Highbase Installation Guide

Installation guide for beta-0.8

Author: Fernando Ipar

fipar@seriema-systems.com

Revision: 101

Last update: Jun 15, 2008

Status

This document is still a draft and might be incomplete in it's coverage. The goal is to have complete and revised document by the time the cluster gets to 1.0

Introduction

The goal of this document is to help System Administrators setup a highly available MySQL system using the scripts that make up the highbase project.

The audience is expected to have some basic skills in shell usage and MySQL administration.

I'll start off by covering the requisites checklist (what's verified by the pre-install.sh script on the automatic installation) and then move onto the prerequisite installation part.

Finally, I'll describe the highbase.conf file and explain how to start and stop the cluster.

Prerequisites

The following is a checklist of what you'll need in order to run highbase. If you can make the cluster run without any of these components, please let me know, using the e-mail address found in the cover page.

- MySQL, with internal replication configured between Master and Slave host. Development is currently tested on version 5 of the engine, but the code includes compatibility for versions 3.X and 4.X.
- The mysql and mysqladmin client binaries
- bash 2 or greater
- syslog
- perl
- gcc and make
- ssh
- usleep or bc
- smbclient, **only if you want netbios alerts**

Unfortunately, I don't have the minimal versions required for perl, gcc and make. If you find out that your installed version doesn't work, please let me know.

The system is known to work on **Fedora Core 5** and **6** and in **CentOS 4**, with **MySQL 4** and **5** (support for 3.23 is included in the code, but I haven't tested that in years and I won't be testing it anymore). Work is under way for testing under Ubuntu 7.

Package installation

To begin the installation process, extract the **highbase-<release>.tar.bz2** into a directory of your choice. The **/usr/highbase** directory is the one I've used in my test systems, and the default directory, but it's by no means mandatory. You'll need to set the environment variable **HIGHBASE_HOME** to point to the place where you untar the files.

External packages

Next, you'll need to install fping, fake and mysql-monitor, according to these instructions:

fping

Change directory to the place where you untarred the highbase.tar.bz2 file. All the external packages are found in the extern directory. Change to that directory and type:

```
#> tar xzvf fping*gz && rm -f fping*gz && cd fping* && ./configure && make && make check && make install && make clean
```

This should run with no errors.

fake

Change directory to extern/ again, and type:

```
#>
```

Once again, this should run with no errors.

mysql.monitor

Jeff Buchbinder contributed many things to the source tree (like removing all references to the name mysql-ha), including a new version of the MySQL monitoring routine written in C, with improved performance.

Both the slave and master routine expect this program to be installed at **$HIGHBASE_HOME**, so in order to do this, just:

```
#> cd src/mysql-monitor; make; cp mysql-monitor $HIGHBASE_HOME; cd ../..
```

NOTE: In order to build the C monitor, you'll need to install MySQL's C libraries for your systems. In Red Hat based systems, theses are usually called mysql-devel. In Ubuntu, these are called libmysqlclient15-dev

Cluster installation

At this point, you need to populate the /etc/highbase.conf file.

The cluster configuration file is sourced by our shell scripts, so you should leave no blank space between a configuration key (i.e. 'CLUSTER_IP') and it's value.

We'll review each parameter providing a description and a default value.

Parameter	*Description*	*Default value*
CLUSTER_IP	It's the public IP that is shared between master and slave nodes. This is the unique service IP provided to the outside world.	None
CLUSTER_NETMASK	The network mask associated with the previous IP address	None
CLUSTER_BROADCAST	The broadcast address associated with the previous IP address	None
CLUSTER_DEVICE	The device to which to attach CLUSTER_IP	eth0
MYSQL_USER	The MySQL user used for service	replicator

Parameter	*Description*	*Default value*
	verification. The mysql.monitor connects to the other node using this user.	
MYSQL_PASSWORD	The password associated with the previous user	replicatorpwd
MYSQL_DATABASE	The database we verify upon testing the connection	test
MASTER_SLEEP_TIME	The amount of seconds the master routine sleeps between rounds	60
SLAVE_SLEEP_TIME	The same, for the slave routine	60
SSH_PATIENCE	Timeout, in seconds, for remote commands executed through ssh	40
MONITOR_PATIENCE	Timeout in seconds for mysql.monitor	20
MONITOR_CHK_THRESHOLD	If mysql.monitor fails, we wait this amount of time and perform a second test before marking the service as down. If no units are specified, the time is interpreted as seconds. Otherwise, you can specify ms (milliseconds) and us (microseconds).	20
MYSQL_KILL_WAIT	Amount of time we wait before checking after running mysql_kill. mysql_kill kills every MySQL process except for the replication thread, in case the server is not responding because it's choked. The units are handled in the same way as MONITOR_CHK_THRESHOLD,	60
MYSQL_RESTART_WAIT	The same, for mysql_restart. mysql_restart attempts to restart mysqld.	60
ATTEMPT_KILL	This boolean flag controls the behavior of the slave routine. If set to 1, it'll attempt to run mysql_kill on the remote host before doing a mysql_restart and finally a takeover. If set to the default, it'll go straight to the mysql_restart.	0
FPING_ATTEMPTS	When we decide the master is gone, we try to fping it to see if the machine is running. We try ATTEMPTS times.	3
SLAVE	IP or host name for the slave node	slave-node
SIG_KILL_WAIT	This is used on the takeover, and it's the amount of time we wait after issuing a	5

Parameter	*Description*	*Default value*
	SIGTERM on any remaining mysql processes and before issuing a SIGKILL. Keep in mind that the slave is waiting for us to finish the failover and won't start the service until we do. The units are handled in the same way as MONITOR_CHK_THRESHOLD.	
DB_USER	This is the root user for the MySQL database server	root
DB_PASSWORD	The associated password	rootpwd
NOTIFY_EMAIL	The email address where to send notifications	root@łocalhost

Fake configuration

If you installed fake properly, you'll find the **/etc/fake** directory on your system, and inside, the **instance_config** subdirectory.

