DECOUPLED L1 CACHE IN GPUs

Summer Project Stage 1 Report

Submitted in partial fulfillment of the requirements
for the completion of

Summer Project

by
Neeraj Prabhu
(Roll No. 200070049)

Under the guidance of
Prof. Virendra Singh

Department of Electrical Engineering
Indian Institute of Technology Bombay
August 2022

Acknowledgement

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

Neeraj Prabhu
Electrical Engineering
IIT Bombay

Abstract

Graphics Processing Units (GPUs) use caches to provide on-chip bandwidth as a way to address the memory wall. However, they are not always efficiently utilized for optimal GPU performance. The main source of this inefficiency stems from the tightly-coupled design of cores with L1 caches.

First, such a design assumes a per-core private local L1 cache in which each core independently caches the required data. This allows the same cache line to get replicated across cores, which wastes precious cache capacity. Second, due to the many-to-few traffic pattern, the tightly-coupled design leads to low per-core L1 bandwidth utilization while L2 is heavily utilized.

To address these inefficiencies, the L1 cache is separated from the GPU core and a new DC-L1 cache design is proposed. Decoupling the L1 cache from the GPU core reduces data replication across the L1s and increase their bandwidth utilization. Specifically, the paper investigates how to aggregate the DC-L1s and how to effectively design the NoC to improve performance while also reducing NoC area and power.

Contents

List of Figures

Chapter 1

Introduction

GPUs employ a 2 level cache hierarchy where each core consists of a private L1 cache, which is connected through an NoC to a shared and banked L2 cache. However, this type of architecture leads to inefficient utilization of resources. Due to the private nature of the L1 cache, a lot of data replication takes place across the cache lines. This leads to a reduction in bandwidth due to lower L1 hit rates and wastes its capacity. Apart from this, the large number of L1s and few L2 caches leades to a lot of ressure on the L2 cache and less on the L1.

The proposed solution to this is the use of Decoupled-L1 caches. This involves the aggregation of multiple L1 caches, in which each DC-L1 cache is accessed by the respective cores. This reduces data replication and improves cache bandwidth utilization. However, extreme aggregation may lead to a low peak L1 bandwidth and reduce performance. To balance the trade-off between replication waste and NoC overheads, a clustered DC-L1 design is proposed. The L1 caches are clustered and the data is shared only between the cluster of cores, instead of all DC-L1 cores. [1]

1.1 Motivation

1.1.1 Cache Line Replication across L1 Caches

On a miss, each core independently fetches the required data from the L2 cache. This may lead to replication across L1s if the cores request the same cache line, leading to wasted cache capacity. The replication ratio varies according to different applications. The waste due to data replication may not affect all applications. Only the applications that are sensitive to larger cache space are expected to benefit if the wasted cache space is eliminated. Applications with low L1 miss rates are not affected in the private L1 cache design. For replication sensitive applications, the reduction in L1 miss rates leads to higher on-chip bandwidth.

1.1.2 Low L1 Cache Utilization

The tight coupling of the L1 caches and GPU cores along with the many-to-few communication pattern puts more pressure on the few L2 banks and less pressure on the many L1 caches. This leads to low bandwidth utilization of the per-core L1 caches.

Chapter 2

Proposed Idea

2.1 Decoupled L1 Design

A DC-L1 node simply contains the DC-L1 cache, two queues to handle the traffic from/to the GPU core, and two queues to handle the traffic to/from the L2 and memory partitions. There are also 2 NoCs in the proposed design. The first NoC connects the GPU cores to the DC-L1 caches and the second NoC connects the DC-L1 caches with the shared L2 cache.

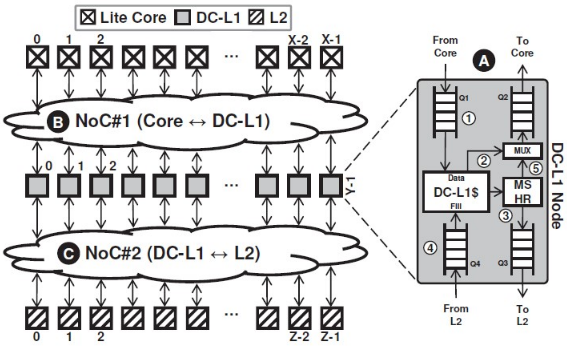


Figure 2.1: Decouple L1 Cache Design and NoC

Each request is queued by the DC-L1 in a FIFO manner. In the case of a read miss, the DC-L1 queues the read request fro the L2 cache and loads the reply once received. For a write miss, the no cache line is allocated to the DC-L1 and the data is directly written into the L2 cache.

