The Impact of Maximum Vector Length on Cache Management Techniques in RISC-V Vector Extension

Shunya Nomura
Graduate School of Information Sciences,
Tohoku University
Sendai, Miyagi, Japan
nomura.shunya.r5@dc.tohoku.ac.jp

Jiaheng Liu
RIKEN Center for Computational Science
Kobe, Hyogo, Japan
jiaheng.liu@riken.jp

Keichi Takahashi
D3 Center, The University of Osaka
Toyonaka, Osaka, Japan
Cyberscience Center, Tohoku University
Sendai, Miyagi, Japan
keichi@tohoku.ac.jp

Hiroyuki Takizawa
Cyberscience Center, Tohoku University
Sendai, Miyagi, Japan
takizawa@tohoku.ac.jp

1 Introduction

In recent years, computers have faced increasing demands for energy efficiency. Single Instruction Multiple Data (SIMD) execution exploits data-level parallelism (DLP) to enable efficient processing. RISC-V [6], an open-source instruction set architecture, provides the RISC-V Vector Extension (RVV) as its SIMD-style instruction set. In RVV, the instruction's vector width is decoupled from the hardware datapath width, allowing the same binary to execute on processors with varying Maximum Vector Length (MVLs).

Since RVV leverages DLP, a single vector instruction can operate on many data elements in parallel. Consequently, effective cache management is critical to accelerate data movement and keep supplying data to functional units. Indeed, Gómez et al. [3] employed cache bypassing for low-locality data as part of optimizing HPCG on a vector processor. While such cache-targeted optimizations exist, to our knowledge there has been no report on systematic evaluation of cache management techniques across different MVLs. Changes in MVL can plausibly alter cache access behavior; for example, alignment effects are frequently discussed in vector computing and are particularly salient for shorter vector lengths, where the performance impact of unaligned accesses is more pronounced.

In this paper, we focus on two cache management techniques: cache replacement policies and non-temporal hints. We hypothesize that, even for the same program under RVV, processors with different MVLs exhibit distinct memory-access patterns, which in turn can change the choice of the most effective cache management techniques. Our goal is to clarify how the MVL of a processor influences the efficacy of cache management under RVV.

2 Evaluated Cache Management Techniques

This paper focuses on cache management techniques, specifically replacement policies and non-temporal hint instructions.

2.1 Replacement Policies

A replacement policy is an algorithm that determines which cache line to evict when a new cache line is inserted into the cache. In this paper, we compare the following three replacement policies.

- Least Recently Used (LRU)
- Static Re-Reference Interval Predictor (SRRIP) [1]
- Signature-based Hit Predictor (SHiP) [2]

LRU, the most fundamental replacement policy, leverages temporal locality by expecting that recently accessed lines will be re-referenced in the near future. SRRIP employs a value called the Re-Reference Prediction Value (RRPV) to learn access patterns and selects the victim line based on its RRPV. The SHiP policy uses signatures to capture access patterns, incorporating the learned results into the RRPV assignment when inserting cache lines, similar to SRRIP.

2.2 Non-Temporal Hints

A non-temporal hint explicitly indicates low memory access locality to the cache, thereby preventing cache pollution. In this paper, we evaluate three methods for utilizing non-temporal hints.

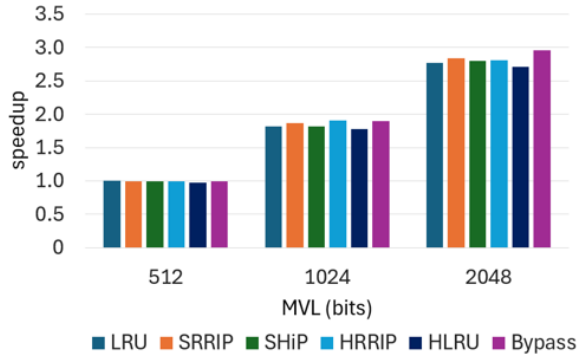
- Cache Bypass
- Hint-assisted RRIP (HRRIP)
- Hint-assisted LRU (HLRU)

In Cache Bypass, when a memory access is marked with a low-locality hint, a cache miss does not trigger the allocation of a new cache line. In HRRIP and HLRU, when such a hint is provided, the new cache line is inserted with the lowest replacement priority, making it more likely to be evicted.

