Cache-aware Roofline Model :: Tool

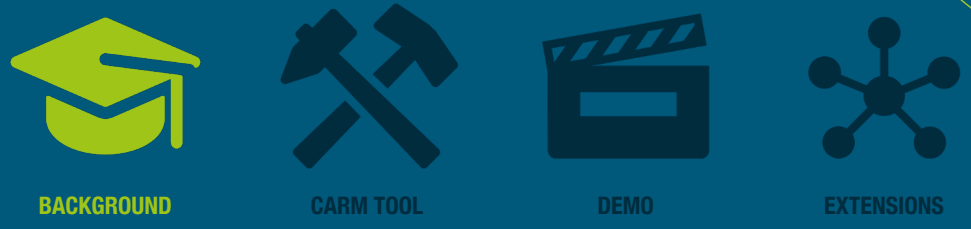
Aleksandar Ilic and José Brito

Alexandre Rodrigues, José Morgado, Diogo Matos,
Diogo Marques, Frederico Pratas, Leonel Sousa





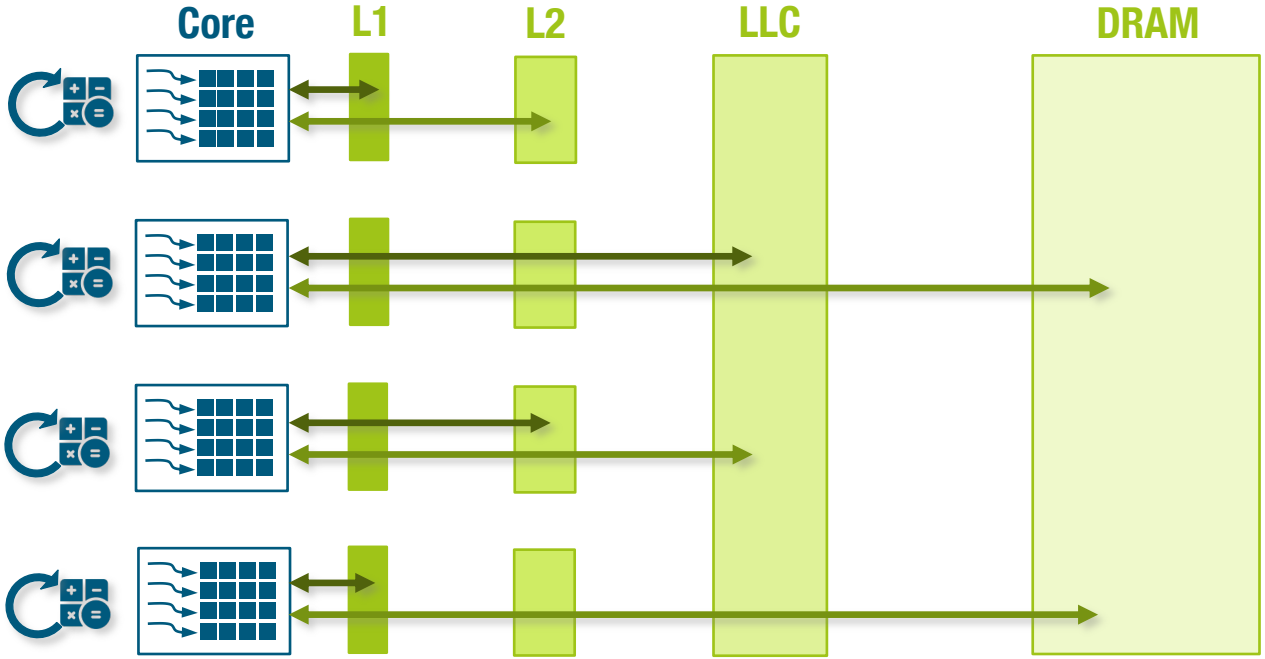
Cache-aware Roofline Model



Cache-aware Roofline Model: Outline

A. Ilic, F. Pratas and L. Sousa, "Cache-aware Roofline Model: Upgrading the Loft", IEEE Computer Architecture Letters (2014)
J. Morgado, L. Sousa and A. Ilic, "CARM Tool: Cache-Aware Roofline Model Automatic Benchmarking and Application Analysis", IISWC (2024)

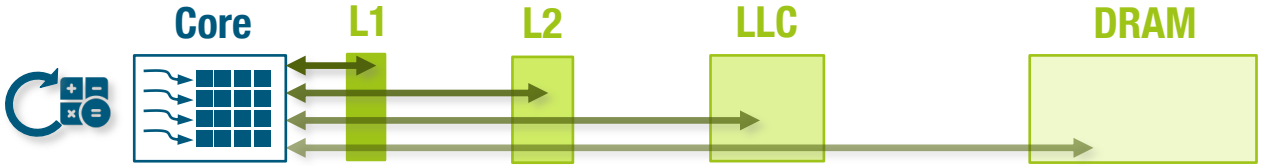
Roofline in a nutshell



Communication overlapped with computation

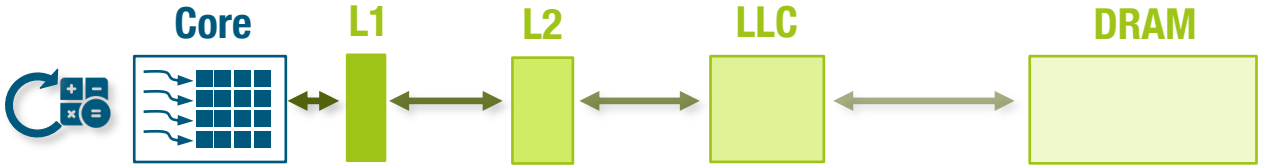
Max performance capped by peak compute throughput or available bandwidth (processor's view)

What is bandwidth?



Cache-aware Roofline Model (CARM)[1]: Bandwidth as seen by the core

- Obtained via micro-benchmarking



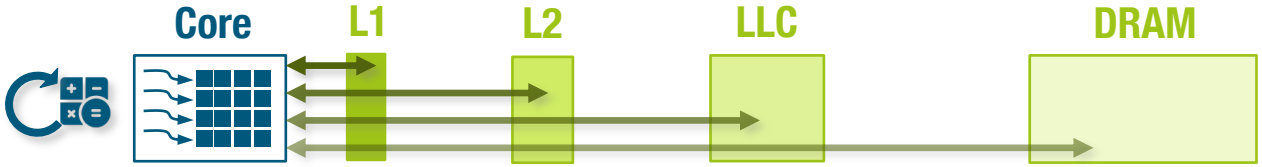
Original Roofline Model (ORM)[2]: Bandwidth between memory levels

- Can be obtained from data-sheets

[1] A. Ilic, F. Pratas and L. Sousa, "Cache-aware Roofline Model: Upgrading the Loft", IEEE Computer Architecture Letters (2014)

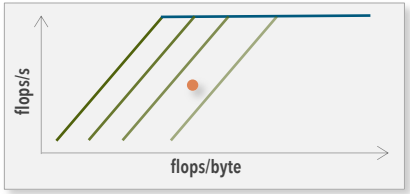
[2] S. Williams, A. Waterman, D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures", Commun. ACM (2009)

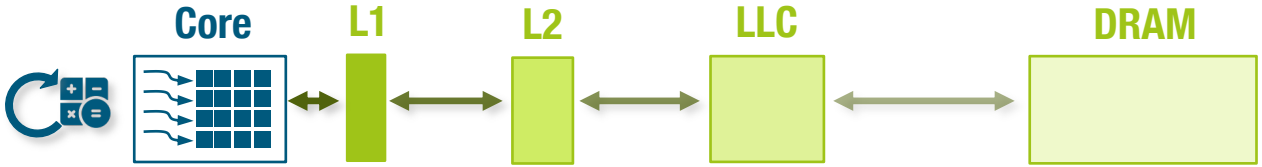
Implications …



Cache-aware Roofline Model[1]