2.2 Private DC-L1 Caches

Multiple DC-L1 nodes are grouped together into a larger DC-L1 node. Each DC-L1 node is accessed privately by a group of cores via a crossbar in NoC#1. For example, in Pr40 configuration (80 DC-L1s aggregated into 40 DC-L1s with double the capacity), 2 cores access each DC-L1 through a 2x1 crossbar.

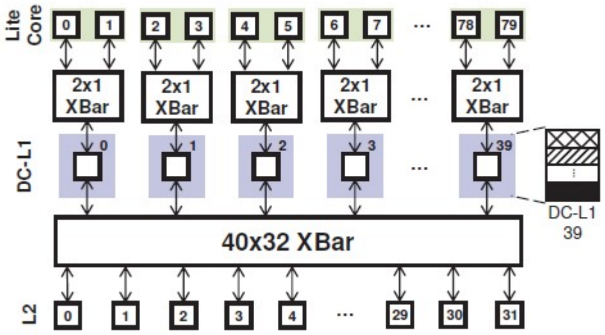


Figure 2.2: Pr40 Design

The L1 miss rate drops in the case of Pr40, Pr20 and Pr10 designs. However, Pr20 and Pr10 reduce the average performance for replication sensitive applications while Pr40 increases the IPC by 15%. This is due to the drop in peak L1 bandwidth due to using smaller crossbars in NoC#2.

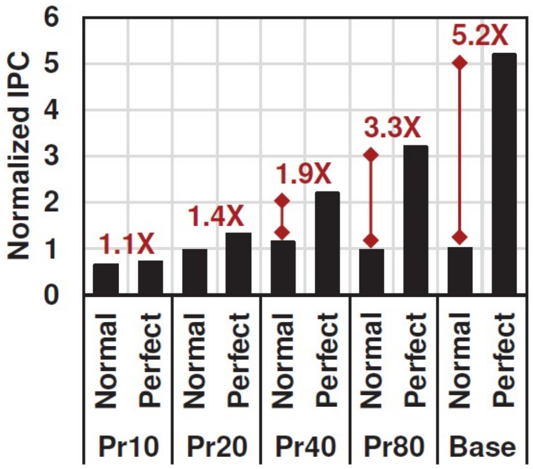


Figure 2.3: Perfect vs Normal Performance of different Private DC-L1 configurations

The perfect performance in the above chart refers to 100% hit rate and compares the actual performance for replication sensitive applications. As can be seen, Pr40 has a higher IPC boost as compared to the other configurations. It also reduces the NoC area and maintains the power consumption.

2.3 Shared DC-L1 Caches

In this design, each DC-L1 node caches from an exclusive non-overlapping address region. This ensures no cache line replication across the DC-1L1 nodes. Certain home bits are used to decide the cache line for the home DC-L1 node.

