<http://blog.imaginea.com/hadoop-a-short-guide/>

1.       Explain About HDFS Blocks

2.       HDFS Blocks and RAM memory consideration and different scenarios on block size and memory size

3.       Explain Job Tracker

4.       How you worked with Structured Data and Unstructured data. More explanation on Unstructured data how you consider writing Map-Reduce program from log files.

5.       How you handled unstructured data in Hive?

6.       How you validate unstructured data (data type) in Map-Reduce

7.       Explain about Partitions in Hadoop

8.       Explain partitions in Hive and you use them

9.       Explain debugging process in Hive

<http://yibingshi1977.wordpress.com/2012/12/27/debug-hive-in-eclipse/>

Sometimes it is necessary for us to debug Hive if we want to know exactly what is going on under the hood, especially when there is some problem. At the meanwhile, it is not very easy to setup a debug environment for Hive. I am going to provide some steps here for your reference.

1. Download and install hadoop.

Please download the version that your hive version works with. For example, I am currently using hive 0.7.1. It requires hadoop 0.20.x. You can download old version of hadoop from [here](http://archive.apache.org/dist/hadoop/core/). After the download is finished, please [configure it to run in pseudo mode](http://hadoop.apache.org/docs/r0.20.2/quickstart.html#PseudoDistributed).

2. Download and build hive

|  |  |
| --- | --- |
| 1  2 | $ svn co <http://svn.apache.org/repos/asf/hive/tags/release-0.7.1>  $ ant clean package eclipse-files |

Then in your eclipse, import the created hive project according to [this article](https://cwiki.apache.org/confluence/display/Hive/GettingStarted+EclipseSetup).

3. Configure your hive

The only thing need to be setup is the hadoop home directory. This is done by set the HADOOP\_HOME environment variable in hive-env.sh.

|  |  |
| --- | --- |
| 1 | HADOOP\_HOME=/directory/to/hadoop-0.20.1 |

Since everything in build directory gets lost if you run “ant clean”, I suggest you put your configure files in <HIVE\_HOME>/conf and then make symbolic links to them under <HIVE\_HOME>/build/dist/conf.

After this, starts up your hive and run some commands to make sure it works fine. Exit hive after that.

4. Modify your hive startup shell script to enable JVM remote debug in hadoop.

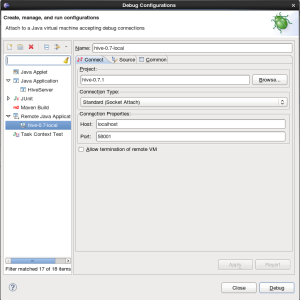
This is done by adding the following to <HIVE\_HOME>/build/dist/bin/hive (backup first):

|  |  |
| --- | --- |
| 1 | HADOOP\_OPTS="$HADOOP\_OPTS -Xdebug -Xrunjdwp:transport=dt\_socket,address=58001,server=y,suspend=n " |

After this, starts up hive again. This time, the JVM remote debugging should have been enabled. You can run netstat command to make sure that the port of 58001 (specified as parameter) is listened.

5. Connect your eclipse to hive to do debugging (look at [this article](http://javarevisited.blogspot.com.au/2011/02/how-to-setup-remote-debugging-in.html) for reference):

* Select your project in project explorer, select menu Run->Debug Configurations. Then add a new “Remote Java Application” according to below snapshot.

[](http://yibingshi1977.wordpress.com/2012/12/27/debug-hive-in-eclipse/eclipse/)

* Click “Debug” to connect eclipse to hive process.
* Set breakpoints in eclipse. Then run commands in hive shell. If any breakpoint is hit, hive process will be suspended and you can do any debug you want in eclipse

<http://javarevisited.blogspot.com.au/2011/02/how-to-setup-remote-debugging-in.html>

**How to remote debug Java application in Eclipse IDE**

* **Remote debugging** is not a new concept and many of you are aware of this just for who don’t know what is remote debugging? It’s a way of debugging any process could be **Java**or C++ running on some other location from your development machine.  Since [debugging Java application](http://javarevisited.blogspot.sg/2011/07/java-debugging-tutorial-example-tips.html)is essential part of Java development and ability to debug your application not only saves time but also increase productivity. *Local debugging* is the best way to debug Java program in my opinion and should always be preferred over remote debugging as discussed in my post [How to debug Java program in Eclipse](http://javarevisited.blogspot.sg/2011/07/java-debugging-tutorial-example-tips.html), but if local debugging is not possible and there is no way to debug your process then **remote debugging Java application**is the solution.  All Major IDE like NetBeans, Eclipse, IntelliJ allows you to remote debug Java program but I mostly use Eclipse for remote debugging Java application because it's free and standard IDE in Investment banks for Java projects. By the this is second article on Eclipse IDE in Javarevisited after sharing [Top 30 Eclipse keyboard shortcuts](http://javarevisited.blogspot.sg/2010/10/eclipse-tutorial-most-useful-eclipse.html) to improve productivity in previous post.
* **When use Java remote debugging in Eclipse**
* Many of us work on a Java project which runs on [Linux operating system](http://javarevisited.blogspot.sg/2011/03/unix-command-tutorial-working-fast-in.html) and we do development mostly on Windows using Eclipse IDE. Since I am working in Investment banking and finance domain I have seen use of Linux server for running electronic trading application quite a lot, which makes Java development difficult because you don't have [code](http://javarevisited.blogspot.sg/2011/09/code-review-checklist-best-practice.html)running on your development machine. Some time we managed to run the Java project in Eclipse IDE in windows itself which is essential for development and debugging purpose but many times its not possible due to various reason e.g. your project depends upon some of the [platform dependent library](http://javarevisited.blogspot.sg/2011/02/how-to-execute-native-shell-commands.html)or some Linux module whose windows version may not be available or your project is too big to run on windows and its heavily connected to upstream and downstream system and it's almost impossible to create same environment in your windows machine for development. On such situation my approach to work is isolate the work I am doing and test that with the help of [mock objects](http://javarevisited.blogspot.sg/2012/08/best-practices-to-write-junit-test.html), [Threads](http://javarevisited.blogspot.sg/2011/02/how-to-implement-thread-in-java.html)or by trying to run that module independently but this is also not a desired solution in some cases where you need to debug the Java project at run time to find out some subtle issues, This is the time to use **Eclipse for remote debugging Java application.**
* [Java Remote debugging in Eclpse IDE](http://javarevisited.blogspot.com/2011/02/how-to-setup-remote-debugging-in.html)
* **Eclipse**IDE provides us most useful feature called **"Remote debugging"** by using which you can debug Java application running on remote [Linux or Windows Server](http://javarevisited.blogspot.sg/2011/09/find-hostname-from-ip-address-to.html) from your windows machine. believe me this become absolutely necessary in some condition and knowing how to setup remote debugging and working of remote debugging in eclipse can greatly improve your productivity. In this[Eclipse tutorial](http://javarevisited.blogspot.sg/2010/10/eclipse-tutorial-most-useful-eclipse.html) I will try to explain **eclipse remote debugging**or**how to setup remote debugging in eclipse.**
* Now let's see how we can *setup remote debugging in Eclipse*:
* 1) First setup your Java project in Eclipse.
* 2) Select your project, go to "Run" Menu option and select **"Debug Configurations**"

|  |
| --- |
| [remote debugging eclipse](https://lh3.googleusercontent.com/-iWoAgN9lqQc/TWh5Gh4oMeI/AAAAAAAAACc/4H8QmFzsvqg/s1600/remote_Debugging_1.PNG) |
| Remote debugging with eclipse 1 |

* 3) This will open **Debug Configuration** window select **"Remote Java Application"** icon on left side, Right click and say "New".

|  |
| --- |
| [How to do  remote debugging in eclipse](https://lh6.googleusercontent.com/-A81IBEHKfmg/TWh5GigXdcI/AAAAAAAAACg/GMYbSQj404g/s1600/remote_Debugging_2.PNG) |
| eclipse remote debugging 2 |

* 4) After clicking on New, Eclipse will create Remote Java Application configuration for your selected project. Now next step is to setup [host and port](http://javarevisited.blogspot.sg/2010/10/basic-networking-commands-in-linuxunix.html)for **remote debugging** in Eclipse.

|  |
| --- |
| [How to setup Java remote debugging in Eclipse](https://lh5.googleusercontent.com/-27BHWq1KSF0/TWh5Goiew1I/AAAAAAAAACY/2IgVQE1m2-s/s1600/remote_Debugging_3.PNG) |
| Remote debugging in Eclipse 3 |