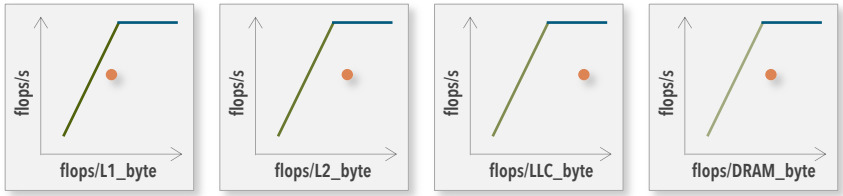
- One model, one arithmetic intensity
- One application "point"





Original Roofline Model[2]

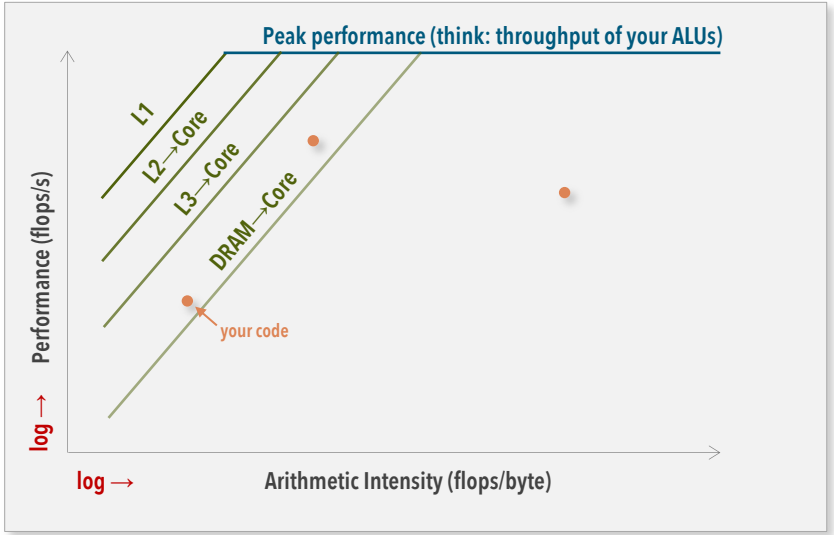
- Several models, several intensities
- Several application "points"



[1] A. Ilic, F. Pratas and L. Sousa, "Cache-aware Roofline Model: Upgrading the Loft", IEEE Computer Architecture Letters (2014)

[2] S. Williams, A. Waterman, D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures", Commun. ACM (2009)

Implications … bring cool features



Cache-aware Roofline Model

- Shows absolute architecture maximums

(You can't break them! Can your application exploit them?)

How to "plot" my code?

- CARM arithmetic intensity is exactly what you expect it to be!

code

```
...
  float a = A[i];
  float b = B[i];
  float c = a*b;
  C[i] = c;
...
```

1 flop

12 bytes

AI = 1/12

assembly

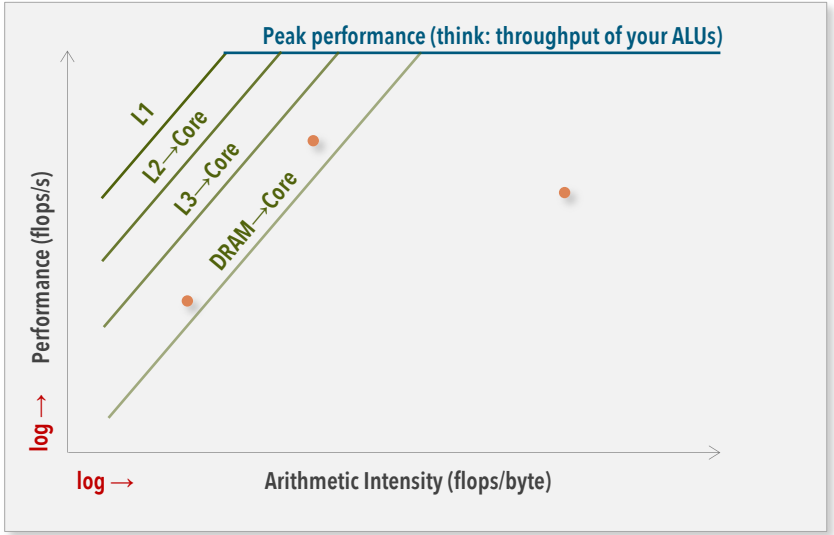
```
...
ld  r1, mem[add1]
ld  r2, mem[add2]
mul r3, r1, r2
st  mem[add3], r3
...
```

counters

```
...
MEM_RETIRED.LOADS
MEM_RETIRED.STORES
INST_RETIRED.FLOPS
CPU_CLK_CYCLES.ALL
...
```

 A. Ilic, F. Pratas and L. Sousa, "Cache-aware Roofline Model: Upgrading the Loft", IEEE Computer Architecture Letters (2014)

Implications … bring cool features



Cache-aware Roofline Model

- Shows absolute architecture maximums

(You can't break them! Can your application exploit them?)

How to "plot" my code?

- CARM arithmetic intensity is exactly what you expect it to be!

Intel Advisor Roofline feature

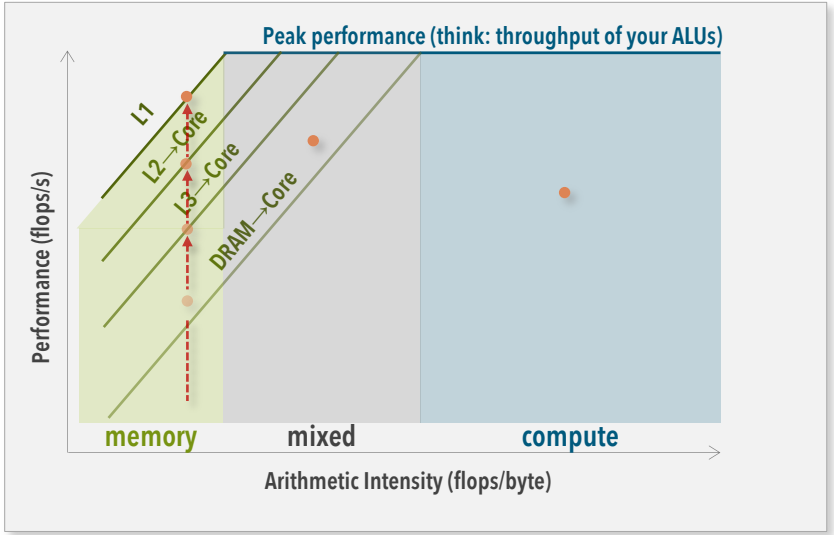
- CARM is there since 2017

CARM Tool (in-house)

- Open-source at Champ Hub @ Github
- Support all major CPUs and GPUs
- PMU- and DBI-based application profiling

A. Ilic, F. Pratas and L. Sousa, "Cache-aware Roofline Model: Upgrading the Loft", IEEE Computer Architecture Letters (2014)
J. Morgado, L. Sousa and A. Ilic, "CARM Tool: Cache-Aware Roofline Model Automatic Benchmarking and Application Analysis", IISWC (2024)

Implications … bring cool features



memory bound
(improve access pattern, use of caches)

mixed
(all kinds of everything)

compute bound
(vectorize, parallelize…)

Cache-aware Roofline Model

- Shows absolute architecture maximums

(You can't break them! Can your application exploit them?)

How to "plot" my code?

- CARM arithmetic intensity is exactly what you expect it to be!

How to use CARM?

① Detect the boundness region

- What are my expected maximums?
- Provides first optimization hints

② Draw an imaginary vertical line

- What are my main bottlenecks? (observe intersected lines)
- Focus your optimization (aim at surpassing the line above)

③ Optimize your code: Break above roofs!

