

# Top-down model of a A64FX to study data layout optimization of a CFD code

AHUG@ISC22, Hamburg

Filippo Mantovani  
filippo.mantovani@bsc.es

Fabio Banchelli (\*)  
fabio.banchelli@bsc.es

Marta Garcia-Gasulla  
marta.garcia@bsc.es



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

# Emerging Technology Clusters deployed at BSC

MareNostrum 4

General purpose block

Emerging Technology Clusters (CTEs)

Intel Xeon Platinum

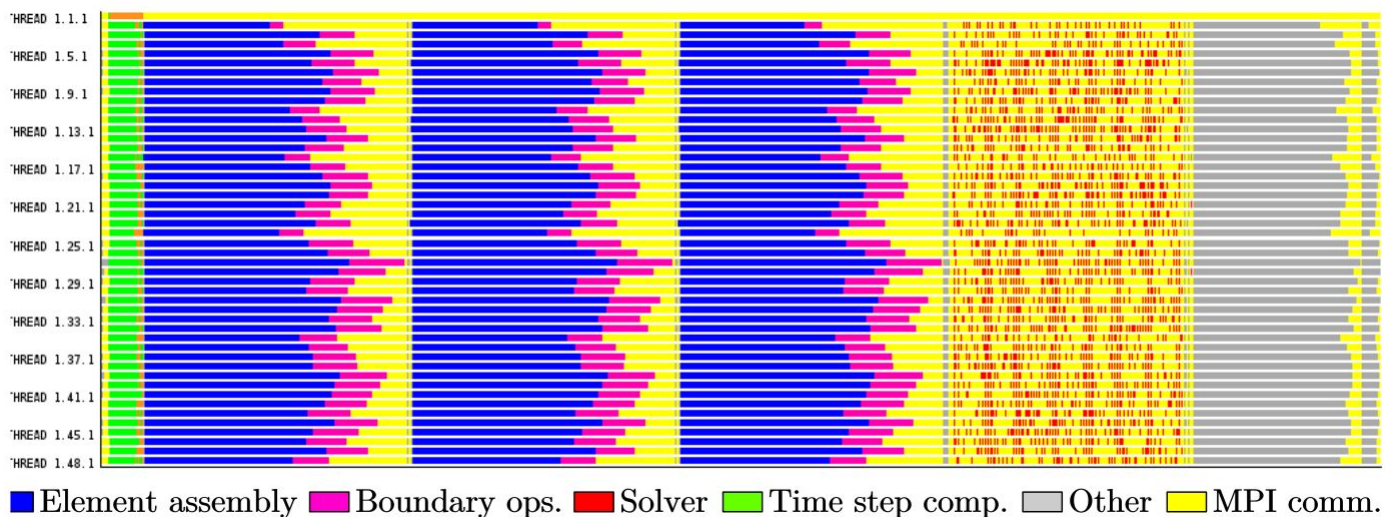
CTE Arm - A64FX processors

CTE AMD - AMD Rome + AMD Radeon Instinct MI50

CTE Power - IBM POWER9 + NVIDIA Volta GPUs

# Application under study: Alya

- Computational Fluid Dynamics code developed at BSC
- Fortran + MPI, with no external dependencies
- Iterative timesteps with distinct computational phases



# Application under study: Alya

- Implemented a data layout modification that changes the packing of mesh elements
- Introduced a compile-time parameter: **VECTOR\_SIZE**

```
Ae(in,jn) = Ae(in,jn) + Jac(ig) * N(in,ig) * N(jn,ig)
```



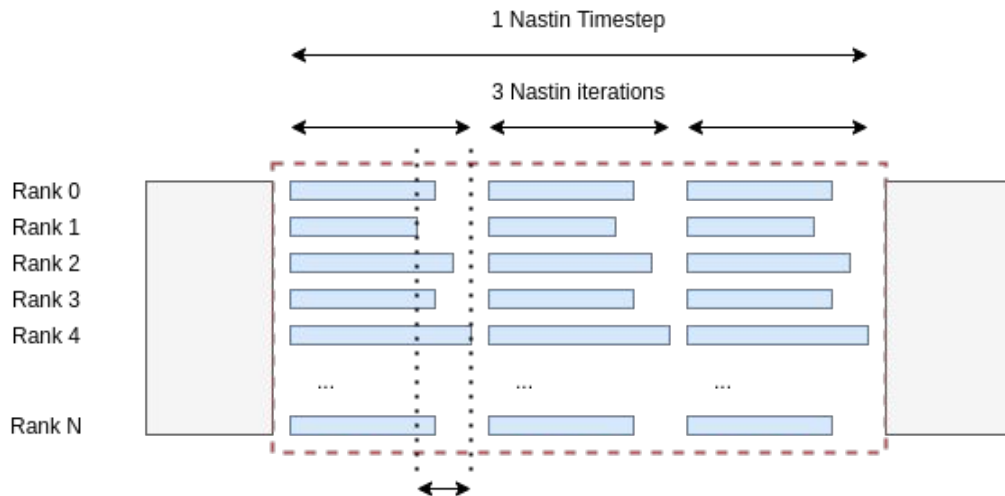
```
Ae(1:VECTOR_SIZE,in,jn) =  
  Ae(1:VECTOR_SIZE,in,jn) + Jac(1:VECTOR_SIZE,ig) * N(in,ig) * N(jn,ig)
```



How does the code modification affect the code?