* 5) Now put the host name and port on which your process is listening for debugger in Linux machine. Check the **"Allow termination of remote VM"** check box if you would like to ***close java application running on Linux*** from eclipse.
* 6) Now you are all set to **remote debug** your Java project from Eclipse IDE but before starting to debug make sure your Java process is started with [Java remote debug settings](http://javarevisited.blogspot.sg/2011/11/hotspot-jvm-options-java-examples.html) and listening on same host and port, otherwise eclipse will not able to connect successfully.
* 7) To debug just click the "**Debug"** button in last screen where we have **setup host and port**.
* 8) You can also debug by going to "**Debug Configurations**" selecting your project in "**Remote Java Application”** and clicking on **"DEBUG"**.
* **Java remote debug setting and JVM debug options**
* In order to remote debug a Java application from Eclipse, that application must be started with following [JVM debug options](http://javarevisited.blogspot.sg/2011/11/hotspot-jvm-options-java-examples.html):
* **java -Xdebug -Xrunjdwp:transport=dt\_socket,address=8001,server=y  suspend=y -jar stockTradingGUI.jar**
* This will start java application stockTradingGUI into debug mode using **Java Debug Wire Protocol (jdwp) protocol** and it will listen on port 8001 suspend=y will ensure that that application will not start running until Eclipse connect it on speicified debug port.It also important to note that application must be start before Eclipse tries to connect it other wise Eclipse will throw error "**Failed to connect to remote VM. Connection refused**" or "Connection refused: connect"
* Enjoy remote debugging in Eclipse J
* **How to fix "Failed to connect to remote VM. Connection refused"**
* "Failed to connect to remote VM. Connection refused"  is the most common error [Java programmer](http://javarevisited.blogspot.sg/2011/06/top-programming-interview-questions.html) face while trying to remote debug Java application, there can be multiple reason why you are getting "Failed to connect to remote VM. Connection refused" error, here are some of the reason  this error occurs in Eclipse :
* 1) Network or Firewall issue
* Since Eclipse uses TCP socket connection to connect to remote machine, remote host must be accessible from your network and corresponding remote debug port must be open and ready to accept connection. You can check this by using [networking commands in windows and Linux](http://javarevisited.blogspot.sg/2010/10/basic-networking-commands-in-linuxunix.html) like ping or telnet.
* 2) Java application not running on remote host
* This is probably most common reason of "**Failed to connect to remote VM. Connection refused**". If [Java application is not running](http://javarevisited.blogspot.sg/2011/11/run-java-program-from-command-prompt.html) there will no one to accept connection. Just restart the Java application in debug mode.
* 3) Incorrect host and port combination
* One of the human error which causes "Failed to connect to remote VM. Connection refused". Some time typo on [hostname](http://javarevisited.blogspot.sg/2011/11/run-java-program-from-command-prompt.html)or port causes Eclipse trying to connect different servers or different port, which is obvious reason you get this error.
* 4) Missing JVM debug settings
* In order to accept remote connection from Eclipse your Java application must be running with c[orrect debug settings based upon your JVM](http://javarevisited.blogspot.sg/2011/11/hotspot-jvm-options-java-examples.html). If you are JVM is not running on debug mode you will get "Failed to connect to remote VM. Connection refused"
* **Java Remote debugging Tips  in Eclipse IDE**
* Here are few tips which is very useful while remote debugging any [Java program into Eclipse IDE](http://javarevisited.blogspot.sg/2011/11/run-java-program-from-command-prompt.html), this not only help you to avoid some common errors in Eclipse but also makes remote debugging much easier :
* **Tip:** In **JVM DEBUG parameters** there is a parameter called "suspend" which takes value as "y" or "n". so if you want to debug the process from start set this parameter as  "**suspend=y**" and your Java application will wait until Eclipse remotely connect to it. Otherwise if you want to run your program and later want eclipse to be connected that set this as "**suspend=n**" so your java application will run normally and after eclipse remotely connected to it, it will stop on [breakpoints](http://www.youtube.com/watch?v=p1cqCjwGOtg&feature=plcp).
* **Tip:** Use start up script to put JVM debug parameter and use a variable e.g. isDebugEnabled and also REMOTE\_DEBUG\_PORT in the [shell script](http://javarevisited.blogspot.sg/2011/06/special-bash-parameters-in-script-linux.html)and export this variable when you want to remote debug your Java application. This will be very handy and will require just one time setup work.
* **Tip:** if you get error **"Failed to connect to remote VM. Connection refused**" or **"Connection refused:** **connect"** then there might be two possibility one your java program is not running on remote host and other you are giving incorrect port or host name after verifying these two things if issue still persists then try giving full name of the host.
* **Tip**: You also need to ensure that you run the same code base in eclipse which is deployed in your remote machine so that what you debug and see in **eclipse**is true and real. you also need to ensure that your code is compile with **debug option "-g"** so that **eclipse**can easily gather debug info e.g. information about local variable. by default java only generate  line numbers and source file information.with  **debug option -g your class file size might be more** because it would contain some debug information.
* Note: Recently I have wrote another article [**10 tips on debugging Java Program in eclipse**](http://javarevisited.blogspot.com/2011/07/java-debugging-tutorial-example-tips.html) which is a collection of my java debugging tips and explains some advanced java debugging concept like conditional break point, how to debug multi-threaded programs in Java, Step filtering to avoid debugging system classes in Java, logical view to see the content of collection classes like [HashMap](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html)or [ArrayList in Java](http://javarevisited.blogspot.sg/2011/09/difference-vector-vs-arraylist-in-java.html).
* Read more: <http://javarevisited.blogspot.com/2011/02/how-to-setup-remote-debugging-in.html#ixzz2e5WCujeG>

## Some Hadoop and Hive Gotchas and Developer Tips

May 21, 2012 by [Buddhika Chamith](http://chamibuddhika.wordpress.com/author/chamibuddhika/" \o "Posts by Buddhika Chamith)

This is my log on several mistakes (some pretty dumb on the hindsight :) ) that I did while getting started with Hadoop and Hive some time back, along with some tricks on debugging Hadoop and Hive. I am using Hadoop 0.20.203 and Hive 0.8.1.

#### localhost: Error: JAVA\_HOME is not set

This almost undecipherable and cryptic error message :) during Hadoop startup (namenode/jobtracker etc.) says Hadoop cannot find the Java installation. Wait!! I have already set JAVA\_HOME enviornment variable?? Seems it’s not enough. So where else to set it? Turns out that you have to set JAVA\_HOME in hadoop-env.sh present in conf folder to get the elephant moving.

#### Name node mysteriously fails to start

When you start the namenode things seems fine except for the fact that the server is not up and running. And of course I hadn’t formatted the HDFS on the namenode. So why should it work right? :) So there goes. Format the darn namenode before doing anything distributed with Hadoop.

|  |  |
| --- | --- |
| 1 | bin/hadoop namenode -format |

#### java.io.IOException Call to localhost/127.0.0.1:9000 failed on local exception: java.io.EOFException

This one was bit tricky. After fiddling and struggling for some time found out that Hadoop dependency version used in the JobClient in order to communicate with JobTracker is different from the version that’s present inside the running Hadoop instance. Hadoop uses a homegrown RPC mechanism to communicate with job tracker and name nodes. And it seems certain different Hadoop versions have incompatibilities in this interface.

Now it’s time for some debugging tips.

#### Debugging Hadoop Local (Standalone) mode

Add debugging options for JVM as follows in conf/hadoop-env.sh.

|  |  |
| --- | --- |
| 1 | export HADOOP\_OPTS="-agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=[DEBUG\_PORT]" |

#### Debugging Hive Server

Start Hive with following command line to remote debug Hive.

|  |  |
| --- | --- |
| 1 | ./hive --service hiveserver --debug[port=[DEBUG\_PORT],mainSuspend=y,childSuspend=y] |

10.   Different Joins in Hive

11.   Difference between right join and outer join in Hive

12.   Difference between left semi join and inner join

13.   How you done serialization in Hadoop

Can you please walk me through the following scenario how it is done in

Hadoop Map Reduce and what happens internally?  
  
Input File: Employee\_Sal.csv (File attached)  
O/P:  
Sum(Salary)  
Average(Salary)  
Count(ID)  
Avg\_Sal\_Month = Average(Salary/12)  
  
First row in the file has column names.

2.How to avoid shuffling in the mapping o/p? When I try to print integers 1 to 10 in ascending order, map o/p is printing in some weird order, why?

3.   In the mapping output of reverse string, o/p is pair. How to get just the ‘key’ value?

a.       Example:

Input File:

                                    rama.txt

Output File:

                                  revstrop/par\*

niassuh rim     0

-- How to avoid this 0 in the part-0000 file??

-- We have not used Reducer class here?

3. Where is this map reduce programming used in real time and when sqoop/hive/pig are used? Which one to choose when?

4.Like what was the source? How many files? How did you load into HDFS? Design and development, what requirements? How did you deliver? Etc…

1.      Hadoop API –

**Is it same what we are using? Mapper Class, Reducer Class and Configuration class**

2.      What is Streaming API? Have you used any of that?

3.      What is Driver Code?

**– Is it same as Configuration class?**

4.      Input file Format types – PDF? Zipped File? not splittable file format?

5.      When to use Partitioner? And how to determine the optimal number of Reducers for a job?

6.      what is a secondary sort? How to snaffle work in mapreduce?