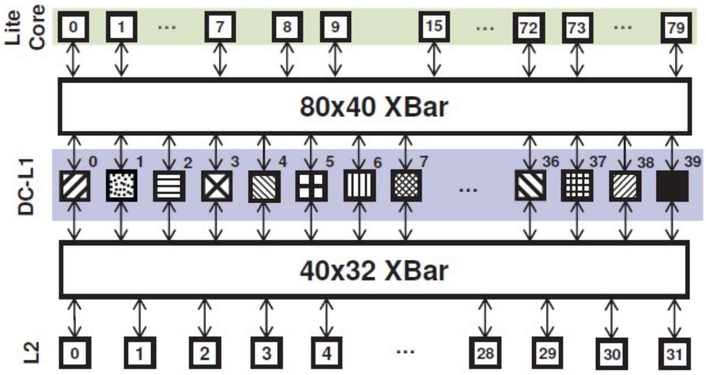
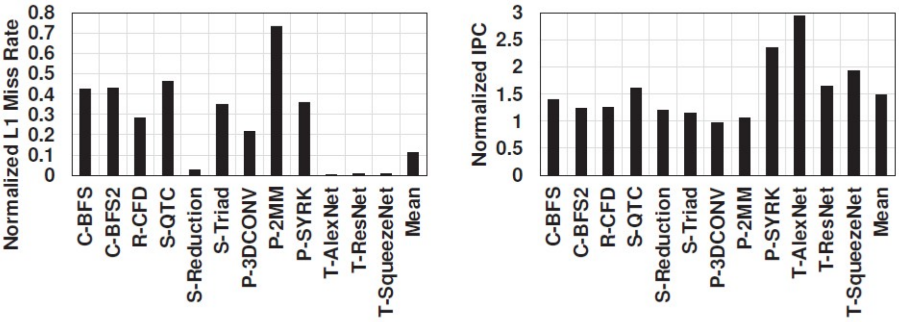


Figure 2.4: Sh40 Design

For replication sensitive applications in the Sh40 configuration, the DC-L1 miss rate drops significantly, at an average of 89%. This is because the applications have high data replication which is eliminated in the shared DC-L1 design. This improves the on-chip bandwidth. Although Sh40 improves performance, it increases the NoC area and power required in both NoC#1 and NoC#2. Some replication insensitive applications suffer a drop in performance due to work distribution imbalance.



(a) L1 miss rate (b) IPC

Figure 2.5: Sh40 Performance for various applications

2.4 Clustered Shared DC-L1 Caches

To reduce the NoC area and power overhead, a cluster-based shared model is proposed where eplication is eliminated across the DC-L1s in the same cluster. The NoCs are broken into small crossbars connecting a group of cores to a clustered DC-L1 node and the DC-L1 node to L2 slices instead of all the L2 nodes.

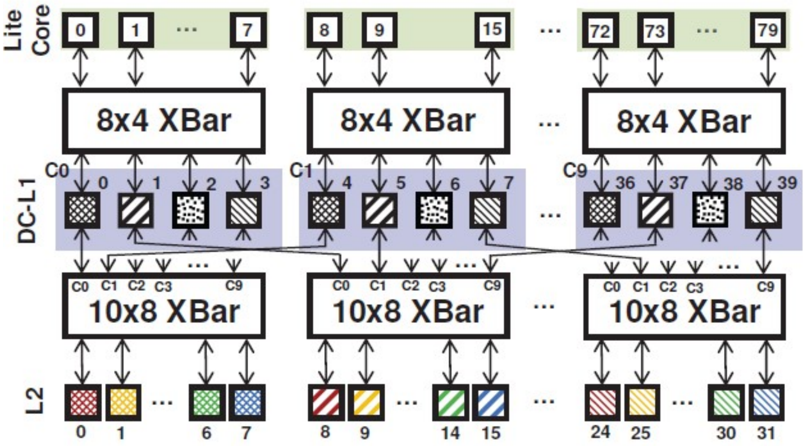


Figure 2.6: Sh40+C10 Design

The majority of the replication sensitive applications perform better with C1 because of their sensitivity to the additional effective cache capacity achieved by eliminating the replication. On the other hand, some applications perform better with clustering. This is because the controlled replication using clustering balances the useful L1 bandwidth from the additional cache capacity and from having multiple copies of a given cache line. The Sh40+C10 design drastically improves performance for replication insensitive applications.

To further boost the performance, the frequency of the NoC crossbars is doubled and the configuration is called Sh40+C10+Boost.This is a balanced design which limits replication and achieves significant performance improvements while reducing NoC area and power requirements.

Chapter 3

Future Work

I have understood the various configurations of using DC-L1 designs and their performances. In the future, I aim to understand the working of GPGPU-Sim and its source codes, especially for the L1 caches. Once I have completed this, I will modify the source code to meet the necessary design changes for the various configurations mentioned above. I will analyze the performance obtained and compare the same for various applications.

References

[1] M. A. Ibrahim, O. Kayiran, Y. Eckert, G. H. Loh, and A. Jog, “Analyzing and leveraging decoupled l1 caches in gpus,” in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 467–478, 2021.