Absolute S/W Testing Notes

70k messages/second, 3kb payload

>>> 210\*60\*60\*8/1024

5906

~6gb/day

180gb/month

2.12TB/year

2-3ms per request using HttpServlet

300-500 requests/second, need 140 instances to get 70k/second. Script to scale over AWS to 100-200 machines. Not a big deal.

Try netty async generator still 2-3ms per request generation?

Netty start from here: <http://docs.jboss.org/netty/3.2/guide/html_single/index.html#d0e290>

The newer documentation for Netty 3.4:

<http://netty.io/docs/stable/guide/html/>

Goal: to write a client or server program using Netty with user defined messages.

Discard Server:

The simplest server consists of a 4 step setup:

1. Define a ctor which includes the port and maybe inet address for the server

public DiscardServer(int p){

this.port= p;

}

1. Define the main method which calls the run() method to run the server

new DiscardServer(8080).run();

1. Define the run method which calls ServerBootstrap to create the Server Socket Channels and the ChannelPipeline to register the Server Event handler.
   1. new ServerBootstrap(new NioServerSocketChannelFactory(Executors.newCachedThreadPool(), Executors.newCachedThreadPool()))
   2. server.setPipelineFactory(new ChannelPipelineFactory(){

public ChannelPipeline getPipeline(){

return Channels.pipeline(new DiscardServerHandler())

}

)

1. Define the Event handler, the Discard Server only accepts data so define only an upstream handler using SimpleChannelUpstreamHandler. If you need both upstream and downstream define SimpleChannelHandler. To define a channel handler you are given a ChannelHandlerContext and ChannelEvent through functions void handleDownstream(ChannelHandlerContext ctx, ChannelEvent ev) and void handleUpstream(ChannelHandlerContext ctx, ChannelEvent ev). Extend from SimpleChannelUpstreamHandler and implement 4 methods:
   1. public long getTransferredBytes(){

return numBytes;

}

* 1. public void handleUpstream(ChannelHandlerContext ctx, MessageEvent e){

super.handleUpstream(ctx,e);

}

* 1. public void messageReceived(ChannelHandlerContext ctx, MessageEvent e){

numBytes = numBytes+((ChannelBuffer)e.getMessage()).readableBytes();

}

* 1. public void exceptionCaught(ChannelHandlerContext ctx, ExceptionEvent e){}

To turn the DiscardServer into an EchoServer we have to modify the HandlerFunction in DiscardServerHandler and write back what we see in public void messageReceived(ContextHandler ctx, MessageEvent e). Add the line:

public void messageReceived(ContextHandler ctx, MessageEvent e){

numBytes = numBytes + ((ChannelBuffer)e.getMessage()).getReadable();

e.getChannel().write(e.getMessage());

}

**Hadoop Configuration**

Hadoop Versions: last stable trunk before different branches from eBay, Yahoo(1.0.X). Hadoop-0.20.205 contains security for different users using Kerberos and write file append(HBASE LOSE DATA FIX). Does not have MR2 or RAID. In production at various sites. Frozen as of 11/2011. Good for most applications. Past compatability with other older versions of Hadoop Components Hue, SQOOP, Flume, Hive, Hbase. Have to match components with Hadoop core release.

Hadoop 1.0.X: not in production yet. Contains new programming model for non mapreduce jobs such as MPI. <http://hadoop.apache.org/common/docs/r1.0.1/releasenotes.html> lists enhancements and bug fixes over 0.20.205.

Ignore these:

Hadoop-0.21: has append, RAID, symlinks, but no security.

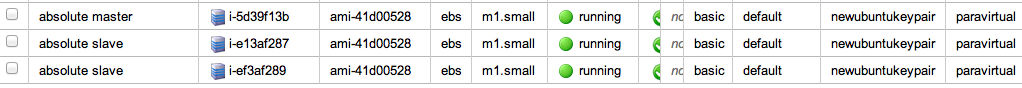
Hadoop-0.22:

Hadoop-0.23: eBay Distribution customized for eBay.

Hadoop POC Using hadoop-0.20.205 and/or Hadoop 1.0.x on Redhat 6.1 in Amazon AWS

Installing in cluster mode:

1. Start a medium Redhat instance to be master with Namenode and Region Server, start 3 Redhat instances, small is 1.7GB, each dameon takes 1GB. Might be too small.



