TÉCNICO LISBOA

The CARM Tool

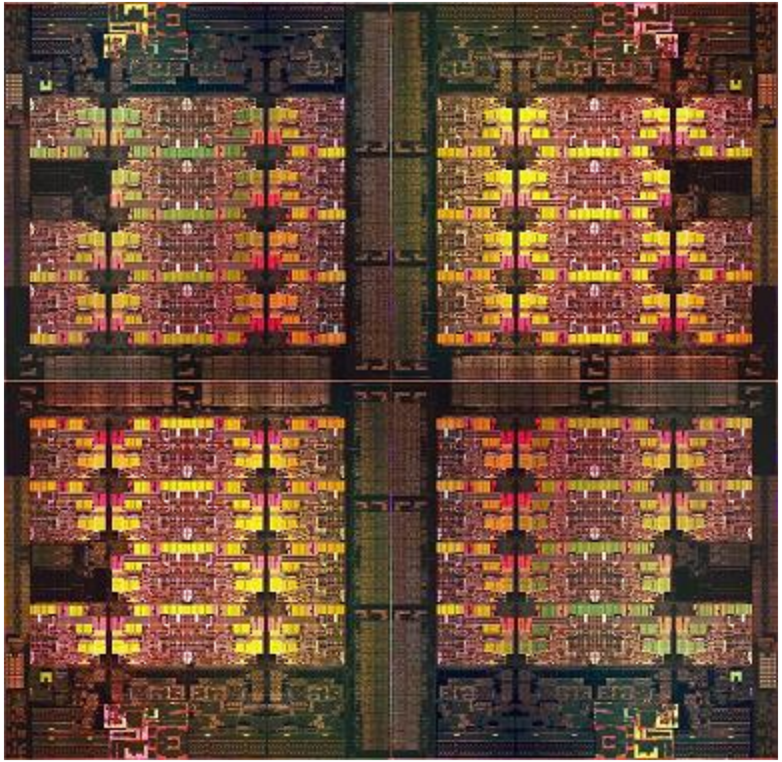
Cache-aware Roofline Model for HPC

José Morgado
Leonel Sousa
Aleksandar Ilic

28th POP Seminar
5 September 2024

Motivation

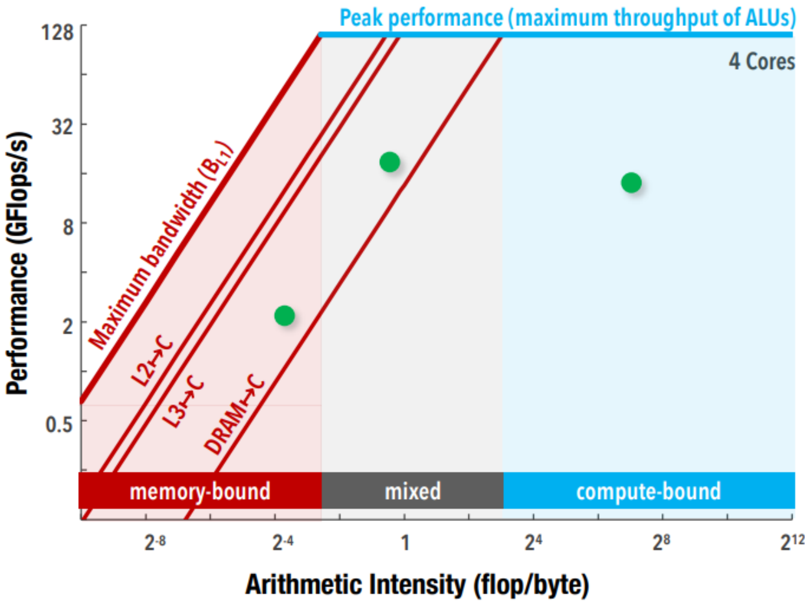
- Modern HPC systems and applications are **complex** and **heterogeneous**
  - Hard to model
  - Hard to optimize
- The CARM as a solution
  - Easy to understand
  - Accurate performance overview
  - Good optimization hints
- CARM is only supported by Intel Advisor



Intel Sapphire Rapids CPU

Motivation

- Modern HPC systems and applications are **complex** and **heterogeneous**
  - Hard to model
  - Hard to optimize
- The CARM as a solution
  - Easy to understand
  - Accurate performance overview
  - Good optimization hints
- CARM is only supported by Intel Advisor



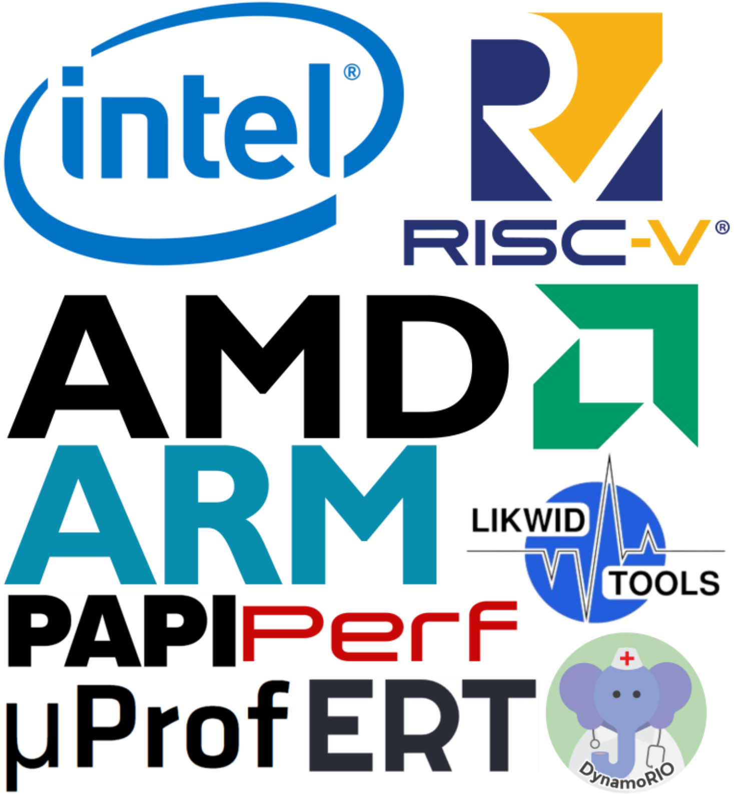
The Cache-Aware Roofline Model

Challenges

- Main contributions of the CARM Tool
  - Porting CARM to AMD/ARM/RISC-V
  - Providing application analysis in the scope of CARM
  - Combining all features in a single tool

- Porting CARM to other architectures
  - Requires tailored microbenchmarks
  - Understanding of underlying architecture

- Experimental results
  - CARM Architecture Analysis
  - CARM Application Analysis



TÉCNICO LISBOA

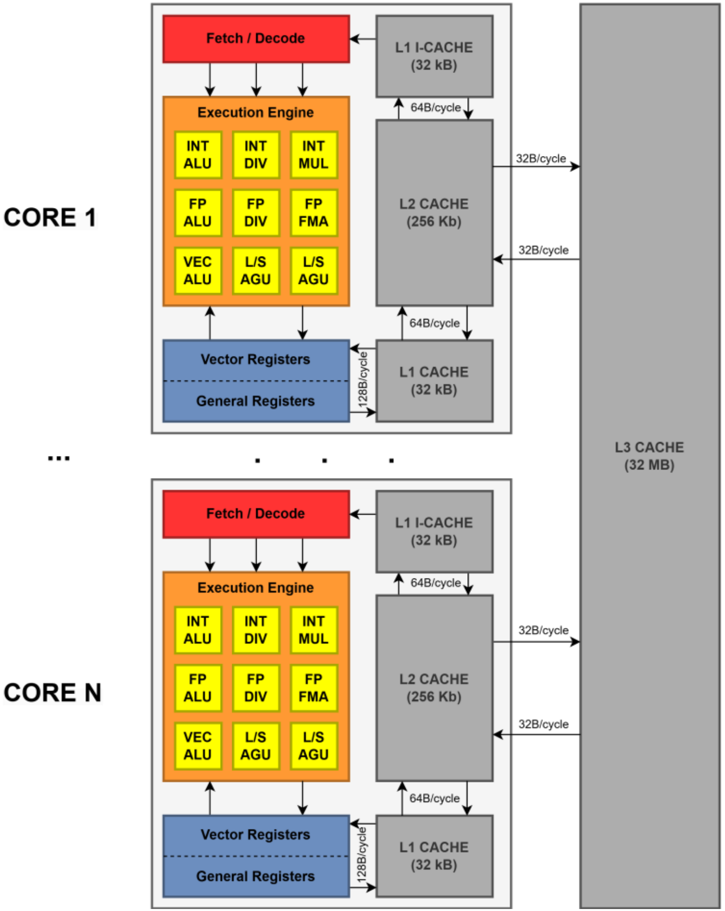
CONTENTS

- **State of the Art**
- The CARM Tool
- Results
- Conclusion

State of the Art

- State of the art CPUs
- The Cache-Aware Roofline Model
- State of the art Roofline Tools
- State of the art PMU and DBI Tools

