International Journal on Advanced Science Engineering Information Technology

Vol.9 (2019) No. 5
ISSN: 2088-5334

Determining Optimal Mining Work Size on the *OpenCL* Platform for the *Ethereum Cryptocurrency*

Pavel V. Sukharev[#1], Dmitry S. Silnov[#2], Maxim O. Shishkin[#3]

[#]*Department of Computer Systems and Technologies, National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, Russia*
E-mails: [1]suharev_p@mail.ru; [2]ds@silnov.pro; [3]petshig@yandex.ru

***Abstract*— In terms of *cryptocurrency*, mining is a process of creating a new transaction block to add it to the blockchain. The *cryptocurrency* protocol should ensure the reliability of new transaction blocks. One of the popular mining protocols is the Proof-of-Work protocol, which requires the miner to perform a certain work to verify its right to add a new block into the blockchain. To perform this work, high-performance hardware is used, such as GPU. On the program level, hardware needs special computing framework, for example, CUDA or OpenCL. In this article, we discuss *Ethereum cryptocurrency* mining using the *OpenCL* standard. The *Ethereum cryptocurrency* is the most popular *cryptocurrency* with GPU-based mining. There are several open-source implementations of the *Ethereum cryptocurrency* miners. The host-part of the *OpenCL*-miner is considered, which makes the research results independent of the mining algorithm and allows using the results of the research in the mining of other *cryptocurrencies*. During the research, we have found the problems, which lead to mining productivity loss, and we are looking for the ways to resolve these problems and thus increase mining performance. As part of solving these problems, we have developed the algorithm for the functioning of the miner and proposed the methodology of determining the optimal size of *OpenCL* work, which allows to reduce the impact of problems found and achieve maximum mining productivity using *OpenCL* framework.**

***Keywords*— cryptocurrency mining; Ethereum; OpenCL; performance optimization.**

I. INTRODUCTION

Cryptocurrencies are relatively new and rapidly developing section of the digital economy. The purpose of cryptocurrencies is to create a distributed system of digital currency. Cryptocurrencies has a number of benefits in comparison to ordinary money - transaction openness, high protection, restricted and known ahead emission, general system transparency. Cryptocurrency funds exchange occurs through transactions. List of all transactions is stored in blockchain. Once the transaction is completed, it remains in the blockchain forever [1], [2].

To conduct a new transaction, it must be added in blockchain. Mine nodes, which compose, add transactions a new transaction blocks for blockchain. To achieve integrity and reliability of transaction, validation mechanisms are used, which miners must follow to have opportunity to add new block to blockchain [3]. One of such mechanisms is Proof-Of-Work, which idea is that miners should perform a certain amount of work to prove the system that it has the right to add a new block [4].

One of the main characteristics of cryptocurrency is its decentralization. Decentralization is a factor, which ensures the system reliability. For example, high centralization increases the risk of "attack 51%" which can inflict a serious harm to cryptocurrency system [4], [5]. To achieve decentralization it is necessary to avoid concentration of system computation resources. However, according to Bitcoin experience, with the cryptocurrency popularity growth the specialized mining solutions are developed, which concentrates computation power in relatively small group of specialized companies and makes cryptocurrency vulnerable for attack [6].

Such problem is tried to solve by creating the cryptocurrency, which has resistance to possibility of creation special mining hardware (ASIC) [7], [8]. One of such cryptocurrencies is Ethereum coin, which limits mining speed with large number of input-output operations, which are difficult to optimize [9], [10]. Now, Ethereum mining uses general-purpose GPUs available to wide range of miners, which increases system decentralization.

To create a new valid block in Ethereum cryptocurrency system, miner must found such block header that its hash value, calculated by ethash algorithm [11], is below the specified target difficulty. To achieve this miners sort through the values of nonce field, which is the part of block header, and count hash value for every nonce. Since the calculations of nonces do not depend on each other, it is a

work with a high level of parallelism. Miner should quickly calculate as many hashes for different nonces as possible. For this purpose, high performance software frameworks are used.