7.      How to Indexing data?

8.      Can you join datasets using Map Reduce jobs? Map Side Join? Reduce-Side join?

9.      Is there any intermediate file generated after mapper (internal)? Where is it created local or hdfs? How can we see it?

10.  What debug tool did you use?

SCENARIO:

iam developing a Hadoop app

I have a lot of files composed by data about daily global weather (**like the \*.op attached file**) and I need to extract some information of these files.

I've achieved the maximum temperature of a year based on the Hadoop Definitive Guide, 2nd edition.

Now I have to sum the occurences of all phenomena in each year by station (the last 6 fields of each line indicates the occurence or not of the phenomena).

The description of the data format is in the GSOD\_DESC.txt.

CAN  U SOLVE ...

 I loaded the data into a hive table and I am able to query max temp, max dewp etc. I had to convert the data file to csv and remove the header information.

Here is the table I created

CREATE TABLE weatherdata (STN INT, WBAN INT, YEARMODA INT, TEMP FLOAT, TEMPDUR INT, DEWP FLOAT, DEWPDUR INT, SLP FLOAT, SLPDUR INT, STP FLOAT, STPDUR INT, VSIB FLOAT, VSIBDUR INT, WDSP FLOAT, WDSPDUR INT, MXSPD FLOAT, GUST FLOAT, MAX FLOAT, MIN FLOAT, PRCP FLOAT, SNDP FLOAT, FRSHTT INT)row format delimited fields terminated by ',' escaped by '\\';

This is how I loaded the CSV file into the table

load data inpath 'WEATHER/WeatherData.csv' into table weatherdata;

This are some queries I am able to run

select max(temp) from weatherdata;

select max(dewp) from weatherdata;

Let me know what else I need to do. Thanks a lot for this project. I am feeling very comfortable with hive now.

<http://www.javacodegeeks.com/2012/07/quartz-2-scheduler-example.html>

# Map Side and Reduce Side Joins

**Joins:-**  
=======

Joins is one of the interesting features available in MapReduce. Joins performed by Mapper are called as Map-side Joins. Joins performed by Reducer can be treated as Reduce-side joins. Frameworks like Pig, Hive, or Cascading has support for performing joins.

Before diving into the implementation let us understand the problem throughly. If we have two datasets, for example, one dataset having user ids, names and the other having the user activity over the application. In-order to find out which user have performed what activity on the application we might need to join these two datasets such as both user names and the user activity will be joined together.

Join can be applied based on the dataset size if one dataset is very small to be distributed across the cluster then we can use Side Data Distribution technique.

**Side Data Distribution:-**  
====================  
Side-Data is the additional data needed by the job to process the main dataset. The critical part is to make this side-data available to all the map or reduce tasks running in the cluster. It is possible to cache the side-data in memory in a static field, so that the tasks running successively in a task tracker will share the data. Using the Task JVM re-use feature we can handle this. When using this feature we should make sure that the amount of memory needed to cache the data should not affect the memory needed for Shuffle and sort phase.

Caching of side data can be done in two ways,

Job Configuration:- Using Job Configuration object setter method we can set the key-value pairs and the same can be retrieved in the map or reduce tasks. We should be careful using this option to not to use huge amount of data to be shared in this way since the configuration is read by the JobTracker, TaskTracker and the child JVMs and everytime the configurations will be loaded into the memory.

**Distributed Cache:-**  
=================  
Side-Data can be shared using the Hadoop’s Distributed cache mechanism. We can copy files and archives to the task nodes when the tasks need to run. Usually this is the preferrable way over the JobConfigurtion.

If both the datasets are too large then we cannot copy either of the datasets to each node in the cluster as we did in the Side data distribution.  
We can still join the records using MapReduce with a Map-side or reduce-side joins.

**Map-Side Joins:-**  
===============  
The inputs for to each map must be partitioned and sorted in a specific way. Each input dataset must be divided into the same number of partitions, and it must be sorted by the same key (the join key) in each source.  All the records for a particular key must reside in the same partition and which is mandatory. A map-side join can be used to join the outputs of several jobs that had the same number of reducers, the same keys, and output files that are not splittable which means  
the ouput files should not be bigger than the HDFS block size.

Using the org.apache.hadoop.mapred.join.CompositeInputFormat class we can achieve this. The join type (Inner or Outer) is configurable using the join expression. for ex,  
func ::= tbl(<class>,”<path>”);  
We can set it to the CompositeInputFormat using,

inner(tbl(org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat.class,  
“hdfs://localhost:8000/usr/data”),  
tbl(org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat.class,  
“hdfs://localhost:8000/usr/activity”));

We can achieve following kind of joins using Map-Side techniques,

1) Inner Join  
2) Outer Join  
3) Override – MultiFilter for a given key, prefered values from the right most source

**Reduce-Side Joins:-**  
=================  
Reduce-Side joins are more simple than Map-Side joins since the input datasets need not to be structured. But it is less efficient as both datasetts have to go through the MapReduce shuffle phase. the records with the same key are brought together in the reducer. We can also use the Secondary Sort technique to control the order of the records.

- See more at: http://bigdatapartnership.com/map-side-and-reduce-side-joins/#sthash.3LeUzri5.dpuf

**DistributeCache**

public class **DistributedCache**

extends [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true)

Distribute application-specific large, read-only files efficiently.

DistributedCache is a facility provided by the Map-Reduce framework to cache files (text, archives, jars etc.) needed by applications.

Applications specify the files, via urls (hdfs:// or http://) to be cached via the [JobConf](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/JobConf.html" \o "class in org.apache.hadoop.mapred). The DistributedCache assumes that the files specified via hdfs:// urls are already present on the [FileSystem](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/fs/FileSystem.html" \o "class in org.apache.hadoop.fs) at the path specified by the url.

The framework will copy the necessary files on to the slave node before any tasks for the job are executed on that node. Its efficiency stems from the fact that the files are only copied once per job and the ability to cache archives which are un-archived on the slaves.

DistributedCache can be used to distribute simple, read-only data/text files and/or more complex types such as archives, jars etc. Archives (zip, tar and tgz/tar.gz files) are un-archived at the slave nodes. Jars may be optionally added to the classpath of the tasks, a rudimentary software distribution mechanism. Files have execution permissions. Optionally users can also direct it to symlink the distributed cache file(s) into the working directory of the task.

DistributedCache tracks modification timestamps of the cache files. Clearly the cache files should not be modified by the application or externally while the job is executing.

Here is an illustrative example on how to use the DistributedCache:

// Setting up the cache for the application

1. Copy the requisite files to the FileSystem:

$ bin/hadoop fs -copyFromLocal lookup.dat /myapp/lookup.dat

$ bin/hadoop fs -copyFromLocal map.zip /myapp/map.zip

$ bin/hadoop fs -copyFromLocal mylib.jar /myapp/mylib.jar

$ bin/hadoop fs -copyFromLocal mytar.tar /myapp/mytar.tar

$ bin/hadoop fs -copyFromLocal mytgz.tgz /myapp/mytgz.tgz

$ bin/hadoop fs -copyFromLocal mytargz.tar.gz /myapp/mytargz.tar.gz

2. Setup the application's JobConf:

JobConf job = new JobConf();

DistributedCache.addCacheFile(new URI("/myapp/lookup.dat#lookup.dat"),

job);

DistributedCache.addCacheArchive(new URI("/myapp/map.zip", job);

DistributedCache.addFileToClassPath(new Path("/myapp/mylib.jar"), job);

DistributedCache.addCacheArchive(new URI("/myapp/mytar.tar", job);

DistributedCache.addCacheArchive(new URI("/myapp/mytgz.tgz", job);

DistributedCache.addCacheArchive(new URI("/myapp/mytargz.tar.gz", job);

3. Use the cached files in the [Mapper](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/Mapper.html)

or [Reducer](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/Reducer.html):

public static class MapClass extends MapReduceBase

implements Mapper<K, V, K, V> {

private Path[] localArchives;

private Path[] localFiles;

public void configure(JobConf job) {

// Get the cached archives/files

localArchives = DistributedCache.getLocalCacheArchives(job);

localFiles = DistributedCache.getLocalCacheFiles(job);

}

public void map(K key, V value,

OutputCollector<K, V> output, Reporter reporter)

throws IOException {

// Use data from the cached archives/files here

// ...

// ...

output.collect(k, v);

}

}

It is also very common to use the DistributedCache by using [GenericOptionsParser](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/util/GenericOptionsParser.html" \o "class in org.apache.hadoop.util). This class includes methods that should be used by users (specifically those mentioned in the example above, as well as [addArchiveToClassPath(Path, Configuration)](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/filecache/DistributedCache.html" \l "addArchiveToClassPath(org.apache.hadoop.fs.Path, org.apache.hadoop.conf.Configuration))), as well as methods intended for use by the MapReduce framework (e.g., [JobClient](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/JobClient.html" \o "class in org.apache.hadoop.mapred)). For implementation details, see [TrackerDistributedCacheManager](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/filecache/TrackerDistributedCacheManager.html) and [TaskDistributedCacheManager](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/filecache/TaskDistributedCacheManager.html" \o "class in org.apache.hadoop.filecache).