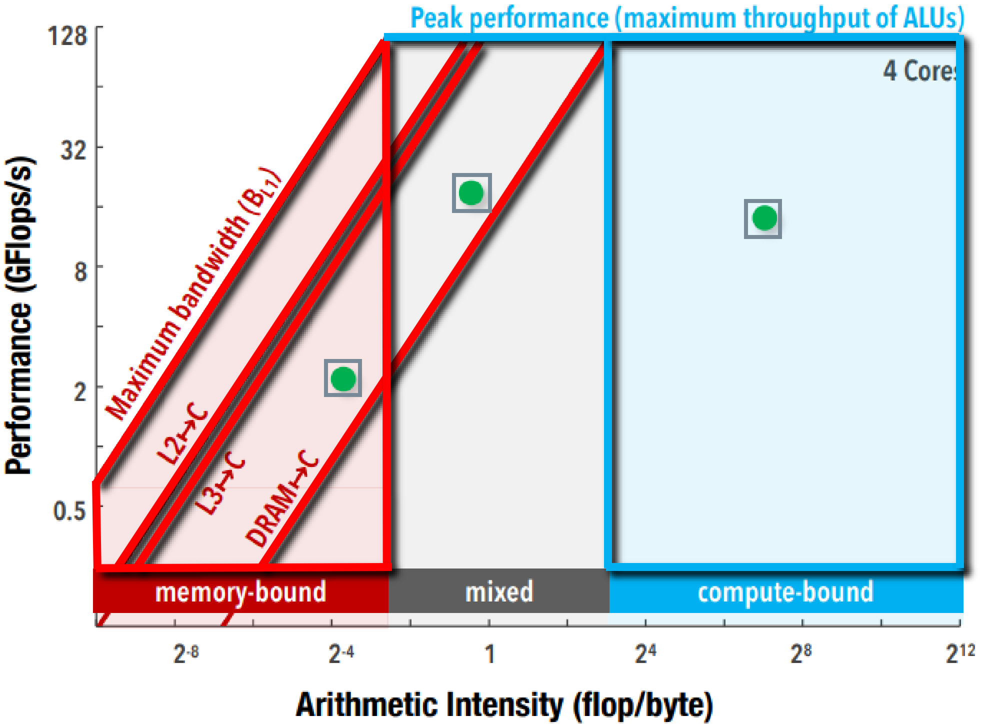
State of the art - CPUs



Vendor	Vector ISA Extensions
Intel	SSE AVX2 AVX-512
AMD	SSE AVX2 AVX-512
ARM	Neon SVE
RISC-V	RVV

The Cache-aware Roofline Model

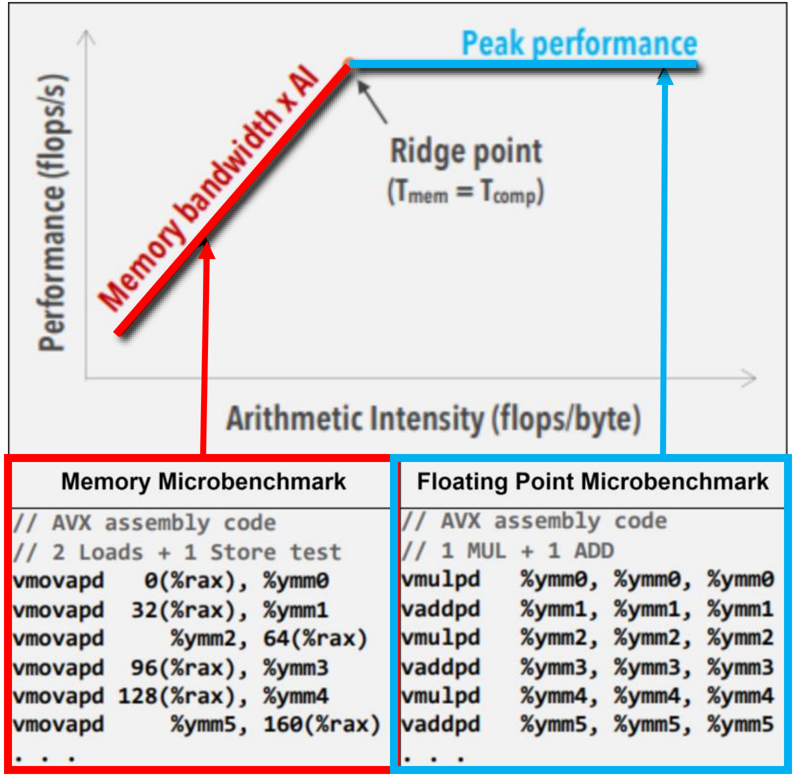
- How does CARM work?
  - Sloped Roof
  - Flat Roof
- How to generate CARM?
  - Floating-Point Microbenchmarks
  - Memory Microbenchmarks
- The CARM Tool
  - Automatic Benchmarking
  - Automatic CARM Generation



A. Ilic, F. Pratas and L. Sousa, "Cache-aware Roofline model: Upgrading the loft"

The Cache-aware Roofline Model

- How does CARM work?
  - Sloped Roof
  - Flat Roof
- How to generate CARM?
  - Floating-Point Microbenchmarks
  - Memory Microbenchmarks
- The CARM Tool
  - Automatic Benchmarking
  - Automatic CARM Generation



Memory Microbenchmark

```
// AVX assembly code
// 2 Loads + 1 Store test
vmovapd    0(%rax), %ymm0
vmovapd   32(%rax), %ymm1
vmovapd      %ymm2, 64(%rax)
vmovapd   96(%rax), %ymm3
vmovapd  128(%rax), %ymm4
vmovapd      %ymm5, 160(%rax)
. . .
```

Floating Point Microbenchmark

```
// AVX assembly code
// 1 MUL + 1 ADD
vmulpd    %ymm0, %ymm0, %ymm0
vaddpd    %ymm1, %ymm1, %ymm1
vmulpd    %ymm2, %ymm2, %ymm2
vaddpd    %ymm3, %ymm3, %ymm3
vmulpd    %ymm4, %ymm4, %ymm4
vaddpd    %ymm5, %ymm5, %ymm5
. . .
```

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

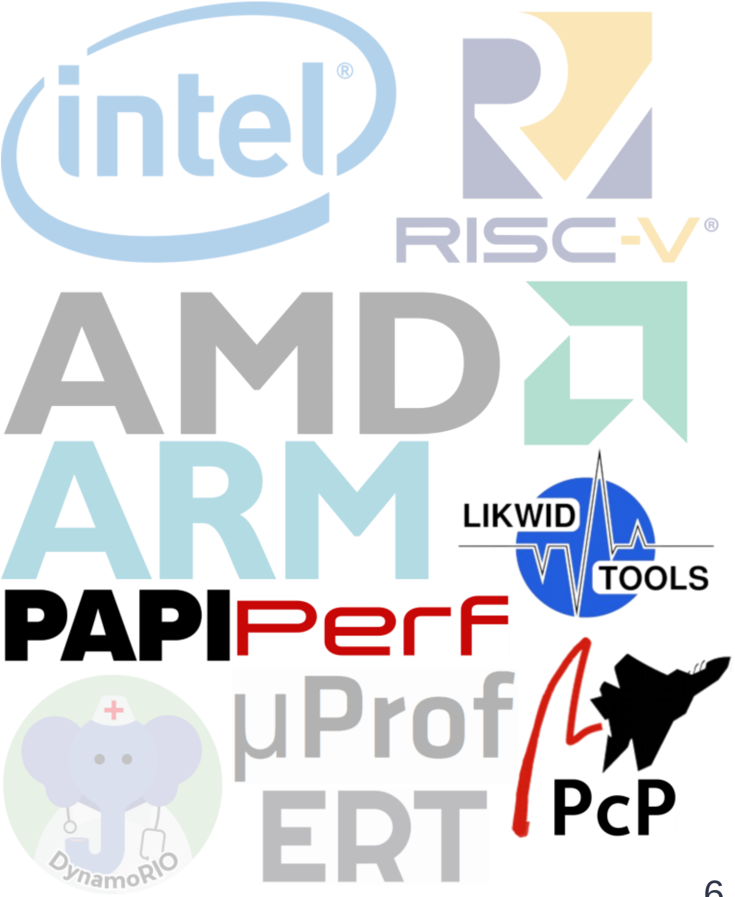
State of the art – Roofline Tools

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

State of the art – PMU Tools

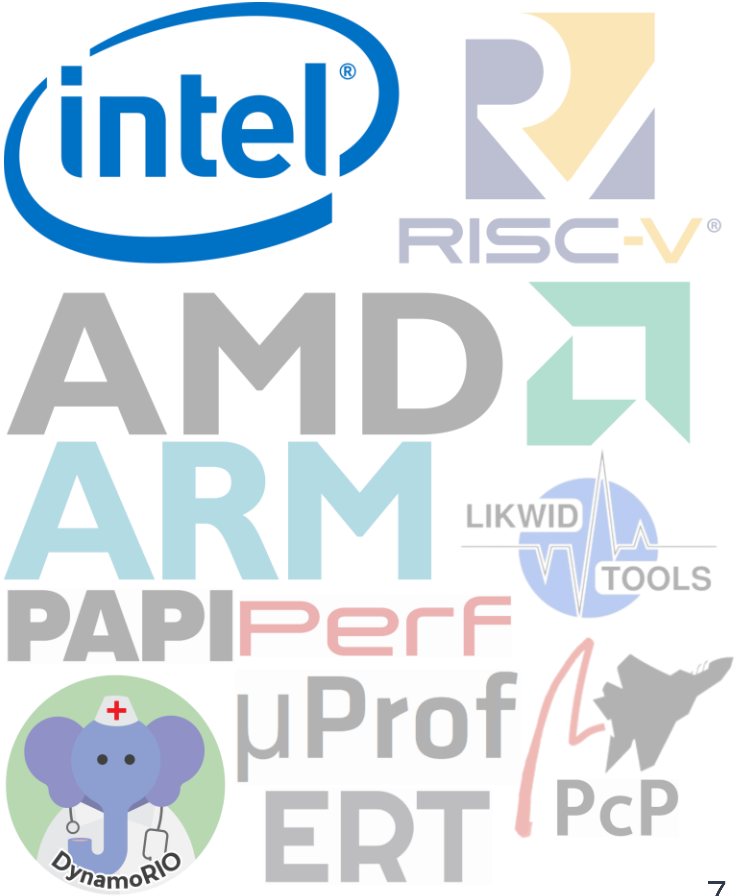
Tools Features	PAPI	Perf
Supports	Intel | AMD ARM	Intel | AMD ARM | RISC-V
Advantage	Portability	Availability
Disadvantage	Only ROI*	Portability

