

Opracowanie i wytworzenie testowej sieci blockchain w środowisku testowym

Projekt: "Przeprowadzenie prac badawczych i rozwojowych umożliwiających wdrożenie inteligentnego kontraktu opartego o technologię blockchain"

Kontekst: Wybrana sieć testowa będzie użyta do badań związanych z opracowywaniem metod Inteligentnych Kontraktów oraz opracowywaniem sposobów na umieszczanie danych w rozproszonej sieci Blockchain. Uruchomienie własnej sieci testowej blockchain opartej o jedną z powstałych sieci (np. Fork BTC). Nastąpi uruchomienie własnej sieci testowej blockchain opartej o jedną z powstałych sieci. Bezpieczne odizolowanym od publicznej sieci środowisko testowe.

Autorzy:

- Blicharski Bartłomiej
- Martin Morawiec

Opracowanie i wytworzenie testowej sieci blockchain w środowisku testowym	0
Wykonanie i przetestowanie Fork BTC	2
Przygotowanie przestrzeni do wytworzenia bloku genesis i kompilacji forka Bitcoin.	2
Tworzenie bloku genesis	3
Pobieranie i zmiana nazwy forku	4
Edytowanie parametrów w plikach	4
Uruchomienie testowej sieci Ethereum w środowisku VPN	5
Instalacja geth w systemie Ubuntu	5
Wytworzenie portfeli	6
Wygenerowanie nowego genesis	6
Dodawanie node'ów statycznych	8
Obsługa	9
Deploy Kontraktów	11
Instalacja i konfiguracja systemu linux pod maszynę obliczeniową	12
W systemie	12
Uruchamianie claymore	13
Automatyczne uruchamianie Claymore	13
BIOS Maszyny	14
Optymalizacja kart graficznych	14
Podkręcanie kart graficznych	15
Główne czynności przy optymalizacji kart	15
Zapisywanie fabrycznego biosu	15
Zmiana timingów	16
Undervolting	16
Zmiana częstotliwości rdzenia i pamięci	17
Co może wskazywać, że karta tego samego typu jest inna?	17
AtiFlash	19
GPU-Z	20
Testowanie poprawności modyfikacji kart - HWInfo	20
Źródła w internecie	22
Maszyny obliczeniowe - budowa puli maszyn	23
Notatnik roboczy	24

1. Wykonanie i przetestowanie Fork BTC

1.1. Przygotowanie przestrzeni do wytworzenia bloku genesis i kompilacji forka Bitcoin.

Dla operacyjnego systemu należy przygotować środowisko np. Wirtualne środowisko Vagrant z systemem operacyjnym Ubuntu 12.04 LTS (Linux 3.2.0-23-generic_x86_64). Kompilacja kryptowaluty bitcoin¹ wymaga dużej ilości pamięci operacyjnej ram (minimum 1.5 GB) dlatego wirtualny system powinien być ustawiony np. Na 4GB.

Wymagane biblioteki do kompilacji:

Biblioteka	Cel	Opis
gcc	Genesis block	Kompilacja GenesisBlockZero dla bloku Genesis
libssl	Crypto	Generowanie liczb losowych, kryptografia krzywej eliptycznej
libboost	Utility	Biblioteka do wątków, struktur danych itp
libevent	Networking	Niezależna od systemu operacyjnego asynchroniczna sieć

Opcjonalne biblioteki według typu kompilacji:

Biblioteka	Cel	Opis
miniupnpc	UPnP Support	Firewall-jumping support
libdb4.8	Berkeley DB	Portfel (wymagane tylko przy włączonym portfelu)
qt	GUI	GUI toolkit (wymagany tylko przy włączonym GUI)
protobuf	Payments in GUI	Format wymiany danych używany do protokołu płatności (wymagany tylko przy włączonym GUI)
libqrencode	QR codes in GUI	Do generowania kodów QR (wymagane tylko przy włączonym GUI)

¹ <https://github.com/bitcoin/bitcoin/tree/master/doc#building>

univalue	Utility	Parsowanie i enkodowanie JSON
libzmq3	ZMQ notification	Opcjonalne, umożliwia generowanie notyfikacji ZMQ (wymaga ZMQ w wersji większej niż 4)

1.2. Tworzenie bloku genesis

Blok należy wygenerować przy pomocy programu napisanego w języku C++ GenesisBlockZero. Wyjścia użyjemy w części ustawienia parametrów bloku. Kod trzeba ustawić wg. Własnych potrzeb i skompilować przed uruchomieniem.

- COIN - Maksymalna ilość monet,
- CENT - Ilość miejsc po przecinku
- startNonce - Początkowa trudność
- unixtime - aktualny czas (timestamp), powinien być bliski czasu wytworzenia pierwszego bloku.

```
# Kompilacja
gcc genesisblock.c -o genesisgen -lcrypto
# Generowanie bloku
./genesisgen
04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61deb649f6bc
3f4cef38c4f35504e51ec112de5c384df7ba0b8d578a4c702b6bf11d5f "Nostromo -
Only unconventional thinking brings original solutions" 486604799
```

Wyjście z programu genesisgen:

```
Coinbase:
04ffff001d0104414e6f7374726f6d6f202d204f6e6c7920756e636f6e76656e74696f6e
616c207468696e6b696e67206272696e6773206f726967696e616c20736f6c7574696f6e
73

PubkeyScript:
4104678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61deb649f6
bc3f4cef38c4f35504e51ec112de5c384df7ba0b8d578a4c702b6bf11d5fac

Merkle Hash:
51dc1cd2b1744c950346cc61beb1599a12346ae185ed936ee50c8be969b06e50
Byteswapped:
506eb069e98b0ce56e93ed85e16a34129a59b1be61cc4603954c74b1d21cdc51
Generating block...
```

```
1285198 Hashes/s, Nonce 2564497054
Block found!
Hash: 0000000035723ec668cfca8e81d5e2198bbd0039d314772a8591244e1e69e267
Nonce: 2564588227
```

Pobieranie i zmiana nazwy forku

```
git clone https://github.com/bitcoin/bitcoin.git
mv bitcoin-master nostromo
cd nostromo
# Rename Bitcoin to new own Nostromo coin
find ./ -type f -readable -writable -exec sed -i "s/Bitcoin/Nostromo/g"
{} \;
find ./ -type f -readable -writable -exec sed -i "s/BitCoin/Nostromo/g"
{} \;
find ./ -type f -readable -writable -exec sed -i "s/BTC/NST/g" {} \;
find ./ -type f -readable -writable -exec sed -i "s/bitcoin/nostromo/g"
{} \;
find ./ -type f -readable -writable -exec sed -i
"s/bitcoind/nostromod/g" {} \;
find ./ -execdir rename 's/bitcoin/nostromo/' '{}' \; # Rename files
```

Edytowanie parametrów w plikach

```
# ./nostromo/src/amount.h
26 static const CAmount MAX_MONEY = 100000000 * COIN;
```

Tutaj używamy wyjścia z wygenerowanego bloku Genesis

```
# ./nostromo/src/chainparams.cpp
51 const char* pszTimestamp = "Nostromo - Only unconventional thinking
brings original solutions";
52 const CScript genesisOutputScript = CScript() <<
ParseHex("04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61
deb649f6bc3f4cef38c4f35504e51ec112de5c384df7ba0b8d578a4c702b6bf11d5f")
<< OP_CHECKSIG;

122 genesis = CreateGenesisBlock(1532519350, 2564588227, 0x1D00FFFF, 1,
50 * COIN);
123 consensus.hashGenesisBlock = genesis.GetHash();
124 assert(consensus.hashGenesisBlock ==
```

```
uint256S("0x0000000035723ec668cfca8e81d5e2198bbd0039d314772a8591244e1e69
e267"));
125 assert(genesis.hashMerkleRoot ==
uint256S("0x506eb069e98b0ce56e93ed85e16a34129a59b1be61cc4603954c74b1d21c
dc51"));
```

```
# ./nostromo/src/clientversion.cpp
15 const std::string CLIENT_NAME("Nostromo");
```

Kompilacja i instalacja

```
./autogen.sh
./configure
make
make install # instalacja (opcjonalnie)
```

Uruchomienie

```
nostromod -datadir=/home/username/1 -server -rpcuser=nostromo -rpcpassword=nostromo
-rpcport=9332 -daemon
```

```
nostromod -datadir=/home/username/2 -server -rpcuser=nostromo -rpcpassword=nostromo
-rpcport=19332 -nolisten -daemon
```

Instalace poolu

2. Uruchomienie testowej sieci Ethereum w środowisku VPN

Na potrzeby prac z Inteligentnymi Kontraktami zostanie utworzona prywatna sieć Ethereum zasilana maszynami obliczeniowymi. Całość będzie działała w prywatnej sieci VPN. Należy pamiętać, że przy pobieraniu chaindata jest opcja `--fast`, która powoduje, że ściągają się tylko hashe transakcji, a nie całe transakcje z zawartością. Jest to sposób na szybsze zsynchronizowanie się z siecią. Trzeba pamiętać, jeżeli rozpocznie się tryb light z opcją `--fast`, nie możemy pobierać chaindata zwykłego z pełnymi transakcjami i odwrotnie. Poniższa instrukcja przedstawia jak uruchomić i korzystać z prywatnej sieci Ethereum.

