MULTI-POOL

ice00 of ice team[1]

contents

abstract

Multi-pool is an open source software for helping in setting up MPOS pools into a server. There are many articles into the web that describe how to setting up a pool, but they lack in some details:

- they did not discuss about security aspect (so all is done with *root* user)
- they discuss in one pool setup, but what happen if you want more pools in a given server?

After my 2 year experiences with this software I recommend to use this approach even if you need only one pool into your server.

[1] *Ice Team is on http://iceteam.altervista.org*

1 introduction

If you want to install a MPOS pool into your server there are some steps you need to achieve:

- Install all the packages that the software of pool need for working (for example boost library for C++ compilation, python for stratum, mysql for DB and so on).
- Compile the daemon of the coin. It is than used for having the pool wallet.
- Install and configure a stratum instance for let the miners to mine into the pool.
- Install and configure the MPOS pool.

You have so many peaces of software to install into the server, at least 4 (daemon, wallet, stratum and MPOS) and this is to repeat for every coin you install into the server.

Without a little of order you will get lost in all that files, above all when you have to modify one configuration files after months you have set up a pool!

So, if you want to use *multi-pool* software there is only 2 prerequisites:

1. The server should be a *deb* based, so for example Debian and Ubuntu.
2. The coin source must be in a *git* repositories.

If you did not have one of the above, you need to go by hand.

2 structure

We already say that *multi-pool* use a fixed structure that help in managing all the stuff, so lets see it.

We use the classical Unix approach to put same kind of file in same directories so the permission are shared, but with the exception that we not start from **/** directory, but we instead put it inside **/opt/multi-pool/**[1].

This help us in have all in one point but take separate different stuff (so not mix MPOS software and stratum software for example).

Inside that directory there is this three (we assume to have 3 coins: *bitcoin*, *litecoin* and *ppcoin* to manage):

- **bin**: it contains all the daemon binaries (usually they are called like *coin name* terminated with *d*, so we can have all binaries together). In our example we should have those files in it: *bitcoind*, *litecoind*, *ppcoind*. This directory and all the files have special permission for security reasons, but this will be see later.
- **src**: it contains all the coin source code. Here there will be one directory for each coin, so expect to have something like: *bitcoin*, *srclitecoin*, *ppcoinsrc*[2]
- **script**: it contains all the script used by *multi-pool* software.

1 This is a prefix that you can change if you want by modify the *startup.sh* script
2 The name is the one that are into the *git* archive

- **wallet**: this is a directory were coin wallet and blockchain are located. There is one directory for each coin and we use the coin prefix for them. So expect to have the directories *btc*, *ltc* and *ppc*. Here we use special permission like the *bin* directory.
- **coin**: what happen if you want to modify a configuration file of a coin? Simple: go here and you will find a link that points in the correct position. So here we have one directory for coin and inside there are the three important configuration file: the daemon configuration, the stratum configuration, the MPOS configuration.
- **mpos**: here we have the installation of mpos for each coin, so you will find one directory for coin, as we see before.
- **stratum**: here we have the installation of stratum for each coin, so you will find one directory for coin, as we see before.

Assuming the above coins example, our directories are:

```
/opt/multi-pool/
├── bin
├── src
├── script
├── wallet
│   ├── btc
│   ├── ltc
│   └── ppc
├── coin
│   ├── btc
│   ├── ltc
│   └── ppc
├── mpos
│   ├── btc
│   ├── ltc
│   └── ppc
└── stratum
    ├── btc
    ├── ltc
    └── ppc
```

3 permission

One important point to keep in mind about the pool setup is the permission to gives to the various software that are running. We don't discuss about server security (I assume that you have installed a secure SSH connection for managing the server and setup fail2ban to stop access attempts at least), but we concentrate only in pools. The first question to ask is: is it necessary to run a given software as root? The answer is simple: if a software **don't need a root permission to run**, then **don't run it as root**! This is valid for all, in our case we apply it for the coin daemon as it has two possible fault:

1. Even if the code is tried (it is all based onto Bitcoin original source code), there could be some bugs not jet fount that can be used by an hacker for taking the control over the daemon.