GPU mining is performed using, depending on the GPU manufacturer, CUDA [12] and OpenCL [13] technologies. OpenCL is an open standard for writing program executing parallel computation on different types of processing devices, such as CPU, GPU, FPGA. OpenCL program consists of a host-part and kernel-part. Host-part is executed on control device (generally, CPU). It configures and sends computational task to the GPU, and read the results. Kernel-part is responsible for task computation and runs on high-performance computation device (for example, GPU). OpenCL specifies programming language, based on C. In addition, OpenCL 2.1 implements OpenCL C++ [14]. In addition, OpenCL provides API for host-part and kernel-part interaction. Because OpenCL is an open standard, everybody can port it at any computing hardware. There are many OpenCL standards both proprietary (such as NVIDIA OpenCL implementation) and open-source (for example, AMD ROCm OpenCL implementation). There are many enthusiasts, who can also develop their own OpenCL compiler. For example, pocl project [15] provides LLVM compiler for a different set of architectures. Now, there are implementations of OpenCL for all popular GPUs exist.

In this article, we discuss Ethereum cryptocurrency mining using the OpenCL standard. We find the problems, which lead to mining productivity loss and looking for ways to increase its performance. The host-part of the OpenCL-miner is considered, which makes the research results independent of the mining algorithm and allows using the results of the research in the mining of other cryptocurrencies.

II. MATERIAL AND METHODS

A. OpenCL

Consider the basics of the OpenCL framework. When user wants to execute computation program on OpenCL, he should do an initial configuration. First, user should choose OpenCL platform, which will hold the work processing. Further, user should choose OpenCL devices, which would be used for computation. After that user defines OpenCL context for chosen devices. OpenCL context manages memory, command-queues, kernel objects, etc.

OpenCL uses command-queues for device task processing. Command-queues are defined in bounds of a OpenCL context. In the OpenCL command-queues, you can send commands such as reading or writing a memory buffer, or sending tasks to execute on device. To create a new computational task, you should compile OpenCL kernel object from sources, and send execution arguments. Creation of the command queue, kernel object, and buffers occurs once within the initialization.

To launch kernel object, OpenCL uses *clEnqueueNDRangeKernel()* function. This function accepts command-queue in which program will be enqueued, the kernel object itself and work sizes *local_work_size* and *global_work_size* as parameters. Parameter *local_work_size* defines size of an OpenCL work-group. Parameter *global_work_size* defines total work size. For reading results, function *clEnqueueReadBuffer()* is used. It accepts command-queue and reading buffer as parameters. In addition, it can be specified, if buffer read is blocking or not.

The minimum OpenCL computational element is the work-item. One work-item corresponds to one thread on the real computing device. The total number of work-items is determined by the *global_work_size* parameter. If the number of work-items is greater than the allowed number of threads running on the device, the extra work-items are waiting for the device resources to be freed. Working elements are combined into work-groups. The size of the work-group is determined by the *local_work_size* parameter. For each device there is an optimal size of the work-group, which defined by its the hardware architecture and allows to achieve maximum performance [16].

B. Open-source Ethereum Miners

There are few miners for Ethereum cryptocurrency. Since miner implementation quality is defined by its mining speed, users tend to choose the most productive mining solution, and highlight it among others, thus decrease mining implementation variety. Some of the miners have a closed code and require the user to pay developer fees for its usage. An example of such closed-source miner is a Claymore project [17].

In addition, there are few open-source Ethereum miners, such as sgminer and ethminer. Sgminer is a GPU miner for a large number of cryptocurrencies, including Ethereum. Developers can add new cryptocurrency mining realisations as a kernel-parts of OpenCL code.

Ethminer mining is an open-source miner solely for Ethereum cryptocurrency [18]. It supports both CUDA and OpenCL frameworks. Ethminer is the most popular open-source Ethereum miner.

In this article we use open-source ethminer for mining and testing.

Current open-source Ethereum miners OpenCL implementations have cyclic mining algorithm:

- Launching mining process using the OpenCL function clEnqueueNDRangeKernel() with specified work size values local_work_size and global_work_size.
- Blocking waiting for work completion and reading mining result using the OpenCL function clEnqueueReadBuffer ().

