Erlangen Regional Computing Center

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

The A64FX processor :

Understanding streaming kernels and sparse matrix-vector multiplication

webinar

Christie Alappat, Jan Laukemann, Thomas Gruber, Georg Hager, Gerhard Wellein, Nils Meyer, Tilo Wettig

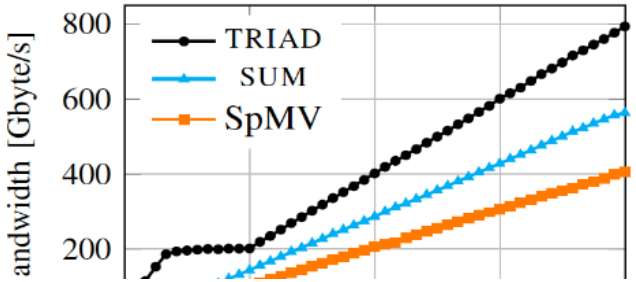
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
University of Regensburg, Germany

Performance Modeling of Streaming Kernels and Sparse Matrix-Vector Multiplication on A64FX

Christie Alappat, Jan Laukemann, Thomas Gruber,
Georg Hager, Gerhard Wellein
Erlangen Regional Computing Center
Friedrich-Alexander-Universität Erlangen-Nürnberg
Erlangen, Germany
christie.alappat@fau.de

Nils Meyer, Tilo Wettig
Department of Physics
University of Regensburg
Regensburg, Germany
tilo.wettig@ur.de

***Abstract*—The A64FX CPU powers the current #1 supercomputer on the Top500 list. Although it is a traditional cache-based multicore processor, its peak performance and memory bandwidth rival accelerator devices. Generating efficient code for such a new architecture requires a good understanding of its performance features. Using these features, we construct the Execution-Cache-Memory (ECM) performance model for the A64FX processor in the FX700 supercomputer and validate it**



Accepted for the 11th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS). Preprint: arXiv:2009.13903

NEWS

Japan Captures TOP500 Crown with Arm-Powered Supercomputer

TOP500 The List.

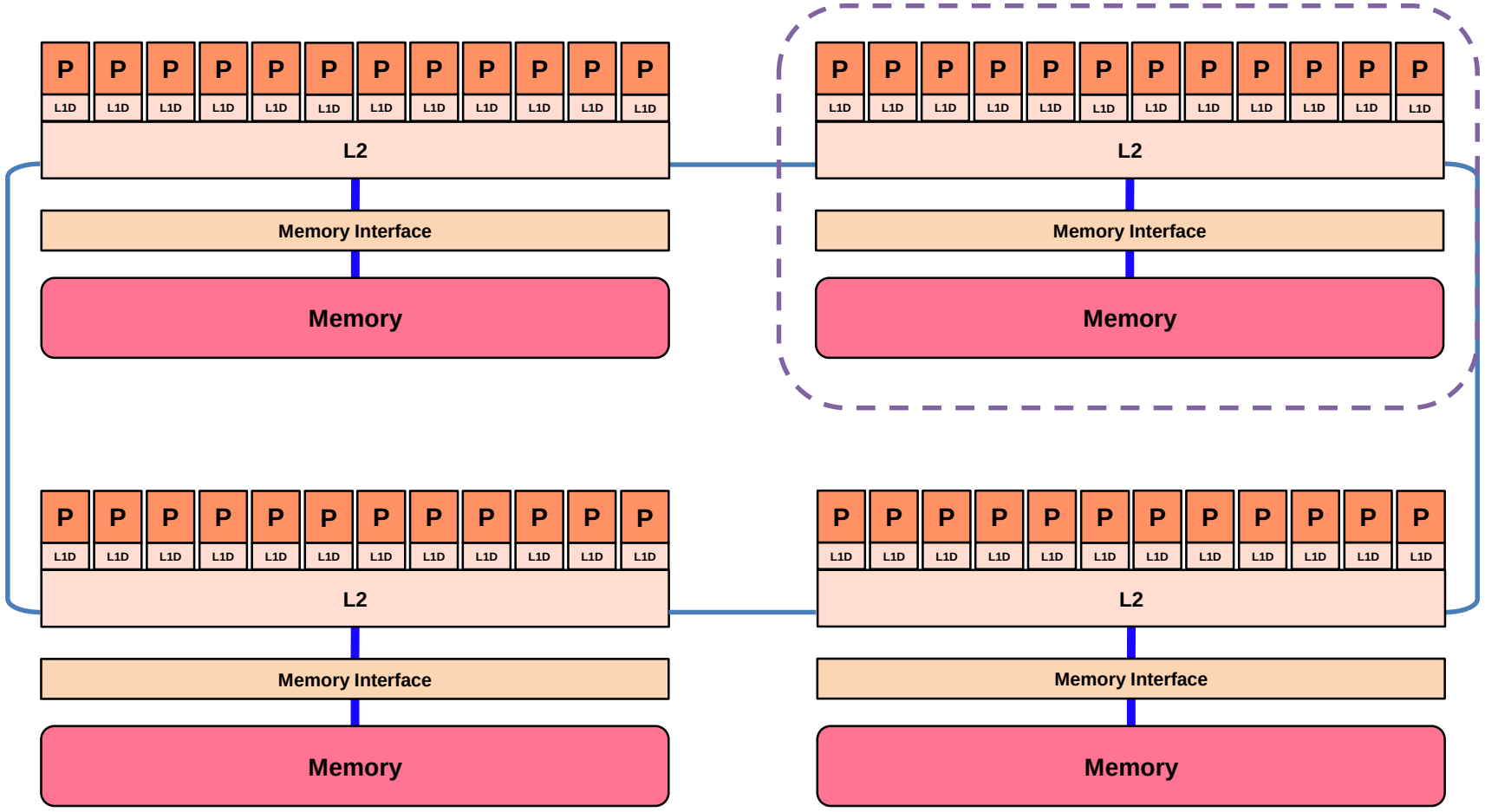
June 22, 2020

FRANKFURT, Germany; BERKELEY, Calif.; and KNOXVILLE, Tenn.—The 55th edition of the TOP500 saw some significant additions to the list, spearheaded by a new number one system from Japan. The latest rankings also reflect a steady growth in aggregate performance and power efficiency.

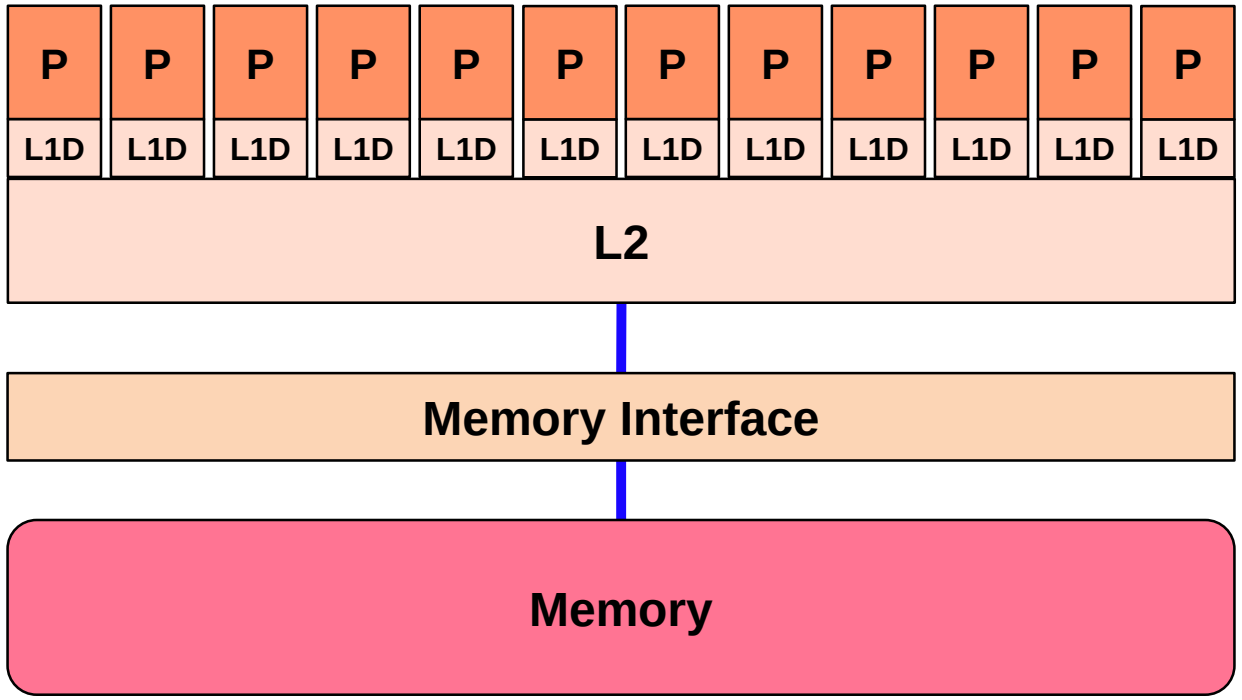
The new top system, Fugaku, turned in a High Performance Linpack (HPL) result of 415.5 petaflops, besting the now second-place Summit system by a factor of 2.8x. Fugaku, is powered by Fujitsu's 48-core A64FX SoC, becoming the first number one system on the list to be powered by ARM processors. In single or further reduced precision, which are often used in machine learning and AI applications, Fugaku's peak performance is over 1,000 petaflops (1 exaflops). The new system is installed at RIKEN Center for Computational Science (R-CCS) in Kobe, Japan.

Source : https://www.top500.org/news/japan-captures-top500-crown-arm-powered-supercomputer/

A64FX – FX700

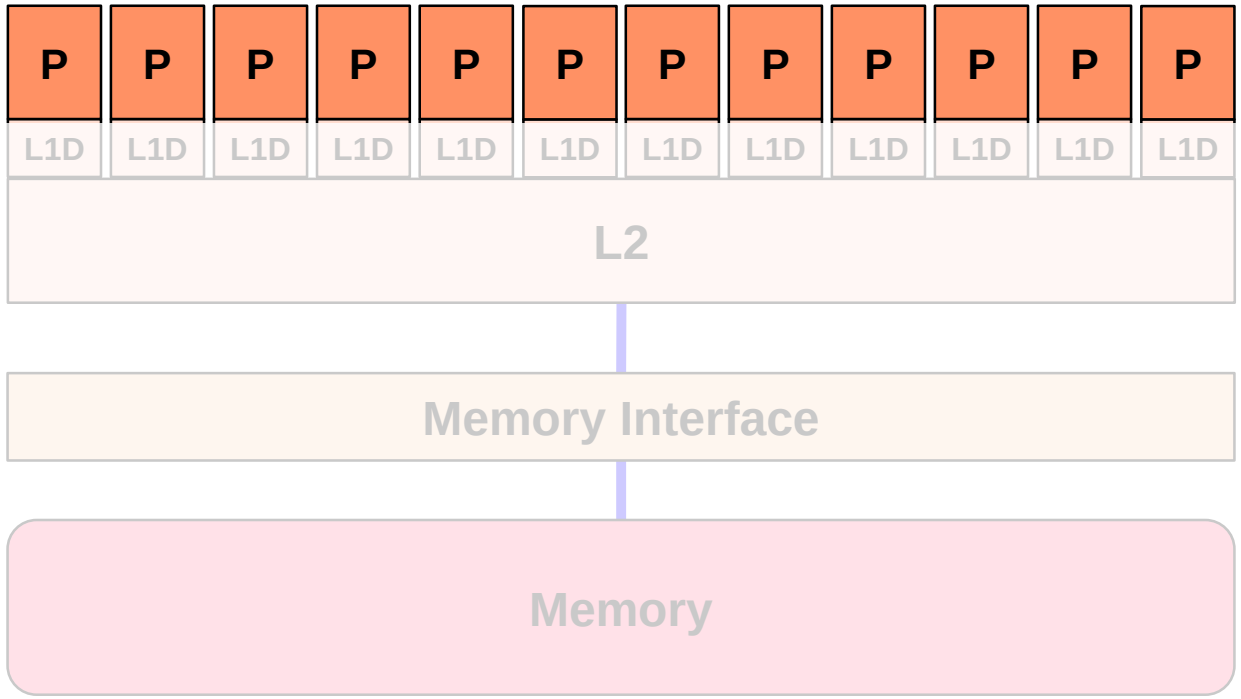

P
L1D
L2
Memory Interface
Memory
1 CMG
4 CMGs
per node
48 cores
per node

A64FX – FX700



1 CMG

A64FX – FX700



1 CMG

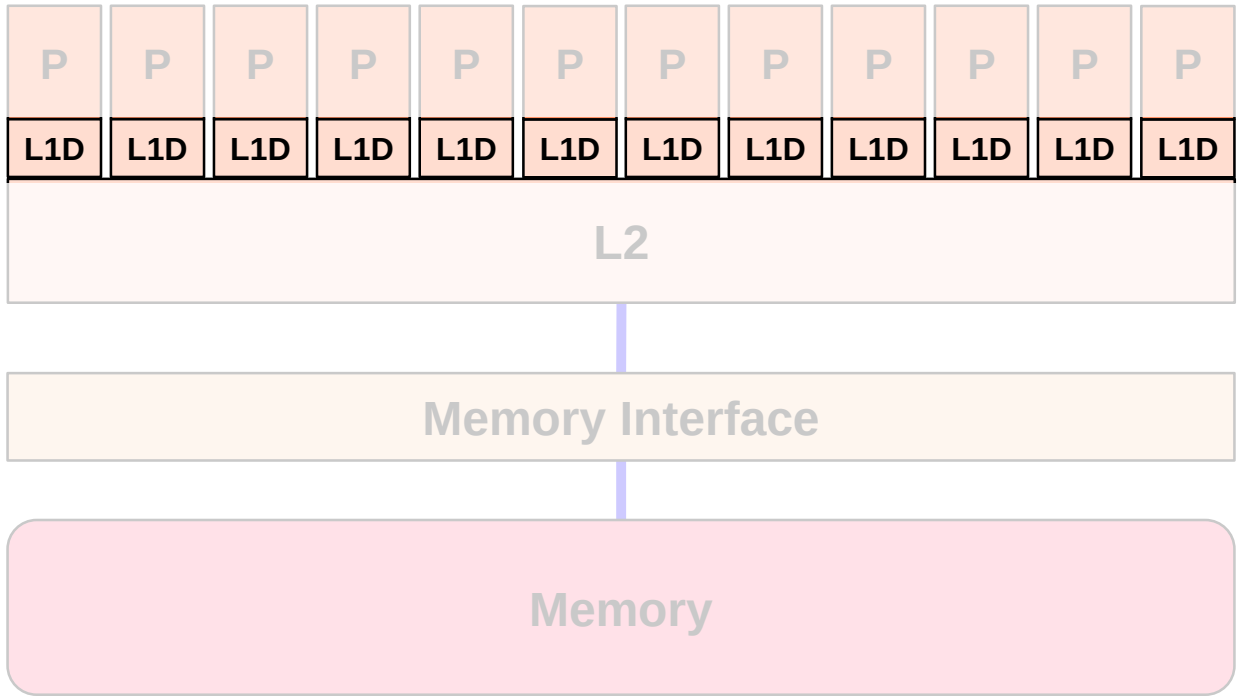
Core

- Clock : 1.8 GHz
- Instruction set : Armv8.2-A+SVE
- Maximum VL : 512 bit (8 double)

For example on GCC compiler v 10.1.1 use :