*ROI – Region of Interest



State of the art – DBI Tools

Tools Features	Intel SDE	DynamoRIO
Supports	Intel | AMD	Intel | AMD | ARM ? RISC-V ?
Advantage	Detailed Analysis	Customizable Analysis
Disadvantage	Not Customizable	Overhead
Open-Source	No	Yes



CONTENTS

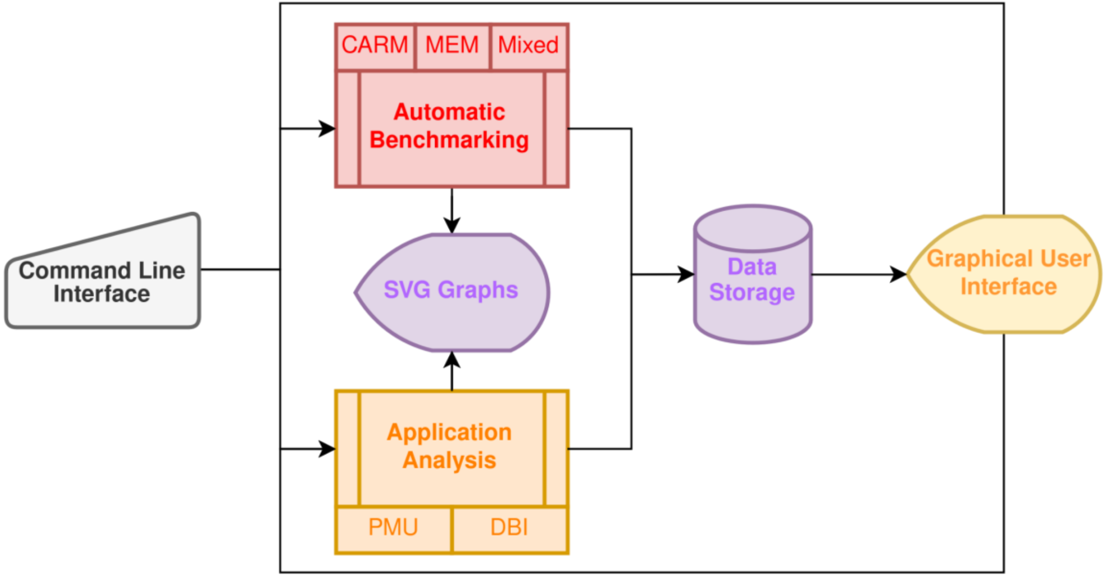
- State of the Art
- **The CARM Tool**
- Results
- Conclusion

The CARM Tool

- High-Level Overview
  - Graphical User Interface
  - Automatic Benchmarking
  - Application Analysis
- Low-Level Overview
  - Benchmark Generation
  - Frequency Measuring
  - Benchmarking

The CARM Tool – High Level

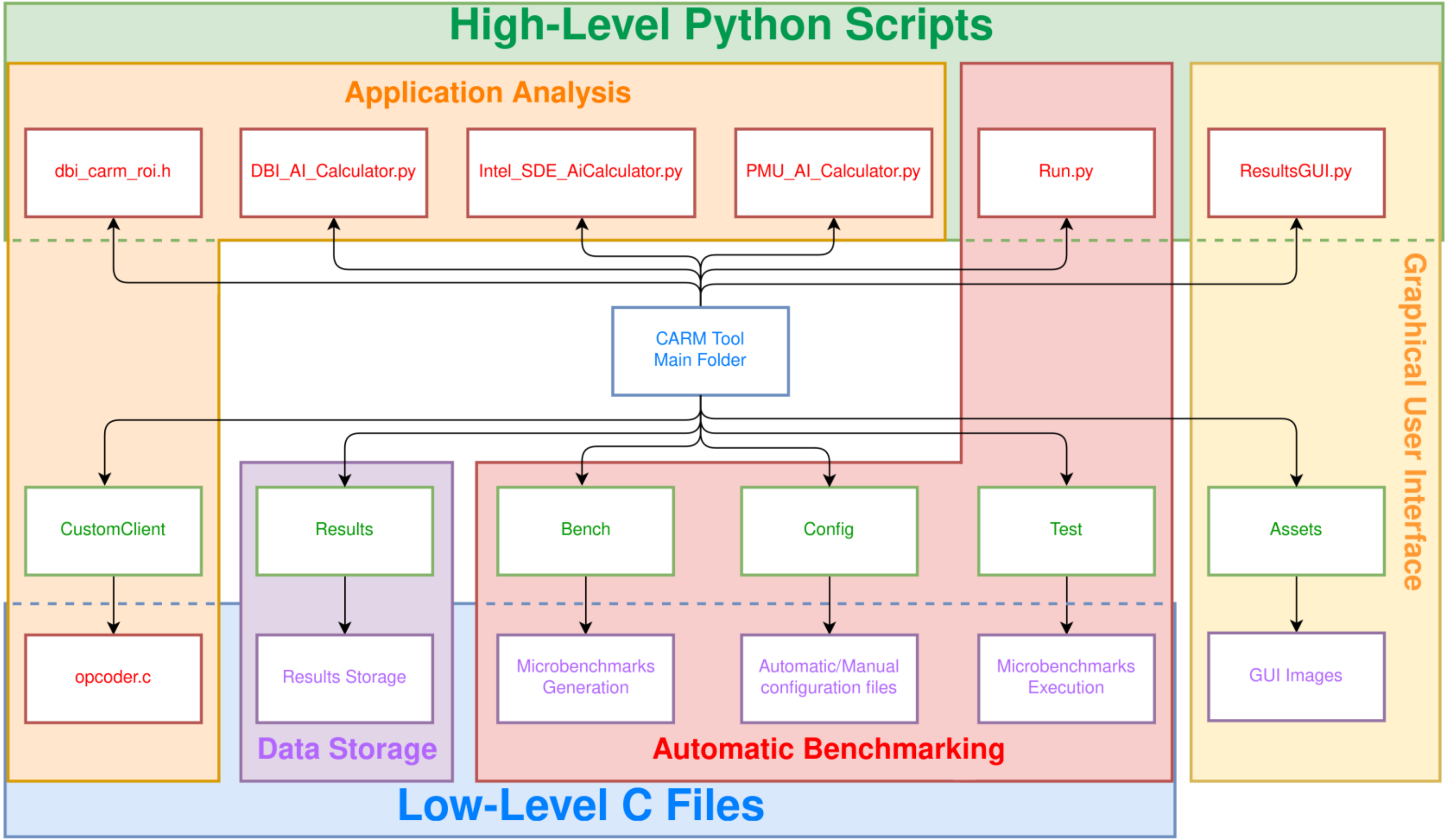
- Interfacing
  - Command Line Interface
  - Graphical User Interface
- Automatic Benchmarking
  - CARM Benchmarks
  - Memory / FP / Mixed Benchmarks
- Application Analysis
  - PMU Analysis
  - DBI Analysis



The CARM Tool – Supported ISAs

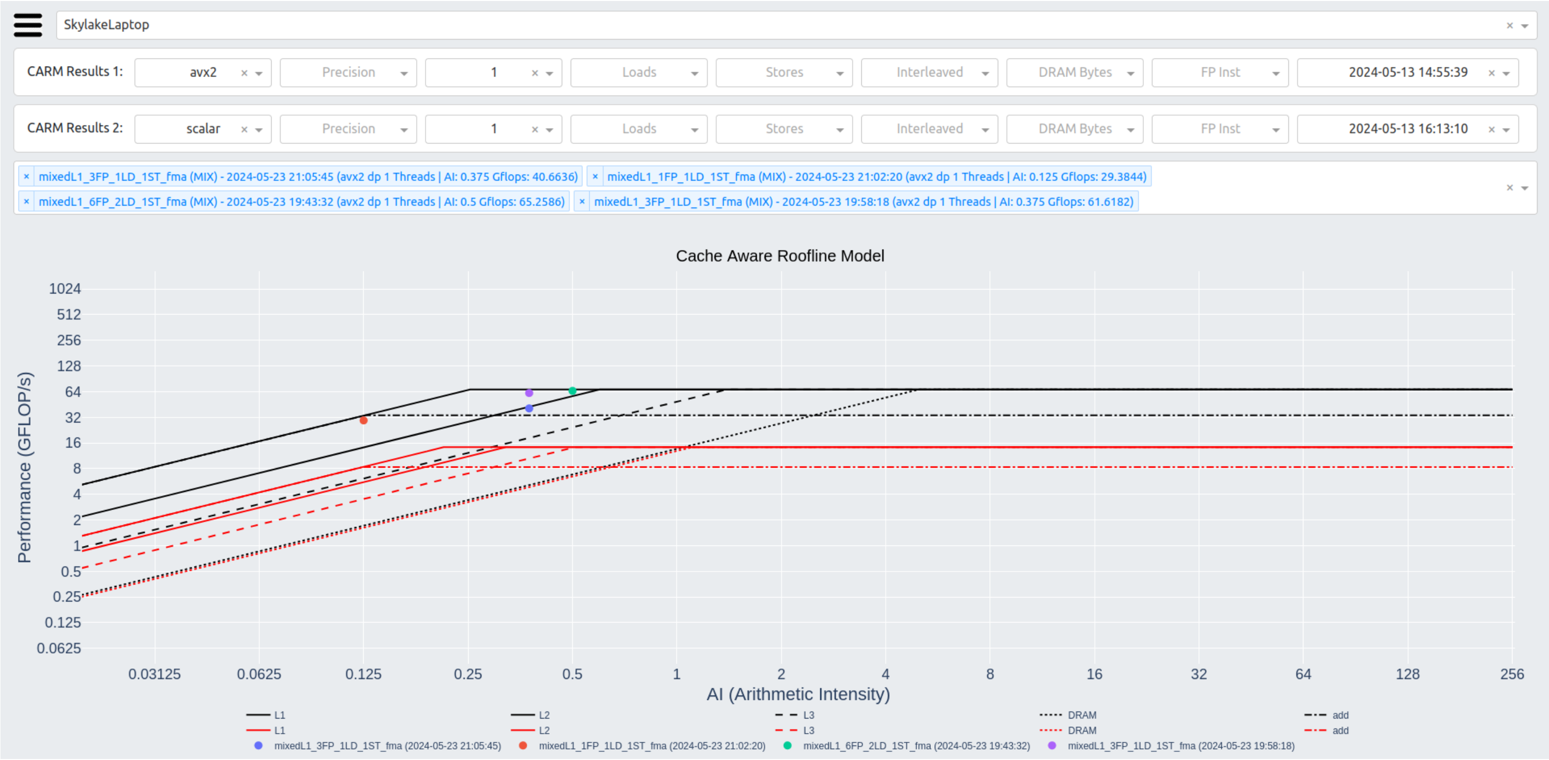
intel®	AMD	arm	RISC-V
Scalar	Scalar	Scalar	Scalar
SSE	SSE	Neon	RVV 0.7.1 | 1.0
AVX2	AVX2	SVE	
AVX512	AVX512		

