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

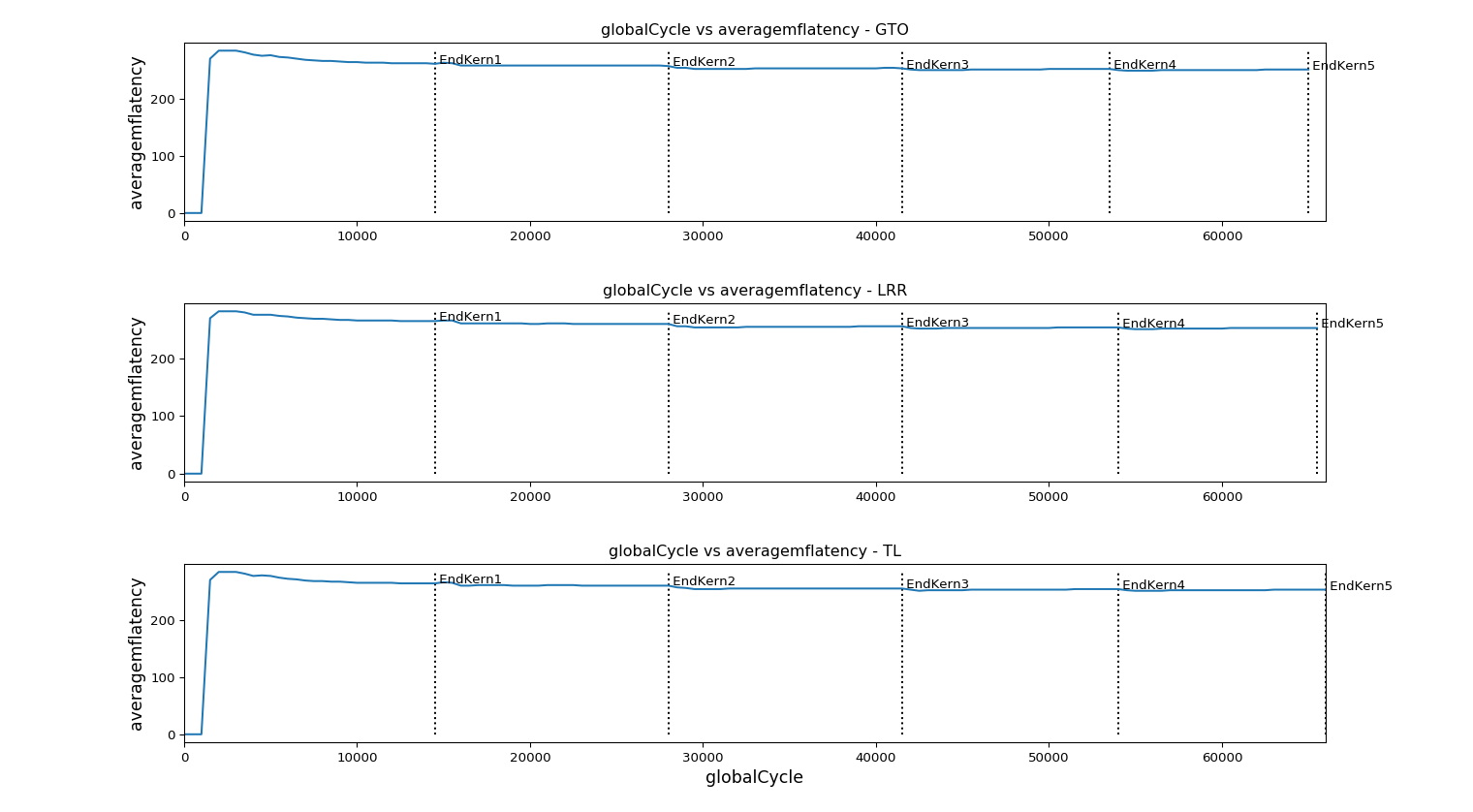
|  |  |  |
| --- | --- | --- |
| **Algorithm** | **L1 data cache**  **miss rate** | **Average memory**  **fetch latency** |
| **LRR** | 0.5680 | 253 |
| **TL** | 0.5680 | 252 |
| **GTO** | 0.5680 | 251 |