2. The author of the coin can have modify the source code for inserting a backdoor to let him access your server.

Even if all the two scenario seems remote, well the second one is already happened!

If an hacker can access to your server or can have access to your daemon, he can move your coins from your wallet to his wallet[3]. To prevent that a coin daemon performs one of the two above actions we need to run it not as *root* user. The way *multi-pool* implement this is by creation an user and a group that is called *coind* (coin daemon). That user is abstract as we did not need to access with it in the server. Then we need to set the daemon (example *bitcoind*) as of user *coind* and group *coind*. This is done with those instructions:

```
chown coind bitcoind
chgrp coind bitcoind
```

This means that only user *coind* or of group *coind* are the owner. However for making the final trick, we need to be sure that only *coind* user run it. This is achieved by forcing the file to be *setui*! Normally you think that **setui** is for forcing a program to be executed as *root*, but this is true when that file is of *root* user and you are not a root user. This is the opposite; you are a root user but you force the program to be executed as a normal *coind* user. This operation is executed by this instruction that force *setui* on:

```
chmod xx bitcoind
```

Now when you execute a coin daemon it will be executed as normal *coind* user. So, this implied that you need to have the wallet directory of *coind* to user/group too, because if they are of root user, the daemon can't write to it. Can we make more better? Yes, we can create a different user for every coin, like *btccoind*, *ltccoind* and *ppccoind*, but unfortunately this is not jet coded inside *multi-pool*. At the actual point *stratum* and *mpos* are executed as root user as alternative user usage is not jet implemented.

Another point that I planned to change but it is not jet done, it that daemon compilation is done by *root* user, even if then daemon is executed by *coind* user. What is wrong with this? Simple: potentially the programmer of the coin (in this case an hacker) could change the *configure* script of *automake* to execute instructions not related to compilation that can be executed when we run the compilation task!

Last point in security is related to Mysql database permission. Here we create a new user and a new password for every coin. This is not a big security for hackers, as password is not random, but a fixed one. It is useful to prevent that for error we manage the database of another coin (mysql gives error as we log in with a user that cannot manage a different database).

3 I had proposed in the past a way to avoid that an hacker can move coins from pool wallet to his wallet. This procedure need that a wallet is marked as special (pool one) when the wallet is created, then the coins moved from pool wallet to other wallet are subject to a predefined time of maturity before they can be spent. That time is configurable and could be, for example 1 week, 10 days, and so on. Now if a pool owner find that his wallet was compromised as his coins were moved to another wallet, he can use a function of revoke that transaction. That function take a survey onto all the addresses where that wallet has sent coins in the past and if the majority of them accept the survey at the end of maturity time, then an automatic transaction moves the coin from hacker wallet to pool wallet. If a wallet is compromised, the pool owner can inform his miners by pool message about the problem and have that them all, except for the hacker, accept to revoke that transaction. All this need a proper implementation and a choose of maturity times that not penalize miner. For example 1 days could be good time if the pool owner monitors pool wallet every day.

4 startup

For using the multi-pool software you have to download it (we use the temp directory):

```
cd /tmp
git clone https://github.com/ice00/multi-pool.git
cd multi-pool/
```

Now you should concentrate in the *startup.sh* script. If you don't want to use the prefix */opt/multi-pool/* you had to modify the script and change the *BASE_DIR* variable. As soon as you execute the script it performs:

- It creates the directories structures as we describe before.
- It copies the scripts inside the *script* directory.
- It creates the user *coind* and the group *coind*.
- It assigns the group and user **coind** to *wallet* and *bin* directories.
- It installs all the DEB packages that are not installed in a standard server installation that are needed by all the other steps (that list should be updated by hand, for example boost library tend to changes and more recent coin will use it).
- It installs some components of stratum that can be shared between each coin[4]. They are shared because the installation move them inside binary directories of the server.

