1.3. Not-so-quick Start Guide

1.3.1. Requirements

HBase has the following requirements. Please read the section below carefully and ensure that all requirements have been satisfied. Failure to do so will cause you (and us) grief debugging strange errors and/or data loss.

1.3.1.1. java

Just like Hadoop, HBase requires java 6 from Oracle. Usually you'll want to use the latest version available except the problematic u18 (u24 is the latest version as of this writing).

1.3.1.2. hadoop

This version of HBase will only run on Hadoop 0.20.x. It will not run on hadoop 0.21.x (nor 0.22.x). HBase will lose data unless it is running on an HDFS that has a durable sync. Currently only the branch-0.20-append branch has this attribute[1]. No official releases have been made from this branch up to now so you will have to build your own Hadoop from the tip of this branch. Michael Noll has written a detailed blog, Building an Hadoop 0.20.x version for HBase 0.90.2, on how to build an Hadoop from branch-0.20-append. Recommended.

Or rather than build your own, you could use Cloudera's CDH3. CDH has the 0.20-append patches needed to add a durable sync (CDH3 betas will suffice; b2, b3, or b4).

Because HBase depends on Hadoop, it bundles an instance of the Hadoop jar under its lib directory. The bundled Hadoop was made from the Apache branch-0.20-append branch at the time of this HBase's release. It is *critical* that the version of Hadoop that is out on your cluster matches what is Hbase match. Replace the hadoop jar found in the HBase lib directory with the hadoop jar you are running out on your cluster to avoid version mismatch issues. Make sure you replace the jar all over your cluster. For example, versions of CDH do not have HDFS-724 whereas Hadoops branch-0.20-append branch does have HDFS-724. This patch changes the RPC version because protocol was changed. Version mismatch issues have various manifestations but often all looks like its hung up.

Can I just replace the jar in Hadoop 0.20.2 tarball with the *sync*-supporting Hadoop jar found in HBase?

You could do this. It works going by a recent posting up on the mailing list.

Hadoop Security

HBase will run on any Hadoop 0.20.x that incorporates Hadoop security features -- e.g. Y! 0.20S or CDH3B3 -- as long as you do as suggested above and replace the Hadoop jar that ships with HBase with the secure version.

1.3.1.3. ssh

ssh must be installed and sshd must be running to use Hadoop's scripts to manage remote

Hadoop and HBase daemons. You must be able to ssh to all nodes, including your local node, using passwordless login (Google "ssh passwordless login").

1.3.1.4. DNS

HBase uses the local hostname to self-report it's IP address. Both forward and reverse DNS resolving should work.

If your machine has multiple interfaces, HBase will use the interface that the primary hostname resolves to.

If this is insufficient, you can set hbase.regionserver.dns.interface to indicate the primary interface. This only works if your cluster configuration is consistent and every host has the same network interface configuration.

Another alternative is setting hbase.regionserver.dns.nameserver to choose a different nameserver than the system wide default.

1.3.1.5. NTP

The clocks on cluster members should be in basic alignments. Some skew is tolerable but wild skew could generate odd behaviors. Run NTP on your cluster, or an equivalent.

If you are having problems querying data, or "weird" cluster operations, check system time!

1.3.1.6. ulimit and nproc

HBase is a database. It uses a lot of files all at the same time. The default ulimit -n -- i.e. user file limit -- of 1024 on most *nix systems is insufficient (On mac os x its 256). Any significant amount of loading will lead you to FAQ: Why do I see "java.io.IOException...(Too many open files)" in my logs?. You may also notice errors such as

```
2010-04-06 03:04:37,542 INFO org.apache.hadoop.hdfs.DFSClient: Exception increateBlockOutputSt
2010-04-06 03:04:37,542 INFO org.apache.hadoop.hdfs.DFSClient: Abandoning block blk_-69355249
```

Do yourself a favor and change the upper bound on the number of file descriptors. Set it to north of 10k. See the above referenced FAQ for how. You should also up the hbase users' nproc setting; under load, a low-nproc setting could manifest as OutOfMemoryError [2] [3].