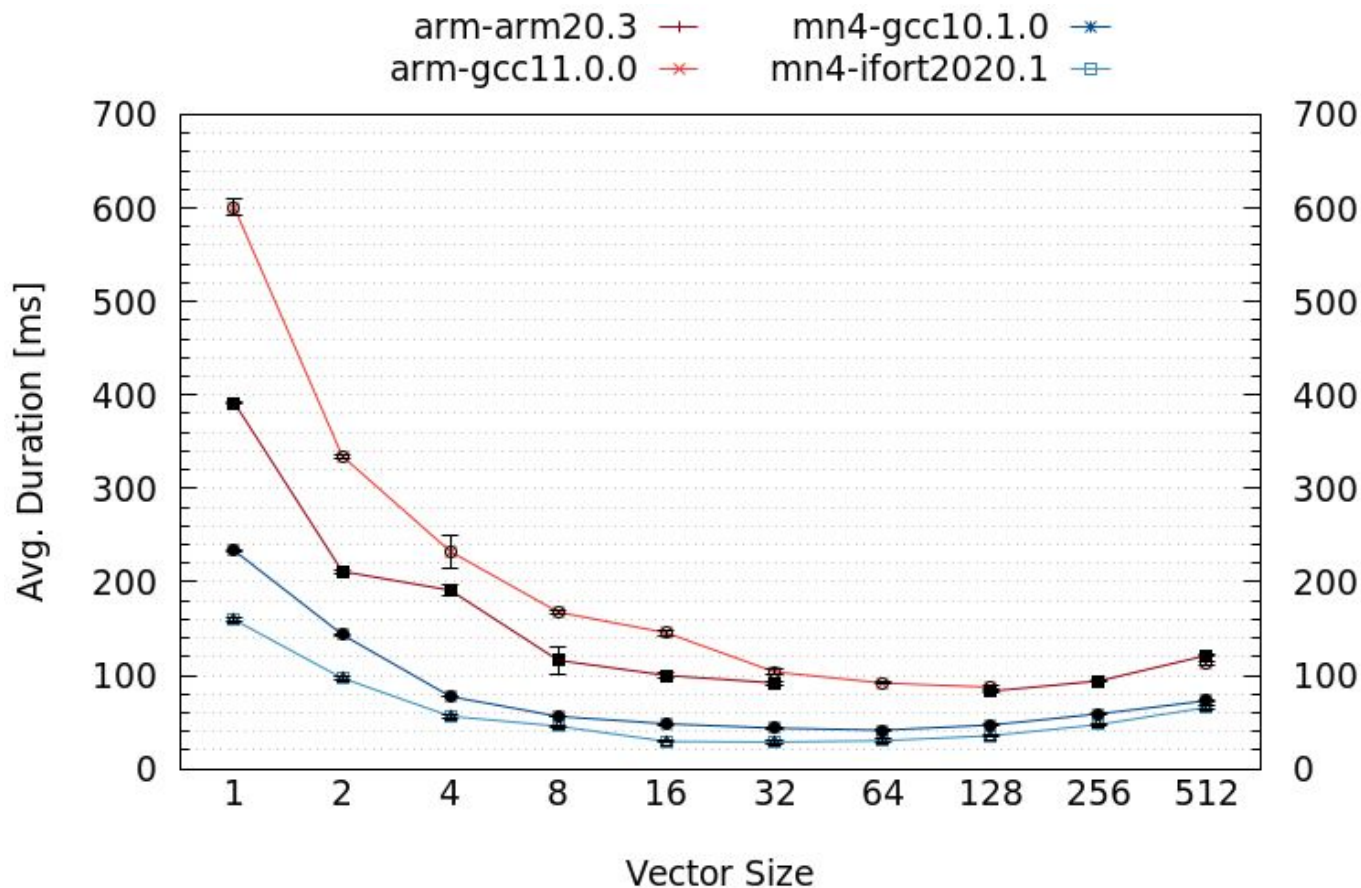
# Application under study: Alya

- Only focus on the matrix assembly code (Nastin module)
- Disregard load balance between MPI ranks
- Focus on computation bursts
- Measure metrics using timestamps and PAPI counters



**Elapsed Time**

# Average duration per cluster of bursts



# Average duration per cluster of bursts

- General trends
  - Duration decreases as VECTOR\_SIZE increases
  - Duration flattens around VECTOR\_SIZE {32, 64}
  - Duration increases for Vector Size  $\geq 128$



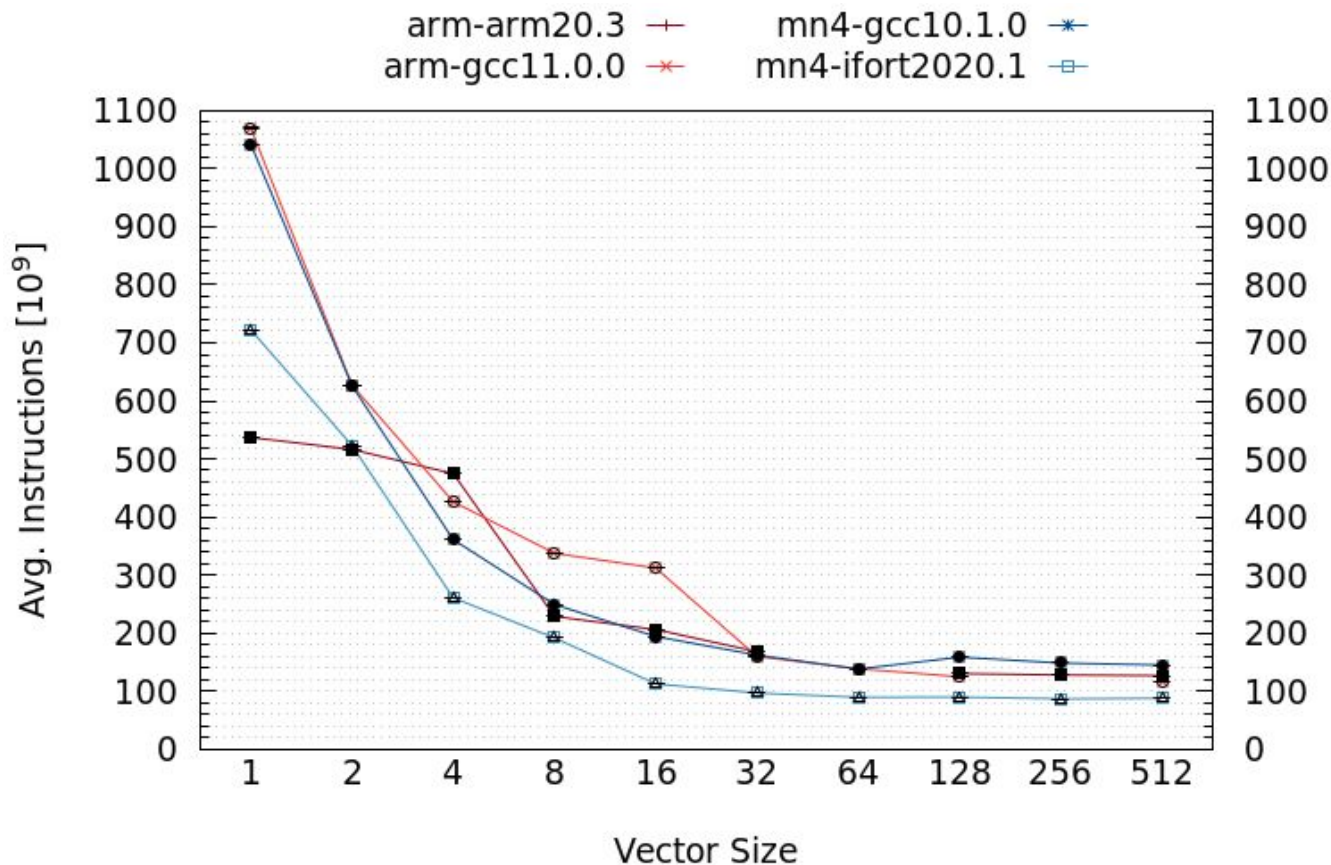
# What does affect the execution time

- Execution Time (T)
  - Number of executed Instructions (N)
  - Instructions Per Cycle (IPC)
  - Frequency (f)
    - Fixed on all machines
- } ISA and micro-architecture dependent  
Not comparable across machines

