SE346 LAB Report

GPU Architecture

姓名： 姚子航

班级： F1603702

学号： 516030910293

日期： 2019/06/11

Lab I: Benchmarking

1.1&1.2 Download Rodinia and build BFS & Simulate BFS with GPGPU-sim

根据文档中教程下载 Rodinia 并使用 GPGPU-sim 模拟 BFS

1.3 What's your L1 data cache miss rate?

运行 BFS Benchmark 后，我将标准输出重定向到 Lab1-result.txt 中，在 Benchmark 运行中会多次测量数据，所以 log 中含有多个 miss rare，使用 grep 命令筛选出 L1D_total_cache_miss_rate：

L1D_total_cache_miss_rate = 0.3234
L1D_total_cache_miss_rate = 0.3388
L1D_total_cache_miss_rate = 0.3806
L1D_total_cache_miss_rate = 0.4407
L1D_total_cache_miss_rate = 0.4034
L1D_total_cache_miss_rate = 0.4529
L1D_total_cache_miss_rate = 0.4064
L1D_total_cache_miss_rate = 0.4202
L1D_total_cache_miss_rate = 0.4535
L1D_total_cache_miss_rate = 0.4578
L1D_total_cache_miss_rate = 0.3944
L1D_total_cache_miss_rate = 0.3973
L1D_total_cache_miss_rate = 0.3927
L1D_total_cache_miss_rate = 0.3924
L1D_total_cache_miss_rate = 0.3922
L1D_total_cache_miss_rate = 0.3919

可以得到 L1 data cache miss rate 为 0.3919

1.4 What's your average memory fetch latency?

与平均访存延时相关的项为 averagemflatency。按照之前的做法得到以下输出：

averagemflatency = 259
averagemflatency = 220
averagemflatency = 196
averagemflatency = 184
averagemflatency = 172
averagemflatency = 170
averagemflatency = 177
averagemflatency = 177
averagemflatency = 233
averagemflatency = 232
averagemflatency = 229
averagemflatency = 229

averagemflatency = 227
averagemflatency = 227
averagemflatency = 227
averagemflatency = 227

可以得到平均的访存延迟为 227

1.5 Name several characteristic of BFS based on your results, such as cache, memory and instructions.

为了探索 BFS workload 的特性，我将 BFS workload 与其他 workload (**gaussian、myocyte、lud)**的运行结果进行比较，得到下表，在运行其他 workload 的时候我遇到了一些环境问题:

1.在 make 的时候报错-lcudart 未找到，经过检查我发现是因为 makefile 中设置了链接库路径为/usr/local/cuda/lib,而该路径并未存在，为了解决这个问题我执行如下命令，问题解决。

sudo cp -r $CUDA_HOME/lib64/* /*usr*/local/cuda/lib/

2.在 make 的时候报错 nvcc 未找到，经过检查我发现是 makefile 写的不对，将 nvcc 路径改成 CUDA_DIR 下的 bin 路径，问题解决。

运行结果如下：

	L1D total cache miss rate	**L1I total data miss rate**	**Average memory fetch latency**	**simulation rate (cycle/sec)**
BFS	0.3919	0.0136	227	2132
gaussian	0.4545	0.1732	162	4155
myocyte	0.4915	0.1060	141	7015
lud	0.6162	0.0264	335	3947

从表中可以看出，BFS 的 data miss rate 较低，说明数据局部性比较好，instruction miss rate 相比其他 workload 也非常低，说明其指令重复执行较多，局部性较好；但是 BFS 的访存时延相比其他 workload 较高，只比 lud 要低；但其模拟速率和 **gaussian、myocyte、lud** 比都比较小，说明其代码执行比较耗时。

Lab II: Schedule policy

2.3 What is the runtime, simulation rate (cycle/sec) for each configuration?

依次更改调度策略，运行 pathfinder，分析结果得到下表：

Algorithm	Simulation rate (cycle/sec)	Simulation rate (inst/sec)
LRR	3343	335134
TL	4201	418918
GTO	4102	418918

Algorithm	Runtime (sec)
LRR	20
TL	16
GTO	16

可以看到，TL 和 GTO 的运行时间与模拟频率比较相近，而 LRR 的模拟频率较低，也造成了 LRR 的运行时间较长，这说明了 LRR 的调度策略较为复杂模拟复杂性较大因而模拟频率较低。