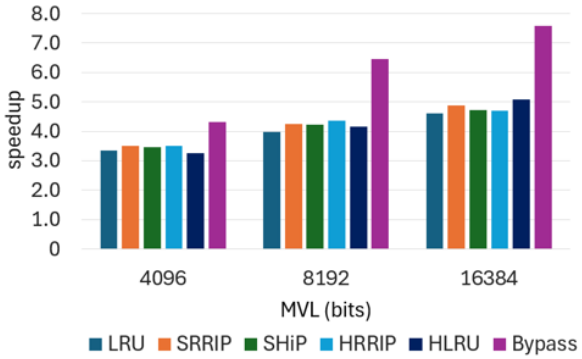
3 Evaluation

In this paper, we employ a gem5-based implementation of a RISC-V vector processor model [4, 5]. The benchmarks are selected from the RISC-V Vectorized Benchmark Suite (RiVEC) [4], specifically jacobi-2d and spmv. For both benchmarks, aligned versions of the matrices or arrays are also prepared to examine the impact of alignment. Using these benchmarks, we compare six cache management techniques.

SC 25, November, St.Louis, Missouri
.

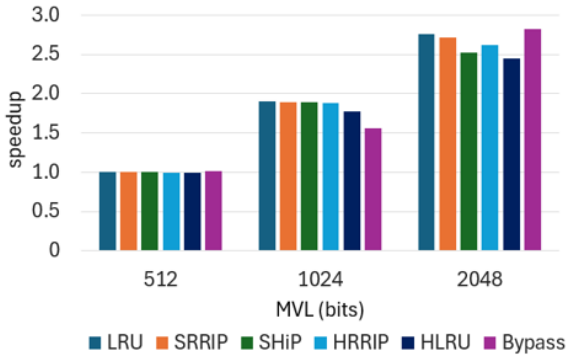


(a) For MVLs ranging from 512 bits to 2048 bits.

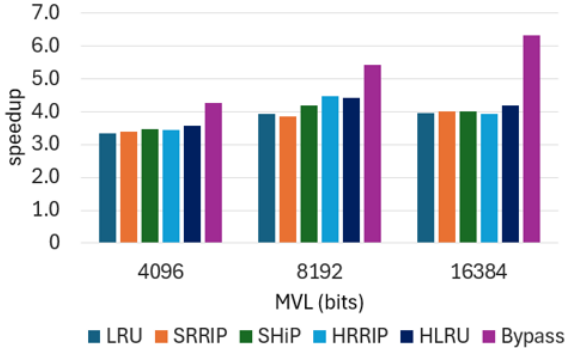


(b) For MVLs ranging from 4096 bits to 16384 bits.

Figure 1: Speedup of jacobi-2d for each MVL and cache management technique (normalized to the case of LRU with MVL = 512 bits).



(a) For MVLs ranging from 512 bits to 2048 bits.



(b) For MVLs ranging from 4096 bits to 16384 bits.

Figure 2: Speedup of aligned jacobi-2d for each MVL and cache management technique (normalized to the case of LRU with MVL = 512 bits).

Figure 1 shows the jacobi-2d results for different cache management techniques. The optimal technique depends on the MVL, with cache bypass showing the largest performance variation due to MVL-induced changes in access patterns. Since both scalar and vector instructions access the working set, changes in MVL alter the runtime vector length and the frequency of scalar accesses, thereby modifying the access pattern. Figure 2 shows the comparison results when data are aligned to cache-line boundaries. Compared to the unaligned case, the performance trends of the cache management techniques differ, indicating that alignment alters access patterns and influences the effectiveness of cache management.

Figure 3 shows the comparison results for spmv. In contrast to jacobi-2d, the performance trends across cache management techniques are more consistent. However, in the case of cache bypass, performance degradation is observed for short MVLs. This is likely because the impact of unaligned accesses varies with MVLs. Indeed, when arrays are aligned, as shown in Figure 4, the performance drop disappears. These results indicate that, for certain cache management techniques, the performance impact of unaligned accesses depends on MVL, leading to MVL-dependent performance variation.