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**

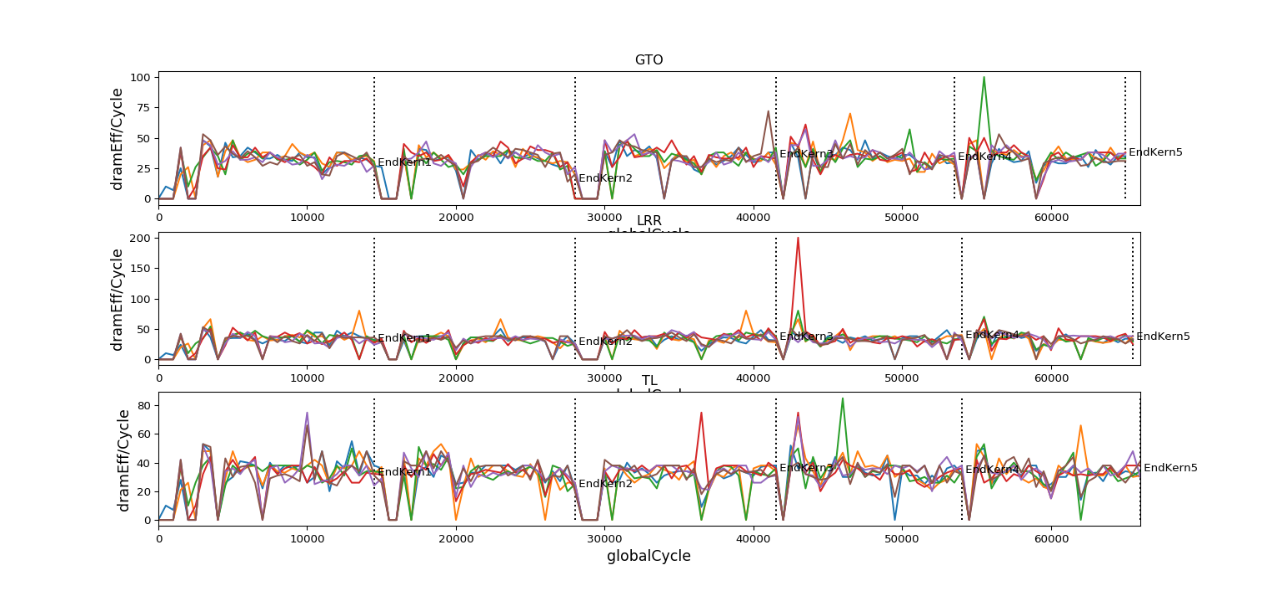
通过设置gpusim.config中的visualizer\_enabled=1开启可视化信息收集，运行workload会生成可视化数据文件，使用AerialVersion对LabⅡ所得的数据进行可视化，在过程中我也碰到了一些依赖模块未找到的问题，为了解决环境问题我使用了conda来提供一个纯净的python2.7环境。结果如下：

