Precision Aware Bank Separated Data Placement for Enhanced DRAM Performance in Mixed-Precision HPC Workloads

Divyansh S. Maura
dsmaura@ucdavis.edu
University of California, Davis
Department of Electrical & Computer Engineering
Davis, CA, USA

Venkatesh Akella
akella@ucdavis.edu
University of California, Davis
Department of Electrical & Computer Engineering
Davis, CA, USA

Abstract

Mixed-precision (MP) computing, which uses data with different bit-widths, is a promising way to improve the performance and energy efficiency of High-Performance Computing (HPC) workloads. While decreasing the precision of data representation saves storage, the benefits in other critical DRAM performance metrics, such as data movement, power consumption, and row buffer locality, do not always scale proportionally. This discrepancy is due to architectural constraints in DRAM subsystems, where fixed access granularities and bank contention limit the benefits of precision reduction. To address this, a novel DRAM optimization technique is proposed: precision-aware bank-separated data placement. This method involves dedicating specific DRAM banks to store data of a particular precision, which can improve data locality and reduce row buffer issues, leading to better performance and energy efficiency. Preliminary results suggest that this approach improves row-buffer hits by an average of 24% and reduces DRAM activation energy by 12%. Sensitivity studies on different architectures like DDR4, LPDDR, and HBM show that architectural characteristics influence how effective this approach is

1 Introduction

Modern high-performance computing (HPC) and machine learning workloads increasingly rely on mixed-precision arithmetic to balance computational efficiency with numerical accuracy [6]. By leveraging lower precision formats where the application's accuracy requirements allow, significant reductions in computational cost and memory footprint can be achieved. [4] While halving the precision in terms of bit length(for eg, 16-bits to 8-bits) reduces storage requirements by 50%, prior work demonstrates that gains in metrics such as bandwidth utilization, power efficiency, and row-buffer locality are often sublinear—improving by only 11–24% in practice. [1, 7] This discrepancy arises from architectural constraints in DRAM subsystems, where fixed access granularities and bank contention limit the benefits of precision reduction. Optimization approaches assume idealized memory systems where data movement and bank parallelism scale with precision—an assumption invalidated by modern DRAM architectures like DDR4 and HBM [5] This causes poor spatial locality and frequent row activations, undercutting the bandwidth and power advantages of reduced precision formats [7, 3]. We hypothesize the aforementioned factors open up the potential for optimizations that can increase the improvement in other parameters to a similar degree and achieve a better degree of reduction across all parameters. To realize this added potential, we propose a novel DRAM optimization technique: bank-separated data placement based on precision. This work shows that by dedicating specific DRAM banks to store data of a particular precision, we can improve data locality and reduce row buffer issues leading to better performance in terms of bandwidth utilization and energy efficiency

The primary motivation for this work is the empirical observation that reducing precision does not proportionally improve memory system performance. Several underlying reasons contribute to this phenomenon.

Consider the transition from a 32-bit floating-point (FP32) representation to a sixteen-bit floating-point (FP16) representation. Ideally, we would expect a 50% reduction in storage requirements and a corresponding 50% decrease in data volume transferred to and from DRAM. However, our initial investigations and observations on HPC workloads, such as STREAM, suggest that the improvements in other memory usage metrics are often significantly less than 50% [2].

When the memory footprint is halved, the actual number of memory access requests might not be the same as the instruction overheads related to memory access and address calculation remain largely unchanged. Furthermore, the burst-oriented nature of DRAM access means that more data elements of mixed-precision can be coalesced into the memory burst as compared to single precision. This leads to a less than ideal reduction in total data movement.

More importantly, row buffer locality, a critical factor in DRAM performance, do not scale propotionally in mixed-precision scenarios when data of different precisions are interleaved within the same DRAM banks. Frequent switching between data of different types can lead to increased row buffer misses and higher access latencies, negating some of the potential benefits of reduced data size. Our preliminary analysis indicates that moving from FP32 to FP16 might only yield around a maximum of 30% improvement in row buffer hit rate, far from the expected 50% reduction in data size. This discrepancy highlights the need for more intelligent data placement strategies that are aware of the precision of the data being stored.

While our main analysis and evaluation is based on software-based simulations, we also conducted a preliminary profiling study using real workloads executed on a local machine. This comparative evaluation of single-precision versus mixed-precision workloads demonstrates that key memory system parameters—such as bandwidth utilization, power consumption, and row-buffer locality—do not scale proportionally with the storage savings achieved through reduced-precision formats. These observations, summarized in Table 1, provide empirical support for our hypothesis.

