

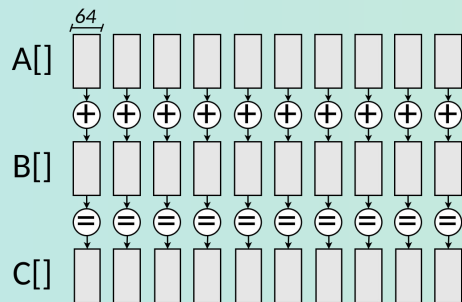
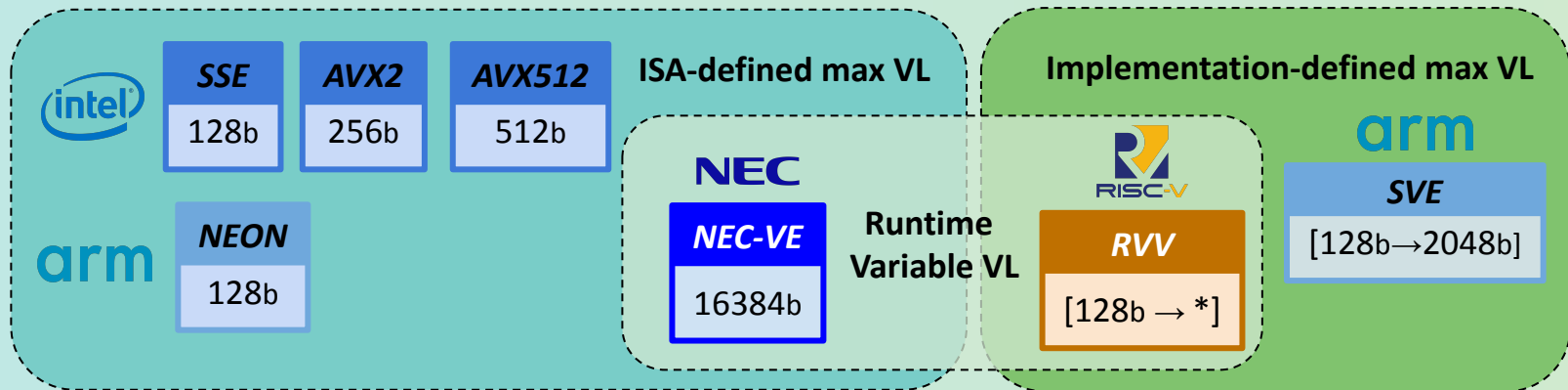
# Short reasons for long vectors in HPC CPUs

A study based on RISC-V

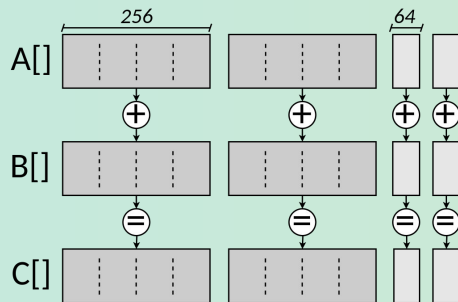
Pablo Vizcaino, Giorgos Ieronymakis, Nikolaos Dimou, Vassilis Papaefstathiou, Jesus Labarta, Filippo Mantovani

# 1.1- Context: Vector computing in HPC

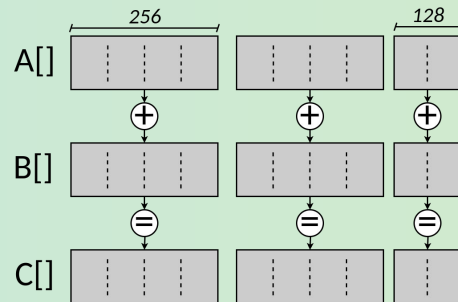
- Most modern architectures have SIMD extensions:



Scalar processor



SIMD (e.g., AVX2)



Variable VL (e.g., RVV)