-msve-vector-bits=512 -march=armv8.2-a+sve

A64FX – FX700

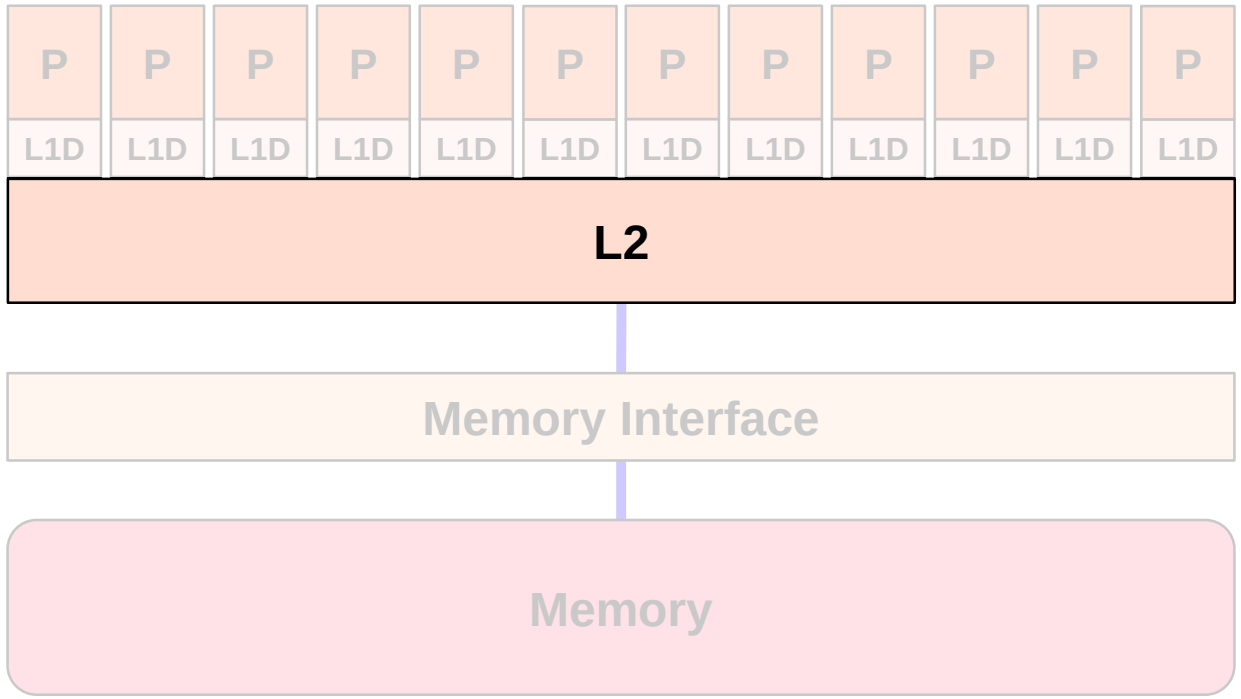


1 CMG

L1D cache

- Size : 64 KiB
- Topology : Private cache
- Cache line size : 256 bytes

A64FX – FX700

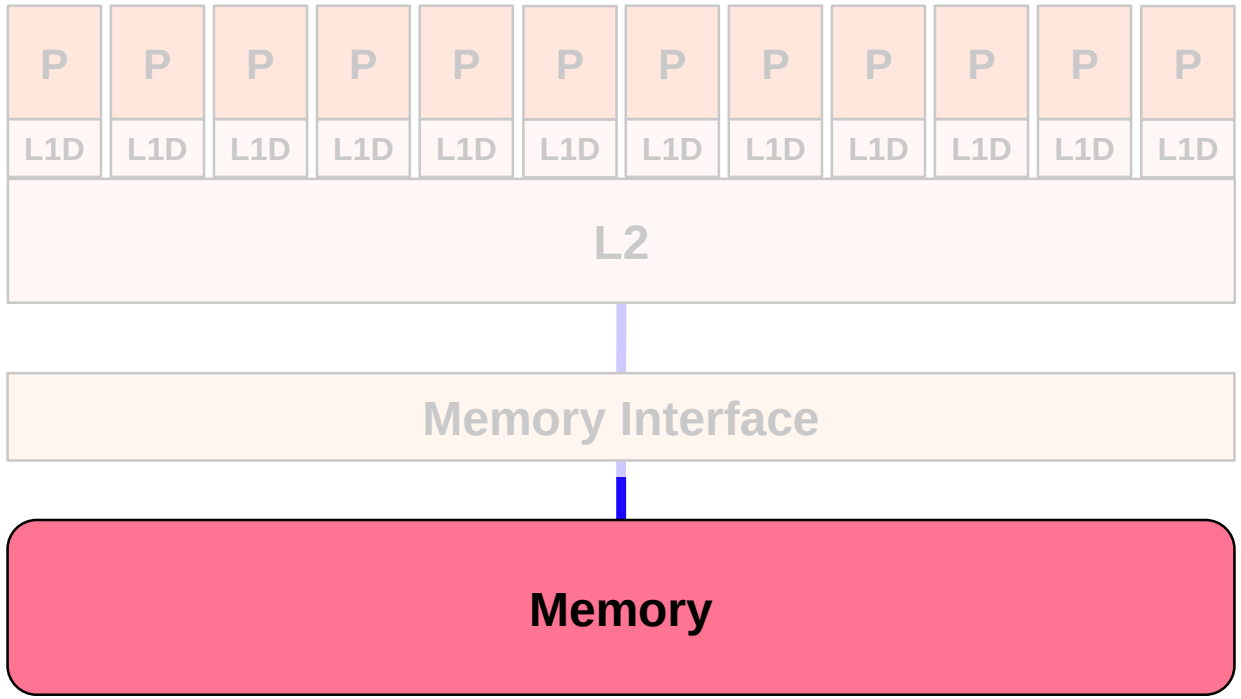


1 CMG

L2 cache

- Size : 8 MiB
- Topology : Shared within 1 CMG
- Cache line size : 256 bytes

A64FX – FX700

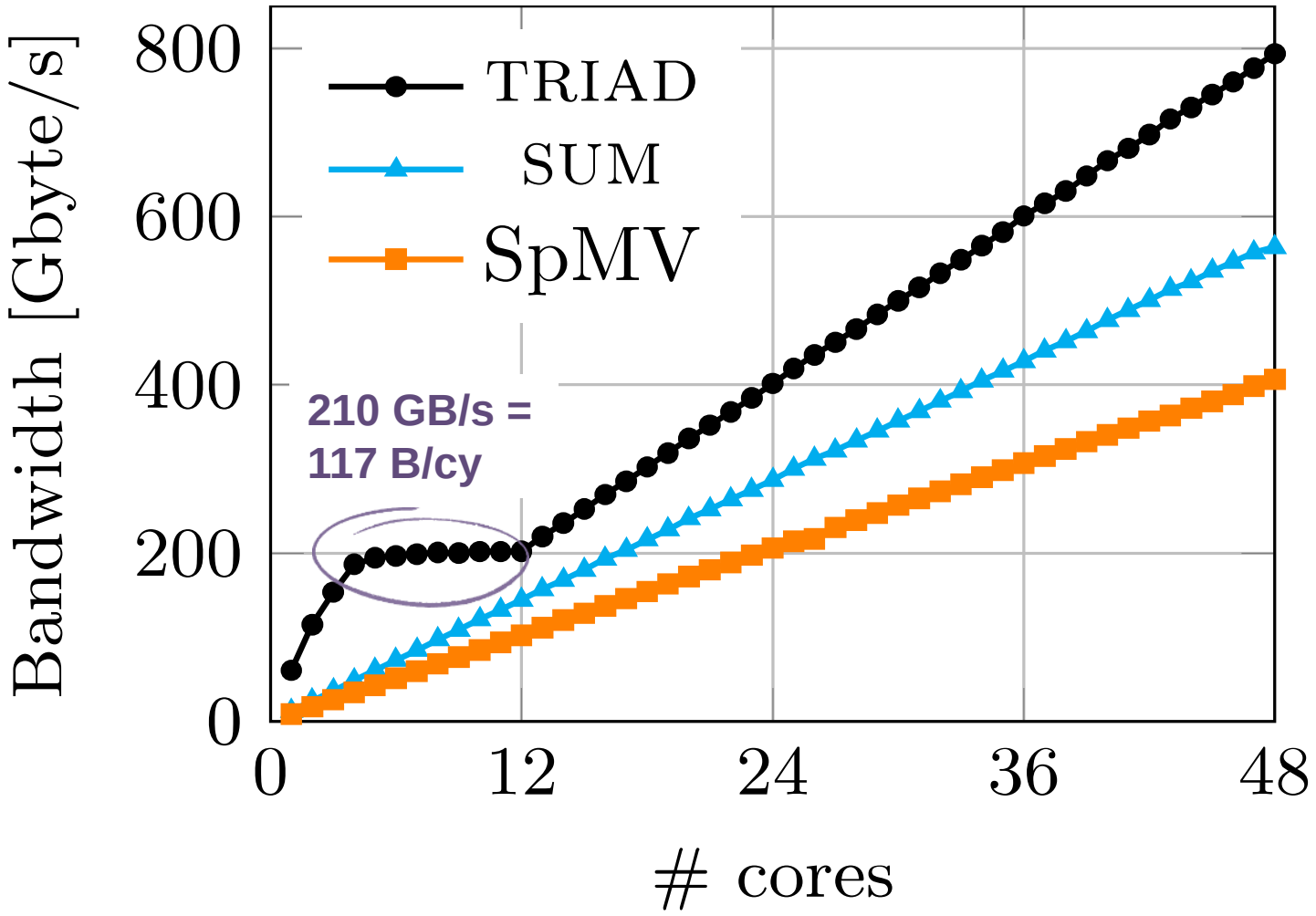


1 CMG

Main Memory

- Size : 4 x 8 GiB
- Type : HBM2

Motivation

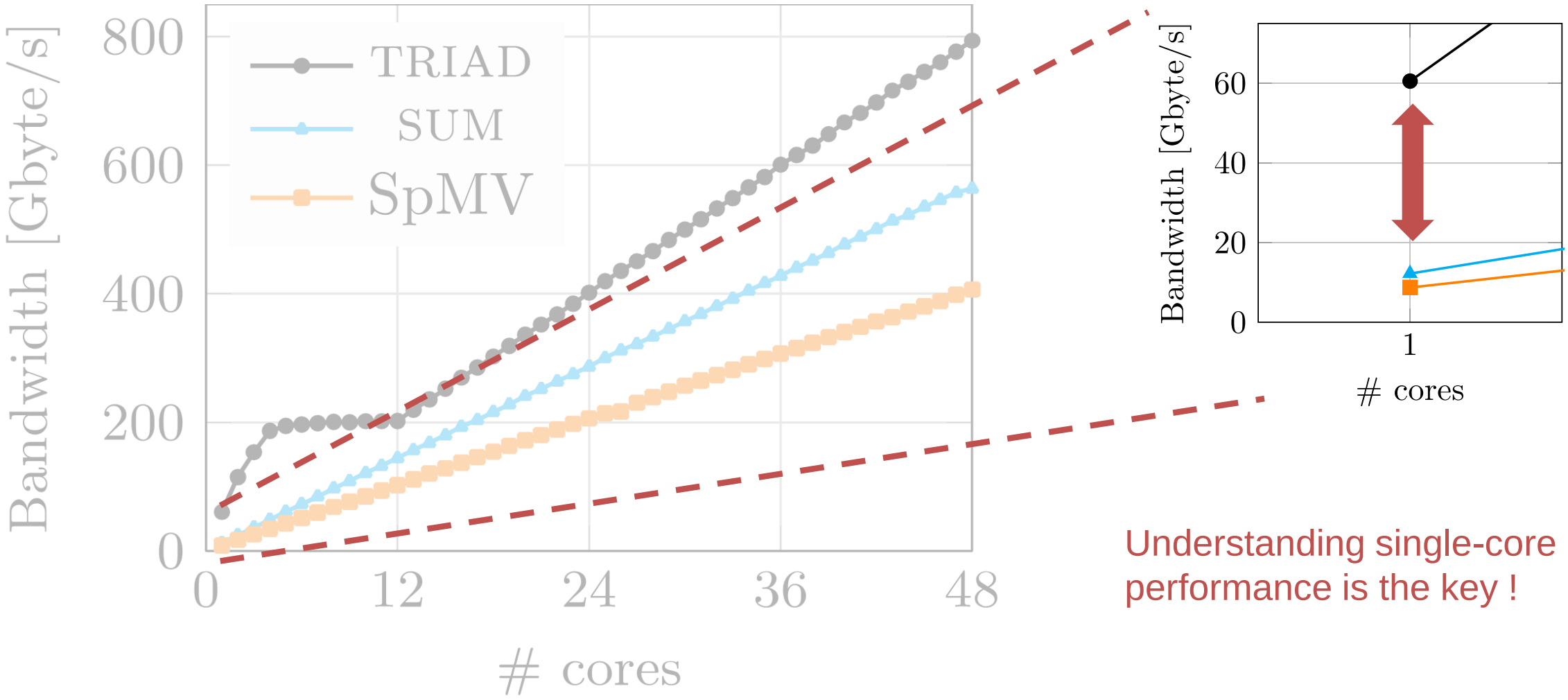


Thread pinning : Compact

Clear memory bandwidth saturation for STREAM TRIAD (`a[i] = b[i] + s*c[i]`).

But why not for SUM (`s += a[i]`) and SpMV (`b = Ax`) ?

Motivation

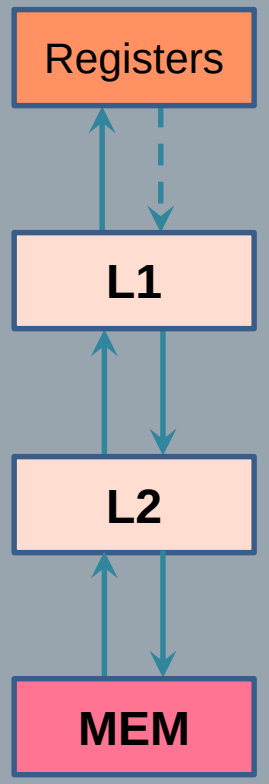


Understanding single-core performance is the key !

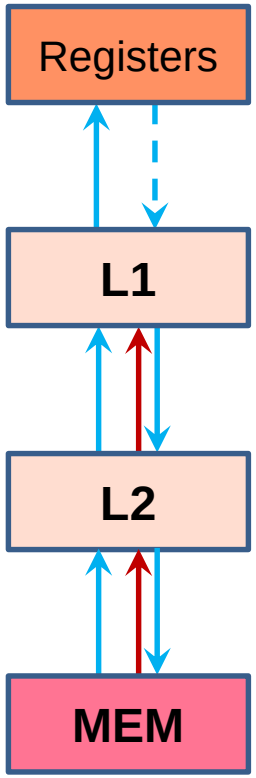
Thread pinning : Compact

Single core analysis

ECM model

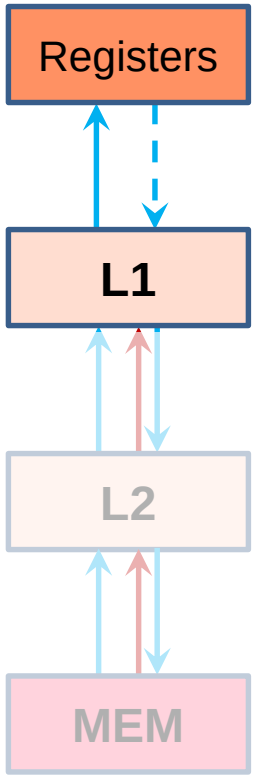


Single core ECM model



Execution-Cache-Memory (ECM) model helps us to understand and analyze the single-core performance.

Single core ECM model

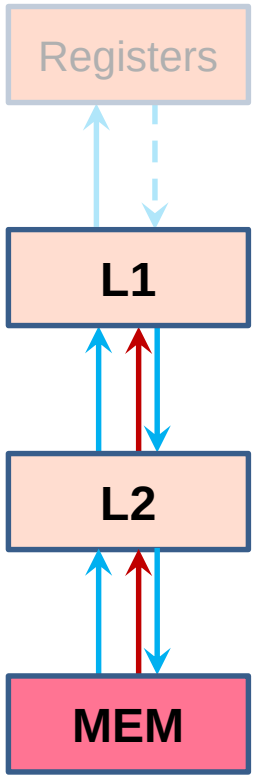


Execution-Cache-Memory (ECM) model helps us to understand and analyze the single-core performance.

3 major components :

1) In-core

Single core ECM model



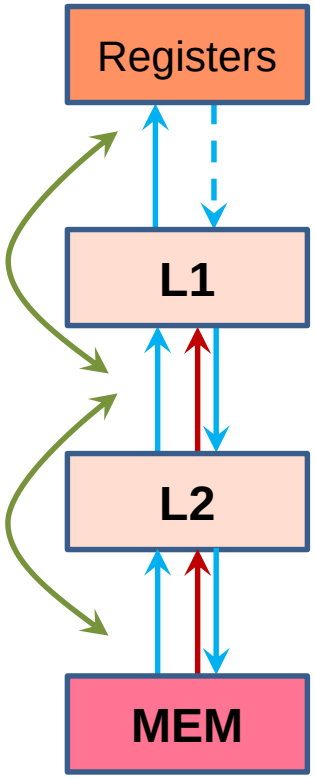
Execution-Cache-Memory (ECM) model helps us to understand and analyze the single-core performance.

3 major components :

1) In-core

2) Data transfer through memory hierarchies

Single core ECM model



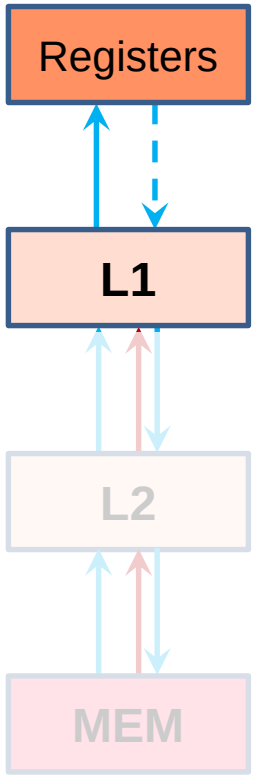
Can these transfers be overlapped or not ?

Execution-Cache-Memory (ECM) model helps us to understand and analyze the single-core performance.

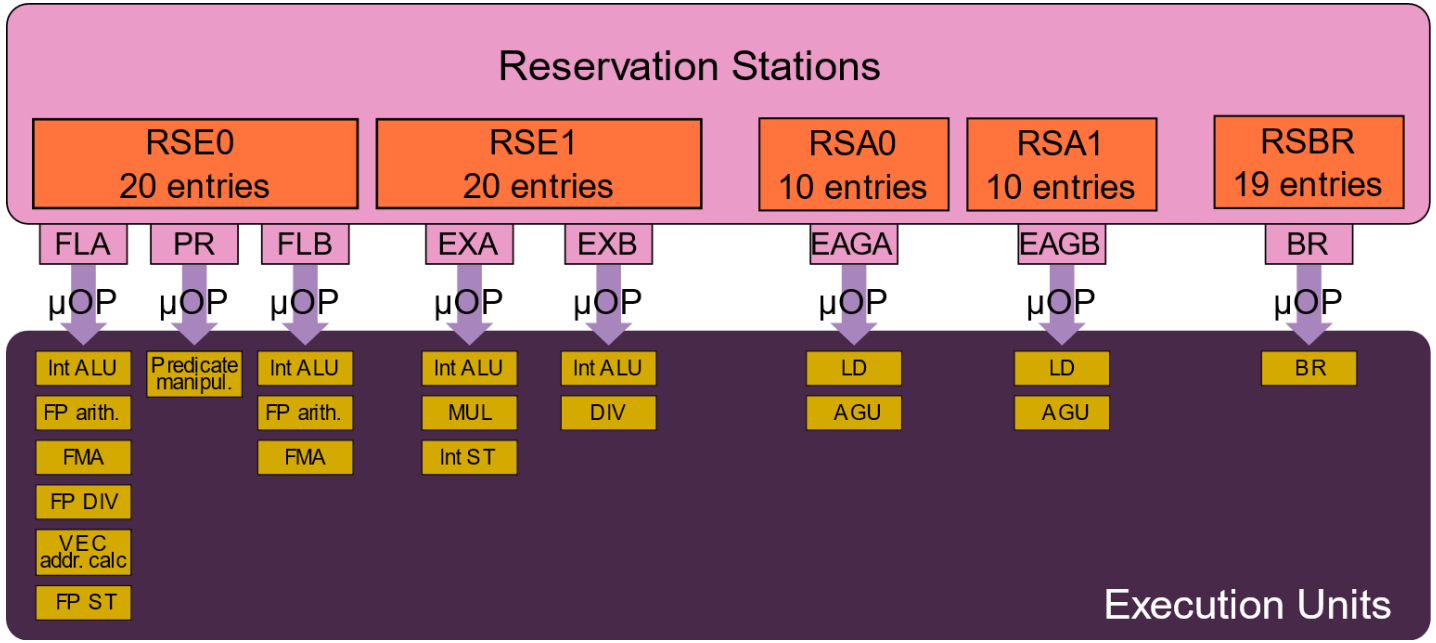
3 major components :

