Memory Bandwidth: What can we improve?

Francis Birck Moreira[1], **Marco Antônio Zanata Alves**[1]

[1]Instituto de Informática – Universidade Federal do Paraná (UFPR)
Curitiba/PR

{fbm,mazalves}@inf.ufpr.br

Abstract. *As the gap between processor performance and memory latency widens, the computer architecture research area is always out for new solutions for efficient memory access. In this paper, we analyze the maximum achievable gain by simulating an ideal DRAM memory that always services requests as row buffer hits, thus minimizing memory access latency and queuing. We reproduced the state-of-the-art ConGen2 technique from Natale et al. to measure how much improvement is still available. ConGen2 minimizes row buffer misses according to a min-k-cut solution based on all memory accesses of a target application. We observed that ConGen2 improves memory usage by 2.30% on average, while the ideal memory gains 36.88% on average for the chosen benchmarks. This gap leaves a wide margin of benefits to be gained.*

1. Introduction

The computer industry's focus on processor performance has driven technological innovation in the last decades. However, the memory industry has focused on increasing memory capacity and bandwidth, while the memory latency has lagged behind the processor performance significantly [Mutlu et al. 2020]. This culture led to the current "memory wall" in the architecture industry [Santos et al. 2021]. The improvements in the processor design are meaningless if the processor is idly waiting for memory most of the time. Researchers are now developing processing-in-memory (PIM) as a solution to this issue [Santos et al. 2021]. The idea behind PIM is to avoid the high cost of memory bandwidth and transference latency by adding processing capabilities to the memory [Santos et al. 2021].

However, memory bandwidth is not entirely used due to inherent limitations of its functionality. Open row accesses are much faster than accesses on a closed row, but large systems cannot map the memory efficiently for all applications. Hence, we propose exploring the mapping of addresses to DRAM to improve bandwidth usage.

2. Experiments

Natale et al. [Natale et al. 2020] have recently developed methodologies aimed to do this exact proposal for limited and small systems. Their work transforms a list of addresses into a min-k-cut problem, where they find the k bits of the addresses that minimize changes between the addresses, thus minimizing row accesses to different rows. k is defined depending on the size of the system. For instance, a system with 48 bits of addressing, 4 channels (2 bits), 8 banks (3 bits), 8192 B of row buffer (13 bits), would require $k = 30$. We have replicated the code of their work for memory-intensive applications of the SPEC-CPU 2017 and NAS-NPB benchmarks. We chose applications with high

memory bandwidth consumption. We used the SiNUCA simulator [Alves et al. 2015] to measure the performance attained by the ConGen2 technique in 200 M instruction traces in comparison to an oracle where every access to the memory is treated as a row buffer hit in a different bank. We chose 200 M instruction traces as the simulator does not detail context switches, estimating this amount to take roughly 1-10 ms.

Figure 1 shows the results. We can see ConGen2 is far from ideal in terms of performance benefits. Our replication is also not precise, as we must make assumptions for the position of channel and rank bits which are not described in their paper.

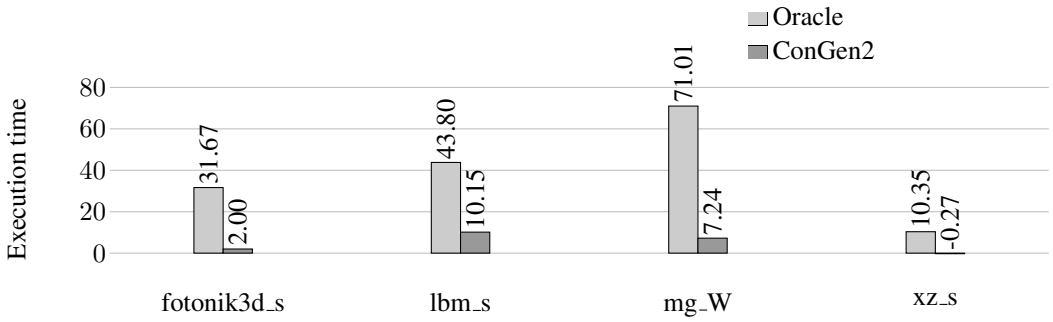

Figure 1. Application performance. Baseline is DRAM, Oracle is a DRAM where every access is a row buffer hit, and ConGen2 is Natale et al.'s work.

3. Conclusions

In conclusion, we can see that the replicated technique only considers row buffer miss and hit, being inadequate for a real system with multiple channels. However, there is merit to their technique, as seen in the row buffer miss reduction. In the future, we intend to leverage this technique for specific pages controlled by the programmer, where we can infer the best bit mapping for a page with compiler profiling.

Agradecimentos

Gostaríamos de agradecer ao Instituto Serrapilheira pelo financiamento desta pesquisa.

References

Alves, M. A. Z. et al. (2015). Sinuca: A validated micro-architecture simulator. In *2015 IEEE 17th HPCC*, pages 605–610. IEEE.

Mutlu, O. et al. (2020). A modern primer on processing in memory. *arXiv preprint arXiv:2012.03112*.

Natale, V. et al. (2020). Efficient generation of application specific memory controllers. In *MEMSYS*, pages 233–247.

Santos, P. C. et al. (2021). Survey on near-data processing: Applications and architectures. *JICS*, 16(2):1–17.