# 1.2- Context: Long Vector Lengths (VL)

SSE, NEON: **128b**

AVX2: **256b**

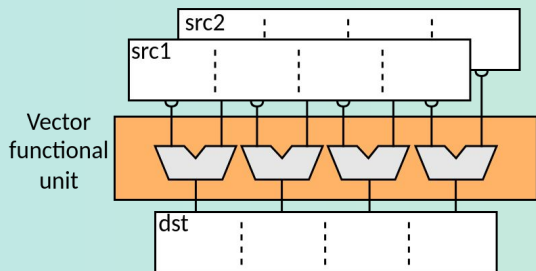
AVX512: **512b**

SVE: **2048b**

NEC, RVV: **16384b**

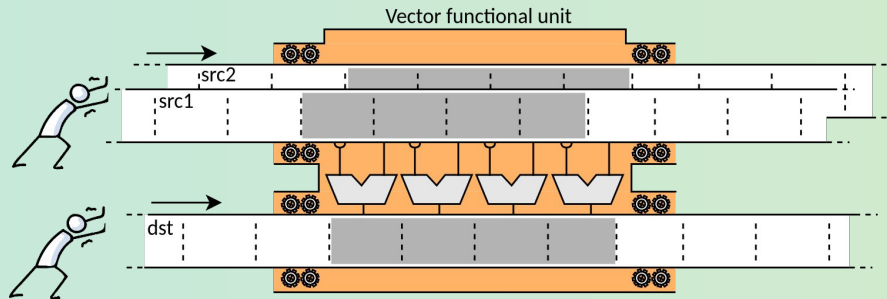
## Short VL

- As many Functional Units as VL.
- Vector instructions executed in 1 cycle



## Long VL

- Cannot afford (area, power, cost) hundreds of Functional Units
- Vector instructions are executed on multiple cycles



## 1.2- Context: Long Vector Lengths (VL)

Why do we design VPUs with  $VL > \text{Num. Functional Units}$ ?

### Reduce the number of instructions

- Fewer arith/mem instructions (*vectorized*)
- Fewer loop control instructions (*removed*)

### Instantiate more work with fewer instructions

- Reduce pressure on CPU front-end.
- More resistance to stalls (e.g. branch miss predictions).

In this paper we focus on...

**Mitigating memory latency**

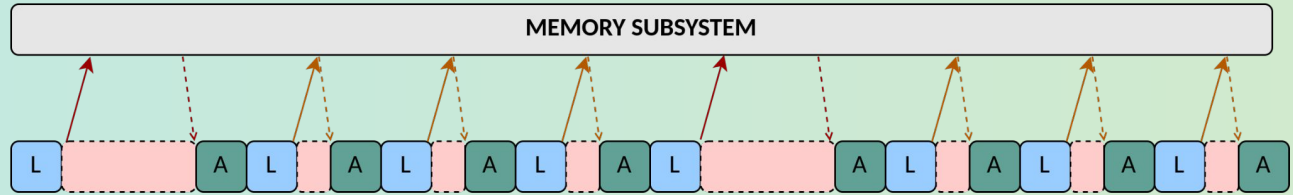


## 2.1- Long vectors hiding memory latency

- Basic loop structure  $\rightarrow$   $\text{for } i=0:8 \left\{ \begin{array}{l} \text{LOAD}[i] \rightarrow \text{ARITH} \end{array} \right\}$