1) In-core
2) Data transfer through memory hierarchies
3) Overlap hypothesis

Incore



Machine knowledge



Incore

Application knowledge

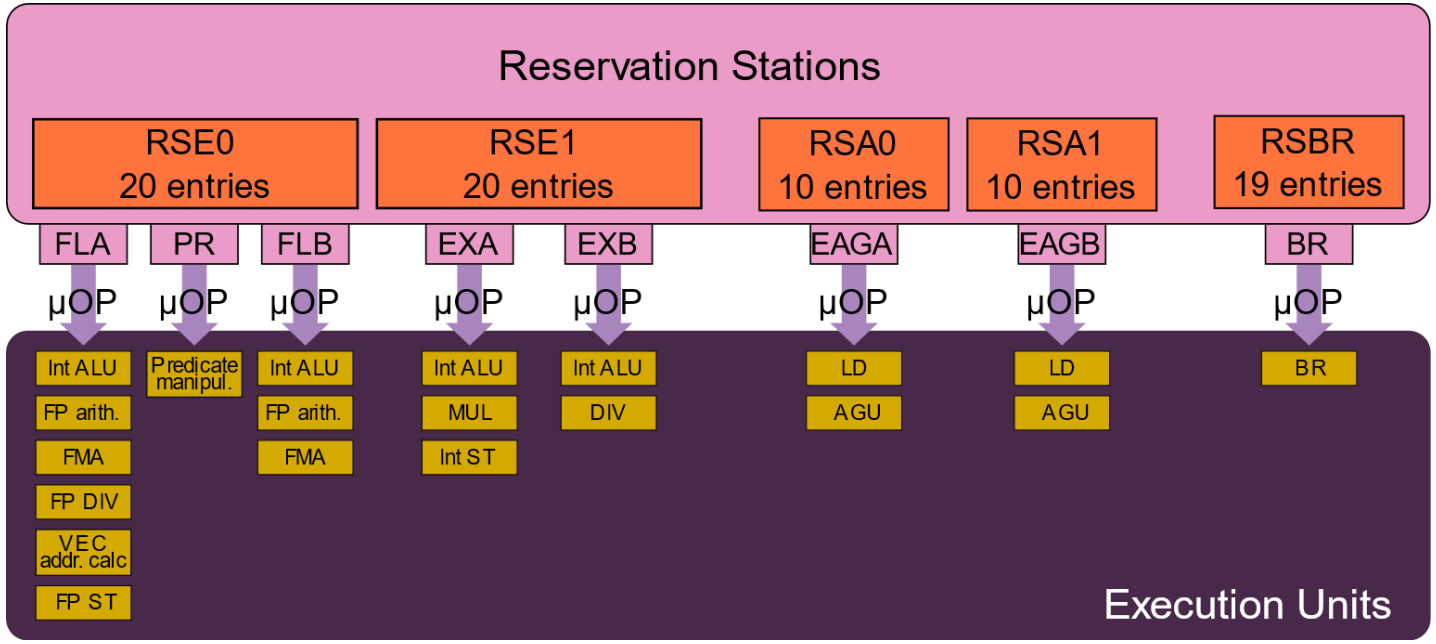
STREAM TRIAD

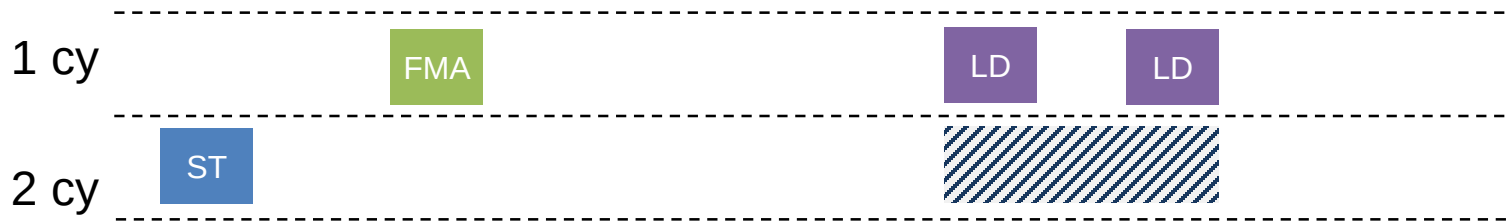
a[i] = b[i] + s * c[i]

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```

2cy / VL

Machine knowledge





Incore

Application knowledge

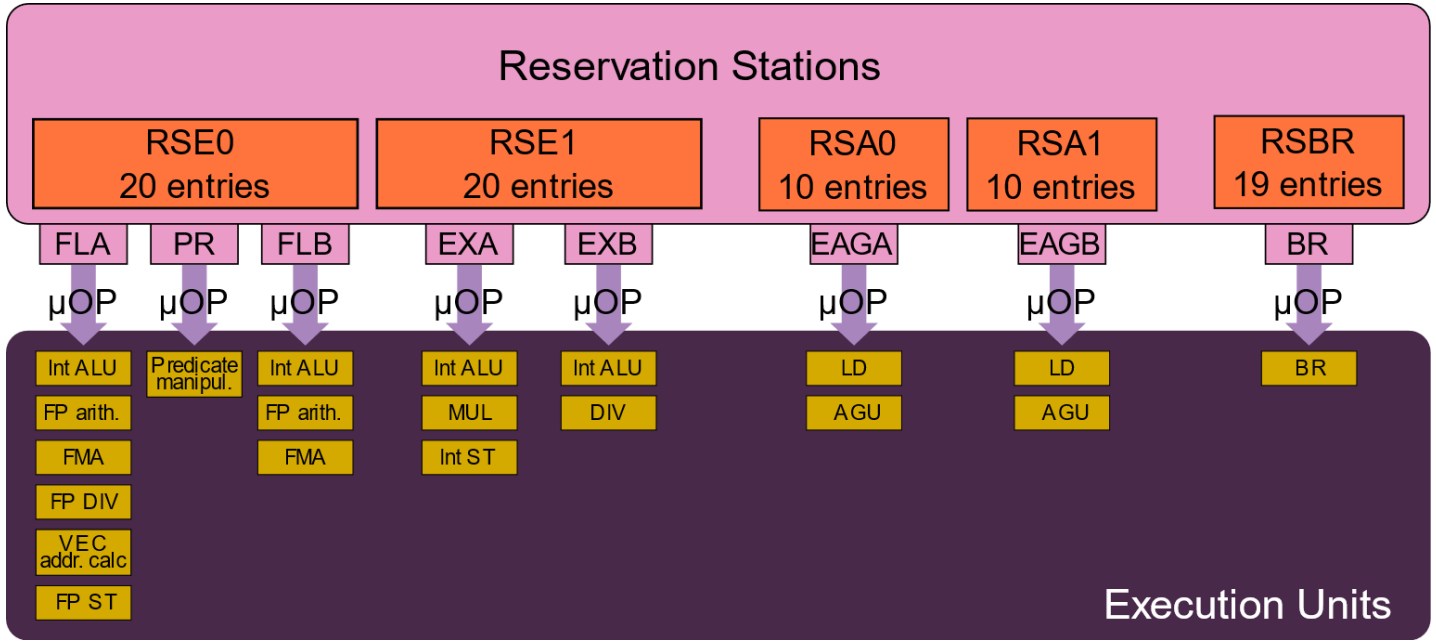
STREAM TRIAD

a[i] = b[i] + s * c[i]

```
.L18:
 ld1d z4.d, p5/z, [x21, x9, lsl 3]
 ld1d z5.d, p5/z, [x20, x9, lsl 3]
 fmad z5.d, p5/m, z2.d, z4.d
 st1d z5.d, p5, [x19, x9, lsl 3]
 add x8, x9, 8
 whilelo p5.d, w8, w7
 b.any .L18
```

2cy / VL

Machine knowledge



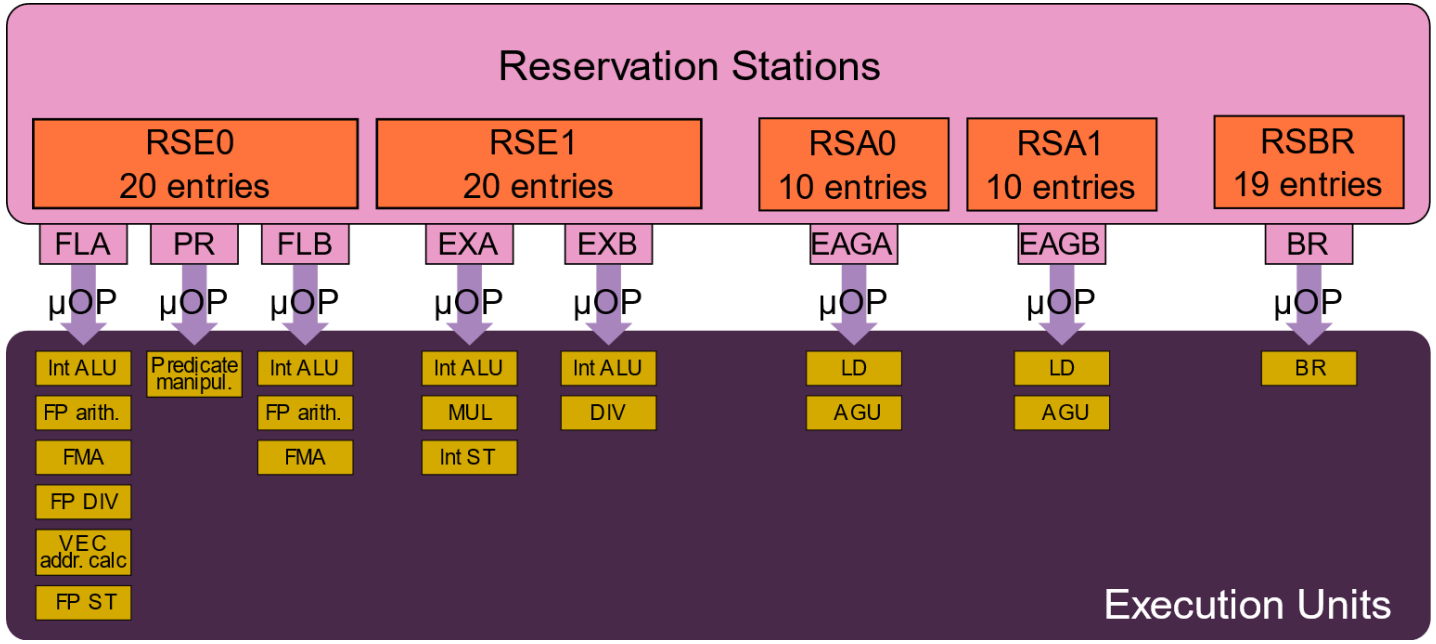
OSACA

Tool to do this static analysis and predict the incore contribution.

https://github.com/RRZE-HPC/OSACA

Incore

Instruction	Reciprocal Throughput [cy]	Latency [cy]
ld1d	0.5	11
st1d	1.0	–
fadd	0.5	9
fmad	0.5	9
faddv	11.5	49



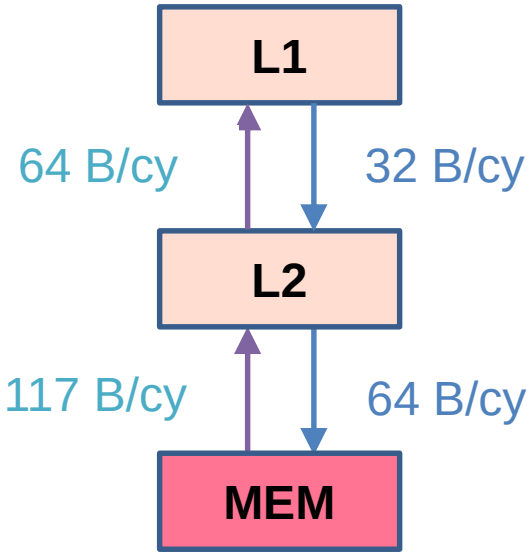
Tool to do this static analysis and predict the incore contribution.

https://github.com/RRZE-HPC/OSACA

Data transfer

Machine knowledge

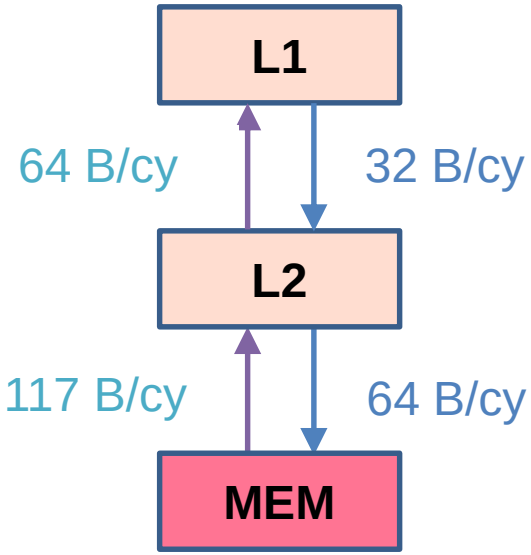
FX700



Data transfer

Machine knowledge

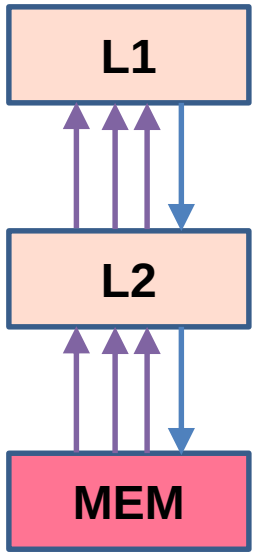
FX700



Application knowledge

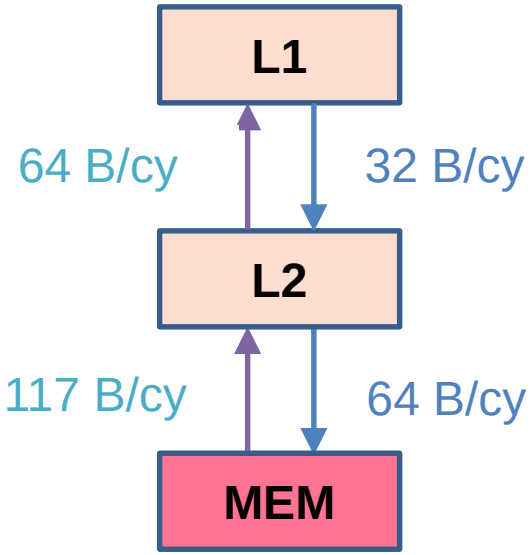
STREAM TRIAD

```
a[i] = b[i] + s*c[i]
```



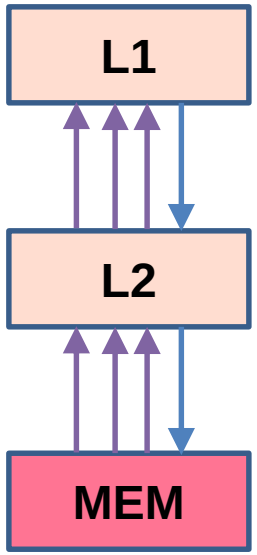
Data transfer

Machine knowledge
FX700

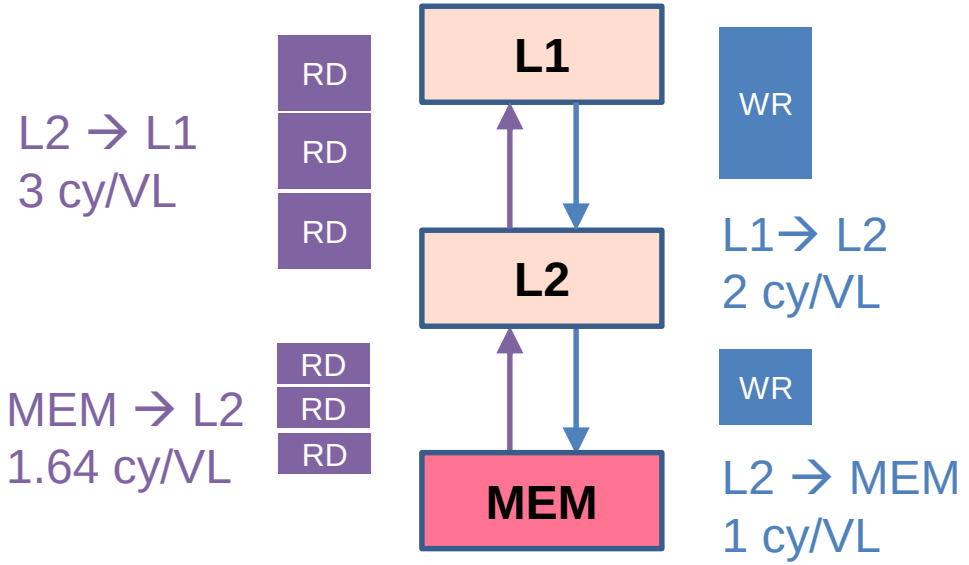


Application knowledge
STREAM TRIAD

```
a[i] = b[i] + s*c[i]
```

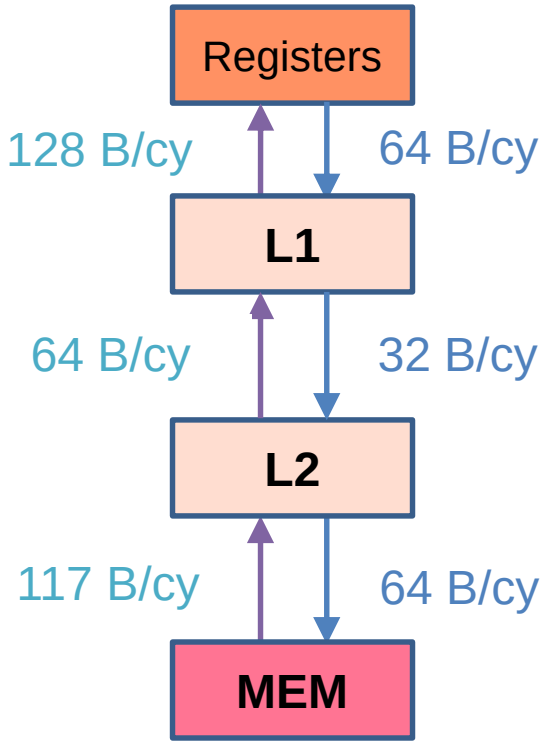


ECM prediction
STREAM TRIAD on FX700



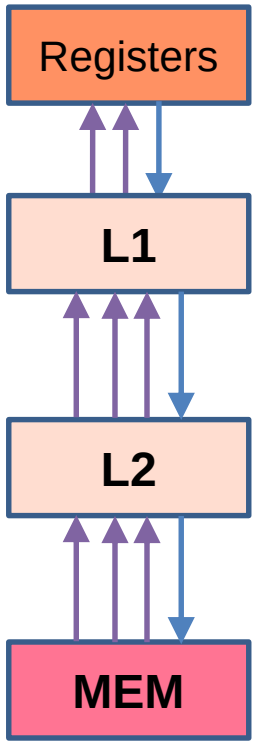
ECM model

Machine knowledge
FX700

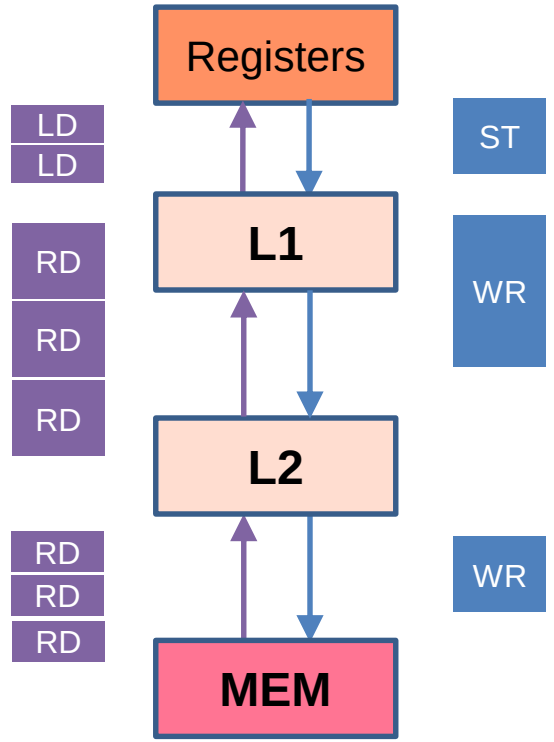


Application knowledge
STREAM TRIAD

```
a[i] = b[i] + s*c[i]
```



ECM prediction
STREAM TRIAD on FX700

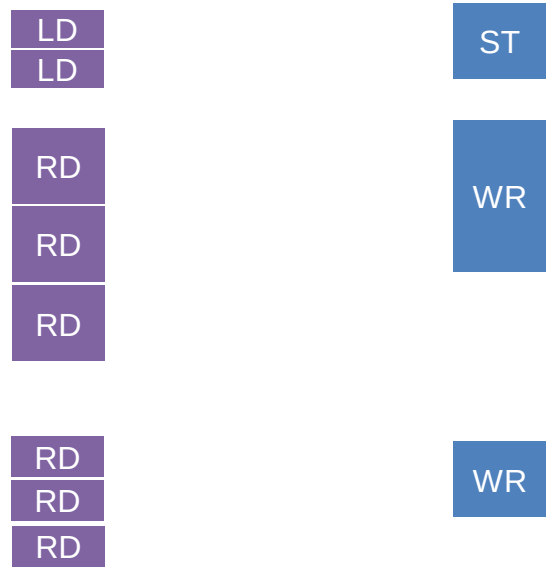


ECM model

How do these boxes overlap?