Using Distributed Cache in Hadoop (Hadoop 0.20.2)

Quite often we encounter situation when we need certain files like configuration files, jar libraries, xml files, properties files etc to be present in Hadoops processing nodes at the time of its execution. Quite understandably Hadoop has a feature called Distributed Cache which helps in sending those readonly files to the task nodes. In Hadoop environment jobs are basically map-Reduce jobs and the necessary readonly files are copied to the tasktracker nodes at the beginning of job execution process. The default size of distributed cache in Hadoop is about 10 GB but We can control the size of the distributed cache by explicitly defining its size in hadoop’s configuration file local.cache.size.

Thus, Distributed cache is a mechanism to caching readonly data over Hadoop cluster. The sending of readOnly files occurs at the time of job creation and the framework makes the cached files available to the cluster nodes at their computational time.

The following distributed cache java program sends the necessary xml files to the task executing nodes prior to job execution.

Java Program/Tutorial of Distributed Cache usage in Hadoop

Hadoop Version : 0.20.2

Java Version: Java-SE-1.6

The program below consists of two classes. The DcacheMapper class and the parent Class. Job is initialized in the base class. Job is initialized pointing to the location in HDFS where the file to be sent to all nodes is present. When the setup method in parent class is executed we can retrieve the distributed configuration file and read it for our usage.

API doc for Distributed Cache can be found at the given URL.

<http://hadoop.apache.org/common/docs/r0.20.2/api/org/apache/hadoop/filecache/DistributedCache.html>

Scheduling in Hadoop

Hadoop is a general-purpose system that enables high-performance processing of data over a set of distributed nodes. But within this definition is the fact that Hadoop is a multi-tasking system that can process multiple data sets for multiple jobs for multiple users at the same time. This capability of multi-processing means that Hadoop has the opportunity to more optimally map jobs to resources in a way that optimizes their use.

Up until 2008, Hadoop supported a single scheduler that was intermixed with the JobTracker logic. Although this implementation was perfect for the traditional batch jobs of Hadoop (such as log mining and Web indexing), the implementation was inflexible and could not be tailored. Further, Hadoop operated in a batch mode, where jobs were submitted to a queue, and the Hadoop infrastructure simply executed them in the order of receipt.

Luckily, a bug report (HADOOP-3412) was submitted for an implementation of a scheduler that was independent of the JobTracker. More importantly, the new scheduler is pluggable, which allows use of new scheduling algorithms to help optimize jobs that have specific characteristics. A further advantage to this change is the increased readability of the scheduler, which has opened it up to greater experimentation and the potential for a growing list of schedulers to specialize in Hadoop's ever-increasing list of applications.

With this change, Hadoop is now a multi-user data warehouse that supports a variety of different types of processing jobs, with a pluggable scheduler framework providing greater control. This framework allows optimal use of a Hadoop cluster over a varied set of workloads (from small jobs to large jobs and everything in between). Moving away from FIFO scheduling (which treats a job's importance relative to when it was submitted) allows a Hadoop cluster to support a variety of workloads with varying priority and performance constraints.

Note: This article assumes some knowledge of Hadoop. See Resources for links to an introduction to the Hadoop architecture and the practical Hadoop series for installing, configuring, and writing Hadoop applications.

The core Hadoop architecture

A Hadoop cluster consists of a relatively simple architecture of masters and slaves (see Figure 1). The NameNode is the overall master of a Hadoop cluster and is responsible for the file system namespace and access control for clients. There also exists a JobTracker, whose job is to distribute jobs to waiting nodes. These two entities (NameNode and JobTracker) are the masters of the Hadoop architecture. The slaves consist of the TaskTracker, which manages the job execution (including starting and monitoring jobs, capturing their output, and notifying the JobTracker of job completion). The DataNode is the storage node in a Hadoop cluster and represents the distributed file system (or at least a portion of it for multiple DataNodes). The TaskTracker and the DataNode are the slaves within the Hadoop cluster.

Figure 1. Elements of a Hadoop cluster

Diagram showing all the elements of a Hadoop cluster

Note that Hadoop is flexible, supporting a single node cluster (where all entities exist on a single node) or a multi-node cluster (where JobTracker and NameNodes are distributed across thousands of nodes). Although little information exists on the larger production environments that exist, the largest known Hadoop cluster is Facebook's, which consists of 4000 nodes. These nodes are split among several sizes (half include 8- and 16-core CPUs). The Facebook cluster also supports 21PB of storage distributed across the many DataNodes. Given the large number of resources and the potential for many jobs from numerous users, scheduling is an important optimization going forward.

Back to top

Hadoop schedulers

Since the pluggable scheduler was implemented, several scheduler algorithms have been developed for it. The sections that follow explore the various algorithms available and when it makes sense to use them.

FIFO scheduler

The original scheduling algorithm that was integrated within the JobTracker was called FIFO. In FIFO scheduling, a JobTracker pulled jobs from a work queue, oldest job first. This schedule had no concept of the priority or size of the job, but the approach was simple to implement and efficient.

Fair scheduler

The core idea behind the fair share scheduler was to assign resources to jobs such that on average over time, each job gets an equal share of the available resources. The result is that jobs that require less time are able to access the CPU and finish intermixed with the execution of jobs that require more time to execute. This behavior allows for some interactivity among Hadoop jobs and permits greater responsiveness of the Hadoop cluster to the variety of job types submitted. The fair scheduler was developed by Facebook.

The Hadoop implementation creates a set of pools into which jobs are placed for selection by the scheduler. Each pool can be assigned a set of shares to balance resources across jobs in pools (more shares equals greater resources from which jobs are executed). By default, all pools have equal shares, but configuration is possible to provide more or fewer shares depending upon the job type. The number of jobs active at one time can also be constrained, if desired, to minimize congestion and allow work to finish in a timely manner.

To ensure fairness, each user is assigned to a pool. In this way, if one user submits many jobs, he or she can receive the same share of cluster resources as all other users (independent of the work they have submitted). Regardless of the shares assigned to pools, if the system is not loaded, jobs receive the shares that would otherwise go unused (split among the available jobs).

The scheduler implementation keeps track of the compute time for each job in the system. Periodically, the scheduler inspects jobs to compute the difference between the compute time the job received and the time it should have received in an ideal scheduler. The result determines the deficit for the task. The job of the scheduler is then to ensure that the task with the highest deficit is scheduled next.

You configure fair share in the mapred-site.xml file. This file defines the properties that collectively govern the behavior of the fair share scheduler. An XML file—referred to with the property mapred.fairscheduler.allocation.file—defines the allocation of shares to each pool. To optimize for job size, you can set the mapread.fairscheduler.sizebasedweight to assign shares to jobs as a function of their size. A similar property allows smaller jobs to finish faster by adjusting the weight of the job after 5 minutes (mapred.fairscheduler.weightadjuster). Numerous other properties exist that you can use to tune loads over the nodes (such as the number of maps and reduces that a given TaskTracker can manage) and define whether preemption should be performed. See Resources for a link to a full list of configurable properties.

Capacity scheduler

The capacity scheduler shares some of the principles of the fair scheduler but has distinct differences, too. First, capacity scheduling was defined for large clusters, which may have multiple, independent consumers and target applications. For this reason, capacity scheduling provides greater control as well as the ability to provide a minimum capacity guarantee and share excess capacity among users. The capacity scheduler was developed by Yahoo!.

In capacity scheduling, instead of pools, several queues are created, each with a configurable number of map and reduce slots. Each queue is also assigned a guaranteed capacity (where the overall capacity of the cluster is the sum of each queue's capacity).

Queues are monitored; if a queue is not consuming its allocated capacity, this excess capacity can be temporarily allocated to other queues. Given that queues can represent a person or larger organization, any available capacity is redistributed for use by other users.

Another difference of fair scheduling is the ability to prioritize jobs within a queue. Generally, jobs with a higher priority have access to resources sooner than lower-priority jobs. The Hadoop road map includes a desire to support preemption (where a low-priority job could be temporarily swapped out to allow a higher-priority job to execute), but this functionality has not yet been implemented.

Another difference is the presence of strict access controls on queues (given that queues are tied to a person or organization). These access controls are defined on a per-queue basis. They restrict the ability to submit jobs to queues and the ability to view and modify jobs in queues.

You configure the capacity scheduler within multiple Hadoop configuration files. The queues are defined within hadoop-site.xml, and the queue configurations are set in capacity-scheduler.xml. You can configure ACLs within mapred-queue-acls.xml. Individual queue properties include capacity percentage (where the capacity of all queues in the cluster is less than or equal to 100), the maximum capacity (limit for a queue's use of excess capacity), and whether the queue supports priorities. Most importantly, these queue properties can be manipulated at run time, allowing them to change and avoid disruptions in cluster use.

