class
```

Running your Hadoop Jobs. Once you created the jar file, you can run it as follows.

```
$ hadoop jar myJob.jar myJob <hadoop job arguments>
```

The last argument of the command *before* `<hadoop job arguments>` is the Java class that contains the `public static int main()` method. If you created your jar file with a `manifest.txt` file as as discussed above, you can omit the Java class name:

```
$ hadoop jar myJob.jar <hadoop job arguments>
```

`<hadoop job arguments>` are any of the command-line arguments you need to pass to the `public static int main()` of your program. Often these are the locations of the input files and output directory.

Monitoring your job. Hadoop provides a web service allowing one to monitor the status of Hadoop jobs via a browser. Because cslvm31 does not have pinholes set, this is only possible if you are currently on campus, or running a campus VPN client on your machine.

The URL for the Hadoop monitor is

http://cslvm31.csc.calpoly.edu:8088/cluster

Third party Jars. For more complex Hadoop jobs you often may need third-party Jar files to be used with the `Mapper` and/or `Reducer` code. Because the `Mapper` and `Reducer` jobs run in separate JVMs on the cluster, their classpath environments are different than the classpath envoronment of the `public static int main()`. Fortunately, Hadoop allows us to pass third-party jars to the cluster JVMs using the `-libjars` option.

The full command to run a Hadoop job from myJob.jar file that relies on a third-party jar file `foo.jar` and another third party jar file `bar.jar`:

```
$ hadoop jar myJob.jar myJob -libjars foo.jar,bar.jar <hadoop job arguments>
```