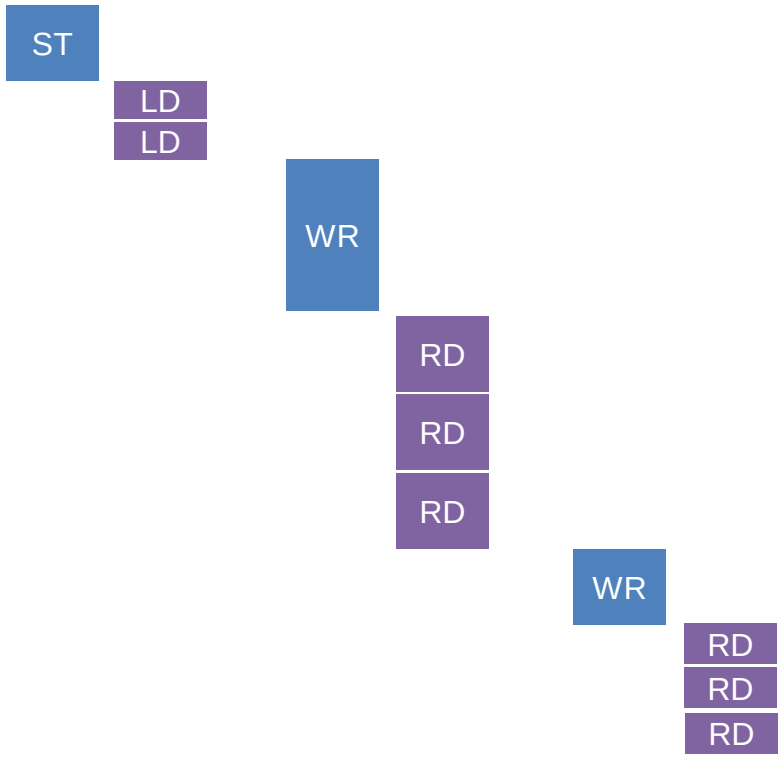
ECM prediction

STREAM TRIAD on FX700



ECM model

Hypothesis 1 : No overlap

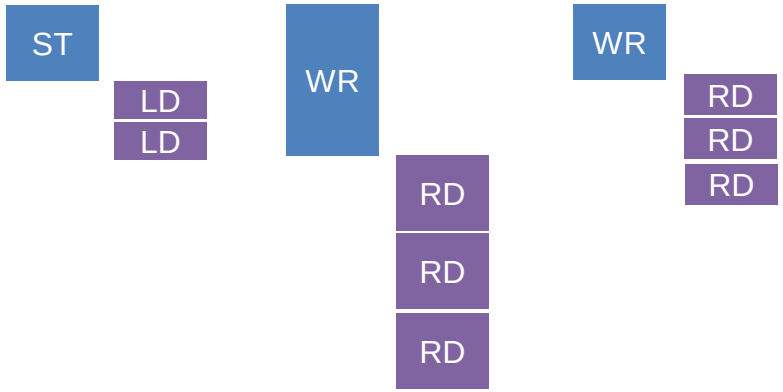


ECM model

Hypothesis 2 : Full overlap

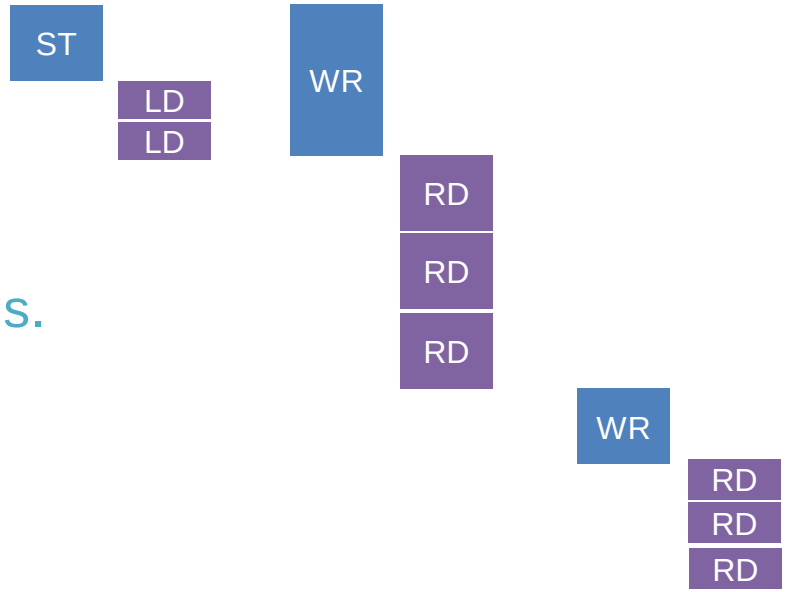
ECM model

Hypothesis 3 : Full overlap + half-duplex



ECM model

Hypothesis 4 : L1L2 overlap + half-duplex



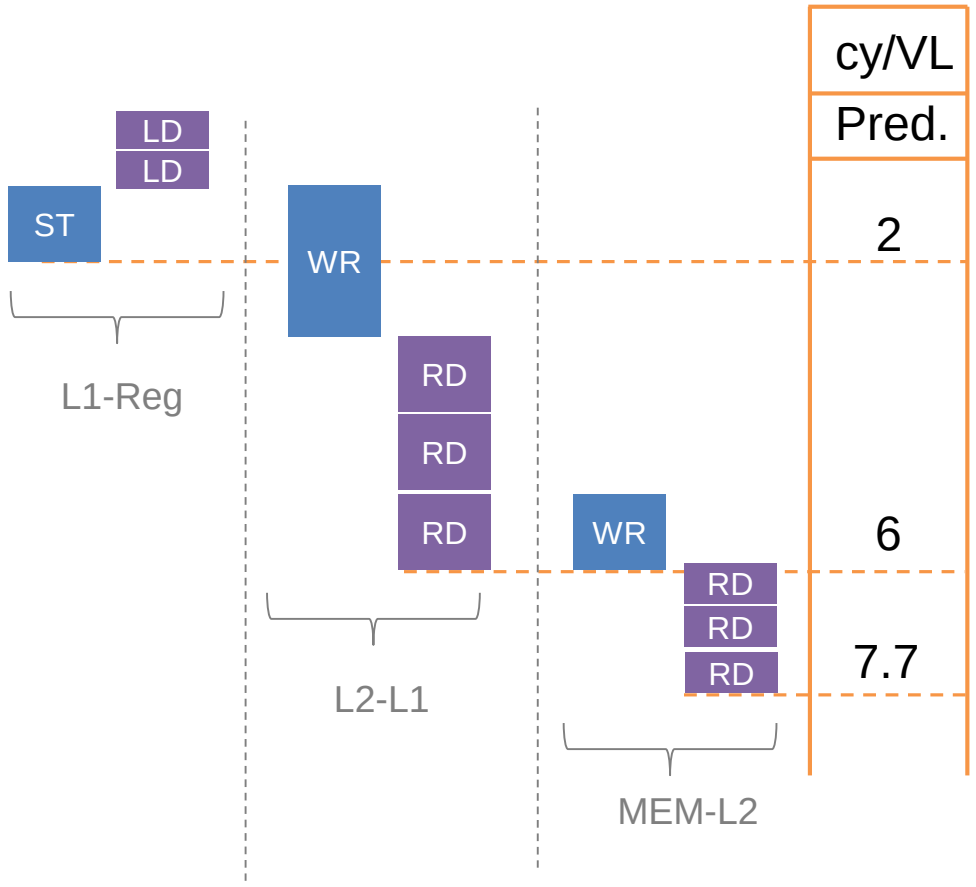
There are numerous combinations.

How do we find the correct one?

Compare measurements with predictions.

ECM model

The best hypothesis for FX700

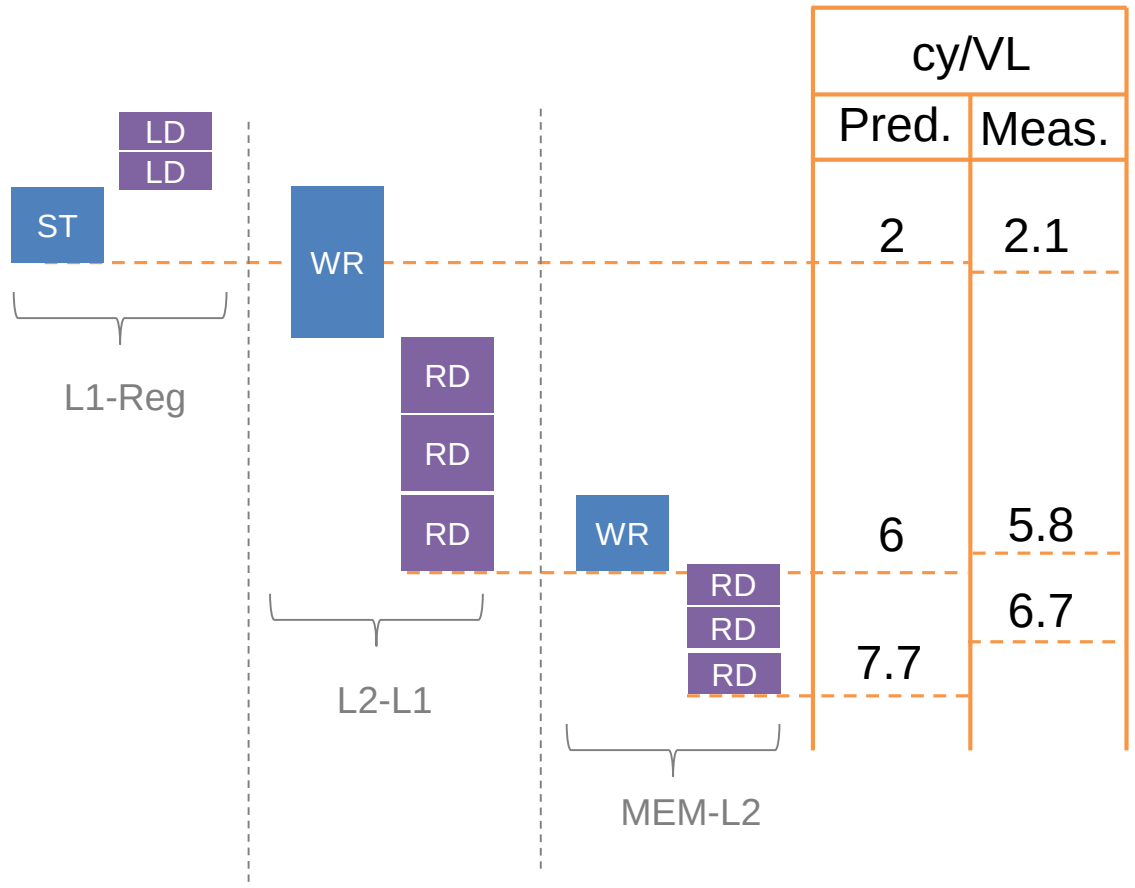


How do we find the correct one?

Compare measurements with predictions.

ECM model

The best hypothesis for FX700

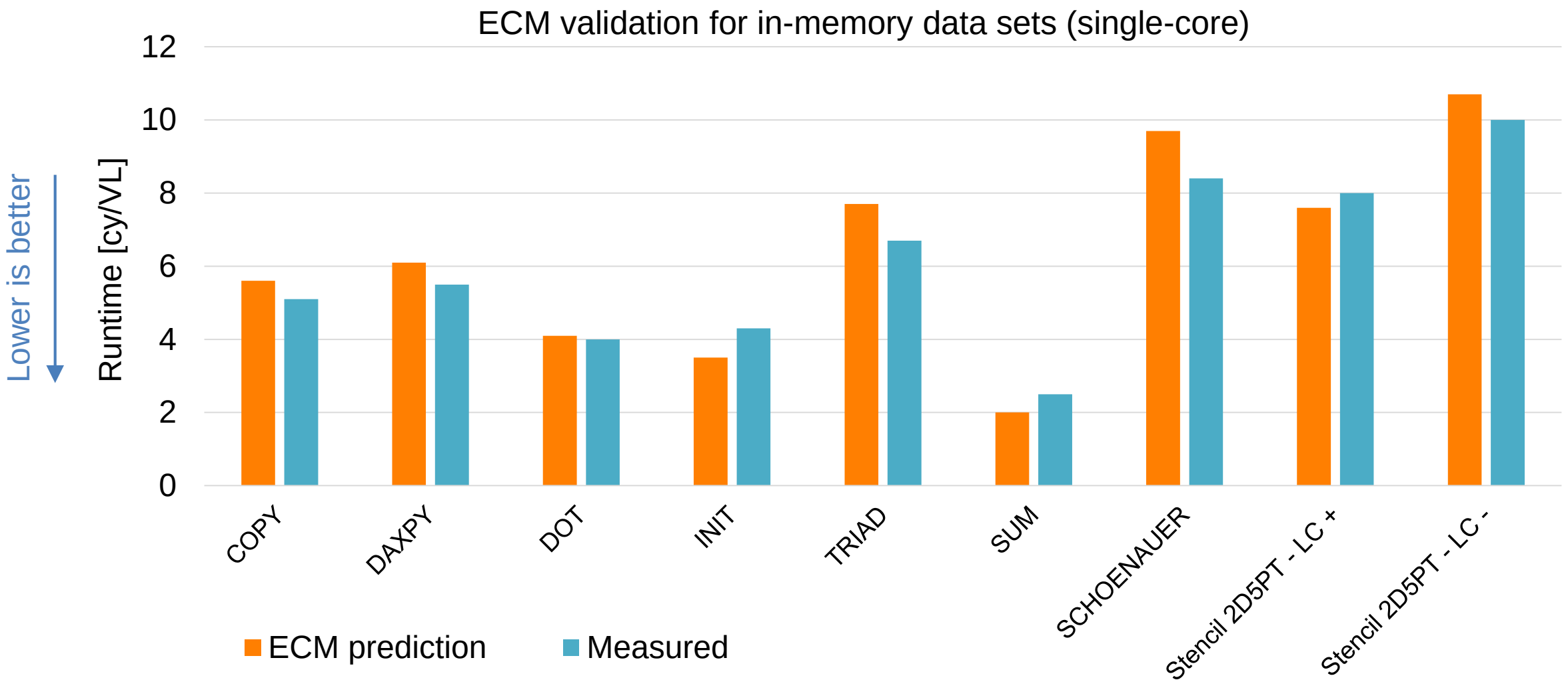


How do we find the correct one?

Compare measurements with predictions.

A systematic way of identifying overlap hypothesis is presented in : Hofmann et.al., 2020, Bridging The Architecture Gap: Abstracting Performance-relevant Properties Of Modern Server Processors, https://doi.org/10.14529/jsfi200204

