Prefetch based acceleration in GPGPU

M.Tech Project Stage 1 Report

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

by
Gazula Srinivas Bapaiah Naidu
(Roll No. 203079007)

Under the guidance of
Prof. Virendra Singh

Department of Electrical Engineering
Indian Institute of Technology Bombay
October 2022

Acknowledgement

I express my gratitude to my guide Prof. Virendra Singh for providing me the opportunity to work on this topic.

Srinivas Gazula
Electrical Engineering
IIT Bombay

Abstract

The texture cache in GPUs were used to put images onto triangles. In modern GPGPUs texture cache is under utilized. On the other hand when tens of warps are running concurrently, GPUs suffer from cache contention due to limited cache size. Prior techniques proposed to use spare registers as victim cache along with warp throttling. Whenever GPU encounters applications that have irregular control flow and are memory intensive with many data-dependent accesses, the execution efficiency will be low to memory access latency. Prior techniques proposed to use spare registers to prefetch data for such data-dependent load instructions. In this report, a proposal is put forward to utilize the texture-cache either as victim cache or to store the prefetched data. Thus the spare registers can be freed and maybe used by other functional units.

Contents

List of Figures

Chapter 1

Introduction

More applications which are parallel and irregular are being GPU enabled. GPU is a better solution for handling such parallel and data intensive applications, if we consider energy and power efficiency. Whenever GPU encounters applications that have irregular control flow and are memory intensive with many data-dependent accesses, the execution efficiency will be low to memory access latency. As shown in figure-1.1, there are a significant number of unused registers available. Prior studies[1] have shown how to utilize these registers to prefetch for data-dependent load instructions. GPUs suffer significant slowdowns while executing memory-intensive workloads. Due to lack of cache space there will be a significant drop in the performance in such memory intensive workloads. Prior studies[2] have used warp-throttling to limit the number of active warps and also used both static and dynamic spare registers to store the evicted cache lines. i.e., The spare registers were used as victim cache.

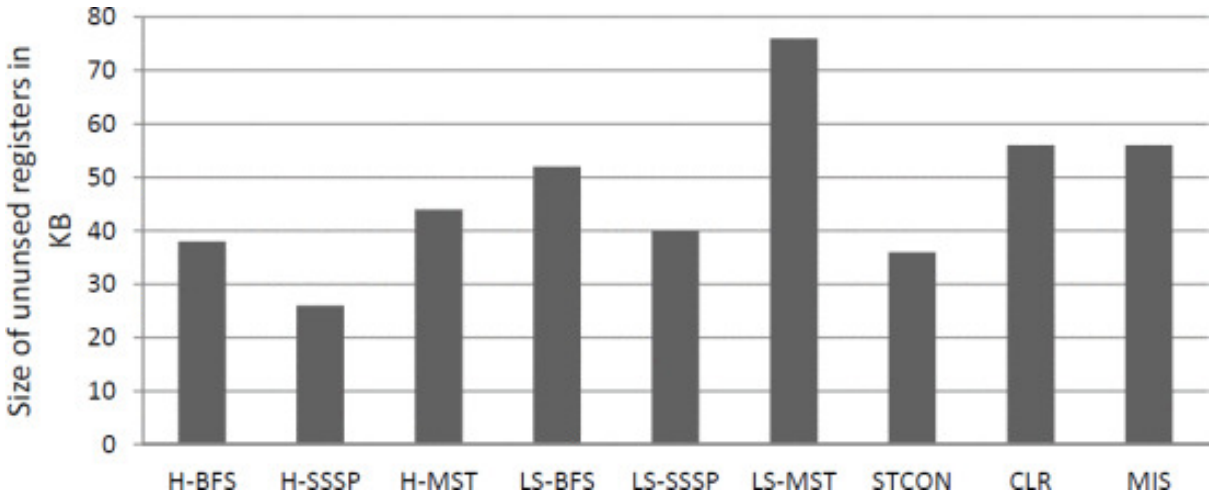

Figure 1.1: Unused registers out of 128KB register file [1]

The texture cache[3] in GPUs were used to put images onto triangles. In modern GPGPUs, texture cache is not being utilized in most of the applications. In this report, an idea is proposed to use texture cache either for prefetching or as a victim cache based on the application. Thus we can avoid warp-throttling and reduce the availability of dynamically unused registers and also use the spare registers in some other functional units.

1.1 Background

1.1.1 GPU

GPUs work along with CPUs as co-processors to increase the speed of the computation. Functions to be executed on CUDA devices are called kernels. An instance of a kernel is called a thread. Kernel consists of thousands of threads. Threads that execute the same instruction are grouped into a fixed size batch called Warp(NVIDIA) or a Wavefront(AMD).