Other approaches

Although not a scheduler per se, Hadoop also supports the idea of provisioning virtual clusters from within larger physical clusters, called Hadoop On Demand (HOD). The HOD approach uses the Torque resource manager for node allocation based on the needs of the virtual cluster. With allocated nodes, the HOD system automatically prepares configuration files, and then initializes the system based on the nodes within the virtual cluster. Once initialized, the HOD virtual cluster can be used in a relatively independent way.

HOD is also adaptive in that it can shrink when the workload changes. HOD automatically de-allocates nodes from the virtual cluster after it detects no running jobs for a given time period. This behavior permits the most efficient use of the overall physical cluster assets.

HOD is an interesting model for deployments of Hadoop clusters within a cloud infrastructure. It offers an advantage in that with less sharing of the nodes, there is greater security and, in some cases, improved performance because of a lack of contention within the nodes for multiple users' jobs.

Back to top

When to use each scheduler

From the discussion above, you can see where these scheduling algorithms are targeted. If you're running a large Hadoop cluster, with multiple clients and different types and priorities of jobs, then the capacity scheduler is the right choice to ensure guaranteed access with the potential to reuse unused capacity and prioritize jobs within queues.

Although less complex, the fair scheduler works well when both small and large clusters are used by the same organization with a limited number of workloads. Fair scheduling still provides the means to non-uniformly distribute capacity to pools (of jobs) but in a simpler and less configurable way. The fair scheduler is useful in the presence of diverse jobs, because it can provide fast response times for small jobs mixed with larger jobs (supporting more interactive use models).

Back to top

Future developments in Hadoop scheduling

Now that the Hadoop scheduler is pluggable, you should see new schedulers developed for unique cluster deployments. Two in-process schedulers (from the Hadoop issues list) include the adaptive scheduler and the learning scheduler. The learning scheduler (MAPREDUCE-1349) is designed to maintain a level of utilization when presented with a diverse set of workloads. Currently, this scheduler implementation focuses on CPU load averages, but utilization of network and disk I/O is planned. The adaptive scheduler (MAPREDUCE-1380) focuses on adaptively adjusting resources for a given job based on its performance and user-defined business goals.

Back to top

Conclusion

The introduction of the pluggable scheduler was yet another evolution in cluster computing with Hadoop. The pluggable scheduler permits the use (and development) of schedulers optimized for the particular workload and application. The new schedulers have also made it possible to create multi-user data warehouses with Hadoop, given the ability to share the overall Hadoop infrastructure with multiple users and organizations.

Hadoop is evolving as its use models evolve and now supports new types of workloads and usage scenarios (such as multi-user or multi-organization big data warehouses). The new flexibility that Hadoop provides is a great step toward more optimized use of cluster resources in big data analytics.

### Specifying Configuration Variables with the -D Option

You can specify additional configuration variables by using "-D <property>=<value>".

#### Specifying Directories

To change the local temp directory use:

-D dfs.data.dir=/tmp

To specify additional local temp directories use:

-D mapred.local.dir=/tmp/local

-D mapred.system.dir=/tmp/system

-D mapred.temp.dir=/tmp/temp

**Note:** For more details on jobconf parameters see: [mapred-default.html](http://hadoop.apache.org/core/docs/current/mapred-default.html)

#### Specifying Map-Only Jobs

Often, you may want to process input data using a map function only. To do this, simply set mapred.reduce.tasks to zero. The Map/Reduce framework will not create any reducer tasks. Rather, the outputs of the mapper tasks will be the final output of the job.

-D mapred.reduce.tasks=0

To be backward compatible, Hadoop Streaming also supports the "-reduce NONE" option, which is equivalent to "-D mapred.reduce.tasks=0".

#### Specifying the Number of Reducers

To specify the number of reducers, for example two, use:

$HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/hadoop-streaming.jar \

-D mapred.reduce.tasks=2 \

-input myInputDirs \

-output myOutputDir \

-mapper org.apache.hadoop.mapred.lib.IdentityMapper \

-reducer /bin/wc

<http://hadoop.apache.org/docs/stable/streaming.html#Hadoop+Streaming>

### Hadoop Partitioner Class

Hadoop has a library class, [KeyFieldBasedPartitioner](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/lib/KeyFieldBasedPartitioner.html), p> that is useful for many applications. This class allows the Map/Reduce framework to partition the map outputs based on certain key fields, not the whole keys. For example:

$HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/hadoop-streaming.jar \

-D stream.map.output.field.separator=. \

-D stream.num.map.output.key.fields=4 \

-D map.output.key.field.separator=. \

-D mapred.text.key.partitioner.options=-k1,2 \

-D mapred.reduce.tasks=12 \

-input myInputDirs \

-output myOutputDir \

-mapper org.apache.hadoop.mapred.lib.IdentityMapper \

-reducer org.apache.hadoop.mapred.lib.IdentityReducer \

-partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner

Here, -D stream.map.output.field.separator=. and -D stream.num.map.output.key.fields=4 are as explained in previous example. The two variables are used by streaming to identify the key/value pair of mapper.

The map output keys of the above Map/Reduce job normally have four fields separated by ".". However, the Map/Reduce framework will partition the map outputs by the first two fields of the keys using the -D mapred.text.key.partitioner.options=-k1,2 option. Here, -D map.output.key.field.separator=. specifies the separator for the partition. This guarantees that all the key/value pairs with the same first two fields in the keys will be partitioned into the same reducer.

This is effectively equivalent to specifying the first two fields as the primary key and the next two fields as the secondary. The primary key is used for partitioning, and the combination of the primary and secondary keys is used for sorting. A simple illustration is shown here:

Output of map (the keys)

11.12.1.2

11.14.2.3

11.11.4.1

11.12.1.1

11.14.2.2

Partition into 3 reducers (the first 2 fields are used as keys for partition)

11.11.4.1

-----------

11.12.1.2

11.12.1.1

-----------

11.14.2.3

11.14.2.2

Sorting within each partition for the reducer(all 4 fields used for sorting)

11.11.4.1

-----------

11.12.1.1

11.12.1.2

-----------

11.14.2.2

11.14.2.3

### Hadoop Comparator Class

Hadoop has a library class, [KeyFieldBasedComparator](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/lib/KeyFieldBasedComparator.html), that is useful for many applications. This class provides a subset of features provided by the Unix/GNU Sort. For example:

$HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/hadoop-streaming.jar \

-D mapred.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedComparator \

-D stream.map.output.field.separator=. \

-D stream.num.map.output.key.fields=4 \

-D map.output.key.field.separator=. \

-D mapred.text.key.comparator.options=-k2,2nr \

-D mapred.reduce.tasks=12 \

-input myInputDirs \

-output myOutputDir \

-mapper org.apache.hadoop.mapred.lib.IdentityMapper \

-reducer org.apache.hadoop.mapred.lib.IdentityReducer

The map output keys of the above Map/Reduce job normally have four fields separated by ".". However, the Map/Reduce framework will sort the outputs by the second field of the keys using the -D mapred.text.key.comparator.options=-k2,2nr option. Here, -n specifies that the sorting is numerical sorting and -r specifies that the result should be reversed. A simple illustration is shown below:

Output of map (the keys)

11.12.1.2

11.14.2.3

11.11.4.1

11.12.1.1

11.14.2.2

Sorting output for the reducer(where second field used for sorting)

11.14.2.3

11.14.2.2

11.12.1.2

11.12.1.1

11.11.4.1

### Hadoop Aggregate Package

Hadoop has a library package called [Aggregate](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/lib/aggregate/package-summary.html). Aggregate provides a special reducer class and a special combiner class, and a list of simple aggregators that perform aggregations such as "sum", "max", "min" and so on over a sequence of values. Aggregate allows you to define a mapper plugin class that is expected to generate "aggregatable items" for each input key/value pair of the mappers. The combiner/reducer will aggregate those aggregatable items by invoking the appropriate aggregators.