After completion of reading mining results from GPU new mining command will be launched with new nonce parameters and cycle will be repeated. If valid block is found, then miner will send its info to others. If new mining task is received, miner changes its mining parameters. So, the miner repeatedly launches GPU processing task with different nonces for current header and looking for proper result. Each OpenCL device work-item takes a specific nonce depending on its ID.

This mining process is used in OpenCL host-part implementation of all popular open-source Ethereum miners. Fig.1 shows the algorithm of the completely mining process.

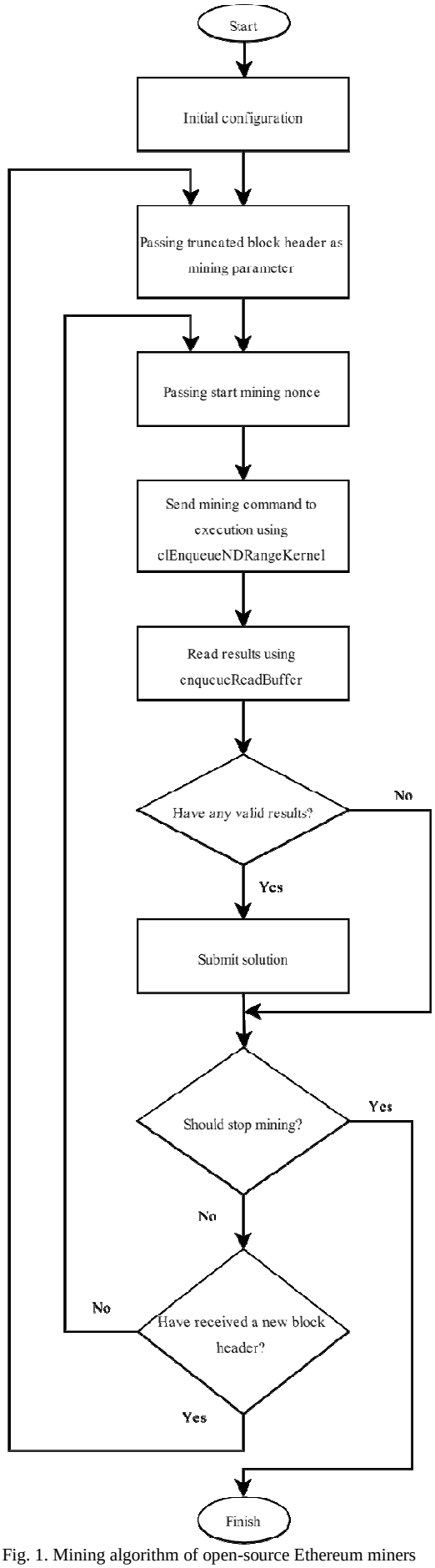

Fig. 1. Mining algorithm of open-source Ethereum miners

C. Cryptocurrency Mining Process

Mining uses the generated header of the current block. Block header contains information about the previous block, as well as Merkle Patricia Tree root hashes [19], which are the root hashes for Ethereum block trees: storage tree, transaction tree, receipt tree. These parameters differ for each new block in cryptocurrency system. Thus, the mining process can only be carried out for the actual block header with the current transaction list and information about the previous block.

When a new block is created in the cryptocurrency system and added to the blockchain, the information about received block is distributed to all system miners. At the same time, a change of work takes place on the miners, and the miners begin to form a new block based on the new data obtained.

Since GPU execution requires a certain computation time, there is often occurs a situation when the new computing task comes from server, while the GPU has not finished current cycle of execution of old work. This happens because GPU execution is not instantaneous, and miner cannot send another work to the device before finishing the previous one.

In this case, the work that is processed on GPU becomes obsolete and unsuitable for obtaining a correct result. As a result, part of work time of miner is wasted. The percentage of wasted work depends on the time spent by the GPU for one cycle of work. Therefore, execution time of one cycle of work should be decreased to reduce percentage of wasted work.

Execution time of one cycle of work depends linearly on whole work size *global_work_size*, which we specify as an OpenCL parameter every time we send new work to GPU. This parameter determines the number of hashes that the miner must calculate in the current work cycle [20].