5 coin

This is a procedure to apply for every coin and most of the task is done by *makecoin.sh* script. Actually you have to modify by hand this script even if it was planned (but not actually implemented) to take that information by a file passed as parameter. The information to change are:

- **COIN_NAME**: this is the coin name.
- **COIN_PREF_LOW**: this is the coin prefix, like *btn*, *ltc* and *ppc* in lowercase.
- **COIN_PREF_HIGH**: this is the coin prefix, like *BTN*, *LTC* and *PPC* in uppercase.
- **BIN_DAEMON**: this is the name of the daemon binary program.
- **SRC_PATH**: this is the name of the directory that contains the source code.
- **SRC_GITHUB**: this is the git repository with the coin source code.
- **COIN_CONFIG**: this is the name of configuration file used by the daemon

lets see some examples:

4 https://github.com/Tydus/litecoin_scrypt.git, https://github.com/ahmedbodi/stratum.git, https://github.com/scr34m/vertcoin_scrypt

- **Bitcoin**:

```
COIN\_NAME=bitcoin
COIN\_PREF\_LOW=btc
COIN\_PREF\_HIGH=BTC
BIN\_DAEMON=bitcoind
SRC\_PATH=bitcoin
SRC\_GITHUB=https://github.com/bitcoin/bitcoin
COIN\_CONFIG=bitcoin.conf
```

- **Litecoin**:

```
COIN\_NAME=litecoid
COIN\_PREF\_LOW=ltc
COIN\_PREF\_HIGH=LTC
BIN\_DAEMON=litecoind
SRC\_PATH=litecoin
SRC\_GITHUB=https://github.com/litecoin-project/litecoin
COIN\_CONFIG=litecoin.conf
```

- **Ppcoin**:

```
COIN\_NAME=ppccoin
COIN\_PREF\_LOW=ppc
COIN\_PREF\_HIGH=PPC
BIN\_DAEMON=
SRC\_PATH=
SRC\_GITHUB=https://github.com/ppcoin/ppcoin
COIN\_CONFIG=
```

Now you can execute the script as root that is located onto:

```
/opt/multi-pool/script/makecoin.sh
```

This script will do those actions (here *xxx*=coin shortcut, like *btn*, *ltc*, *ppc*):

- It downloads the source code of the coin inside the */opt/multi-pool/src* directory.
- It creates the directory *obj* inside the *src* directory of the coin (this is need as sometimes it in not present, but used by compilation).
- It gives executable permission to *build_detect_platform* as sometimes it is not set and it is used by compilation.
- It compiles the source.
- It copy the daemon to */opt/multi-pool/bin* directory and make it of user *coind* and group *coind* with *setui* bit set.
- It creates the directory */opt/multi-pool/stratum/xxx* and here it downloads the stratum software[5].
- It creates the directory */opt/multi-pool/mpos/xxx* and here it download the MPOS software[6].

5 https://github.com/ahmedbodi/stratum-mining.git
6 https://github.com/MPOS/php-mpos

- It changes the permission of some MPOS folder to be of user *www-data* as they are to be processed by Apache server (without this a user that look at pool page will find an empty screen).
- It creates a database called with the coin prefix (*xxx*) and a password that is three times it, then adds permission to that user for the database[7].
- It populates the database using the sql script inside the MPOS installation of that coin[8].
- It creates the directory */opt/multi-pool/wallet/xxx*, setting the user and group with *coind*, then create a preconfigured empty coin file.
- It creates links inside the */opt/multi-pool/coin/xxx* directory to the configuration file in Wallet, MPOS and stratum directory. Did you need to modify a script configuration file? Don't lose time in going inside the many directories of server but go directly here.
- It creates a link inside */var/www/xxx* to the MPOS website index directory. Now you have populate the Apache files only by making a link to MPOS[9].
- It adds entries about this coin in the special crons script of *multi-pool* and generate and activate the script for control the coin[10].