- You should move up (as your performance improves)
- Unless you restructure the code, or your compiler decides so…

A. Ilic, F. Pratas and L. Sousa, "Cache-aware Roofline Model: Upgrading the Loft", IEEE Computer Architecture Letters (2014)
D. Marques, et.al., "Performance analysis with Cache-aware Roofline Model in Intel Advisor", HPCS (2017)









Cache-aware Roofline Model: Outline

J. Morgado, L. Sousa and A. Ilic, "CARM Tool: Cache-Aware Roofline Model Automatic Benchmarking and Application Analysis", IISWC (2024)

State of the art – Roofline Tools

	Intel Advisor	AMD uProf	ERT	CARM Tool
Supported Architectures	Intel	AMD	Intel | A	Intel | AMD ARM | RISC-V
Supported Roofline	CARM	ORM	? CARI	CARM
Application Analysis	DBI	PMUs	No sup	DBI | PMUs
Open-Source	No	No	Yes	Yes

SUPPORTED ISA (SIMD extensions)

intel. AMD — SCALAR | SSE | AVX2 | AVX512

arm — SCALAR | NEON | SVE

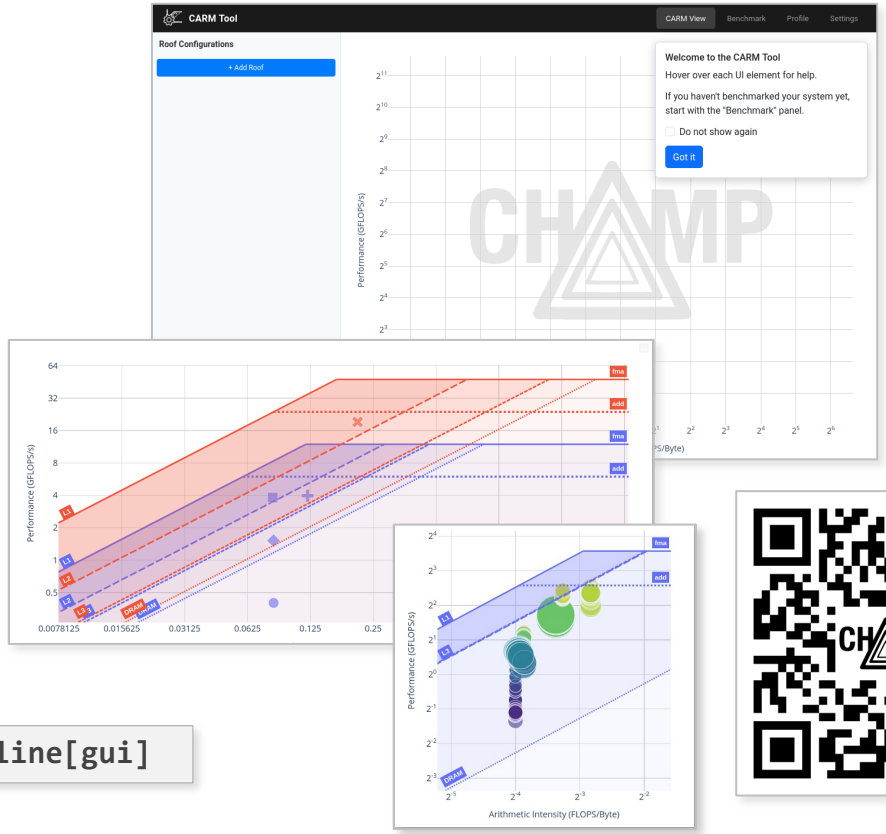
RISC-V — SCALAR | RVV 0.7.1 | RVV 1.0

 J. Morgado, L. Sousa and A. Ilic, "CARM Tool: Cache-Aware Roofline Model Automatic Benchmarking and Application Analysis", IISWC (2024)

The CARM Tool :: High-level overview

DEMO (SOON)

- **Interface: Two types**
  - Command Line Interface (CLI)
  - Graphical User Interface (GUI)
- **Automatic Benchmarking**
  - CARM Benchmarks
  - Memory / FP / Mixed Benchmarks
- **Application Analysis**
  - PMU: PAPI, perf
  - DBI: DynamoRIO, Intel SDE



TRY IT!

```
pip install carm-roofline[gui]
```

 J. Morgado, L. Sousa and A. Ilic, "CARM Tool: Cache-Aware Roofline Model Automatic Benchmarking and Application Analysis", IISWC (2024)

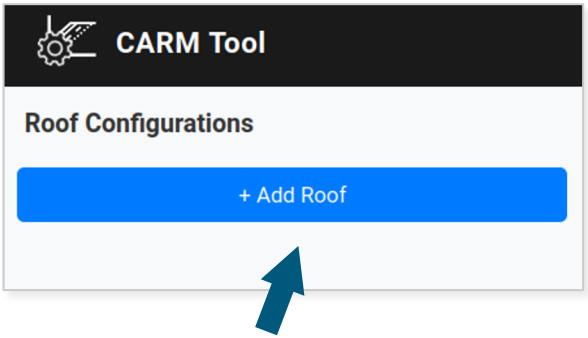
Starting the CARM tool

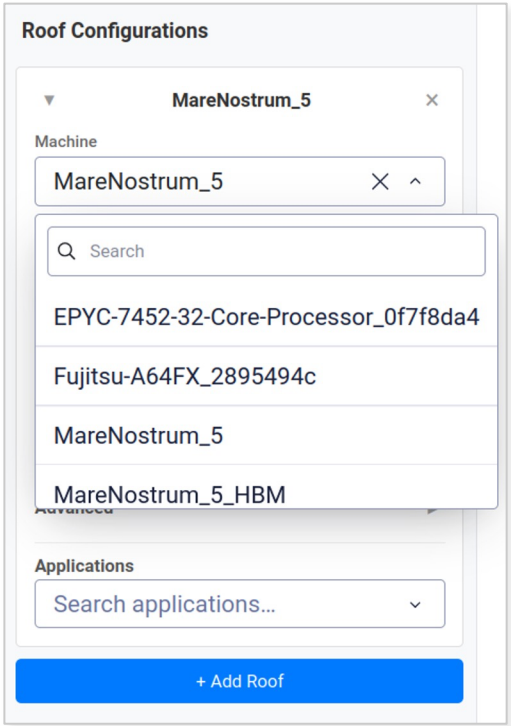
From the GUI (web interface)

- **From the CLI, open the web interface**

```
carm gui
```

- **Select the roofline for your machine**
  - As many roofline micro-benchmarking results as you need





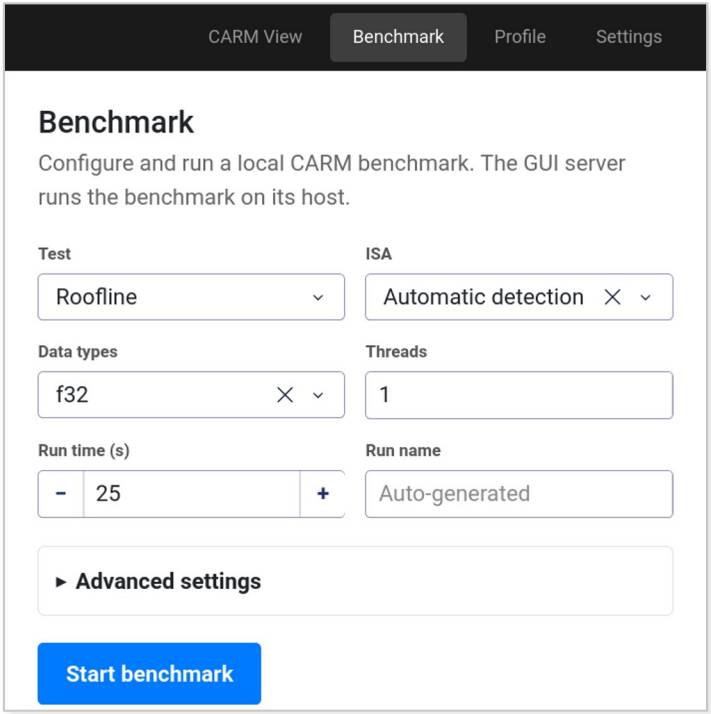
How to benchmark your machine?

