The Bicameral Cache: a split cache for RISC-V vector architectures

Susana Rebolledo Ruiz, Borja Pérez Pavón, Jose Luis Bosque Orero, Peter Hsu
{rebolledos, perezpavonb, bosquejl}@unican.es, peterhsu3333@gmail.com

Computer Architecture and Technology

Cofinanciado por la Unión Europea

Summary

This poster presents the design and evaluation of a cache organization for RISC-V vector processor that splits scalar and vector references into two partitions with different characteristics. The proposal, called Bicameral Cache, is specifically aimed at improving the performance of vector applications, by suppressing the interference caused by scalar instructions and arranging vector elements consecutively to guarantee their continuity. An additional prefetching option to populate vector cache lines in advance is included to improve performance, by exploiting the spatial locality of vector references.

Methodology

Evaluation criteria:

- **Bicameral Cache** performance compared to white cache (WC); a conventional scalar cache with the same organization as the Scalar Cache in BC, but double the capacity.
- **Prefetching effectiveness** compared to an ideal version that provides maximum coverage by completely filling, with no extra penalty, the whole VC line on the first compulsory miss that fetches a sector from memory.

Modelled arquitecture

CPU	RISC-V single-core in-order vector processor
DRAM	4 GB DDR RAS-CAS-PRE (11-11-28 cycles) Row-Bank-Column (15-8-3 bits) 8 banks, 1 working at a time
Bus	2 x 256 bit

Implementation on **Cavatools** [1], an open-source RISC-V ISA simulator running on a x86 Linux system. Extended to support event-driven ex ecution, to enable the modelling of timing constraints related to memory accesses.

Benchmark	Input size
`axpy` [2]	2048 KB
`blackscholes` [2]	512 opt.
`jacobi-2d` [2]	32 el.
`pathfinder` [2]	1024x128 el.
`mv`	4096x4096 el.

mv: matrix-vector multiplication, opt.: options, el.: elements.

Architectural VLENs: {256, 512, 1024, 2048} bits.

Conclusions

The Bicameral Cache improves performance on stride-1 benchmarks. The average best-case speedup is 1.31x over the baseline, and up to 1.57x if prefetching is enabled. For non-stride-1 workloads, performance remains mostly unchanged with the base configuration, improving by 11% with prefetching.
These results are driven by a substantial reduction in the average memory access time and are achieved just by restructuring the available resources to better exploit locality. Finally, prefetching proved to be a succesful optimization.

The Bicameral Cache (BC)

The Bicameral Cache (BC) is a cache design for vector architectures, composed by 2 data caches with different structures and geometries; the Scalar Cache (SC) and the Vector Cache (VC).

Features:

- **Sectorized:** lines are divided in sectors (min. data transfer unit).
- **Mutual exclusivity:** *cross lookup* needed after miss on *native lookup*.
- **Vector migration:** sector migrates from SC to VC on vector *cross hit*.
- **Vector prefetching** to prefill VC lines.

Objectives:

- Preserving the locality of each type of data, preventing scalar data from interfering with vector and vice versa.
- Exploiting the spatial locality of vector accesses (long VC lines and prefetching).
- Favoring the performance of vector instructions by prioritising the presence of vector data in the VC (scalar accesses can reference them maintaining continuity).

Scalar Cache (SC)

Data referenced by *scalar memory instructions*
64KB, 4-way set-associative, 64 B lines (1 sector), 8-line write buffer.

Set	Line	Sector
0	0	
	1	
	2	
	3	
⋮	⋮	⋮
255	0	
	1	
	2	
	3	

WB0	
WB1	
WB2	
WB3	
WB4	
WB5	
WB6	
WB7	

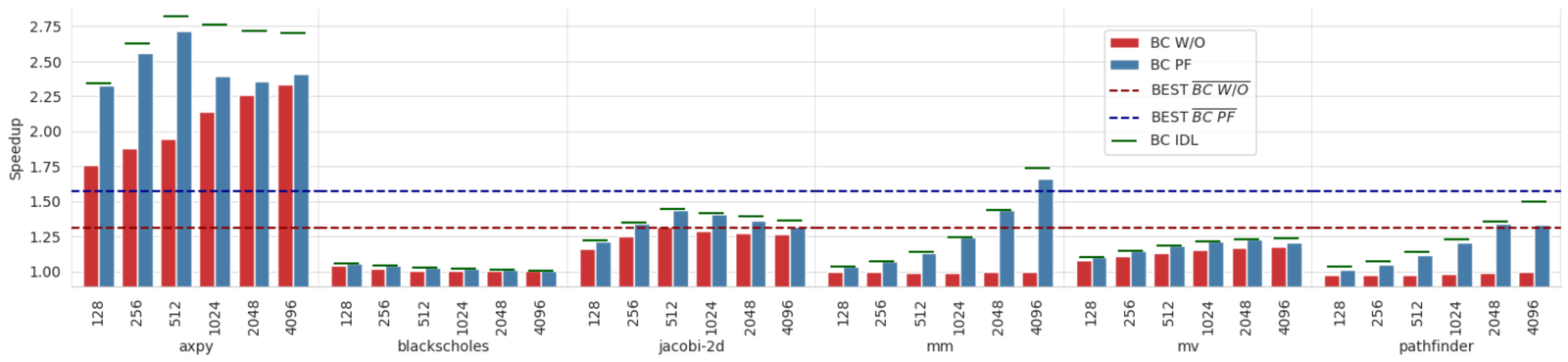
Vector Cache (VC)

Data referenced by *vector memory instructions*
64KB, fully associative, 1024 B long lines (16 sectors), 8-line dynamic, embedded write buffer.

