Nutch0.9-Hadoop0.10-Tutorial

How to setup Nutch and Hadoop on Ubuntu 6.06

Prerequisites

To run Hadoop one needs at least 2 computers to make use of a real distributed file system, it can also run on a single machine but than no use is made of the distributed capabilities.

Nutch is written in Java, so the java compiler and runtime are needed as well as ant. Hadoop makes use of ssh clients and servers on all machines. Lucene needs an servlet container, I used tomcat5.

To be able to login as root with su execute the following command and enter the new password for root as prompted.

```
sudo passwd
```

Login as root

```
su
```

Enable the universe and multiverse repositories by editing the apt sources.list file.

```
vi /etc/apt/sources.list
```

Or execute the following if you are in the Netherlands and are using Ubuntu 6.06 Dapper.

```
echo "deb http://nl.archive.ubuntu.com/ubuntu/ dapper universe multiverse" >> /etc/apt/sources.list
echo "deb-src http://nl.archive.ubuntu.com/ubuntu/ dapper universe multiverse" >> /etc/apt/sources.list
```

Update the apt cache

```
apt-get update
```

Install the necessary packages for Nutch (java and ssh) on all machines

```
apt-get install sun-java5-jre
apt-get install ssh

update-alternatives --config java
#select /usr/lib/jvm/java-1.5.0-sun/jre/bin/java
```

And for the search web server

```
apt-get install apache2
apt-get install sun-java5-jdk
apt-get install tomcat5
```

Configure tomcat by editing /etc/default/tomcat5

```

vi /etc/default/tomcat5
#Add JAVA_HOME=/usr/lib/jvm/java-1.5.0-sun/
```

Or execute the following

```
echo "JAVA_HOME=/usr/lib/jvm/java-1.5.0-sun/" >> /etc/default/tomcat5
```

Build nutch

Download Nutch, this includes Hadoop and Lucene. I used the latest nightly build, which was at the time of writing 2007-02-06. Nutch nightly

Unpack the tarball to nutch-nightly and build it with ant.

```
tar -xvzf nutch-2007-02-06.tar.gz
cd nutch-nightly
mkdir /nutch-build
echo "/nutch-build" >> build.properties
ant package
```

Setup

Prepare the machines

Create the nutch user on each machine and create the necessary directories for nutch

```
ssh root@???

mkdir /nutch-0.9.0
mkdir /nutch-0.9.0/search
mkdir /nutch-0.9.0/filesystem
mkdir /nutch-0.9.0/local
mkdir /nutch-0.9.0/home

groupadd users
useradd -d /nutch-0.9.0/home -g users nutch
passwd nutch

chown -R nutch:users /nutch-0.9.0
exit
```

Install and configure nutch and hadoop

Install nutch on the namenode (the master) and add the following variables to the hadoop-env.sh shell script.

```
ssh nutch@???
cp -Rv /nutch-build/* /nutch-0.9.0/search/

echo "export HADOOP_HOME=/nutch-0.9.0/search" >> /nutch-0.9.0/search/conf/hadoop-env.sh
echo "export JAVA_HOME=/usr/lib/jvm/java-1.5.0-sun" >> /nutch-0.9.0/search/conf/hadoop-env.sh
echo "export HADOOP_LOG_DIR=/nutch-0.9.0/search/logs" >> /nutch-0.9.0/search/conf/hadoop-env.sh
echo "export HADOOP_SLAVES=/nutch-0.9.0/search/conf/slaves" >> /nutch-0.9.0/search/conf/hadoop-env.sh

exit
```

Configure SSH

Create ssh keys so that the nutch user can login over ssh without being prompted for a password.

```
ssh nutch@???
cd /nutch-0.9.0/home
ssh-keygen -t rsa
```

```
#! Use empty responses for each prompt
#  Enter passphrase (empty for no passphrase):
#  Enter same passphrase again:
#  Your identification has been saved in /nutch-0.9.0/home/.ssh/id_rsa.
#  Your public key has been saved in /nutch-0.9.0/home/.ssh/id_rsa.pub.
#  The key fingerprint is:
#  a6:5c:c3:eb:18:94:0b:06:a1:a6:29:58:fa:80:0a:bc nutch@localhost
```