From the CLI:

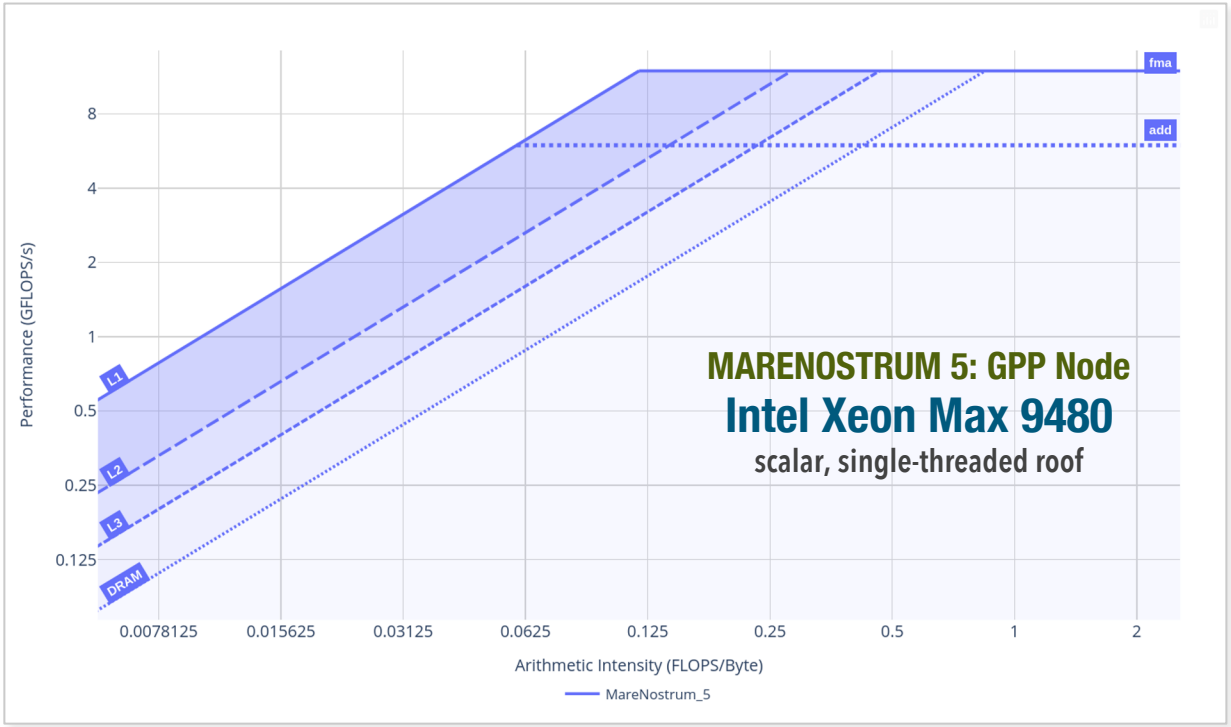
```
carm benchmark --threads ... --data-type ...
```

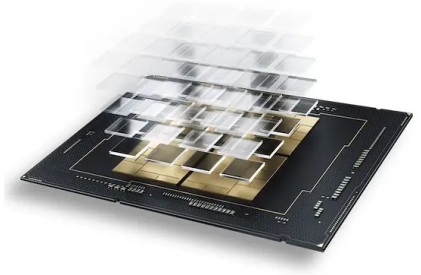
- **Automatically detects cache size, frequency, etc.**
- **Generates and runs:**
  - 4 Memory microbenchmarks (L1, L2, L3, DRAM)
  - 2 Arithmetic microbenchmark (add, fused multiply-add)
- **Stores the results ready for plotting**

From the GUI (web interface)



Rooflining Supercomputers: MareNostrum 5





Application Profiling

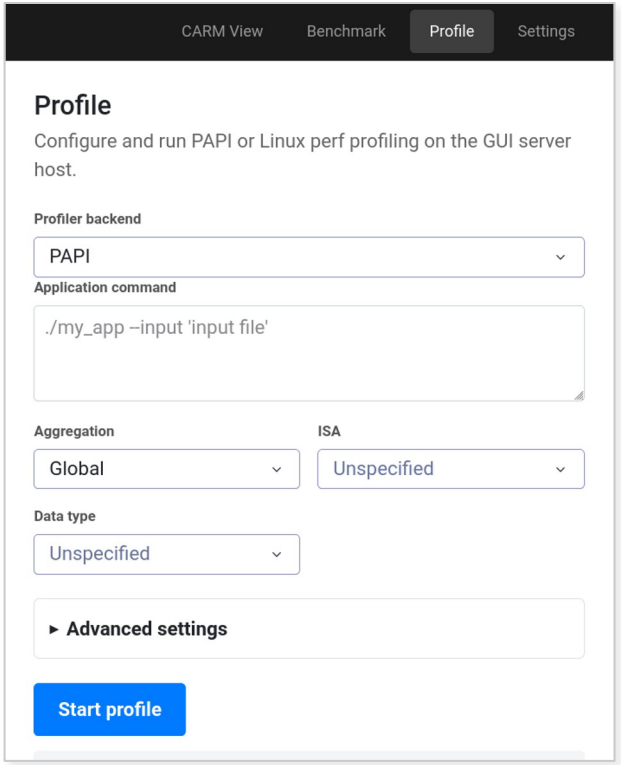
- **Two application profiling facilities**
  - PMU (hardware counters) via PAPI and Linux perf
  - DBI (dynamic binary instrumentation): DynamoRIO, Intel SDE
- **Support for ROI (region of interest)**
  - Mark only the part of the code that is of interest

From the CLI

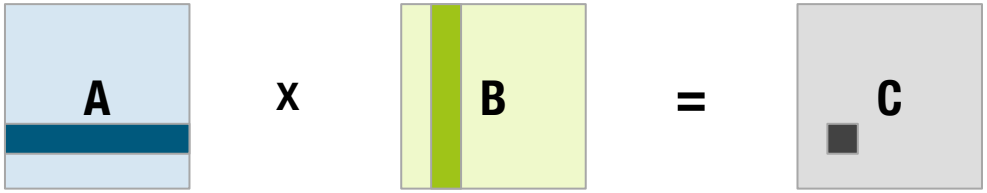
```
carm profile -- ./your-app your_arg=7
```

- **PMU option: Captures PAPI output**
  - Application annotated with PAPI region-of-interest calls
- **Processes and stores results for plotting**

From the GUI (web interface)



Optimizing Matrix Multiplication on MareNostrum 5



APPLICATION: DGEMM

Matrix multiplication: C = A x B
Double precision: 8-byte elements
Squared matrices: N = 2048 elements

ARCHITECTURE: MareNostrum5 GPP

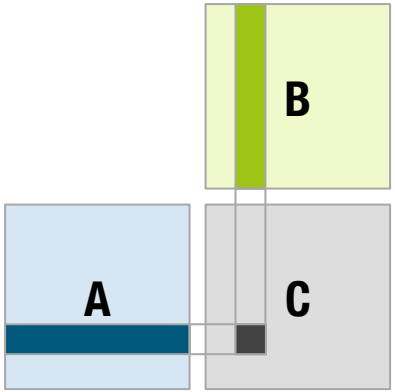
Intel Xeon Max 9480 (single-thread)
Caches:
L1: 48 KiB | L2: 2 MiB | L3: 225 MiB

2048^2 Elements
8 Bytes
× 3 Matrices
96 MiB

Fits in the L3 cache!

