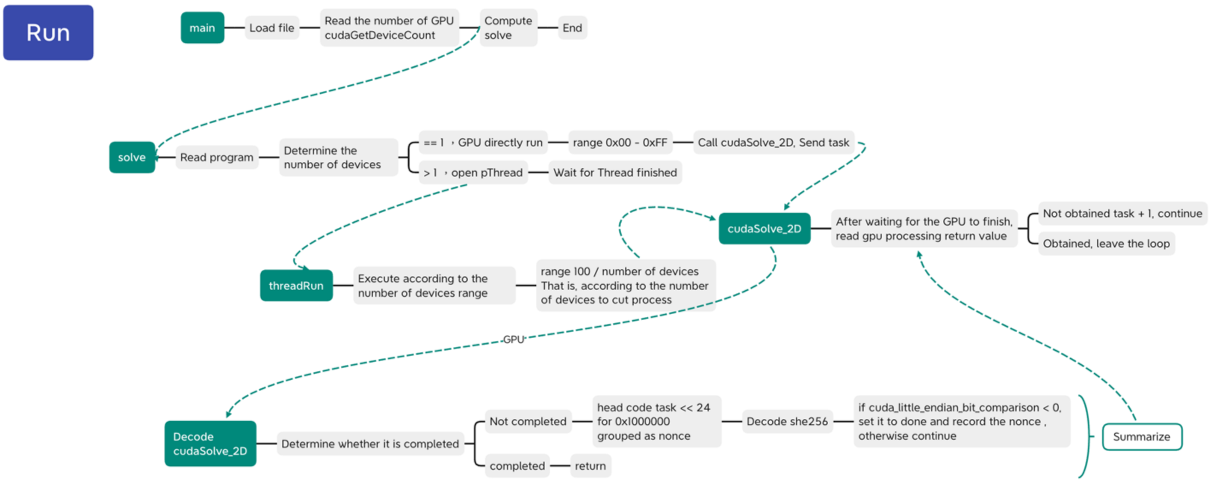
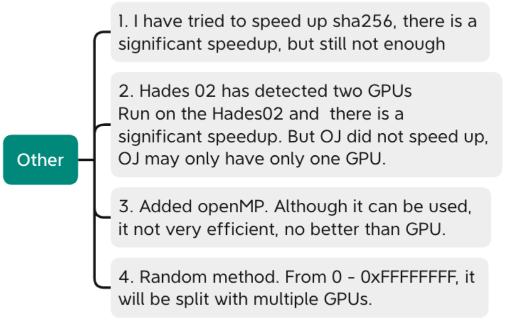
HW4: Bitcoin Miner

ISA 110065504 葉之晴

1. Your implementation.


Run
main
Load file
Read the number of GPU cudaGetDeviceCount
Compute solve
End
solve
Read program
Determine the number of devices
== 1 ， GPU directly run
range 0x00 - 0xFF
Call cudaSolve_2D, Send task
> 1 ， open pThread
Wait for Thread finished
cudaSolve_2D
After waiting for the GPU to finish, read gpu processing return value
Not obtained task + 1, continue
Obtained, leave the loop
threadRun
Execute according to the number of devices range
range 100 / number of devices That is, according to the number of devices to cut process
GPU
Decode cudaSolve_2D
Determine whether it is completed
Not completed
head code task << 24 for 0x1000000 grouped as nonce
Decode she256
if cuda_little_endian_bit_comparison < 0, set it to done and record the nonce , otherwise continue
completed
return
Summarize

2. The parallelization and optimization techniques you used in your solution.



1. I have tried to speed up sha256, there is a significant speedup, but still not enough

2. Hades 02 has detected two GPUs
Run on the Hades02 and there is a significant speedup. But OJ did not speed up, OJ may only have only one GPU.

3. Added openMP. Although it can be used, it not very efficient, no better than GPU.

4. Random method. From 0 - 0xFFFFFFFF, it will be split with multiple GPUs.

a. sha265

Some code has been changed. For example, I added “#pragma unroll 16” to speed up for loops in the program. There is a significant speedup, but still not enough.

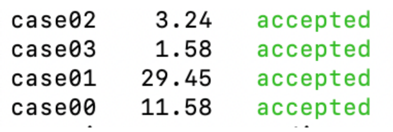
b. GPU

The original setting of the code was to use a lot of them, but it was later found that there was only one GPU on online judge. If the program run on the hader2, you can see the result of two running.

c. OpenMP

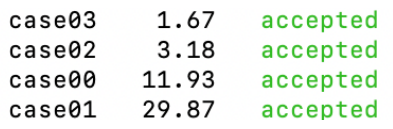
I used OpenMP in the hw4.cu. However, there is no significant improvement. In theory, it should reduce some time, but the actual test effect is not as expected.

- Without OpenMP



```
case02     3.24   accepted
case03     1.58   accepted
case01    29.45   accepted
case00    11.58   accepted
```

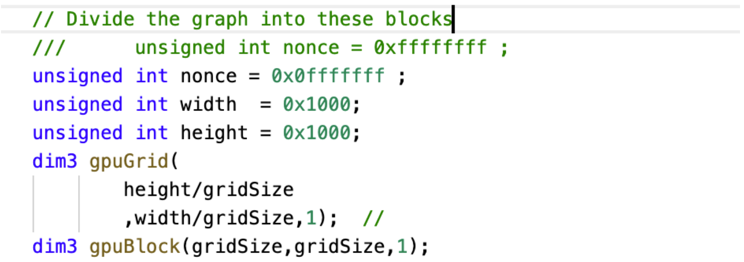
- With OpenMP



```
case03     1.67   accepted
case02     3.18   accepted
case00    11.93   accepted
case01    29.87   accepted
```