2.1. Instalacja geth w systemie Ubuntu

```
sudo apt-get install software-properties-common
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install ethereum
```

2.2. Wytworzenie portfeli

Teraz tworzymy portfele (przynajmniej dwa) i zapisujemy ich kopie zapasowe oraz adresy w bezpiecznym miejscu. Poniższa komenda przedstawia metodę na utworzenie nowych kont w sieci ethereum.

```
geth --datadir .ethereum/ account new
```

2.3. Wygenerowanie nowego genesis

Do wytworzenia sieci Ethereum od zera - czyli nowego genesis użyjemy narzędzia *puppeth*, które umożliwia szybkie stworzenie własnej prywatnej sieci. Narzędzie to jest dostępne wraz z *geth*, pozwala na wydobywanie informacji o sieci, konfigurowanie nowego genesis, śledzenie serwera zewnętrznego.

W configu, który wybierzemy dla nowego genesis - Clique (proof-of-authority) bloki wykopywać mogą tylko portfele, którym zezwolimy na to na początku. Nie powoduje to generowania kryptowaluty.

Uruchamiamy narzędzie *puppeth*

Następnie w konsoli:

```
Please specify a network name to administer (no spaces, please)
> [Jakaś nazwa sieci, bez spacji]

What would you like to do? (default = stats)
1. Show network stats
2. Configure new genesis
3. Track new remote server
4. Deploy network components
> 2

Which consensus engine to use? (default = clique)
1. Ethash - proof-of-work
2. Clique - proof-of-authority
> 2

How many seconds should blocks take? (default = 15)
> 15
//pusty blok bez transakcji waży 1024 bajty, z transakcjami więcej,
wykopywany co 15 sekund po czasie może zająć dużo miejsca (5.7MB dziennie
```

jeśli bloki są puste). Duży czas powoduje, że trzeba czekać na transakcje

Which accounts are allowed to seal? (mandatory at least one)

```
> 0x[Pierwszy adres konta, które generowaliśmy(możemy dodać więcej, puste kończy wpisywanie)]
```

```
> 0x
```

```
//tylko te adresy mogą kopać
```

```
//uwaga, (N/2+1) kopaczy musi być online, N to ilość adresów, które dodamy
```

Which accounts should be pre-funded? (advisable at least one)

```
> 0x[Drugi adres konta, które generowaliśmy (tak samo jak wyżej)]
```

```
> 0x
```

```
//te adresy dostaną ether na początku
```

Specify your chain/network ID if you want an explicit one (default = random)

```
> [ID sieci (najlepiej <1337, podobno duże (z randoma) mogą powodować błędy)]
```

What would you like to do? (default = stats)

1. Show network stats
2. Manage existing genesis
3. Track new remote server
4. Deploy network components

```
> 2
```

1. Modify existing fork rules
2. Export genesis configuration
3. Remove genesis configuration

```
> 2
```

Which file to save the genesis into? (default = clique.json)

```
> genesis.json
```

```
INFO [02-15|18:24:22] Exported existing genesis block
```

```
> ctrl+c
```

Wybór silnika - dostępne są dwie opcje, Ethash(proof-of-work) i Clique(proof-of-authority). Ethash - stosowana w blockchainie Ethereum metoda oparta o wykopywanie bloków poprzez rozwiązywanie skomplikowanych problemów matematycznych. Clique - metoda oparta na zaufanych klientach, blok musi zostać potwierdzony przez większość z zaufanych klientów, wtedy dodawany jest do blockchainu. W naszym przypadku wybraliśmy Clique,

ponieważ nie wymaga sporej ilości obliczeń, co pozwala na szybkie działanie blockchainu oraz zmniejszenie zużycia energii.

Następnie edytujemy plik genesis.json i dla ostatniego konta (prawie na końcu pliku) ustawiamy balance na np. 2000000000000000000000 (wei = 200 ether)

Inicjalizujemy nową sieć blockchain

```
geth --datadir .ethereum/ init genesis.json
```

Uruchamiamy blockchain, pierwszego głównego node'a

```
geth --datadir=.ethereum/ --networkid [id sieci] --port 30303 --rpc  
--rpcport 8545 --nodiscover console
```

@param datadir - ścieżka, gdzie będzie trzymany blockchain

@param --networkid - id sieci, musi być takie samo na wszystkich node'ach i odpowiadać id w genesis

@param --nodiscover - node nie szuka automatycznie peerów (dodajemy je ręcznie poprzez plik static_nodes.json[na stałe] albo funkcją admin.addPeer([enode]))

console - odpala konsolę javascript

2.4. Dodawanie node'ów statycznych

Dodanie kolejnego node na innej maszynie odbywa się poprzez zainstalowanie geth, tak jak na głównym nodzie, gdzie stworzyliśmy genesis.

Tworzymy portfele:

```
geth --datadir .ethereum/ account new  
//można stworzyć portfel przed utworzeniem genesis.json na głównym nodzie  
i wtedy dodać adres do tych, które mogą kopać (nie trzeba wtedy przenosić portfela)
```

Kopiujemy genesis.json z głównego node'a na nową maszynę i inicjalizujemy blockchain:

```
geth --datadir .ethereum/ init genesis.json
```

W konsoli na głównym nodzie odpalamy komendę:

```
admin.nodeInfo.enode
```

output:

```
"enode://3d51324b952c006dc25e164eab79c06122491eb256e67ab658a631b0af551bc  
0c93afb36cae0f595ef15f49d6d2b54cd55fce24b7c7a0e7b62d9b2e46c1a54f9@[::]:3
```

```
0303?discport=0"
```

Usuwamy '?discport=0' i zamieniamy '[':']' na ip maszyny

Po zmianie:

```
"enode://3d51324b952c006dc25e164eab79c06122491eb256e67ab658a631b0af551bc0c93afb36cae0f595ef15f49d6d2b54cd55fce24b7c7a0e7b62d9b2e46c1a54f9@172.28.128.3:30303"
```

Na drugiej maszynie tworzymy plik '.ethereum/static-nodes.json' i dodajemy do niego dane, które kopiowaliśmy (zachowując strukturę jsona).

```
[ "enode://3d51324b952c006dc25e164eab79c06122491eb256e67ab658a631b0af551bc0c93afb36cae0f595ef15f49d6d2b54cd55fce24b7c7a0e7b62d9b2e46c1a54f9@172.28.128.3:30303" ]
```

Gdyby nodów było więcej to oddziela się je przecinkami. Plik static_nodes powoduje, że odpalony node automatycznie łączy się / synchronizuje z danym nodem (głównym)

Odpalamy tą samą komendą co główny blockchain

```
geth --datadir=.ethereum/ --networkid [id sieci] --port 30303 --rpc --rpcport 8545 --nodiscover console
```

Jeśli chcemy obliczać bloki na nowym nodzie, to musimy skopiować plik z portfelem, który może obliczać nowe bloki (chyba, że dodaliśmy portfel do genesis.json). Plik znajduje się w .ethereum/keystore (na głównym nodzie), musimy go skopiować na nowego node'a do tej samej lokalizacji.

2.5. Obsługa

Jeśli portfel generowaliśmy na danym nodzie (albo przenieśliśmy jego klucz z keystore) to możemy zamiast całego adresu portfela używać *eth.accounts[i]*, gdzie *i* to numer konta *[0..n-1]* (wg daty wygenerowania portfela).