Naïve GEMM implementation

- **Every iteration:**
  - 2 loads, 1 store (24 bytes)
  - 2 operations (mul, add)
- **Arithmetic intensity: 2 flops / 24 bytes**




```
1  void gemm_naive(double *A, double *B, double *C)
2  {
3      for (int i = 0; i < N; i++) {
4          for (int j = 0; j < N; j++) {
5              for (int k = 0; k < N; k++) {
6                  C[i * N + j] += A[i * N + k] * B[k * N + j];
7              }
8          }
9      }
10 }
```

STORE 8 bytes

LOAD 8 bytes

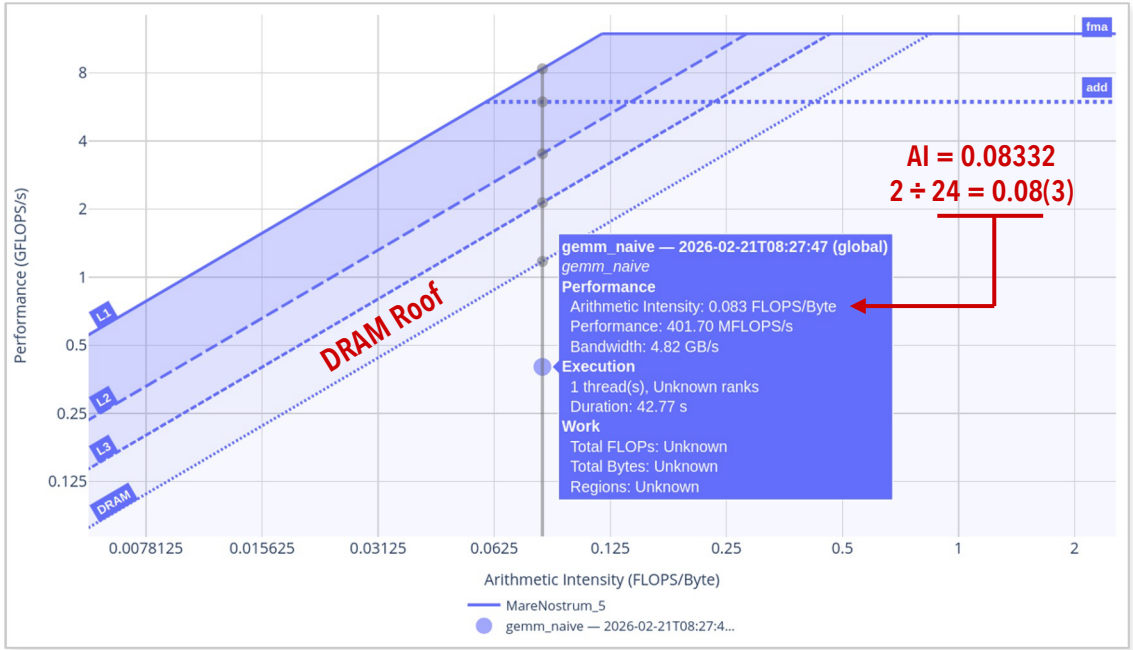
LOAD 8 bytes

2 FLOPS

Rooflining Naïve DGEMM

[1] Basic implementation (row major)

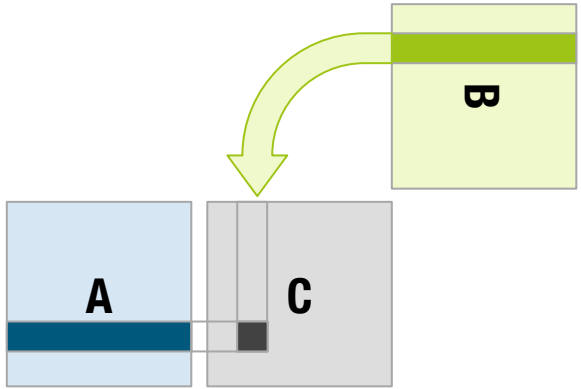




Optimizing GEMM: B Transpose

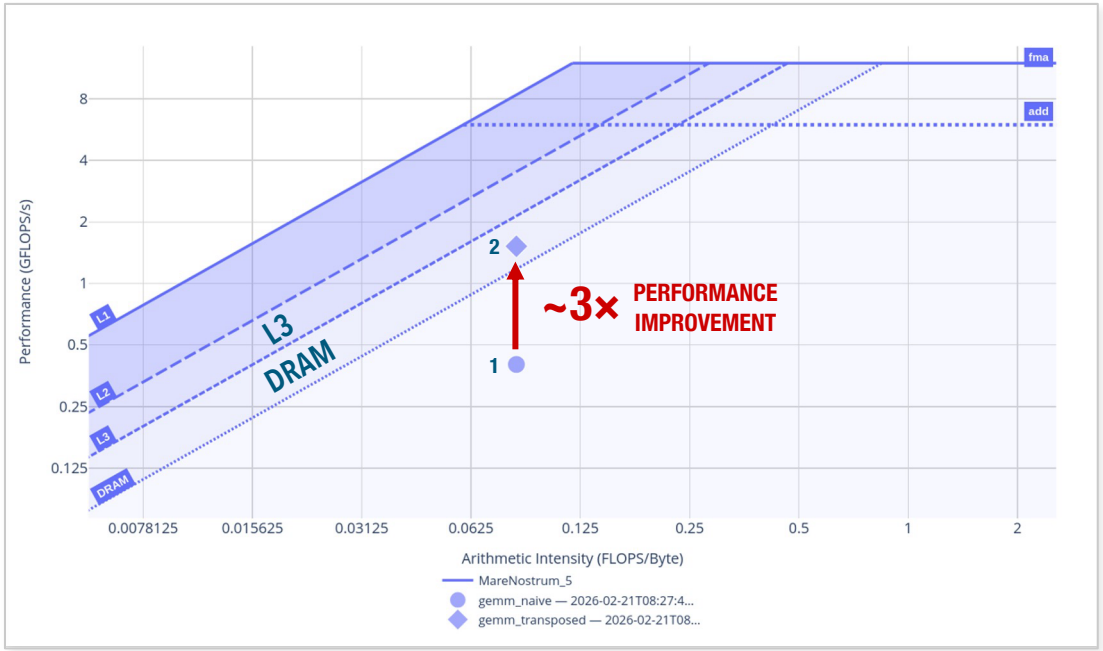
- Transposing the B matrix
- Column-major order in B
  - Contiguous memory accesses
  - Better locality

- Arithmetic intensity: 2 flops / 24 bytes




```
1  void gemm_transposed(double *A, double *B_T, double *C)
2  {
3      for (int i = 0; i < N; i++) {
4          for (int j = 0; j < N; j++) {
5              for (int k = 0; k < N; k++) {
6                  C[i * N + j] += A[i * N + k] * B_T[j * N + k];
7              }
8          }
9      }
10 }
```

Optimizing GEMM: B Transpose



[1] Basic implementation (row major)

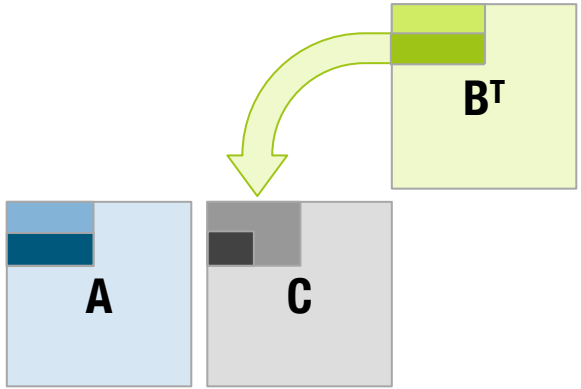


[2] Transposed B (improved mem. access)