To use Aggregate, simply specify "-reducer aggregate":

$HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/hadoop-streaming.jar \

-D mapred.reduce.tasks=12 \

-input myInputDirs \

-output myOutputDir \

-mapper myAggregatorForKeyCount.py \

-reducer aggregate \

-file myAggregatorForKeyCount.py \

The python program myAggregatorForKeyCount.py looks like:

#!/usr/bin/python

import sys;

def generateLongCountToken(id):

return "LongValueSum:" + id + "\t" + "1"

def main(argv):

line = sys.stdin.readline();

try:

while line:

line = line[:-1];

fields = line.split("\t");

print generateLongCountToken(fields[0]);

line = sys.stdin.readline();

except "end of file":

return None

if \_\_name\_\_ == "\_\_main\_\_":

main(sys.argv)

### Hadoop Field Selection Class

Hadoop has a library class, org.apache.hadoop.mapred.lib.FieldSelectionMapReduce, that effectively allows you to process text data like the unix "cut" utility. The map function defined in the class treats each input key/value pair as a list of fields. You can specify the field separator (the default is the tab character). You can select an arbitrary list of fields as the map output key, and an arbitrary list of fields as the map output value. Similarly, the reduce function defined in the class treats each input key/value pair as a list of fields. You can select an arbitrary list of fields as the reduce output key, and an arbitrary list of fields as the reduce output value. For example:

$HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/hadoop-streaming.jar \

-D map.output.key.field.separa=. \

-D mapred.text.key.partitioner.options=-k1,2 \

-D mapred.data.field.separator=. \

-D map.output.key.value.fields.spec=6,5,1-3:0- \

-D reduce.output.key.value.fields.spec=0-2:5- \

-D mapred.reduce.tasks=12 \

-input myInputDirs \

-output myOutputDir \

-mapper org.apache.hadoop.mapred.lib.FieldSelectionMapReduce \

-reducer org.apache.hadoop.mapred.lib.FieldSelectionMapReduce \

-partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner

The option "-D map.output.key.value.fields.spec=6,5,1-3:0-" specifies key/value selection for the map outputs. Key selection spec and value selection spec are separated by ":". In this case, the map output key will consist of fields 6, 5, 1, 2, and 3. The map output value will consist of all fields (0- means field 0 and all the subsequent fields).

The option "-D reduce.output.key.value.fields.spec=0-2:5-" specifies key/value selection for the reduce outputs. In this case, the reduce output key will consist of fields 0, 1, 2 (corresponding to the original fields 6, 5, 1). The reduce output value will consist of all fields starting from field 5 (corresponding to all the original fields).

## Frequently Asked Questions

### How do I use Hadoop Streaming to run an arbitrary set of (semi) independent tasks?

Often you do not need the full power of Map Reduce, but only need to run multiple instances of the same program - either on different parts of the data, or on the same data, but with different parameters. You can use Hadoop Streaming to do this.

### How do I process files, one per map?

As an example, consider the problem of zipping (compressing) a set of files across the hadoop cluster. You can achieve this using either of these methods:

1. Hadoop Streaming and custom mapper script:
   * Generate a file containing the full HDFS path of the input files. Each map task would get one file name as input.
   * Create a mapper script which, given a filename, will get the file to local disk, gzip the file and put it back in the desired output directory
2. The existing Hadoop Framework:
   * Add these commands to your main function:
   * FileOutputFormat.setCompressOutput(conf, true);
   * FileOutputFormat.setOutputCompressorClass(conf, org.apache.hadoop.io.compress.GzipCodec.class);
   * conf.setOutputFormat(NonSplitableTextInputFormat.class);
   * conf.setNumReduceTasks(0);
   * Write your map function:
   * public void map(WritableComparable key,
   * Writable value,
   * OutputCollector output,
   * Reporter reporter)
   * throws IOException {
   * output.collect((Text)value, null);
   * }
   * Note that the output filename will not be the same as the original filename

### How many reducers should I use?

See the Hadoop Wiki for details: [Reducer](http://hadoop.apache.org/docs/stable/mapred_tutorial.html#Reducer)

### If I set up an alias in my shell script, will that work after -mapper?

For example, say I do: alias c1='cut -f1'. Will -mapper "c1" work?

Using an alias will not work, but variable substitution is allowed as shown in this example:

$ hadoop dfs -cat samples/student\_marks

alice 50

bruce 70

charlie 80

dan 75

$ c2='cut -f2'; $HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/hadoop-streaming.jar \

-D mapred.job.name='Experiment'

-input /user/me/samples/student\_marks

-output /user/me/samples/student\_out

-mapper \"$c2\" -reducer 'cat'

$ hadoop dfs -ls samples/student\_out

Found 1 items/user/me/samples/student\_out/part-00000 <r 3> 16

$ hadoop dfs -cat samples/student\_out/part-00000

50

70

75

80

### Can I use UNIX pipes?

For example, will -mapper "cut -f1 | sed s/foo/bar/g" work?

Currently this does not work and gives an "java.io.IOException: Broken pipe" error. This is probably a bug that needs to be investigated.

### What do I do if I get the "No space left on device" error?

For example, when I run a streaming job by distributing large executables (for example, 3.6G) through the -file option, I get a "No space left on device" error.

The jar packaging happens in a directory pointed to by the configuration variable stream.tmpdir. The default value of stream.tmpdir is /tmp. Set the value to a directory with more space:

-D stream.tmpdir=/export/bigspace/...

### How do I specify multiple input directories?

You can specify multiple input directories with multiple '-input' options:

hadoop jar hadoop-streaming.jar -input '/user/foo/dir1' -input '/user/foo/dir2'

### How do I generate output files with gzip format?

Instead of plain text files, you can generate gzip files as your generated output. Pass '-D mapred.output.compress=true -D mapred.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec' as option to your streaming job.

### How do I provide my own input/output format with streaming?

At least as late as version 0.14, Hadoop does not support multiple jar files. So, when specifying your own custom classes you will have to pack them along with the streaming jar and use the custom jar instead of the default hadoop streaming jar.

### How do I parse XML documents using streaming?

You can use the record reader StreamXmlRecordReader to process XML documents.

hadoop jar hadoop-streaming.jar -inputreader "StreamXmlRecord,begin=BEGIN\_STRING,end=END\_STRING" ..... (rest of the command)

Anything found between BEGIN\_STRING and END\_STRING would be treated as one record for map tasks.

### How do I update counters in streaming applications?

A streaming process can use the stderr to emit counter information.reporter:counter:<group>,<counter>,<amount> should be sent to stderr to update the counter.

### How do I update status in streaming applications?

A streaming process can use the stderr to emit status information. To set a status,reporter:status:<message> should be sent to stderr.

### How do I get the JobConf variables in a streaming job's mapper/reducer?

See [Configured Parameters](http://hadoop.apache.org/docs/stable/mapred_tutorial.html#Configured+Parameters). During the execution of a streaming job, the names of the "mapred" parameters are transformed. The dots ( . ) become underscores ( \_ ). For example, mapred.job.id becomes mapred\_job\_id and mapred.jar becomes mapred\_jar. In your code, use the parameter names with the underscores.

### How do I get the JobConf variables in a streaming job's mapper/reducer?

See the [Configured Parameters](http://hadoop.apache.org/docs/stable/mapred_tutorial.html#Configured+Parameters). During the execution of a streaming job, the names of the "mapred" parameters are transformed. The dots ( . ) become underscores ( \_ ). For example, mapred.job.id becomes mapred\_job\_id and mapred.jar becomes mapred\_jar. In your code, use the parameter names with the underscores.

**Joins with Map-Reduce:**

<http://www.javacodegeeks.com/2012/05/joins-with-map-reduce.html>

 Which are the three modes in which Hadoop can be run?  
The three modes in which Hadoop can be run are:  
1. standalone (local) mode  
2. Pseudo-distributed mode  
3. Fully distributed mode  
  
What are the features of Stand alone (local) mode?  
In stand-alone mode there are no daemons, everything runs on a single JVM. It has no DFS  
and utilizes the local file system. Stand-alone mode is suitable only for running MapReduce  
programs during development. It is one of the most least used environments.  
  
What are the features of Pseudo mode?  
Pseudo mode is used both for development and in the QA environment. In the Pseudo  
mode all the daemons run on the same machine.  
  
Can we call VMs as pseudos?  
No, VMs are not pseudos because VM is something different and pesudo is very specific to  
Hadoop.  
  
What are the features of Fully Distributed mode?  
Fully Distributed mode is used in the production environment, where we have ‘n’ number  
of machines forming a Hadoop cluster. Hadoop daemons run on a cluster of machines.  
There is one host onto which Namenode is running and another host on which datanode is  
running and then there are machines on which task tracker is running. We have separate  
masters and separate slaves in this distribution.  
  
Does Hadoop follows the UNIX pattern?  
Yes, Hadoop closely follows the UNIX pattern. Hadoop also has the ‘conf‘ directory as in the  
case of UNIX.  
  
In which directory Hadoop is installed?  
Cloudera and Apache has the same directory structure. Hadoop is installed in cd  
/usr/lib/hadoop-0.20/.  
  
What are the port numbers of Namenode, job tracker and task tracker?  
The port number for Namenode is ’70′, for job tracker is ’30′ and for task tracker is ’60′.  
]  
What is the Hadoop-core configuration?  
Hadoop core is configured by two xml files:  
1. hadoop-default.xml which was renamed to 2. hadoop-site.xml.  
These files are written in xml format. We have certain properties in these xml files, which  
consist of name and value. But these files do not exist now.  
  
What are the Hadoop configuration files at present?  
There are 3 configuration files in Hadoop:  
1. core-site.xml  
2. hdfs-site.xml  
3. mapred-site.xml  
These files are located in the conf/ subdirectory.  
  
How to exit the Vi editor?  
To exit the Vi Editor, press ESC and type :q and then press enter.  
  
What is a spill factor with respect to the RAM?  
Spill factor is the size after which your files move to the temp file. Hadoop-temp directory is  
used for this.  
  