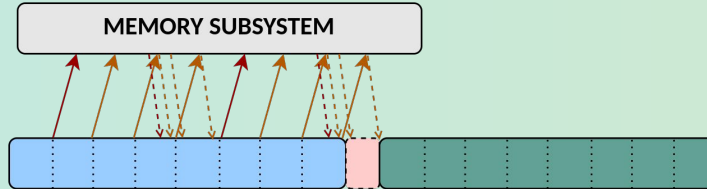
- Simple scalar pipeline:**

- Pipeline stalls
- Pay all the latencies

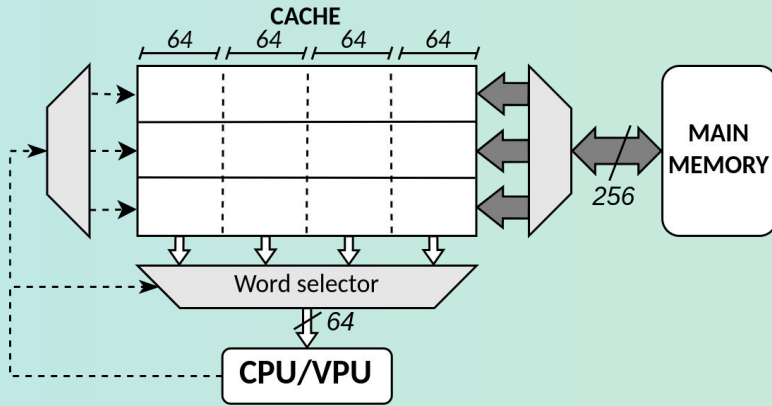


- Long vector pipeline**

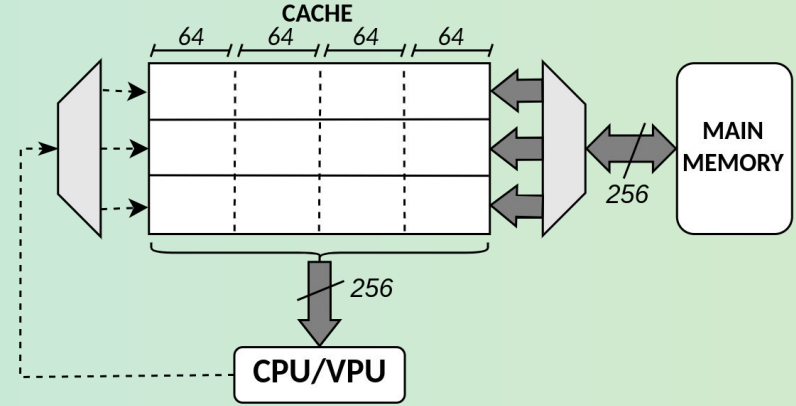
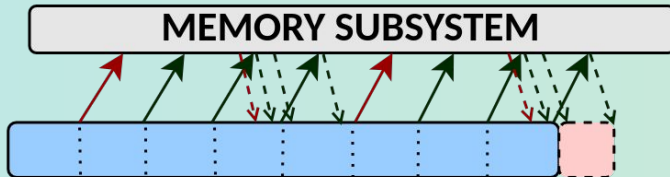
- Send request after request
- Overlap latencies



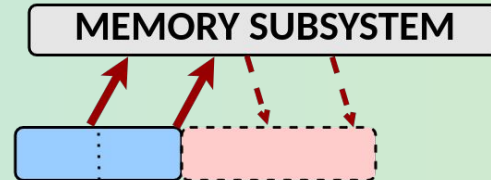
## 2.2- Long vectors are hungry for bandwidth



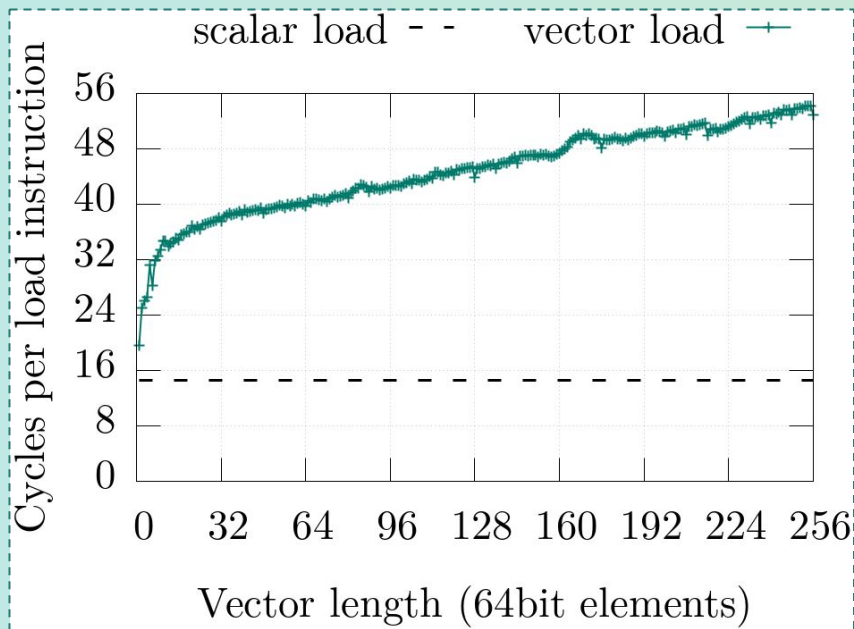
- Normally, the CPU ↔ CACHE bus is 64B wide.
- But cache is indexed at line granularity.
- 8 consecutive accesses → 8 petitions (to 2 lines)