$$T = N * 1/IPC * 1/f$$

# Instructions

# Average instructions per cluster of bursts

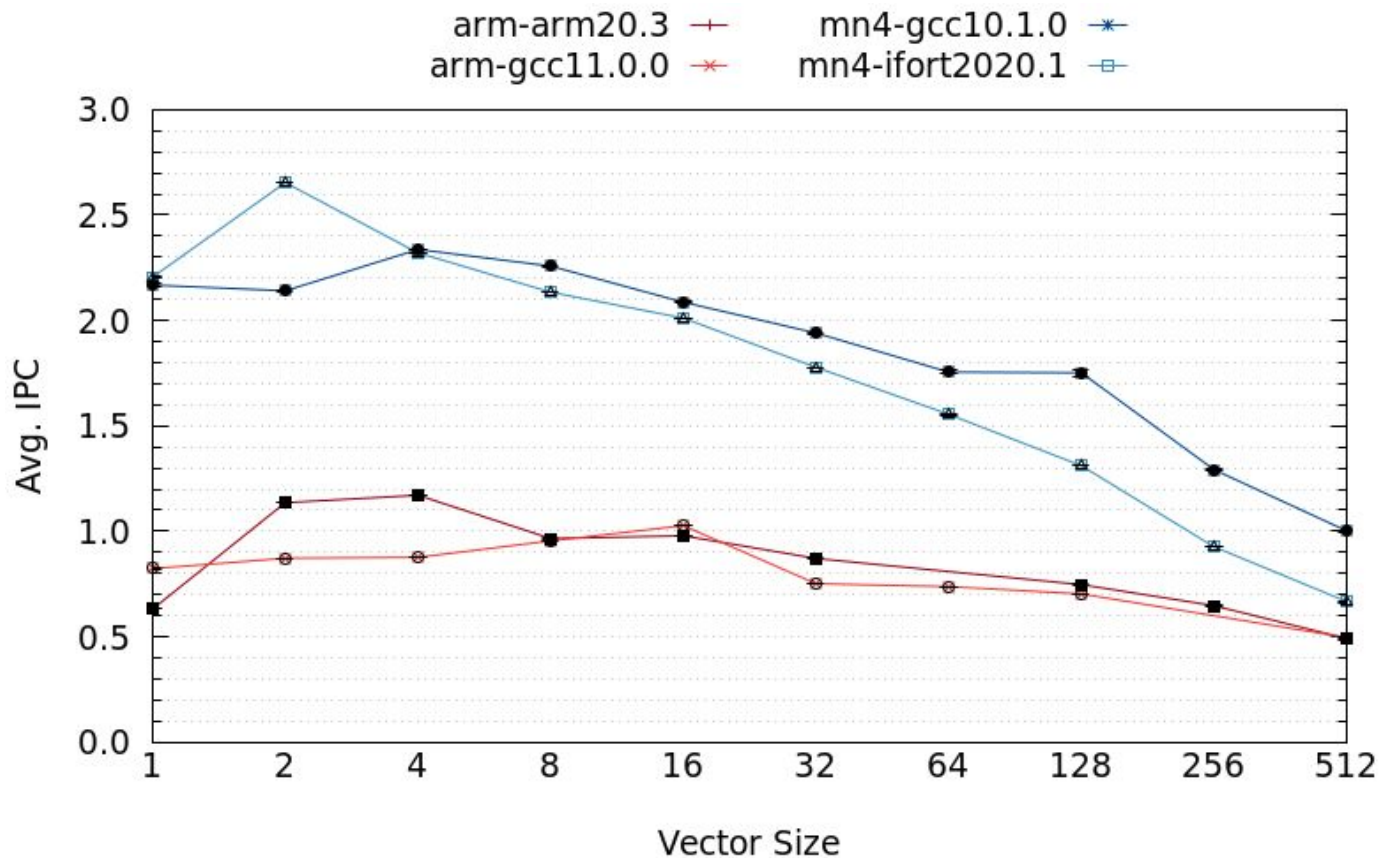


# Average instructions per cluster of bursts

- General trends
  - #Instructions decreases as VECTOR\_SIZE increases
  - Curve flattens for VECTOR\_SIZE > 64
  - There is noticeable difference between compilers in the same machine
    - In MareNostrum4 and CTE-Arm → Vendor compiler is best
- Specific case: VECTOR\_SIZE = 512
  - GNU Compiler in MareNostrum4  $\approx$  Either compiler in CTE-Arm

IPC

# Average IPC per cluster of bursts



# Average IPC per cluster of bursts

- MareNostrum4
  - Steady decrease
  - Vector Size  $\leq 32 \rightarrow$  Vendor compiler  $\approx$  GNU Compiler
  - Vector Size  $\geq 64 \rightarrow$  Vendor compiler  $<$  GNU Compiler
- CTE-Arm
  - Curve is flatter compared to MareNostrum4
  - Vendor compiler  $\approx$  GNU Compiler
  - Lowest IPC across all systems  $\rightarrow$  IPC is not comparable across machines

**Going deeper...**



# Going deeper...

- Instructions
  - *Which kind of instructions are executed the most?*
    - Study instruction mix (ISA dependent)
  - *Are there optimizations missed by the compiler?*
    - Analyze compiler remarks and generated code (Compiler dependent)
- Instructions Per Cycle
  - *Which kind of instructions take up the most cycles?*
    - Study IPC stacks / Top-Down model (ISA & uArch dependent)

# Going deeper...

- Proposal
    1. Study IPC stacks / Top-Down model
    2. Study instruction mix
    3. Study other uArch events (cache misses, stalls, etc.)
  - Outside our scope (for now)
    - Analyze compiler remarks and generated code
- } Guided by Top-Down

# Going deeper...

- Reduce exploration space
  - Select *best* compiler in each machine
  - “*Best*” → Lowest average duration
- Machine x Compiler under study
  - MareNostrum4 → Intel Compiler 2020.1
  - CTE-Arm → Arm Compiler 20.3

# Top-Down model

# Top-Down model

- Key question: *Where is my machine spending its cycles?*
  - “*where*” can be attributed to instructions and/or hardware resources
- Model
  - Hierarchical metrics
  - Drill down the path that represents the limiting factor
- Method
  - Compute metrics by reading performance counters

# Top-Down model in MareNostrum4

- Directly ported from:

A Top-Down method for performance analysis and counters architecture

<https://doi.org/10.1109/ISPASS.2014.6844459>

- Able to construct Top-Down model

-  First level

-  Second level

-  Memory Bound

-  Compute Bound (missing counters)

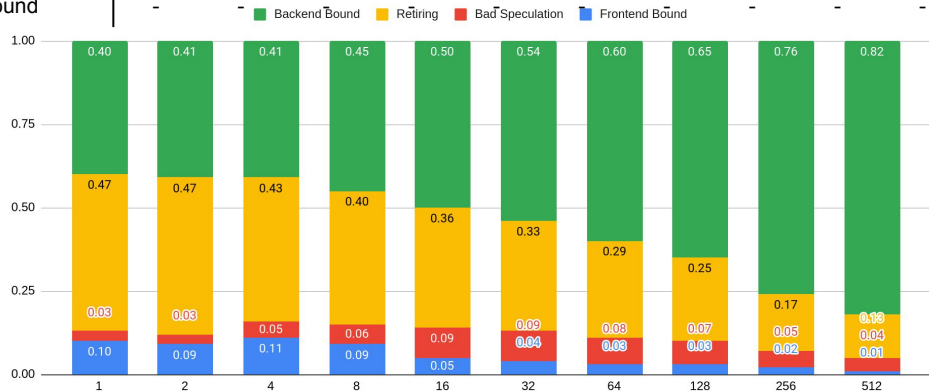
# Top-Down model in MareNostrum4

- Comments
  - First level → Metrics add up to 1 (relative to total *slots*)
  - Second level → Metrics do not add up to 1
  - Requires 2 counter sets
- Depends on micro-architectural parameters
  - #uops issued per cycle (*slots*) → 6
  - #cycles lost because misprediction → 6

