

Manual

This manual is very long and detailed, and read it together with Kaichun's report.

To me, the most important part of Kaichun's report is **BASIC OBSERVATIONS AND SIMULATIONS**.

Some conceptions like **kernal, warp, SM = SIMT = CU**.

The overall process, **fetch, decode, issue, read operands, execute, writeback.**

Mainly dig into the part **issue & read operands**, which is mainly done by the **operand collector**

some explorations about nvprof

Though we will do the remaining job mainly with **CUPTI**, I found **nvprof** is easier and lighter. So, I first try to use **nvprof** to do the similar work as what CUPTI will do. There is no obvious reason, but I just want to try first, since CUPTI manual is very long.

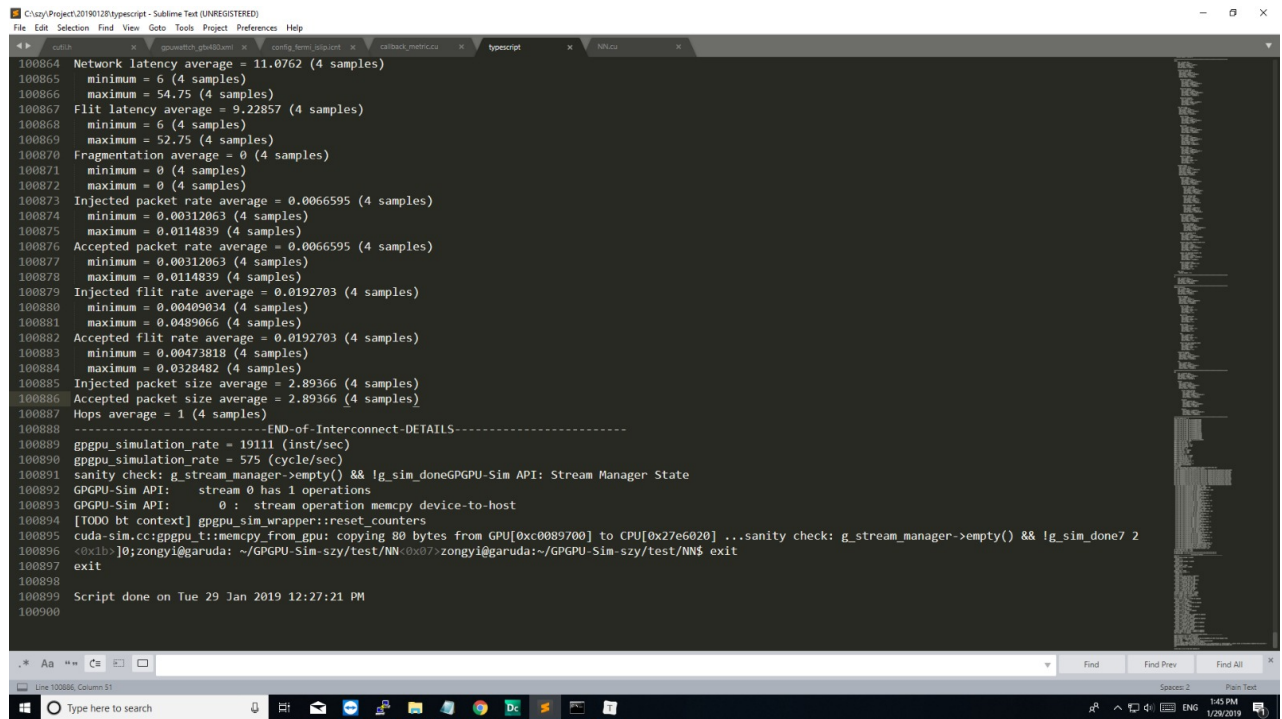
```

$ cd ~/gpgpu-sim_dev/half2/test/NN
$ ./cubxdump_complete_output_WHx4V
cubxdump_complete_output_WHx4V
cubxdump_complete_output_WHx4V
cubxdump_complete_output_rkTDGB
$ cat
download_regex.js
event_log.txt
floating_point_examples
gpusim_init_state.txt
gpgpusim.config
gpgpusim_metric_trace_report_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_metric_trace_report_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_metric_trace_report_Mon-Jan-28-22-00-10-2019.log.gz
gpgpusim_power_report_Mon-Jan-28-21-53-50-2019.log
gpgpusim_power_report_Mon-Jan-28-21-58-21-2019.log
gpgpusim_power_report_Mon-Jan-28-22-00-10-2019.log
gpgpusim_power_trace_report_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_power_trace_report_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_power_trace_report_Mon-Jan-28-22-00-10-2019.log.gz
gpgpusim_steady_state_tracking_report_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_steady_state_tracking_report_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_steady_state_tracking_report_Mon-Jan-28-22-00-10-2019.log.gz
gpgpusim_visualizer_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_visualizer_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_visualizer_Mon-Jan-28-22-00-10-2019.log.gz
$ cp watch_gtx480.xml
half2 state
half2 inexact
half2_operator_overload.cuh
$ cd ~/gpgpu-sim_dev/half2/test/NN nvprof .NN_float 2
===== Error: application not found.
$ cd ~/gpgpu-sim_dev/half2/test/NN nvprof .NN_float 2
$ cat no membar = 2051
Number of items = 10000
Number of rows = 28
Number of cols = 28
--129601- NVPROF is profiling process 129601, command: ./NN_float 2
NUM=2
7 2
--129601- Profiling application: ./NN_float 2
--129601- Profiling result:
Type Time(s) Time Calls Avg Min Max Name
GPU activities: 85.794s 4.421secs 6 236.34us 3.320ms 886.04us [CUDA memory alloc]
2.79s 41.343us 1 41.343us 41.343us 41.343us executeThirdLayer(float*, float*, float*)
0.57s 0.4160us 1 0.4160us 0.4160us 0.4160us executeSecondLayer(float*, float*, float*)
0.37s 5.5040us 1 5.5040us 5.5040us 5.5040us executeFourthLayer(float*, float*, float*)
0.27s 4.0000us 1 4.0000us 4.0000us 4.0000us executeFirstLayer(float*, float*, float*)
0.22s 2.2320us 1 2.2320us 2.2320us [CUDA memory alloc]