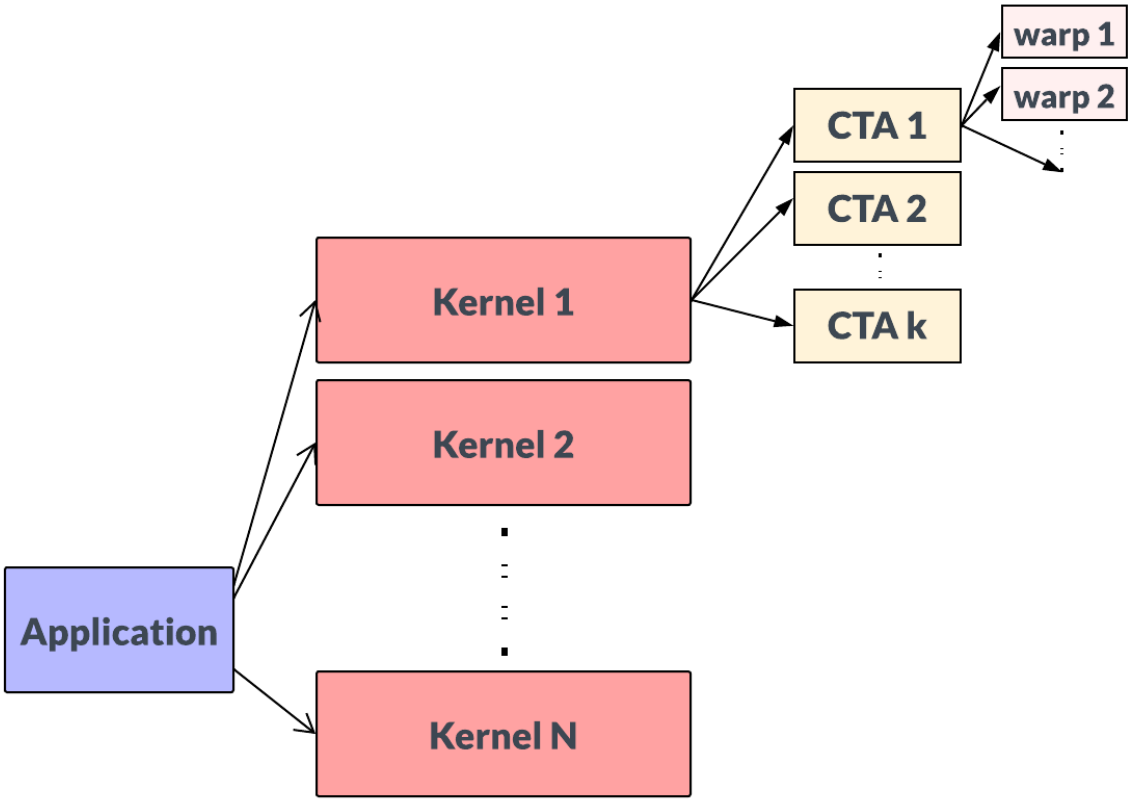

Figure 1.2: GPU Programmer's model

Each Kernel is spawned into several CTAs(Cooperative Thread Arrays). Each CTA has multiple Warps. These Warps are to be scheduled to different SMs(Streaming Multiprocessor). SM is a basic computation unit of the GPU. They achieve massive parallelism by adopting SIMT(Single Instruction Multiple Thread) Programming Model. GPUs have thousands of SIMT cores on chip.

A warp is group of 32 threads and are executed in a lock-step manner. A thread block is the granularity at which work is assigned to GPU cores called Streaming Multiprocessors (SMs).

1.1.2 Spare Registers

The register file size available to a SM will be in tens of KB. If we assume 256KB as register file size for 16 SMs. The total register file size will be 4MB which is huge when compared to L1 cache in general.

- **Statically Unused Registers**: SM limits the maximum number of registers allocated to a warp. The registers that are left left over with an SM after such allocations are called statically unused registers.

- **Dynamically Unused Registers**: A warp may be halted due to a memory latency or unavailability of resources. Some warps allotted to an SM may be terminated if they finish their execution. In the above both cases, the registers of the warp are not be utilized. Such registers are called as Dynamically Unused registers

1.1.3 Texture-Cache

The texture cache is a *read-only* cache. It will store the texture image data which has to be applied over other surfaces. This texture image data will be written once and read many number of times. It is placed closer to the shader-unit to reduce the latency. The hit rate of texture cache is high because of heavy reuse between neighbouring pixels.