There, you must create a file named **CLUSTER_IP**.cfg, with the following contents:

------ BEGIN **CLUSTER_IP**.cfg ----

SPOOF_IP=**CLUSTER_IP**

SPOOF_NETMASK=**CLUSTER_NETMASK**

SPOOF_BROADCAST=**CLUSTER_BROADCAST**

TARGET_INTERFACE=**CLUSTER_DEVICE**

------ END **CLUSTER_IP**.cfg ----

You can also create this file by issuing the following commands:

```
#> cd /usr/highbase
#> export HIGHBASE_HOME=/usr/highbase
#> ./setup_fake.sh
```

Configure passwordless ssh

In order to work, the scripts need passwordless ssh working for either the root user or some unprivileged user between both nodes. The unprivileged user account is user selectable, but we use highbase in our systems and it's a nice name for coherence. You must create a file named 'ssh_user' under the HIGHBASE_HOME directory with a single line containing the user name you'll use for passwordless ssh.

In order to setup passwordless ssh between your nodes, create a private/public key pair on each node with the following command[1]:

#> ssh-keygen -t dsa

SECURITY CONSIDERATION:

It's up to you to decide if you want to protect the private key with a passphrase or not. The cluster uses the ssh-agent at startup, so if you decide to use a passphrase, you'll need to enter it every time you start the cluster.

Next, copy the generated **$HOME/.ssh/id_dsa.pub** file to the other node, and add it to the $HOME/.ssh/authorized_keys file, where $HOME is the cluster user (root, highbase or the user you chose) home directory.

1 If you're using the highbase or another unprivileged account, su – to that account before issuing this commands

Repeat this procedure for the other node.

You can improve security by including '**from=<host>'** before the ssh-dsa string in the authorized_keys file.

Before running the cluster for the first time, remember to connect from one node to another using the cluster user, in order to accept the host key. Otherwise, the cluster will always end up killing remote commands due to timeout.

Configure sudo

If you decide to setup passwordless ssh using an underprivileged account, **then your system must have sudo installed**, and you must configure this properly.

Create a file in $HIGHBASE_HOME called sudo_prefix with the text 'sudo ' (**that's WITH A SPACE after the last letter**).

Finally, modify /etc/sudoers by adding a line similar to the following:

```
highbase   ALL=NOPASSWD:/usr/local/sbin/fping, /sbin/fuser, /bin/ps, /bin/kill, /etc/init.d/mysqld, /sbin/shutdown, /usr/bin/fake, /sbin/ifconfig
```

To obtain these path names, **cd to $HIGHBASE_HOME** and **as root** source the **compat.sh** script and execute the **get_sudoers_line** function. If the cluster doesn't work with these values, then you'll have to manually edit compat.sh and modify the default entries for these programs.

Configure replication

Set up replication according to the instructions found on MySQL's manual:

http://dev.mysql.com/doc/refman/5.0/en/replication.html

Configure MySQL access

In order to check the service health, highbase uses the mysql.monitor script from the mon package (available from ftp.kernel.org and it's mirrors, please use the mirrors in case you want to download mon or use an updated version of mysql.monitor). For this script to work, you'll need to issue the following statements in MySQL:

```
GRANT SELECT ON <DB>.* TO '<USER>'@'<HOST>' IDENTIFIED BY
'<PASSWORD>'
```

where:

<DB> is **MYSQL_DATABASE**

'<USER>' is **MYSQL_USER**.

'<HOST'> is the **CLUSTER_IP** when you issue this on the slave node, and the slave node's IP when you issue this on the master.

'<PASSWORD>' is **MYSQL_PASSWORD**.

Configure the node role

You must now tell highbase which role it should assume on which node. For this purpose, we use the role.include file. This must be a copy of either slave.include or master.include, depending on the node you're installing.

Here's an example for the master node:

#> ln master.include role.incude

Install the rc-script

Copy the rc-script from $HIGHBASE_HOME to /etc/init.d and chmod it 755

You can now start the service using this script. We're still working on issues when the script is started with the rc-script and the private key has been protected with a passphrase, so you might want to start the cluster using ./configurator.sh directly from $HIGHBASE_HOME while we work it out.

Run the cluster

- Start highbase on the master node, by either:
  - service highbase start (in redhat/fedora)
  - /usr/highbase/configurator.sh (better for testing, since you'll be hooked up to the output
- Start highbase on the slave node.

Test failover scenarios and report bugs to highbase-devel@lists.sourceforge.net