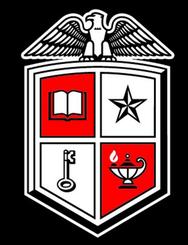
Concurrent Dynamic Memory Coalescing on GoblinCore-64 Architecture

Xi Wang, John D. Leidel, Yong Chen

Department of Computer Science, Texas Tech University



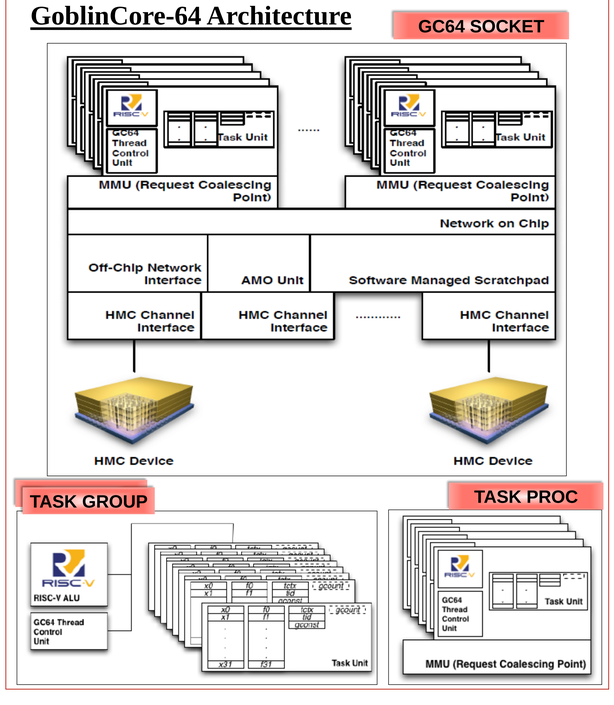
Abstract

The majority of modern microprocessors are architected to utilize multi-level data caches as a primary optimization to reduce the latency and increase the perceived bandwidth from an application. The spatial and temporal locality provided by data caches work well in conjunction with applications that access memory in a linear fashion. However, applications that exhibit random or non-deterministic memory access patterns often induce a significant number of data cache misses, thus reducing the natural performance benefit from the data cache.

In response to the performance penalties inherently present with non-deterministic applications, we have constructed a unique memory hierarchy within the GoblinCore-64 (GC64) architecture explicitly designed to exploit memory performance from irregular memory access patterns. The GC64 architecture combines a RISC-V-based core coupled with latency-hiding architectural features to a memory hierarchy with Hybrid Memory Cube (HMC) devices. In order to cope with the inherent non-determinism of applications and to exploit the packetized interface presented by the HMC device, we develop a methodology and associated implementation of a dynamic memory coalescing unit for the GC64 memory hierarchy that permits us to statistically sample memory requests from non-deterministic applications and coalesce them into the largest possible HMC payload requests.

In this work, we present two parallel methodologies and associated implementations for coalescing non-deterministic memory requests into the largest potential HMC request by constructing a binary tree representation of the live memory requests from disparate cores. We present the coalesced HMC memory request results from applications that exhibit linear and non-linear memory request patterns compiled for a RISC-V core in contrast with a traditional memory hierarchy.

GoblinCore-64 Architecture

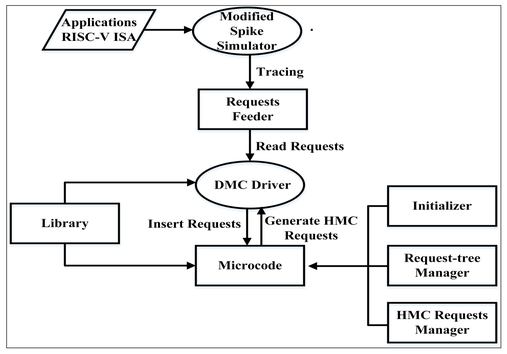


Introduction

This work introduces a concurrent dynamic memory coalescing model in GoblinCore-64 (GC-64) machine.

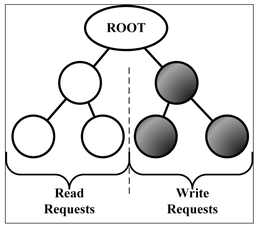
- The core machine model and instruction set are based on the RISC-V [1] instruction set architecture. We utilize the three-dimensional stacked memory devices in the form of Hybrid Memory Cube (HMC) devices [2] as the basis for the GC64 main memory, which provides uniquely high bandwidth over traditional DDR-based memory units alongside a packetized memory interface [3].
- The concurrent processing methodology and associated are designed and implemented in order to coalesce memory accesses from disparate GC64 cores into the largest potential HMC memory requests. We utilize a parallel, tree-based methodology in order to optimize the process of coalescing disparate read and write requests prior to dispatch HMC requests in a dynamic memory coalescing unit or DMC. In comparison with the conventional serial method, this concurrent DMC mechanism further decreases the number of the requests flushed into the hybrid memory cubes. Further, the new approach increases the efficiency of the memory coalescing, especially when coupled to multiple HMC devices or when executing applications whose memory request patterns are unusually non-deterministic.