The CARM Tool – High Level



The CARM Tool – Graphical User Interface

Results Visualization



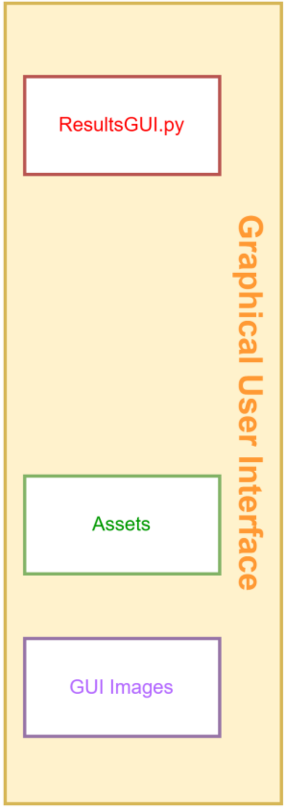
SkylakeLaptop

CARM Results 1: avx2 | Precision | 1 | Loads | Stores | Interleaved | DRAM Bytes | FP Inst | 2024-05-13 14:55:39

CARM Results 2: scalar | Precision | 1 | Loads | Stores | Interleaved | DRAM Bytes | FP Inst | 2024-05-13 16:13:10

- mixedL1_3FP_1LD_1ST_fma (MIX) - 2024-05-23 21:05:45 (avx2 dp 1 Threads | AI: 0.375 Gflops: 40.6636)
- mixedL1_1FP_1LD_1ST_fma (MIX) - 2024-05-23 21:02:20 (avx2 dp 1 Threads | AI: 0.125 Gflops: 29.3844)
- mixedL1_6FP_2LD_1ST_fma (MIX) - 2024-05-23 19:43:32 (avx2 dp 1 Threads | AI: 0.5 Gflops: 65.2586)
- mixedL1_3FP_1LD_1ST_fma (MIX) - 2024-05-23 19:58:18 (avx2 dp 1 Threads | AI: 0.375 Gflops: 61.6182)

Cache Aware Roofline Model
Performance (GFLOP/s)
1024
512
256
128
64
32
16
8
4
2
1
0.5
0.25
0.125
0.0625
0.03125
0.0625
0.125
0.25
0.5
1
2
4
8
16
32
64
128
256
AI (Arithmetic Intensity)
L1
L2
L3
DRAM
add
L1
L2
L3
DRAM
add
mixedL1_3FP_1LD_1ST_fma (2024-05-23 21:05:45)
mixedL1_1FP_1LD_1ST_fma (2024-05-23 21:02:20)
mixedL1_6FP_2LD_1ST_fma (2024-05-23 19:43:32)
mixedL1_3FP_1LD_1ST_fma (2024-05-23 19:58:18)



Graphical User Interface

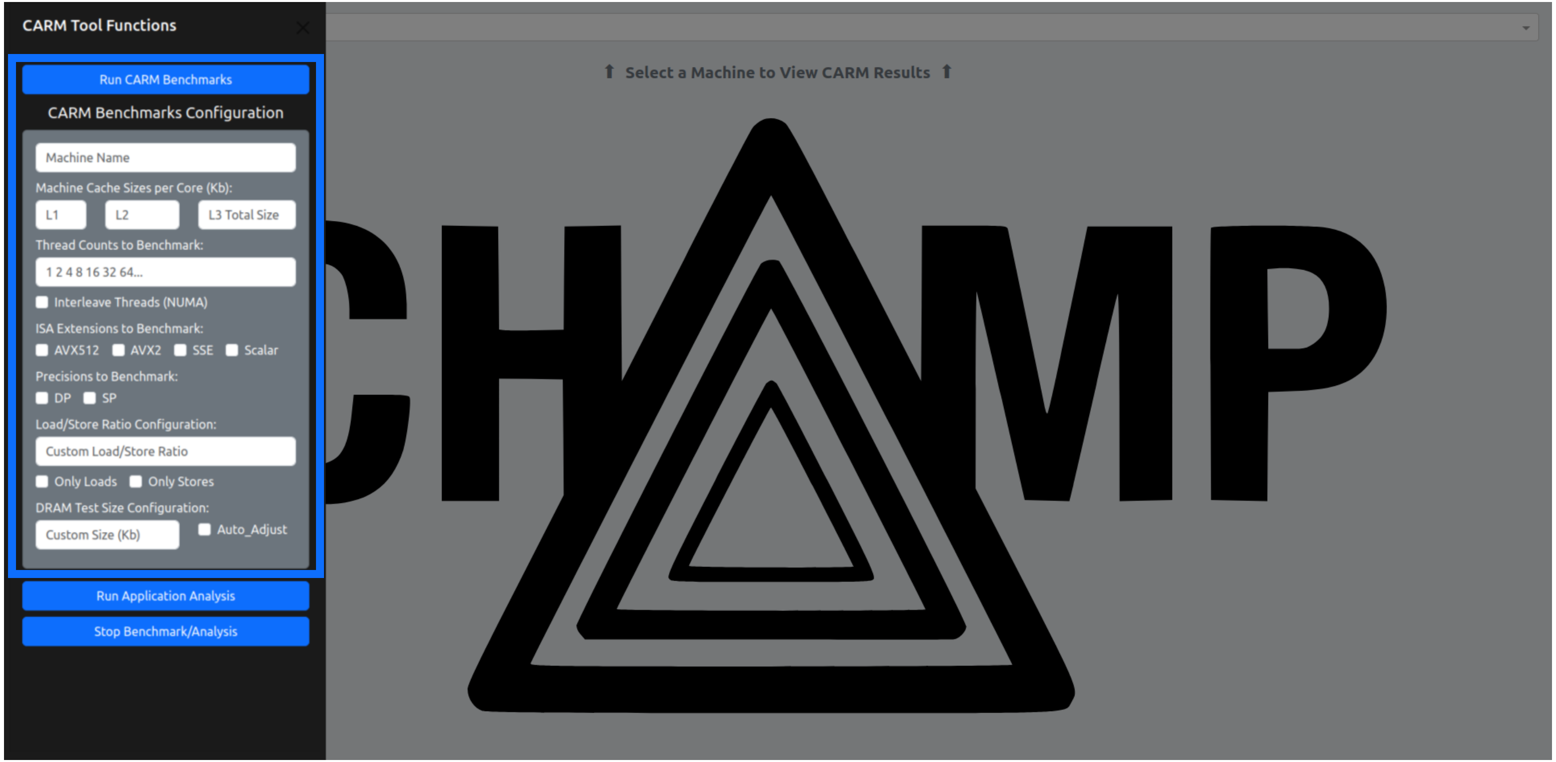
ResultsGUI.py

Assets

GUI Images

The CARM Tool – Graphical User Interface

Benchmark Execution



CARM Benchmarks Configuration

Machine Name

Machine Cache Sizes per Core (Kb):

L1 L2 L3 Total Size

Thread Counts to Benchmark:

1 2 4 8 16 32 64...

□ Interleave Threads (NUMA)

ISA Extensions to Benchmark:

□ AVX512 □ AVX2 □ SSE □ Scalar

Precisions to Benchmark:

□ DP □ SP

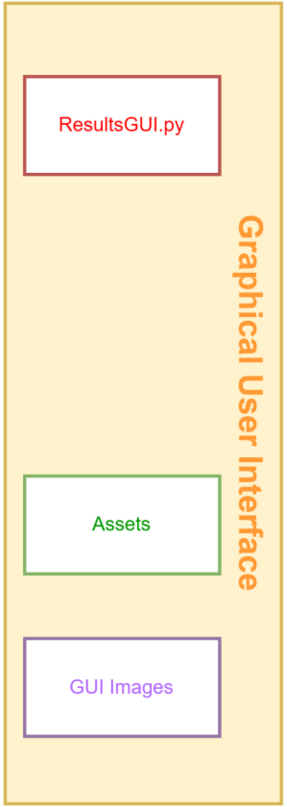
Load/Store Ratio Configuration:

Custom Load/Store Ratio

□ Only Loads □ Only Stores

DRAM Test Size Configuration:

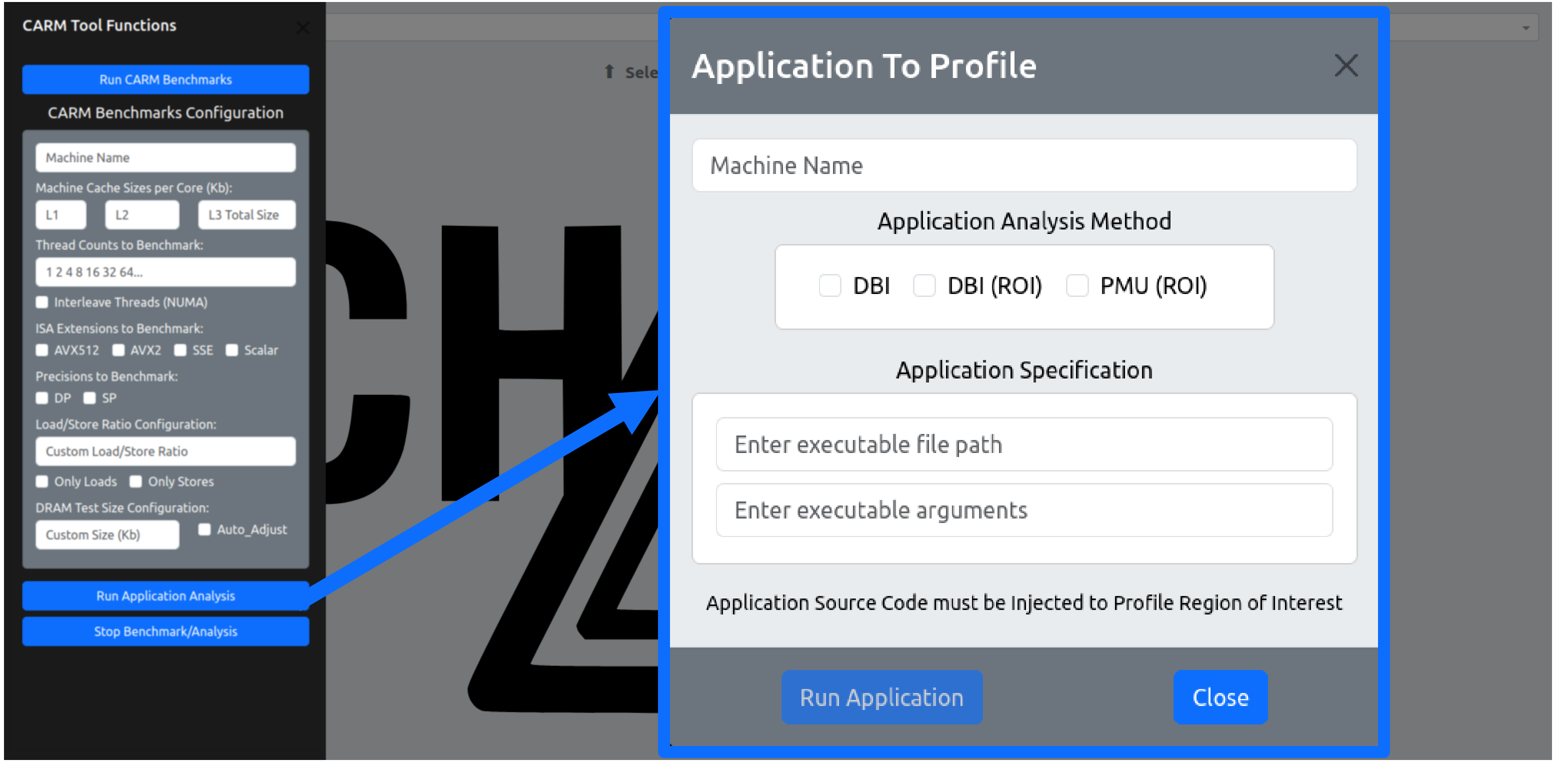
Custom Size (Kb) □ Auto_Adjust



TÉCNICO LISBOA

The CARM Tool – Graphical User Interface

Application Profiling



CARM Tool Functions

Run CARM Benchmarks

CARM Benchmarks Configuration

Machine Name

Machine Cache Sizes per Core (Kb):

L1 | L2 | L3 Total Size

Thread Counts to Benchmark:

1 2 4 8 16 32 64...

□ Interleave Threads (NUMA)

ISA Extensions to Benchmark:

□ AVX512 □ AVX2 □ SSE □ Scalar

Precisions to Benchmark:

□ DP □ SP

Load/Store Ratio Configuration:

Custom Load/Store Ratio

□ Only Loads □ Only Stores

DRAM Test Size Configuration:

Custom Size (Kb) □ Auto_Adjust

Run Application Analysis

Stop Benchmark/Analysis

Application To Profile

Machine Name

Application Analysis Method

□ DBI □ DBI (ROI) □ PMU (ROI)

Application Specification

Enter executable file path

Enter executable arguments

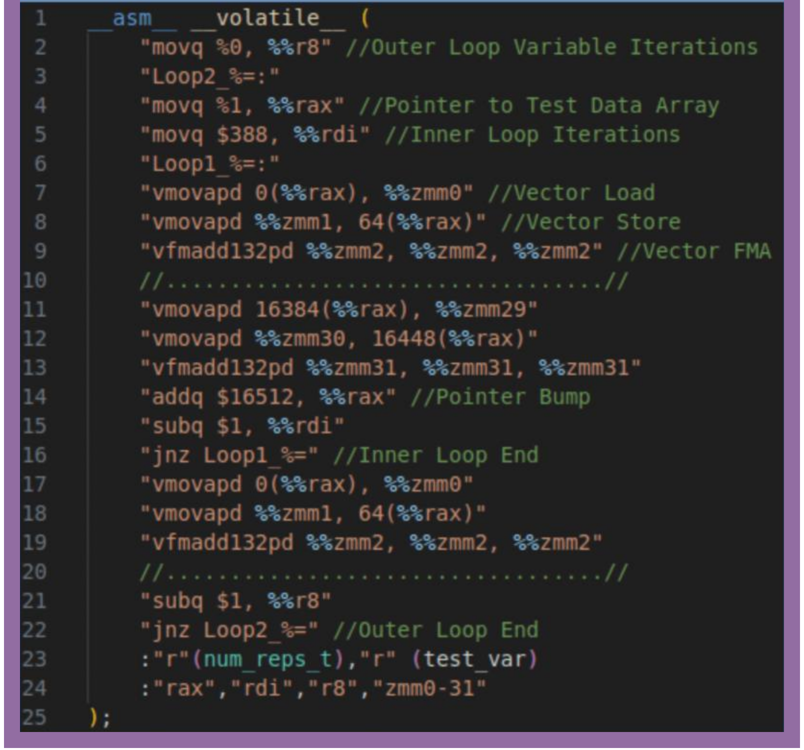
Application Source Code must be Injected to Profile Region of Interest

Run Application | Close

ResultsGUI.py
Assets
GUI Images
Graphical User Interface

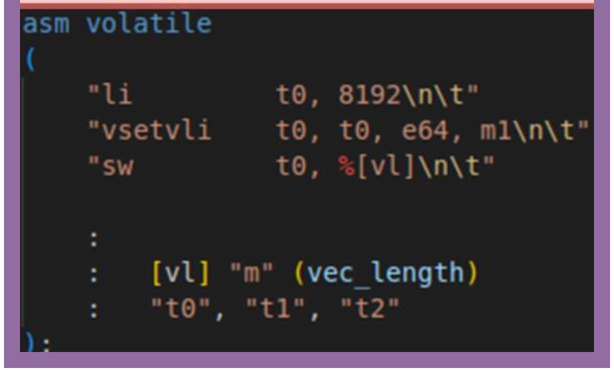
The CARM Tool – Automatic Benchmarking

AVX512 Microbenchmarks Pseudo-Code



```
__asm__ __volatile__ (
    "movq %0, %%r8" //Outer Loop Variable Iterations
    "Loop2_%=:"
    "movq %1, %%rax" //Pointer to Test Data Array
    "movq $388, %%rdi" //Inner Loop Iterations
    "Loop1_%=:"
    "vmovapd 0(%%rax), %%zmm0" //Vector Load
    "vmovapd %%zmm1, 64(%%rax)" //Vector Store
    "vfmadd132pd %%zmm2, %%zmm2, %%zmm2" //Vector FMA
    //.................................//
    "vmovapd 16384(%%rax), %%zmm29"
    "vmovapd %%zmm30, 16448(%%rax)"
    "vfmadd132pd %%zmm31, %%zmm31, %%zmm31"
    "addq $16512, %%rax" //Pointer Bump
    "subq $1, %%rdi"
    "jnz Loop1_%=" //Inner Loop End
    "vmovapd 0(%%rax), %%zmm0"
    "vmovapd %%zmm1, 64(%%rax)"
    "vfmadd132pd %%zmm2, %%zmm2, %%zmm2"
    //.................................//
    "subq $1, %%r8"
    "jnz Loop2_%=" //Outer Loop End
    :"r"(num_reps_t),"r" (test_var)
    :"rax","rdi","r8","zmm0-31"
);
```

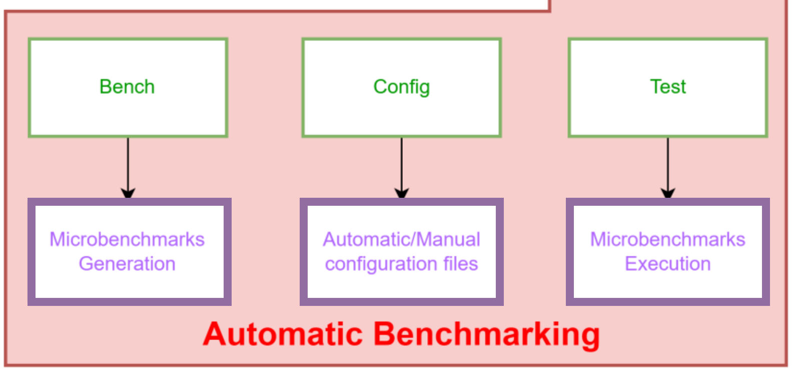
RISC-V RVV Vector Length Detection



```
asm volatile
(
    "li         t0, 8192\n\t"
    "vsetvli    t0, t0, e64, m1\n\t"
    "sw         t0, %[vl]\n\t"

    :
    :    [vl] "m" (vec_length)
    :    "t0", "t1", "t2"
);
```