2.4 L1 cache miss rate & memory fetch latency

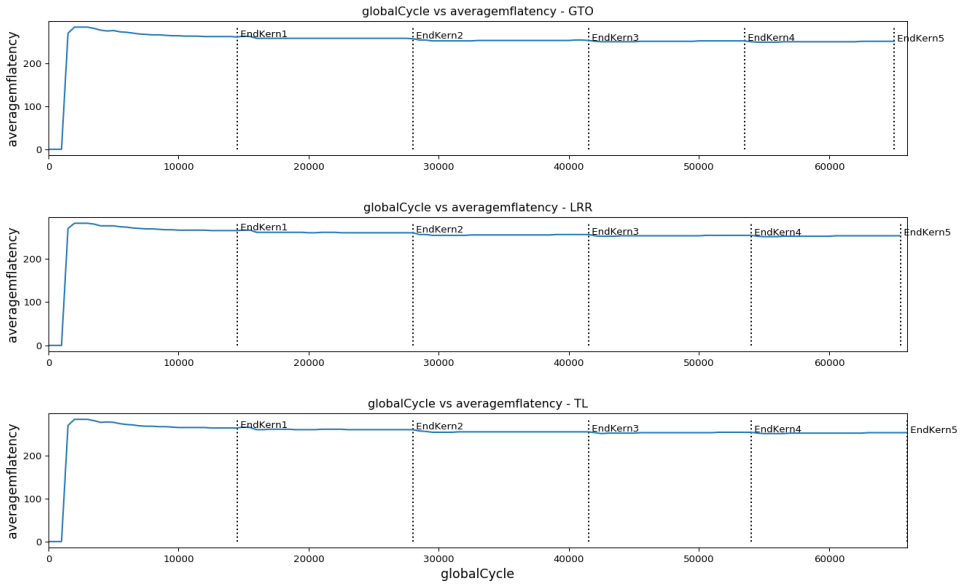
Algorithm	L1 data cache miss rate	Average memory fetch latency
LRR	0.5680	253
TL	0.5680	252
GTO	0.5680	251

可以看出，三种算法的 cache miss rate 和访存 latency 都非常接近，这说明调度算法的选择对 pathfinder 这个 workload 作用不明显，可能是其他因素制约这 pathfinder 的运行。

Lab III: Exploration with AerialVersion

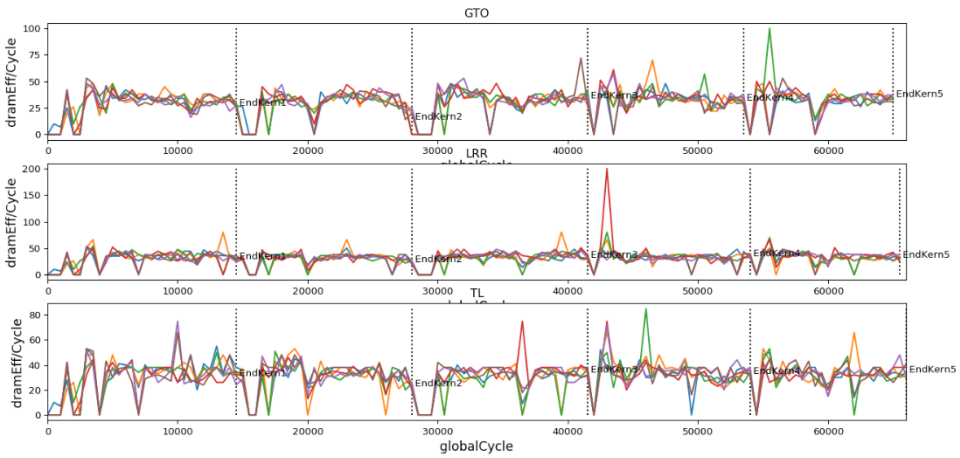
通过设置 gpusim.config 中的 visualizer_enabled=1 开启可视化信息收集，运行 workload 会生成可视化数据文件，使用 AerialVersion 对 Lab II 所得的数据进行可视化，在过程中我也碰到了一些依赖模块未找到的问题，为了解决环境问题我使用了 conda 来提供一个纯净的 python2.7 环境。结果如下：