Metric	Single precision	Multi precision	Percentage change
Storage	2.2 GB	1.1 GB	-50
Data movement	43.51 GB-s	32.2 GB-s	-26
Mem power	1186 mW	1028 mW	-16.66
Activation energy	130.5 mJ	108.8 mJ	-16.6
Row hits	53	66.5	25.3

Table 1: Experimental Data to Illustrate the Problem

2 Precision-Aware Bank-Separated Data Placement

To mitigate the non-proportional benefits and improve DRAM performance in mixed-precision workloads, we propose a static bank-separated data placement strategy based on the precision of the data. The core idea is to dedicate specific DRAM banks to store data of a particular precision. For a system utilizing two primary precisions, for example, FP16 and INT8, one set of banks would primarily store FP16 data, while another set would store INT8 data. This static partitioning ensures that requests to one bank exhibit homogeneous access characteristics, improving row-buffer hit potential. While for experimentation this approach was implemented manually via trace profiling and manipulation in the software simulator DRAMSim3, we also propose a memory controller-based precision identification mechanism using hints via precision identifying metadata and a modified address translation mechanism.

To implement this strategy, the memory controller needs to be aware of the precision associated with each memory access request. Experimentally achieved through manual trace analysis and tagging, a more practical approach for a real system would involve modifications at a higher level. During the code generation phase, the compiler has significant knowledge about the data types and precision used for different memory accesses. The compiler can use that data to generate a separate descriptor table that maps memory regions (identified by their address ranges) to their dominant precision. The precision can be annotated via a 1-bit or 2-bit tag depending on the number of precision levels that the banks need to be grouped into. For our further discussion, we will be assuming two precision levels, hence annotated by a 1-bit tag. This table can be loaded as an additional cache added to the memory controller. The memory controller can then consult these tables to determine the precision of data being accessed within those regions. Whenever a memory access request arrives, the memory controller extracts the address and consults the descriptor tables to determine the precision associated with that address range.

Furthermore, inside the memory controller, the address translation unit is modified to take into consideration the precision tag taken from the descriptor table while selecting the bank. The standard addressing, as followed in DRAMSim3, follows the pattern-Channel(3 bits)-Ranks(1 bit)-BankGroup(2 bits)-Bank(2 bits)-Row(16 bits)-Column(4 bits). To separate the banks into sets based on, we replace the MSB with the precision tag while the remaining lower order bits act as the index for a bank in the precision set. This ensures that the additional optimizations from the bank organization are disrupted minimally. In this approach, this new address mapping can be limited to one particular set of banks or a rank

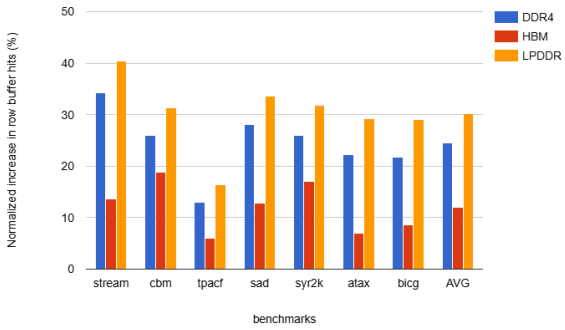

Figure 1: Percentage increase in the row-buffer hits as compared to normal bank setup

to prevent these changes from disrupting the memory access of the programs that don't need to be precision aware. The allocation of banks for different precision levels can be determined based on the prevalence and access frequency of each precision in the target workload. For workloads with a dominant lower precision, a larger number of banks could be allocated to it to maximize the benefits. This allocation could be static, determined at compile time, or during application initialization, based on profiling the workload characteristics.

3 Evaluation Methodology and Results

The proposed technique was evaluated using DRAMSim3, a trace-based cycle-accurate DRAM simulator. The traces for the simulation were produced using the Intel Pin tool. Three types of memory configurations were used for the simulation: DDR4, LPDDR, and HBM. The benchmarks used were from three suites—STREAM, deal.II, and Polybench—and included stream, cbm, tpacf, sad, syr2k, atax, and bicg

The following parameters were evaluated during simulation - (a) *Row Buffer Hit Rate:* The percentage of memory accesses that hit the currently active row in a DRAM bank, (b) *Bandwidth utilization* normalized to execution time: The total volume of data transacted through DRAM while also showing increased bandwidth utilization efficiency, (c) *Activation Energy Consumption:* The total energy consumed due to the row buffers being activated, and *DRAM Power Consumption:* The total power used by the DRAM system during the execution.