Run.py



Bench

Config

Test

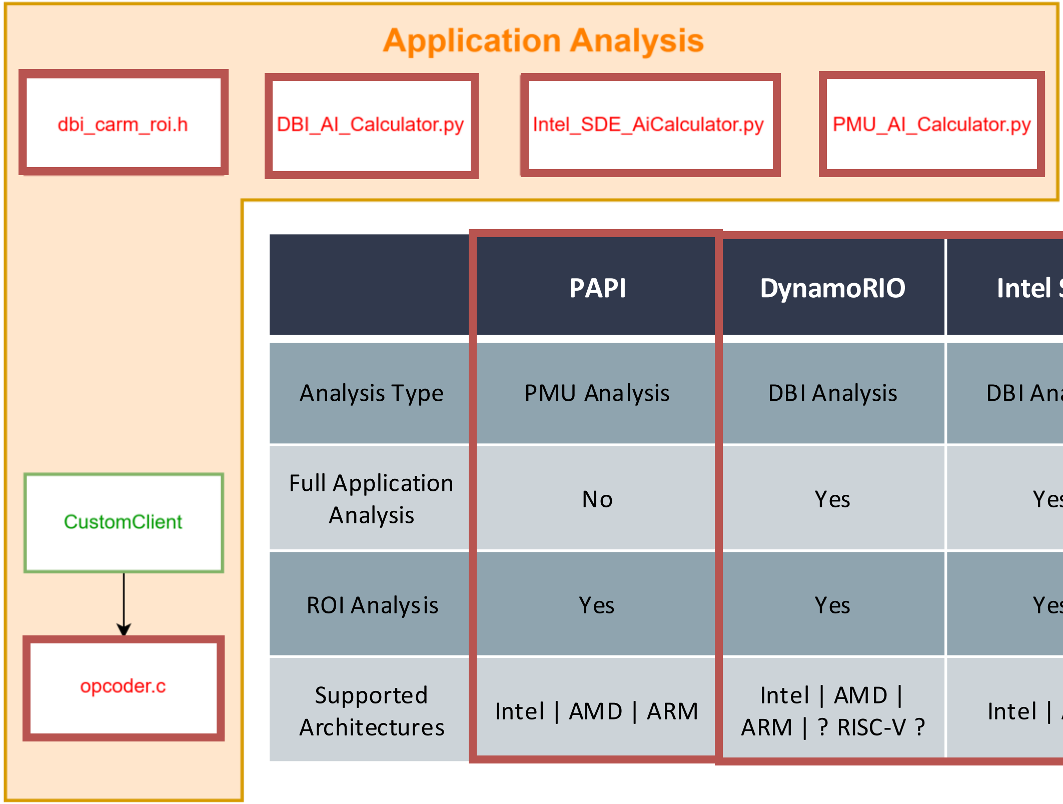
Microbenchmarks Generation

Automatic/Manual configuration files

Microbenchmarks Execution

Automatic Benchmarking

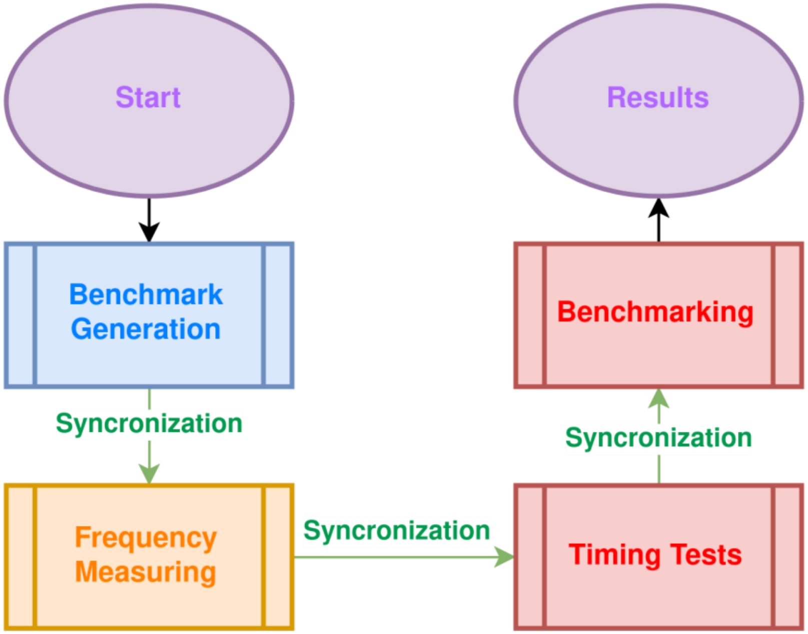
The CARM Tool – Application Analysis



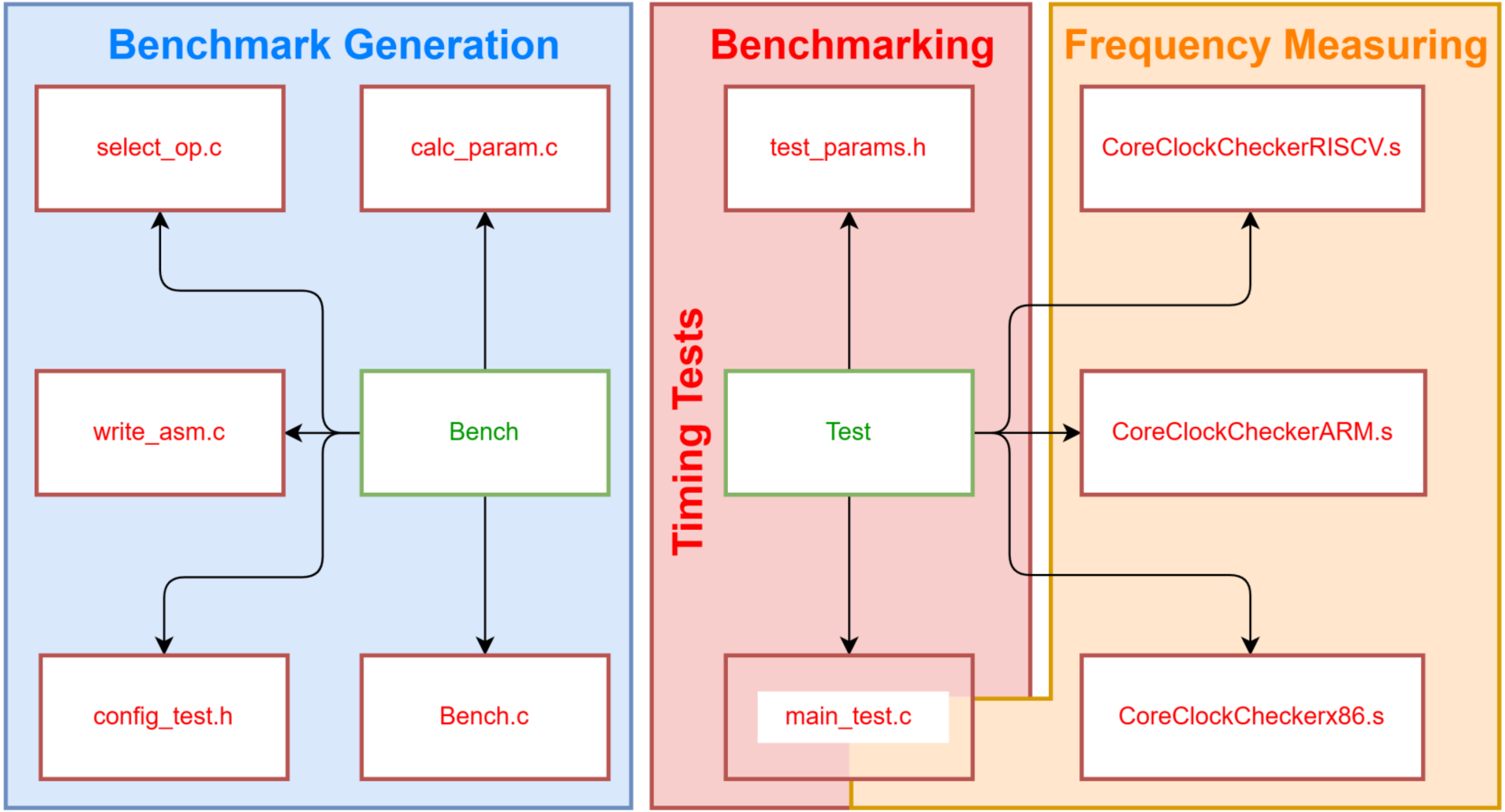
	PAPI	DynamoRIO	Intel SDE
Analysis Type	PMU Analysis	DBI Analysis	DBI Analysis
Full Application Analysis	No	Yes	Yes
ROI Analysis	Yes	Yes	Yes
Supported Architectures	Intel | AMD | ARM	Intel | AMD | ARM | ? RISC-V ?	Intel | AMD

The CARM Tool – Low Level

- Benchmark Generation
  - Following a general structure
  - Adapted to each ISA extension
- Frequency Measuring
  - Assembly based
  - Adapted to each ISA
- Benchmarking
  - Timing tests
  - Actual benchmarking