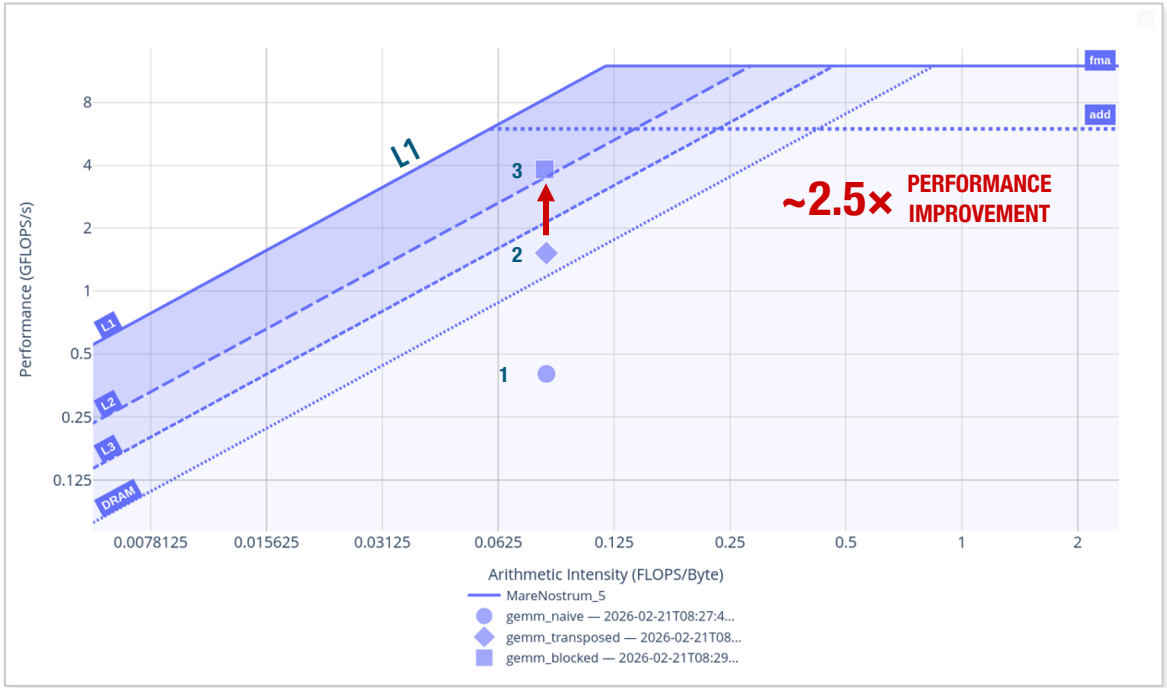
Optimizing GEMM: Matrix blocking

- **Matrix blocking: Better cache locality**
  - Reuse data in higher-level caches
  - 32-element block – 24 KiB < L1
  - Higher effective memory bandwidth
- **Arithmetic intensity: 2 flops / 24 bytes**




```
1  void gemm_blocked(double *A, double *B, double *C)
2  {
3      for (int i0 = 0; i0 < N; i0 += BLOCK_SIZE) {
4          for (int j0 = 0; j0 < N; j0 += BLOCK_SIZE) {
5              for (int k0 = 0; k0 < N; k0 += BLOCK_SIZE) {
6                  for (int i = i0; i < i0 + BLOCK_SIZE; i++) {
7                      for (int j = j0; j < j0 + BLOCK_SIZE; j++) {
8                          for (int k = k0; k < k0 + BLOCK_SIZE; k++) {
9                              C[i * N + j] += A[i * N + k] * B[j * N + k];
10                         }
```

Optimizing GEMM: Blocking



[1] Basic implementation (row major)



[2] Transposed B (improved mem. access)



[3] Cache blocking

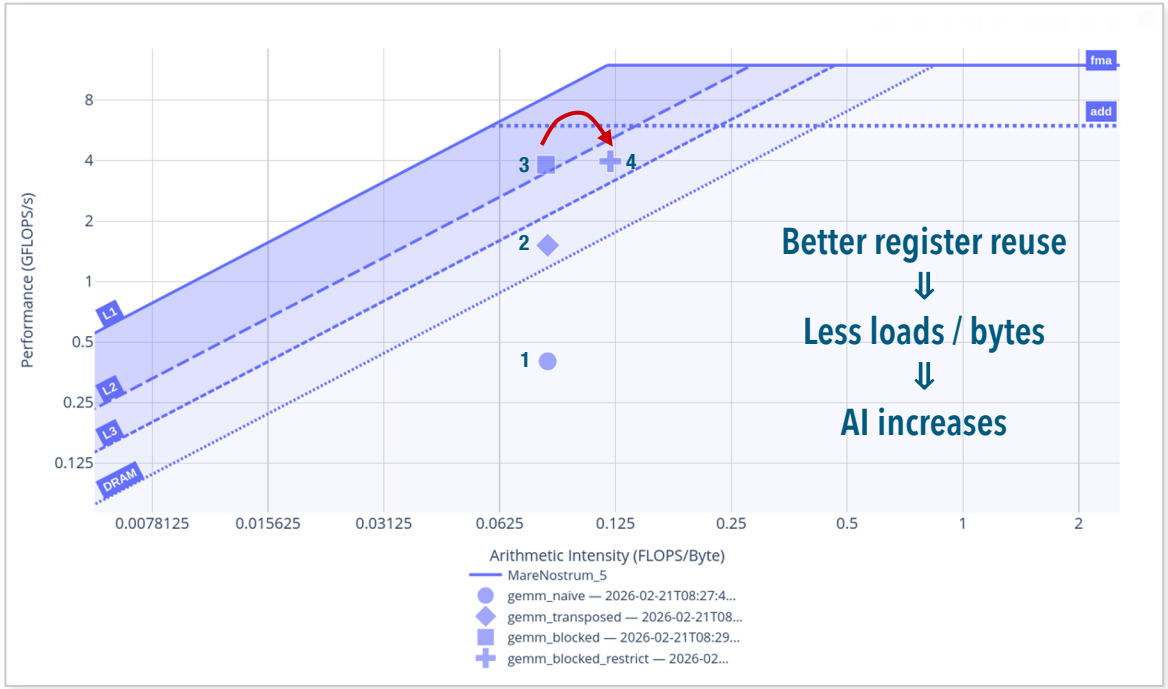


Optimizing GEMM: Restrict

- `restrict` keyword
- **Tells the compiler the pointers don't alias**
  - "No other pointer will access this pointer's data"
  - Avoids redundant loads (register reuse)

```
1  void gemm_blocked_restrict(double *restrict A, double *restrict B, double *restrict C)
2  {
3      for (int i0 = 0; i0 < N; i0 += BLOCK_SIZE) {
4          for (int j0 = 0; j0 < N; j0 += BLOCK_SIZE) {
5              for (int k0 = 0; k0 < N; k0 += BLOCK_SIZE) {
6                  for (int i = i0; i < i0 + BLOCK_SIZE; i++) {
7                      for (int j = j0; j < j0 + BLOCK_SIZE; j++) {
8                          for (int k = k0; k < k0 + BLOCK_SIZE; k++) {
9                              C[i * N + j] += A[i * N + k] * B[j * N + k];
10                         }
```

Optimizing GEMM: Restrict



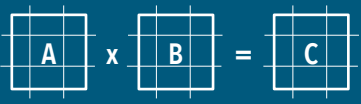
[1] Basic implementation (row major)



[2] Transposed B (improved mem. access)