Można zamienić ilość etheru na wei poleceniem:

```
web3.toWei([kwota w etherze], "ether")
```

W drugą stronę:

```
web3.fromWei([kwota w etherze], "ether")
```

Sprawdzenie balansu danego portfela:

```
web3.fromWei(eth.getBalance([adres portfela]), "ether")
```

Odblokowanie portfela:

```
personal.unlockAccount([adres portfela])  
//lub  
personal.unlockAccount([adres portfela],[haslo])  
//lub  
personal.unlockAccount([adres portfela],[haslo],[liczba komend na  
ktore odblokowujemy portfel])  
//np. Jeśli chcemy wykonać 10 transakcji to nie musimy 10 razy  
odblokowywać tylko podać 10 jako 3 parametr. 0 odblokowuje na stałą
```

Do wykonania transakcji musimy mieć odblokowany portfel. Poniższy przykład przesyła wartości pomiędzy portfelami.

```
eth.sendTransaction({from:[portfel z którego wysyłamy], to:[portfel do  
którego wysyłamy], value: [kwota w WEI]})
```

Żeby transakcja została zatwierdzona w blockchainie trzeba obliczyć blok za pomocą minera lub sieci minerów.

```
miner.start([ilość wątków]) // startuje minera  
Miner.stop // zatrzymuje minera
```

Przydatna metoda, która pokazuje balans wszystkich kont, których portfele znajdują się na danym nodzie:

```
function checkAllBalances() {  
  var i =0;  
  eth.accounts.forEach(function(e){  
    console.log("  eth.accounts["+i+"]: " + e + " \tbalance: " +  
web3.fromWei(eth.getBalance(e), "ether") + " ether");  
  
    i++;  
  })  
};
```

Wywołanie jak w normalnym javascript

```
checkAllBalances();
```

2.6. Deploy Kontraktów

Musimy mieć zainstalowany solc (solidity compiler),
<http://solidity.readthedocs.io/en/develop/installing-solidity.html>

Kontrakty piszemy w języku Solidity i zapisujemy jako plik z rozszerzeniem .sol
Tworzymy katalog, w którym będziemy zapisywać pliki potrzebne do deployu (.abi i .bin)

Do kompilacji wykonujemy komendę:

```
solcjs -o [nazwa katalogu] --bin --abi [nazwa_pliku].sol
```

W katalogu powinny pojawić się pliki .abi i .bin dla każdej klasy. Do deployu potrzebna nam jest tylko ostatnia (nadrzędne nie)

W konsoli geth odpalamy

```
var factory = eth.contract([zawartość pliku .abi])
var compiled = "0x" + "[zawartość pliku .bin]"

var deployed = factory.new([parametry do
konstruktora], {from:eth.accounts[0], data:compiled, gas:47000000},
function(e, contract){
    if(e) {
        console.error(e);
        return;
    }
    if(!contract.address) { console.log("Contract transaction send:
TransactionHash: " + contract.transactionHash + " waiting to be
mined...");
    } else {
        console.log("Contract mined! Address: " + contract.address);
        console.log(contract);
    }
})
```

Można użyć innego konta, albo zmienić ilość gazu (wymaganie zależne od blockchainu i jego ustawień). Deploy do blockchainu, tak jak każda zmiana, kosztuje kryptowalutę więc konto musi mieć środki żeby móc wykonać deploy. Po deployu kontraktu trzeba obliczyć blok, żeby móc z niego korzystać.

3. Instalacja i konfiguracja systemu linux pod maszynę obliczeniową

Do uruchomienia i skonfigurowania maszyny użyjemy systemu Ubuntu 16.04 oraz konfiguracji pozwalającej na uruchomienie programu obliczającego w momencie startu systemu operacyjnego. Uruchomienie obliczeń bloków odbędzie się przed pojawieniem się ekranu z logowaniem, dostęp do maszyny będzie możliwy przez protokół SSH lub po odpowiednim uruchomieniu systemu np. w trybie awaryjnym / edycji grub.

Przy instalacji systemu należy nadać maszynie odpowiedni "hostname", który będzie unikalny i rozpoznawalny przy większej ilości koparek np. B001, B002 itp. Koniecznie trzeba zaznaczyć opcję "no automatic updates" ponieważ instalujemy określone sterowniki i nie chcemy aby system sam wykonywał aktualizację np. przy restarcie systemu (których, zdarza się, że jest wiele). Zaznaczenie tej opcji, może spowodować, automatyczną aktualizację kernela, co z kolei może spowodować utratę mocy obliczeniowych kart. Co do użytkownika komputera, warto użyć jakiegoś systemu loginu i haseł.

W dodatkowych komponentach systemu zaznaczamy "ssh server", który jest potrzebny do bezproblemowego łączenia się z maszyną protokołem SSH. Po zakończeniu procesu, instalator poinformuje o automatycznym ponownym uruchomieniu maszyny.

3.1. W systemie

1. Logujemy się do systemu. Przechodzimy na roota za pomocą komendy "sudo su"
2. Wykonujemy update systemu wpisując komendę "sudo apt update", później "sudo apt upgrade"
3. Dobrym nawykiem jest zmodyfikować port dostępu SSH na customowy np. 8880 zamiast portu 22, a następnie zablokować możliwość logowania userem root.
4. Za pomocą vi "vi /etc/ssh/sshd_config" edytujemy zawartość pliku zmieniając właściwości protokołu SSH. W konfiguracji ssh_config zmieniamy PORT (jedna z początkowych linii) na nasz port np. 8880. Następnie zmieniamy wartość parametru "PermitRootlogin" na "no".
5. Pobieramy plik libcurl3 za pomocą komendy "apt install libcurl3" i restartujemy system za pomocą komendy "reboot". Możemy pominąć ten krok, jeżeli używamy programów do obliczeń, które tego nie wymagają. W naszym przypadku będziemy używali programu claymore, który do wersji 11.7 wymaga używania tej biblioteki.
6. Do naszego folderu "home/user" ściągamy nasze sterowniki do kart graficznych oraz claymore. **Bardzo ważne jest żeby nie robić tego na roocie aby zwykły użytkownik miał do nich dostęp.**
7. Wypakowujemy sterowniki do kart za pomocą komendy "tar" (np. tar --xz -zxf w przypadku plików tar.xz czy tar -zxvf w przypadku plików .tar.gz)
8. Jeżeli korzystamy z *claymore* to przystąpimy do konfiguracji pliku start.bash znajdującym się w folderze "claymore". Edytujemy plik "start.bash" za pomocą edytora vi (vi start.bash) i edytujemy w nim wartość:

```
#export GPU_FORCE_64BIT_PTR=0
export GPU_MAX_HEAP_SIZE=100
export GPU_USE_SYNC_OBJECTS=1
export GPU_MAX_ALLOC_PERCENT=100
export GPU_SINGLE_ALLOC_PERCENT=100
```

```
/home/eth/claymore/ethdcrminer64 -epool eu1.ethermine.org:4444 -ewal
0x79AE2031C5eA2E1bfd96d413179ab38504B7E3Cb.B005 -epsw x -mode 1 -ftime 10
```

9. Niezależnie od oprogramowania, którego używamy z powyższej konfiguracji ważne jest ustawienie zmiennych GPU_MAX_HEAP_SIZE, GPU_USE_SYNC_OBJECTS, GPU_MAX_ALLOC_PERCENT, GPU_SINGLE_ALLOC_PERCENT.
10. Dobrym zwyczajem jest również skonfigurowanie pliku stronicowania (np. 16 GB).
11. Należy zainstalować sterowniki amdgpu pro (np. 17.30 AMDGPU-Pro Beta Mining Driver for Linux®)
12.). Instalujemy sterowniki z prawami roota “sudo ./amdgpu-pro-install”
13. Jako root wpisujemy komendę “usermod -aG video eth” oraz “usermod -aG video root” w celu dodania użytkownika eth oraz root do grupy “video” w celu udostępnienia możliwości korzystania ze sterownika OCL/Vulkan przez usera.
14. Edytujemy plik “/etc/default/grub”. Wpisujemy do LINUX_DEFAULT wartość “amdgpu.vm_fragment_size=9”, która umożliwi zwiększenie tablicy stron pamięci tzw. PTE (Page Table Entries)
15. Po tej czynności z prawami roota musimy wykonać aktualizację grub’a komendą “update-grub”, a następnie zresetować maszynę.