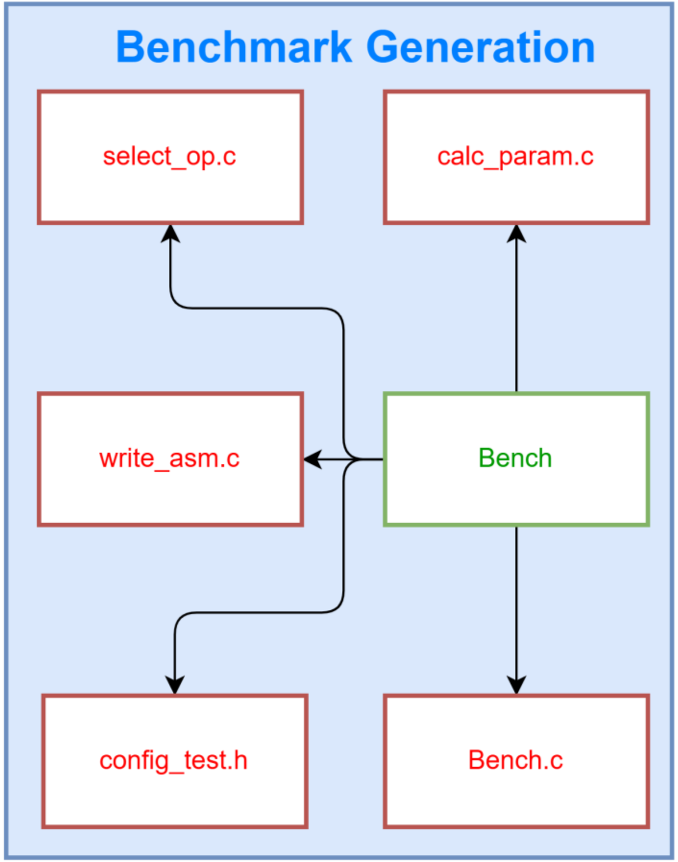


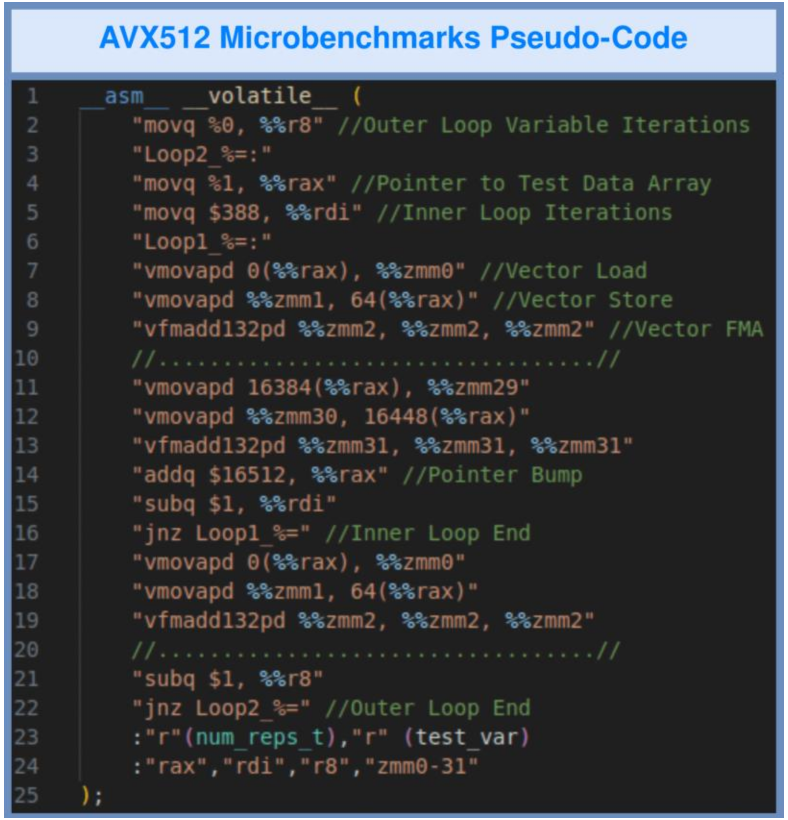
The CARM Tool – Low Level



TÉCNICO LISBOA

The CARM Tool – Benchmark Generation



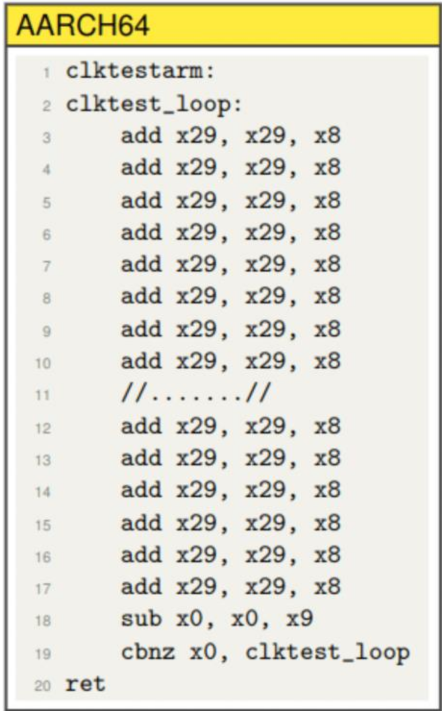

AVX512 Microbenchmarks Pseudo-Code

```
__asm__ __volatile__ (
    "movq %0, %%r8" //Outer Loop Variable Iterations
    "Loop2_%=:"
    "movq %1, %%rax" //Pointer to Test Data Array
    "movq $388, %%rdi" //Inner Loop Iterations
    "Loop1_%=:"
    "vmovapd 0(%%rax), %%zmm0" //Vector Load
    "vmovapd %%zmm1, 64(%%rax)" //Vector Store
    "vfmadd132pd %%zmm2, %%zmm2, %%zmm2" //Vector FMA
    //................................//
    "vmovapd 16384(%%rax), %%zmm29"
    "vmovapd %%zmm30, 16448(%%rax)"
    "vfmadd132pd %%zmm31, %%zmm31, %%zmm31"
    "addq $16512, %%rax" //Pointer Bump
    "subq $1, %%rdi"
    "jnz Loop1_%=" //Inner Loop End
    "vmovapd 0(%%rax), %%zmm0"
    "vmovapd %%zmm1, 64(%%rax)"
    "vfmadd132pd %%zmm2, %%zmm2, %%zmm2"
    //................................//
    "subq $1, %%r8"
    "jnz Loop2_%=" //Outer Loop End
    :"r"(num_reps_t),"r" (test_var)
    :"rax","rdi","r8","zmm0-31"
);
```

The CARM Tool – Frequency Measuring

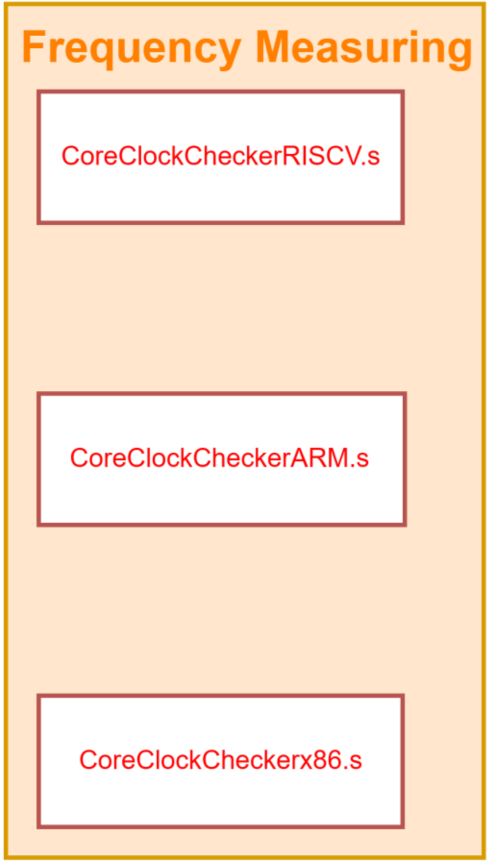
- Frequency Measuring Approach
  - Assembly function that leads to 1 IPC
  - Adapted to each ISA
  - Under 1% error on all tested machines

- Timing methods considered
  - Time Stamp Counter (TSC) – Intel | AMD
  - Clockgettime function – ARM | RISC-V



AARCH64

```
clktestarm:
clktest_loop:
     add x29, x29, x8
     add x29, x29, x8
     add x29, x29, x8
     add x29, x29, x8
     add x29, x29, x8
     add x29, x29, x8
     add x29, x29, x8
     add x29, x29, x8
     //......//
     add x29, x29, x8
     add x29, x29, x8
     add x29, x29, x8
     add x29, x29, x8
     add x29, x29, x8
     add x29, x29, x8
     sub x0, x0, x9
     cbnz x0, clktest_loop
ret
```



Frequency Measuring

CoreClockCheckerRISCV.s

CoreClockCheckerARM.s

CoreClockCheckerx86.s

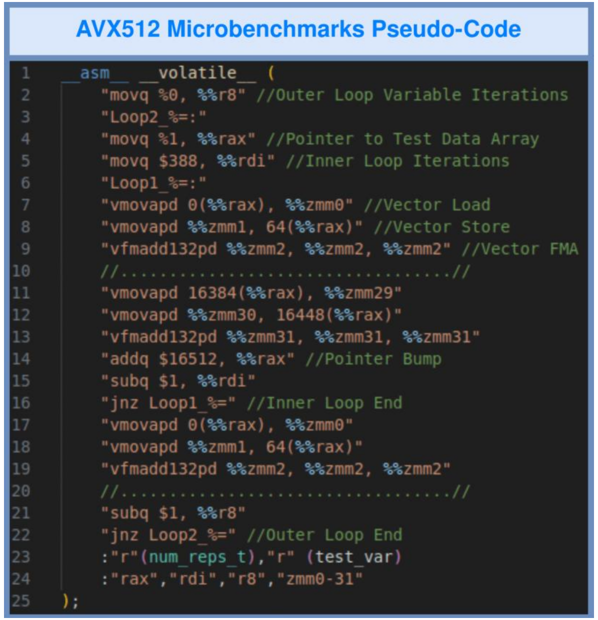
The CARM Tool – Bechmarking

Timing Tests

- Preliminary execution of benchmarks
- Ensures benchmarks last enough time
- Avoids unnecessarily long benchmarking