To be clear, upping the file descriptors and nproc for the user who is running the HBase process is an operating system configuration, not an HBase configuration. Also, a common mistake is that administrators will up the file descriptors for a particular user but for whatever reason, HBase will be running as some one else. HBase prints in its logs as the first line the ulimit its seeing. Ensure its correct. [4]

1.3.1.6.1. ulimit on Ubuntu

If you are on Ubuntu you will need to make the following changes:

In the file /etc/security/limits.conf add a line like:

```
hadoop  -       nofile  32768
```

Replace hadoop with whatever user is running Hadoop and HBase. If you have separate users, you will need 2 entries, one for each user. In the same file set nproc hard and soft limits. For example:

```
hadoop soft/hard nproc 32000
```

.

In the file /etc/pam.d/common-session add as the last line in the file:

```
session required  pam_limits.so
```

Otherwise the changes in /etc/security/limits.conf won't be applied.

Don't forget to log out and back in again for the changes to take effect!

1.3.1.7. dfs.datanode.max.xcievers

An Hadoop HDFS datanode has an upper bound on the number of files that it will serve at any one time. The upper bound parameter is called xcievers (yes, this is misspelled). Again, before doing any loading, make sure you have configured Hadoop's conf/hdfs-site.xml setting the xceivers value to at least the following:

```
    <property>
      <name>dfs.datanode.max.xcievers</name>
      <value>4096</value>
    </property>
```

Be sure to restart your HDFS after making the above configuration.

Not having this configuration in place makes for strange looking failures. Eventually you'll see a complain in the datanode logs complaining about the xcievers exceeded, but on the run up to this one manifestation is complaint about missing blocks. For example: 10/12/08 20:10:31 INFO hdfs.DFSClient: Could not obtain block blk_XXXXXXXXXXXXXXXXXXXXXX_YYYYYYYY from any node: java.io.IOException: No live nodes contain current block. Will get new block locations from namenode and retry...[5]

1.3.1.8. Windows

HBase has been little tested running on windows. Running a production install of HBase on top of windows is not recommended.

If you are running HBase on Windows, you must install Cygwin to have a *nix-like environment for the shell scripts. The full details are explained in the Windows Installation guide. Also search our user mailing list to pick up latest fixes figured by windows users.

1.3.2. HBase run modes: Standalone and Distributed

HBase has two run modes: Section 1.3.2.1, "Standalone HBase" and Section 1.3.2.2, "Distributed". Out of the box, HBase runs in standalone mode. To set up a distributed deploy, you will need to configure HBase by editing files in the HBase conf directory.

Whatever your mode, you will need to edit conf/hbase-env.sh to tell HBase which java to use. In this file you set HBase environment variables such as the heapsize and other options for the JVM, the preferred location for log files, etc. Set JAVA_HOME to point at the root of your java install.

1.3.2.1. Standalone HBase

This is the default mode. Standalone mode is what is described in the Section 1.2, "Quick Start" section. In standalone mode, HBase does not use HDFS -- it uses the local filesystem instead -- and it runs all HBase daemons and a local ZooKeeper all up in the same JVM. Zookeeper binds to a well known port so clients may talk to HBase.

1.3.2.2. Distributed

Distributed mode can be subdivided into distributed but all daemons run on a single node -- a.k.a *pseudo-distributed*-- and *fully-distributed* where the daemons are spread across all nodes in the cluster [6].

Distributed modes require an instance of the *Hadoop Distributed File System* (HDFS). See the Hadoop requirements and instructions for how to set up a HDFS. Before proceeding, ensure you have an appropriate, working HDFS.

Below we describe the different distributed setups. Starting, verification and exploration of your install, whether a *pseudo-distributed* or *fully-distributed* configuration is described in a section that follows, Section 1.3.2.3, "Running and Confirming Your Installation". The same verification script applies to both deploy types.

1.3.2.2.1. Pseudo-distributed

A pseudo-distributed mode is simply a distributed mode run on a single host. Use this configuration testing and prototyping on HBase. Do not use this configuration for production nor for evaluating HBase performance.