# Top-Down model in MareNostrum4

- Vector Size [1,8]
  - Retiring, Backend Bound
- Vector Size > 8
  - Backend Bound
  - Memory Bound
  - L1 Bound stays flat
  - L2 Bound increases
- Vector Size 512
  - L3 Bound increases

Vector Size	1	2	4	8	16	32	64	128	256	512
Frontend Bound	0.10	0.09	0.11	0.09	0.05	0.04	0.03	0.03	0.02	0.01
Bad Speculation	0.03	0.03	0.05	0.06	0.09	0.09	0.08	0.07	0.05	0.04
Retiring	0.47	0.47	0.43	0.40	0.36	0.33	0.29	0.25	0.17	0.13
Backend Bound	0.40	0.41	0.41	0.45	0.50	0.54	0.60	0.65	0.76	0.82
Memory Bound	0.06	0.08	0.09	0.12	0.21	0.25	0.31	0.42	0.62	0.75
L1 Bound	0.04	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.05
L2 Bound	0.05	0.06	0.08	0.10	0.21	0.30	0.37	0.37	0.33	0.28
L3 Bound	0.01	-	-	0.03	-	0.04	0.04	0.05	0.09	0.18
#L3HitFraction	0.01	-	-	0.01	-	0.01	0.01	0.01	0.01	0.01
Ext. Mem. Bound	-	-	-	-	-	-	-	-	-	-
Stores Bound	?	?	?	?	?	?	?	?	?	?
Compute Bound	-	-	-	-	-	-	-	-	-	-

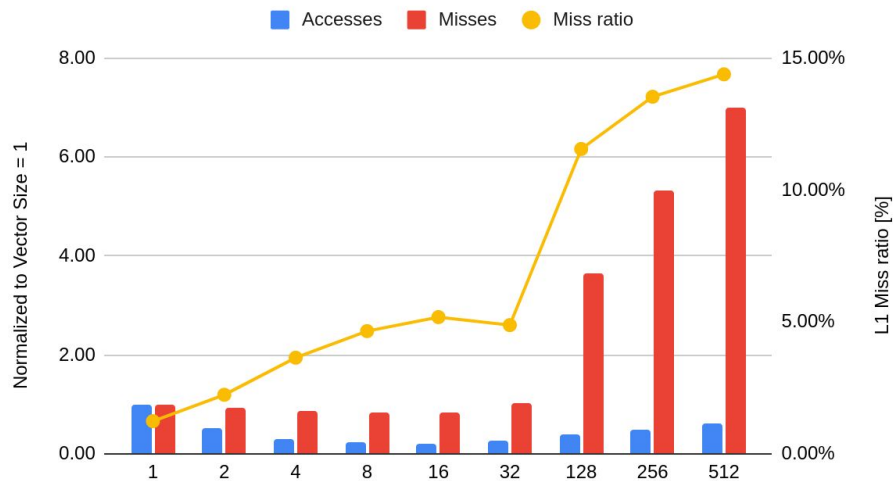




# Going deeper...

- Losing cycles because of L2 misses (L3 seems an issue for vector size > 512)...
- Possible reasons:
  - $\Delta$  Number of cache accesses ( $\text{PAPI\_L2\_TCA}$ )
  - $\Delta$  Cache miss ratio  
( $\text{PAPI\_L2\_TCM} / \text{PAPI\_L2\_TCA}$ )

Nastin - L2 Accesses



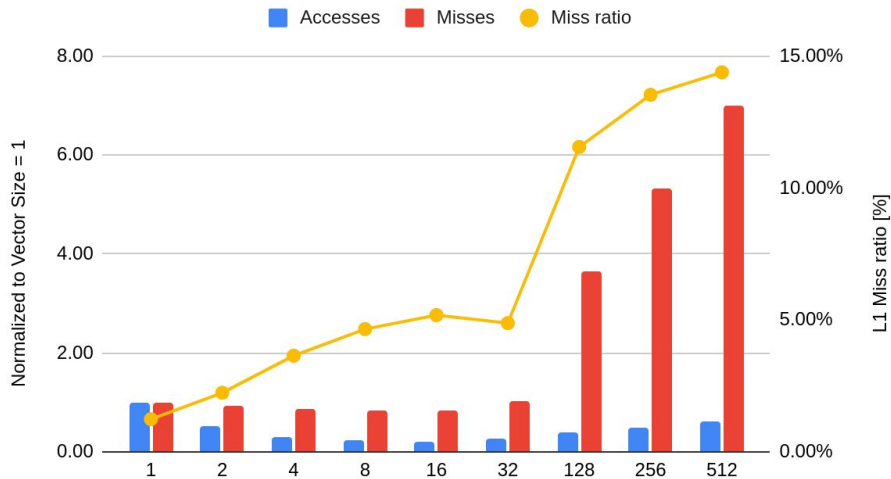
# Going deeper...

- Losing cycles because of L2 misses (L3 seems an issue for vector size > 512)...
- Possible reasons:

✗ –  $\Delta$  Number of cache accesses ( $PAPI\_L2\_TCA$ )

✓ –  $\Delta$  Cache miss ratio  
( $PAPI\_L2\_TCM / PAPI\_L2\_TCA$ )

Nastin - L2 Accesses



# Top-Down model in CTE-Arm

- Constructed following hierarchy found in:

A64FX Microarchitecture Manual

<https://github.com/fujitsu/A64FX/tree/master/doc>

- Able to construct Top-Down-*like* model
  -  Three levels

# Top-Down model in CTE-Arm

- Comments
  - Different metrics from original Top-Down model
  - Metrics of each level add up to 1 (relative to parent node in hierarchy)
  - Hierarchy is centered around cycles where no instructions were committed
  - Requires 3 counter sets
- Does not depend on micro-architectural parameters

# Top-Down model in CTE-Arm

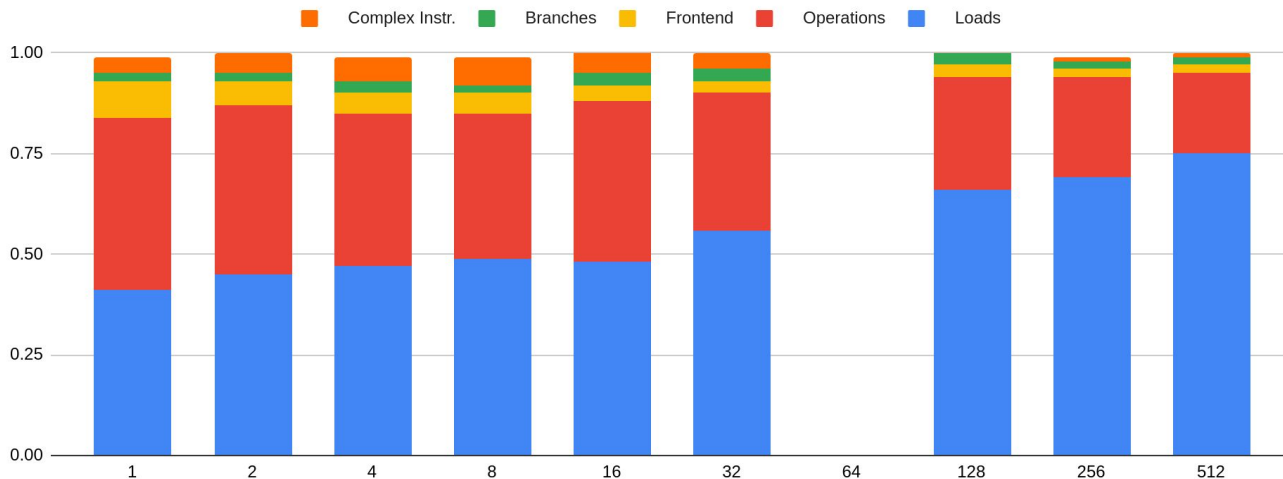
- Complex Instruction (UOP\_ONLY\_COMMIT)
  - Cycles where one or more uops were completed, but not a whole instruction
- Branches (BR\_COMP\_WAIT)
  - Waiting for branch direction determination
- Frontend (ROB\_EMPTY)
  - Waiting for instructions to fill the ROB
- Operations (EU\_COMP\_WAIT)
  - Waiting for operations to complete (instruction latency)
- Loads (LD\_COMP\_WAIT)
  - Waiting for memory access to complete

