Analysis of memory scheduling policies

Aneta Ivaničová

The ACME accelerator architecture.

Main memories have become performance bottlenecks, which means that processors are spending their time waiting for memory operations. For this reason, we are presenting an analysis of the memory scheduler called BLISS in this article.

The memory bandwidth is the maximum amount of data that can be received and written to the memory or read from the memory and returned to the processor per unit of time.[3] The memory throughput, the actual transfer rate, which is limited by the memory bandwidth, has made main memory a performance bottleneck. Although maximizing memory throughput is beneficial in some cases, in other cases it might degrade overall system performance. In a system where multiple cores share a common memory interface, concurrent memory requests from different threads executing on different cores can interfere with each other while accessing the shared DRAM main memory system. This is also called inter-thread interference and it degrades system performance and slows down applications. To mitigate inter-thread interference we decided to study four state-of-the-art memory schedulers, then we picked one of them, thoroughly analyzed it, implemented it in an open source RISC-V ISA based simulator named Coyote, then tests began, and after that evaluation was expected to start. Our choice for the memory scheduler was the Blacklisting Memory Scheduler called BLISS. Among the reasons were that BLISS helps to achieve higher system performance and fairness, while incurring low hardware cost and scheduling latency. Also, BLISS does not implement total order-based ranking of memory requests, which leads to lower complexity, so that strict double data rate memory timing protocols can be met.

MEEP, ACME and Coyote

The project revolves around MEEP (MareNostrum Experimental Exascale Platform) which is a flexible FPGA-based emulation platform that serves as a basis for creating European-based chips and an infrastructure to enable rapid prototyping. The MEEP project is currently emulating a self-hosted accelerator called ACME (Accelerated Compute Memory Engine), which has two main components: the VAS tiles and the memory tiles. The VAS tile is a cluster of 8 scalar cores, each core supports a Vector Processor Unit (VPU) and two Systolic Array units. A scalar core has its own L1 instruction and data caches, and a L2 data cache slice which also can act as a scratchpad. A memory tile consists of a memory controller, a slice of high bandwidth memory, a JTLB (address-translation cache) and the MCPU which is designed to manage memory requests and related operations. ACME aims to improve the performance of dense (compute-bound) and sparse (memory-bandwidth-bound) workloads, and to find the balance between the memory hierarchy design and the number of fused multiply-add (FMA) units available in the system that the performance depends on. MEEP proposes Coyote which is a performance modeling tool based on two open source simulators called Sparta and Spike. Coyote provides detailed insights at various levels and granularity, while focusing on data movement and the modeling of the memory hierarchy of the system.[5]

The main memory

In case of ACME, the main memory is a high bandwidth memory (HBM), which is a stacked double data rate (DDR) memory that is connected via an interposer to an FPGA. At the bottom of the

HBM is a base logic die. On the top of the logic die are stacked DRAM dies, which are connected through-silicon vias (TSVs), as shown in Figure 1. Each slice of core DRAM die has two channels with eight independent bank groups. The utilization of memory banks enables concurrent DRAM main memory accesses and increases memory bandwidth. All banks within a channel share the command, address and data buses of the channel. Each bank has a structure of a two-dimensional array of rows and columns.

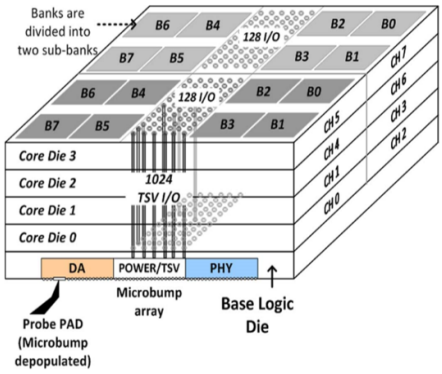


Figure 1: The high bandwidth memory architecture.[4]

Memory access terminologies

Before moving on to the memory scheduler we need to explain a term called a row-buffer hit. So, when a processor generates a memory request, the data is searched after in all levels of cache memory and if the data is not found in the last level cache (LLC), then the memory controller looks for the data in the main memory by sending the physical address of the data via the memory address bus. On a data access, the entire row containing the data is copied into a structure called the row buffer. A subsequent access to the same row can be served from the row buffer itself and it does not need to access the array. This is called a row-buffer hit. On the other hand, when accessing a different row, the previous row of data must be returned, and then the next row can be accessed. This type of access is called a row-buffer miss or conflict.