Is fs.mapr.working.dir a single directory?  
Yes, fs.mapr.working.dir it is just one directory.  
  
Which are the three main hdfs-site.xml properties?  
The three main hdfs-site.xml properties are:  
1. dfs.name.dir which gives you the location on which metadata will be stored and where  
DFS is located – on disk or onto the remote.  
2. dfs.data.dir which gives you the location where the data is going to be stored.  
3. fs.checkpoint.dir which is for secondary Namenode.  
  
How to come out of the insert mode?  
To come out of the insert mode, press ESC, type :q (if you have not written anything) OR  
type :wq (if you have written anything in the file) and then press ENTER.  
  
What is Cloudera and why it is used?  
Cloudera is the distribution of Hadoop. It is a user created on VM by default. Cloudera  
belongs to Apache and is used for data processing.  
  
What happens if you get a ‘connection refused java exception’ when you type hadoop  
fsck /?  
It could mean that the Namenode is not working on your VM.  
We are using Ubuntu operating system with Cloudera, but from where we can  
  
download Hadoop or does it come by default with Ubuntu?  
This is a default configuration of Hadoop that you have to download from Cloudera or from  
Edureka’s dropbox and the run it on your systems. You can also proceed with your own  
configuration but you need a Linux box, be it Ubuntu or Red hat. There are installation  
steps present at the Cloudera location or in Edureka’s Drop box. You can go either ways.  
  
What does ‘jps’ command do?  
This command checks whether your Namenode, datanode, task tracker, job tracker, etc are  
working or not.  
  
How can I restart Namenode?  
1. Click on stop-all.sh and then click on start-all.sh OR  
2. Write sudo hdfs (press enter), su-hdfs (press enter), /etc/init.d/ha (press enter) and  
then /etc/init.d/hadoop-0.20-namenode start (press enter).  
  
What is the full form of fsck?  
Full form of fsck is File System Check.  
  
How can we check whether Namenode is working or not?  
To check whether Namenode is working or not, use the command /etc/init.d/hadoop-  
0.20-namenode status or as simple as jps.  
  
What does the command mapred.job.tracker do?  
The command mapred.job.tracker lists out which of your nodes is acting as a job tracker.  
  
What does /etc /init.d do?  
/etc /init.d specifies where daemons (services) are placed or to see the status of these  
daemons. It is very LINUX specific, and nothing to do with Hadoop.  
  
How can we look for the Namenode in the browser?  
If you have to look for Namenode in the browser, you don’t have to give localhost:8021, the  
port number to look for Namenode in the brower is 50070.  
  
How to change from SU to Cloudera?  
To change from SU to Cloudera just type exit.  
  
Which files are used by the startup and shutdown commands?  
Slaves and Masters are used by the startup and the shutdown commands.  
  
What do slaves consist of?  
Slaves consist of a list of hosts, one per line, that host datanode and task tracker servers.  
  
What do masters consist of?  
Masters contain a list of hosts, one per line, that are to host secondary namenode servers.  
  
What does hadoop-env.sh do?  
hadoop-env.sh provides the environment for Hadoop to run. JAVA\_HOME is set over here.  
  
Can we have multiple entries in the master files?  
Yes, we can have multiple entries in the Master files.  
  
Where is hadoop-env.sh file present?  
hadoop-env.sh file is present in the conf location.  
  
In Hadoop\_PID\_DIR, what does PID stands for?  
PID stands for ‘Process ID’.  
  
What does /var/hadoop/pids do?  
It stores the PID.  
  
What does hadoop-metrics.properties file do?  
hadoop-metrics.properties is used for ‘Reporting‘ purposes. It controls the reporting for  
Hadoop. The default status is ‘not to report‘.  
  
What are the network requirements for Hadoop?  
The Hadoop core uses Shell (SSH) to launch the server processes on the slave nodes. It  
requires password-less SSH connection between the master and all the slaves and the  
secondary machines.  
  
Why do we need a password-less SSH in Fully Distributed environment?  
We need a password-less SSH in a Fully-Distributed environment because when the cluster  
is LIVE and running in Fully  
Distributed environment, the communication is too frequent. The job tracker should be  
able to send a task to task tracker quickly.  
  
Does this lead to security issues?  
No, not at all. Hadoop cluster is an isolated cluster. And generally it has nothing to do with  
an internet. It has a different kind of a configuration. We needn’t worry about that kind of a  
security breach, for instance, someone hacking through the internet, and so on. Hadoop has  
a very secured way to connect to other machines to fetch and to process data.  
  
On which port does SSH work?  
SSH works on Port No. 22, though it can be configured. 22 is the default Port number.  
  
Can you tell us more about SSH?  
SSH is nothing but a secure shell communication, it is a kind of a protocol that works on a  
Port No. 22, and when you do an SSH, what you really require is a password.  
  
Why password is needed in SSH localhost?  
Password is required in SSH for security and in a situation where passwordless  
communication is not set.  
  
Do we need to give a password, even if the key is added in SSH?  
Yes, password is still required even if the key is added in SSH.  
  
What if a Namenode has no data?  
If a Namenode has no data it is not a Namenode. Practically, Namenode will have some  
data.  
  
What happens to job tracker when Namenode is down?  
When Namenode is down, your cluster is OFF, this is because Namenode is the single point  
of failure in HDFS.  
  
What happens to a Namenode, when job tracker is down?  
When a job tracker is down, it will not be functional but Namenode will be present. So,  
cluster is accessible if Namenode is working, even if the job tracker is not working.  
  
Can you give us some more details about SSH communication between Masters and  
the Slaves?  
SSH is a password-less secure communication where data packets are sent across the slave.  
It has some format into which data is sent across. SSH is not only between masters and  
slaves but also between two hosts.  
  
What is formatting of the DFS?  
Just like we do for Windows, DFS is formatted for proper structuring. It is not usually done  
as it formats the Namenode too.  
  
Does the HDFS client decide the input split or Namenode?  
No, the Client does not decide. It is already specified in one of the configurations through  
which input split is already configured.  
  
In Cloudera there is already a cluster, but if I want to form a cluster on Ubuntu can  
we do it?  
Yes, you can go ahead with this! There are installation steps for creating a new cluster. You  
can uninstall your present cluster and install the new cluster.  
  
Can we create a Hadoop cluster from scratch?  
Yes we can do that also once we are familiar with the Hadoop environment.  
  
Can we use Windows for Hadoop?  
Actually, Red Hat Linux or Ubuntu are the best Operating Systems for Hadoop. Windows is  
not used frequently for installing Hadoop as there are many support problems attached  
with Windows. Thus, Windows is not a preferred environment for Hadoop.

<http://hadoopbymanjunath.blogspot.in/2013/06/hadoop-interview-questions-which-covers.html>