API calls: 96.30s 605.87ms 9 67.31ms 27.64us 605.49ms cudaMemcpy
399s 49.52ms 49.52ms 672.72us cudaMemcpy
0.21s 1.2634ms 180 6.7200us 170ms 278.03us cuDeviceGetAttribute
0.05s 323.91us 2 161.96us 155.86us 168.06us cuDeviceTotalMem
0.02s 134.03us 4 33.507us 16.844us 77.291us cudaLaunch
0.02s 100.36us 2 54.182us 51.275us 57.088us cuDeviceGetName
0.00s 11.069us 12 922ms 342ms 5.9730us cudaSetupArgument
0.00s 0.0890us 4 1.0220us 563ms 2.1800us cudaConfigureCall
0.00s 2.746ms 3 746ms 168ms 1.6800us cuDeviceGetCount
0.00s 1.3204us 4 331ms 193ms 573ms cuDeviceGet
--129601- Unified Memory profiling result:
Device "TITAN V (0)"
Count Avg Size Min Size Max Size Total Size Total Time Name
7 1.390752ms 0 1.390752ms 0 0 0 0
$ cd ~/gpgpu-sim_dev/half2/test/NN
$ ./cubxdump_complete_output_WHx4V
cubxdump_complete_output_WHx4V
cubxdump_complete_output_WHx4V
cubxdump_complete_output_rkTDGB
$ cat
download_regex.js
event_log.txt
floating_point_examples
gpusim_init_state.txt
gpgpusim.config
gpgpusim_metric_trace_report_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_metric_trace_report_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_metric_trace_report_Mon-Jan-28-22-00-10-2019.log.gz
gpgpusim_power_report_Mon-Jan-28-21-53-50-2019.log
gpgpusim_power_report_Mon-Jan-28-21-58-21-2019.log
gpgpusim_power_report_Mon-Jan-28-22-00-10-2019.log
gpgpusim_power_trace_report_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_power_trace_report_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_power_trace_report_Mon-Jan-28-22-00-10-2019.log.gz
gpgpusim_steady_state_tracking_report_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_steady_state_tracking_report_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_steady_state_tracking_report_Mon-Jan-28-22-00-10-2019.log.gz
gpgpusim_visualizer_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_visualizer_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_visualizer_Mon-Jan-28-22-00-10-2019.log.gz
$ cp watch_gtx480.xml
half2 state
half2 inexact
half2_operator_overload.cuh
$ cd ~/gpgpu-sim_dev/half2/test/NN nvprof .NN_float 2
===== Error: application not found.
$ cd ~/gpgpu-sim_dev/half2/test/NN nvprof .NN_float 2
$ cat no membar = 2051
Number of items = 10000
Number of rows = 28
Number of cols = 28
--129601- NVPROF is profiling process 129601, command: ./NN_float 2
NUM=2
7 2
--129601- Profiling application: ./NN_float 2
--129601- Profiling result:
Type Time(s) Time Calls Avg Min Max Name
GPU activities: 85.794s 4.421secs 6 236.34us 3.320ms 886.04us [CUDA memory alloc]
2.79s 41.343us 1 41.343us 41.343us 41.343us executeThirdLayer(float*, float*, float*)
0.57s 0.4160us 1 0.4160us 0.4160us 0.4160us executeSecondLayer(float*, float*, float*)
0.37s 5.5040us 1 5.5040us 5.5040us 5.5040us executeFourthLayer(float*, float*, float*)
0.27s 4.0000us 1 4.0000us 4.0000us 4.0000us executeFirstLayer(float*, float*, float*)
0.22s 2.2320us 1 2.2320us 2.2320us [CUDA memory alloc]
API calls: 96.30s 605.87ms 9 67.31ms 27.64us 605.49ms cudaMemcpy