# Top-Down model in CTE-Arm

Vector Size	1	2	4	8	16	32	64	128	256	512
4 Commits / cycle	0.08	0.19	0.20	0.15	0.14	0.12	-	0.12	0.10	0.08
3 Commits / cycle	0.05	0.06	0.06	0.06	0.07	0.07	-	0.04	0.04	0.03
2 Commits / cycle	0.05	0.06	0.04	0.05	0.05	0.05	-	0.05	0.04	0.03
1 Commits / cycle	0.08	0.08	0.10	0.09	0.10	0.09	-	0.08	0.07	0.06
0 Commits / cycle	0.75	0.62	0.60	0.65	0.64	0.68	-	0.73	0.76	0.82
└ Loads	0.41	0.45	0.47	0.49	0.48	0.56	-	0.66	0.69	0.75
└ L1 Miss	0.33	0.50	0.45	0.48	0.45	0.47	-	0.39	0.41	0.48
└ L2 Miss	0.14	0.21	0.17	0.23	0.20	0.27	-	0.37	0.35	0.45
└ L2 Prefetch	0.04	0.08	0.08	0.09	0.09	0.18	-	0.16	0.13	0.08
└ Prefetch ports	0.00	0.00	0.00	0.00	0.00	0.00	-	0.00	0.00	0.00
└ Instr. Latency	0.49	0.21	0.30	0.20	0.26	0.08	-	0.08	0.11	-0.01
└ <b>Total</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>0.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
└ Operations	0.43	0.42	0.38	0.36	0.40	0.34	-	0.28	0.25	0.20
└ Frontend	0.09	0.06	0.05	0.05	0.04	0.03	-	0.03	0.02	0.02
└ Branches	0.02	0.02	0.03	0.02	0.03	0.03	-	0.03	0.02	0.02
└ Complex Instr.	0.04	0.05	0.06	0.07	0.06	0.04	-	0.01	0.01	0.01
└ <b>Total</b>	<b>0.99</b>	<b>1.00</b>	<b>0.99</b>	<b>0.99</b>	<b>1.01</b>	<b>1.00</b>	<b>0.00</b>	<b>1.01</b>	<b>0.99</b>	<b>1.00</b>
<b>Total</b>	<b>1.01</b>	<b>1.01</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.01</b>	<b>0.00</b>	<b>1.02</b>	<b>1.01</b>	<b>1.02</b>

# Top-Down model in CTE-Arm

- “Operations”  $\approx$  “Retiring”  $\rightarrow$  Same behavior as in MareNostrum4
- “Loads” becomes the dominant factor



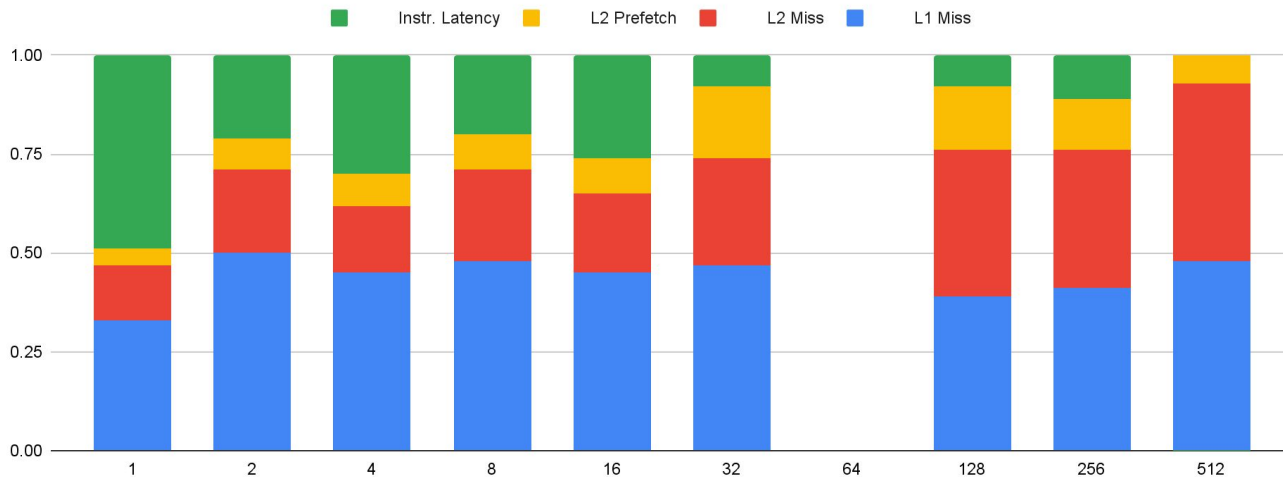
# Top-Down model in CTE-Arm

- Instruction latency (LD\_COMP\_WAIT\_EX)
  - Invariable cost of load memory access
- L2 Miss (LD\_COMP\_WAIT\_L2\_MISS)
  - Waiting for L2 miss to resolve
- L1 Miss (LD\_COMP\_WAIT\_L1\_MISS)
  - Waiting for L1 miss + L2 hit to resolve
- L2 prefetch stall (LD\_COMP\_WAIT\_PFP\_BUSY)
  - Waiting for L2 prefetch resources
- Other



# Top-Down model in CTE-Arm

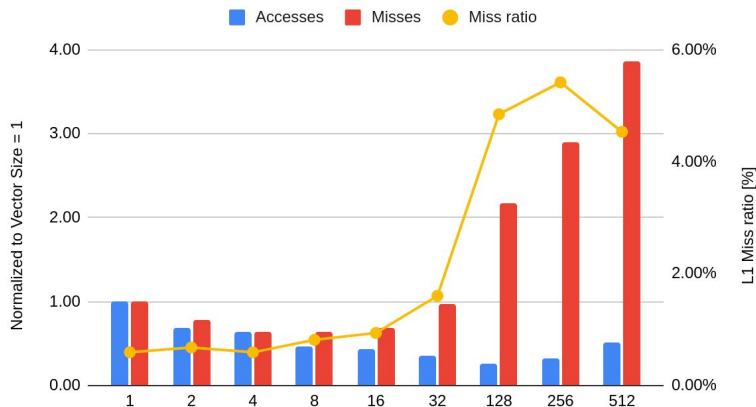
- Inside “Loads”
  - Vector Size = 1 → “Instr. Latency” is dominant
  - Vector Size [2,32] → “L1 Miss” becomes dominant
  - Vector Size > 32 → “L2 Miss” becomes dominant



