Weekly Report - Diving into the Ethminer Source Code

MSc Project
Runchao Han

December 9, 2017

1 Progress of the Last Week

1. Imported `Ethminer` and `CCMiner` projects to IDEs (QtCrator, Codeblocks)
2. Learned about the Ethash implementation of `Ethminer`
3. Learned about `GNU Make`, `CMake` and `CUDA`

2 Building C/C++ Projects by make/cmake

(I have had no knowledge about the GCC toolchain before this, so I wrote this brief summary)

Most of blockchain projects are written by C++, especially the mining modules. To develop a big project, automatic building tools are applied throughout the development. `GCC`[1] toolchain is the most widely used tools for C/C++ projects, which includes:

- GNU make: an automation tool for compilation and build
- GNU Compiler Collection (GCC): a suite of compilers for several programming languages
- GNU Binutils: a suite of tools including linker, assembler and other tools
- GNU Bison: a parser generator, often used with the Flex lexical analyser
- GNU m4: an m4 macro processor
- GNU Debugger (GDB): a code debugging tool
- GNU build system (autotools): Autoconf, Automake and Libtool

This section mainly talks about building related projects.

2.1 Automatically Building Projects by make

`GNU Make`[2] is the automatic building toool for C/C++ projects. By writing user-defined rules in the `Makefile` the `make` command will compile the project automatically following the rules.

[1] https://gcc.gnu.org/
[2] https://www.gnu.org/software/make

2.2 Cross-platform Building Tool cmake

Besides `GNU Make`, a wide range of automatic building tools exist, like `Qmake` and `MS Nmake`. Moreover, cross-platform compiling is always an issue when building a project.

`CMake` aims at designing a platform-independent makefile `CMakeList.txt` which can generate specific makefiles in specific environments so that the building process can be customised. For example, with the `CMakeList.txt`, the `Makefile` of `GNU Make`, the `Microsoft Visual Studio` project file can be generated.

2.3 Process of Building Ethminer

`Ethminer` project is built upon `CMake`, so the process is:

1. `cd`: the project directory
2. `mkdir ./build`: generate a folder for the future generated `GNU Make` project
3. `./configure`: check dependencies
4. `cd buld && cmake .. -DETHASHCUDA=ON`: to generate a `GNU Make` project from the `CMakeList.txt` with CUDA (CUDA is not included by default)
5. `make`: compile the whole project

3 Importing cmake/make Project Into IDEs

IDEs like Codeblocks and QtCreator maintain their own project files instead of `Makefile` or `CMakeList.txt`. Therefore, issues of importing projects exist.

3.1 Importing CMake Project to IDEs

To import the `Cmake` project, we can:

- Use inherited `cmake -g` command to generate the specific project file for the corresponding IDE. For example, `cmake .. -G"CodeBlocks - Unix Makefiles"` generates the `Codeblocks` project file with the existing `CMakeList.txt`
- Some IDEs can import projects by `CMakeList.txt`, like `QtCreator`.

3.2 Importing Make Project to IDEs

As for `GNU Make` projects, no inherited project file generation methods exist. So if the chosen IDE does not support importing `Make` projects, we can only copy and paste.

Currently I'm not sure which IDE has better support for CMake/Make projects.

4 Ethminer Source Code Explained

`Ethminer`[3] is a fairly huge project supporting mining in a mining pool, which implements the Ethash PoW algorithm.

[3] https://github.com/ethereum-mining/ethminer

4.1 Ethminer Project Structure

The project consists of the main program with libraries. All subdirectories are listed below:

- ethminer: The main miner program
- libapicore: API for connecting to mining pools
- libdevcore: Classes related to codings (e.g. RLP coding) and producing logs
- libethash: Classes implementing common submodules for Ethash, like FNV Hash and SHA3, but the core algorithm is not in this directory
- libethash-cl/cuda: The core implementation of Ethash by OpenCl/CUDA
- libethcore: Classes from the Ethereum required by the mining process, like the main Miner class
- libstratum: Stratum is the future mining protocol for cryptocurrencies. Currently it is optional to install and not adopted by any mainstream blockchains.

As for profiling the mining algorithm, we mainly care about `ethminer`, `libethash`, `libethash-cuda` and `libethcore` subdirectories.

4.2 CUDA Mining Module Explained

A brief class/invocation diagram is drawn in Fig. 1 showing the invocation order of the Ethash mining process.

Currently it is confirmed that *compute_hash*() in Fig. 1 is the *hashimoto*() process in the Ethash Wiki[4], which is parallelised by CUDA.

CUDAMiner class is the main mining program which controls the whole mining process, like kickoff(), pause() and GPU configurations.

libethash/ethash_cuda_miner_kernel.h also contains the DAG generation and cache generation functionalities.

5 Miscellaneous

- I'm not sure which IDE I should use (QtCreator seems to have poor support for CMake projects.), which needs further research.

6 Next Week's Plan

1. Add timestamps to the Ethash implementation to make analysis
2. Import ccminer to IDE
3. Output stack graphs if I can

No reference this week because the work in this week is about setting up the programming environment and reading the source code.

[4]https://github.com/ethereum/wiki/wiki/Ethash

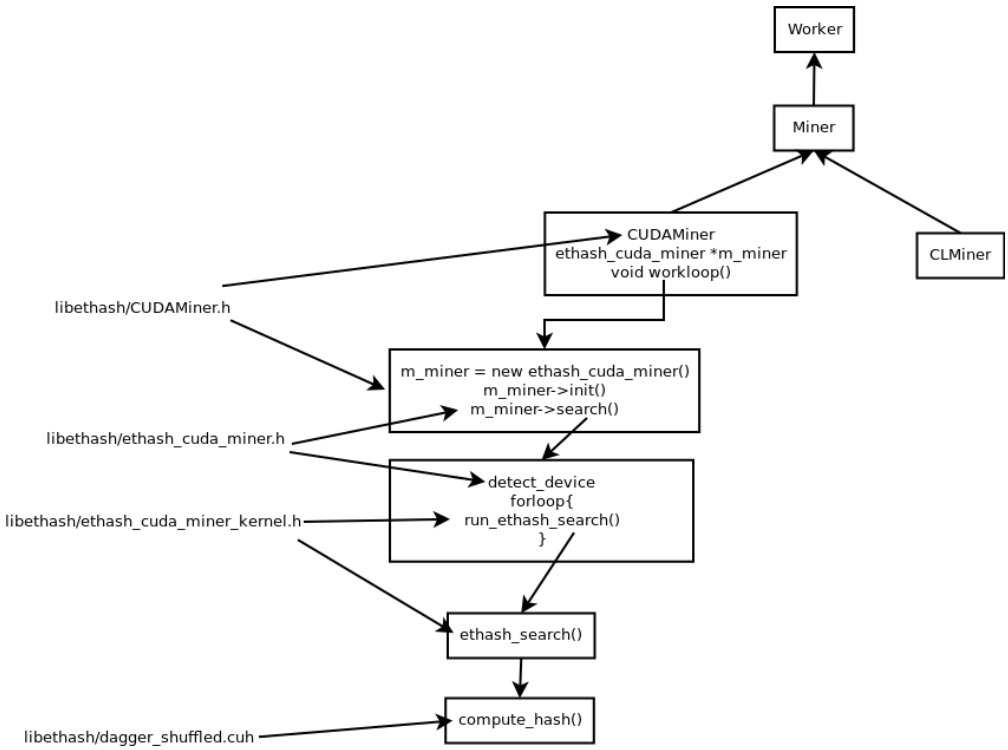

Figure 1: An informal class diagram of Ethminer.