6 script

One of the script created by *makecoin.sh* is the startup one that is inserted inside */etc/init.d*. This allow that the daemon and stratum processes are automatically executed at startup.

The first thing made by the script is to run the daemon and after 2 seconds it starts the stratum too.

Here there is the first problem: if we start the daemon, after what time will we start the stratum process? Well, daemon startup time depends from coin to coin and even from your wallet status (a wallet with 10 transactions open soon, a wallet with 150.000 transactions need 5 minutes to open). We also need that the blockchain is full sync before starting the stratum, otherwise miner mine to old blocks and in best case you will have all orphan blocks and in worst case you will make a coin fork if your pool control more that 50% of network.

So the best way is that you run again the coin startup script by hand when you see that daemon is in good state after you have reboot the server or make a maintenance that stop the process[11].

```
#!/bin/bash
#
# $COIN_PREF_LOW - $COIN_NAME - $BASE_DIR
RETVAL=0;
```

7 During this phase it asks for *mysql* password of *root* user
8 During this phase it asks for the coin password (tree times the coin prefix)
9 In my opinion this is the best approach even for one pool only. I don't like to see mixed code inside /var/www (so MPOS internal logic in PHP and pool user PHP pages).
10 In next section they will be described
11 Don't be angly: we have a way to pick up automatically the stratum process after a reasonable time , but we see it later.

```
start() {
echo \"Starting $COIN_PREF_HIGH\"
$BASE_DIR/bin/$BIN_DAEMON -datadir=$BASE_DIR/wallet/$COIN_PREF_LOW/
&
sleep 2
cd $BASE_DIR/stratum/$COIN_PREF_LOW/stratum-mining/
twistd -y launcher.tac &
}

stop() {
echo \"Stopping $COIN_PREF_HIGH\"
kill -15 \$(cat $BASE_DIR/stratum/$COIN_PREF_LOW/stratum-mining/twistd.pid)
sleep 1
kill -9 \$(cat $BASE_DIR/stratum/$COIN_PREF_LOW/stratum-mining/twistd.pid)
$BASE_DIR/bin/$BIN_DAEMON -datadir=$BASE_DIR/wallet/$COIN_PREF_LOW/ stop
}

restart() {
stop
start
}

case \"$1\" in
start)
  start
;;
stop)
  stop
;;
restart)
  restart
;;
*)

echo $\"Usage: $0 {start|stop|restart}\"
exit 1
esac

exit $RETVAL
```

The name of the script is coin_ followed by the coin prefix, so here an example of how to start and stop the bitcoin (btc) coin:

```
/etc/init.d/coin_btc start
/etc/init.d/coin_btc stop
```

If you want to manually add a script for automatic startup coin *xxx* you have to do:

```
update-rc.d coin_xxx defaults
```

Multi-pool has other 3 important scripts inserted into */opt/multi-pool/info*:

1. crons_fast.sh
2. crons_slow.sh

3. crons.sh

The content of each script is automatically added by the *makecoin.sh* procedure. What are those script for?

The answer is simple: MPOS need to run some of his scripts at a given rate for updating his internal processes: find blocks, run payout, give maintenance and so on. Historically there was the *run_crons.sh* script to run but it is deprecated as the best way is to call some more specialized script[12]. MPOS manual says that *run-statistics.sh* and *run-maintenance.sh* should be ran often, while *run-payout.sh* can be ran less often.