3.1 平均访存 latency



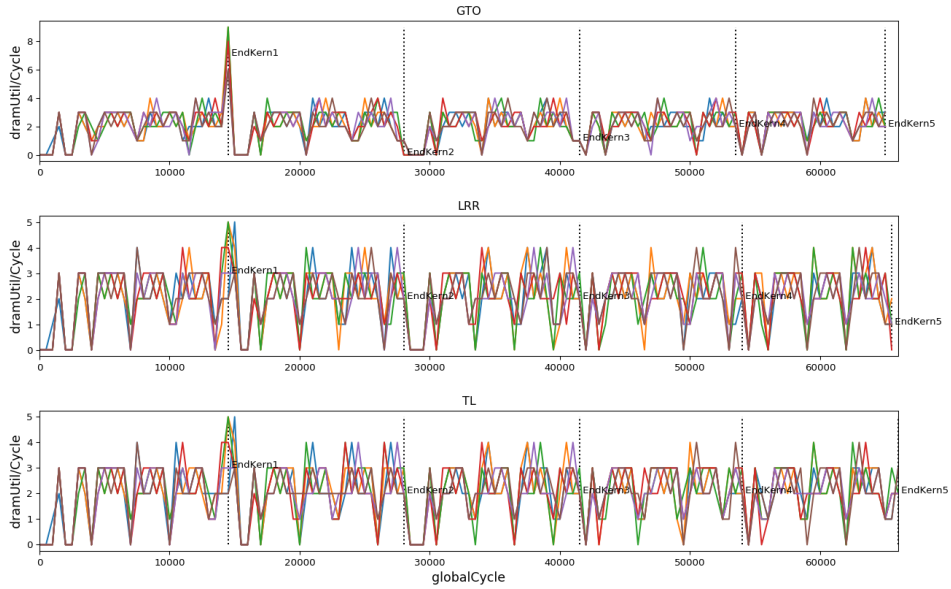
正如 Lab2 数据所体现的，三中调度策略的访存 latency 区别不大，且变化较小总体快速趋于平稳。

3.2 DRAM 通道的使用效率(dramEff)



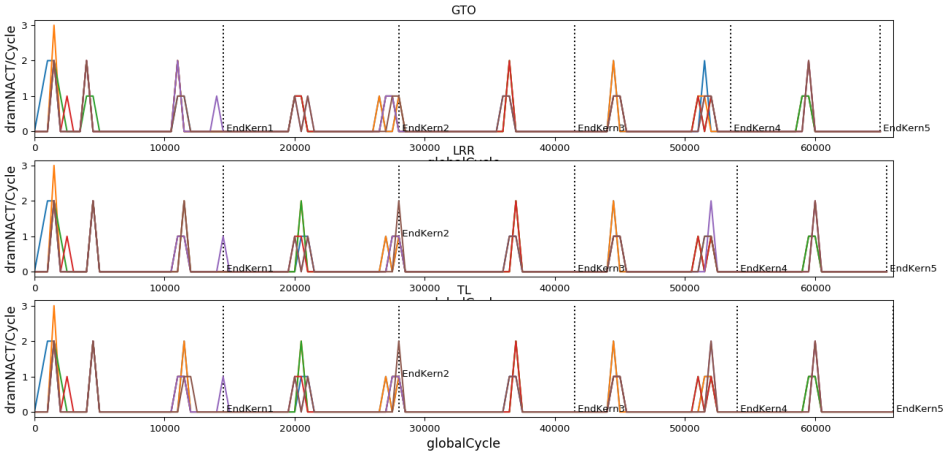
LRR 算法的通道利用率整体较高，峰值也比其他两个要高，说明 LRR 在有 pending request 的时候对于 dram channel 的利用率较高。