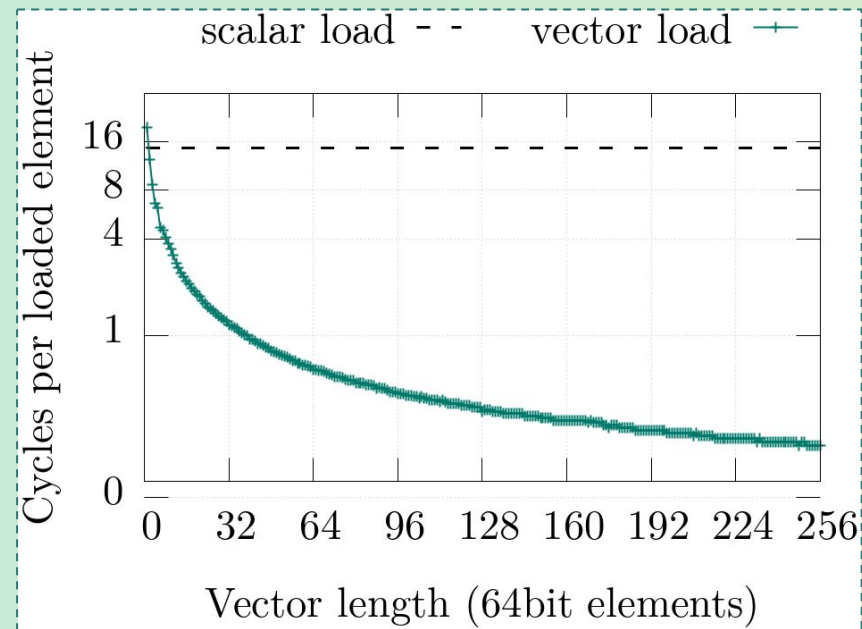
- Vector memory instructions generate dozens of consecutive accesses.
- Making the CPU ↔ CACHE bus wider reduces the amount of required petitions (e.g. from 8 to 2)



## 2.3- Experimental results of long vectors




Instructions with higher VL take longer to execute...

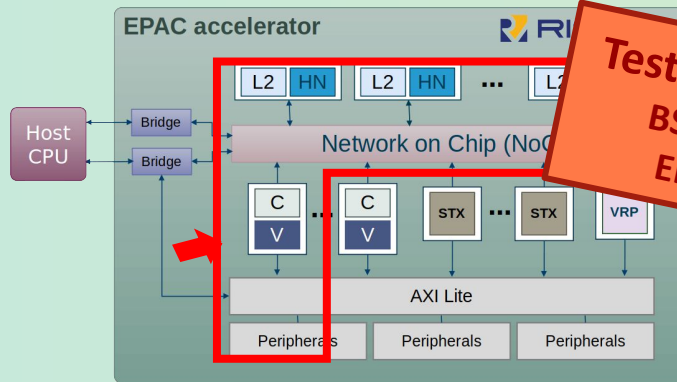


... But are more efficient on a per-element basis!



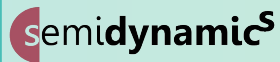
# 3- Experimental Setup

- European Processor Initiative (EPI) 
- European Processor Accelerator (EPAC)
  - RISC-V based
  - **VEC**, STX, VRP, ...