The memory scheduler

To mitigate inter-thread interference, BLISS separates threads into two groups: interference-causing and vulnerable-to-interference. When a large number of consecutive requests are served from the same thread other threads will likely stall, therefore BLISS counts the number of consecutive requests served from the same thread. When this count exceeds a threshold, BLISS places the thread into the interference-causing group, also called the blacklisted group, and other threads are placed in the vulnerable-to-interference group. BLISS has two components: The Blacklisting Mechanism and The Memory Scheduling Mechanism. The Blacklisting Mechanism needs to keep track of the following three quantities: the Thread ID of the last scheduled request, the Number Of Requests Served from a thread, and the Blacklist Status of each thread. Before a request is issued by the memory controller, it compares the thread ID of the current request and the Thread ID of the last scheduled request. If they are the same, then the Number Of Requests Served counter is incremented. However, if they are different then the counter is reset to zero and the Thread ID register of the last scheduled request is updated with the thread ID of the current request. If the Number Of Requests Served counter exceeds a blacklisting threshold, which can be four according to the research papers, then the thread ID of the current request is blacklisted, and the counter is reset to zero. The blacklisting information is cleared periodically after every Clearing Interval, which is ten thousand cycles in the papers. In addition, this information is used by the Memory Scheduling Mechanism to determine the scheduling priority of a request. Requests from threads that are placed in the vulnerable-to-interference group or also called non-blacklisted threads are prioritized, then row-buffer hit requests follow because they optimize bandwidth utilization, and finally older requests are prioritized for forward progress. It is important to prioritize threads which are vulnerable to interference because they are compute-bound, therefore it is beneficial when they spend less time waiting for memory operations.

Conclusion

BLISS optimizes applications by exploiting the imbalance in number of memory requests among the threads. Threads that have less number of memory requests are prioritized over threads which in comparison are more memory-intensive. Since applications ported to Coyote (AXPY, Matmul, Somier, SPMV) exhibit little imbalance in terms of number of memory requests among its threads, it would be interesting to port a few applications that could demonstrate suitable use cases for the BLISS optimization. We hope to do this in our future work for evaluation purposes.

References

[1] Fell, A., Mazure, D., Garcia, T., Pérez, B., Teruel, X., Wilson, P., & Davis, J. (2021). The MareNostrum Experimental Exascale Platform (MEEP). *Supercomputing Frontiers And Innovations*, 8(1), 62-81. doi:http://dx.doi.org/10.14529/jsfi210105

[2] Subramanian, L., Lee, D., Seshadri, V., Rastogi, H., & Mutlu, O. (2014, October). The blacklisting memory scheduler: Achieving high performance and fairness at low cost. In 2014 IEEE 32nd International Conference on Computer Design (ICCD) (pp. 8-15). IEEE.

[3] John Burke 2015, TechTarget, viewed 28 August 2021, <https://searchnetworking.techtarget.com/definition /throughput>

[4] Jun, H., Cho, J., Lee, K., Son, H. Y., Kim, K., Jin, H., & Kim, K. (2017, May). Hbm (high bandwidth memory) dram technology and architecture. In 2017 IEEE International Memory Workshop (IMW) (pp. 1-4). IEEE.

[5] Pérez, B.; Fell, A.; Davis, J.D. Coyote: an open source simulation tool to enable RISC-V in HPC. A: Design, Automation and Test in Europe Conference and Exhibition. "2021 Design, Automation & Test in Europe Conference & Exhibition (DATE): Grenoble, France, 1-5 February 2021: proceedings". Institute of Electrical and Electronics Engineers (IEEE), 2021, p. 130-135. ISBN 978-3-9819263-5-4. DOI 10.23919/DATE51398.2021.9474080.

PRACE SoHPCProject Title

Analysis of data management policies in HPC architectures

PRACE SoHPCSite

Barcelona Supercomputing Center, Spain

PRACE SoHPCAuthors

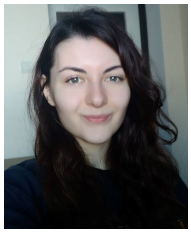
Aneta Ivaničová, Slovakia

PRACE SoHPCMentor

Borja Pérez, BSC, Spain

PRACE SoHPCContact

Aneta, Ivaničová, Matej Bel University in Banská Bystrica
Phone: +421 918 962 114
E-mail:
aivanicova@student.umb.sk



Aneta Ivaničová

PRACE SoHPCSoftware applied

Coyote

PRACE SoHPCMore Information

github.com/borja-perez/Coyote

PRACE SoHPCAcknowledgement

I would like to express my gratitude to my project partner Regina M. Gachomba, to our mentor Borja Pérez, to our co-mentor Teresa G. Cervero, to Rahul Shrivastava, to Alexander Fell, and to the whole MEEP team at the Barcelona Supercomputing Center.

PRACE SoHPCProject ID

2101