3.2. Uruchamianie claymore

Przechodzimy do folderu “claymore” (Przykładowa lokalizacja : “cd /home/eth/claymore”) i uruchamiamy plik start.bash wpisując “./start.bash”. Aby przerwać obliczanie bloku i przejść z powrotem do systemu należy posłużyć się kombinacją klawiszy “Ctrl + C”.

3.3. Automatyczne uruchamianie Claymore

Tą funkcjonalność powinniśmy wdrażać tylko i wyłącznie gdy jesteśmy pewni, że cały system skonfigurowaliśmy poprawnie.

Przełączamy się na roota (“sudo su”), edytujemy plik “rc.local” (“vi /etc/rc.local”). Nad kodem “return 0” wpisujemy ścieżkę do naszego start.bash. Przykładowa ścieżka:

```
/home/eth/claymore/start.bash
```

Istnieją różne sposoby na automatyczne uruchamianie skryptów do obliczeń. Skrypty można również uruchamiać jako serwisy działające w tle i uruchamiane po poprawnym uruchomieniu systemu operacyjnego. Nie jest, aż tak istotny sposób, który zostanie użyty, powinien być zależny i dopasowany potrzeb.

3.4. BIOS Maszyny

W BIOSie maszyny obliczeniowej należy:

1. Ustawić automatyczne uruchamianie systemu po odzyskaniu zasilania. Należy szukać opcji "Restore on AC/Power Loss".
2. Wyłączyć niepotrzebne komponenty takie jak nieużywane porty, HD Audio itp.
3. Ustawiać "Primary Graphic Adapter" na "Onboard" aby po podłączeniu kart sygnał video był nadawany przez wbudowaną w kartę graficzną.
4. Można zmienić w BIOSie ustawienia "Pci Express Link Speed" z gen1 na gen2 lub odwrotnie. Najlepiej żeby wszystkie ustawienia na płycie były takie same. Zauważyliśmy, że czasem przestawienie tej wartości np. z gen1 na gen2 zwiększyło stabilność urządzenia.

4. Optymalizacja kart graficznych

UWAGA: Nie należy przesadzać ze zbyt mocnym podkręcaniem kart ponieważ wtedy mogą ulegać awarii. Karty nie powinny się mocno grzać, należy zapewnić odpowiednie chłodzenie karcie oraz całej maszynie.

W przypadku częstego występowania awarii na konkretnych kartach, przyczyn można szukać w:

1. Wadliwych riserach.
2. Zbyt dużej ilości podłączonych kart lub innego sprzętu do jednego kabla zasilającego SATA. Do jednego kabla powinny być podłączone maksymalnie dwie karty.
3. Zbyt mocnym lub błędnym "podkręcanie" rdzenia lub pamięci, udervoltingiem.
4. Zbyt mocnym grzaniu się kart. Również z miejsca w którym znajdują się karty musi być odpowiednio odprowadzone ciepło. Zauważyliśmy, że często ulegają awarii karty, które znajdują się nad innymi elementami, które podgrzewają otoczenie np. zasilaczami.
5. Wpływ może mieć również: źle podpięte okablowanie, słaby zasilacz.

4.1. Podkręcanie kart graficznych

Na początek należy uzbroić się w cierpliwość. Głównym celem zmiany ustawień fabrycznych kart graficznych jest optymalizacja ich działania przy obliczaniu bloków w technologii blockchain. Domyślnie karty graficzne przygotowane są do użytku np. w grach komputerowych czy też oprogramowania 3D, renderingu, obliczeń OpenCL czy CUDA. W ten sposób posiadają wysoką wydajność i stabilność w programach wykorzystujących akcelerację graficzną, lecz również wysoki pobór prądu i bezpieczne ustawienia zegarów rdzenia i pamięci. Karty z serii RX 5xx posiadają fabryczne wartości tych parametrów zależne od producenta, serii, a nawet konkretnego egzemplarza. Patrząc przez pryzmat mocy obliczeniowej w sieci Ethereum, przykładowymi wartościami fabrycznymi mogą być np. 24 Mh/s przy poborze mocy 140W. Docelowe wartości, które staramy się uzyskać to około 80W-110W (im mniej tym lepiej) i 27 - 31Mh/s (im więcej tym lepiej). Najważniejszym parametrem jest stabilność! na którą wpływ ma bardzo wiele czynników.

Karty muszą być tak podkręcone tak aby maszyna mogła działać stabilnie przez długi okres, a jej żywotność była jak najlepsza. Lepsza jest karta która stabilnie liczy z prędkością 29 Mh/s, niż karta która liczy z prędkością 32 Mh/s, ale ulega małym awariom po kilku godzinach. Kolejnym ważnym parametrem jest pobór energii, ponieważ od niego zależy to jak mocno karta się grzeje. Jeśli możemy zyskać 20W poboru mniej, kosztem 1Mh/s to warto "stracić" 1Mh/s niż "używać" dodatkowo 10% mocy całej karty.

Ważne:

1. Karty podkręcamy pojedynczo!
2. Należy pamiętać że modyfikując bios tracimy gwarancję !
3. Należy pamiętać o kopii BIOSU, każdej z karty GPU.

4.2. Główne czynności przy optymalizacji kart

Z odpowiednim ustawieniem zegarów i czasów możliwe jest osiągnięcie wyższych wyników obliczeniowych przy niższym zużyciu energii elektrycznej. Konieczne jest wykonanie backupu biosa wybranej karty programem GPU-Z. Natomiast programem PolarisBiosEditor można eksperymentować z różnymi ustawieniami taktowania pamięci aby osiągnąć lepsze rezultaty.

1. Zapisywanie fabrycznego biosu

Pierwszą czynnością jaką wykonujemy to zapisanie fabrycznego biosu z karty za pomocą programu ATI Flash². Należy zapisać BIOS oznaczyć tak żeby było wiadomo z której karty pochodzi, np. "RX580_8G_HYNIX_01_orig.rom". Tutaj warto opracować sobie system oraz oznaczać jasno również z wersją modyfikacji np.

² AtiFlash

“RX580_8G_HYNIX_01_mod_1.rom”. Koniecznie trzeba pamiętać o zapisie oryginalnych BIOS-ów w bezpieczne miejsce.

2. Zmiana timingów

Za pomocą programu do edycji biosu (np Polaris Bios Editor) otwieramy fabryczny bios karty. Zamieniamy paski timingów zgodnie z tutorialami dostępnymi np. na forum [bitcointalk](#)³. Istnieją różne metody, podstawowa to skopiowanie wartości z paska 1750 lub 1500 do wszystkich większych wartości. Można też kupić gotowe timingi, lub całe biosy do danej karty, są pomocne jednakże nie robią za nas całej pracy. Ważna jest świadomość tego co się robi, dlatego warto również zaprzyjaźnić się z terminami i zdobyć odpowiednią wiedzę. Modyfikowane timing strapy to podstawowymi paskami timingów z kart graficznych z wyostrzonymi czasami dla lepszej wydajności. Mogą być zdekodowane np. programem SRBPolaris. Niektóre programy potrafią automatycznie wykryć typ karty i zastosować odpowiednie wartości np. zmodyfikowana wersja PolarisBiosEditor z [mining-bios.eu](#).

Uber Mix dla ELPIDA:

```
777000000000000022AA1C00315A5436A0550F15B68C1506004082007C041420CA8980A9020004C01712262B612B3715
```

Uber Mix dla HYNIX:

```
999000000000000022559D0010DE5B4480551312B74C450A00400600750414206A8900A00200312010112D34A42A3816
```

```
777000000000000022AA1C00B56A6D46C0551017BE8E060C006AE6000C081420EA8900AB030000001B162C31C0313F17
```

Uber Mix dla MICRON:

```
777000000000000022AA1C0073626C41B0551016BA0D260B006AE60004061420EA8940AA030000001914292EB22E3B16
```

Uber Mix dla SAMSUNG:

Uber-Mix 3.1

```
777000000000000022CC1C00AD615C41C0590E152ECC8608006007000B031420FA8900A00300000010122F3FBA354019
```

Uber-Mix 3.2

```
777000000000000022CC1C00CEE55C46C0590E1532CD66090060070014051420FA8900A00300000012123442C3353C19
```

3. Undervolting

Ten krok ma na celu zmniejszenie poboru mocy karty. W sekcji memory programu Polaris Bios Editor zmniejszamy wartość mV z 1000 do 950. W zakładce GPU interesuje nas również kolumna mV, wartości podobne do “65288” oznacza dynamiczną wartość.