**Test-chip available at:**  
 BSC booth - #1269  
 EPI booth - #213


## Avispado (CPU)



- In-order.
- Implements the rvv0.7 extension
- 32KB data cache
- Gazillion™ misses

## Vitruvius [1] (VPU)



- 16384 bits per register
- Implements the rvv0.7 extension
- 8 Lanes → 16Flop/cycle
- FPU [2] 

## L2 Homenode



- 4 Homenodes of 256KB → 1MB L2
- MESI-based coherence

## Network On Chip (NoC)



[1] Francesco Minervini et al. 2023. Vitruvius+: An Area-Efficient RISC-V Decoupled Vector Coprocessor for High Performance Computing Applications. ACM Transactions on Architecture and Code Optimization 20, 2 (2023), 1–25.

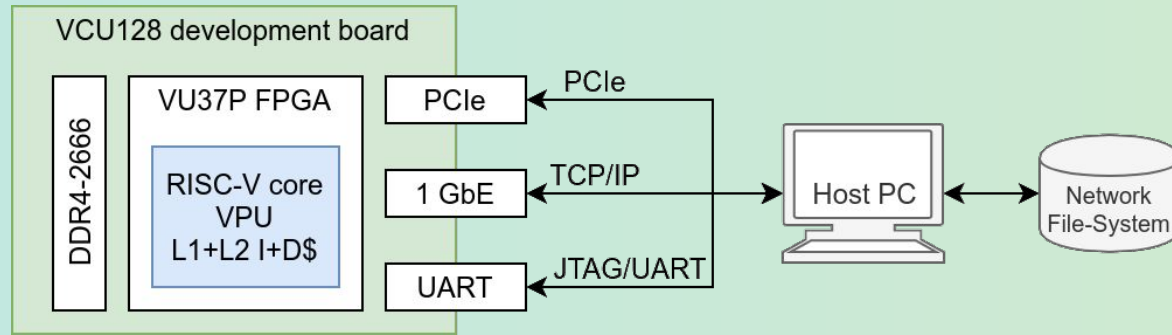
[2] Mate Kovač et al. 2023. FAUST: design and implementation of a pipelined RISC-V vector floating-point unit. Microprocessors and Microsystems (2023), 104762





# 3- Experimental Setup: Software Development Vehicles

- In BSC we have a cluster of **FPGA-based Software Development Vehicles (FPGA-SDV) [3]** nodes:
  - **EPAC/VEC** RTL (*CPU, VPU, L2HN, NoC*) mapped into an FPGA, running at 50MHz.
  - Full HPC software stack, NFS, performance analysis → Perfect environment for *HW-SW codesign*.



[3] Filippo Mantovani et al. (2023, May). Software Development Vehicles to enable extended and early co-design: a RISC-V and HPC case of study. In International Conference on High Performance Computing (pp. 526-537). Cham: Springer Nature Switzerland.

# 3- Experimental Setup

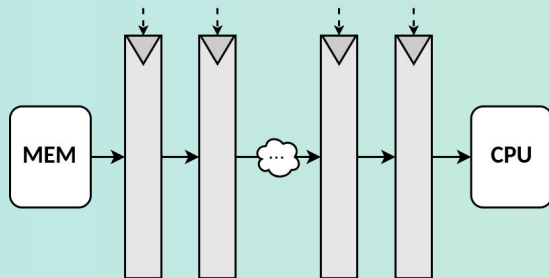
- The FPGA environment allows to easily change hardware parameters:

## Latency Controller



Hardware module:

- Artificially add memory latency.
- Can be changed on runtime
- Emulate a slower memory

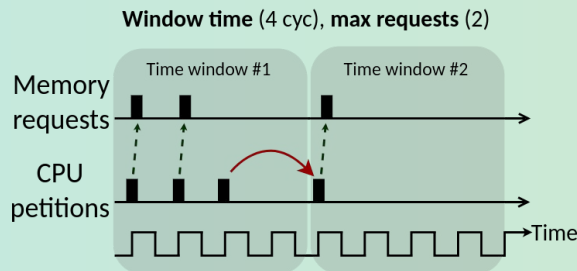


## Bandwidth Limiter



Hardware module:

- Throttle memory bandwidth
- Can be changed on runtime
- Generate fake contention



## Max. Vector-Length



Custom CSR

- Can be changed on runtime
- Same binary, different max VL
- Study the effect that a smaller VPU would have

## 4- Codes selected for the evaluation

- Four in-house vectorized kernels targeting RVV:
  - **SpMV[4]**: Sparse matrix-vector multiplication → Used in HPCG (Top500)
  - **BFS[5]**: Breadth-First-Search → Building block of many graph algorithms
  - **PR[5]**: Page-Rank → Used by Google to rank webpages
  - **FFT[6]**: Fast Fourier Transform → Many scientific applications
- The inputs have been selected as to exceed the caches capacities.