Dynamic Memory Coalescing Components:



- **Spike Simulator:** modified to read all the memory requests from RISC-V cores and flush them in to the DMC Driver.
- **DMC Driver:** instantiates the microcode with synthetic memory traces and provides basic runtime debugging information.
- **Microcode:** consists of the **Request Tree Manager** and the **HMC Request Manager** units.
- **Request Tree Manager:** ingests the incoming memory requests and manages the internal tree state.
- **HMC Request Manager:** converts the memory requests to the equivalent HMC memory requests in the correct packetized form [4].

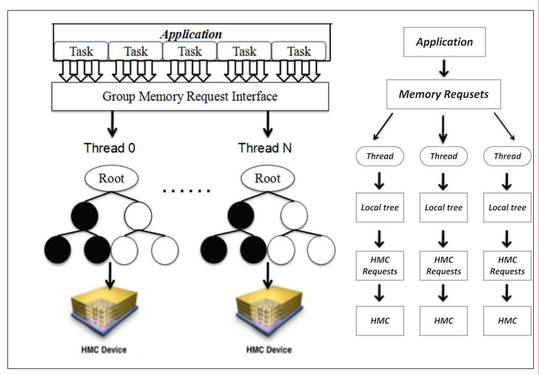
Tree Structure of Memory Requests



The coalescing tree is built as the sorting binary tree. Each node in the binary tree contains:
Task Operation: (Read/Write) and memory operation size (1,2,4,8 bytes)

- **Task ID**: source task ID for each specific request
- **Address:** The source address of the respective memory request

Architecture of Concurrent DMC Model:

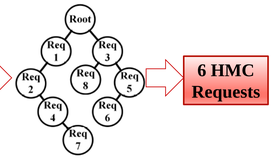


We design two different concurrent coalescing algorithms for the concurrent DMC methodology, the **address partitioned algorithm** (APA) and **work partitioned algorithm** (WPA).

- **APA**: the whole memory space is partitioned into several partitions or HMC devices. Each thread will only insert the request whose address falls into respective memory partition.
- **WPA**: based on partitioning the memory space, read and write requests are also partitioned. Two threads are assigned to work on the same memory partition. One thread handles the read requests, correspondingly, the other thread only handles the write requests.

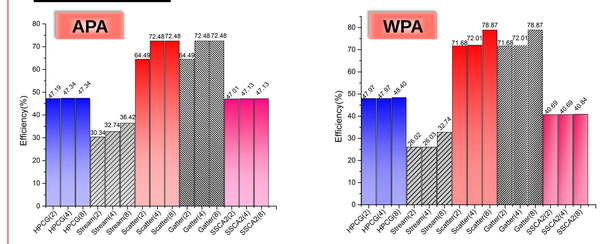
Coalescing Tree Logic

Request	OP	Address
1	RD16	0X10009FFF
2	RD8	0X000F1000
3	WR16	0X00001008
4	RD8	0X1000A008
5	WR8	0X100F0008
6	WR16	0X0000101F
7	RD16	0X1000A010
8	WR8	0X00001000



- ❖ The coalescing tree logic will only coalesce the requests with contiguous address.
- ❖ After coalesce the 8 requests in the table, request 4 and 7, request 3 and 8 will be coalesced into a HMC read and write request respectively. However, all the other requests cannot be coalesced.

Test Results



- We test a set of applications: HPCG, STREAM, Scatter & Gather, SSCAv2 2, 4 and 8 threads with the concurrent DMC model to justify the efficacy of our approach.
- The performance and scalability are stable and outperform the lack of coalescing as the thread concurrency scales from 2 to 8. The scatter and gather test case provides particularly good scalability at 8 threads as it coalesces 72.48% and 78.87% of the incoming memory accesses with APA and WPA respectively.

Conclusion & Future Exploration

- In this work, we have presented a new methodology to the core dynamic memory coalescing approach using concurrent DMC model
- We illustrate two parallel algorithms that increase the overall efficacy and scalability of coalescing memory requests designed for one or more hybrid memory cube devices.
- We also demonstrate the efficiency of this design through the test of STREAM benchmark, scatter/gather memory requests, HPCG and SSCAv2.

The future direction of the research will focus on the extension of the research based on the architectural direction of the GoblinCore-64 project.

- We will continue to utilize and expand our support for our use of hybrid memory cube devices specification 2.1 and conduct more tests on various benchmarks to do a more comprehensive analysis.
- We will also continue to improve the efficacy of the DMC methodology by optimizing the microcode implementing the microcode directly in RISC-V assembly in order to remove as much of the microcode latency as possible.

References:

[1] RISC-V ISA Specification. http://riscv.org/specifications/

[2] Hybrid Memory Cube. http://www.hybridmemorycube.org/

[3] GoblinCore-64. http://gc64.org/

[4] Concurrent Dynamic Memory Coalescing in GoblinCore-64 Architectire, MEMSYS 16' October 3-6, 2016, Washington, DC, USA, 2016 ACM. ISBN 123-4567-24-567/08/06, DOI: 10.475/123_4

Contacts

Xi Wang [xi.wang@ttu.edu]

John D. Leidel [john.leidel@ttu.edu]

Yong Chen [yong.chen@ttu.edu]