<http://www.aired.in/2013/04/interview-questions-and-answers-for.html>

What is a JobTracker in Hadoop? How many instances of JobTracker run on a Hadoop Cluster? JobTracker is the daemon service for submitting and tracking MapReduce jobs in Hadoop. There is only One Job Tracker process run on any hadoop cluster. Job Tracker runs on its own JVM process. In a typical production cluster its run on a separate machine. Each slave node is configured with job tracker node location. The JobTracker is single point of failure for the Hadoop MapReduce service. If it goes down, all running jobs are halted. JobTracker in Hadoop performs following actions(from Hadoop Wiki:) Client applications submit jobs to the Job tracker. The JobTracker talks to the NameNode to determine the location of the data The JobTracker locates TaskTracker nodes with available slots at or near the data The JobTracker submits the work to the chosen TaskTracker nodes. The TaskTracker nodes are monitored. If they do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different TaskTracker. A TaskTracker will notify the JobTracker when a task fails. The JobTracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may may even blacklist the TaskTracker as unreliable. When the work is completed, the JobTracker updates its status. Client applications can poll the JobTracker for information. How JobTracker schedules a task? The TaskTrackers send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the JobTracker that it is still alive. These message also inform the JobTracker of the number of available slots, so the JobTracker can stay up to date with where in the cluster work can be delegated. When the JobTracker tries to find somewhere to schedule a task within the MapReduce operations, it first looks for an empty slot on the same server that hosts the DataNode containing the data, and if not, it looks for an empty slot on a machine in the same rack. What is a Task Tracker in Hadoop? How many instances of TaskTracker run on a Hadoop Cluster A TaskTracker is a slave node daemon in the cluster that accepts tasks (Map, Reduce and Shuffle operations) from a JobTracker. There is only One Task Tracker process run on any hadoop slave node. Task Tracker runs on its own JVM process. Every TaskTracker is configured with a set of slots, these indicate the number of tasks that it can accept. The TaskTracker starts a separate JVM processes to do the actual work (called as Task Instance) this is to ensure that process failure does not take down the task tracker. The TaskTracker monitors these task instances, capturing the output and exit codes. When the Task instances finish, successfully or not, the task tracker notifies the JobTracker. The TaskTrackers also send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the JobTracker that it is still alive. These message also inform the JobTracker of the number of available slots, so the JobTracker can stay up to date with where in the cluster work can be delegated. What is a Task instance in Hadoop? Where does it run? Task instances are the actual MapReduce jobs which are run on each slave node. The TaskTracker starts a separate JVM processes to do the actual work (called as Task Instance) this is to ensure that process failure does not take down the task tracker. Each Task Instance runs on its own JVM process. There can be multiple processes of task instance running on a slave node. This is based on the number of slots configured on task tracker. By default a new task instance JVM process is spawned for a task. How many Daemon processes run on a Hadoop system? Hadoop is comprised of five separate daemons. Each of these daemon run in its own JVM. Following 3 Daemons run on Master nodes NameNode - This daemon stores and maintains the metadata for HDFS. Secondary NameNode - Performs housekeeping functions for the NameNode. JobTracker - Manages MapReduce jobs, distributes individual tasks to machines running the Task Tracker. Following 2 Daemons run on each Slave nodes DataNode – Stores actual HDFS data blocks. TaskTracker - Responsible for instantiating and monitoring individual Map and Reduce tasks. What is configuration of a typical slave node on Hadoop cluster? How many JVMs run on a slave node?     Single instance of a Task Tracker is run on each Slave node. Task tracker is run as a separate JVM process.     Single instance of a DataNode daemon is run on each Slave node. DataNode daemon is run as a separate JVM process.     One or Multiple instances of Task Instance is run on each slave node. Each task instance is run as a separate JVM process. The number of Task instances can be controlled by configuration. Typically a high end machine is configured to run more task instances. What is the difference between HDFS and NAS ? The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. Following are differences between HDFS and NAS     In HDFS Data Blocks are distributed across local drives of all machines in a cluster. Whereas in NAS data is stored on dedicated hardware.     HDFS is designed to work with MapReduce System, since computation are moved to data. NAS is not suitable for MapReduce since data is stored seperately from the computations.     HDFS runs on a cluster of machines and provides redundancy usinga replication protocal. Whereas NAS is provided by a single machine therefore does not provide data redundancy. How NameNode Handles data node failures? NameNode periodically receives a Heartbeat and a Blockreport from each of the DataNodes in the cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly. A Blockreport contains a list of all blocks on a DataNode. When NameNode notices that it has not recieved a hearbeat message from a data node after a certain amount of time, the data node is marked as dead. Since blocks will be under replicated the system begins replicating the blocks that were stored on the dead datanode. The NameNode Orchestrates the replication of data blocks from one datanode to another. The replication data transfer happens directly between datanodes and the data never passes through the namenode. Does MapReduce programming model provide a way for reducers to communicate with each other? In a MapReduce job can a reducer communicate with another reducer? Nope, MapReduce programming model does not allow reducers to communicate with each other. Reducers run in isolation. Can I set the number of reducers to zero? Yes, Setting the number of reducers to zero is a valid configuration in Hadoop. When you set the reducers to zero no reducers will be executed, and the output of each mapper will be stored to a separate file on HDFS. [This is different from the condition when reducers are set to a number greater than zero and the Mappers output (intermediate data) is written to the Local file system(NOT HDFS) of each mappter slave node.] Where is the Mapper Output (intermediate kay-value data) stored ? The mapper output (intermediate data) is stored on the Local file system (NOT HDFS) of each individual mapper nodes. This is typically a temporary directory location which can be setup in config by the hadoop administrator. The intermediate data is cleaned up after the Hadoop Job completes. What are combiners? When should I use a combiner in my MapReduce Job? Combiners are used to increase the efficiency of a MapReduce program. They are used to aggregate intermediate map output locally on individual mapper outputs. Combiners can help you reduce the amount of data that needs to be transferred across to the reducers. You can use your reducer code as a combiner if the operation performed is commutative and associative. The execution of combiner is not guaranteed, Hadoop may or may not execute a combiner. Also, if required it may execute it more then 1 times. Therefore your MapReduce jobs should not depend on the combiners execution. What is Writable & WritableComparable interface?     org.apache.hadoop.io.Writable is a Java interface. Any key or value type in the Hadoop Map-Reduce framework implements this interface. Implementations typically implement a static read(DataInput) method which constructs a new instance, calls readFields(DataInput) and returns the instance.     org.apache.hadoop.io.WritableComparable is a Java interface. Any type which is to be used as a key in the Hadoop Map-Reduce framework should implement this interface. WritableComparable objects can be compared to each other using Comparators. What is the Hadoop MapReduce API contract for a key and value Class?     The Key must implement the org.apache.hadoop.io.WritableComparable interface.     The value must implement the org.apache.hadoop.io.Writable interface. What is a IdentityMapper and IdentityReducer in MapReduce ?     org.apache.hadoop.mapred.lib.IdentityMapper Implements the identity function, mapping inputs directly to outputs. If MapReduce programmer do not set the Mapper Class using JobConf.setMapperClass then IdentityMapper.class is used as a default value.     org.apache.hadoop.mapred.lib.IdentityReducer Performs no reduction, writing all input values directly to the output. If MapReduce programmer do not set the Reducer Class using JobConf.setReducerClass then IdentityReducer.class is used as a default value. What is the meaning of speculative execution in Hadoop? Why is it important? Speculative execution is a way of coping with individual Machine performance. In large clusters where hundreds or thousands of machines are involved there may be machines which are not performing as fast as others. This may result in delays in a full job due to only one machine not performaing well. To avoid this, speculative execution in hadoop can run multiple copies of same map or reduce task on different slave nodes. The results from first node to finish are used. When is the reducers are started in a MapReduce job? In a MapReduce job reducers do not start executing the reduce method until the all Map jobs have completed. Reducers start copying intermediate key-value pairs from the mappers as soon as they are available. The programmer defined reduce method is called only after all the mappers have finished. If reducers do not start before all mappers finish then why does the progress on MapReduce job shows something like Map(50%) Reduce(10%)? Why reducers progress percentage is displayed when mapper is not finished yet? Reducers start copying intermediate key-value pairs from the mappers as soon as they are available. The progress calculation also takes in account the processing of data transfer which is done by reduce process, therefore the reduce progress starts showing up as soon as any intermediate key-value pair for a mapper is available to be transferred to reducer. Though the reducer progress is updated still the programmer defined reduce method is called only after all the mappers have finished. What is HDFS ? How it is different from traditional file systems? HDFS, the Hadoop Distributed File System, is responsible for storing huge data on the cluster. This is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant.     HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware.     HDFS provides high throughput access to application data and is suitable for applications that have large data sets.     HDFS is designed to support very large files. Applications that are compatible with HDFS are those that deal with large data sets. These applications write their data only once but they read it one or more times and require these reads to be satisfied at streaming speeds. HDFS supports write-once-read-many semantics on files. What is HDFS Block size? How is it different from traditional file system block size? In HDFS data is split into blocks and distributed across multiple nodes in the cluster. Each block is typically 64Mb or 128Mb in size. Each block is replicated multiple times. Default is to replicate each block three times. Replicas are stored on different nodes. HDFS utilizes the local file system to store each HDFS block as a separate file. HDFS Block size can not be compared with the traditional file system block size. What is a NameNode? How many instances of NameNode run on a Hadoop Cluster? The NameNode is the centerpiece of an HDFS file system. It keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself. There is only One NameNode process run on any hadoop cluster. NameNode runs on its own JVM process. In a typical production cluster its run on a separate machine. The NameNode is a Single Point of Failure for the HDFS Cluster. When the NameNode goes down, the file system goes offline. Client applications talk to the NameNode whenever they wish to locate a file, or when they want to add/copy/move/delete a file. The NameNode responds the successful requests by returning a list of relevant DataNode servers where the data lives. What is a DataNode? How many instances of DataNode run on a Hadoop Cluster? A DataNode stores data in the Hadoop File System HDFS. There is only One DataNode process run on any hadoop slave node. DataNode runs on its own JVM process. On startup, a DataNode connects to the NameNode. DataNode instances can talk to each other, this is mostly during replicating data. How the Client communicates with HDFS? The Client communication to HDFS happens using Hadoop HDFS API. Client applications talk to the NameNode whenever they wish to locate a file, or when they want to add/copy/move/delete a file on HDFS. The NameNode responds the successful requests by returning a list of relevant DataNode servers where the data lives. Client applications can talk directly to a DataNode, once the NameNode has provided the location of the data. How the HDFS Blocks are replicated? HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks; all blocks in a file except the last block are the same size. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are write-once and have strictly one writer at any time. The NameNode makes all decisions regarding replication of blocks. HDFS uses rack-aware replica placement policy. In default configuration there are total 3 copies of a datablock on HDFS, 2 copies are stored on datanodes on same rack and 3rd copy on a different rack. - See more at: <http://www.aired.in/2013/04/interview-questions-and-answers-for.html#sthash.sanDik8T.dpuf>

<http://www.freshershine.com/job-tips/technical-interview/hadoop/hadoop-interview-questions-and-answers-hdfs/2/>