D. Mining Overhead Time

It must be noted that every software has some overhead time, which is spent on various machine operations that are not related to main work. These operations are such as context change, new work uploading, results reading. Typically, the execution time for this operation is negligible.

However, in case of mining with small size of the *global_work_size*, and, as a result, small execution time, this overhead time can amount a significant percent of the whole time spent to work.

Because miner implements cyclic calculation of block header hash, it calls identical OpenCL functions cyclically. When miner cycle work size for *clEnqueueNDRangeKernel()* function is reduced, number of cycles per second increases, which leads to number of *clEnqueueNDRangeKernel()* and *clEnqueueReadBuffer()* calls increase.

It can be expected that in case of launching two miners with different size of work on the same equipment, a miner with a large work size will perform a slightly larger amount of hash calculations.

E. Determining Lost Work Percentage

As a result, it can be argued about presence of two factors, which reduce mining productivity. Influence of one of this factors enhance when work size of one mining cycle gets

bigger, and influence of second factor enhance when work size of mining cycle decreases. Fig. 2 shows this problem schematically. As we can see, a miner with a large work size has less total overhead time that allows it to calculate more hashes per second, but in the case of a new block, miner loses more work due to a slower switching of the task.

The aim of this work is to find the optimal size of work in one mining cycle, which allows to reach the maximum value of the calculated actual hashes in case of long-term mining in the real Ethereum system.

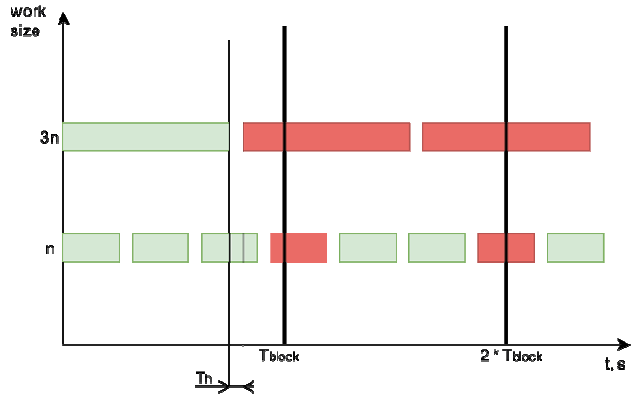

Fig. 2. Diagram of the lost work of the miner when a new block is received and when the work size is reduced in one cycle.

Evaluate percentage of work, which miner loses when new block header from server is receiving. In case of task change, the whole cycle of calculations is lost, regardless of when exactly the information about the new task came. Therefore, during the mining, one cycle of calculations will be lost each time the information about the new task has arrived. The percentage of lost work will depend on the execution time of one cycle and the time of obtaining a new task by the miner. Suppose that execution time of one cycle of miner is t_w, then the percentage of lost work is described by equation (1).

$$k_1 = \frac{t_w}{T_{block}} \tag{1}$$

where T_{block} - value equal to new block mean time in Ethereum cryptocurrency system and which is constant during long periods of time [21]. Fig. 3 shows the change of new block mining mean time of Ethereum cryptocurrency. As we can see, system tends to maintain constant mean time of new block over long periods.

By decreasing *global_work_size* value, we can reduce the time spent per cycle and thereby reduce the percentage of useless work in equation (1).

Let us measure influence of overhead time on mining efficiency. Overhead time is a part of completely mining cycle time. Thus, if we denote overhead time as t_h, then percent of time, spending on non-important operations, will be equal to

$$k_2 = \frac{t_h}{t_w} \tag{2}$$

$$t_w = t_h + t_u \tag{3}$$

where t_u - time of useful work in one cycle of mining, spent directly on hash calculation.