Once you have confirmed your HDFS setup, edit conf/hbase-site.xml. This is the file into which you add local customizations and overrides for <xreg></xreg> and Section 1.3.2.2.2.3, "HDFS Client Configuration". Point HBase at the running Hadoop HDFS instance by setting the hbase.rootdir property. This property points HBase at the Hadoop filesystem instance to use. For example, adding the properties below to your hbase-site.xml says that HBase should use the /hbase directory in the HDFS whose namenode is at port 9000 on your local machine, and that it should run with one replica only (recommended for pseudo-distributed mode):

```
<configuration>
  ...
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://localhost:9000/hbase</value>
    <description>The directory shared by RegionServers.
    </description>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
    <description>The replication count for HLog and HFile storage. Should not be greater than HDFS
    </description>
  </property>
  ...
</configuration>
```

Note

Let HBase create the hbase.rootdir directory. If you don't, you'll get warning saying HBase needs a migration run because the directory is missing files expected by HBase (it'll create them if you let it).

Note

Above we bind to localhost. This means that a remote client cannot connect. Amend accordingly, if you want to connect from a remote location.

Now skip to Section 1.3.2.3, "Running and Confirming Your Installation" for how to start and verify your pseudo-distributed install.[7]

1.3.2.2.2. Fully-distributed

For running a fully-distributed operation on more than one host, make the following configurations. In hbase-site.xml, add the property hbase.cluster.distributed and set it to true and point the HBase hbase.rootdir at the appropriate HDFS NameNode and location in HDFS where you would like HBase to write data. For example, if you namenode were running at namenode.example.org on port 9000 and you wanted to home your HBase in HDFS at /hbase, make the following configuration.

```
<configuration>
  ...
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://namenode.example.org:9000/hbase</value>
    <description>The directory shared by RegionServers.
    </description>
  </property>
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
    <description>The mode the cluster will be in. Possible values are
      false: standalone and pseudo-distributed setups with managed Zookeeper
      true: fully-distributed with unmanaged Zookeeper Quorum (see hbase-env.sh)
    </description>
  </property>
  ...
</configuration>
```

1.3.2.2.2.1. regionservers

In addition, a fully-distributed mode requires that you modify conf/regionservers. The Section 1.3.3.1.2, "regionservers" file lists all hosts that you would have running HRegionServers, one host per line (This file in HBase is like the Hadoop slaves file). All servers listed in this file will be started and stopped when HBase cluster start or stop is run.

1.3.2.2.2.2. ZooKeeper

A distributed HBase depends on a running ZooKeeper cluster. All participating nodes and clients need to be able to access the running ZooKeeper ensemble. HBase by default manages a ZooKeeper "cluster" for you. It will start and stop the ZooKeeper ensemble as part of the HBase start/stop process. You can also manage the ZooKeeper ensemble independent of HBase and just point HBase at the cluster it should use. To toggle HBase management of ZooKeeper, use the HBASE_MANAGES_ZK variable in conf/hbase-env.sh. This variable, which defaults to true, tells HBase whether to start/stop the ZooKeeper ensemble servers as part of HBase start/stop.

When HBase manages the ZooKeeper ensemble, you can specify ZooKeeper configuration using its native zoo.cfg file, or, the easier option is to just specify ZooKeeper options directly in conf/hbase-site.xml. A ZooKeeper configuration option can be set as a property in the HBase hbase-site.xml XML configuration file by prefacing the ZooKeeper option name with hbase.zookeeper.property. For example, the clientPort setting in ZooKeeper can be changed by setting the hbase.zookeeper.property.clientPort property. For all default values used by HBase, including ZooKeeper configuration, see Section 3.1.1, "HBase Default Configuration". Look for the hbase.zookeeper.property prefix [8]

You must at least list the ensemble servers in hbase-site.xml using the hbase.zookeeper.quorum property. This property defaults to a single ensemble member at localhost which is not suitable for a fully distributed HBase. (It binds to the local machine only and remote clients will not be able to connect).

How many ZooKeepers should I run?

You can run a ZooKeeper ensemble that comprises 1 node only but in production it is recommended that you run a ZooKeeper ensemble of 3, 5 or 7 machines; the more members an ensemble has, the more tolerant the ensemble is of host failures. Also, run an odd number of machines. There can be no quorum if the number of members is an even number. Give each ZooKeeper server around 1GB of RAM, and if possible, its own dedicated disk (A dedicated disk is the best thing you can do to ensure a performant ZooKeeper ensemble). For very heavily loaded clusters, run ZooKeeper servers on separate machines from RegionServers (DataNodes and TaskTrackers).