Copy the key for the master (the namenode) to the authorized_keys file that will be copied to the other machines (the slaves).

```
cd /nutch-0.9.0/home/.ssh
cp id_rsa.pub authorized_keys
```

Configure Hadoop

Edit the mapred-default.xml configuration file. If it's missing, create it, with the following content:

```
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>
  <name>mapred.map.tasks</name>
  <value>2</value>
  <description>
    This should be a prime number larger than multiple number of slave hosts,
    e.g. for 3 nodes set this to 17
  </description>
</property>

<property>
  <name>mapred.reduce.tasks</name>
  <value>2</value>
  <description>
    This should be a prime number close to a low multiple of slave hosts,
    e.g. for 3 nodes set this to 7
  </description>
</property>

</configuration>
```

Note: do NOT put these properties in hadoop-site.xml. That file (see below) should only contain properties characteristic to the cluster, and not to the job. Misplacing these properties leads to strange and difficult to debug problems - e.g. inability to specify the number of map/reduce tasks programmatically (Generator and Fetcher depend on this).

Edit the hadoop-site.xml configuration file.

```
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
  <name>fs.default.name</name>
  <value>???:9000</value>
  <description>
    The name of the default file system. Either the literal string
    "local" or a host:port for NDFS.
  </description>
</property>

<property>
  <name>mapred.job.tracker</name>
  <value>???:9001</value>
  <description>
    The host and port that the MapReduce job tracker runs at. If
    "local", then jobs are run in-process as a single map and
    reduce task.
  </description>
</property>

<property>
  <name>mapred.tasktracker.tasks.maximum</name>
  <value>2</value>
  <description>
    The maximum number of tasks that will be run simultaneously by
    a task tracker. This should be adjusted according to the heap size
    per task, the amount of RAM available, and CPU consumption of each task.
  </description>
</property>

<property>
  <name>mapred.child.java.opts</name>
  <value>-Xmx200m</value>
  <description>
    You can specify other Java options for each map or reduce task here,
    but most likely you will want to adjust the heap size.
  </description>
</property>

<property>
  <name>dfs.name.dir</name>
  <value>/nutch-0.9.0/filesystem/name</value>
</property>

<property>
  <name>dfs.data.dir</name>
  <value>/nutch-0.9.0/filesystem/data</value>
</property>

<property>
  <name>mapred.system.dir</name>
  <value>$/nutch-0.9.0/filesystem/mapreduce/system</value>
</property>

<property>
  <name>mapred.local.dir</name>
  <value>/nutch-0.9.0/filesystem/mapreduce/local</value>
</property>

<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>

</configuration>
```

Configure Nutch

Edit the nutch-site.xml file. Take the contents below and fill in the value tags.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
  <name>http.agent.name</name>
  <value></value>
  <description>HTTP 'User-Agent' request header. MUST NOT be empty -
  please set this to a single word uniquely related to your organization.

  NOTE: You should also check other related properties:

  http.robots.agents
  http.agent.description
  http.agent.url
  http.agent.email
  http.agent.version

  and set their values appropriately.

  </description>
</property>

<property>
  <name>http.agent.description</name>
  <value></value>
  <description>Further description of our bot- this text is used in
  the User-Agent header.  It appears in parenthesis after the agent name.
  </description>
</property>

<property>
  <name>http.agent.url</name>
  <value></value>
  <description>A URL to advertise in the User-Agent header.  This will
   appear in parenthesis after the agent name. Custom dictates that this
   should be a URL of a page explaining the purpose and behavior of this
   crawler.
  </description>
</property>

<property>
  <name>http.agent.email</name>
  <value></value>
  <description>An email address to advertise in the HTTP 'From' request
   header and User-Agent header. A good practice is to mangle this
   address (e.g. 'info at example dot com') to avoid spamming.
  </description>
</property>
</configuration>
```

Edit the crawl-urlfilter.txt file to edit the pattern of the urls that have to be fetched.