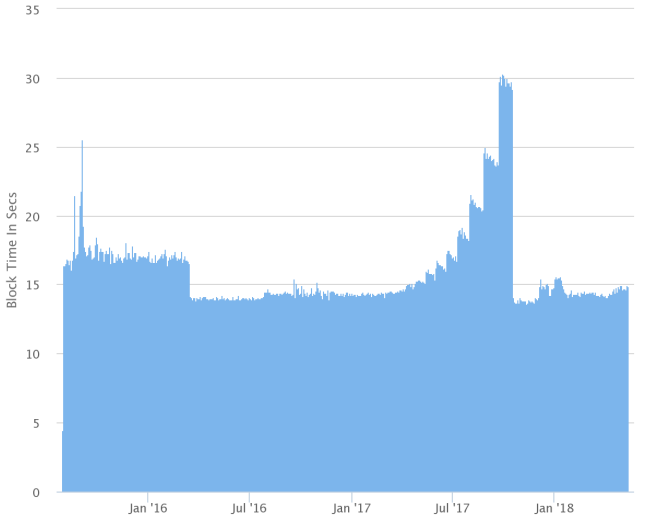

Fig. 3. Change of new block mining mean time in Ethereum cryptocurrency since its launch

Equation (2) limits infinite reduction of the work size sent to GPU. However, it needs to be noted, that t_h does not depend on *global_work_size* and is always approximately the same within the same computing platform.

In order to calculate the optimal size of the *global_work_size* mining operation, it is necessary to determine the time of the work cycle in which the useful time of the miner will be maximal. As follows from equations (1), (2) time t_w should achieve some compromise value, when coefficients k_1 and k_2 will have minimal impact.

F. Determining Optimal Cycle Execution Time

It can be seen that coefficients k_1 and k_2 have similar meaning. They both describe the percentage of time the miner spends on useless work. Therefore, after summing (1) and (2), we will get the value of the total coefficient, describing the percentage of time spent on all useless work.

$$k = \frac{t_w}{T_{block}} + \frac{t_h}{t_w} \tag{4}$$

$$k = k_1 + k_2 \tag{5}$$

Fig. 4 represents the form of the function (4). The optimal t_w value corresponds to the situation when the left part of (4) achieves a minimum value. To find this situation, differentiate the equation and find points of minimum:

$$\frac{1}{T_{block}} - \frac{t_h}{t_w^2} = 0 \tag{6}$$

$$t_w = \sqrt{(T_{block} \times t_h)} \tag{7}$$

Equation (7) describes the optimal execution time for one Ethereum mining cycle for a specific hardware system with

overhead time t_h and at a time when the new block mean time in the system is equal to T_{block}.

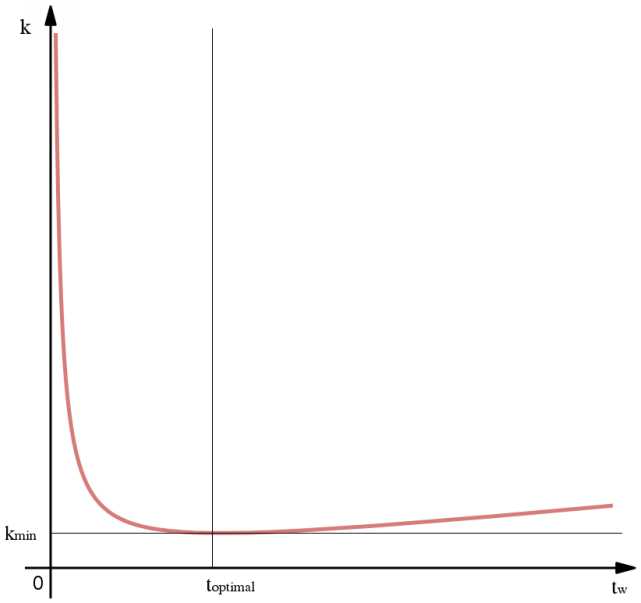

Fig. 4. Graph of work loss versus cycle execution time function

It should be noted that the new block time varies for each new block, as the overhead time varies for each calculation cycle. In equation (7), we operate with the mean values of these time variables, since we consider the performance of the miner in the case of long-term operation.

III. Results and Discussion

A. Experimental Determining of the Optimal Work Size

In order to determine the optimal running time of one cycle, we need to find an approximate mean overhead time. It can be found for a specific hardware experimentally by running two cycles of mining speed measurements for different values of *global_work_size*.

In the course of the experiment, it is necessary to obtain the following parameters for different mining work sizes:

- total number of calculated hashes,
- number of actual hashes,
- number of obsolete hashes,
- number of cycles of calculations,
- time of experiment,
- the amount of new block headers during the experiment.