Model validation



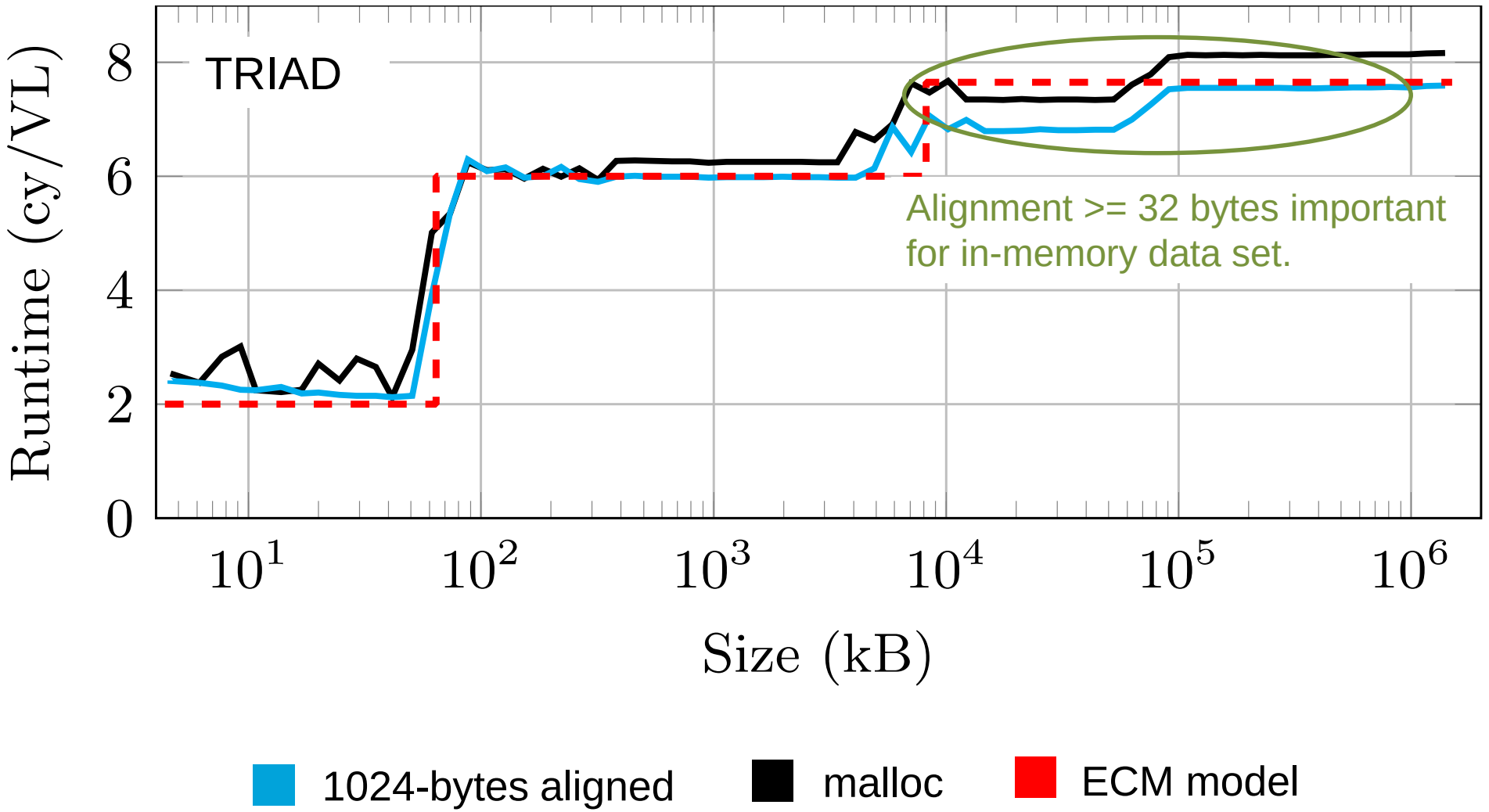
Erlangen Regional Computing Center

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

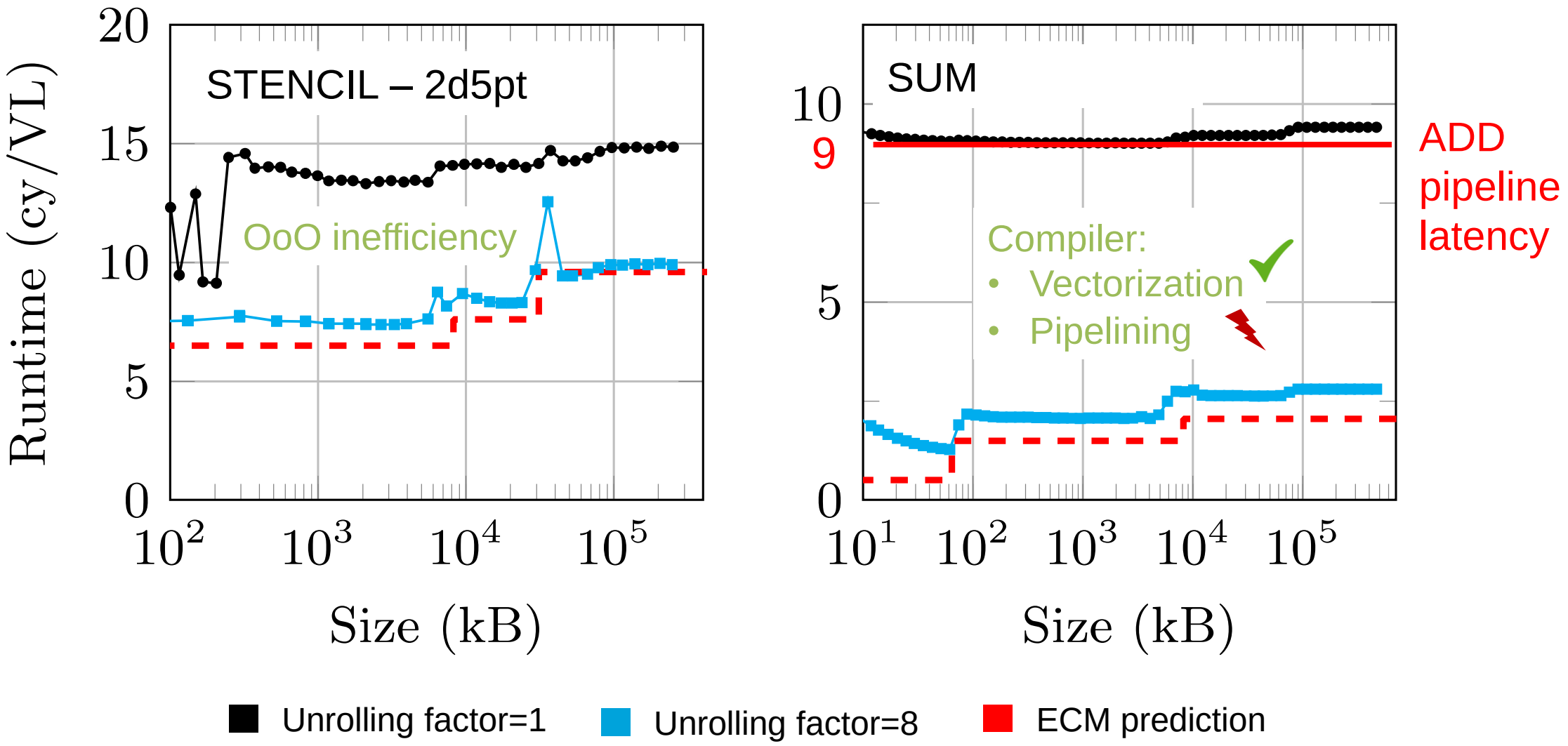
Some things to take care of on FX700

High Performance Computing

Data alignment is important



Unrolling plays an important role



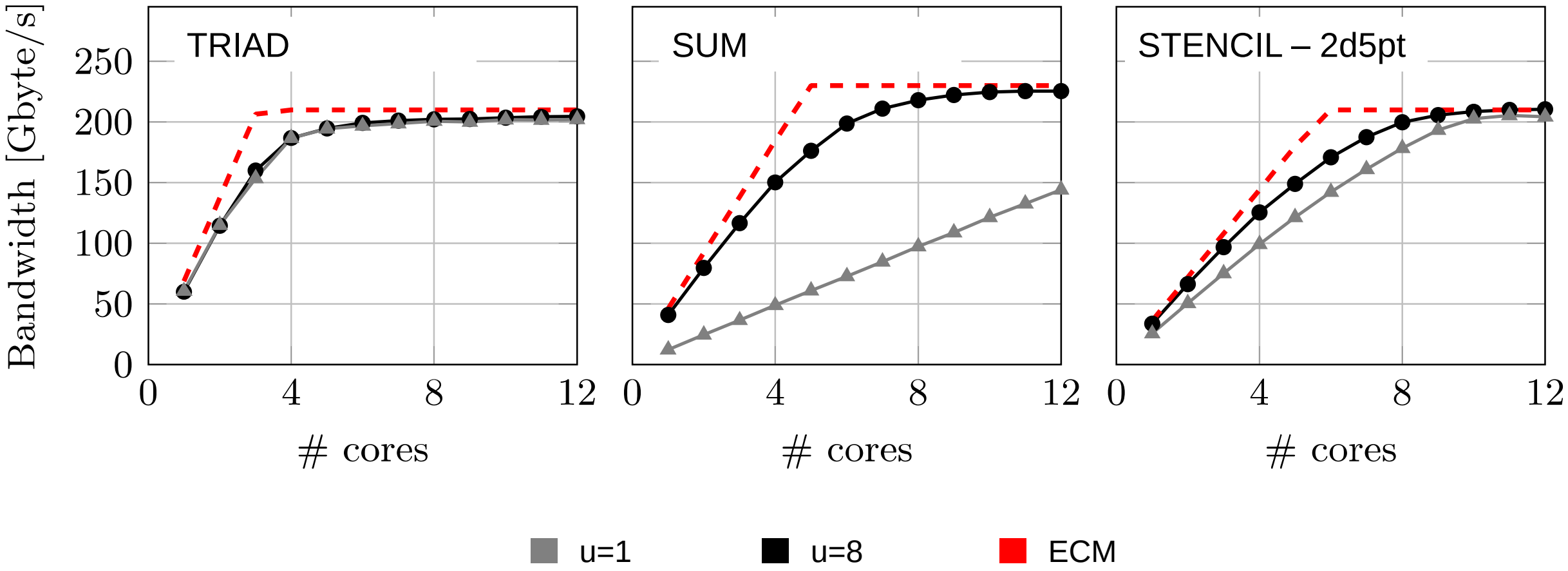
Erlangen Regional Computing Center

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Multicore saturation

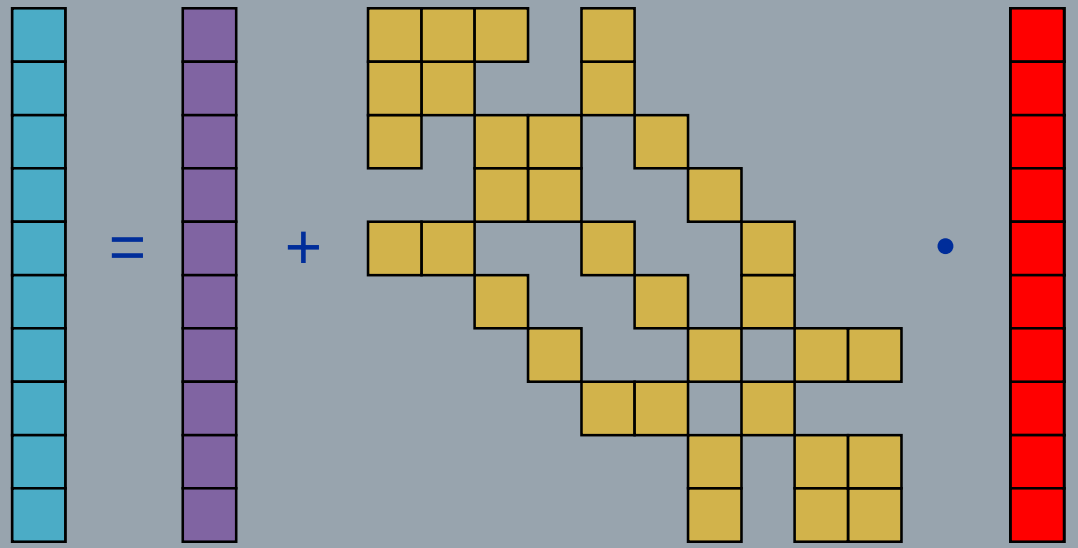
High Performance Computing

Multicore


TRIAD
SUM
STENCIL – 2d5pt
Bandwidth [Gbyte/s]
250
200
150
100
50
0
0
4
8
12
# cores
u=1
u=8
ECM

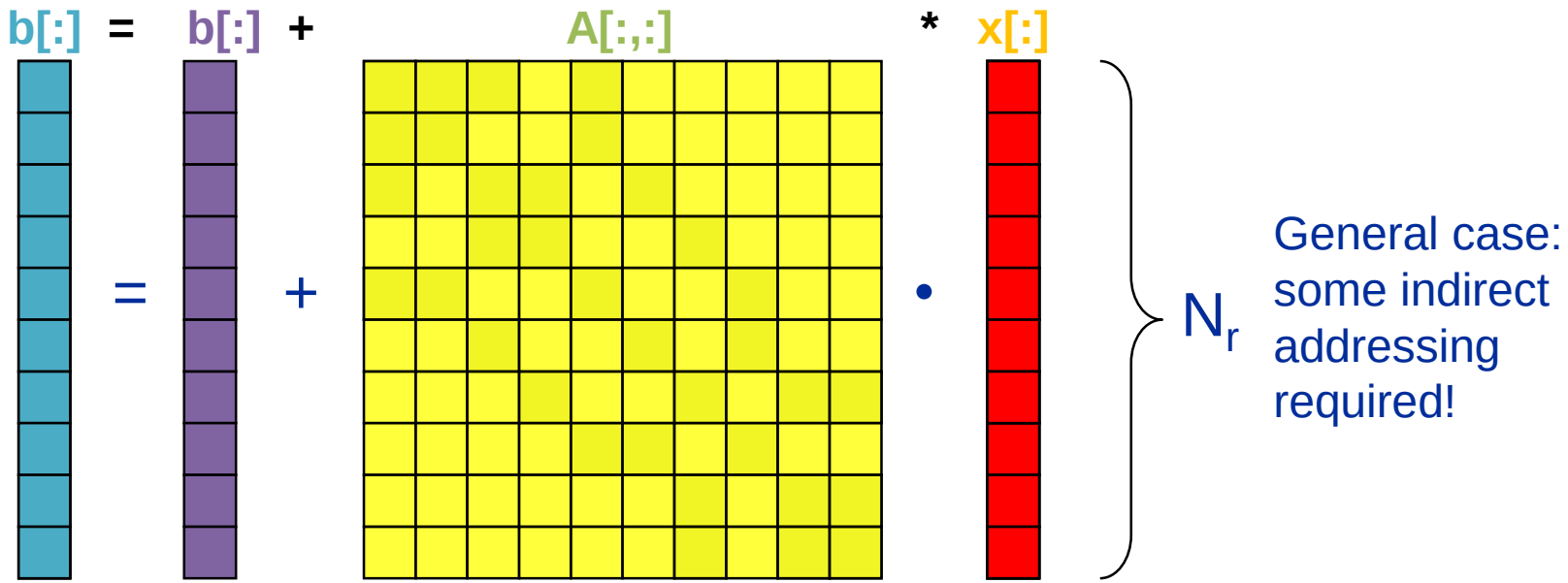
SpMV

Sparse Matrix-Vector Multiplication



SpMV

Sparse Matrix-Vector Multiplication (SpMV) : `b=Ax`



General case: some indirect addressing required!

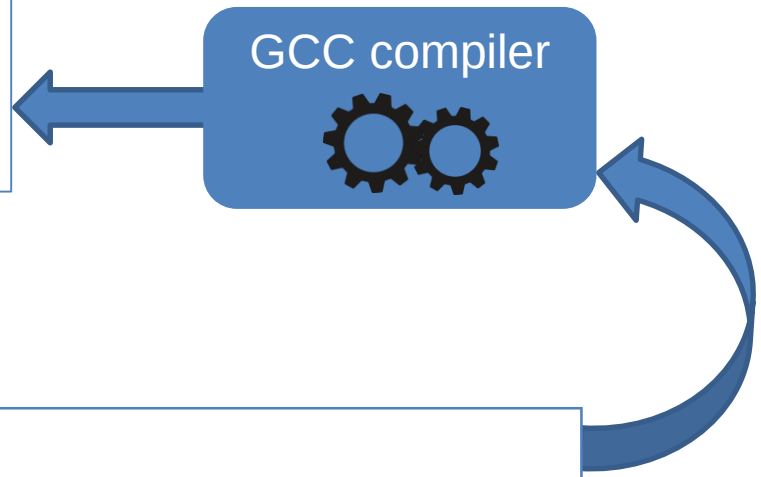
In Compressed Row Storage (CRS) format

```
for i = 0:nrows-1 //Long outer-loop
    for j = row_ptr[i]:row_ptr[i+1]-1 // Short inner-loop
        b[i] = b[i] + A[j] * x[col_idx[j]]
```

Assembly of the short inner-loop

```
.L6:
        ld1sw     z0.d, p0/z, [x17, x20, lsl 2]
        ld1d      z2.d, p0/z, [x18, x20, lsl 3]
        ld1d      z3.d, p0/z, [x30, z0.d, lsl 3]
        add       x20, x20, 8
        fmla      z1.d, p0/m, z3.d, z2.d
        whilelo   p0.d, x20, x14
        b.any     .L6

        faddv     d4, p1, z1.d
```



In Compressed Row Storage (CRS) format

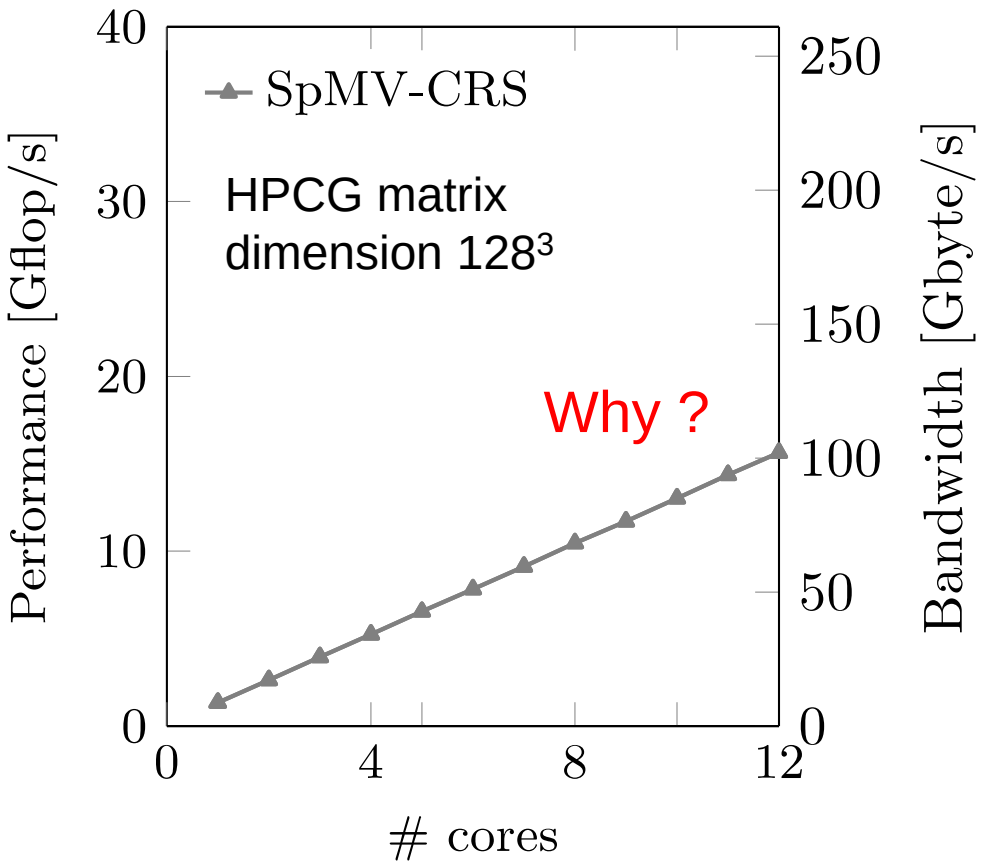
```
for i = 0:nrows-1 //Long outer-loop
    for j = row_ptr[i]:row_ptr[i+1]-1 // Short inner-loop
        b[i] = b[i] + A[j] * x[col_idx[j]]
```

SpMV

Assembly of the short inner-loop

```
.L6:
        ld1sw     z0.d, p0/z, [x17, x20, lsl 2]
        ld1d      z2.d, p0/z, [x18, x20, lsl 3]
        ld1d      z3.d, p0/z, [x30, z0.d, lsl 3]
        add       x20, x20, 8
        fmla      z1.d, p0/m, z3.d, z2.d
        whilelo   p0.d, x20, x14
        b.any     .L6

        faddv     d4, p1, z1.d
```



In Compressed Row Storage (CRS) format

```
for i = 0:nrows-1 //Long outer-loop
    for j = row_ptr[i]:row_ptr[i+1]-1 // Short inner-loop
        b[i] = b[i] + A[j] * x[col_idx[j]]
```

SpMV

Assembly of the short inner-loop

```
.L6:
        ld1sw     z0.d, p0/z, [x17, x20, lsl 2]
        ld1d      z2.d, p0/z, [x18, x20, lsl 3]
        ld1d      z3.d, p0/z, [x30, z0.d, lsl 3]
        add       x20, x20, 8
        fmla      z1.d, p0/m, z3.d, z2.d
        whilelo   p0.d, x20, x14
        b.any     .L6

        faddv     d4, p1, z1.d
```

FMA: Update **z1.d**

Latency: 9 cycles

Loop length : 27

HPCG matrix

Horizontal add of 512-bit register

Throughput = 11.5

ECM model predicts maximum bandwidth of 160 GB/s[1]

→ No saturation ☹

In Compressed Row Storage (CRS) format

```
for i = 0:nrows-1 //Long outer-loop
    for j = row_ptr[i]:row_ptr[i+1]-1 // Short inner-loop
        b[i] = b[i] + A[j] * x[col_idx[j]]
```