399s 49.52ms 49.52ms 672.72us cudaMemcpy
0.21s 1.2634ms 180 6.7200us 170ms 278.03us cuDeviceGetAttribute
0.05s 323.91us 2 161.96us 155.86us 168.06us cuDeviceTotalMem
0.02s 134.03us 4 33.507us 16.844us 77.291us cudaLaunch
0.02s 100.36us 2 54.182us 51.275us 57.088us cuDeviceGetName
0.00s 11.069us 12 922ms 342ms 5.9730us cudaSetupArgument
0.00s 0.0890us 4 1.0220us 563ms 2.1800us cudaConfigureCall
0.00s 2.746ms 3 746ms 168ms 1.6800us cuDeviceGetCount
0.00s 1.3204us 4 331ms 193ms 573ms cuDeviceGet
--129601- Unified Memory profiling result:
Device "TITAN V (0)"
Count Avg Size Min Size Max Size Total Size Total Time Name
7 1.390752ms 0 1.390752ms 0 0 0 0
$ cd ~/gpgpu-sim_dev/half2/test/NN
$ ./cubxdump_complete_output_WHx4V
cubxdump_complete_output_WHx4V
cubxdump_complete_output_WHx4V
cubxdump_complete_output_rkTDGB
$ cat
download_regex.js
event_log.txt
floating_point_examples
gpusim_init_state.txt
gpgpusim.config
gpgpusim_metric_trace_report_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_metric_trace_report_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_metric_trace_report_Mon-Jan-28-22-00-10-2019.log.gz
gpgpusim_power_report_Mon-Jan-28-21-53-50-2019.log
gpgpusim_power_report_Mon-Jan-28-21-58-21-2019.log
gpgpusim_power_report_Mon-Jan-28-22-00-10-2019.log
gpgpusim_power_trace_report_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_power_trace_report_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_power_trace_report_Mon-Jan-28-22-00-10-2019.log.gz
gpgpusim_steady_state_tracking_report_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_steady_state_tracking_report_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_steady_state_tracking_report_Mon-Jan-28-22-00-10-2019.log.gz
gpgpusim_visualizer_Mon-Jan-28-21-53-50-2019.log.gz
gpgpusim_visualizer_Mon-Jan-28-21-58-21-2019.log.gz
gpgpusim_visualizer_Mon-Jan-28-22-00-10-2019.log.gz
$ cp watch_gtx480.xml
half2 state
half2 inexact
half2_operator_overload.cuh
$ cd ~/gpgpu-sim_dev/half2/test/NN nvprof .NN_float 2
===== Error: application not found.
$ cd ~/gpgpu-sim_dev/half2/test/NN nvprof .NN_float 2
$ cat no membar = 2051
Number of items = 10000
Number of rows = 28
Number of cols = 28
--129601- NVPROF is profiling process 129601, command: ./NN_float 2
NUM=2
7 2
--129601- Profiling application: ./NN_float 2
--129601- Profiling result:
Type Time(s) Time Calls Avg Min Max Name
GPU activities: 85.794s 4.421secs 6 236.34us 3.320ms 886.04us [CUDA memory alloc]
2.79s 41.343us 1 41.343us 41.343us 41.343us executeThirdLayer(float*, float*, float*)
0.57s 0.4160us 1 0.4160us 0.4160us 0.4160us executeSecondLayer(float*, float*, float*)
0.37s 5.5040us 1 5.5040us 5.5040us 5.5040us executeFourthLayer(float*, float*, float*)
0.27s 4.0000us 1 4.0000us 4.0000us 4.0000us executeFirstLayer(float*, float*, float*)
0.22s 2.2320us 1 2.2320us 2.2320us [CUDA memory alloc]
API calls: 96.30s 605.87ms 9 67.31ms 27.64us 605.49ms cudaMemcpy
399s 49.52ms 49.52ms 672.72us cudaMemcpy
0.21s 1.2634ms 180 6.72
```