The number of actual hashes is required to estimate the increase in real mining productivity. The total number of hashes demonstrates the absolute speed of mining. It is expected that the total number of hashes will be directly proportional to the work size sent for calculation in one mining cycle. The number of obsolete hashes determines the percentage of obsolete work performed by the miner. The hashes are considered obsolete if they were calculated in a loop, during which a new job came from the server. With the total number of hashes and number on obsolete hashes, we can calculate count of work, which is lost due to mining task switching.

The number of cycles is an important parameter, it determines how many times *clEnqueueNDRangeKernel()* and *clEnqueueReadBuffer()* functions were invoked during the experiment. It is expected that the number of cycles is inversely proportional to the size of the work, since a larger work size requires a longer execution time. The number of cycles is required to link the duration of one cycle and the amount of work in one cycle.

The experiment time is a parameter that allows us to accurately determine the time for which all the cycles of mining were performed. Experiment time is counted with millisecond precision.

Since the experiment is performed under real mining conditions, one should take into account the heterogeneity of the time of finding a new block. For this purpose, the number of new blocks calculated in the Ethereum system during the experiment is counted.

The experiment was conducted in the Ethereum-pool nanopool.

The experiment was carried out on hardware that uses GPU NVIDIA GTX970m for mining. For testing two work sizes were selected: 32 x 32768 and 32 x 8192. This choice is explained by the fact that default ethminer miner work size is 128 x 8192, while optimal OpenCL local work size parameter for GPU NVIDIA GTX970m is 32. Therefore, size 32 x 32768 has the same *global_work_size* as default 128 x 8192 and has *local_work_size* 4 times less than default. Both experiments lasted 10 minutes. The results of experiment are shown in Table 1.

Note that this test fulfils equation (1). Despite the fact that in experiment No. 2 the total hash count is less than in experiment No. 1, the number of actual hashes in test No. 2 bigger because of shorter execution time of one cycle and faster miner block switching. At the same time, count of received new block headers during the test was bigger in test No. 2.

Let us calculate mean overhead time for the received data. Denote t_s - overall test time, n - count of work cycles, which were calculated during the experiment, then one cycle work time, is calculated by equation:

$$t_w = \frac{t_s}{n} \tag{8}$$

$$t_u + t_h = t_w \tag{9}$$

t_u - actual work time on GPU, hence it is proportional to GPU work size. Then if in case of experiment No. 2 execution time is t_u then in experiment No. 1 execution time will be equal 4 * t_u.

Write equation (8) for two tests.

$$4 \times t_u + t_h = \frac{t_{s1}}{n_1} \tag{10.1}$$

$$t_u + t_h = \frac{t_{s2}}{n_2} \tag{10.2}$$

From given pair of equations follows:

$$t_h = \frac{(4\times\frac{t_{s2}}{n_2} - \frac{t_{s1}}{n_1})}{3} \quad (11)$$

$$t_h = 0{,}245485688\ ms \quad (12)$$

TABLE I

RESULTS OF MINING WITH DIFFERENT WORK SIZE

No.	Work Size	Total Hash Count	Actual Hash Count	Obsolete Hash Count	Count of Mining Cycles	Execution Time	Count of New Block Headers During Experiment
1	32x32768=1048576	3382706176	3346006016	36700160	3226	599934	35
2	32x8192=262144	3369598976	3357540352	12058624	12854	599976	46

TABLE II

THE COUNT OF CALCULATED HASHES FOR OPTIMAL WORK SIZE

No.	Work Size	Total Hash Count	Actual Hash Count	Obsolete Hash Count	Count of Mining Cycles	Execution Time	Count of New Block Headers During Experiment
1	32x32768=1048576	3382706176	3346006016	36700160	3226	599934	35
2	32x8192=262144	3369598976	3357540352	12058624	12854	599976	46
3	128x8192=1048576	3380609024	3337617408	42991616	3224	599879	41
4	32x10377=332064	3380079456	3368457216	11622240	10179	599952	35