[1]https://arxiv.org/abs/2009.13903

Erlangen Regional Computing Center

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

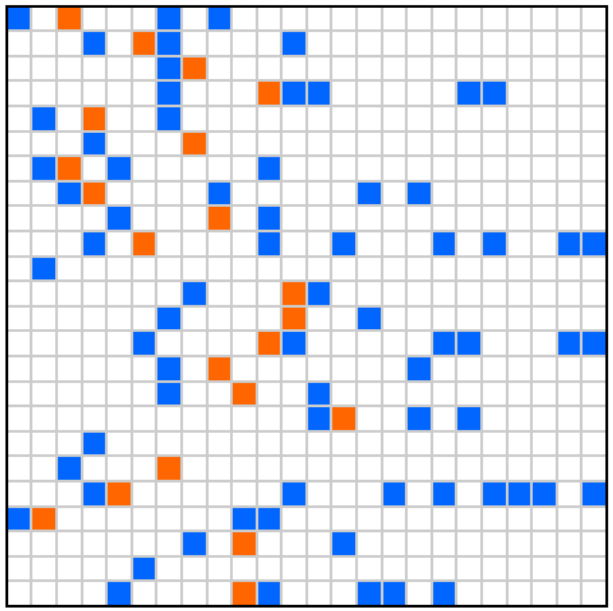
CRS → SELL-C-σ

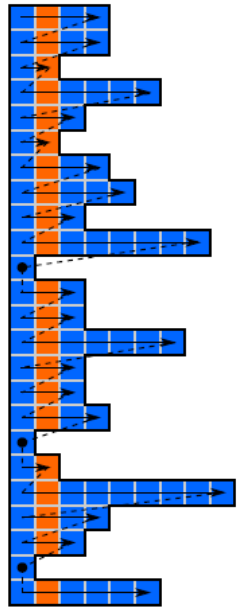
Change data storage format

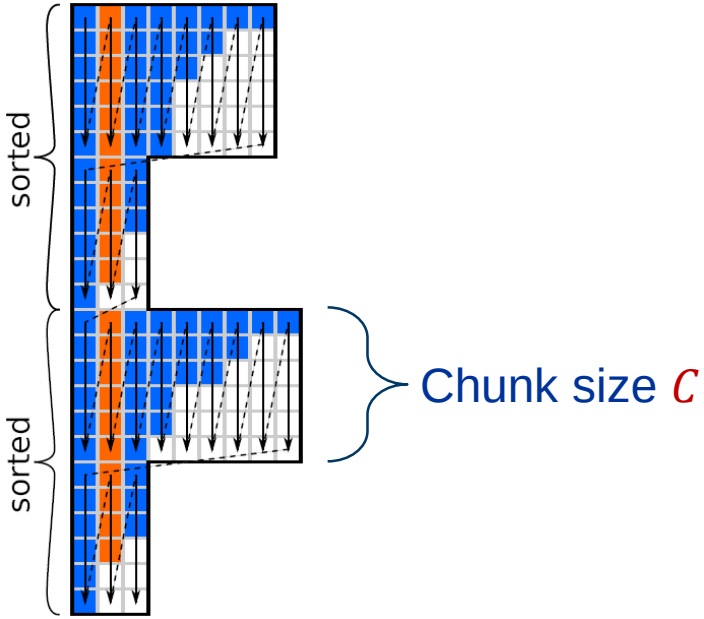
High Performance Computing

SpMV – SELL-C-σ

Vector-friendly SpMV data format/kernel required for A64FX → SELL-C-σ[1]







CRS SELL-C-σ

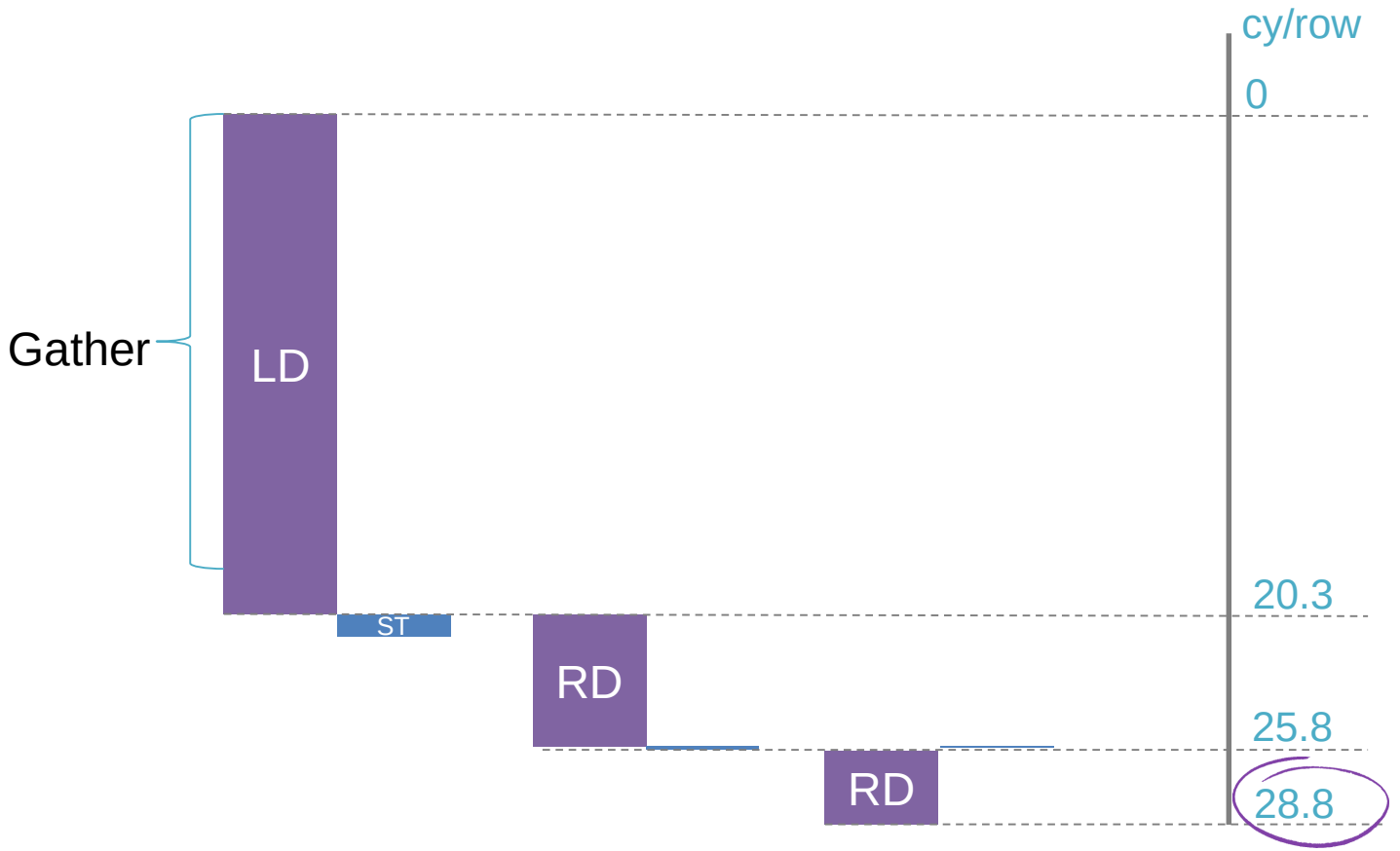
Benefits:

- Vectorization and unrolling along chunk size (C) → long loop and tunable
- No costly horizontal-add (faddv)
- No loss of vector efficiency

[1]M. Kreutzer et al., A Unified Sparse Matrix Data Format For Efficient General Sparse Matrix-vector Multiplication On Modern Processors With Wide Simd Units, SIAM SISC 2014, DOI: 10.1137/130930352

ECM model SELL-C-σ

HPCG matrix, dimension 128^3

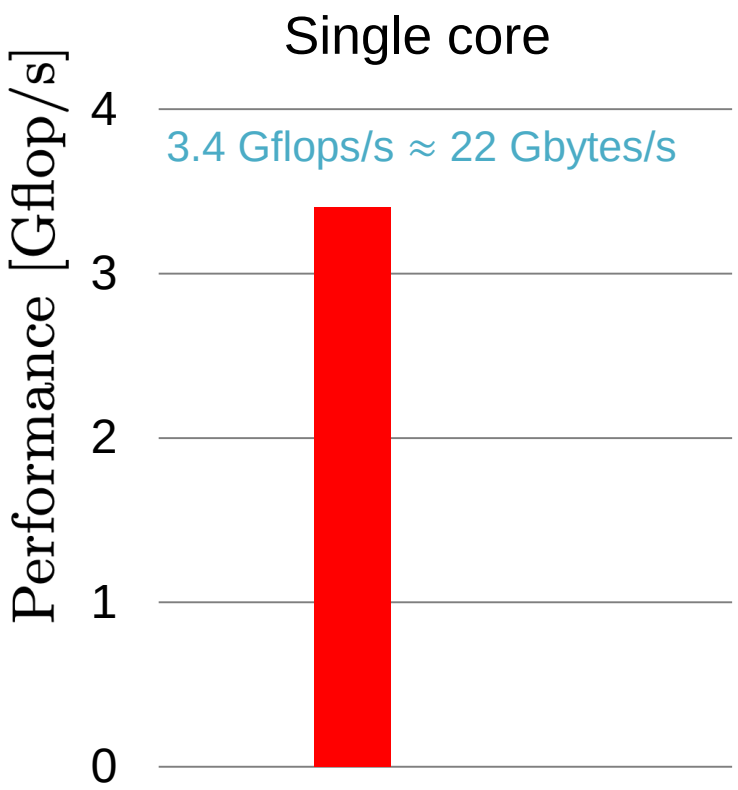


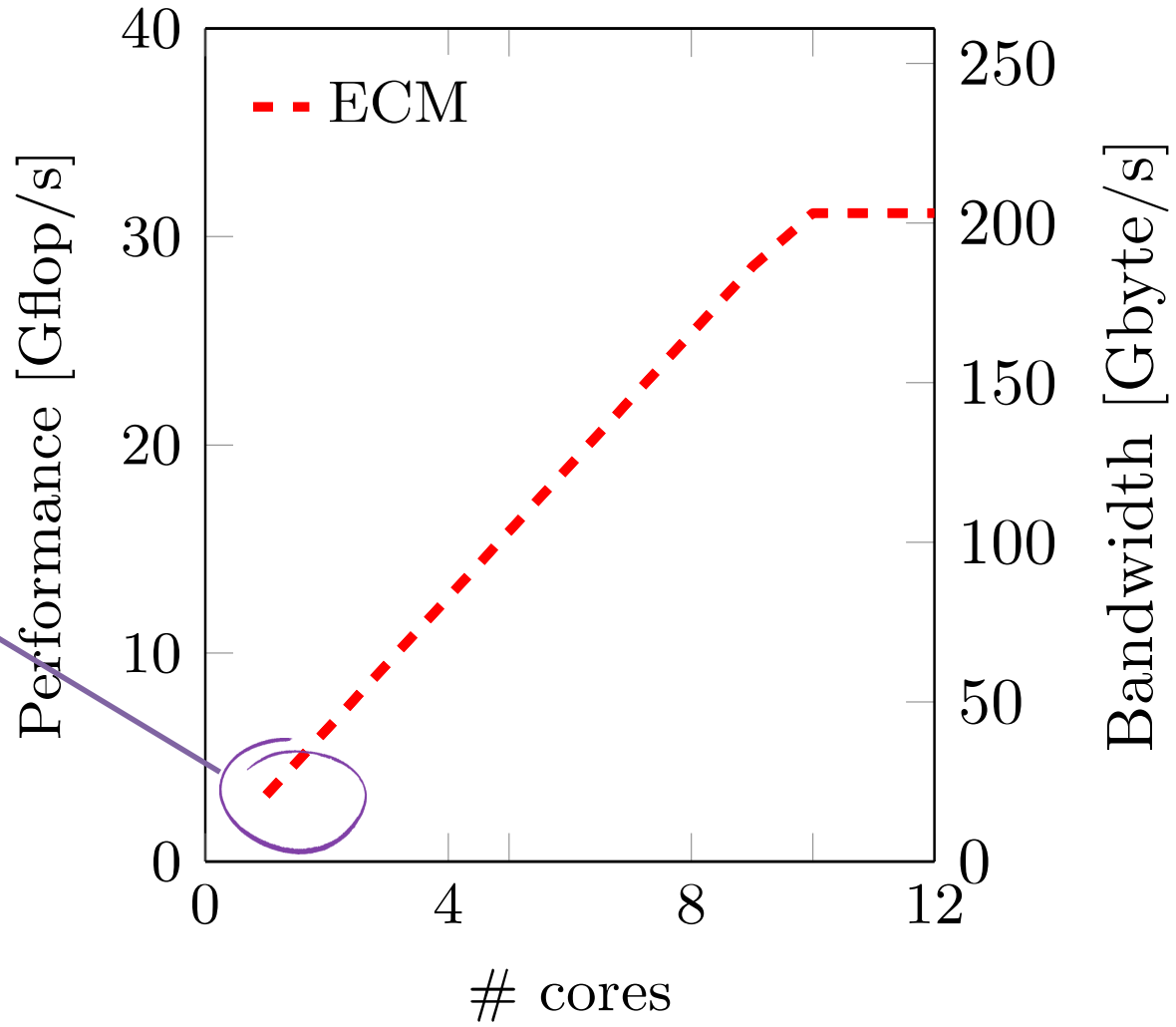
= 3.4 Gflops/s ≈ 22 Gbytes/s

ECM model SELL-C-σ

HPCG matrix, dimension 128^3

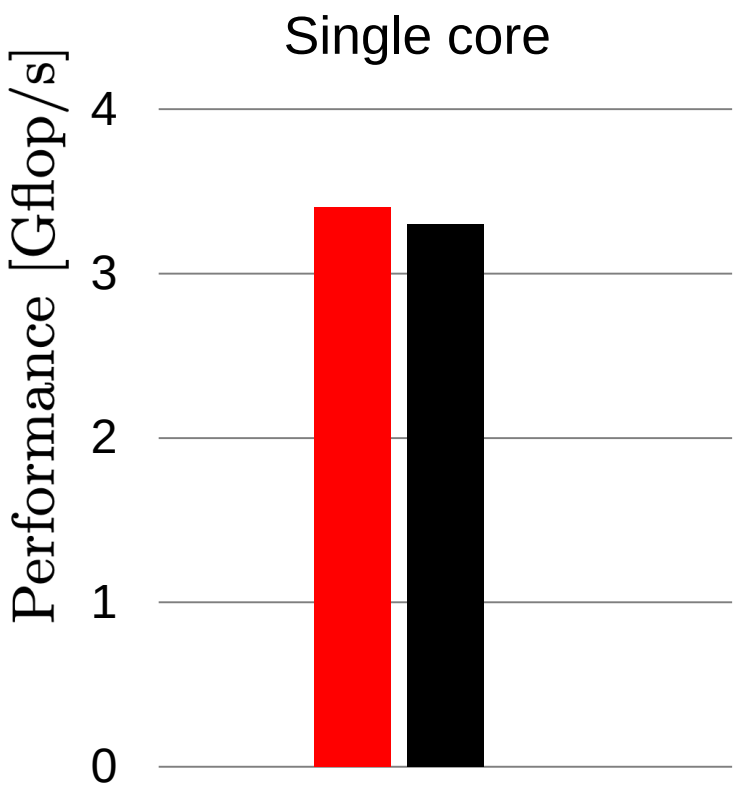
Can we saturate now ?