For example, to have HBase manage a ZooKeeper quorum on nodes *rs{1,2,3,4,5}.example.com*, bound to port 2222 (the default is 2181) ensure HBASE_MANAGE_ZK is commented out or set to true in conf/hbase-env.sh and then edit conf/hbase-site.xml and set hbase.zookeeper.property.clientPort and hbase.zookeeper.quorum. You should also set hbase.zookeeper.property.dataDir to other than the default as the default has ZooKeeper persist data under /tmp which is often cleared on system restart. In the example below we have ZooKeeper persist to /user/local/zookeeper.

```
<configuration>
 ...
 <property>
  <name>hbase.zookeeper.property.clientPort</name>
  <value>2222</value>
  <description>Property from ZooKeeper's config zoo.cfg.
  The port at which the clients will connect.
  </description>
 </property>
 <property>
  <name>hbase.zookeeper.quorum</name>
  <value>rs1.example.com,rs2.example.com,rs3.example.com,rs4.example.com,rs5.example.com</value
  <description>Comma separated list of servers in the ZooKeeper Quorum.
  For example, "host1.mydomain.com,host2.mydomain.com,host3.mydomain.com".
  By default this is set to localhost for local and pseudo-distributed modes
  of operation. For a fully-distributed setup, this should be set to a full
  list of ZooKeeper quorum servers. If HBASE_MANAGES_ZK is set in hbase-env.sh
  this is the list of servers which we will start/stop ZooKeeper on.
  </description>
 </property>
```

```
   <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/usr/local/zookeeper</value>
    <description>Property from ZooKeeper's config zoo.cfg.
    The directory where the snapshot is stored.
    </description>
   </property>
   ...
 </configuration>
```

1.3.2.2.2.1. Using existing ZooKeeper ensemble

To point HBase at an existing ZooKeeper cluster, one that is not managed by HBase, set HBASE_MANAGES_ZK in conf/hbase-env.sh to false

```
...
# Tell HBase whether it should manage it's own instance of Zookeeper or not.
export HBASE_MANAGES_ZK=false
```

Next set ensemble locations and client port, if non-standard, in hbase-site.xml, or add a suitably configured zoo.cfg to HBase's CLASSPATH. HBase will prefer the configuration found in zoo.cfg over any settings in hbase-site.xml.

When HBase manages ZooKeeper, it will start/stop the ZooKeeper servers as a part of the regular start/stop scripts. If you would like to run ZooKeeper yourself, independent of HBase start/stop, you would do the following

```
${HBASE_HOME}/bin/hbase-daemons.sh {start,stop} zookeeper
```

Note that you can use HBase in this manner to spin up a ZooKeeper cluster, unrelated to HBase. Just make sure to set HBASE_MANAGES_ZK to false if you want it to stay up across HBase restarts so that when HBase shuts down, it doesn't take ZooKeeper down with it.

For more information about running a distinct ZooKeeper cluster, see the ZooKeeper Getting Started Guide.

1.3.2.2.2.3. HDFS Client Configuration

Of note, if you have made *HDFS client configuration* on your Hadoop cluster -- i.e. configuration you want HDFS clients to use as opposed to server-side configurations -- HBase will not see this configuration unless you do one of the following:

- Add a pointer to your HADOOP_CONF_DIR to the HBASE_CLASSPATH environment variable in hbase-env.sh.
- Add a copy of hdfs-site.xml (or hadoop-site.xml) or, better, symlinks, under ${HBASE_HOME}/conf, or
- if only a small set of HDFS client configurations, add them to hbase-site.xml.

An example of such an HDFS client configuration is dfs.replication. If for example, you want to run with a replication factor of 5, hbase will create files with the default of 3 unless you do the above to make the configuration available to HBase.

1.3.2.3. Running and Confirming Your Installation

Make sure HDFS is running first. Start and stop the Hadoop HDFS daemons by running bin/start-hdfs.sh over in the HADOOP_HOME directory. You can ensure it started properly by testing the put and get of files into the Hadoop filesystem. HBase does not normally use the mapreduce daemons. These do not need to be started.

If you are managing your own ZooKeeper, start it and confirm its running else, HBase will start up ZooKeeper for you as part of its start process.

Start HBase with the following command:

```
bin/start-hbase.sh
```

Run the above from the HBASE_HOME directory.

You should now have a running HBase instance. HBase logs can be found in the logs subdirectory. Check them out especially if HBase had trouble starting.