TABLE III

THE COUNT OF CALCULATED HASHES FOR OPTIMAL WORK SIZE IN CASE OF MEAN BLOCK TIME

No.	Work size	Count of mining cycles	Total Hash Count	Actual Hash Count	Obsolete Hash Count	Execution Time
1	32x32768=1048576	3226	3382706176	3338665984	44040192	599934
2	32x8192=262144	12854	3369598976	3358588928	11010048	599976
3	128x8192=1048576	3224	3380609024	3336568832	44040192	599879
4	32x10377=332064	10179	3380079456	3366132810	13946646	599952

With a known mean overhead time t_h, mean block time and equation (7) optimal miner cycle execution time can be found. Mean block time at the time of experiment was 14210 milliseconds.

$$t_w = \sqrt{14210\times 0{,}245485688} \quad (13)$$

$$t_w = 59{,}06226902 ms \quad (14)$$

Now, with experimental data and known miner working time, it is possible to find optimal miner work size. Denote w as a multiplier of the useful work of the miner. Then make equation:

$$w\times t_u + t_h = t_w \quad (15)$$

Take one of equations (10.1), (10.2) and calculate coefficient w.

$$w\times t_u + t_h = t_{w1} \quad (16.1)$$

$$t_u + t_h = t_{w2} \quad (16.2)$$

$$w = 1 + \frac{t_{w1} - t_{w2}}{t_u} \quad (17)$$

$$w = 1{,}2667 \quad (18)$$

Result optimal work size is $32\times(8192\times w) = 332064$.

Calculate the expected percentage of lost work for each of the factors for the optimal work size found. We substitute the calculated t_w value into equations (1) (2), and calculate the coefficients k_1 and k_2. Take the actual T_{block} and overhead time t_h. In this case, we get the following values of coefficients k_1 and k_2.

$$k_1 = k_2 = 0{,}00416 \quad (19)$$

Hence, the expected percentage of the lost work of mining is approximately 0.83%.

B. Testing

Carry out an experiment for obtained optimal miner work size. For comparison, take the values of experiments No. 1 and No. 2, and conduct another experiment for the default settings of miner. The results of experiment and its comparison with other experiments are shown in Table 2.

As can be seen from Table 2, the total number of calculated hashes in the case of mining with the optimal work size lesser than the number of calculated hashes in experiments with a larger work size. This is explained by the influence of the coefficient k_2, which increases with a decrease in the size of the work. However, since the total value of k_1 and k_2 for the experiment with the optimal work size is minimal, the number of actual calculated hashes for the experiment with the optimal work size is greater than for all other experiments. The percentage mining production increase can be estimated by comparing actual hash count for different experiments. For the obtained optimal value in comparison with the standard parameters of the mining production, increase is 0.924%.

Note that due to uneven new block time, count of non-actual work cycles for different tests varies. We should expect that in case of long-term mining count of non-actual work cycles would approach the mean frequency. To

estimate obtained performance growth in long-term time, we should equate new block count received by the miner during the experiment to the expected count (at the moment of testing and in terms of 10-minute testing time it is 42) and recalculate the number of actual and obsolete hashes for experimentally obtained count of mining cycles.

As we can see from Table 3, test with calculated optimal work size shows performance growth in comparison to other tests. In comparison to mining with default miner settings, performance growth is 0,886%. We can measure real work loss due to work switching in case of optimal mining work size. We should take a number of obsolete hash count and divide it by total calculated hash count. According to Table 3, mining work loss in case of long-term mining will be 0,413%. This value coincides with the theoretical value of k_1 coefficient from equation (19).

IV. Conclusion

In this paper, we have conducted a research of the Ethereum cryptocurrency mining for GPUs on the OpenCL platform. A search was made for problems that led to the loss of useful work when mining on the OpenCL platform. The following problems were identified: loss of work when receiving a new computational task; slowing down the mining process when the size of the mining work cycle is decreased. For each of the problems found, we derived the equation to estimate the percentage work loss of the miner because of the problem effect.