1. create new user called redhat(do not use reserved users like hadoop or hdfs or else you will get strange errors) and group and su into this once logged into the master redhat instance

delete the password for user redhat

[root@ip-10-10-213-49 ~]# su redhat

[root@ip-10-28-116-108 ~]# adduser --system --shell /bin/bash --user-group --create-home redhat

[root@ip-10-28-116-108 ~]# passwd -d redhat

Removing password for user redhat.

passwd: Success

[root@ip-10-28-116-108 ~]# su redhat

[redhat@ip-10-28-116-108 root]$ cd

[redhat@ip-10-28-116-108 ~]$

1. create ssh keys

[hadoop@ip-10-10-213-49 ~]$ ssh-keygen -t dsa -P '' -f ~/.ssh/id\_dsa

[hadoop@ip-10-10-213-49 ~]$ cat ~/.ssh/id\_dsa.pub >> ~/.ssh/authorized\_keys

[hadoop@ip-10-10-213-49 ~]$ sudo chmod 600 .ssh/authorized\_keys

test using ssh localhost

1. format namenode: bin/hadoop namenode –format

Run in pseudo-distributed mode

edit hdfs-site.xml:

<?xml version="1.0"?>

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

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

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

</configuration>

edit core-site.xml:

<?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>hdfs://localhost:9000</value>

</property>

</configuration>

edit mapred-site.xml:

<?xml version="1.0"?>

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

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

<configuration>

<property>

<name>mapred.job.tracker</name>

<value>localhost:9001</value>

</property>

</configuration>

test hdfs access, make sure you can create and read directories in hdfs:

[hadoop@ip-10-10-213-49 hadoop-0.20.205.0]$ bin/hadoop fs -mkdir /user/$USER

[hadoop@ip-10-10-213-49 hadoop-0.20.205.0]$ bin/hadoop fs -ls /

Found 2 items

drwxr-xr-x - hadoop supergroup 0 2012-05-07 16:40 /tmp

drwxr-xr-x - hadoop supergroup 0 2012-05-07 16:40 /user

**Configure for fully distributed mode:**

All of the configuration files on all the nodes have to be the same and all the nodes need to have hadoop installed on them. Also need to have ssh installed with the id\_dsa.pub from the master in the .ssh/authorized\_keys file.

You should be able to ssh from master node to slave node using the aws public dns addresses.

1. hdfs-site.xml Modify the dfs.replication factor from 1 to 3
2. core-site.xml. Modify fs.default.name to use the AWS public IP addresses:
3. mapred-site.xml. Modify mapred.job.tracker address to use the public IP addresses. Which one do you add? The slaves? Add the b

To automate this setup you can use Apache Whirr

**Hbase Configuration:**

1. modify hdfs-site.xml

<property>

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

</property>

1. For pseudo-distributed mode, point HBase to HDFS in hdfs-site.xml using hbase.rootdir:

<property>

<name>hbase.rootdir</name> <value>hdfs://localhost:8020/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 datanode count. </description>

</property>

Note: don’t use localhost if you want to connect from a remote host. Use the Amazon ec2-xxx.xx.xxx address.

1. Hbase fully distributed configuration requires hdfs-site mods:

<property>

<name>hbase.rootdir</name> <value>hdfs://namenode.example.org:8020/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>

1. Modify conf/regionservers
2. Zookeeper setup using hdfs-site.xml. Note the default port has changed from 2181 to 2222.

<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>

1. Hbase Client configuration, hbase-env.sh add HADOOP\_CONF\_DIR under HBASE\_CLASSPATH environment variable or add copy of hdfs-site.xml or symlink to hdfs-site.xml under ${HBASE\_HOME}/conf
2. Test HDFS by using start-hdfs.sh, you don’t need to do start-all.sh because we don’t need M/R for Hbase. Make sure this works in distributed mode first.

**Hbase:**

Uses HDFS as reliable storage layer. Region servers manage data. Talk to region server like where client talks to name node which talks to data node. Zookeeper new component, coordinates the cluster.

Autosharding, at table which is a sorted list or rows. These tables are split across regions.

**Bigtop Installation: this is a more efficient way to proceed because all of the components are in the distribution including Hive and HDFS. Less work in downloading compatible releases.**

1. Create user hadoop with group hadoop, install the bigtop repo under this user or root. Note: bigtop creates users hdfs and hbase for you when you install hadoop and hbase. Have problem running pi example with IO permissions problem
2. Create user redhat which is not part of group hadoop;

[root@ip-10-28-222-174 ~]# adduser --system --shell /bin/bash --user-group --create-home redhat

[root@ip-10-28-222-174 ~]# su redhat

[redhat@ip-10-28-222-174 root]$ cd

[redhat@ip-10-28-222-174 ~]$

OK this works, 2) cant use reserved word hadoop, probably being used for something elase.

Install SQOOP and test on MSSQL database. Setup connection driver, create tables and download into HBASE/HDFS