[4] Constantino Gómez Crespo, et.al. 2020. *Optimizing sparse matrix-vector multiplication in NEC SX-Aurora vector engine*. (2020).

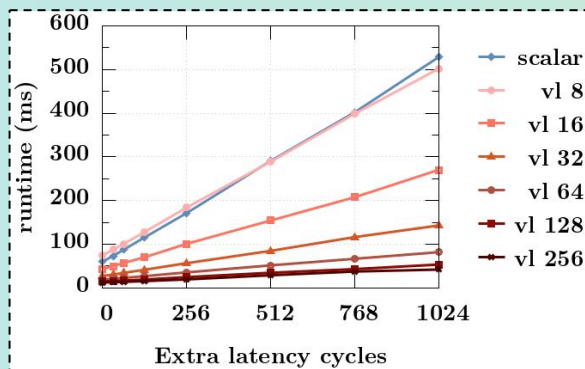
[5] Pablo Vizcaino. 2023. *Implementing and evaluating graph algorithms for long vector architectures*. Master's thesis. Universitat Politècnica de Catalunya.

[6] Pablo Vizcaino et al. 2022. *Acceleration with long vector architectures: Implementation and evaluation of the FFT kernel on NEC SX-Aurora and RISC-V vector extension*. *Concurrency and Computation: Practice and Experience* (2022), e7424.

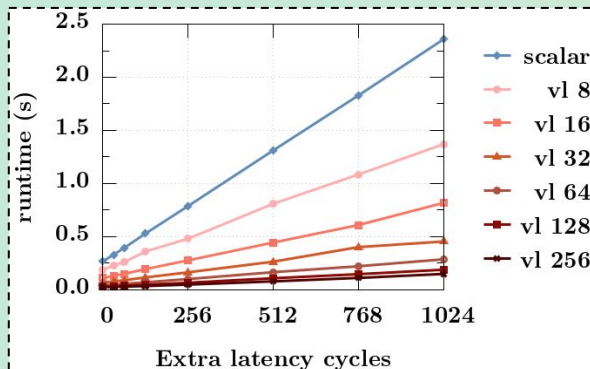


## 5- Adding latency

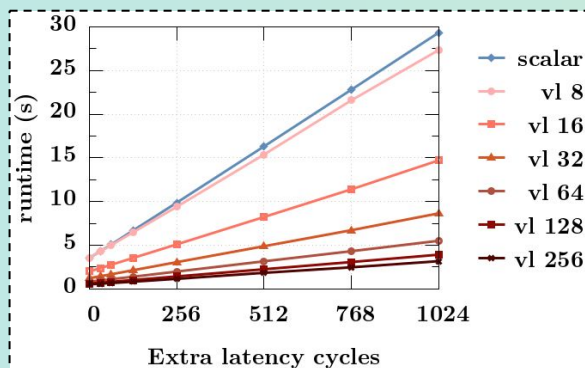
SpMV



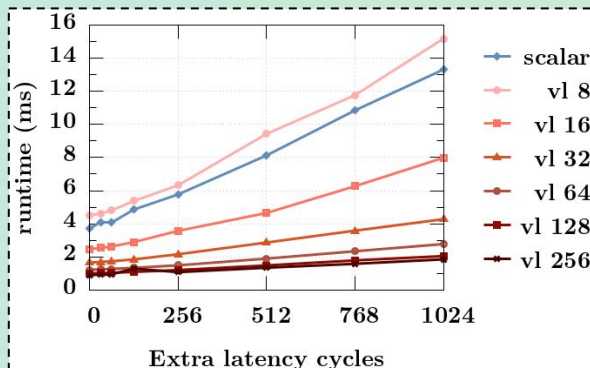
BFS



PR



FFT



### Observations

- Execution time depending on VL and extra latency
- Lower is better
- The four codes benefit from long VLs
- Long VL present flatter curves: latency resistance