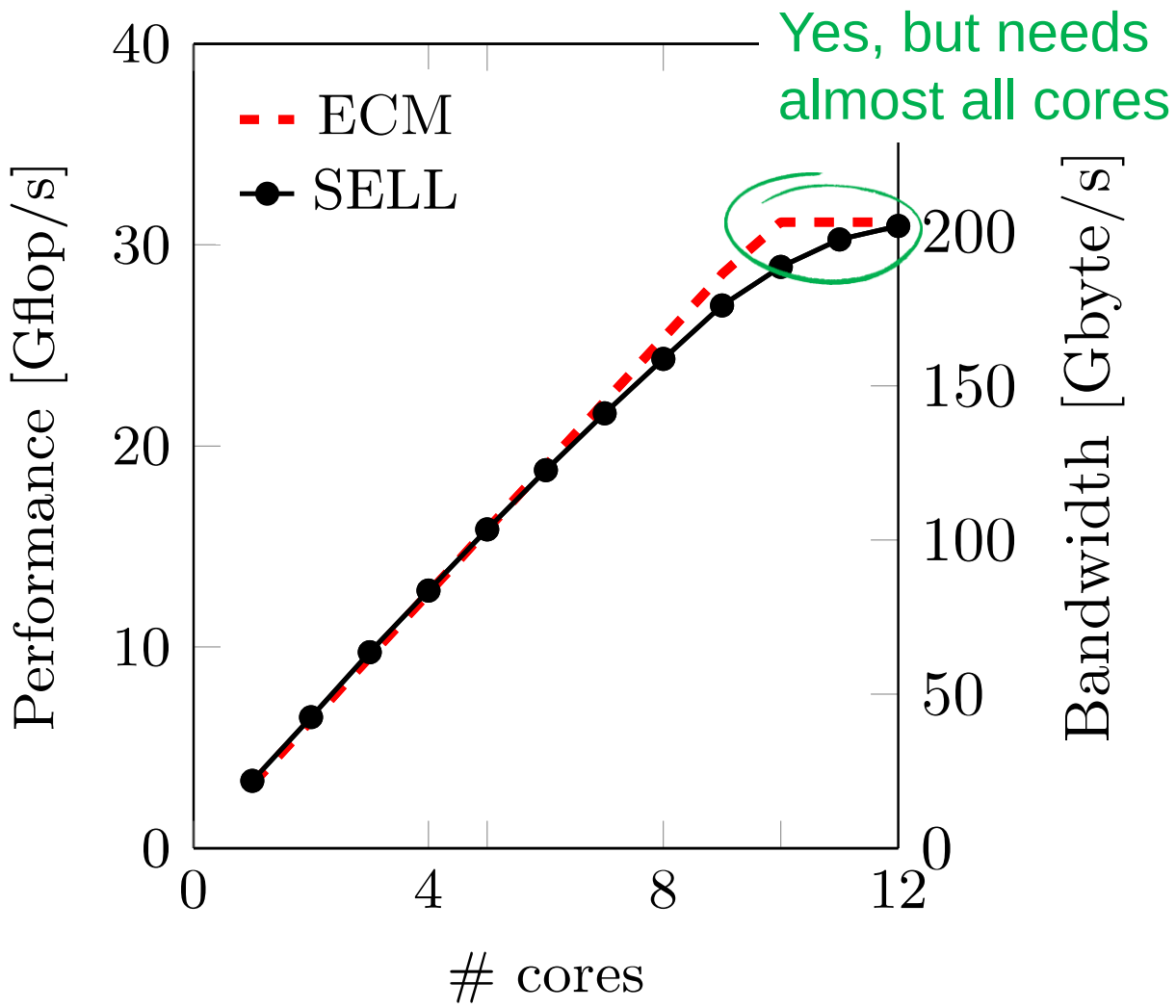


ECM model SELL-C-σ

HPCG matrix, dimension 128^3

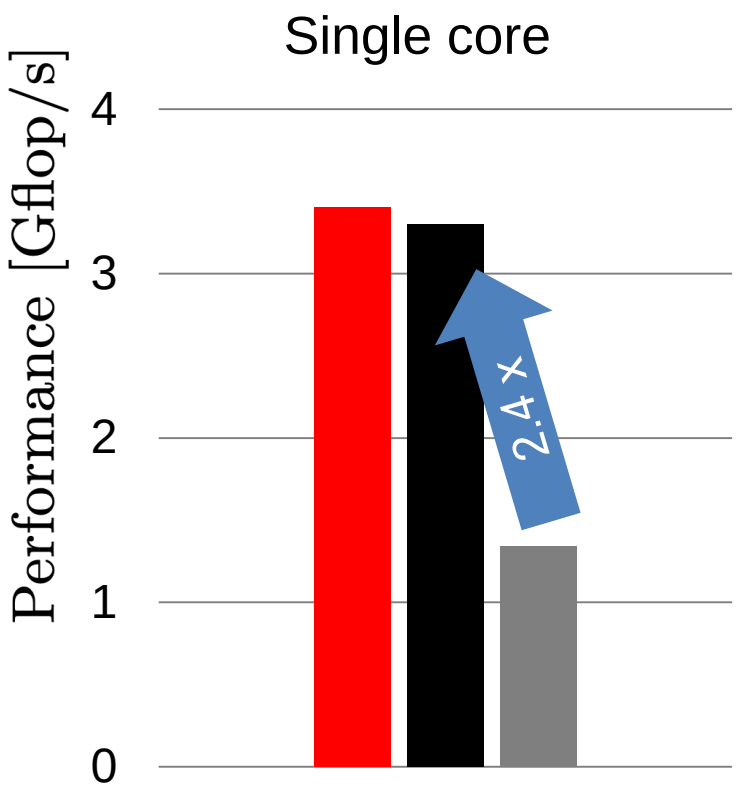


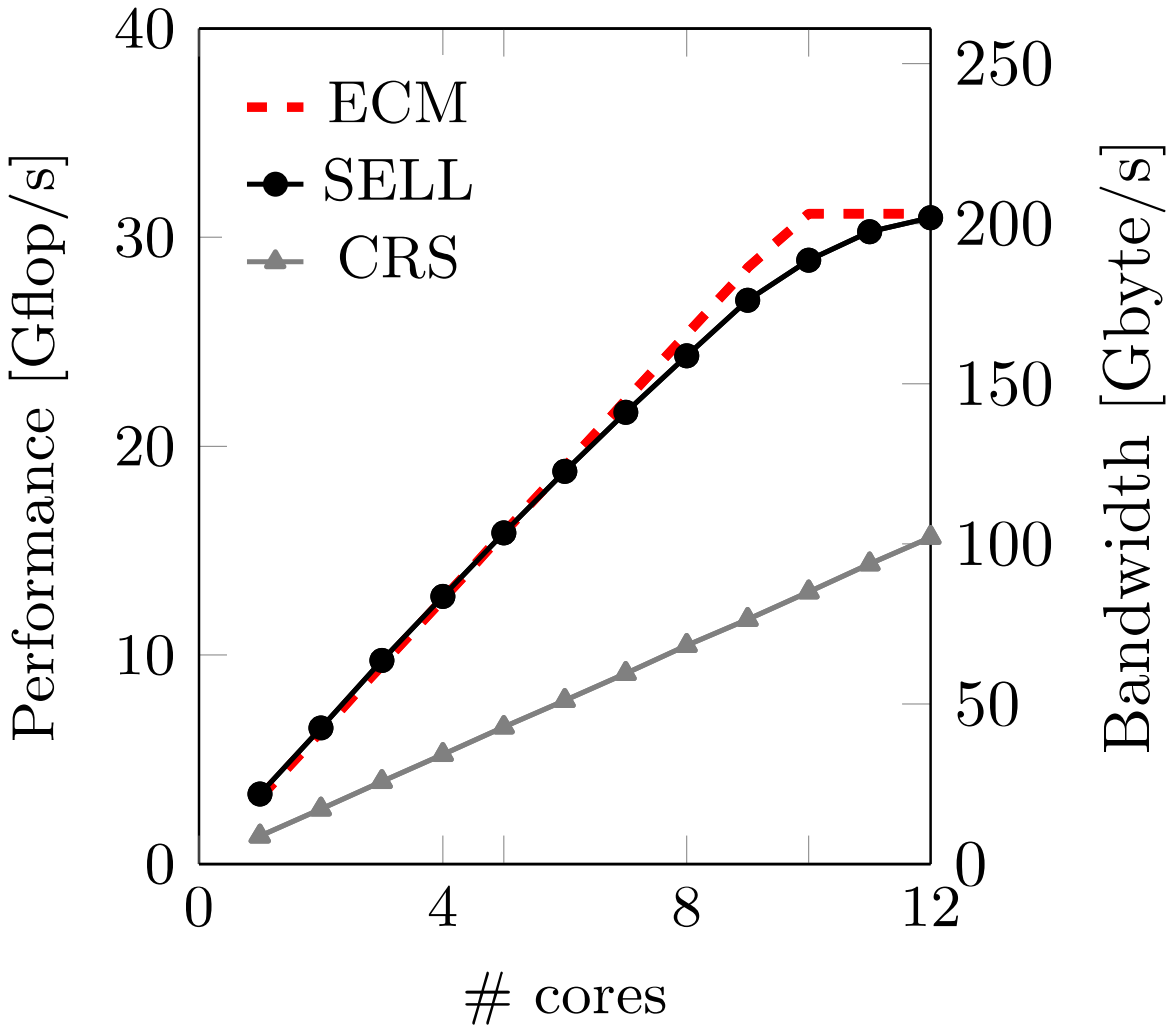
Can we saturate now ?



ECM model SELL-C-σ

HPCG matrix, dimension 128^3





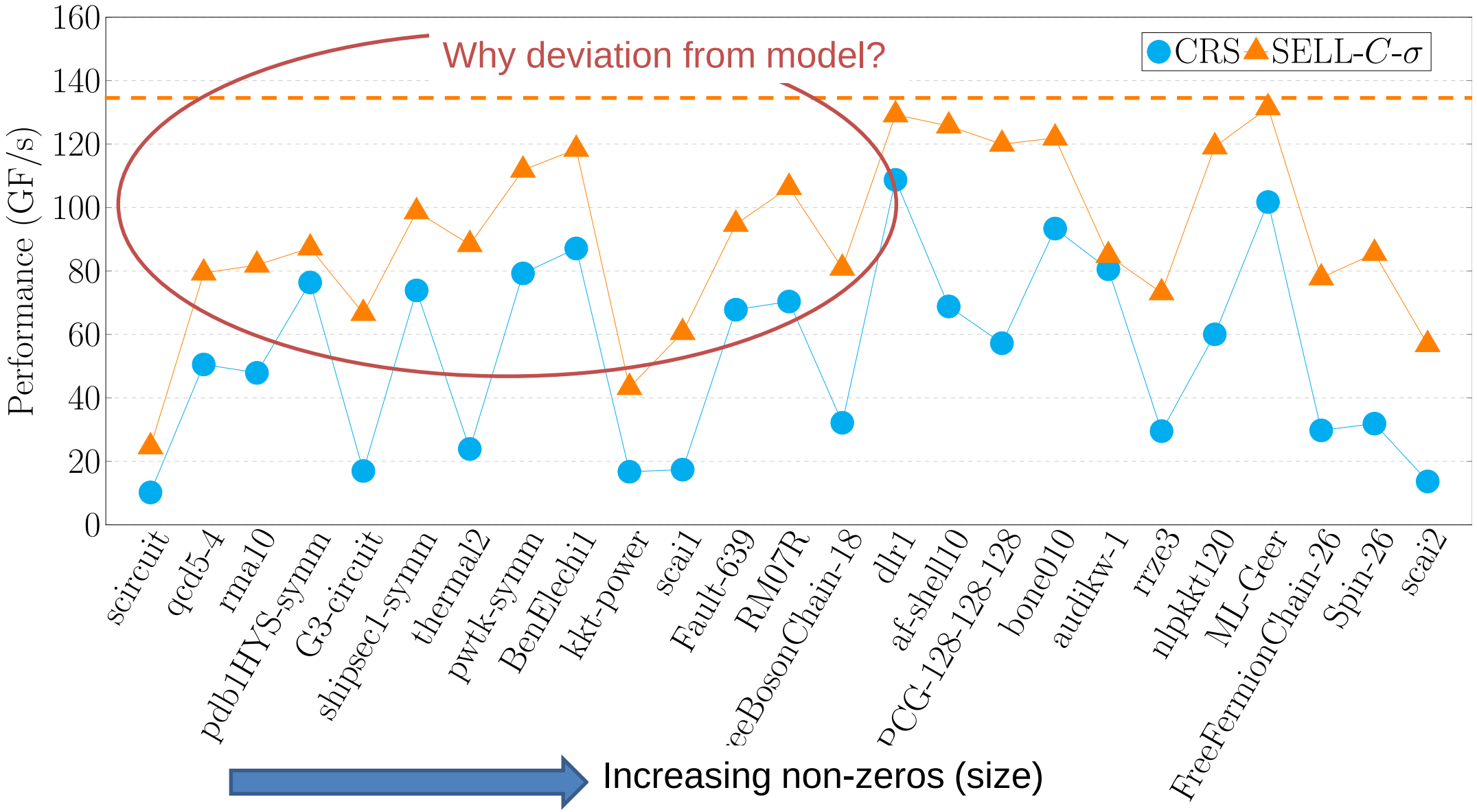
Erlangen Regional Computing Center

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

SpMV performance of FX700

High Performance Computing

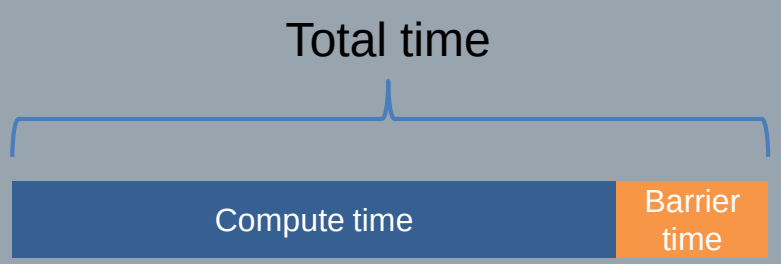
CRS vs SELL-C-σ : 1 Node



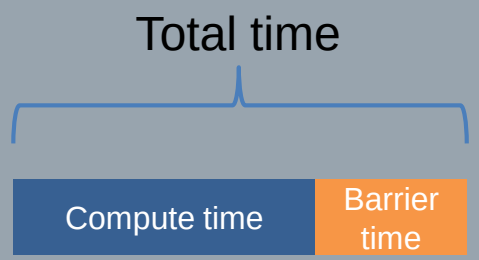
Optimistic Roofline model

I = 1/6 [B/F]

High bandwidth → more BARRIER pain

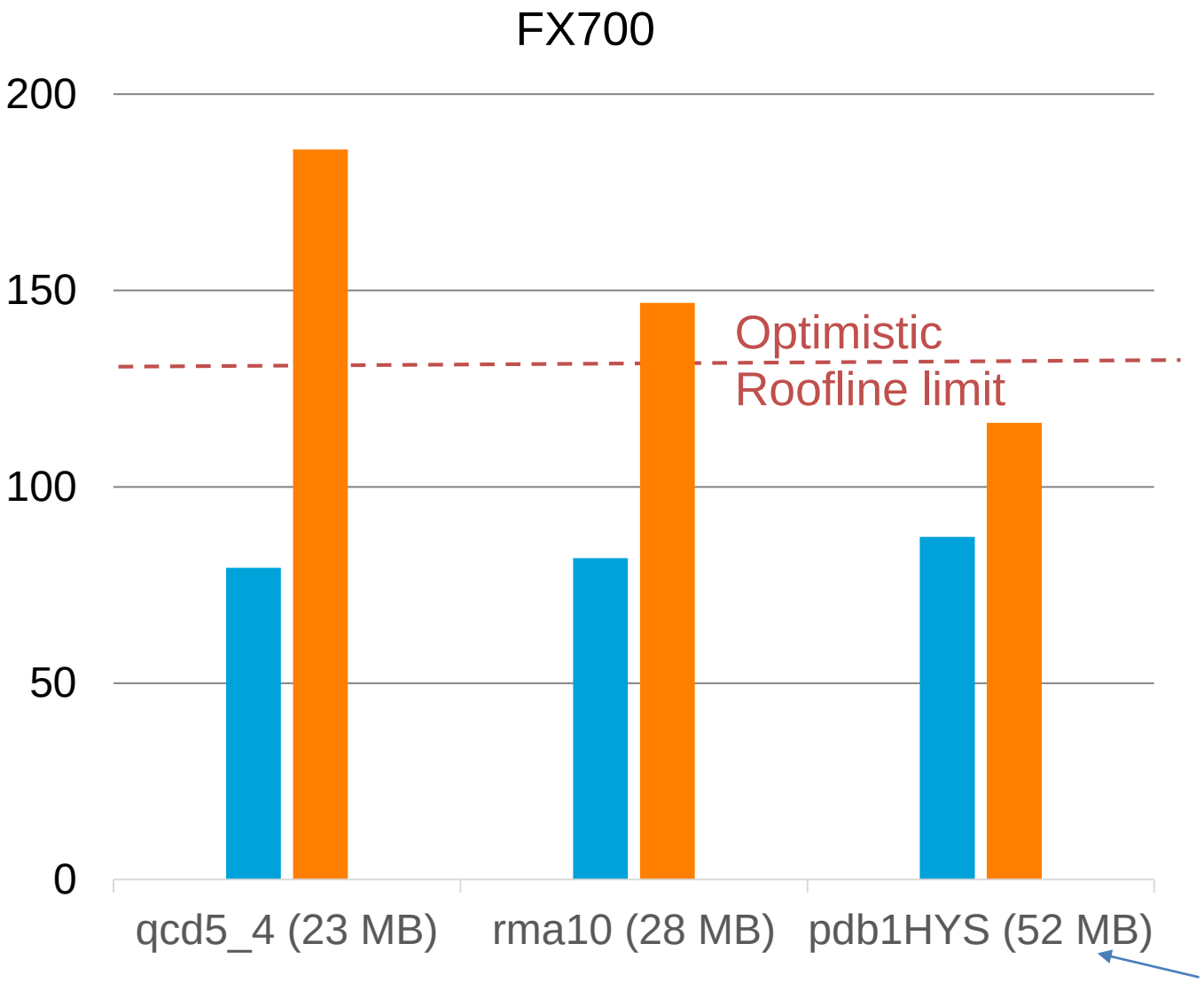


High compute time (slow memory)



Low compute time (fast memory)

BARRIER pain: SpMV



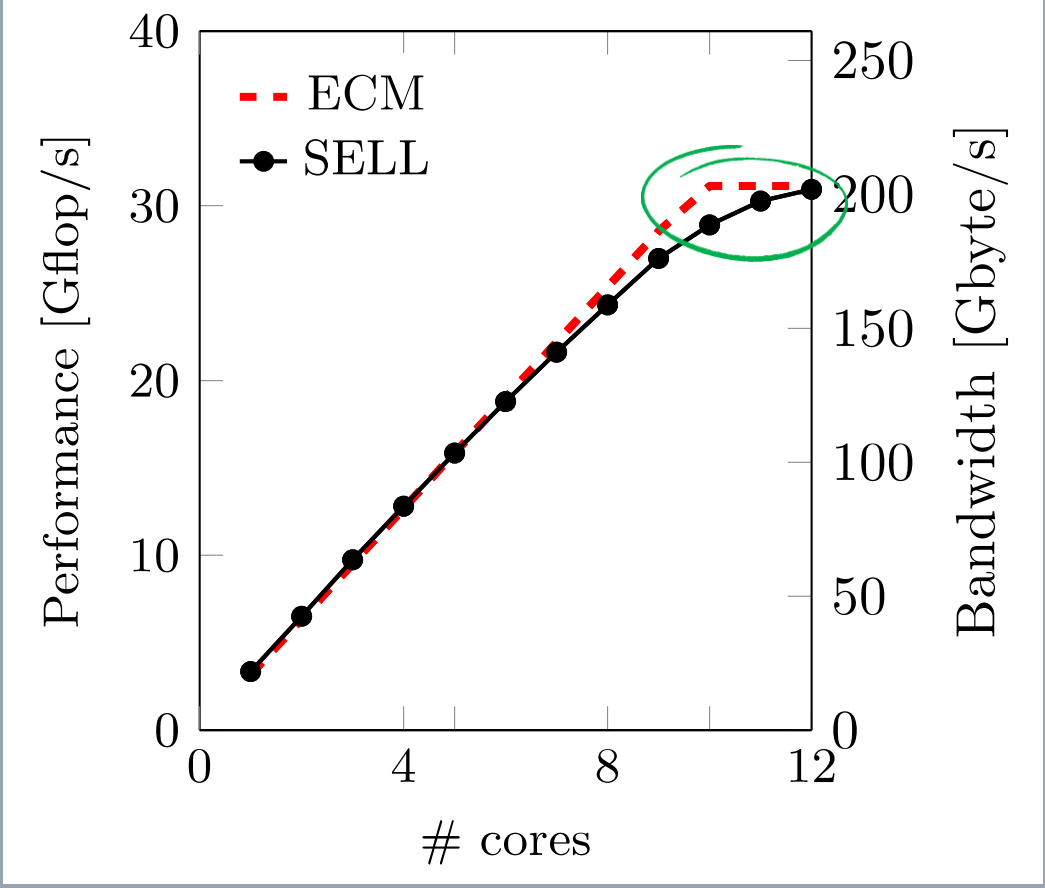
For small matrices, software barrier overhead can be hazardous.

