CUDA Parallel Programming
Homework 5

40647007S 朱健愷

April 22, 2021

1 Source codes

1.1 File Layout

- histogram_1GPU/hist_1gpu_gmem.cu - Main code computed in CPU and GPU using global memory, it is used to calculate histogram of pseudo random number generated exponential distribution random number.
- histogram_1GPU/hist_1gpu_shmem.cu - Main code computed in CPU and GPU using shared memory, it is used to calculate histogram of pseudo random number generated exponential distribution random number.
- Makefile - Script to auto generate executable from code.
- histogram_1GPU/experiment.sh - Script to auto generate results of CPU, GPU using global memory, GPU using shared memory statistic result using different block size.
- histogram_1GPU/result/block_CPU/Output - Output CPU compute result of calculate histogram of pseudo random number generated exponential distribution random number.
- histogram_1GPU/result/gm_block_*/Output_* - Output GPU with global memory compute result of calculate histogram of pseudo random num-

ber generated exponential distribution random number, the suffix represented the block size.

- histogram_1GPU/result/shm_block_*/Output_* - Output GPU with shared memory compute result of calculate histogram of pseudo random number generated exponential distribution random number, the suffix represented the block size.
- notebook/*.png - Plots concluding output result

1.2 Usage

Make code in both histogram_1GPU/ directory Run the experiment.sh script in histogram_1GPU/ directory

```
cd histogram_NGPU
make
sh experiment.sh
```

And it will produce generating exponential distribution random number statistical result using different block size.

2 Result

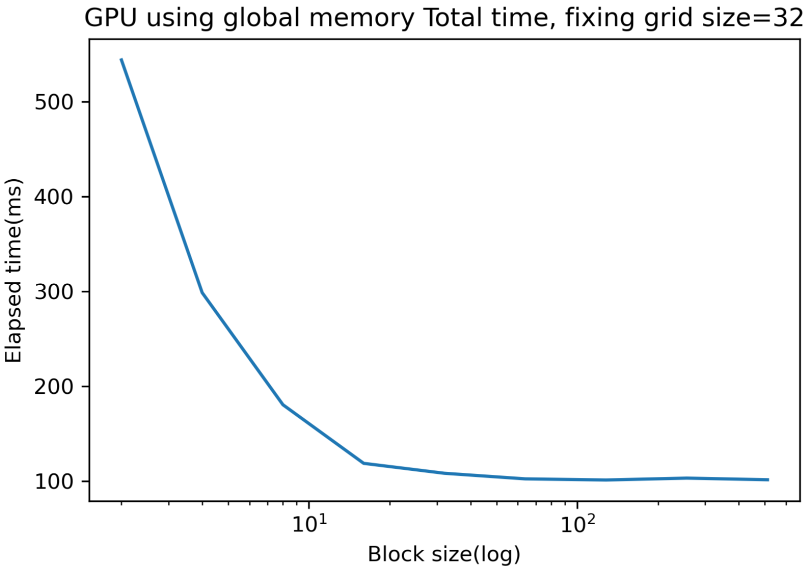
2.1 Experiment environment

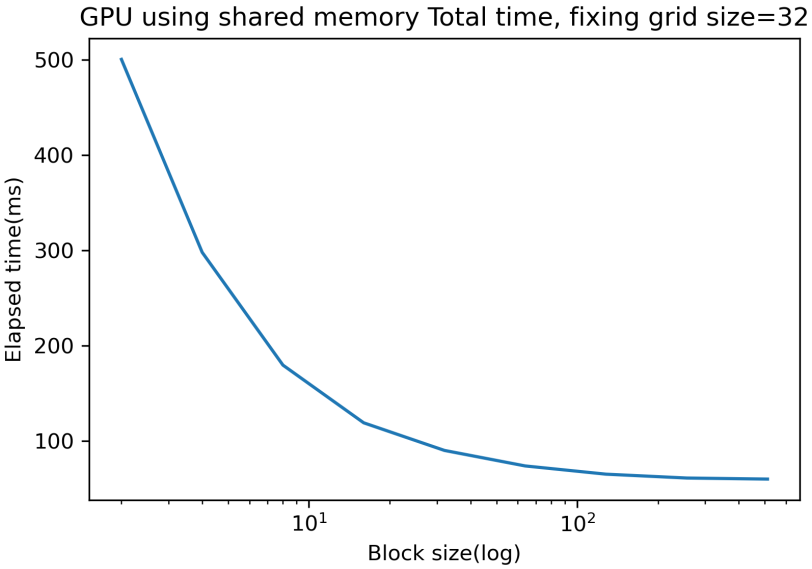
I ran my code on workstation provided in course, below is the Setup of workstation

- Operating system: Linux version 4.19.172 (root@twcp1)
  (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1))
- CPU: Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz
- GPU: Nvidia GTX 1060 6GB
- Memory: 32GB

2.2 Performance

Below two figures showed GPU using global memory case and GPU using shared memory case generating random sample in exponential distribution fixing grid size and change block size.





The CPU total time of generating random sample in exponential distribution is 156.274750 (ms).

And below one figure showed the comparison between the generated random sample in exponential distribution and real exponential distribution.

2.2.1 Observation

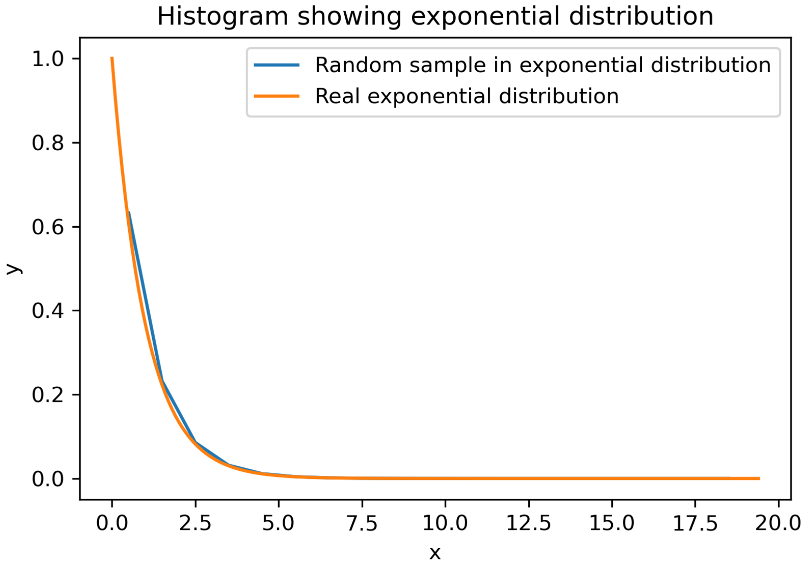
We can observe that the performance of small block size setup in both GPU using global memory case and GPU using shared memory case yield the worse performance in my experiment. This may because of we don't have enough threads in grid to collect histogram datas which range are out of basic per threads in grid(namely, more loop is needed in while loop in device code). So the block size do affect the performance a lot.

The result of utilizing shared memory helps improve the performance in nearly every block size setup.

The optimal block size setup of generating random sample in exponential distribution in my experiment is 128 block size for GPU using global memory case, 512 block size for GPU using shared memory case.

3 Discussion

The speed up of GPU using shared memory version compared to GPU using global memory version may due to the shared memory version first collect histogram data in shared memory, and then collect them to global memory, which largely decrease the time to wait the other threads to perform the atomic operations (namely write to same histogram bins).


Histogram showing exponential distribution
Random sample in exponential distribution
Real exponential distribution
y
x