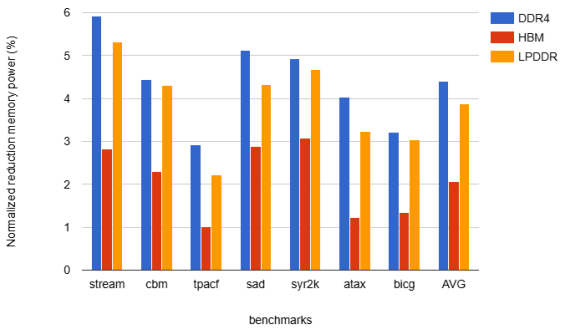

Figure 2: Percentage decrease in the total memory power as compared to normal bank setup

The results (See Figures 1 and Figure 2) show that the bank-separated data placement optimization leads to an average improvement of 24% in row buffer hits compared to a baseline DDR4 system with traditional bank organization. This directly translates to a reduction in row activations and precharges, which results in a 12% decrease in system activation energy and a 4.5% reduction in overall DRAM power consumption. The benchmarks with more regular memory access patterns show greater gains

In LPDDR4, which is designed for energy efficiency, row buffer hit gains are 24% better, but the power savings are lower because pre-existing power optimizations overshadow the new gains. For HBM, which is optimized for high bandwidth and parallelization, the gains in power savings are lower due to its highly parallel bank organization, wide I/O, and high number of banks per chip.

The proposed optimization technique also reduces variability in memory access latency. By grouping data of similar precision within dedicated memory banks, the likelihood of row-buffer conflicts is reduced, thereby minimizing row activations and allowing for more effective access coalescing. The result is a more predictable memory access pattern, leading to lower and more stable latency distributions. This is beneficial for real-time or latency-sensitive workloads, where reduced variability can significantly improve system responsiveness and QoS guarantees.

Our analysis of different precision distributions within the synthetic benchmark revealed that the effectiveness of our optimization is influenced by the relative proportion of each precision. When one precision dominates, dedicating more banks to it can yield the better benefits. However, in scenarios with a more balanced mix, the static partitioning needs to be carefully considered to avoid underutilization.

3.1 Overhead analysis

Implementing precision-aware bank mapping is bound to introduce some overhead. The optimized memory controller will be more complex compared to the default one as it needs to maintain information about the precision of data and implement the logic for mapping addresses to banks based on this information. This could lead to a slight increase in the cycle time of the memory controller. Added complexity will also increase the area overhead. The exact effect of these overheads on overall performance will depend on the specific implementation and characteristics of the workload. However the benefits gained from improved row buffer hits and reduced bank conflicts will outweigh these relatively minor overheads, especially for memory-intensive mixed-precision workloads.

4 Conclusion

We show that though mixed-precision computing reduces storage, other benefits such as row buffer hits, power, and data movement overhead are not proportional to the reduction in the memory precision. To address this, we proposed a novel DRAM optimization technique: static bank-separated data placement based on the precision of the data. Our simulation results on a DDR4 memory system demonstrate the effectiveness of this approach, yielding an average improvement of 24% in row buffer hit rates and a 12% reduction in overall DRAM power consumption. These gains are attributed to the enhanced data locality within banks dedicated to specific precisions, leading to fewer row buffer misses and reduced inter-bank interference. Furthermore, our comparative analysis across different DRAM architectures suggests that the benefits of precision-aware bank separation are likely to extend beyond DDR4.

Acknowledgement

This work was supported in part by NSF Core Award 2225882.

References

[1] Marc Baboulin, Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Julie Langou, Julien Langou, Piotr Luszczek, and Stanimire Tomov. 2009. Accelerating scientific computations with mixed precision algorithms. *Computer Physics Communications*, 180, 12, 2526–2533.

[2] Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Piotr Luszczek, and Stanimir Tomov. 2008. Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy. *ACM Transactions on Mathematical Software (TOMS)*, 34, 4, 1–22.

[3] Stefano Cherubin and Giovanni Agosta. 2020. Tools for reduced precision computation: a survey. *ACM Computing Surveys (CSUR)*, 53, 2, 1–35.

[4] Dorra Ben Khalifa and Matthieu Martel. 2023. Everything you need to know about reduced mixed precision computation in numerical programs.

[5] Michael O. Lam, Jeffrey K. Hollingsworth, Bronis R. de Supinski, and Matthew P. Legendre. 2013. Automatically adapting programs for mixed-precision floating-point computation. In *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing* (ICS '13). Association for Computing Machinery, Eugene, Oregon, USA, 369–378. ISBN: 9781450321303. DOI: 10.1145/2464996.2465018.

[6] Enrico Reggiani, Alessandro Pappalardo, Max Doblas, Miquel Moreto, Mauro Olivieri, Osman Sabri Unsal, and Adrián Cristal. 2023. Mix-gemm: an efficient hw-sw architecture for mixed-precision quantized deep neural networks inference on edge devices. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 1085–1098. DOI: 10.1109/HPCA56546.2023.10071076.

[7] Yutong Wang and Cindy Rubio-González. 2024. Predicting performance and accuracy of mixed-precision programs for precision tuning. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, 1–13.