[3] Cache blocking

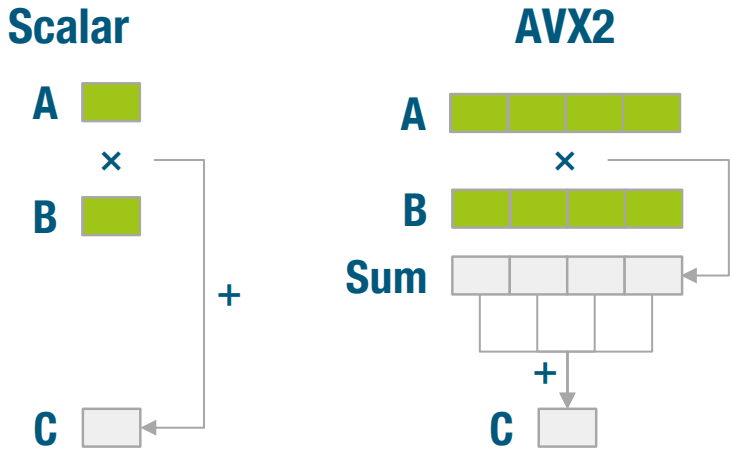


[4] Restrict



Optimizing GEMM: AVX2 Vectorization

- **AVX2 vectorization (256-bit)**
  - Load 4-element vectors of A and B
  - Vector fused multiply-add
  - Horizontal reduction (add the 4 elements)
  - Store in C




```
1  // abridged
2  void gemm_blocked_restrict_avx2(double *restrict A, ...)
3  {
4      // 3 block loops ...
5      // inner i,j loops ...
6      __m256d vsum = _mm256_setzero_pd();
7
8      int k = k0;
9      for (; k <= k0 + BLOCK_SIZE - 4; k += 4) {
10         __m256d va = _mm256_loadu_pd(&A[i * N + k]);
11         __m256d vb = _mm256_loadu_pd(&B[j * N + k]);
12
13         vsum = _mm256_fmadd_pd(va, vb, vsum);
14     }
15
16     /* horizontal reduction */
17     __m128d low = _mm256_castpd256_pd128(vsum);
18     __m128d high = _mm256_extractf128_pd(vsum, 1);
19     __m128d sum2 = _mm_add_pd(low, high);
20     sum2 = _mm_hadd_pd(sum2, sum2);
21
22     double sum = _mm_cvtsd_f64(sum2);
23
24     C[i * N + j] += sum;
25 }
```

Optimizing Matrix Multiplication – Summary

- **Memory-bound?**
  - Improve locality (Bandwidth ⇈)
  - Reduce redundant loads (AI ⇈)
- **Blocking and transposition**
  - Leverage higher caches
  - Bandwidth: 10× ⇈
  - Register reuse
  - Memory-bound ➔ Compute-bound
  - AI 5.5× ⇈

- **Compute-bound?**
  - Vectorize (Peak-performance ⇈)
- **AVX2 / AVX-512 Vectorization**
  - Peak-performance: 8× ⇈

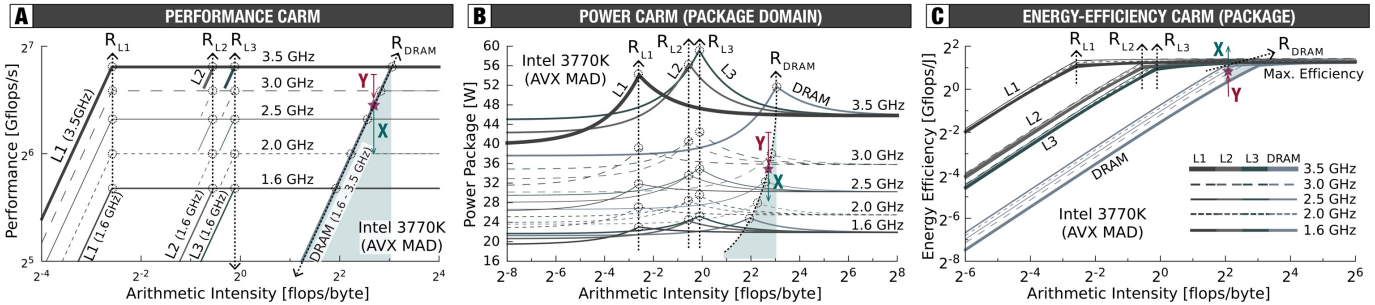
166× SPEEDUP



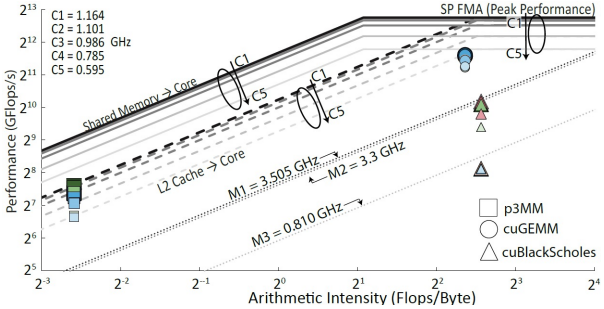
Cache-aware Roofline Model: Outline

Cache-aware Roofline Model: Extensions

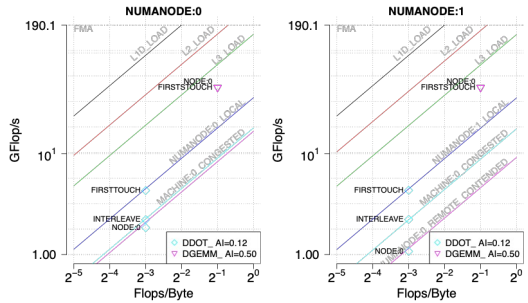
CARM-based DVFS analysis



GPU CARM: Performance, Power, DVFS



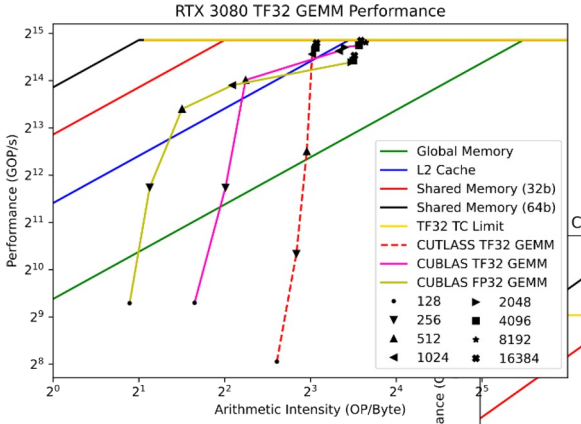
NUMA CARM: Multi-socket, KNL

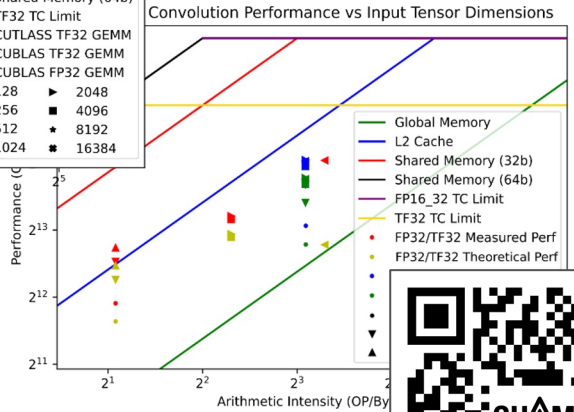


A. Ilic, F. Pratas, L. Sousa, "Beyond the Roofline: Cache-Aware Power and Energy-Efficiency Modeling for Multi-Cores", IEEE Trans. on Computers (2017)
A. Lopes, F. Pratas, L. Sousa, A. Ilic, "Exploring GPU performance, power and energy-efficiency bounds with Cache-aware Roofline Modeling", ISPASS (2017)
N. Denoyelle, B. Goglin, A. Ilic, E. Jeannot, L. Sousa, "Modeling Non-Uniform Memory Access on Large Compute Nodes with the Cache-Aware Roofline Model", IEEE TPDS (2018)

The CARM Tool: GPU Extension

- **Nvidia GPU + Tensor CARM**
  - Compute: CUDA + Tensor Cores
  - Memory: Global, L2 cache, Shared
- **AMD GPU Extension**
  - Targeting MI GPUs via HIP
  - AMD Roc Prof for profiling
- **GPU App Profiling: Nsight Compute**
  - Entire app or specific GPU kernels
  - Potential shown in profiling convolutions etc.