For this we populate *crons_fast.sh* with *run-statistics.sh* and *run-maintenance.sh* and *crons_slow.sh* with *run-payout.sh*. Here the contents of that files for bitcoin, litecoin and ppcoin coins:

```
#! /bin/sh

# multi-pool crons script for fast operations (like 1 minute)
#
# by Ice00
#
# (C) 2014 Ice Team
#
# example of one entry
# /opt/multi-pool/mpos/xxx/run-statistics.sh -d XXX
# /opt/multi-pool/mpos/xxx/cronjobs/run-maintenance.sh -d XXX
nice -n 19 /opt/multi-pool/mpos/btc/run-statistics.sh -d BTC
nice -n 19 /opt/multi-pool/mpos/btc/cronjobs/run-maintenance.sh -d BTC
nice -n 19 /opt/multi-pool/mpos/ltc/run-statistics.sh -d LTC
nice -n 19 /opt/multi-pool/mpos/ltc/cronjobs/run-maintenance.sh -d LTC
nice -n 19 /opt/multi-pool/mpos/ppc/run-statistics.sh -d PPC
nice -n 19 /opt/multi-pool/mpos/ppc/cronjobs/run-maintenance.sh -d PPC
```

If you look at the above or below file, we use the -d option for specify a custom directory for each coin to use. If you have set up one pool maybe you did not need to use this, but with more pools running together it is necessary to avoid that two scripts run in the same temporally directory to avoid problems.

```
#! /bin/sh

# multi-pool crons script for slow task (eg. every 2 minute)
#
# by Ice00
#
# (C) 2014 Ice Team
#
# example for coin xxx
# /opt/multi-pool/mpos/xxx/cronjobs/run-payout.sh -d XXX
nice -n 19 /opt/multi-pool/mpos/btc/cronjobs/run-payout.sh -d BTC
nice -n 19 /opt/multi-pool/mpos/ltc/cronjobs/run-payout.sh -d LTC
nice -n 19 /opt/multi-pool/mpos/ppc/cronjobs/run-payout.sh -d PPC
```

The next file is *crons.sh* that contains a strange rules: we run deprecated *crons.sh* with a forcing action[13]!

12 It is deprecated, but we use it for recovery task.

13 We are not out of mind, in the next sections we will describe why it is used in this form.

```
#! /bin/sh

# multi-pool crons script  (force a running)
#
# by Ice00
#
# (C) 2014 Ice Team
#
# example for coin xxx
#
# /opt/multi-pool/mpos/xxx/cronjobs/run-crons.sh -f
nice -n 19 /opt/multi-pool/mpos/btc/cronjobs/run-crons.sh -f
nice -n 19 /opt/multi-pool/mpos/ltc/cronjobs/run-crons.sh -f
nice -n 19 /opt/multi-pool/mpos/ppc/cronjobs/run-crons.sh -f
```

The last file is a special one: *coin_start.sh*. It is described in next sections but essentially is is used if you want to startup all coins by hand:

```
#! /bin/sh

# multi-pool crons script
#
# by Ice00
#
# (C) 2014 Ice Team
/etc/init.d/coin_btc start
/etc/init.d/coin_ltc start
/etc/init.d/coin_ppc start
```

7 crons

If we want that the MPOS software run correctly, we need to run periodically the two scripts we have created: *crons_fast.sh* and *crons_slow.sh*. The easy way to do this is to add two lines inside *crontab*[14] that are:

```
*/3 * * * * /opt/multi-pool/script/crons_slow.sh
*/1 * * * * /opt/multi-pool/script/crons_fast.sh
```

The first command runs every minute our *crons_fast.sh* so MPOS has the *run-statistic.sh* and *run-mainternance.sh* executed. The second command runs every 3 minutes our *crons_slow.sh* so MPOS *run-payout.sh* is executed.

Ok, now the coin daemon start up automatically at server start up (with *coind_xxx*), while stratum is not stared (unless in near to empty wallet and little blockchains). I suppose that you start again *coind_xxx* to pick up stratum when it is the right moment and all goes good as MPOS scripts are correctly called[15].