³ [bitcointalk.org](#)

Zmieniamy tylko ostatnią cyfrę. Modyfikujemy wartości poniżej "65283", tak żeby było "800" "65282" "65283" "65283" "65283" "65284" "65285" "65286". Kolejny punkt jest również elementem undervoltingu.

4. Zmiana częstotliwości rdzenia i pamięci

Memory clock - zegar pamięci - od niego głównie zależy hashrate. Im większy tym więcej hashrate, ale do pewnego momentu. Należy zachować umiar ponieważ zbyt wielkie wartości powodują niestabilność karty, a nawet jej uszkodzenie. Przeciętne wartości to zwykle 2000 - 2100MHz dla RX 580 i 1750 - 2000MHz dla RX570. Są to wartości zależne od producenta karty, producenta kości pamięci ram (np. HYNIX, SAMSUNG, MICRON, ELPIDA) należy testować stabilność po ich zmianie i najlepiej po znalezieniu stabilnej wartości, zmniejszyć jeszcze odrobinę, np o 20MHz. W Polaris Bios Editor⁴ możemy wpisać tę wartość do biosu, lecz należy to robić wtedy kiedy już mamy działającą wartość. Robi się to poprzez modyfikację największej wartości w zakładce memory, kolumnie MHz.

Core clock - zegar rdzenia - wartość którą należy zmniejszyć, aby poprawić zużycie prądu, bezpieczne wartości dla kart RX 5xx to zwykle 1150 - 1250MHz. W Polaris Bios Editor należy zmienić nie tylko ostatnią wartość, ale też większość poprzednich. Np 300 600 900 1000 1050 1100 1150 1200.

Testowanie różnych wartości najlepiej przeprowadzać za pomocą programu typu Afterburner, po modyfikacji timingów i undervoltingu. Należy pamiętać że program ten nadpisuje ustawienia kontroli napięcia więc wyniki poboru prądu będą podwyższone w trakcie jego uruchomienia.

Typy pamięci - Szczególną uwagę należy zwrócić na różnego rodzaju typy pamięci producentów kart graficznych. Obecnie na rynku wyróżniamy kilka dużych producentów pamięci DRAM tj. Hynix, Elpida, Samsung, Micron. Podczas modyfikowania BIOSu odpowiednia identyfikacja typu pamięci RAM jest najważniejsza. Jest to ściśle powiązane z serią kart tego samego typu. Do karty, która pozornie wygląda i pozornie jest identyczna, montowane są różnego rodzaju pamięci, co jest szczególnie niebezpieczne, gdyż omyłkowo możemy flashować kartę tego samego typu, ale z inną pamięcią RAM, co może skończyć się problemami.

5. Co może wskazywać, że karta tego samego typu jest inna?

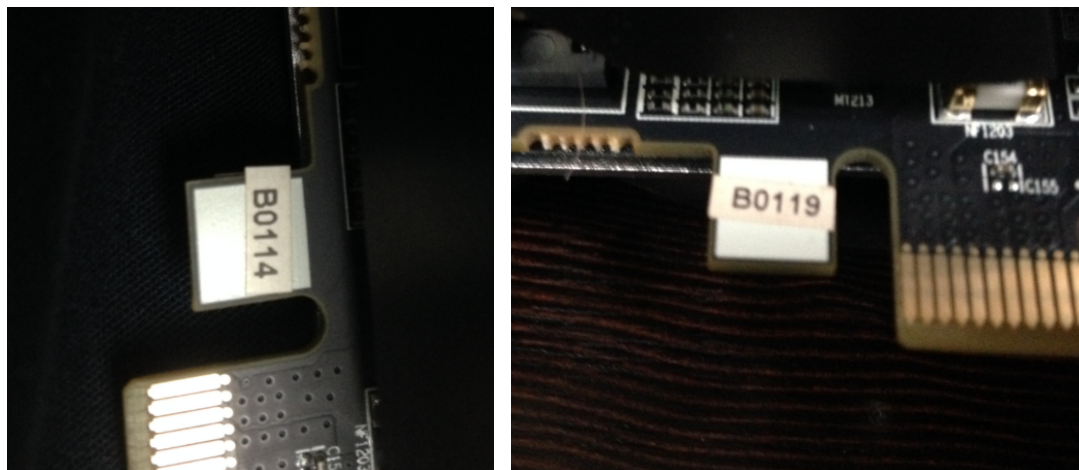
Karty mają opisany numer seryjny, który może być zaprezentowany i umieszczony w różnych miejscach na karcie graficznej. Poniższą kartę możemy zidentyfikować sprawdzając numer seryjny naklejony bezpośrednio na kartę graficzną np. A181600... sugeruje, że karta została wyprodukowana w 2018 roku. Dwie identyczne karty o identycznych parametrach

⁴ Polaris BIOS Editor

mogą się diametralnie różnić np. Karta, której numer seryjny rozpoczyna się od **A1816** może mieć całkowicie inną ilość RAM niż karta rozpoczynająca się od **A1817**.



Karty, których producent pamięci RAM różni się, różnią się oznaczeniami. Np na poniższej fotografii karty dodatkowo różnią się numerem serii B0114 i B0119, co może ale niekoniecznie informuje o tym, że bios karty może być innej wersji / pamięć może mieć innego producenta.



Przykłady Identyfikacji po serii:

- a) Hynix A1637, A1751, A1740, A1745, A1742 (31.5 Mh/s)
- b) Elpida A1641
- c) Samsung A1645, A1747, A1752, A1745 (32 Mh/s)
- d) Micron A1610, A1749, A1748 (30 Mh/s)

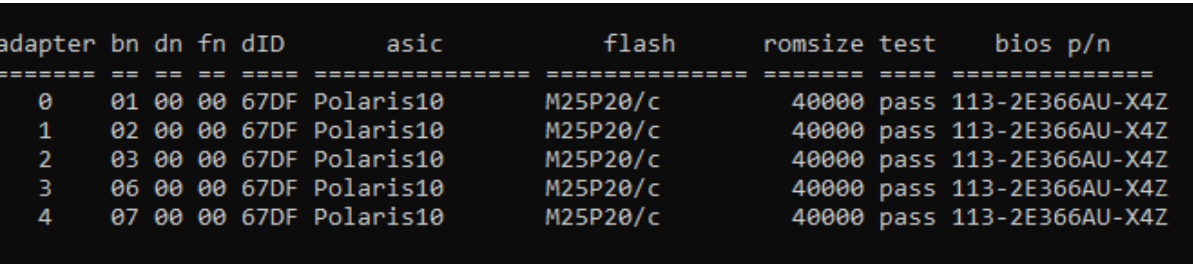
6. AtiFlash

Komenda dla karty `atiflash -p 0 biosname.rom`, gdzie 0 to numer pozycji karty. Zalecamy na kupieniu się wyłącznie na podkręcaniu jednej karty w danym momencie.

Komenda dla wielu kart `atiflash -pa biosname.rom`, niestety odradzam używania wgrywania biosu do wielu kart jednocześnie, jest to sposób, który jest przydatny jedynie wtedy kiedy mamy 100% pewność, że dane karty są tego samego typu, kości RAM są tego samego producenta, karty są tej samej serii. Nawet jeżeli posiadamy karty tego samego producenta o tych samych parametrach, które wyglądają identycznie to należy się upewnić, ponieważ zdarza się, że różnią się one biosem i każda karta zachowuje się odrobinę inaczej. Identyczność jest złudna.

Lista komend:

`Atiflash -i` wyświetla listę dostępnych kart graficznych, po podaniu numeru karty można wyświetlić kartę na konkretnej pozycji np. `Atiflash -i 0..x`



adapter	bn	dn	fn	dID	asic	flash	romsize	test	bios	p/n
0	01	00	00	67DF	Polaris10	M25P20/c	40000	pass	113-2E366AU-X4Z	
1	02	00	00	67DF	Polaris10	M25P20/c	40000	pass	113-2E366AU-X4Z	
2	03	00	00	67DF	Polaris10	M25P20/c	40000	pass	113-2E366AU-X4Z	
3	06	00	00	67DF	Polaris10	M25P20/c	40000	pass	113-2E366AU-X4Z	
4	07	00	00	67DF	Polaris10	M25P20/c	40000	pass	113-2E366AU-X4Z	

`Atiflash -ai` wyświetla zaawansowane informacje nt. Kart graficznych

`Atiflash -p 0..x nazwabiosa.rom` wgranie zmodyfikowanego BIOSa, przy czym 0..x to numer karty graficznej.

