Setup:

On a linux computer, inside a dedicated folder, run the following commands:

|  |
| --- |
| git clone -b 0.17 --single-branch https://github.com/FoXPeeD/bitcoin.git  sudo apt-get update  sudo apt-get install build-essential libtool autotools-dev automake pkg-config libssl-dev libevent-dev bsdmainutils python3 libboost-system-dev libboost-filesystem-dev libboost-chrono-dev libboost-test-dev libboost-thread-dev  sudo apt-get install software-properties-common  sudo add-apt-repository ppa:bitcoin/bitcoin  sudo apt-get install libdb4.8-dev libdb4.8++-dev  cd bitcoin/  ./autogen.sh  ./configure --without-gui --without-miniupnpc  make |

This will pull all files of project under the directory “bitcoin”.

Next, install python and projects required packages:

|  |
| --- |
| sudo apt update  sudo apt install software-properties-common sudo add-apt-repository ppa:deadsnakes/ppa  sudo apt install python3.7  sudo apt-get install python3-pip sudo python3.7 -m pip install paramiko  sudo python3.7 -m pip install boto3  sudo python3.7 -m pip install scp |

Three files of AWS needed:

* ~/config

|  |
| --- |
| [default]  region=us-east-2 |

* ~/credentials

|  |
| --- |
| [default]  aws\_access\_key\_id = <PLACE\_YOUR\_KEY\_HERE>  aws\_secret\_access\_key = <PLACE\_YOUR\_SECRET\_HERE> |

* Authorization file (pem), placed at the base folder.

You also need to create ssh keys (ssh-keygen -t rsa) which will be generated under ~/.ssh folder. After that use chmod 755 cmd on these new files.

Folder structure

Local computer:

Remote AWS servers:

How to run:

The main script is called make\_setup\_test.py.

It receives 4 parameters:

1. Maximum size of blocks in MB.   
   This parameter will also be used as the size of the block which will be timed when mined.  
   This parameter essentially replaces the hard limit of block size, set in original source code.
2. UTXO set size in percentage, out of the 1.8 MB cache size.   
   (e.g. the value 50 will set the UTXO set size to 0.9 MB)  
   Clarification: in the script the dbcache is set to 4 MB, but only 1.8 MB out of this memory allocation is reserved for UTXO set in memory.
3. Number of bitcoin nodes to participate in the experiment (each in its own AWS instance),  
   including the node that acts as the miner.
4. Topology, namely:  
   mesh – all nodes are connected to all nodes (a clique).  
   static – every node is connected to 3 nodes randomly and another connection to the node with the next index (to ensure full connectivity). Random is based on hashes, which means its deterministic and thus consistent between runs.

dynamic – every node is connected to 3 nodes randomly and another connection to the node with the next index (to ensure full connectivity). Not consistent between runs.

Running exemple:

python3.7 make\_setup\_test.py 2 200 20 mesh

The command will initiate a test with block size of 2MB, 3.6MB of UTXO set (200% out of the 1.8MB cache), 20 instances running in AWS and a topology of a clique.

If a folder corresponding to the (first two) parameters mentioned above does not exist on the server (and thus not in the AMI as well), another script will be called to generate the folder required.  
note that generating a new folder is good for only one use. after that, the folder should either be erased or added to an AMI.  
after the script is done, the results will be printed to screen and written to time.txt file at the base folder.  
the measurements are the difference from when the first block notified (of the miner) to the time when some other node notified about a new block. results are comma separated.

For convenience, another script is supplied, called run\_multiple\_tests.py, that can run multiple subsequent tests according to lists of parameters inside it. The script will go over every combination of parameters from all lists (nested for loops for each parameter list) and can do multiple iterations for each combination.   
this script is intended only for parameter combinations that already have a corresponding data directory in the AMI, or at least will work for only one iteration(subsequent iterations will result in an error, because the main scrip will think the folder created on the first iteration is at the AMI too).   
results file (generated by main script) is renamed to reflect the parameters used as a CSV file, and the content is appended to an aggregated results file.

AWS setup requirements

* VPC –needs to assign a public IP.
* Security group – needs to allow ssh connection and all tcp ports in both directions.
* Subnets – needs to assign private IP addresses in the 10.0.2.100 - 10.0.2.199 range.

Parameters to change in script:  
parameters that need to change when adapting the scripts to a new AWS account are marked in main script by “TODO-SETUP” tag.

* Range of private IP addresses, if different from AWS setup requirements.
* pem file name.
* name associated with pem file.
* AMI ID
* Subnet ID
* Security group (sg) ID

Important bitcoind/conf file arguments:  
Mempoolexpiry – by default, when a node launches it loads transactions to mempool from file, it refers to any txs older then two weeks as expired. In order for pre-generated data directories to not expire after a while, this parameter (in hours) is set to a calculated value such that no data directory will expire.  
blocksonly – rising this flag prevents nodes from getting information about transactions and UTXOs from other nodes. This is applied to all nodes except the miner, in order for them to process the blocks “from scratch” (without any preparation or prepared calculations)  
reindex-chainstate – sometimes, when copying the data directories, some meta data or indexes of the blockchains state gets corrupted. This flag recalculates all this meta data at launch from the blocks themselves.

dbcache-