```
cd /nutch-0.9.0/search
vi conf/crawl-urlfilter.txt

change the line that reads:   +^http://([a-z0-9]*\.)*MY.DOMAIN.NAME/
to read:                      +^http://([a-z0-9]*\.)*org/
```

Or if downloading the whole internet is desired edit the nutch-site.xml file so that it includes the following property.

```
<property>
  <name>urlfilter.regex.file</name>
  <value>automaton-urlfilter.txt</value>
</property>
```

Distribute the code and the configuration

Copy the code and the configuration to the slaves

```
scp -r /nutch-0.9.0/search/* nutch@???:/nutch-0.9.0/search
```

Copy the keys to the slave machines

```
scp /nutch-0.9.0/home/.ssh/authorized_keys nutch@???:/nutch-0.9.0/home/.ssh/authorized_keys
```

Check if shhd is ready on the machines

```
ssh ???
hostname
```

Crawling

To crawl using dfs it is first necessary to format the namenode of Hadoop and then start it as well as all the datanode services.

Format the namenode

```
bin/hadoop namenode -format
```

Start all services on all machines.

```
bin/start-all.sh
```

To stop all of the servers use the following command, do not do this now:

```
bin/stop-all.sh
```

To start crawling from a few urls as seeds an url directory is made in which a seed file is put with some seed urls. This file is put into the hdfs, to check if hdfs has stored the directory use the dfs -ls option of hadoop.

```
mkdir urls
echo "http://lucene.apache.org" >> urls/seed
bin/hadoop dfs -put urls urls
bin/hadoop dfs -ls urls
```

Start an initial crawl

```
bin/nutch crawl urls -dir crawled -depth 3
```

On the masternode the progress and status can be viewed with a webbrowser. [http://localhost:50030/|http://localhost:50030/]

Searching

To search in the collected webpages the data that is now on the hdfs is best copied to the local filesystem for better performance. If an index becomes to large for one machine to handle, the index can be split and separate machines handle a part of the index. First we try to perform a search on one machine.

Install nutch for searching

Because the searching needs different settings for nutch than for crawling, the easiest thing to do is to make a sepperate folder for the nutch search part.

```
ssh root@???
mkdir /nutchsearch-0.9.0
chown nutch:users /nutchsearch-0.9.0
exit

ssh nutch@???
cp -Rv /nutch-build /nutchsearch-0.9.0/search
mkdir /nutchsearch-0.9.0/local
```

Configure

Edit the nutch-site.xml in the nutch search directory

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

  <property>
    <name>fs.default.name</name>
    <value>local</value>
  </property>

  <property>
    <name>searcher.dir</name>
    <value>/nutchsearch-0.9.0/local/crawled</value>
  </property>

</configuration>
```

Edit the hadoop-site.xml file and delete all the properties

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

</configuration>
```

Make a local index

Copy the data from dfs to the local filesystem.

```
bin/hadoop dfs -copyToLocal crawled /nutchsearch-0.9.0/local/
```

Test if all is configured properly

```
bin/nutch org.apache.nutch.searcher.NutchBean an
```

The last command should give a number of hits. If the query results in 0 hits there could be something wrong with the configuration, with the index or there are no documents containing the word. Try a few words, if all result in 0 hits most probably the configuration is wrong or the index is corrupt. The configuration problems I came across were: pointing to the wrong index directory and unintentionally using hadoop.

Enable the web search interface

Copy the war file to the tomcat directory

```
rm -rf usr/share/tomcat5/webapps/ROOT*
cp /nutchsearch-0.9.0/*.war /usr/share/tomcat5/webapps/ROOT.war
```

Copy the configuration to the tomcat directory

```
cp /nutchsearch-0.9.0/search/conf/* /usr/share/tomcat5/webapps/ROOT/WEB-INF/classes/
```

Start tomcat

```
/usr/share/tomcat5/bin/startup.sh
```

Open the search page in a webbrowser [http://localhost:8180/|http://localhost:8180/]

Distributed searching

Prepare the other machines that are going to host a part of the index.