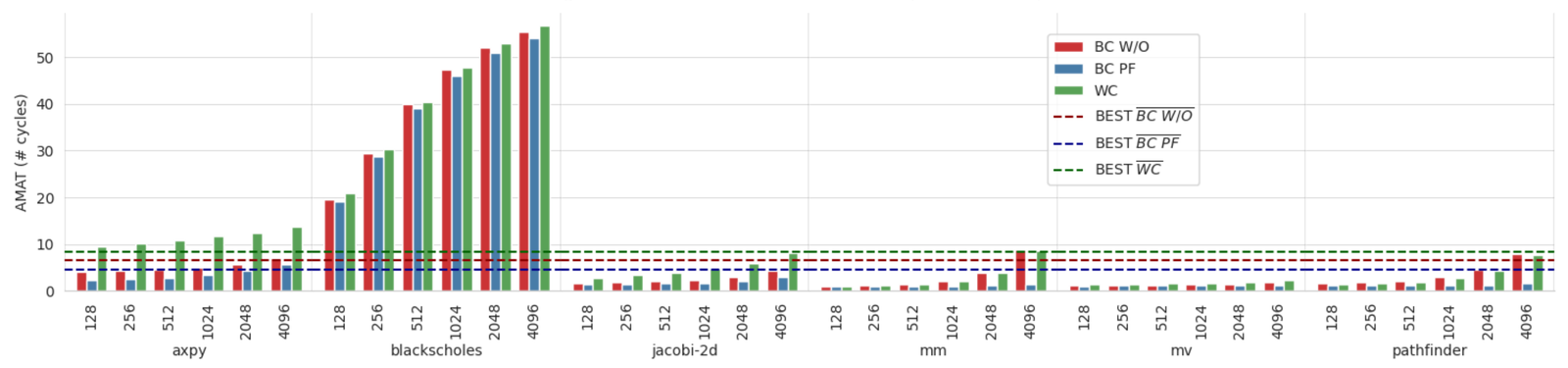
	Sector				
Line	0	1	···	14	15
0			···		
⋮			⋮		
63			···		
WB0			···		
WB1			···		
WB2			···		
WB3			···		
WB4			···		
WB5			···		
WB6			···		
WB7			···		

Results

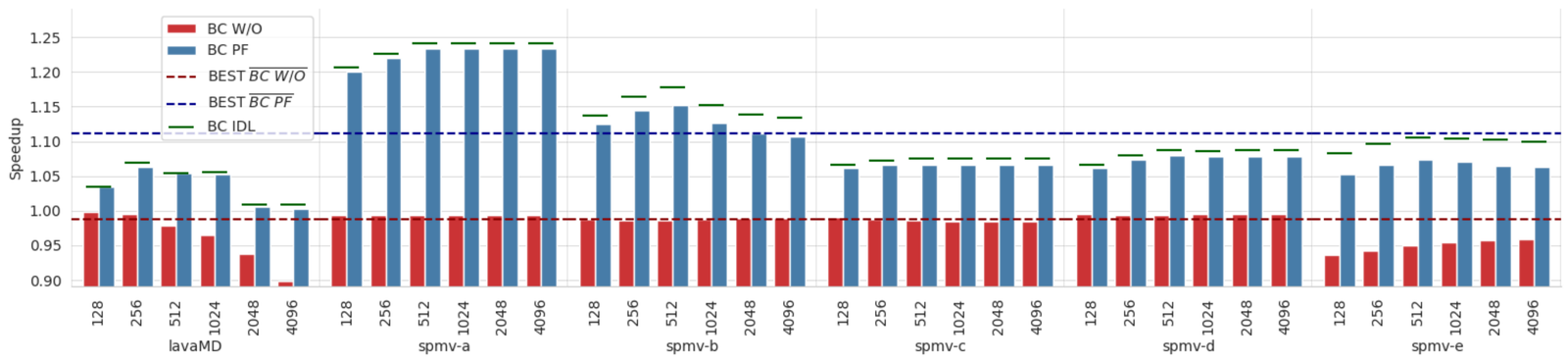
Speedup (Stride-1)



Avg. Mem. Access Time (Stride-1)



Speedup (Non-stride-1)



BC: Bicameral Cache, W/O: without prefetching, PF: with prefetching, IDL: with ideal prefetching.

Acknowledgements

This work has been funded by MICIU/AEI/10.13039/501100011033 together with FEDER, UE under project PID2022-136454NB-C21 and grant PREP2022-000053, and with EU/NextGenerationEU/PRTR under project TED2021-131176B-I00.

References

[1] P. Hsu, *Cavatools*, GitHub.
[2] C. Ramírez *et al.*, "A RISC-V simulator and benchmark suite for designing and evaluating vector architectures," *ACM Trans. Archit. Code Optim.*, vol. 17, no. 4, Nov. 2020, ISSN: 1544-3566.