Hardware barrier might help in these cases.

with barrier

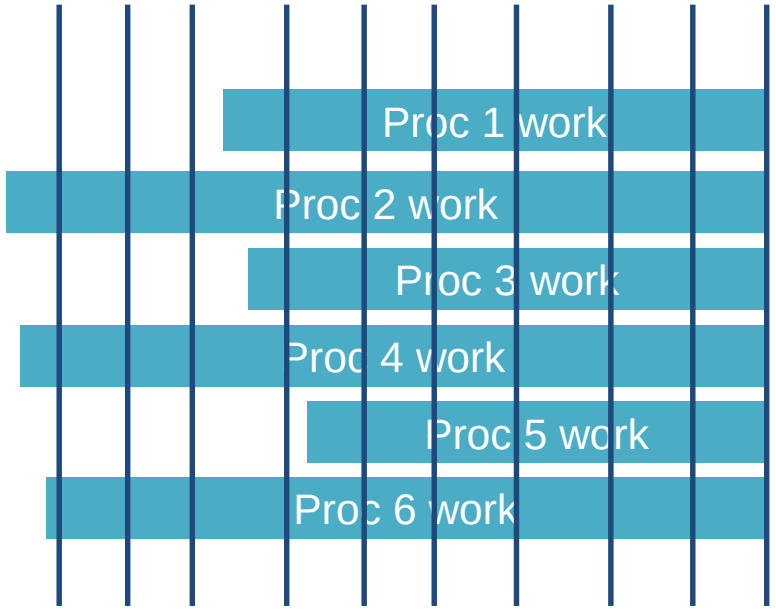
without barrier
(wrong numerics)

Consequence of last core saturation



Load balancing

Early memory saturation absorbs some load imbalance



Time = 9

Case 1: Early saturation
Memory bandwidth saturated
with 3 cores

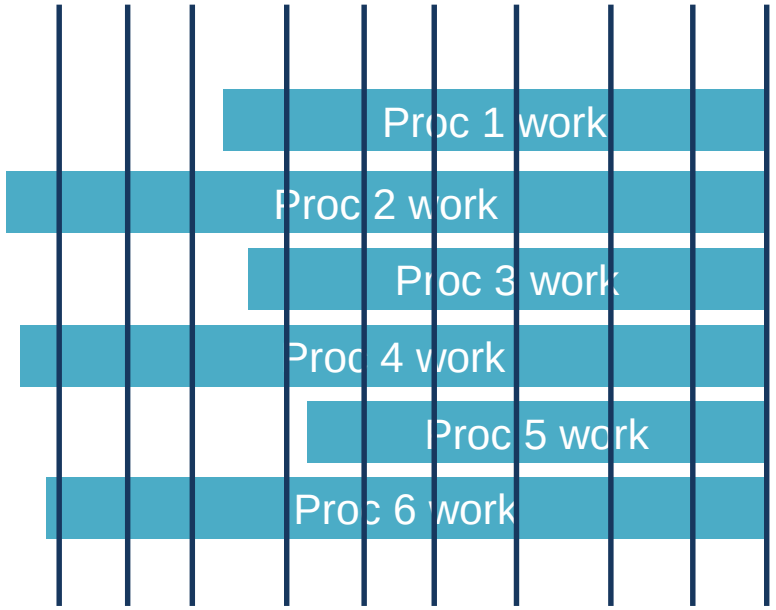
Active cores : 6

At least 3 cores are active most of the time

➔ full bandwidth ☺

Load balancing

Late saturation → Sensitive to load imbalance and irregularities



Time = 9

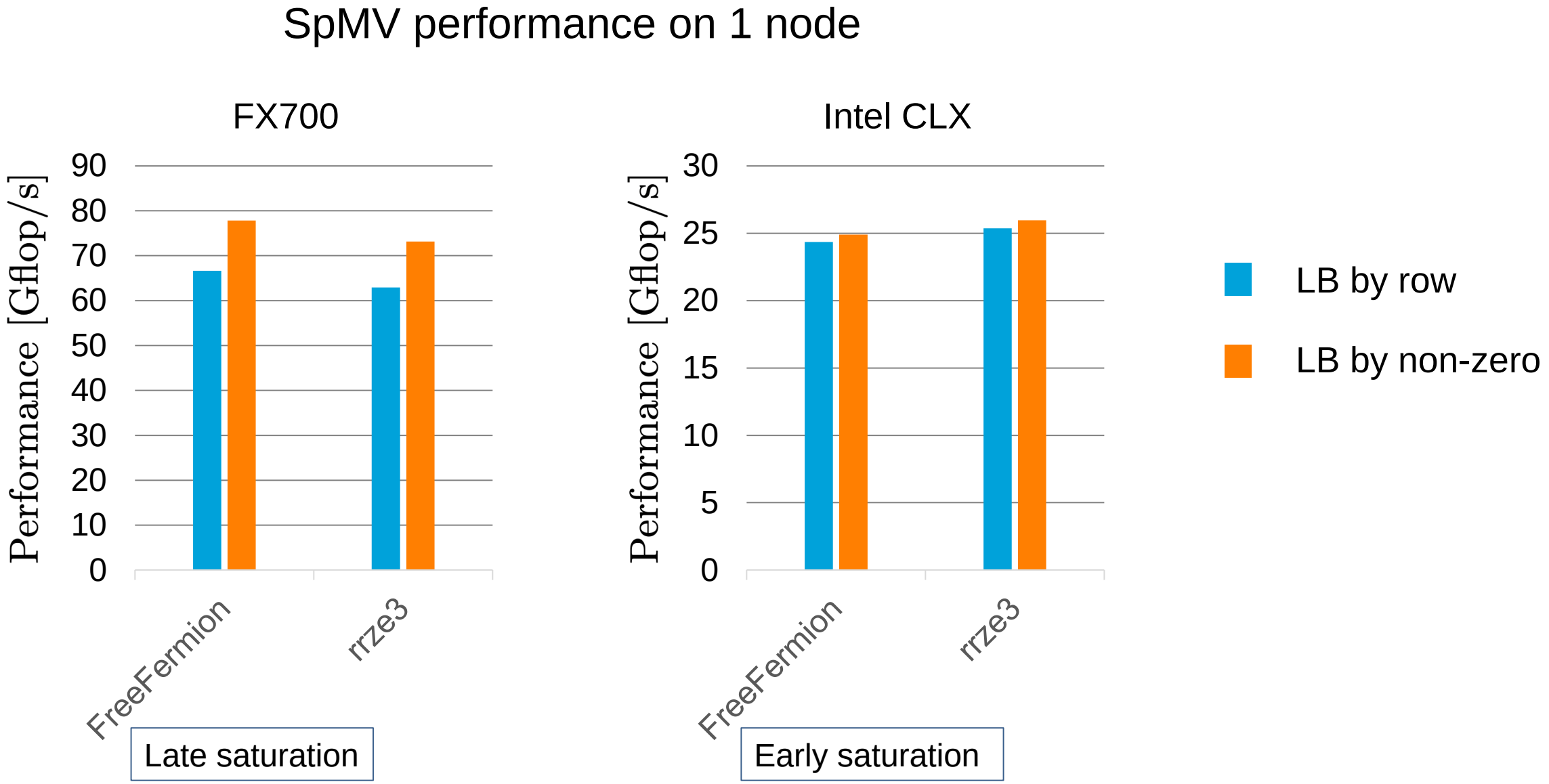
Case 2: Late saturation
Memory bandwidth saturated
with 5 cores

Active cores : 6

Less than 5 cores active

→bandwidth reduction

Load balancing (LB)



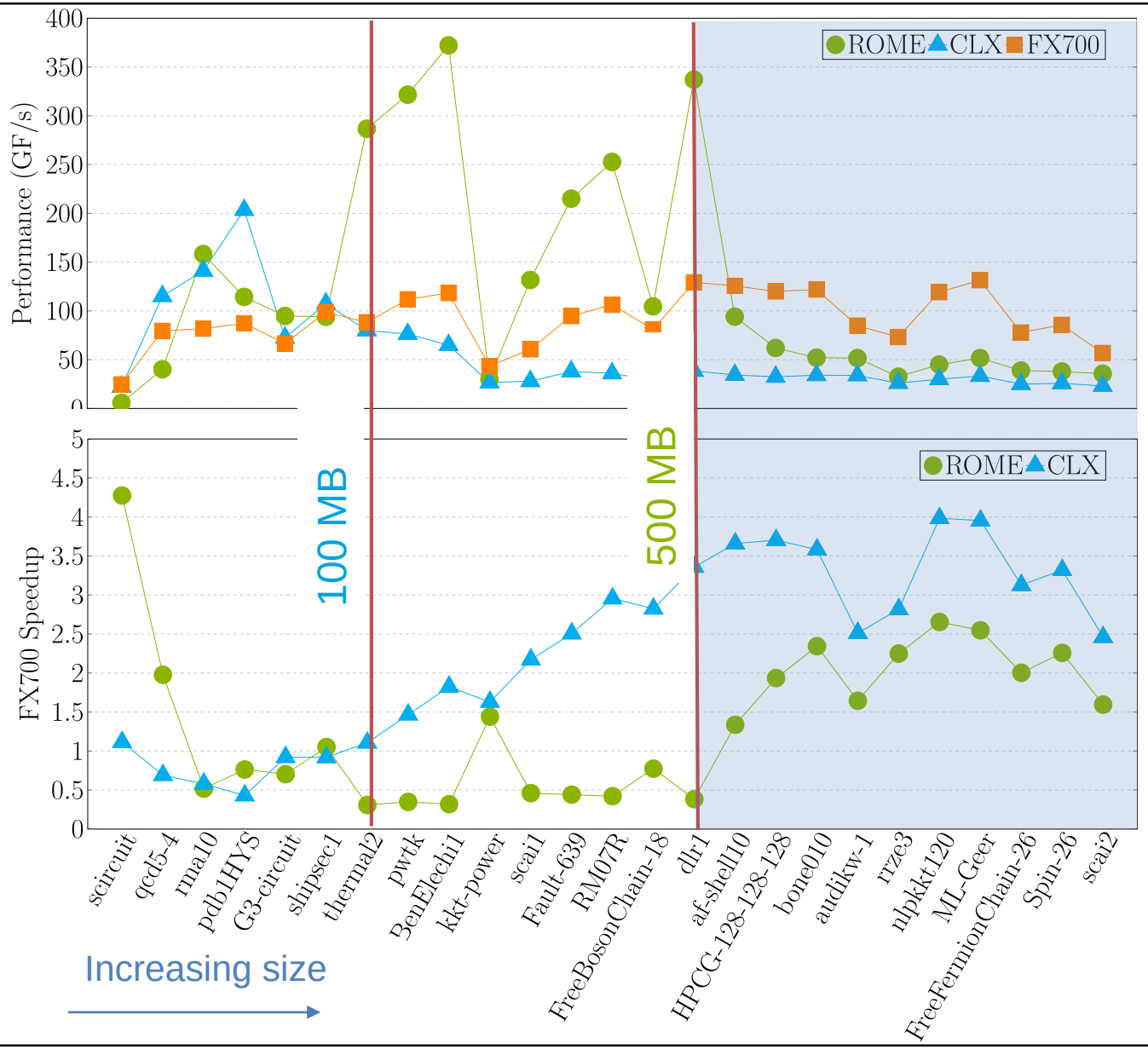
Comparison with other CPUs

1 Node FX700 : 1 x A64FX (48 c @ 1.8 GHz)

1 Node CLX : 2 x Intel Xeon Platinum 8280 (28 c @ 2.7 GHz)

1 Node ROME : 2 x AMD EPYC 7742 (64 c @ 2.25 GHz)

CPU comparison : SpMV



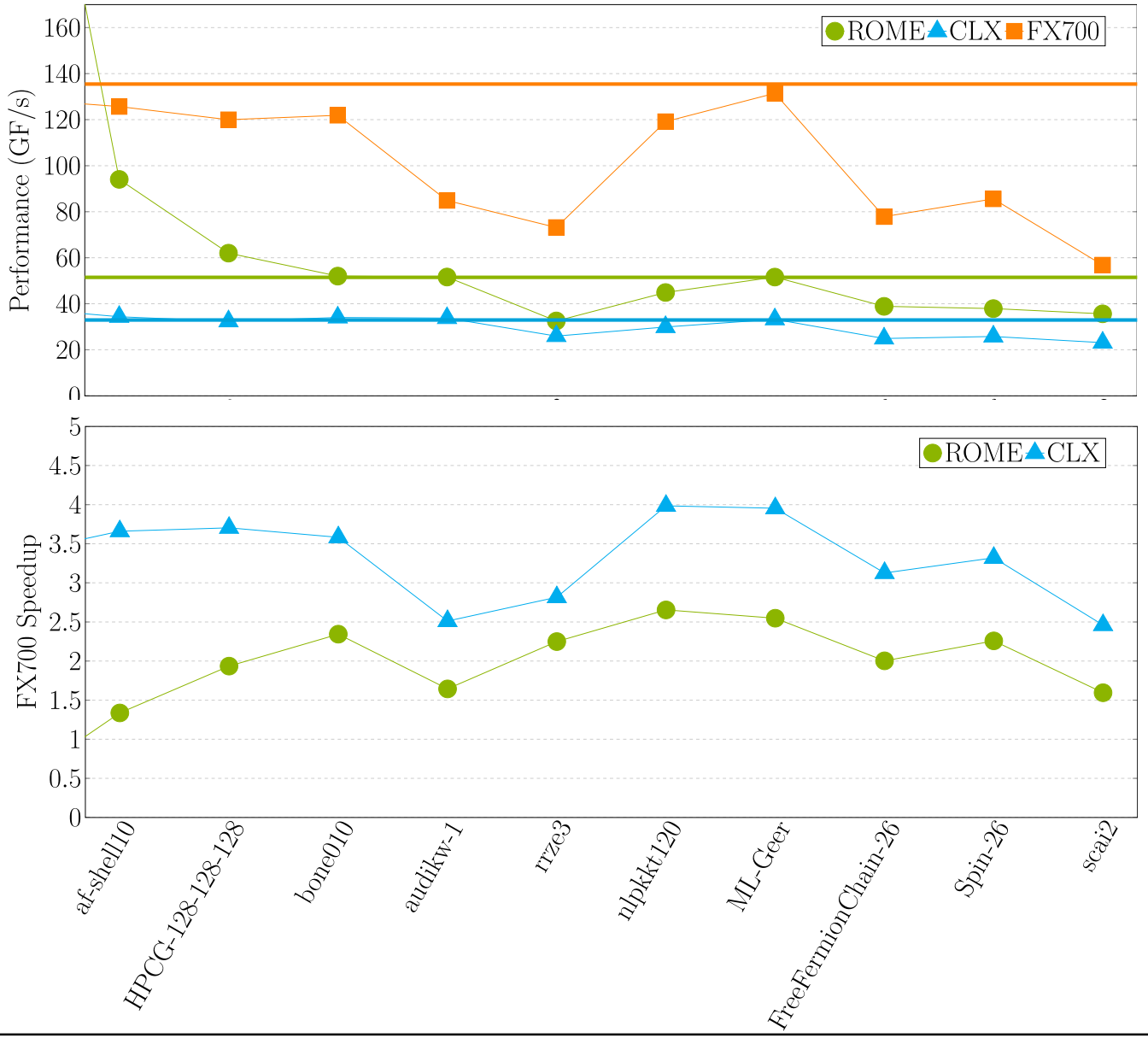
CPU models:

Fujitsu FX700
AMD ROME 7742
Intel Cascade Lake 8280

LLC cache sizes:

FX700 - 32 MB
ROME - 512 MB
CLX - 76 MB

CPU comparison – memory bound cases



ECM/Roofline limits

2× and 3× speedup over ROME and CLX.

CPU models:

Fujitsu FX700
AMD ROME 7742
Intel CLX 8280

Main memory bandwidth:

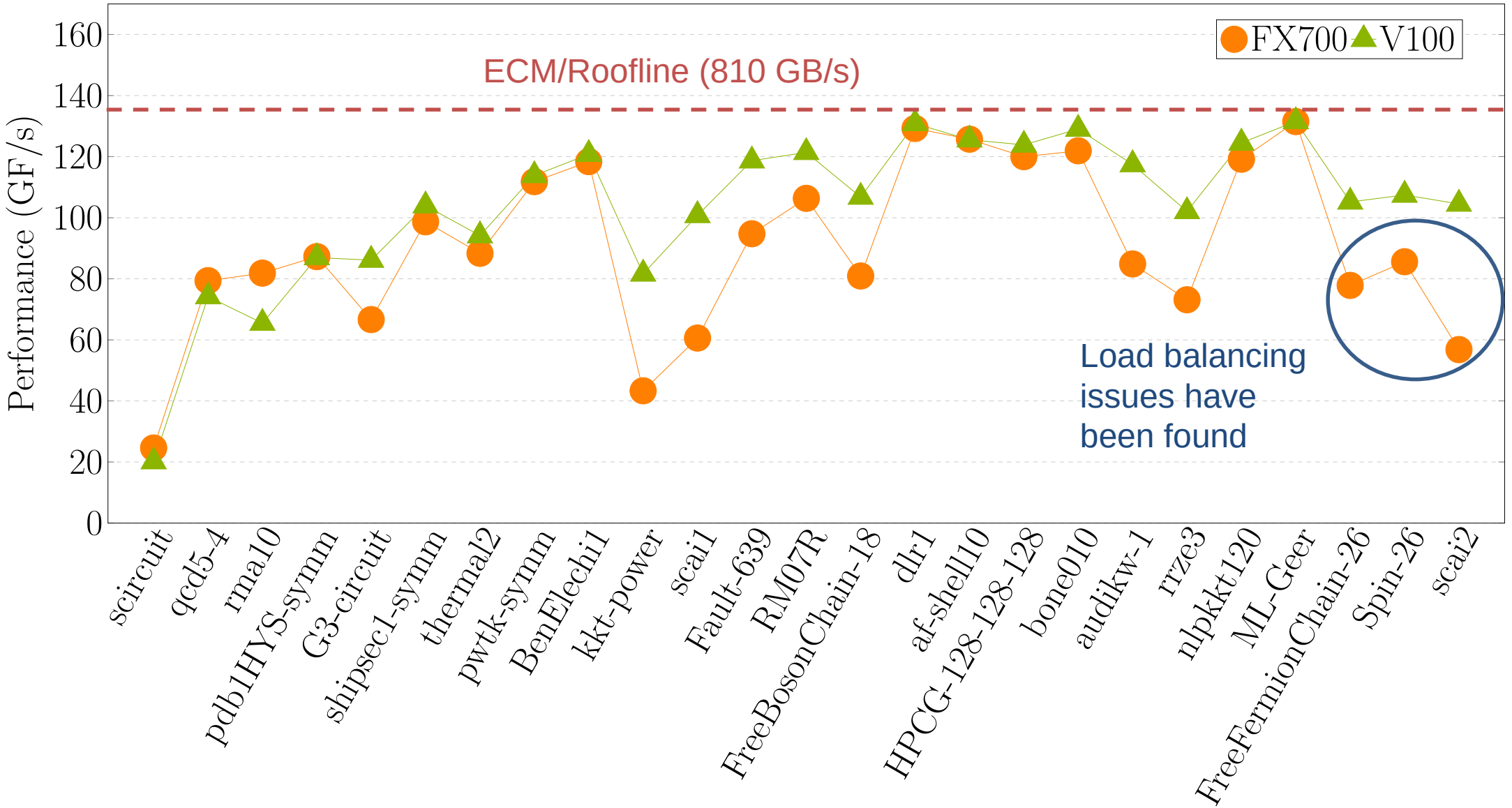
FX700 - 810 GB/s
ROME - 320 GB/s
CLX - 220 GB/s

Erlangen Regional Computing Center

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Comparison with V100 GPU

GPU comparison : SpMV



Conclusion

- ECM model was constructed and used to analyze the single core performance of FX700/A64FX.
- The partially overlapping memory hierarchy allows for high single-core memory bandwidth.
- For large matrices FX700 is 2× and 3× faster than AMD ROME and Intel CLX.
- FX700 is competitive with NVIDIA V100 GPU for SpMV.

Lessons learned

- Proper single-core optimizations required to hide long floating point latency and inefficiencies in OoO → Open-Source compilers have headroom for improvement.
- For SpMV we were able to saturate the bandwidth with the SELL-C-σ format.
- For irregular matrices proper load balancing is required for FX700.
- Barrier/synchronization cost can be painful if hardware barrier is not used.

Thank you

Questions ?

KONWIHR

HLRS
High-Performance Computing Center | Stuttgart

SFB
TRR55