Now what's happen if for some problems the daemon or stratum process died unexpectedly? Well, if you don't start them up soon, you will lose many miners as they went to other pools. For this there is a solution: add a new entry in *crontab* to force coin restart. I put 54 minutes that means:

- when server starts, coin daemon is started

14 Use the command **crontabs -e** for this

15 If you don't do this, even if miners find blocks, the pool did not show anything until you call that MPOS scripts.

- after 57 minutes a restart is forced, so if daemon is up, stratum goes up too this time, otherwise it will be go after other 54 minutes.

```
*/57 * * * * /opt/multi-pool/script/coin_start.sh
```

Is this useful? Yes.

If something goes wrong for sure it happens where you are away (sleeping or working) so this granted that at least in less that 2 hours your pool will going up automatically (and 2 hours is better than 12 hours).

Can we do better? I think not much. If you reduce the time for example to 15 or 20 minutes that means that stratum will be called 15 or 20 minutes after the daemon is up again and this time could be too less for having the blockchain already synched in some situation.

Have we finished? No, I think not.

MPOS is a good software but pool experience teach that the possible errors that MPOS provide for sure they happen first or after and some errors block one of its process until you make it goes on by forcing it (or you need to modify database). In my experience a forcing onto an errors can let you lose some coins (for example by a double payment or paying orphan blocks), but miner will find that the pool is very reliable (it payouts always and statistics are updated). Instead if an error is not recoverable by forcing, well this did not cause other troubles to pool until you go to fix it by hand.

So we call *crons.sh* (a force onto MPOS script) every 59 minutes by adding this line in *crontabs*:

```
*/59 * * * * /opt/multi-pool/script/crons.sh
```

I probably lost coins into this forcing procedure but recovery from MPOS errors every 57 minutes let me see that an errors happened during the night was resolved automatically after some hours at least in 90% of cases.

8 website

With the kind of MPOS managing over Apache applied by *multi-pool* your pools will be accessed by[16]:

- http://multi-pool.info/btc
- http://multi-pool.info/ltc
- http://multi-pool.info/ppc

This could be acceptable, but a better solution is to create multiple site configuration onto Apache. If you register 3 DNS for your domains we could have:

- http://btc.multi-pool.info
- http://ltc.multi-pool.info
- http://ppc.multi-pool.info

This is for sure more convenient for a user, so you should consider to create multi sites configuration in Apache. First, we assume you are able to register new DNS by your provider. Second for each pool you need to create a new site Apache file and add the path to use for that given DNS.

Lets see an example for bitcoin coin. You should copy the default file:

16 The axample is for multi-pool.info domain and for bitcoin, litecoin and ppcoin.

```
cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/btc.conf
```

In his contents, you should popolate with:

```
<VirtualHost *:80>
    ServerAdmin info@multi-pool.info
    ServerName multi-pool.info
    ServerAlias btc.multi-pool.info
    DocumentRoot /var/www/btc/
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Finally, enable that new site and restart Apache sever:

```
service apache2 restart
a2ensite btc.conf
```

9 configuration

Unfortunately due to the high modification rate that MPOS does into configuration file the planning features of having an automatically configuration file creation was not implemented (even for stratum configuration file that is more stable). However there are some hint to follow.

Inside *config.py* it is very reccomented to use a custom name for stratum (the example is for bitcoin coin):

```
STRATUM_MINING_PROCESS_NAME= 'btc-twistd-stratum-mining'
```

Even for memcache is better to change the name depending from coin (example is for bitcoin).