3.3 DRAM 通道利用率(dramUtil)



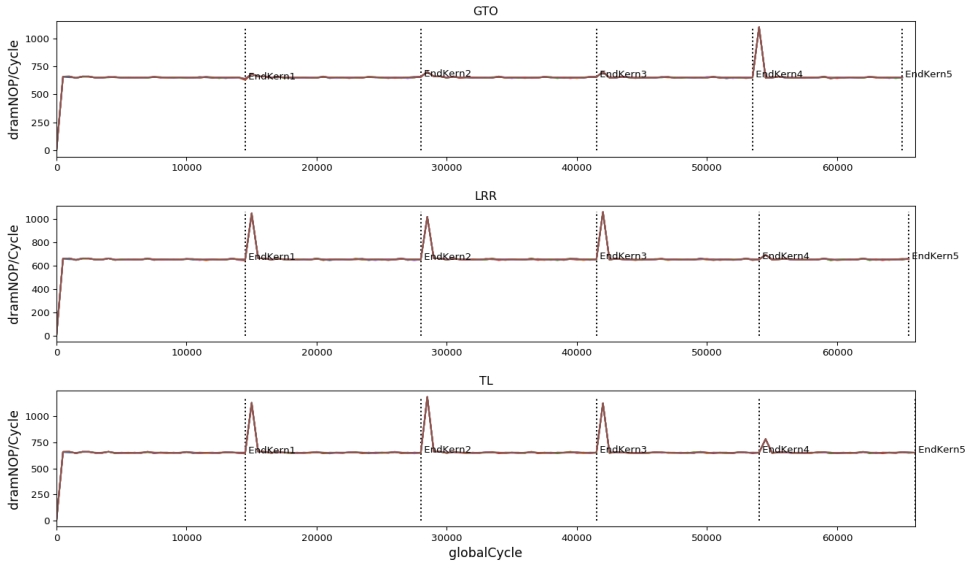
从图中可以看到，整体区别不大，GTO 的利用率峰值较高

3.4 激活指令数目(dramNACT)



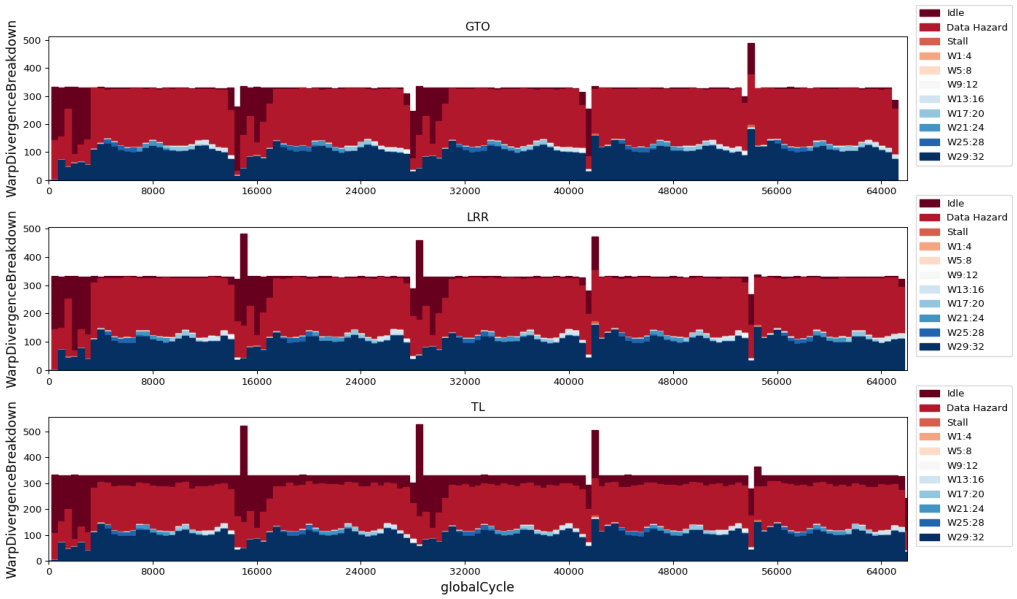
三者发送激活指令(Active Command)的数目与频率都比较一致。

3.5 发送 NOP 指令数目



从图中可以看出，LRR 和 TL 发送 NOP 指令的高峰在前三个 kernel 结束时，而 GTO 与其他两个不同，NOP 高峰在第四个 kernel 结束时。

3.6 warp divergence 分析(WarpDivergenceBreakdown)



从图中可以看出，有 3/5 的 warp 处于 data hazard/Stall 状态，而剩余的 warp 基本处于全部运行状态，说明 pathfinder 这个 workload 中 warp 的 divergence 概率低，未能充分利用 diverse 运行特性。