# 5- Adding latency (Relative slowdown)

SpMV							
Extra latency	0	32	64	128	256	512	768
0	1.00	1.00	1.00	1.00	1.00	1.00	1.00
32	1.22	1.17	1.15	1.10	1.10	1.07	1.05
64	1.45	1.35	1.32	1.24	1.20	1.13	1.12
128	1.92	1.71	1.63	1.50	1.42	1.29	1.26
256	2.85	2.46	2.34	2.05	1.88	1.67	1.57
512	4.83	3.86	3.57	3.06	2.72	2.33	2.34
768	6.68	5.32	4.81	4.20	3.52	2.90	3.03
1024	8.78	6.70	6.26	5.16	4.32	3.61	3.39
	scalar	vl-8	vl-16	vl-32	vl-64	vl-128	vl-256

PR							
Extra latency	0	32	64	128	256	512	768
0	1.00	1.00	1.00	1.00	1.00	1.00	1.00
32	1.22	1.20	1.18	1.19	1.17	1.15	1.13
64	1.44	1.40	1.38	1.37	1.34	1.32	1.27
128	1.89	1.81	1.76	1.75	1.69	1.64	1.57
256	2.79	2.63	2.53	2.50	2.40	2.30	2.20
512	4.61	4.30	4.09	3.99	3.83	3.62	3.51
768	6.44	6.04	5.66	5.48	5.22	4.95	4.70
1024	8.28	7.65	7.31	7.05	6.67	6.30	6.04
	scalar	vl-8	vl-16	vl-32	vl-64	vl-128	vl-256

BFS

BFS							
Extra latency	0	32	64	128	256	512	768
0	1.00	1.00	1.00	1.00	1.00	1.00	1.00
32	1.23	1.19	1.20	1.17	1.12	1.08	1.07
64	1.47	1.37	1.37	1.33	1.32	1.27	1.19
128	1.98	1.90	1.78	1.72	1.65	1.59	1.43
256	2.94	2.53	2.53	2.44	2.38	2.19	1.97
512	4.90	4.27	4.06	3.94	3.89	3.59	3.20
768	6.83	5.72	5.56	5.97	5.20	4.84	4.50
1024	8.80	7.23	7.47	6.76	6.73	6.24	6.03
	scalar	vl-8	vl-16	vl-32	vl-64	vl-128	vl-256

FFT

FFT							
Extra latency	0	32	64	128	256	512	768
0	1.00	1.00	1.00	1.00	1.00	1.00	1.00
32	1.11	1.03	1.03	1.02	1.02	1.02	1.01
64	1.10	1.07	1.06	1.05	1.04	1.04	1.04
128	1.31	1.20	1.17	1.12	1.09	1.09	1.09
256	1.56	1.49	1.44	1.31	1.22	1.20	1.19
512	2.20	2.23	1.88	1.74	1.56	1.47	1.49
768	2.93	2.78	2.54	2.17	1.92	1.78	1.74
1024	3.60	3.59	3.23	2.59	2.28	2.05	2.03
	scalar	vl-8	vl-16	vl-32	vl-64	vl-128	vl-256

## Observations

- Execution time normalized to 0 added latency for each VL (slowdown)