NVIDIA Nsight Compute

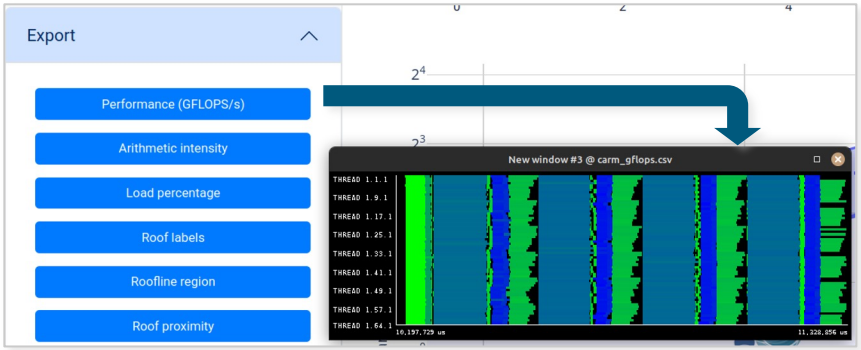
NVIDIA

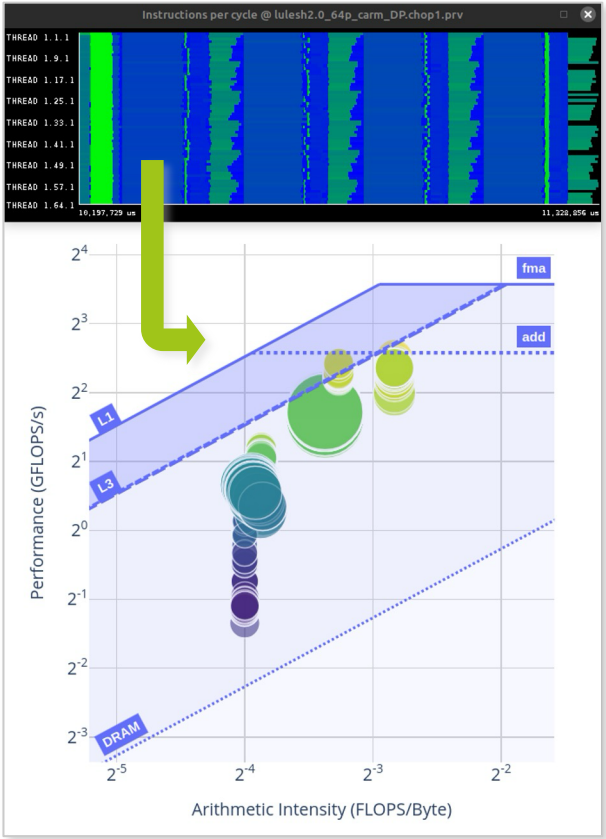
AMD

CHAMP

The CARM Tool: Paraver integration

- **CARM-based application profiling in Paraver**
  - Closley integrated with the Paraver Tool
- **CARM-Paraver integration features**
  - Synchronization with Paraver timelines
  - Importing Paraver timestamp colouring and masks
  - Exporting CARM-related metrics to Paraver





EuroHPC Supercomputers

- **POP3: CARM Tool deployment to HPC Centres**
  - Deployed: Jupiter, MareNostrum 5, Karolina, Deucalion
  - Soon™: LUMI, Vega, Discoverer, Leonardo and Meluxina

- **CARM Dissemination Initiatives**
  - POP3 Webinars; VI-HPS Tools Guide and Blog
  - https://www.vi-hps.org/tools/carm.html

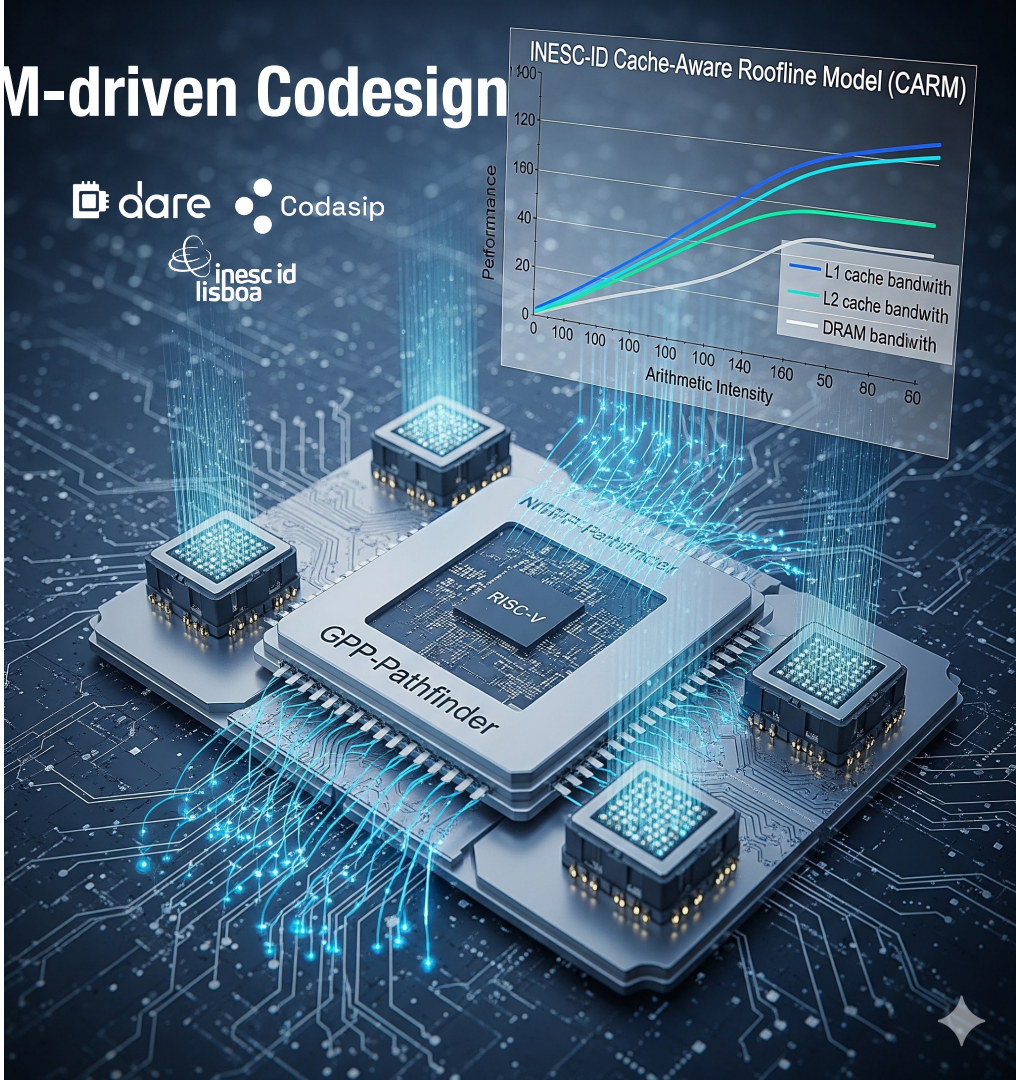


VI-HPS POP

EuroHPC Joint Undertaking

DARE Project: CARM-driven Codesign

- **DARE SGA1 Project**
  - Driving Europe's next-generation HPC systems
  - Focus on performance, efficiency, and sovereignty
- **Codasip: RISC-V GPP Pathfinding**
  - Low-level architecture-aware support for CARM
  - Help optimizing the apps running on the GPP core
- **Insights about characteristics of the architecture**
- **Collaborate with the DARE SW TA**



CARM Tool – Available on Github

- **The CARM Tool is open source and available on Github**
  - https://github.com/champ-hub/carm-roofline

- **The first of many tools to be developed in the scope of CHAMP hub**

CHAMP

Heterogeneous Computing and Performance Modeling Hub

Hub para Computação Heterogénea e Modelação de Performance



inesc id lisboa 20 YEARS

DEFINING TECHNOLOGY

Thank you!

dare

Co-funded by the European Union

EuroHPC Joint Undertaking

FCT Fundação para a Ciência e a Tecnologia

The DARE SGA1 project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101202459. The JU receives support from the European Union's Horizon Europe research and innovation programme and Spain, Germany, Czechia, Italy, Netherlands, Belgium, Finland, Greece, Croatia, Portugal, Poland, Sweden, France and Austria. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or of the granting authority. Neither the European Union nor the granting authority can be held responsible for