## 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)

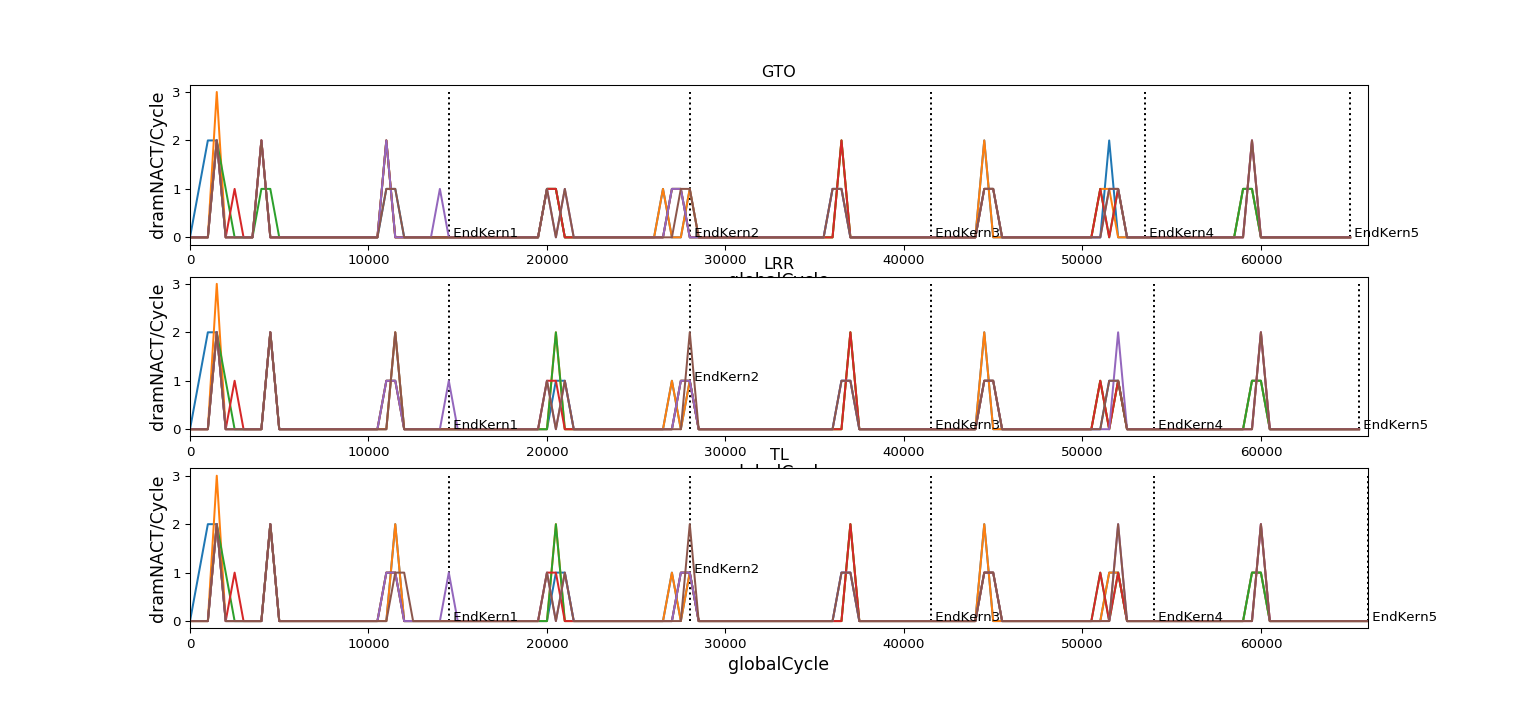


LRR算法的通道利用率整体较高，峰值也比其他两个要高，说明LRR在有pending request的时候对于dram channel的利用率较高。

## 3.3 DRAM通道利用率(dramUtil)

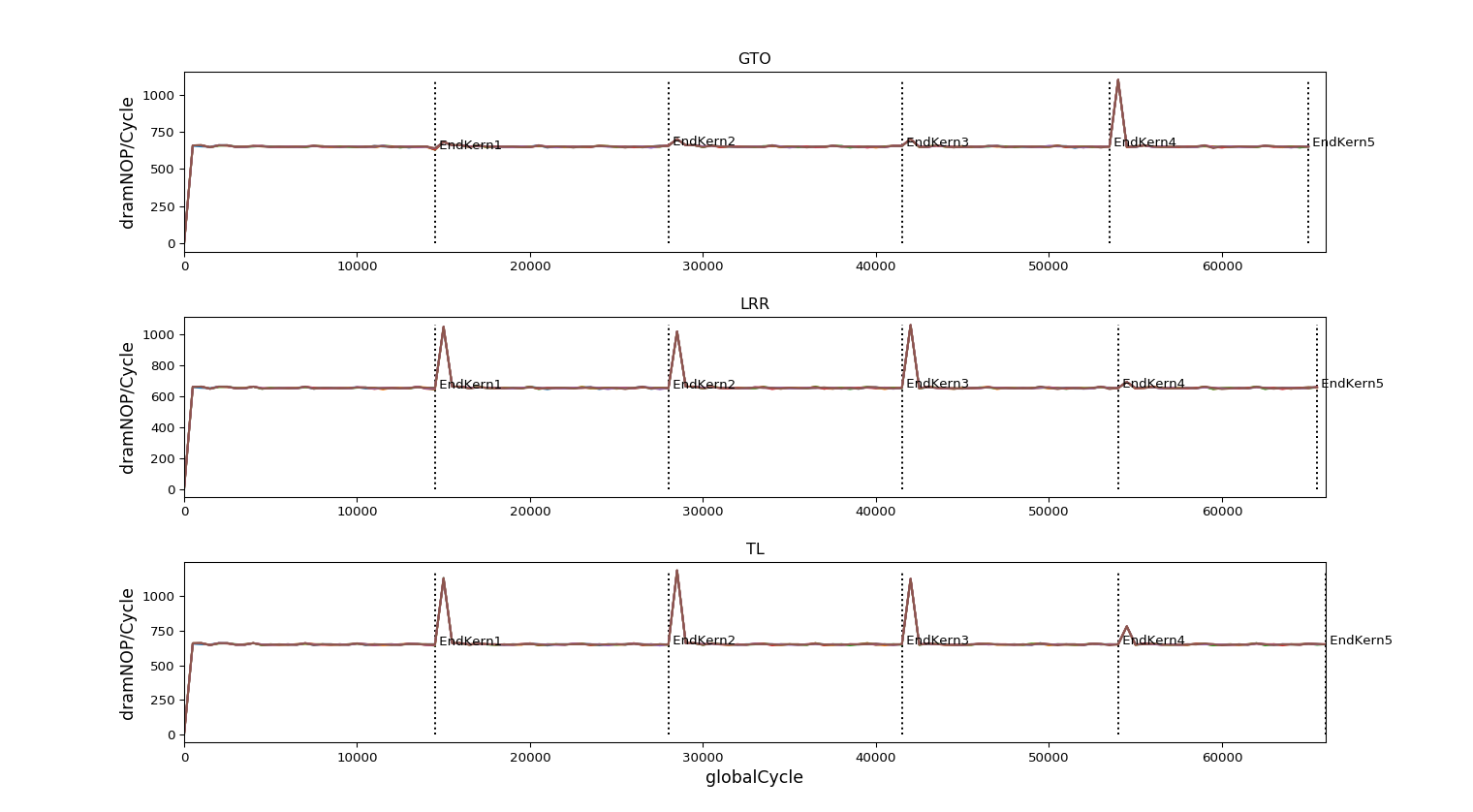