HBase also puts up a UI listing vital attributes. By default its deployed on the Master host at port 60010 (HBase RegionServers listen on port 60020 by default and put up an informational http server at 60030). If the Master were running on a host named master.example.org on the default port, to see the Master's homepage you'd point your browser at http://master.example.org:60010.

Once HBase has started, see the Section 1.2.3, "Shell Exercises" for how to create tables, add data, scan your insertions, and finally disable and drop your tables.

To stop HBase after exiting the HBase shell enter

```
$ ./bin/stop-hbase.sh
stopping hbase...............
```

Shutdown can take a moment to complete. It can take longer if your cluster is comprised of many machines. If you are running a distributed operation, be sure to wait until HBase has shut down completely before stopping the Hadoop daemons.

1.3.3. Example Configurations

1.3.3.1. Basic Distributed HBase Install

Here is an example basic configuration for a distributed ten node cluster. The nodes are named example0, example1, etc., through node example9 in this example. The HBase Master and the HDFS namenode are running on the node example0. RegionServers run on nodes example1-example9. A 3-node ZooKeeper ensemble runs on example1, example2, and example3 on the default ports. ZooKeeper data is persisted to the directory /export/zookeeper. Below we show what the main configuration files -- hbase-site.xml, regionservers, and hbase-env.sh -- found in the HBase conf directory might look like.

1.3.3.1.1. hbase-site.xml

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
 <property>
  <name>hbase.zookeeper.quorum</name>
  <value>example1,example2,example3</value>
```

```
    <description>The directory shared by RegionServers.
    </description>
  </property>
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/export/zookeeper</value>
    <description>Property from ZooKeeper's config zoo.cfg.
    The directory where the snapshot is stored.
    </description>
  </property>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://example0:9000/hbase</value>
    <description>The directory shared by RegionServers.
    </description>
  </property>
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
    <description>The mode the cluster will be in. Possible values are
      false: standalone and pseudo-distributed setups with managed Zookeeper
      true: fully-distributed with unmanaged Zookeeper Quorum (see hbase-env.sh)
    </description>
  </property>
</configuration>
```

1.3.3.1.2. regionservers

In this file you list the nodes that will run RegionServers. In our case we run RegionServers on all but the head node example1 which is carrying the HBase Master and the HDFS namenode

```
  example1
  example3
  example4
  example5
  example6
  example7
  example8
  example9
```

1.3.3.1.3. hbase-env.sh

Below we use a diff to show the differences from default in the hbase-env.sh file. Here we are setting the HBase heap to be 4G instead of the default 1G.

```

$ git diff hbase-env.sh
diff --git a/conf/hbase-env.sh b/conf/hbase-env.sh
index e70ebc6..96f8c27 100644
--- a/conf/hbase-env.sh
+++ b/conf/hbase-env.sh
@@ -31,7 +31,7 @@ export JAVA_HOME=/usr/lib//jvm/java-6-sun/
 # export HBASE_CLASSPATH=

 # The maximum amount of heap to use, in MB. Default is 1000.
-# export HBASE_HEAPSIZE=1000
+export HBASE_HEAPSIZE=4096
```

```
# Extra Java runtime options.
# Below are what we set by default. May only work with SUN JVM.
```

Use `rsync` to copy the content of the conf directory to all nodes of the cluster.

---

[1] See CHANGES.txt in branch-0.20-append to see list of patches involved adding append on the Hadoop 0.20 branch.

[2] See Jack Levin's major hdfs issues note up on the user list.

[3] The requirement that a database requires upping of system limits is not peculiar to HBase. See for example the section *Setting Shell Limits for the Oracle User* in Short Guide to install Oracle 10 on Linux.

[4] A useful read setting config on you hadoop cluster is Aaron Kimballs' Configuration Parameters: What can you just ignore?

[5] See Hadoop HDFS: Deceived by Xciever for an informative rant on xceivering.

[6] The pseudo-distributed vs fully-distributed nomenclature comes from Hadoop.

[7] See Pseudo-distributed mode extras for notes on how to start extra Masters and RegionServers when running pseudo-distributed.

[8] For the full list of ZooKeeper configurations, see ZooKeeper's zoo.cfg. HBase does not ship with a zoo.cfg so you will need to browse the conf directory in an appropriate ZooKeeper download.