`-p <NUM> <FILE>` Zapisz BIOS <FILE>

`-s <NUM> <FILE> [SIZE]` Zapisanie BIOS z karty o <NUM> do pliku <FILE>

`-cf <FILE> [SIZE] [SUM]` Przeliczenie 16-bit checksum dla pliku <FILE>

`-cb <NUM> [SIZE] [SUM]` Przeliczenie 16-bit obrazu BIOS checksum dla adaptera o <NUM>

`-t <NUM>` Test ROM dla karty graficznej o <NUM>

`-v <NUM> <FILE>` Porównaj zawartość ROM pliku z adapterem

`-f` Siłowe flashowanie BIOS, ignoruje sprawdzanie poprawności BIOSU czy sprawdzanie, czy karta jest poprawnie załadowana

`-fa` Siłowe flashowanie BIOS, bez sprawdzania czy karta posiada już ten sam BIOS

`-fm` Siłowe flashowanie BIOS, bez sprawdzania pamięci karty

`-fs` Siłowe flashowanie BIOS, bez sprawdzania SSID karty

`-fp` Siłowe flashowanie BIOS, bez sprawdzania BIOS P/N

`-pcionly` Stosowanie jedynie dla kart graficznych podpiętych pod porty PCI

`-agp` Stosowanie jedynie dla kart graficznych podpiętych pod porty AGP

`-pcie` Stosowanie jedynie dla kart graficznych podpiętych pod porty PCI-E

`-reboot` Siłowe uruchomienie restartu systemu po poprawnym flashowaniu BIOS

7. GPU-Z

Oprogramowanie GPU-Z służy do sprawdzania szczegółowych informacji nt. podłączonych kart graficznych. Umożliwia odczyt z sensorów co przede wszystkim - jest przydatne do testowania karty graficznej np. obciążonej obliczeniami. Sensorami możemy sprawdzić np. ile energii (W), pobiera karta graficzna (przy okazji czy udał nam się undervolting), temperatury rdzenia itp. Programem GPU-Z możemy również wykonać kopię zapasową oryginalnego BIOSu (również programem ATIFLASH). TechPowerUP ponadto posiada bazę danych BIOS-ów, nadesłanych przez producentów oraz przez niezwyfikowanych użytkowników. Baza VGA BIOS Collection⁵ może również okazać się pomocna, gdy np. nie posiadamy oryginalnej wersji BIOSa karty. Poniższe zrzuty ekranu pokazują przykładowe dane dwóch typów kart graficznych.

The image displays two side-by-side screenshots of the TechPowerUp GPU-Z 2.8.0 application. The left screenshot shows the 'Graphics Card' tab for an NVIDIA GeForce GTX 1060 6GB. The right screenshot shows the 'Graphics Card' tab for an AMD Radeon RX 570 Series. Both screenshots provide a comprehensive list of hardware and software details for the respective GPUs.

Parameter	NVIDIA GeForce GTX 1060 6GB	AMD Radeon RX 570 Series
Name	NVIDIA GeForce GTX 1060 6GB	Radeon RX 570 Series
GPU	GP106	Ellesmere
Revision	A1	EF
Technology	16 nm	14 nm
Die Size	200 mm ²	232 mm ²
Release	Jul 19, 2016	Apr 18, 2017
Transistors	4400M	5700M
BIOS Version	86.06.4B.00.4E	015.050.002.001.000000
Subvendor	Gigabyte	Sapphire
Device ID	10DE 1C03 - 1458 3758	1002 67DF - 1DA2 E366
ROPs/TMUs	48 / 80	32 / 128
Bus Interface	PCIe x16 3.0 @ x16 1.1	PCIe x16 3.0 @ x1 1.1
Shaders	1280 Unified	2048 Unified
DirectX Support	12 (12_1)	12 (12_0)
Pixel Fillrate	77.8 GPixel/s	42.9 GPixel/s
Texture Fillrate	129.6 GTexel/s	171.5 GTexel/s
Memory Type	GDDR5 (Samsung)	GDDR5 (Micron)
Bus Width	192 Bit	256 Bit
Memory Size	6144 MB	8192 MB
Bandwidth	216.7 GB/s	224.0 GB/s
Driver Version	23.21.13.8813 (NVIDIA 388.13) / Win10 64	23.20.15017.3010 (Adrenalin 18.2.1) / Win10 64
Driver Date	Oct 27, 2017	Jan 31, 2018
Digital Signature	WHQL	WHQL
GPU Clock	1620 MHz	1340 MHz
Memory	2257 MHz	1750 MHz
Boost	1848 MHz	N/A
Default Clock	1620 MHz	1340 MHz
Default Memory	2257 MHz	1750 MHz
Default Shader	N/A	N/A
NVIDIA SLI	Not supported by GPU	AMD CrossFire
Computing	<input checked="" type="checkbox"/> OpenCL <input checked="" type="checkbox"/> CUDA <input type="checkbox"/> PhysX <input checked="" type="checkbox"/> DirectCompute 5.0	<input checked="" type="checkbox"/> OpenCL <input type="checkbox"/> CUDA <input type="checkbox"/> PhysX <input checked="" type="checkbox"/> DirectCompute 5.0