## 从图中可以看到，整体区别不大，GTO的利用率峰值较高

## 3.4 激活指令数目(dramNACT)

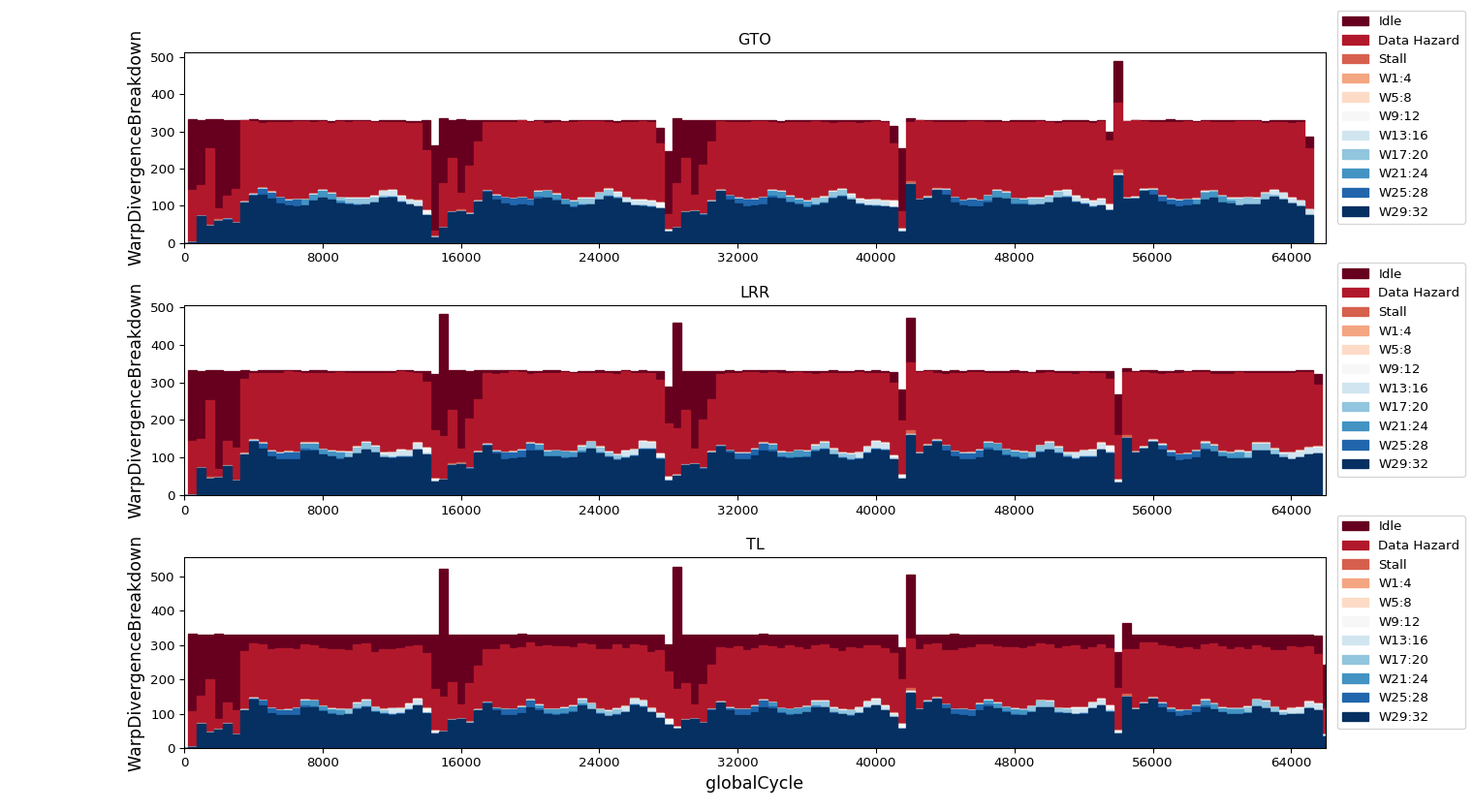


三者发送激活指令(Active Command)的数目与频率都比较一致。

## 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。