```
ssh root@???
mkdir /nutchsearch-0.9.0
mkdir /nutchsearch-0.9.0/search
chown -R nutch:users /nutchsearch-0.9.0
exit
```

Copy the search install directory to other machines.

```
scp -r /nutchsearch-0.9.0/search nutch@???:/nutchsearch-0.9.0/search
```

Configure

Edit the nutch-site.xml so that the searcher.dir property points to the directory containing a search-servers.txt file with a list of ip adresses and ports. Put the ip adresses and ports in a search-servers.txt file in the conf directory:

```
x.x.x.1 9999
x.x.x.2 9999
x.x.x.3 9999
```

Edit the nutch-site.xml file:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

  <property>
    <name>fs.default.name</name>
    <value>local</value>
  </property>

  <property>
    <name>searcher.dir</name>
    <value>/nutchsearch-0.9.0/search/conf/</value>
  </property>

</configuration>
```

Split the index

With the current Nutch and Lucene code there is not a function to split one index into smaller indexes. There is some code published on a mailing list that does pretty much what is needed. [http://www.nabble.com/Lucene-index-manipulation-tools-tf2781692.html#a7760917]

Copy each part of the index to a different machine.

```
???
scp -R /nutchsearch-0.9.0/local/partX/crawled nutch@???:/nutchsearch-0.9.0/local/
```

Start the services

Startup the search services on all the machines that have a part of the index.

```
bin/nutch server 9999 /nutchsearch-0.9.0/local/crawled
```

Restart the master search node

```
/usr/share/tomcat5/bin/shutdown.sh
/usr/share/tomcat5/bin/startup.sh
```

Open the search page in a webbrowser [http://localhost:8180/|http://localhost:8180/]

Crawling more pages

To select links from the index and crawl for other pages there are a couple of nutch commands: generate, fetch and updatedb. The following bash script combines these, so that it can be started with just two parameters: the base directory of the data and the number of pages. Save this file as e.g. bin/fetch, if the data is in crawled01 than `bin/fetch crawled01 10000' selects 10000 links from the index and fetches them.

```
bin/nutch generate $1/crawldb $1/segments -topN $2
segment=`bin/hadoop dfs -ls crawled01/segments/ | tail -1 | grep -o [a-zA-Z0-9/]*`
bin/nutch fetch $segment
bin/nutch updatedb $1/crawldb $segment
```

To build a new index use the following script:

```
bin/hadoop dfs -rmr $1/indexes
bin/nutch invertlinks $1/linkdb $1/segments/*
bin/nutch index $1/indexes $1/crawldb $1/linkdb $1/segments/*
```

Copy the data to local and searching can be done on the new data.

Comments

Number of map reduce tasks

I noticed that the number of map and reduce task has an impact on the performance of Hadoop. Many times after crawling a lot of pages the nodes reported 'java.lang.OutOfMemoryError: Java heap space' errors, this happened also in the indexing part. Increasing the number of maps solved these problems, with an index that has over 200.000 pages I needed 306 maps in total over 3 machines. By setting the mapred.maps.tasks property in hadoop-site.xml to 99 (much higher than what is advised in other tutorials and in the hadoop-site.xml file) that problem is solved.

(ab) As noted above, do NOT set the number of map/reduce tasks in hadoop-site.xml, put them in mapred-default.xml instead.

See http://wiki.apache.org/lucene-hadoop/HowManyMapsAndReduces for more info about the number of map reduce tasks.

Error when putting a file to DFS

If you get a similar error:

```
put: java.io.IOException: failed to create file /user/nutch/.test.crc on client 127.0.0.1 because target-length
is 0, below MIN_REPLICATION (1)
```

it may mean you do not have enough disc space. It happened to me with 90MB disk space available, Nutch 0.9/Hadoop 0.12.2. See also mailing list message.