NOC-BASED SUPPORT OF HETEROGENEOUS CACHE-COHERENCE MODELS FOR ACCELERATORS

Davide Giri, Paolo Mantovani, Luca P. Carloni
Columbia University, New York, USA

ACM/IEEE NOCS 2018, Torino, Italy

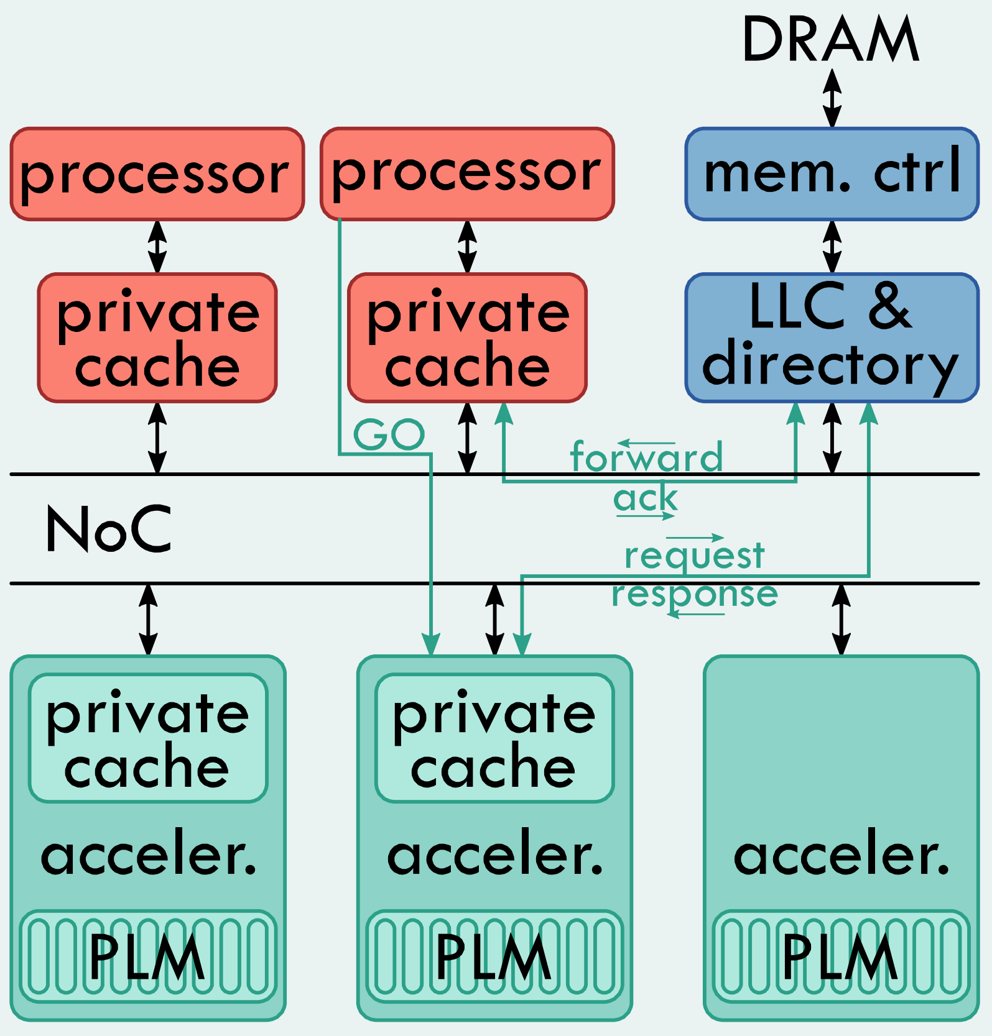
CACHE-COHERENCE MODELS FOR ACCELERATORS

Fully-Coherent

Coherent with caches