- Greener is better

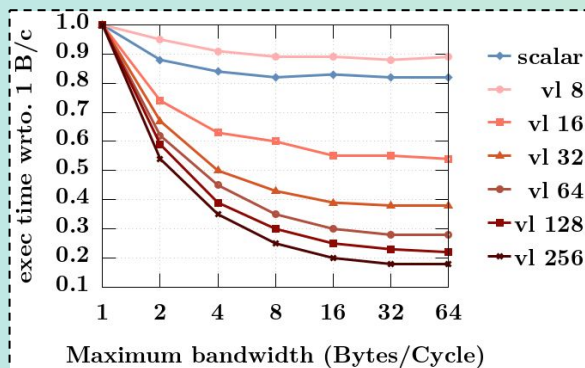
- Added latency slows us

- VL mitigates it

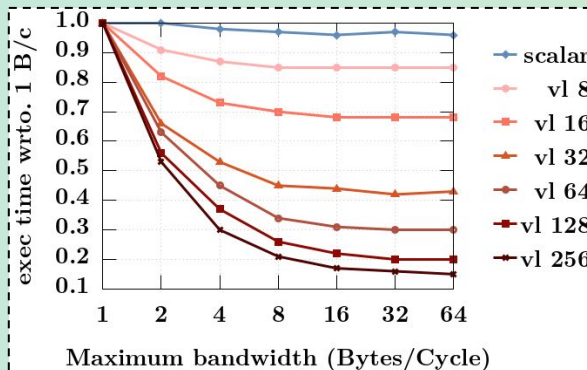


## 6- Throttling bandwidth

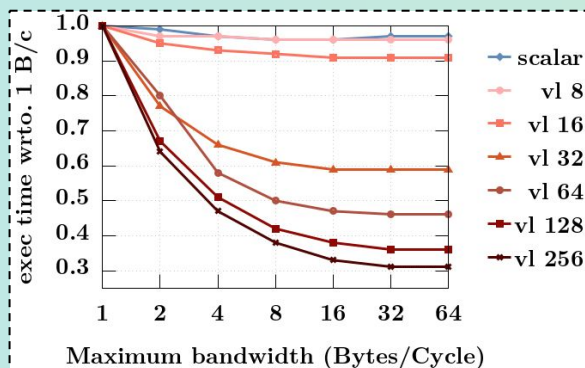
SpMV



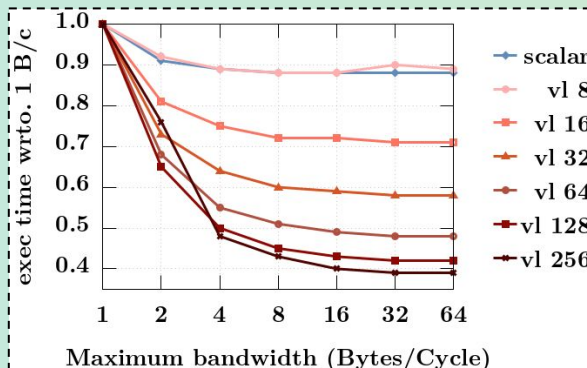
BFS



PR



FFT



### Observations

- Execution time normalized for each VL to BW=1B/c
- Lower is better
- Scalar plateaus early (2B/c)
- Long VL takes advantage of higher bandwidths



# 7- Conclusions

## Our short reasons for long vectors:

- Vectorized implementations are less impaired by memory latency than their scalar counterparts.
- The latency mitigation is stronger the longer the vector length is.
- Vector architectures benefit from high bandwidth systems without needing to increase the core count.

## The SDV methodology:

- Is effective to rapidly test these arguments.
- It allows to test complex workloads and see the effect of vectorization on them.

## In the future we will:

- Expand this study to other applications.
- Study the combined effects of multicore+vector.





# Acknowledgment



This research has received funding from the European High Performance Computing Joint Undertaking (JU) under Framework Partnership Agreement No 800928 (European Processor Initiative) and Specific Grant Agreement No 101036168 (EPI SGA2). The JU receives support from the European Union's Horizon 2020 research and innovation programme and from Croatia, France, Germany, Greece, Italy, Netherlands, Portugal, Spain, Sweden, and Switzerland. The EPI-SGA2 project, PCI2022-132935 is also co-funded by MCIN/AEI /10.13039/501100011033 and by the UE NextGenerationEU/PRTR.



Financed par



We thankfully acknowledge the support of the European Commission via the Horizon Europe research and innovation funding programme, under grant agreement 101092993 (RISER).



**Contact me:** [pablo.vizcaino@bsc.es](mailto:pablo.vizcaino@bsc.es)  
**Visit me:** BSC booth (#1269) or EPI booth (#213)