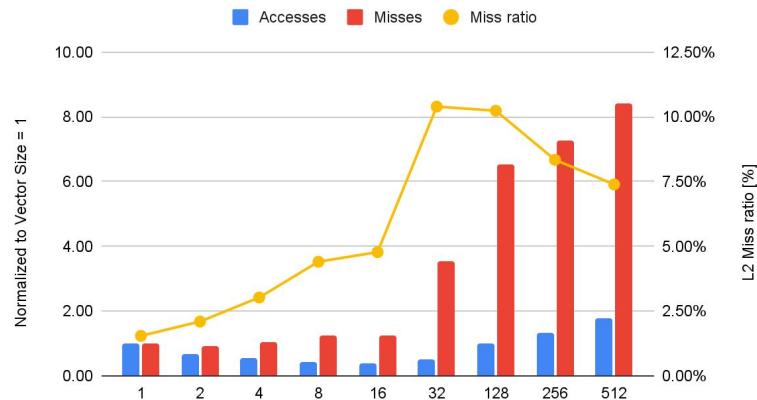
# Going deeper...

- Losing cycles because of L1/L2 misses...
- Possible reasons:
  - $\Delta$  Number of cache accesses
  - $\Delta$  Cache miss ratio

Nastin - L1 Accesses



Nastin - L2 Accesses



# Going deeper...

- Losing cycles because of L1/L2 misses...
- Possible reasons:

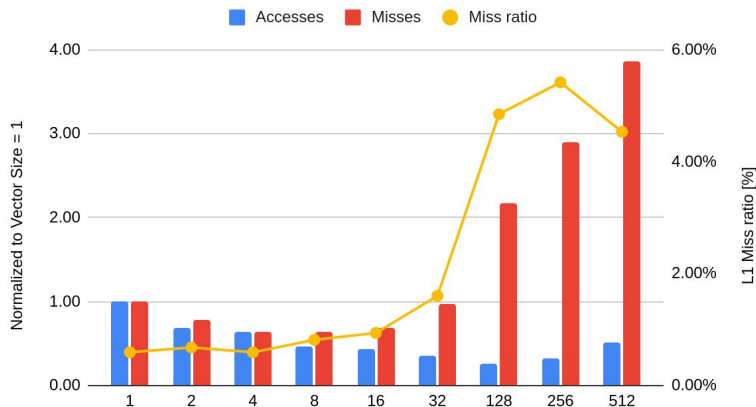


–  $\Delta$  Number of cache accesses

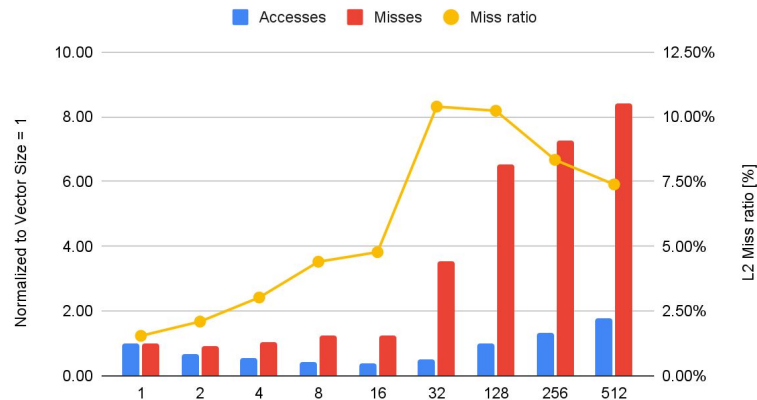


–  $\Delta$  Cache miss ratio

Nastin - L1 Accesses



Nastin - L2 Accesses



**Wrap up**

# Wrap up

- 😊 Top-down model
  - *Where is my machine spending its cycles?*
  - Guided methodology to drill down to the limiting factor
  - Possible to construct in both MareNostrum4 and CTE-Arm
- 😞 Top-down model
  - Each machine has its own metric hierarchy
  - Some hierarchies require micro-architectural parameters
  - The metrics and PAPI counters descriptions are not easy to interpret



**Could it be possible to have a general Top-down model that works on different machines?**



**Can we all agree on a set of counters?**

Filippo Mantovani  
filippo.mantovani@bsc.es

Fabio Banchelli (\*)  
fabio.banchelli@bsc.es

Marta Garcia-Gasulla  
marta.garcia@bsc.es



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*



**Other systems**





# Top-Down model in CTE-AMD

- Based on:

Top-Down Characterization Approximation based on performance counters architecture for AMD processors

<https://doi.org/10.1016/j.simpat.2016.08.006>

- Able to construct Top-Down model
  -  First level
  -  Second level (missing a lot of counters)

# Top-Down model in CTE-AMD

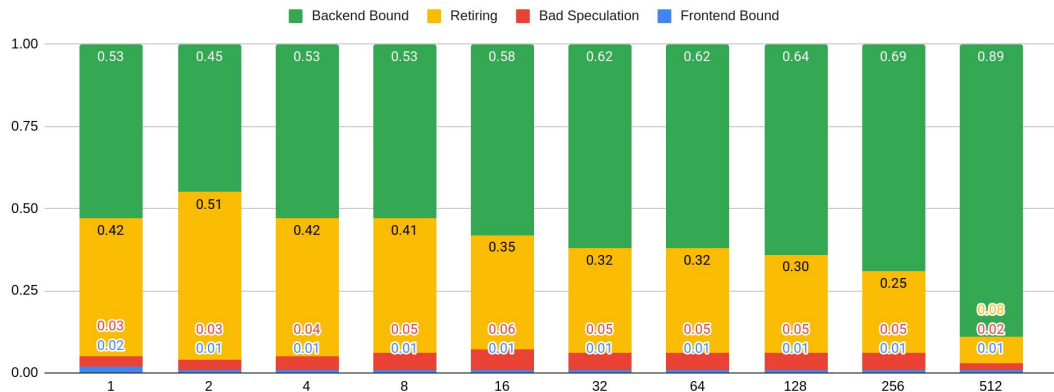
- Comments
  - Requires 2 counter sets
- Depends on micro-architectural parameters
  - #uops issued per cycle (*slots*)  $\rightarrow 6$
  - #cycles lost because misprediction  $\rightarrow 18$

# Top-Down model in CTE-AMD

- Similar behavior to MareNostrum4
- We cannot get more details because we are missing counters

Vector Size	1	2	4	8	16	32	64	128	256	512
Frontend Bound	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Bad Speculation	0.03	0.03	0.04	0.05	0.06	0.05	0.05	0.05	0.05	0.02
Retiring	0.42	0.51	0.42	0.41	0.35	0.32	0.32	0.30	0.25	0.08
Backend Bound	0.53	0.45	0.53	0.53	0.58	0.62	0.62	0.64	0.69	0.89

PAPI Preset Events			
Name	Code	Deriv	Description (Note)
PAPI_TLB_DM	0x80000014	No	Data translation lo
PAPI_TLB_IM	0x80000015	Yes	Instruction translat
PAPI_BR_TKN	0x8000002c	No	Conditional branch i
PAPI_BR_MSP	0x8000002e	No	Conditional branch i
PAPI_TOT_INS	0x80000032	No	Instructions complet
PAPI_BR_INS	0x80000037	No	Branch instructions
PAPI_TOT_CYC	0x8000003b	No	Total cycles
Of 7 available events, 1 is derived.			