Requires

- race free accelerator execution

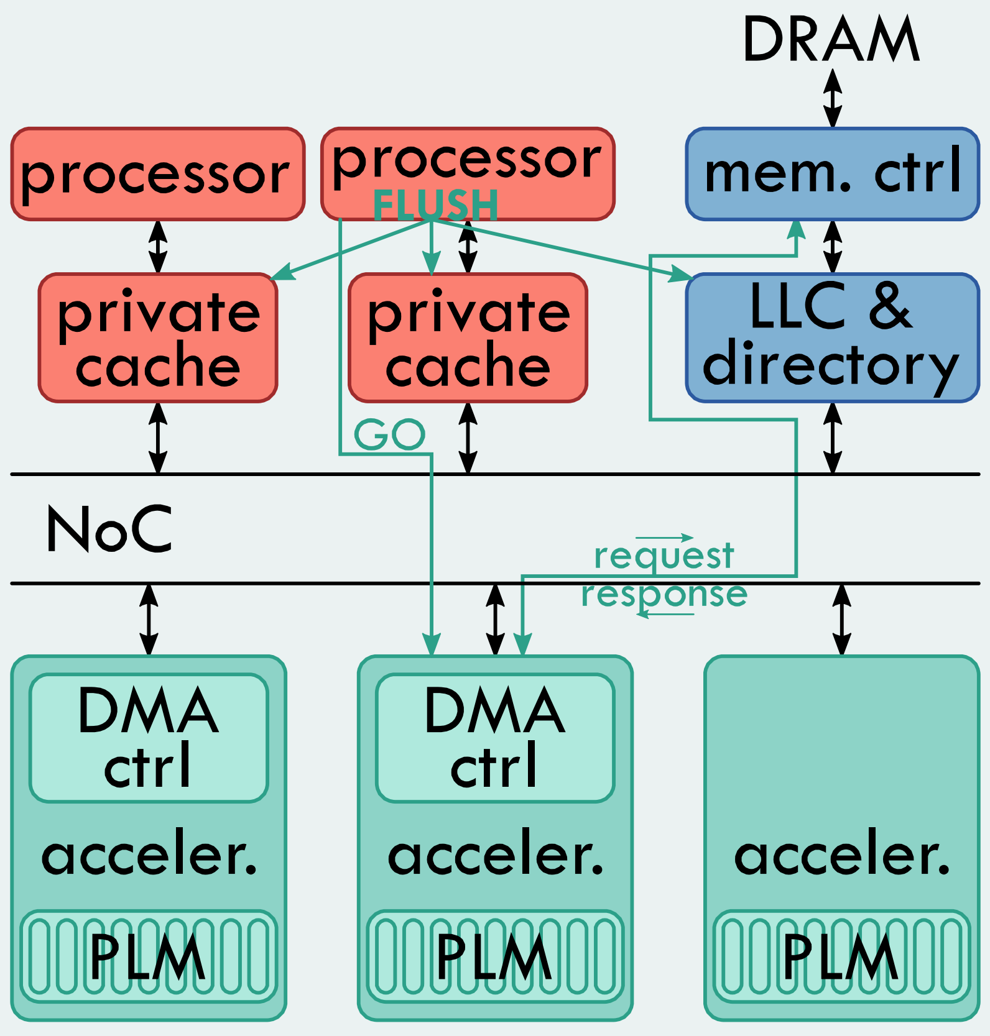


Non-Coherent

Not coherent with caches

Requires

- race free accelerator execution
- flush caches

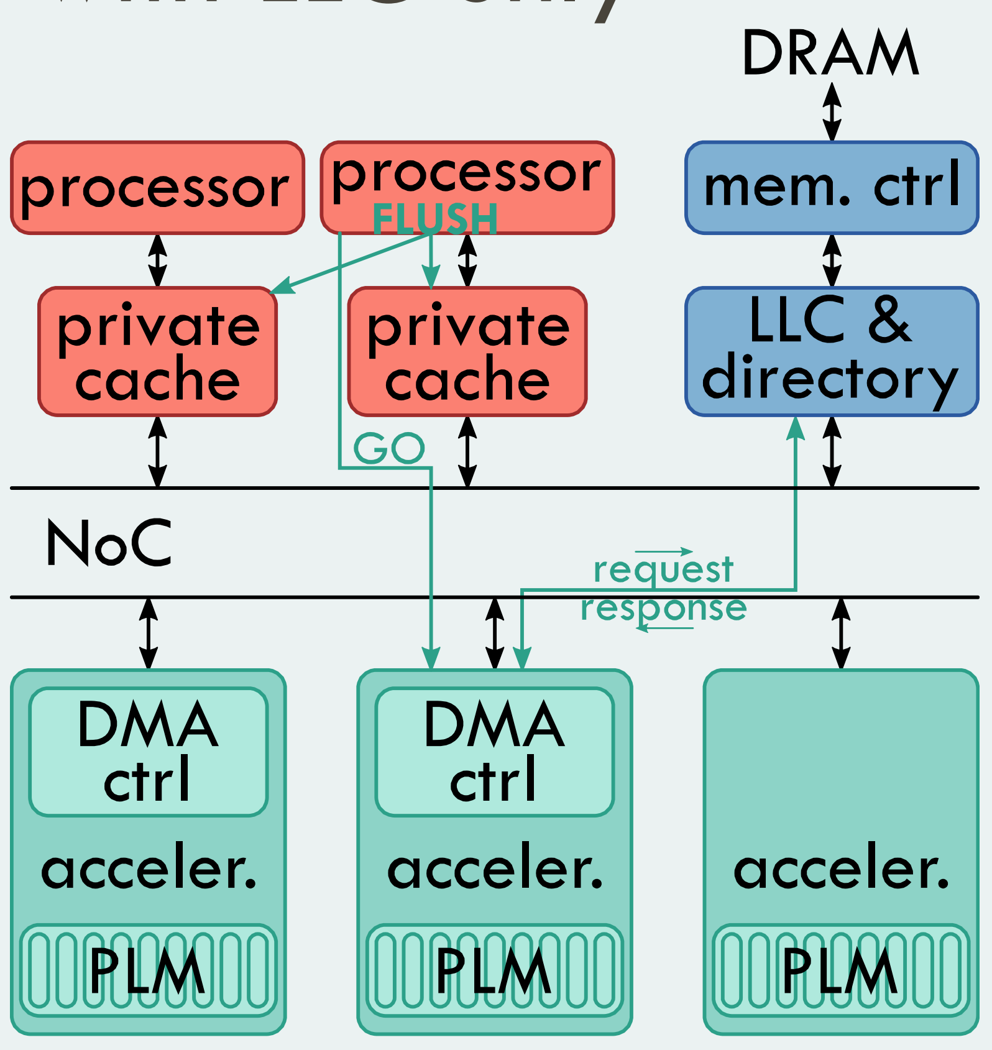


LLC-Coherent

Coherent with LLC only

Requires

- race free accelerator execution
- flush private caches



CONTRIBUTIONS

Protocol.

○ Variation of MESI to support 3 coherence models (NoC-based)

Coherence Models.

○ Show how each model can outperform the others

○ Show that the best choice of model varies at runtime

Architecture. Design of a multi-core NoC-based architecture supporting:

○ Three models of coherence for accelerators

○ Run-time selection of the coherence model for each accelerator

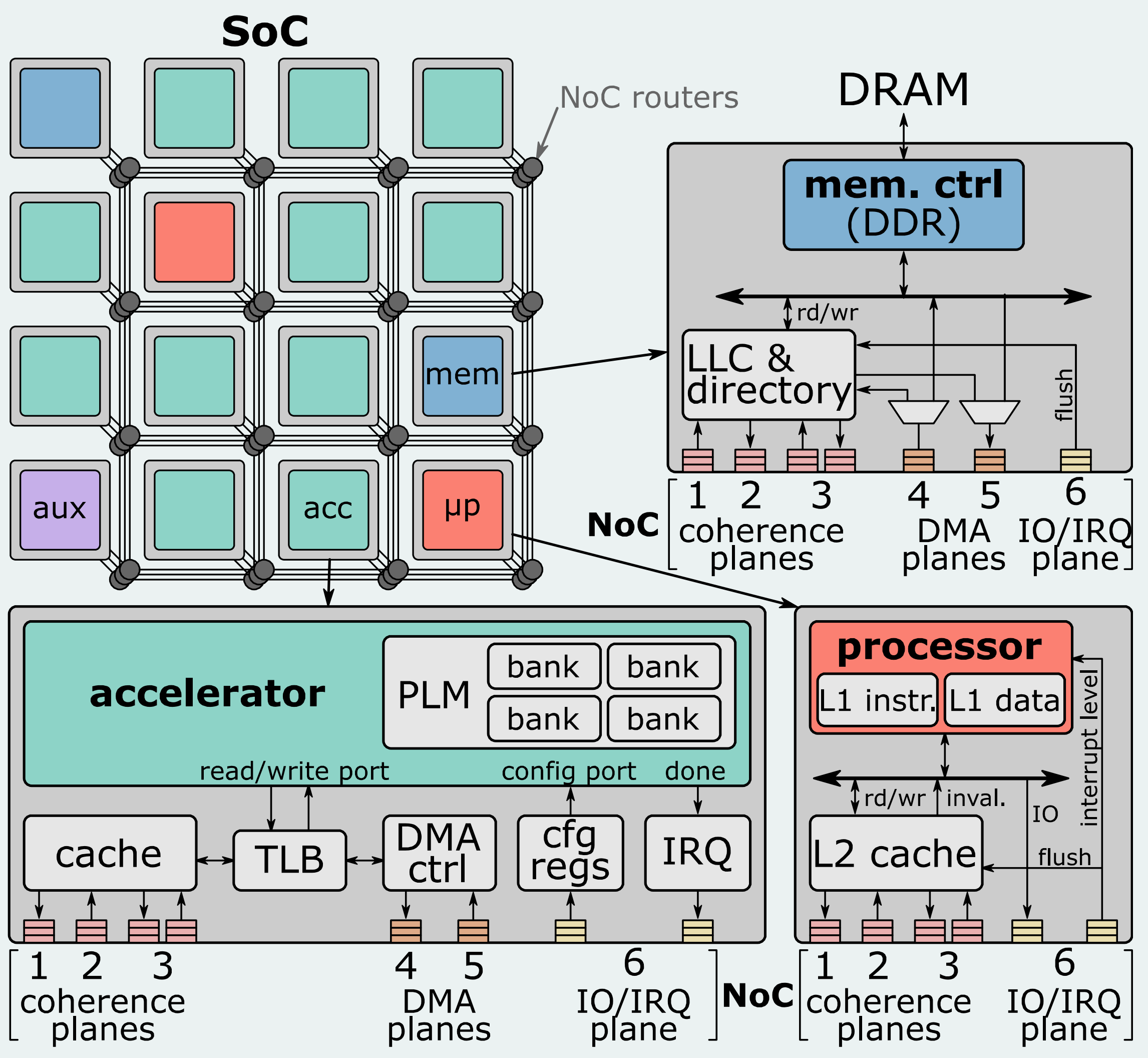
○ Coexistence of heterogeneous coherence models for accelerators

OUR SOC

Our design is based on an instance of **Embedded Scalable Platforms (ESP)** [L. P. Carloni, DAC '16]

We added a cache hierarchy to ESP. Now it can run multi-processor and multi-accelerator applications on Linux SMP.

The accelerator tile supports run-time selection of coherence model.