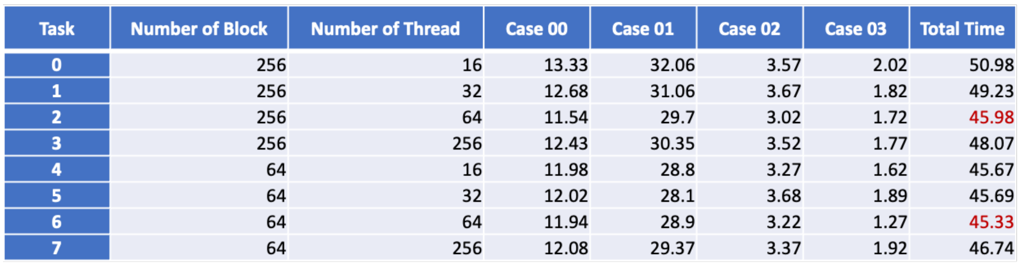
d. The random method

Currently, only from 0 - 0xFFFFFFFF, it will be split with multiple GPUs.

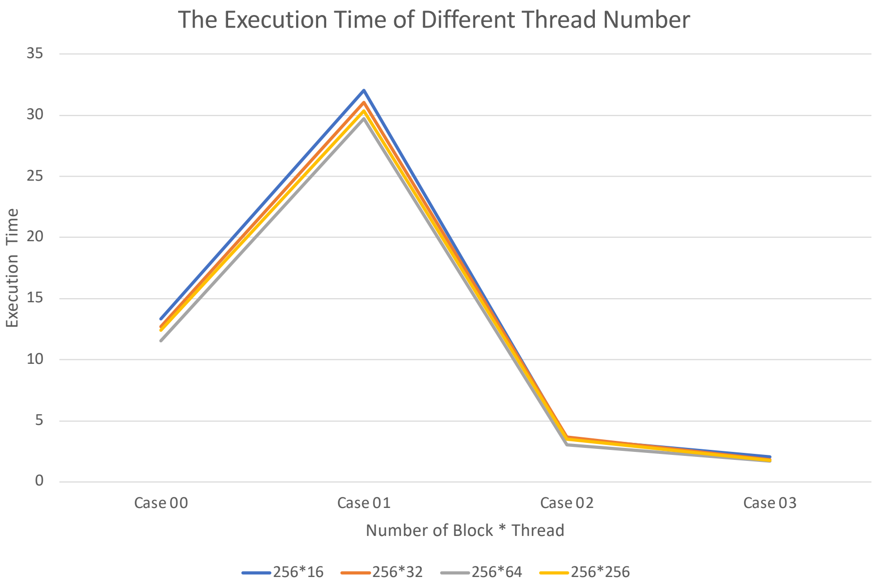


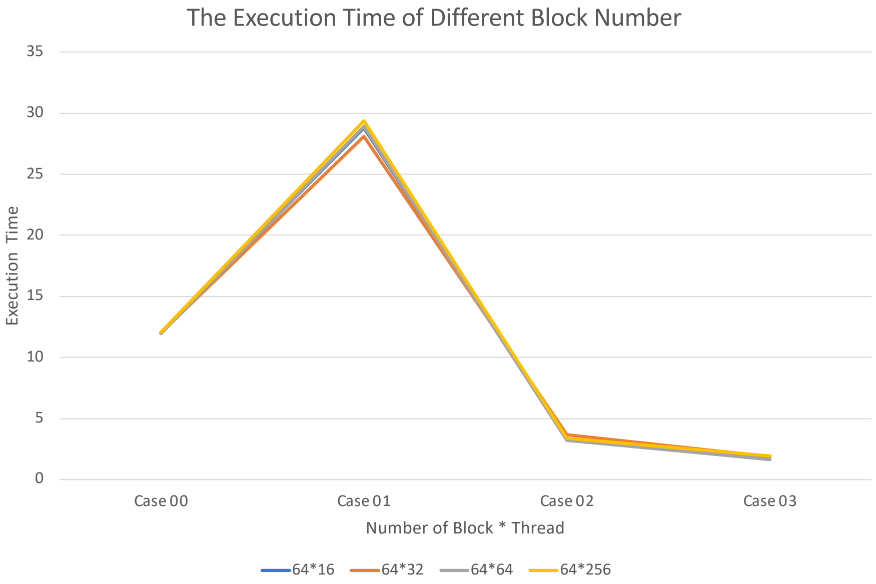
```
// Divide the graph into these blocks
///      unsigned int nonce = 0xffffffff ;
unsigned int nonce = 0x0fffffff ;
unsigned int width  = 0x1000;
unsigned int height = 0x1000;
dim3 gpuGrid(
        height/gridSize
        ,width/gridSize,1);  //
dim3 gpuBlock(gridSize,gridSize,1);
```

3. Experiments of various combinations of the number of blocks & threads (at least 8 combinations) and plot them with the figures.



Task	Number of Block	Number of Thread	Case 00	Case 01	Case 02	Case 03	Total Time
0	256	16	13.33	32.06	3.57	2.02	50.98
1	256	32	12.68	31.06	3.67	1.82	49.23
2	256	64	11.54	29.7	3.02	1.72	45.98
3	256	256	12.43	30.35	3.52	1.77	48.07
4	64	16	11.98	28.8	3.27	1.62	45.67
5	64	32	12.02	28.1	3.68	1.89	45.69
6	64	64	11.94	28.9	3.22	1.27	45.33
7	64	256	12.08	29.37	3.37	1.92	46.74





All in all, the result of the block number of 64 is better than 256 on the whole, and the thread number of 64 is the best.