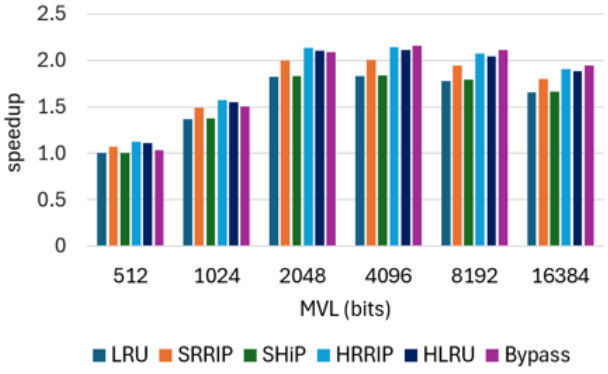


Figure 3: Speedup of spmv for each MVL and cache management technique (normalized to the case of LRU with MVL = 512 bits).

4 Conclusion

The evaluation results in this paper Our results demonstrated that the optimal cache management technique can vary depending on

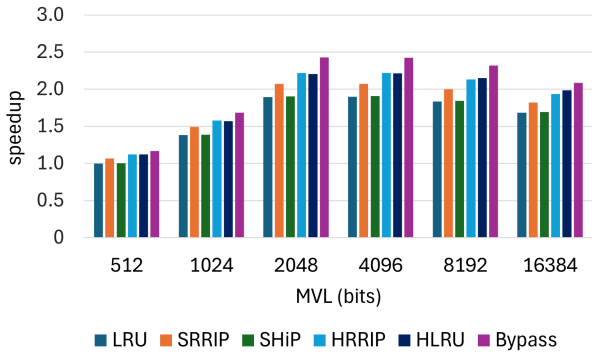


Figure 4: Speedup of aligned spmv for each MVL and cache management technique (normalized to the case of LRU with MVL = 512 bits).

MVL. This variation arises from two key factors: (1) the MVL-dependent change in the ratio between vector and scalar memory instructions, and (2) the MVL-dependent change in the impact of misalignment. These findings highlight the necessity of considering MVL, alongside application characteristics, when selecting appropriate cache management techniques. A technique that is effective on a system with a specific MVL is not necessarily effective on systems with other MVLs.

Acknowledgments

The authors acknowledge the use of ChatGPT, developed by OpenAI to assist with proofreading and enhance manuscript's clarity. The authors are accountable for contents.

References

[1] Aamer Jaleel, Kevin B. Theobald, Simon C. Steely, Jr. and Joel Emer. 2011. High Performance Cache Replacement Using Re-Reference Interval Prediction (RRIP). *ISCA '10: Proceedings of the 37th annual international symposium on Computer architecture* (2011), 60 – 71.

[2] Carole-Jean Wu, Aamer Jaleel, Will Hasenplaugh, Margaret Martonosi,Simon C. Steely Jr., Joel Emer . 2011. SHiP: Signature-based Hit Predictor for High Performance Caching. *MICRO-44: Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture* (2011), 430 – 441.

[3] Constantino Gómez, Filippo Mantovani, Erich Focht and Marc Casas. 2023. HPCG on long-vector architectures: Evaluation and optimization on NEC SX-Aurora and RISC-V. *Future Generation Computer Systems* 143 (2023), 152–162.

[4] Cristóbal Ramírez, César Alejandro Hernández, Oscar Palomar, Osman Unsal, Marco Antonio Ramírez and Adrián Cristal. 2020. A RISC-V Simulator and Benchmark Suite for Designing and Evaluating Vector Architectures. *ACM Transactions on Architecture and Code Optimization (TACO)* 17 (2020), 1–30.

[5] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, David A. Wood. 2011. The gem5 simulator. *ACM SIGARCH Computer Architecture News* 39 (2011), 1 – 7.

[6] RISC-V. 2024. The RISC-V Instruction Set Manual Volume I Unprivileged Architecture.