OUR PROTOCOL

We modified a classic MESI directory-based cache-coherence protocol

○ to make it work over a NoC

○ to support all coherence models for accelerators

Directory controller

Write-back (add a *Valid* state and dirty bit), Recalls, Flush, LLC-coherent read/write requests.

EXCERPT	LLC-coherent Read	LLC-coherent Write
Invalid	Read memory Send data to requestor Go to Valid state	Read memory if misaligned Write to LLC Go to Valid state
Valid	Send data to requestor	Write to LLC
Shared	-	-
Exclusive	-	-
Modified	-	-

Private cache controller

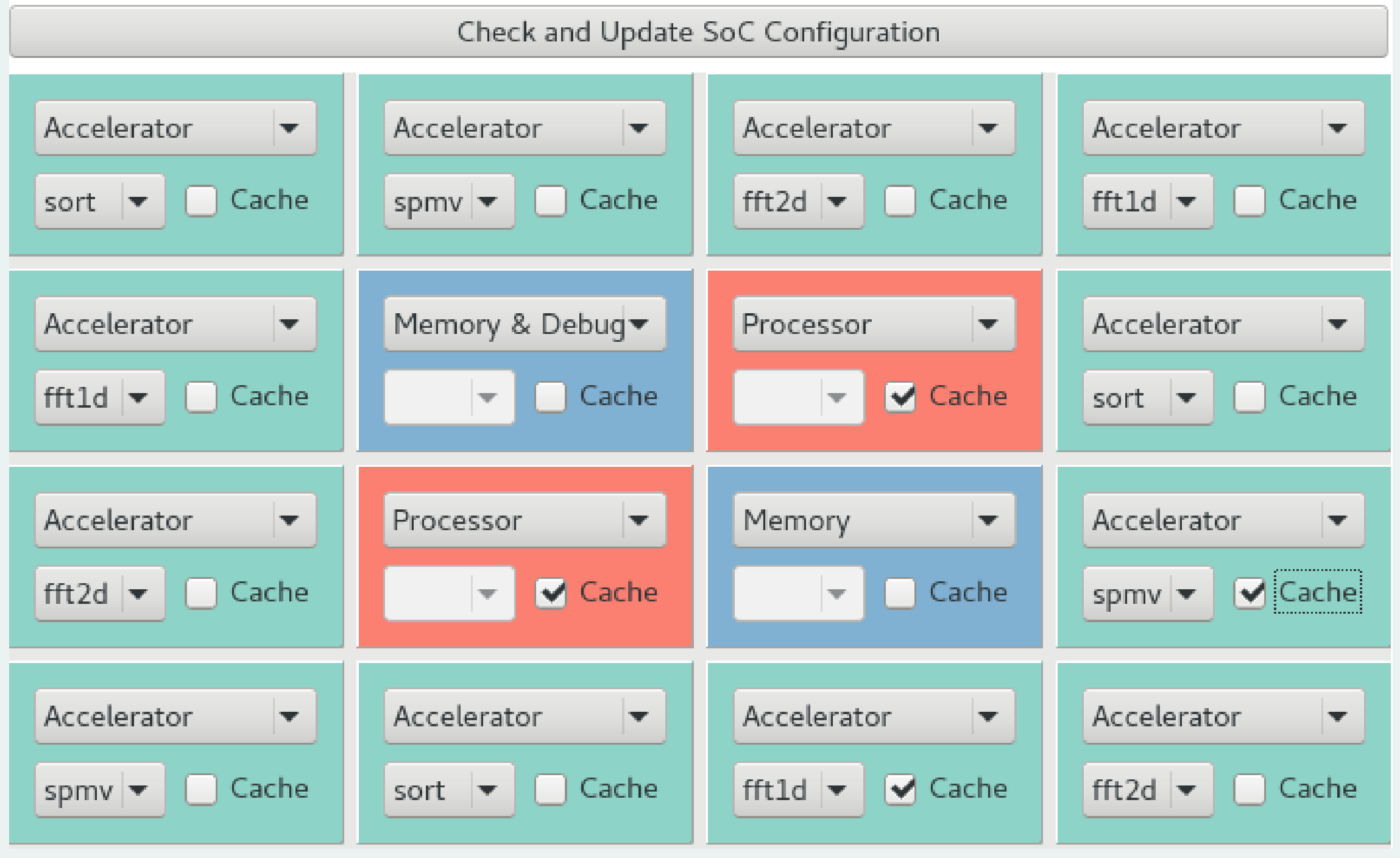
L1 invalidation, Recalls, Flush, Atomic operations.