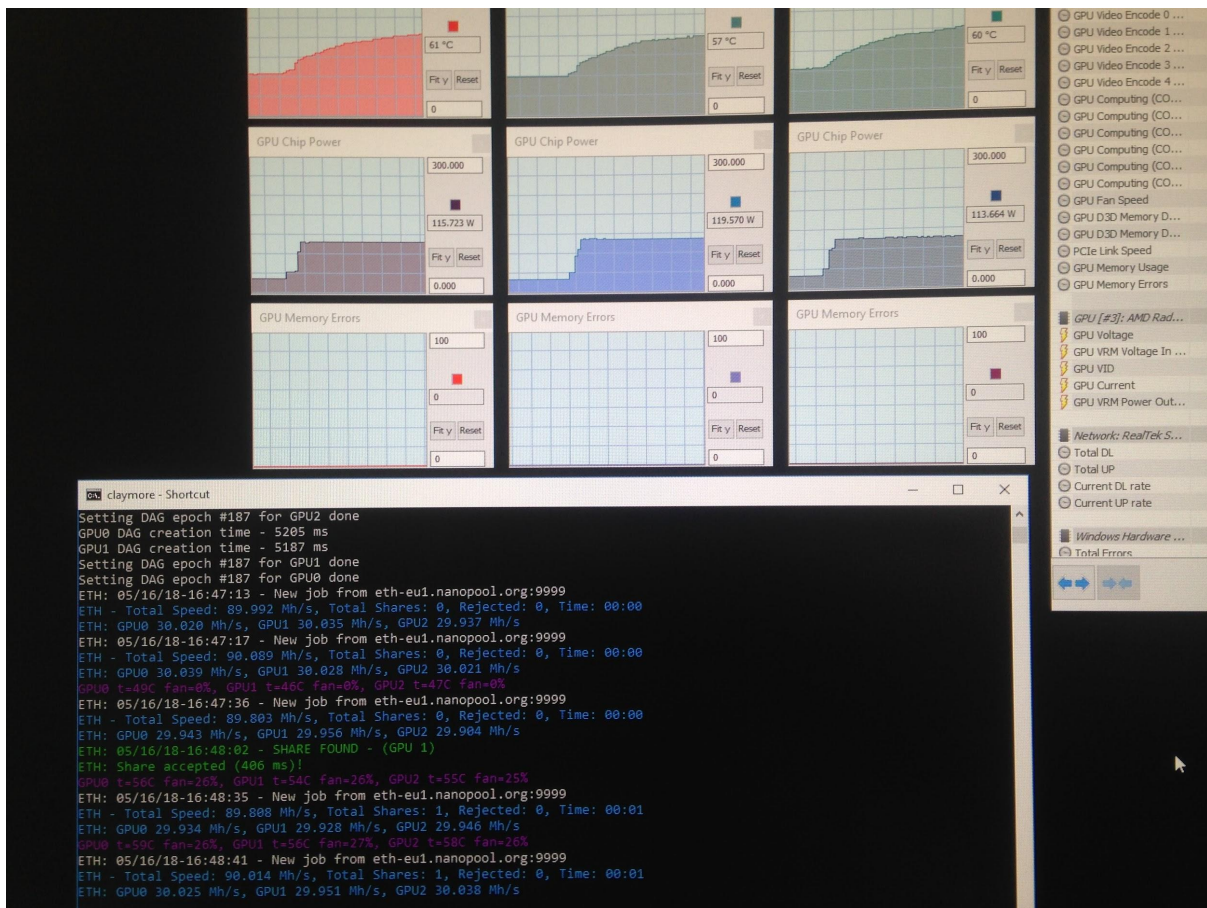
NVIDIA GeForce GTX 1060 6 GB

AMD Radeon RX 570 8GB

8. Testowanie poprawności modyfikacji kart - HWInfo

⁵ <https://www.techpowerup.com/vgabios/>

Do analizy poprawności modyfikacji kart wykorzystuje się narzędzia takie jak HWInfo, które posiadają sensory umożliwiające analizę występowania błędów w pamięci karty graficznej. Jeżeli karta graficzna jest poprawnie zmodyfikowana, takie błędy nie będą występowały. Na poniższym zrzucie ekranu zaprezentowano jak wygląda proces sprawdzania błędów kart graficznych. W tym celu, przystępnie pokazane wyniki z sensorów poszczególnych kart mogą zobrazować aktualny stan modyfikacji. Test powinien odbyć się na obciążonej i rozgrzanej karcie graficznej, dlatego uruchomiony zostaje program obliczeniowy. Zwykle wystarczy test ok. 15 minutowy, żeby zorientować się czy pojawił się chociaż jeden błąd karty graficznej. Jeżeli tak się stanie, należy wykonać ponowną modyfikację BIOSu danej karty graficznej. Błędy pojawiają się nawet mimo tego, że wszystkie karty są równorzędne i powinny być takie same, nawet jeśli pamięci RAM są tych samych producentów. Ważnym parametrem w tym zakresie jest jakość kości ASIC karty graficznej (jakość którą również można sprawdzić programem HWInfo). Generalnie trzeba się zachowywać tak jakby, każda karta tego samego modelu była odrobinę inna.



W przypadku kiedy karta będzie źle podkręcona to objawami jakie będą występowały będą np.:

- Komunikaty o błędnej pracy karty graficznej w oprogramowaniu obliczeniowym np. incorrect shares
- Częste zawieszenia karty graficznej przy obciążeniu np. zawieszenie OpenCL

- Błędne raportowanie siły obliczeniowej - np. karta może zwracać informacje o tym, że jest bardzo mocna (np. pokazywać 50 mh/s), a jej rzeczywista efektywność nie istnieje
- Wysoki pobór mocy, błędy temperature 511 itp. (które mogą wskazywać również na błędy riserów).

Poniższy zrzut ekranu obrazuje w jaki sposób wyglądają przykładowe błędy pracy źle zmodyfikowanej karty pamięci, które często wpływają na pracę całej maszyny obliczeniowej. Efektywność takiej maszyny obliczeniowej jest mimo tego co "raportuje" o wiele niższa, często nawet przeszkadza w normalnej pracy całej maszyny, będąc przy tym dodatkowym źródłem ciepła.

```
Automatic detection of best -dcric values completed
GPU0 t=56C fan=20%, GPU1 t=57C fan=20%, GPU2 t=49C fan=20%, GPU3 t=58C fan=20%, GPU4 t=56C fan=20%
ETH: 05/15/18-21:13:24 - New job from eul.ethermine.org:4444
ETH - Total Speed: 181.542 Mh/s, Total Shares: 0, Rejected: 0, Time: 00:00
ETH: GPU0 30.251 Mh/s, GPU1 30.261 Mh/s, GPU2 30.262 Mh/s, GPU3 30.274 Mh/s, GPU4 30.274 Mh/s
ETH: 05/15/18-21:13:35 - New job from eul.ethermine.org:4444 [ 82.534712] amdgpu 0000:02:00.0:
[ 82.534715] amdgpu 0000:02:00.0: VM_CONTEXT1_PROTECTION_FAULT_ADDR 0x001CED2C
[ 82.534718] amdgpu 0000:02:00.0: VM_CONTEXT1_PROTECTION_FAULT_STATUS 0x08008002
[ 82.534724] amdgpu 0000:02:00.0: VM fault (0x02, vmid 4) at page 1895724, read from 0x001CED2C
[ 82.534733] amdgpu 0000:02:00.0: GPU fault detected: 147 0x09600402
[ 82.534735] amdgpu 0000:02:00.0: VM_CONTEXT1_PROTECTION_FAULT_ADDR 0x00206BAE
[ 82.534738] amdgpu 0000:02:00.0: VM_CONTEXT1_PROTECTION_FAULT_STATUS 0x08004002
[ 82.534743] amdgpu 0000:02:00.0: VM fault (0x02, vmid 4) at page 2124718, read from 0x00206BAE

ETH - Total Speed: 181.427 Mh/s, Total Shares: 0, Rejected: 0, Time: 00:00
ETH: GPU0 30.216 Mh/s, GPU1 30.301 Mh/s, GPU2 30.243 Mh/s, GPU3 30.240 Mh/s, GPU4 30.167 Mh/s
GPU #1 got incorrect share. If you see this warning often, make sure you did not overclock

[1]+ Stopped sh start.bash
eth002:~/claymore$
```

Kolejnym testem sprawdzającym powinno być uruchomienie maszyny, która wydaje nam się stabilna na okres np. 3h w warsztacie, zanim zostanie przeniesiona do serwerowni.

9. Źródła w internecie

Najlepszym źródłem informacji o modyfikacji BIOS-ów pod obliczenia Blockchain są fora i blogi bezpośrednio dotyczące tej technologii. Jedno z najstarszych źródeł informacji jest forum [bitcointalk](http://bitcointalk.org)⁶, które posiada odpowiedni dział wyspecjalizowany do miningu oraz zaawansowane tematy. W społecznościach tych znane są takie osoby jak Heliox czy Eliovp, które specjalizują się w tzw. MODach BIOSów. W wątkach dotyczących miningu i tzw. kręcenia kart, uzyskamy potrzebną wiedzę, aby samodzielnie rozpocząć przygodę z optymalizacją kart graficznych.

Linki:

⁶ bitcointalk.org - jedno z najstarszych źródeł dot. technologii blockchain, miningu, kryptowalut

1. <https://bitcointalk.org/index.php?topic=1954245.0> - szczegółowy tutorial o modyfikacji BIOSów kart graficznych AMD. Bardzo dobre miejsce, aby rozpocząć przygodę z modyfikacją BIOSów
2. <https://forum.ethereum.org>
3. <https://mining-bios.eu> (Kod: 10 % Indeedminers10) - Znajduje się tutaj zmodyfikowana wersja programu PolarisBiosEditor, automatyzująca część pracy związanej z optymalizacją kart graficznych.
4. Heliox: <https://forum.ethereum.org/profile/heliox>
5. Eliovp: <https://bitcointalk.org/index.php?action=profile:u=236770>

10. Maszyny obliczeniowe - budowa puli maszyn

Wszystkie maszyny obliczeniowe w serwerowni powinny mieć przydzielony adres IP z końcówką umożliwiającą szybką identyfikację, co z kolei pozwoli na miare szybkie reagowanie na procesy zachodzące podczas obliczeń takie jak np. zawieszanie się skryptu obliczeniowego. Dostęp do maszyn powinien być taki sam dla każdej maszyn z puli lub w określonym systemie. Ponadto maszyny powinny być w sieci VPN, aby zapewnić do nich dostęp z dowolnego miejsca. Poniższy zrzut ekranu przedstawia przykładową pulę maszyn obliczającą bloki w technologii blockchain Ethereum.