```
MEMCACHE_PREFIX = "btc_stratum_" # Prefix for keys
```

Another important point is to create very random *SALT* and *SALTY* password to use in *config.py* and *global.ini.php*.

10 maintenance

From time to time you will need to enter in MPOS database of a coin for resolving some errors. This is done (for example for the *bitcoin* coin) with:

```
mysql -u btc -p btc
```

It will ask to enter the password (it is *btcbtcbtc* as our *multi-pool* rule).

I want to give you some commands that I use a lot in resolving MPOS error. If you get **E0062 Block has no share_id, not running payouts**, the first thing I do is to run this:

```
UPDATE BLOCKS SET ACCOUNTED=1 WHERE CONFIRMATION=-1 AND ACCOUNTED=0;
```

Essentially it take orphan blocks and make them accounted and as orphans block are not payed, so we did not need to find the exactly *share_id* to use.

If you get **E0078 RPC method did not return 200 OK** must of the case is a payout not processed (maybe it has too much coins in the transaction). You will need to perform a manual payout, so you need to look in the MPOS

transaction the name of user with failed sending status (you see it as there is a whole in the table). Once you have the name (for example *pippo*), do this (*xx* is the result of the first query):

```
SELECT ID FROM ACCOUNTS WHERE USERNAME='pippo';
SELECT COIN_ADDRESS FROM COIN_ADDRESSES WHERE ACCOUNT_ID='xx'
```

Now you should send money again to that coin address by invoking (*X* is the coin address, while *N* is the amount to send):

```
/opt/multi-pool/bin/bitcoind -datadir=/opt/bin/wallet/btc/ sendtoaddress X N
```

One error that should not happen but after a server reboot it sometimes pop up is when you have this: **E0063 Upstream share already assigned to previous block**. In one case this happen when the share auto-increment id was reset by the reboot to 0. So all shares fount are with a id that is below the last fount share.

You need to do this (*xx* is the result of first query, *yy* is of the third):

```
SELECT MAX(SHARE_ID) FROM BLOCKS;
UPDATE SHARES SET ID=ID+xx+1;
SELECY MAX(ID) FROM SHARES;
ALTER TABLE shares AUTO_INCREMENT=yy+1;
```

11 backup

If you want to store information that let you restore a previous step in short time, you need a backup[17]. With the *multi-pool* architecture all you need to do is to store at least the coin configuration file and the database contents, or in the best case the contents of */opt/multi-pool/* directory and the database contents too.

If you backup the whole */opt/multi-pool/*, you need to stop daemon otherwise you cannot backup blockchain in a consistent state. But this is not good for miners connected to your pool: more ofter you make a backup and for more times miners are disconnected.

There is no problem for backing up database when they are running as the dump utilities can handle this situation, so the first backup strategy should be good for miners and in case of a major failure you should rebuild the */opt/multi-pool/* directories with the standard scripts for *startup* and *makecoin* and then populate the configuration file from backup and for database you should use the apposite Mysql utilities.

For sure the new blockchain download can take lot of times, so I recommend to backup blockchain only after stopping daemons for a server reboot for update maintenance. This could short the time of a pool restart after a major crash of the server as you will use a more updated version of it.

Actually the automatic backup process in multi-pool is not ready, so I only backup one time the configuration files and the databases are backup by hand once periodically.

17 We did not considerate a virtual machine backup that can be done by apposite software that run at server hypervisor. They for sure are the best way to restart in zero time after a major failure, but usually they have a cost at every backup if you are using a cloud server.

12 upgrade

Another aspect is the update of stratum (less often), coin daemon (often) and MPOS (very often). This is not coded to be run automatically even if some process could be coded easily.

If you want to update the coin daemon you should go inside the source directory, make a git poll, then a make compilation, a stop of daemon and a copy of it inside binary (the copy preserve permission), then start it again, so this is an example for *bitcoin* coin:

```
cd /opt/multi-pool/src/bitcoin
git pull
cd src
make -f makefile.unix USE_UPNP=-
/opt/multi-pool/bin/bitcoind -datadir=/opt/bin/wallet/btc/ stop
cp bitcoind  ../../../bin
/opt/multi-pool/bin/bitcoind -datadir=/opt/bin/wallet/btc/ &
```

For the stratum software you should update the main installation and the one inside all the coins (here we see for bitcoin), so you need to stop stratum and restart.