RESULTS OVERVIEW

We designed **4 custom accelerators**: Sort, Sparse Matrix-Vector Multiplication, FFT-1D and FFT-2D.

We deployed our SoC on FPGA and we executed applications on Linux SMP.

ESP's GUI. The CAD flow from GUI to bitstream is fully automated.

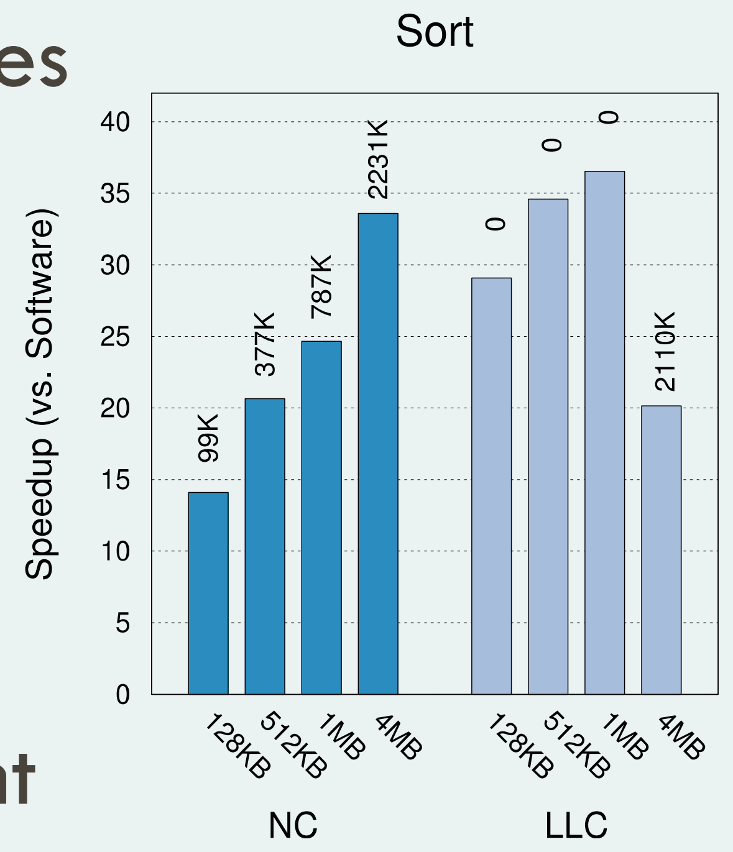

Check and Update SoC Configuration

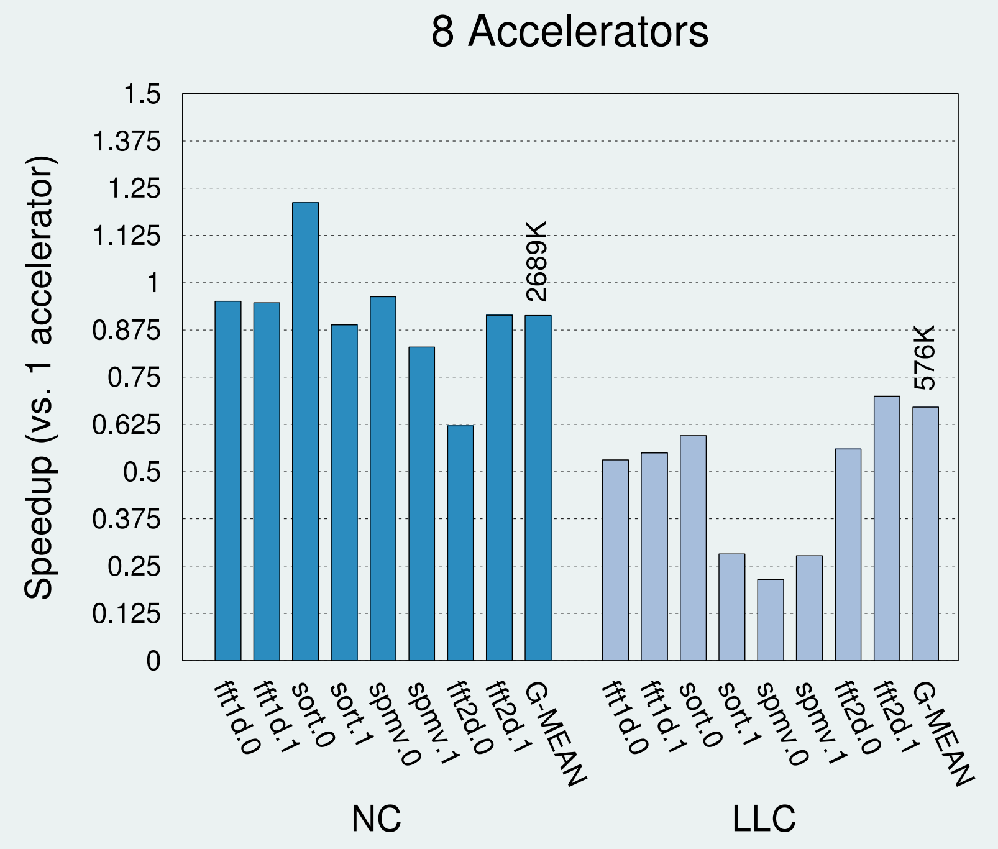
Accelerator sort ☐ Cache	Accelerator spmv ☐ Cache	Accelerator fft2d ☐ Cache	Accelerator fft1d ☐ Cache