According to the problems found, a method was developed to find the optimal size of work for the OpenCL platform, which allows reducing the total percentage of lost work as much as possible. In accordance with the developed methodology, the results were tested on real equipment. The results of the test confirmed the calculated data and showed that use of the developed methodology allows achieving an increase in the productivity of the mining of the Ethereum cryptocurrency, in comparison with the standard settings of the miner.

It should be noted that due to regular DAG size change [22], the speed of mining for the same *global_work_size* value would regularly change. Since the parameter determining the optimal speed of the work of the miner is the cycle time calculated by equation (7), it is necessary to regularly change the value of *global_work_size* to maintain the average execution time of one cycle of calculations. In addition, it is necessary to take into account possible changes in the average time of finding a new block. In addition, this study does not take into account the delay in obtaining a new job, which can play a role in the resulting mining speed.

References

[1] M. Swan. Blockchain: Blueprint for a New Economy. O'Reilly, US, 2015.

[2] C. Decker and R. Wattenhofer, "Information Propagation in the Bitcoin Network," *2013 IEEE Thirteen. Int. Conf. Peer-to-Peer Comput.*, no. September, pp. 1–10, 2013.

[3] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks," *Proc. third Int. Symp. Inf. Process. Sens. networks - IPSN'04*, pp. 259–268, 2004.

[4] A Narayanan, J Bonneau, E Felten, A Miller, S.Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*, Princeton University Press, 2016

[5] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Futur. Gener. Comput. Syst.*, no. Xiaoqi Li, pp. 1–25, 2017.

[6] M. B. Taylor, "Bitcoin and The Age of Bespoke Silicon," *Proc. 2013 Int. Conf. Compilers, Architecture and Synthesis for Embedded Systems*, pp. 1–10, 2013.

[7] K. J. O'Dwyer and D. Malone, "Bitcoin mining and its energy footprint," *ISSC '14 Proc. 25th IET Irish Signals Syst. Conf.*, pp. 280-285, 2014.

[8] J. Barkatullah and T. Hanke, "Goldstrike 1: CoinTerra's first-generation cryptocurrency mining processor for bitcoin," *IEEE Micro*, vol. 35, no. 2, pp. 68–76, 2015.

[9] V. Buterin, "A next-generation smart contract and decentralized application platform," [Online] Available: https://github.com/ethereum/wiki/wiki/White-Paper

[10] T. Dryja, "Hashimoto: I/O bound proof of work," 2009.

[11] Ethash [Online] Available: https://github.com/ethereum/wiki/wiki/Ethash

[12] D. Kirk, "NVIDIA CUDA software and GPU parallel computing architecture," *Proceedings of the 6th international symposium on Memory management - ISMM '07*, pp. 103–104, 2007.

[13] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems," *Computing in Science & Engineering*, vol. 12, no. 3, pp. 66–73, 2010.

[14] Khronos Group, "An Introduction to OpenCL C++" [Online] Available: https://www.khronos.org/assets/uploads/developers/resources/Intro-to-OpenCL-C++-Whitepaper-May15.pdf

[15] P. Jääskeläinen, C. S. de La Lama, E. Schnetter, K. Raiskila, J. Takala, and H. Berg, "pocl: A Performance-Portable OpenCL Implementation," *International Journal of Parallel Programming*, vol. 43, no. 5, pp. 752–785, 2015.

[16] OpenCL Specification [Online] Available: https://www.khronos.org/registry/OpenCL/specs/opencl-1.2.pdf

[17] Ethereum miner with OpenCL, CUDA and stratum support [Online] Available: https://github.com/ethereum-mining/ethminer

[18] Claymore miner [Online] Available: https://github.com/nanopool/Claymore-Dual-Miner

[19] Patricia Tree [Online] Available: https://github.com/ethereum/wiki/wiki/Patricia-Tree

[20] OpenCL User Guide [Online] Available: http://developer.amd.com/wordpress/media/2013/12/AMD_OpenCL_Programming_User_Guide2.pdf

[21] V. Buterin, "Toward a 12-second Block Time." [Online] Available: https://blog.ethereum.org/2014/07/11/toward-a-12-second-block-time/

[22] Dagger Hashimoto [Online] Available: https://github.com/ethereum/wiki/wiki/Dagger-Hashimoto