This is to use it in the command line. Next, I want to write a script and save all the output in the terminal as a file, so I do the following things:

- write a script to run the application using **system("xxxx")**
- use **script** to save the output to a file

This method works. All the output is saved as a file, and we can refer to it conveniently.



```
100864 Network latency average = 11.0762 (4 samples)
100865   minimum = 6 (4 samples)
100866   maximum = 54.75 (4 samples)
100867 Flit latency average = 9.22857 (4 samples)
100868   minimum = 6 (4 samples)
100869   maximum = 52.75 (4 samples)
100870 Fragmentation average = 0 (4 samples)
100871   minimum = 0 (4 samples)
100872   maximum = 0 (4 samples)
100873 Injected packet rate average = 0.0066595 (4 samples)
100874   minimum = 0.00312063 (4 samples)
100875   maximum = 0.0114839 (4 samples)
100876 Accepted packet rate average = 0.0066595 (4 samples)
100877   minimum = 0.00312063 (4 samples)
100878   maximum = 0.0114839 (4 samples)
100879 Injected flit rate average = 0.0192703 (4 samples)
100880   minimum = 0.00409034 (4 samples)
100881   maximum = 0.0489066 (4 samples)
100882 Accepted flit rate average = 0.0192703 (4 samples)
100883   minimum = 0.00473018 (4 samples)
100884   maximum = 0.0328482 (4 samples)
100885 Injected packet size average = 2.89366 (4 samples)
100886 Accepted packet size average = 2.89366 (4 samples)
100887 Hops average = 1 (4 samples)
100888 -----END-of-Interconnect-DETAILS-----
100889 gpgpu_simulation_rate = 19111 (inst/sec)
100890 gpgpu_simulation_rate = 575 (cycle/sec)
100891 sanity check: g_stream_manager->empty() && !g_sim_doneGPGPU-Sim API: Stream Manager State
100892 GPGPU-Sim API:   stream 0 has 1 operations
100893 GPGPU-Sim API:   0 : stream operation memcpy device-to-host
100894 [TODO bt context] gpgpu_sim_wrapper::reset_counters
100895 cuda-sim.cc:gpgpu_t::memcpy_from_gpu: copying 80 bytes from GPU[0xc0089700] to CPU[0x27e6020] ...sanity check: g_stream_manager->empty() && !g_sim_done? 2
100896 <0x1b>]0;zongyi@garuda: ~/GPGPU-Sim-szy/test/NN<0x07>zongyi@garuda:~/GPGPU-Sim-szy/test/NN$ exit
100897 exit
100898
100899 Script done on Tue 29 Jan 2019 12:27:21 PM
100900
```

Maybe in the future when we need to use CUPTI to output and debug, we can use the same method.

CUPTI

It is a little difficult for me now, these are what I did:

- **read the manual (a part)**

To figure out some basic knowledge about **CUPTI**, and know its four types of API(**Activity, Callback, Event, Metric**). The most important API for us I think is the **Activity API**

I have not got my hands dirty enough, and its results and programming methods seem very strange to me right now.:(

But at least , can dig into it, for instance, the output of CUPTI is like the following:

```
Event Trace
_z6kernelIPfEvt_i: (active_warps,1734) (gst_inst_32bit,100)
(active_cycles,423)
_z7kernel2IPfEvt_i: (active_warps,865) (gst_inst_32bit,50)
(active_cycles,418)

Metric Trace
_z6kernelIPfEvt_i: (flop_count_dp,0) (flop_count_sp,100) (inst_executed,52)
_z7kernel2IPfEvt_i: (flop_count_dp,0) (flop_count_sp,50) (inst_executed,26)
```

It is very strange to me at present.... Need to get more familiar with it.

Questions

What is our final goal?

My past understanding is that our project's final goal is to **speed up GPGPU-Sim**. The detailed methods we discussed last week seem reasonable.

But, is GPGPU-Sim the necessary thing? I suppose it is, because we want to speed up it. Without GPGPU-Sim, our final goal is not meaningful.

Time

And in this week, I found that it is a little hard for me to get the progress quickly.

Meanwhile, I found it takes much time to read the manual throughout, if we want to finish our goal, can I just read the necessary parts?

eg, I do not have to get clear about how **Operand Collector** works, I do not have to know much about **Interconnection port**...whether it is FIFO or not will not stand in my way right now.

Strategy

There are two main methods of improving myself:

- **theory**, eg. reading papers , manuals, reports or others
- **practice**, eg. programming without digging into GPGPU-Sim mechanism, just the output. If we have to make entrypoints in the future, read then.

Which is more suitable?

Plan

- Manual?
- CUPTI to trace the application
- transfer information to GPUWattch