Accelerator fft1d ☐ Cache	Memory & Debug ☐ Cache	Processor ☑ Cache	Accelerator sort ☐ Cache
Accelerator fft2d ☐ Cache	Processor ☑ Cache	Memory ☐ Cache	Accelerator spmv ☑ Cache
Accelerator spmv ☐ Cache	Accelerator sort ☐ Cache	Accelerator fft1d ☑ Cache	Accelerator fft2d ☐ Cache

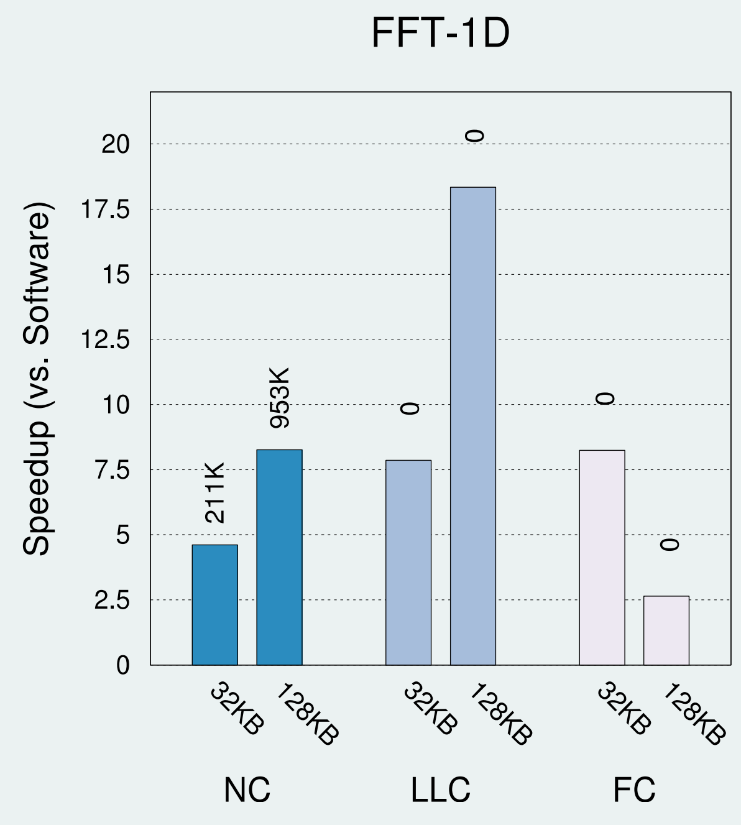
The best coherence model varies with the accelerator **workload size** and with the **number of active accelerators** in the system.