```
cd /opt/multi-pool/stratum/stratum-mining
git pull
python setup.py install
cd /opt/multi-pool/stratum/btc/stratum-mining
git pull
kill -15 \$(cat /opt/multi-pool/stratum/btn/stratum-mining/twistd.pid)
sleep 1
kill -9 \$(cat /opt/multi-pool/stratum/btn/stratum-mining/twistd.pid)
twistd -y launcher.tac &
```

Please that present that the *config.py* should be modify by hand if some new rules are added to it.

Finally the MPOS upgrade is more delicate as it often has change directories structure and the config file that are to be handle changed. However this is the procedure for *bitcoin* coin:

```
cd /opt/multi-pool/mpos/btn
git pull
cd upgrades/
./run_upgrades.php
```

13 missing

During the description of this software we have see some features that is not implemented but that can be a good point to implement it.

- Uses a *coind* user and group different for every coin (this is for the file inside *bin* and *wallet* directories)
- Lets compilation to be executed by not a root user
- Uses not *root* user for MPOS (maybe it is more difficult for stratum to be not *root*)

- Adds a way to specify which stratum software to use (*stratum-mining* vs *NOMP* vs *CoiniumServ*) when we create a coin.
- Let *makecoin.sh* to take coin value from an external file without the need of modify it.
- Automatically adds the scripts to run in automatic to *crontab*.
- Automatically adds the site for Apache for the given coin.
- Add the automatic backup of the coin (configuration file and DB export)
- Add the automatic update of coin daemon, stratum and MPOS software for each coin.
- Add a main page (*index.php* on the domain) that automatically add the pools that are manages by the server (actually I make this by hand).
- Modify the DB of MPOS of each coin at every new pool added to insert the pools available in the server (this is an option in Setting panel of MPOS).

14 history

Multi-pool software was tested during the last 2 year inside the domain **multi-pool.info**[18]

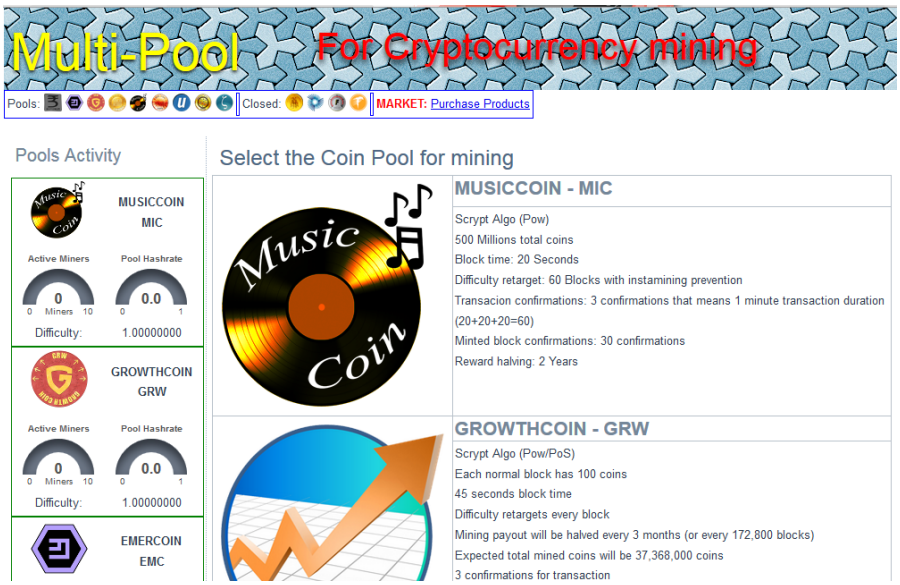

Figure 1: `http://multi-pool.info`

Multi-pool is now used in other servers and we hope that the community start to use it and improve it.

If you want to collaborate with this software check here: `https://github.com/ice00/multi-pool`

18 There is the plat to shutdown the server/domain in the future and to use a Raspberry Pi2 as testing platform.