Name ↕	Reported Hashrate ↕	Current Hashrate ↕	Valid Shares ↕	Stale Shares ↕	Invalid Shares ↕	Last seen ↕
b001	181.6 MH/s	129.2 MH/s	113 (96%)	5 (4%)	0 (0%)	11 minutes ago
b002	170.7 MH/s	190.3 MH/s	168 (97%)	5 (3%)	0 (0%)	12 minutes ago
b003	205.9 MH/s	188.4 MH/s	167 (98%)	4 (2%)	0 (0%)	11 minutes ago
b004	200.8 MH/s	180.6 MH/s	156 (94%)	10 (6%)	0 (0%)	11 minutes ago
b005	205.7 MH/s	212.1 MH/s	187 (97%)	6 (3%)	0 (0%)	12 minutes ago
b006	221.1 MH/s	206.9 MH/s	183 (97%)	5 (3%)	0 (0%)	12 minutes ago
b007	182.8 MH/s	68.4 MH/s	59 (94%)	4 (6%)	0 (0%)	11 minutes ago
b008	201.3 MH/s	204.3 MH/s	178 (95%)	9 (5%)	0 (0%)	11 minutes ago
b009	192.4 MH/s	200.3 MH/s	177 (97%)	5 (3%)	0 (0%)	11 minutes ago
b010	217.7 MH/s	252.8 MH/s	221 (96%)	10 (4%)	0 (0%)	12 minutes ago
b012	151.6 MH/s	142.2 MH/s	126 (98%)	3 (2%)	0 (0%)	12 minutes ago
b013	181.4 MH/s	163.8 MH/s	139 (91%)	13 (9%)	0 (0%)	11 minutes ago
b014	182.9 MH/s	163.6 MH/s	142 (95%)	8 (5%)	0 (0%)	11 minutes ago
b015	240.5 MH/s	229.4 MH/s	200 (95%)	10 (5%)	0 (0%)	11 minutes ago
b016	203.2 MH/s	179.6 MH/s	159 (98%)	4 (2%)	0 (0%)	12 minutes ago

11. Notatnik roboczy

Ustawienia minerów do testowania maszyn obliczeniowych pod Ethereum

a) Oprogramowanie ethminer:

```
/home/eth/ethminer --farm-recheck 200 -G -S eu1.ethermine.org:4444 -FS us.ethermine.org:4444 -O 0x79AE2031C5eA2E1bfD96d413179ab38504B7E3Cb.E_TEST --cl-local-work 256 --cl-global-work 16384
```

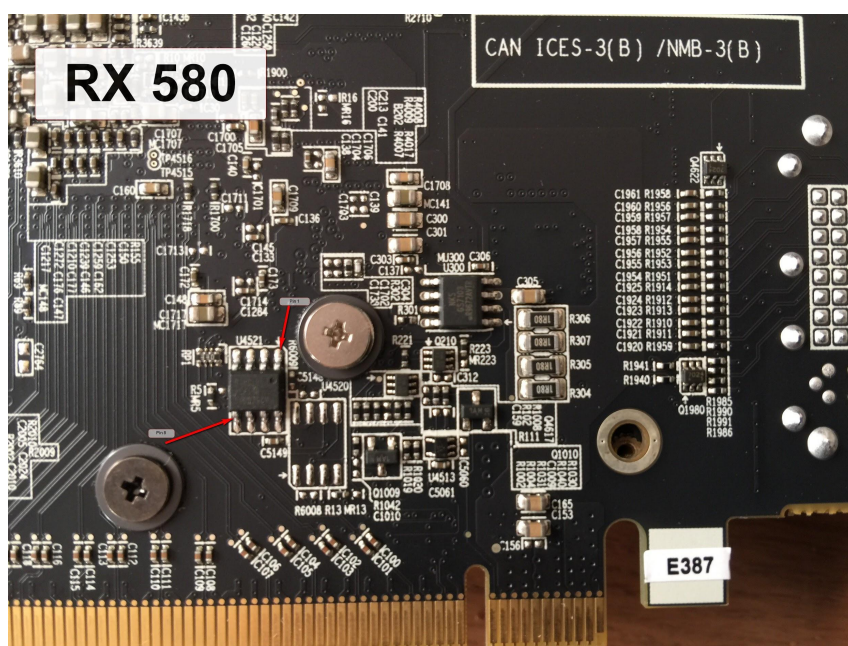
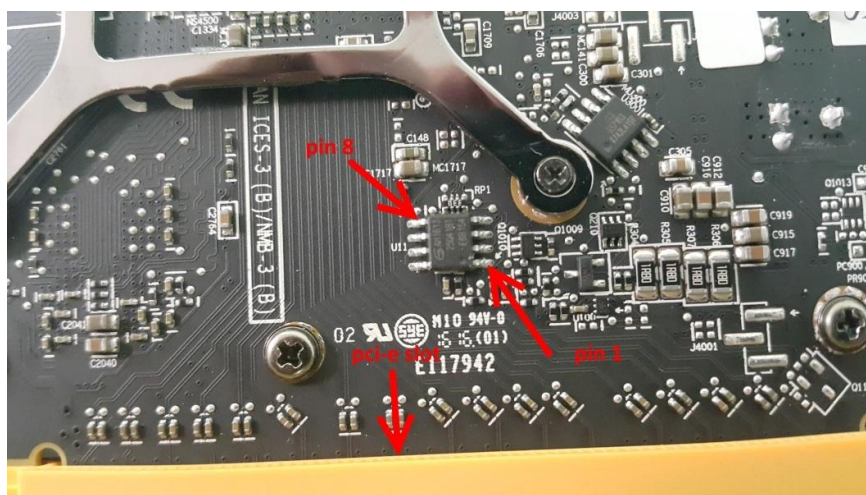
```
ethminer.exe --opencl-platform 1 -M --farm-recheck 200 -G -S eu1.ethermine.org:4444 -FS us.ethermine.org:4444 -O 0x79AE2031C5eA2E1bfD96d413179ab38504B7E3Cb.E_TEST --cl-local-work 256
```

```
ethminer.exe --opencl-platform 1 -M --farm-recheck 200 -G -S eu1.ethermine.org:4444 -FS us.ethermine.org:4444 -O 0x79AE2031C5eA2E1bfD96d413179ab38504B7E3Cb.E_TEST --cl-local-work 256 --cl-global-work 8192 (22Mh/s)
```

```
ethminer.exe -G -P stratum+ssl://0x79AE2031C5eA2E1bfD96d413179ab38504B7E3Cb.B001@eu1.ethermine.org:5555 -RH --cl-local-work 256 --cl-global-work 8192 --opencl-platform 1
```

TIPS:

1. Czasami Atiflash wywala, że się nie powiodło flashowanie, wtedy należy w systemie Windows w menadżerze urządzeń wyłączyć kartę, a następnie ją włączyć, jeśli atiflash wykrywa, że BIOS jest wgrany, należy użyć polecenia atiflash -p nr_karty -f nazwa.rom (z parametrem -f wgrać bios siłowo).
2. Czasami (u nas zdarzyło się to jeden raz), udaje się wgrać BIOSa kompletnie niepoprawnego np. z zupełnie innej karty graficznej. Efektem jest brak możliwości włączenia komputera z wpiętą kartą w port PCI-E (czarny ekran, brak boot). Wtedy mamy kartę tzw. Brick (cegłę). Istnieje sposób na reanimację karty (**sprawdzone dla kart rx 470/480/570/580**). **W pierwszej kolejności powinniśmy się upewnić jakie piny ma bios karty i czego dotyczą.**



Aby umożliwić wgranie biosu do takiej karty, należy połączyć pin 1 z pinem 8 podczas bootowania komputera, w momencie kiedy system się bootuje, należy je odłączyć. Trzeba pamiętać również o poprawnym zidentyfikowaniu czy dana kość jest kością biosu oraz który pin jest pierwszy - zwykle jest opisany np. U11.

SETUP Systemu:

1. USB Stick z linuxem 16.04.4 LTS (kernel 4.4.x - lepiej nie aktualizować, okazuje się, że jeżeli system jest zainstalowany z automatycznymi aktualizacjami to zdarza się utrata mocy obliczeniowej na nowszych kernelach. Automatyczny upgrade kernela po np. Restarcie maszyny np z 4.10 do 4.13 spowodowała u nas spadek mocy obliczeniowej o niemal 50%. Najlepiej zainstalować system w trybie aktualizacji wymagających manualnych czynności.
2. Instalacja oraz utworzenie podobnej konfiguracji kont w systemie

3. W naszym przypadku utworzyliśmy sobie projekt w repozytorium GIT/dysku sieciowym, do którego będziemy wrzucali konfiguracje, potrzebne pliki instalacyjne, skrypty sh automatyzujące część pracy.
4. należy pamiętać o `sudo usermod -a -G video eth`, które doda użytkownika eth do grupy video, podobnie należy zrobić z userem root (nie mamy pewności czy koniecznie).

Notatki robocze:

Ustalenie Prędkości Wentylatorów na karcie graficznej GPU typu AMD w sterownikach AMDGPU PRO

<https://github.com/DominiLux/amdgpu-pro-fans#notes>