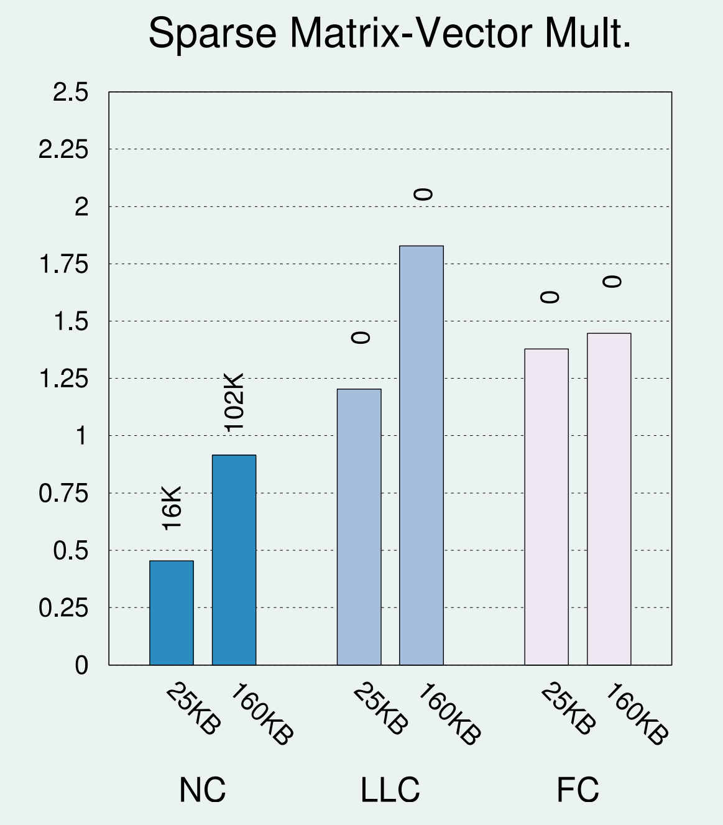
LLC-coherent and fully-coherent models can significantly **reduce accesses to DRAM.**

There is no absolute winner among the coherence models, **the best choice can vary at runtime.**


Sort
Speedup (vs. Software)
99K
377K
787K
2231K
2110K
128KB
512KB
1MB
4MB
128KB
512KB
1MB
4MB
NC
LLC


8 Accelerators
Speedup (vs. 1 accelerator)
2689K
576K
fft1d.0
fft1d.1
sort.0
sort.1
spmv.0
spmv.1
fft2d.0
fft2d.1
G-MEAN
fft1d.0
fft1d.1
sort.0
sort.1
spmv.0
spmv.1
fft2d.0
fft2d.1
G-MEAN
NC
LLC


FFT-1D
Speedup (vs. Software)
211K
953K
32KB
128KB
32KB
128KB
32KB
128KB
NC
LLC
FC


Sparse Matrix-Vector Mult.
16K
102K
25KB
160KB
25KB
160KB
25KB
160KB
NC
LLC
FC