# Top-Down model in CTE-Power

- Constructed following hierarchy found in:

POWER9 Performance Monitor Unit User's Guide

[https://wiki.raptorcs.com/w/images/6/6b/POWER9\\_PMU\\_UG\\_v12\\_28NOV2018\\_pub.pdf](https://wiki.raptorcs.com/w/images/6/6b/POWER9_PMU_UG_v12_28NOV2018_pub.pdf)

- Able to construct Top-Down-*like* model
  -  Four levels

# Top-Down model in CTE-Power

- Comments
  - Different metrics from original Top-Down model
  - Metrics of each level add up to 1 (relative to parent node in hierarchy)
  - Hierarchy is very detailed and is constructed Bottom-Up
  - Requires 10!! counter sets
  - Some counters always read as “0” (with no apparent errors)
- Does not depend on micro-architectural parameters

# Top-Down model in CTE-Power

	Avg									
	1	2	4	8	16	32	64	128	256	512
PM_RUN_CYC Speedup	1.00	1.75	2.78	4.01	5.09	5.94	6.58	6.86	7.02	6.56
PM_ICT_NOSLOT_CYC	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.02	0.02	0.03
PM_ISSUE_HOLD	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL	0.60	0.60	0.56	0.55	0.53	0.54	0.53	0.53	0.53	0.56
PM_CMPLU_STALL_BRU	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_EXEC_UNIT	0.11	0.12	0.14	0.15	0.15	0.13	0.13	0.13	0.13	0.11
PM_CMPLU_STALL_LS (*)	0.89	0.88	0.86	0.85	0.85	0.87	0.87	0.87	0.87	0.89
PM_CMPLU_STALL_LSAQ	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
PM_CMPLU_STALL_LRQ_FULL	0.40	0.46	0.50	0.57	0.52	0.29	0.34	0.40	0.40	0.09
PM_CMPLU_STALL_SRQ_FULL	0.53	0.48	0.44	0.37	0.44	0.68	0.63	0.56	0.57	0.30
PM_CMPLU_STALL_LSAQ_ARB	0.08	0.06	0.06	0.05	0.04	0.03	0.03	0.04	0.03	0.01
PM_CMPLU_STALL_EMQ	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.01
PM_CMPLU_STALL_ERAT_MISS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
PM_CMPLU_STALL_EMQ_FULL	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_LRQ	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
PM_CMPLU_STALL_LMQ_FULL	0.00	0.00	0.01	0.01	0.02	0.04	0.04	0.06	0.05	0.12
PM_CMPLU_STALL_ST_FWD	0.89	0.90	0.90	0.91	0.92	0.90	0.91	0.90	0.89	0.84
PM_CMPLU_STALL_LHS	0.11	0.10	0.09	0.09	0.06	0.06	0.05	0.04	0.06	0.04
PM_CMPLU_STALL_LSU_MFSR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_LARX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_LRQ_OTHER	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_DCACHE_MISS	0.07	0.09	0.11	0.12	0.11	0.10	0.11	0.12	0.12	0.21
PM_CMPLU_STALL_LOAD_FINISH	0.19	0.19	0.19	0.21	0.18	0.15	0.15	0.15	0.14	0.13
PM_CMPLU_STALL_SRQ	0.50	0.46	0.42	0.34	0.28	0.21	0.19	0.16	0.16	0.14
PM_CMPLU_STALL_STORE_DATA	0.01	0.01	0.02	0.03	0.05	0.05	0.06	0.07	0.08	0.07
PM_CMPLU_STALL_LWSYNC	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_HWSYNC	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_EIEIO	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_STCX	0.99	0.98	0.97	0.96	0.93	0.93	0.91	0.90	0.88	0.88
PM_CMPLU_STALL_SLB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_TEND	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_PASTE	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_TLBIE	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_STORE_PIPE_ARB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_STORE_FIN_ARB	0.00	0.01	0.01	0.01	0.02	0.02	0.03	0.03	0.04	0.04
PM_CMPLU_STALL_STORE_FINISH	0.16	0.19	0.21	0.27	0.37	0.49	0.52	0.54	0.55	0.48
PM_CMPLU_STALL_LSU_FIN	0.06	0.05	0.05	0.04	0.03	0.02	0.02	0.02	0.02	0.01
PM_CMPLU_STALL_NTC_FLUSH	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_NTC_DISP_FIN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_THRD	0.10	0.10	0.09	0.08	0.07	0.05	0.05	0.04	0.04	0.04
PM_1PLUS_PPC_CMPL	0.13	0.14	0.16	0.17	0.18	0.18	0.18	0.18	0.18	0.16
Other	0.14	0.13	0.15	0.16	0.19	0.20	0.22	0.22	0.23	0.21

# Top-Down model in CTE-Power

- “PM\_CMPLU\_STALL” ≈ “Backend Bound” → Always limiting factor
- “PM\_CMPLU\_STALL\_THRD” are cycles due to other hw threads → 10%???
- “Other” accounts for 13%-23% of total cycles 😞

	1	2	4	8	16	32	64	128	256	512
PM_RUN_CYC Speedup	1.00	1.75	2.78	4.01	5.09	5.94	6.58	6.86	7.02	6.56
PM_ICT_NOSLOT_CYC	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.02	0.02	0.03
PM_ISSUE_HOLD	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL	0.60	0.60	0.56	0.55	0.53	0.54	0.53	0.53	0.53	0.56
PM_CMPLU_STALL_THRD	0.10	0.10	0.09	0.08	0.07	0.05	0.05	0.04	0.04	0.04
PM_1PLUS_PPC_CMPL	0.13	0.14	0.16	0.17	0.18	0.18	0.18	0.18	0.18	0.16
Other	0.14	0.13	0.15	0.16	0.19	0.20	0.22	0.22	0.23	0.21

# Top-Down model in CTE-Power

- “PM\_CMPLU\_STALL\_LS”  $\approx$  “Memory Bound”  $\rightarrow$  Always limiting factor
- “PM\_CMPLU\_STALL\_EXEC\_UNIT”  $\approx$  “Compute Bound”

	1	2	4	8	16	32	64	128	256	512
PM_RUN_CYC Speedup	1.00	1.75	2.78	4.01	5.09	5.94	6.58	6.86	7.02	6.56
PM_CMPLU_STALL	0.60	0.60	0.56	0.55	0.53	0.54	0.53	0.53	0.53	0.56
PM_CMPLU_STALL_BRU	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_EXEC_UNIT	0.11	0.12	0.14	0.15	0.15	0.13	0.13	0.13	0.13	0.11
PM_CMPLU_STALL_LS (*)	0.89	0.88	0.86	0.85	0.85	0.87	0.87	0.87	0.87	0.89
PM_CMPLU_STALL_NTC_FLUSH	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_NTC_DISP_FIN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00