Each SM has a dedicated L1 Texture cache, It is connected to L2 cache via a crossbar. To avoid the orientation dependency textures are stored in memory in a tiled manner. Each texture is broken into a series of n x n tiles and all the texels in the tile are stored before moving onto the next tile in the tile are stored before moving onto the next tile.

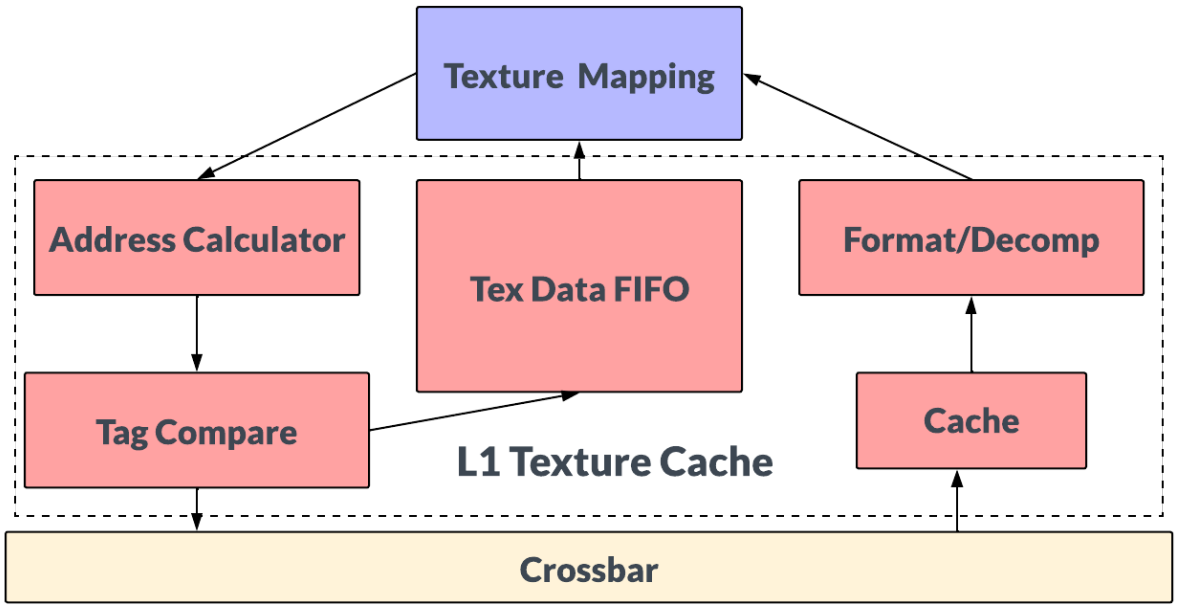

Figure 1.3: Texture Cache Architecture

The internal architecture of a texture cache is shown in figure-2.2. In addition to shader units, there will be texture units. When a shader performs a texture read, it sends a request to the texture unit. The texture unit batches up a bunch of requests and performs the addressing math on them—selecting mip levels and anisotropy, converting UVs to texel coordinates. Once it knows which texels it needs, it reads them through the texture cache.

Chapter 2

Related Work

The highly irregular memory accesses and data-dependencies in graph algorithms will increase the latency. To improve the performance we can optimize cache hierarchy to improve the performance in CPU[4]. In GPU, there is similarity in cache access pattern of same load across warps that execute same instruction stream[5].Using this similarity, Linebacker[2] is implemented over warp-throttling where both statically and dynamically unused registers are used as victim cache space. Spare registers[1] were also used to prefetch for graph algorithms in GPUs.

References

[1] N. B. Lakshminarayana and H. Kim, "Spare register aware prefetching for graph algorithms on gpus," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 614–625, 2014.

[2] Y. Oh, G. Koo, M. Annavaram, and W. W. Ro, "Linebacker: Preserving victim cache lines in idle register files of gpus," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, pp. 183–196, 2019.

[3] M. Doggett, "Texture caches," *IEEE Micro*, vol. 32, no. 3, pp. 136–141, 2012.

[4] A. Basak, S. Li, X. Hu, S. M. Oh, X. Xie, L. Zhao, X. Jiang, and Y. Xie, "Analysis and optimization of the memory hierarchy for graph processing workloads," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 373–386, 2019.

[5] G. Koo, Y. Oh, W. W. Ro, and M. Annavaram, "Access pattern-aware cache management for improving data utilization in gpu," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 307–319, 2017.