# Top-Down model in CTE-Power

- Inside “PM\_CMPLU\_STALL”
  - “PM\_CMPLU\_STALL\_LS”  $\approx$  “Memory Bound”  $\rightarrow$  Always limiting factor
  - “PM\_CMPLU\_STALL\_EXEC\_UNIT”  $\approx$  “Compute Bound”

	1	2	4	8	16	32	64	128	256	512
PM_RUN_CYC Speedup	1.00	1.75	2.78	4.01	5.09	5.94	6.58	6.86	7.02	6.56
PM_CMPLU_STALL	0.60	0.60	0.56	0.55	0.53	0.54	0.53	0.53	0.53	0.56
PM_CMPLU_STALL_BRU	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_EXEC_UNIT	0.11	0.12	0.14	0.15	0.15	0.13	0.13	0.13	0.13	0.11
PM_CMPLU_STALL_LS (*)	0.89	0.88	0.86	0.85	0.85	0.87	0.87	0.87	0.87	0.89
PM_CMPLU_STALL_NTC_FLUSH	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM_CMPLU_STALL_NTC_DISP_FIN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

# Top-Down model in CTE-Power

- Inside “PM\_CMPLU\_STALL\_LS”
  - “PM\_CMPLU\_STALL\_SRQ” → Limiting factor up to Vector Size 8
  - “PM\_CMPLU\_STALL\_STORE\_FINISH” → Limiting factor Vector Size > 8
  - What do these counters monitor? 🤔 (do not have answer yet)

	1	2	4	8	16	32	64	128	256	512
PM_CMPLU_STALL_LS (*)	0.89	0.88	0.86	0.85	0.85	0.87	0.87	0.87	0.87	0.89
PM_CMPLU_STALL_LSAQ	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
PM_CMPLU_STALL_EMQ	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.01
PM_CMPLU_STALL_LRQ	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
PM_CMPLU_STALL_DCACHE_MISS	0.07	0.09	0.11	0.12	0.11	0.10	0.11	0.12	0.12	0.21
PM_CMPLU_STALL_LOAD_FINISH	0.19	0.19	0.19	0.21	0.18	0.15	0.15	0.15	0.14	0.13
PM_CMPLU_STALL_SRQ	0.50	0.46	0.42	0.34	0.28	0.21	0.19	0.16	0.16	0.14
PM_CPMLU_STALL_STORE_FINISH	0.16	0.19	0.21	0.27	0.37	0.49	0.52	0.54	0.55	0.48
PM_CMPLU_STALL_LSU_FIN	0.06	0.05	0.05	0.04	0.03	0.02	0.02	0.02	0.02	0.01

# Going deeper... Vector Size <= 8

- `PM_CMPLU_STALL_SRQ`

Aggregation of different events (`PM_CMPLU_STALL_STORE_DATA`,  
`PM_CMPLU_STALL_LWSYNV`, `PM_CMPLU_STALL_HWSYNC`, etc.)

- Most of these counters measure always 0 or represent a small fraction of cycles
- The most relevant counter in this group is `PM_CPMLU_STALL_STCX`

*Finish stall because the NTF instruction<sup>1</sup> was a stcx<sup>2</sup> waiting for a response from the L2 cache.*

1. NTF: Next-To-Finish instruction
2. stcx: Store reserved sequences give the programmer/compiler the ability to share storage in an effective manner by exchanging locks and updating atomic variables between program threads.

# Going deeper... STCX Instructions

- Checking the binaries for the STCX instruction... No occurrences

```
$ objdump -d <binary> | grep stcx | wc
```

0            0            0

- Where do the cycle stalls due to STCX instructions come?
  - Could it be from the MPI library?
  - Could it be from the tracing library (Extrae)?
  - Could it be from another system library?

⚠ Where to go next? ⚠

# Going deeper... Vector Size > 8

- PM\_CMPLU\_STALL\_STORE\_FINISH

*Finish stall because the NTF instruction<sup>1</sup> was a store with all its dependencies met, just waiting to go through the LSU pipe to finish*

- Interpretation: Load-Store-Unit (LSU) is either...
  - a. Full → Store has to wait to enter the LSU and be processed
  - b. Working → Store has reached the LSU, but its process takes time

⚠ Where to go next? ⚠

1. NTF: Next-To-Finish instruction

# Compiler errors in CTE-Arm

- Arm HPC Compiler + OpenMPI + VECTOR\_SIZE=64 (runtime)

```
Alya.x: malloc.c:2392: sysmalloc: Assertion `(old_top == initial_top (av) && old_size == 0) || ((unsigned long) (old_size) >= MINSIZE && prev_inuse (old_top) && ((unsigned long) old_end & (pagesize - 1)) == 0)' failed.
```

- GNU Compiler 11.0.0 + Fujitsu MPI + VECTOR\_SIZE=1 (traces & data processing)
  - 3/3 runs validate “ALYA CALCULATIONS CORRECT”
  - 1/3 runs yield expected results
  - 2/3 have some weird data points

- GNU Compiler 8.3.1 + Fujitsu MPI + VECTOR\_SIZE=2 (compilation)

```
../../Sources/modules/nastin/nsi_element_assembly_split_oss_default.f90:786:0:
```

```
    elaup(DEF_VECT,idof1,jnode) = elaup(DEF_VECT,idof1,jnode) - fact1(DEF_VECT)
```

```
internal compiler error: in expand_expr_real_1, at expr.c:10889
```

```
Please submit a full bug report, with preprocessed source if appropriate.
```

```
See <http://bugzilla.redhat.com/bugzilla> for instructions.
```

```
make[1]: *** [makefile:14560: Objects_x/nsi_element_assembly_split_oss_default.o]
```

```
Error 1
```



- GNU Compiler 8.3.1 + Fujitsu MPI + VECTOR\_SIZE=4 (compilation)

```
../..Sources/alya/domain/mod_elmgeo_vector.f90:282:0:
```

```
&                                + xjaci(DEF_VECT,2,1,igaus) * deriv(2,j,igaus)
```

```
internal compiler error: in expand_expr_real_1, at expr.c:10889
```

```
Please submit a full bug report, with preprocessed source if appropriate.
```

```
See <http://bugzilla.redhat.com/bugzilla> for instructions.
```

```
make[1]: *** [makefile:20683: Objects_x/mod_elmgeo_vector.o] Error 1
```

- GNU Compiler 8.3.1 + Fujitsu MPI + VECTOR\_SIZE=128 (runtime)

```
--| ALYA  START TIME STEP 1, t=  5.595419E-06
```

```
--| ALYA  SOLVE CHEMIC (1)
```

```
--| ALYA  SOLVE TEMPER (1)
```

```
--| ALYA  SOLVE NASTIN (MC)
```

```
--| ALYA  SOLVE NASTIN (MC)
```

```
--| ALYA  SOLVE NASTIN (MSC)
```

```
--| ALYA  END TIME STEP
```

Program received signal SIGSEGV: Segmentation fault - invalid memory reference.