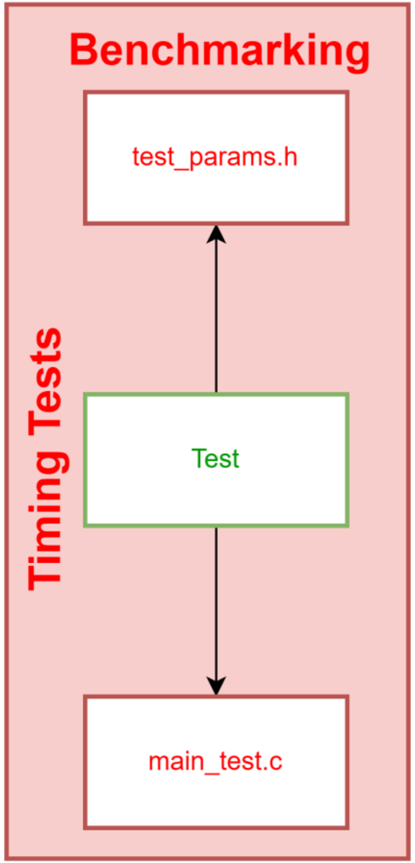
Benchmarking

- Tests are repeated 1024 times
- Thread barriers ensure parallel execution
- Best run per thread is selected



AVX512 Microbenchmarks Pseudo-Code

```
__asm__ __volatile__ (
    "movq %0, %%r8" //Outer Loop Variable Iterations
    "Loop2_%=:"
    "movq %1, %%rax" //Pointer to Test Data Array
    "movq $388, %%rdi" //Inner Loop Iterations
    "Loop1_%=:"
    "vmovapd 0(%%rax), %%zmm0" //Vector Load
    "vmovapd %%zmm1, 64(%%rax)" //Vector Store
    "vfmadd132pd %%zmm2, %%zmm2, %%zmm2" //Vector FMA
    //..................................//
    "vmovapd 16384(%%rax), %%zmm29"
    "vmovapd %%zmm30, 16448(%%rax)"
    "vfmadd132pd %%zmm31, %%zmm31, %%zmm31"
    "addq $16512, %%rax" //Pointer Bump
    "subq $1, %%rdi"
    "jnz Loop1_%=" //Inner Loop End
    "vmovapd 0(%%rax), %%zmm0"
    "vmovapd %%zmm1, 64(%%rax)"
    "vfmadd132pd %%zmm2, %%zmm2, %%zmm2"
    //..................................//
    "subq $1, %%r8"
    "jnz Loop2_%=" //Outer Loop End
    :"r"(num_reps_t),"r" (test_var)
    :"rax","rdi","r8","zmm0-31"
);
```



CONTENTS

- State of the Art
- The CARM Tool
- **Results**
- Conclusion

Results

- CARM based Architecture Analysis
- Comparison with state of the art Roofline Tools
  - Intel Advisor
  - Empirical Roofline Toolkit
- CARM based SpMV Application Analysis

Results - CARM Architecture Analysis

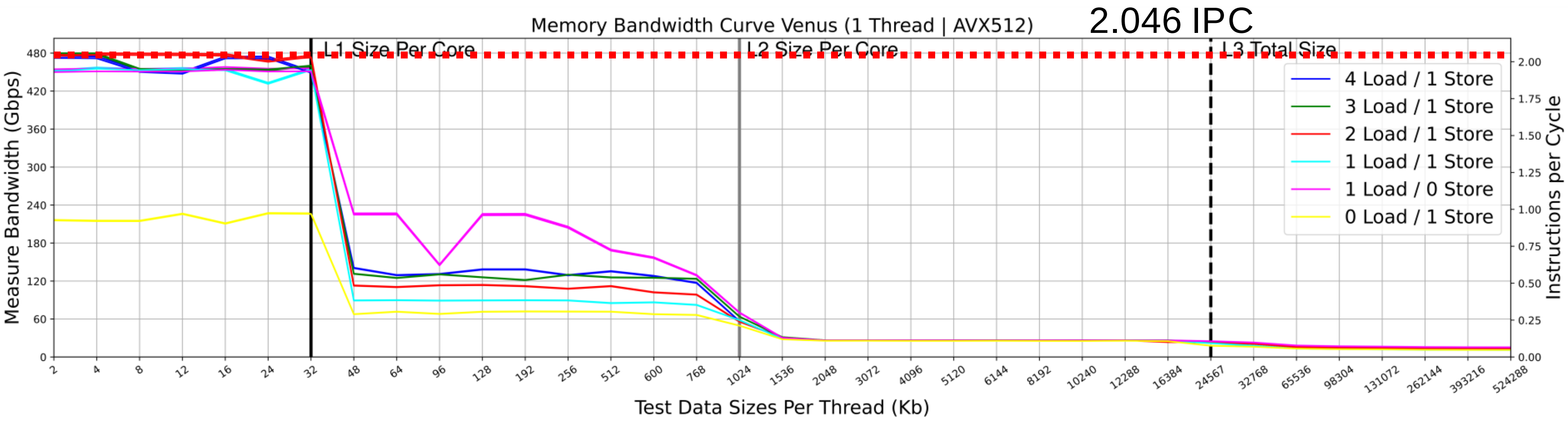
- Machines Utilized
  - One from each vendor
  - Covering all ISA extensions supported

- Analysis Objective
  - Verify if CARM benchmarks can reach architectural limits

	Venus	Cara	Armq	Milk-V
Vendor	Intel	AMD	ARM	RISC-V
Architecture	Skylake-X	Zen 3	Vulcan	XuanTie C920
ISA Extensions	SSE | AVX2 | AVX-512	SSE | AVX2	Neon	RVV 0.7.1
FP | LD/ST Units	2 | 3	2 | 3	2 | 2	2 | 1

Results - CARM Memory Analysis

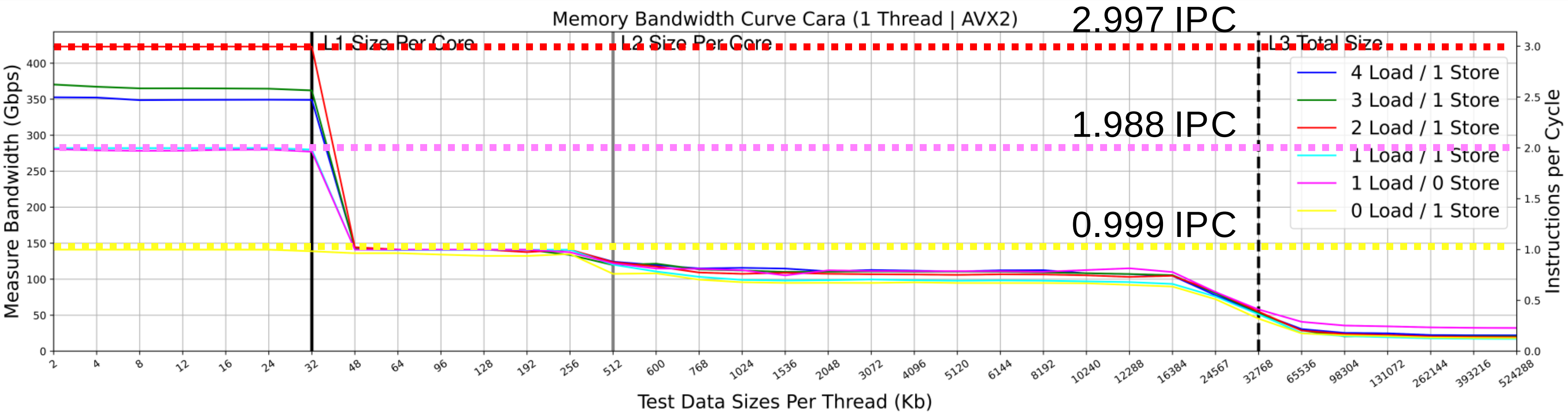
- Best Load/Store Ratio
  - Follows Ld/St unit ratio
  - Loads outperform stores
- Memory Architectural Limits
  - Accurately achieved for L1
  - Progressively harder to reach for lower memory levels



Results - CARM Memory Analysis

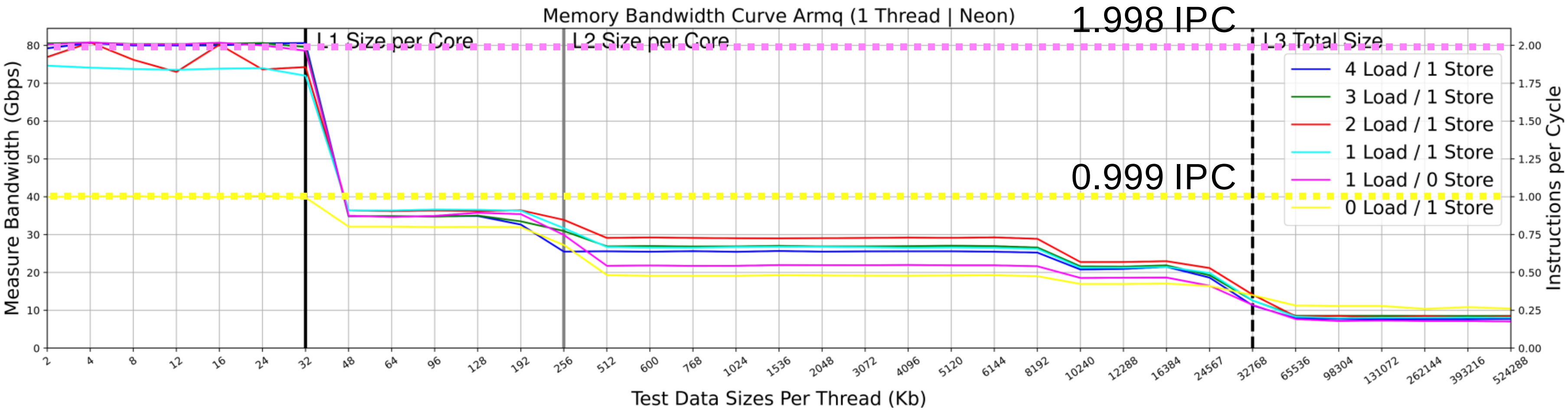
- Best Load/Store Ratio
  - Follows Ld/St unit ratio
  - Loads outperform stores

- Memory Architectural Limits
  - Accurately achieved for L1
  - Progressively harder to reach for lower memory levels



Results - CARM Memory Analysis

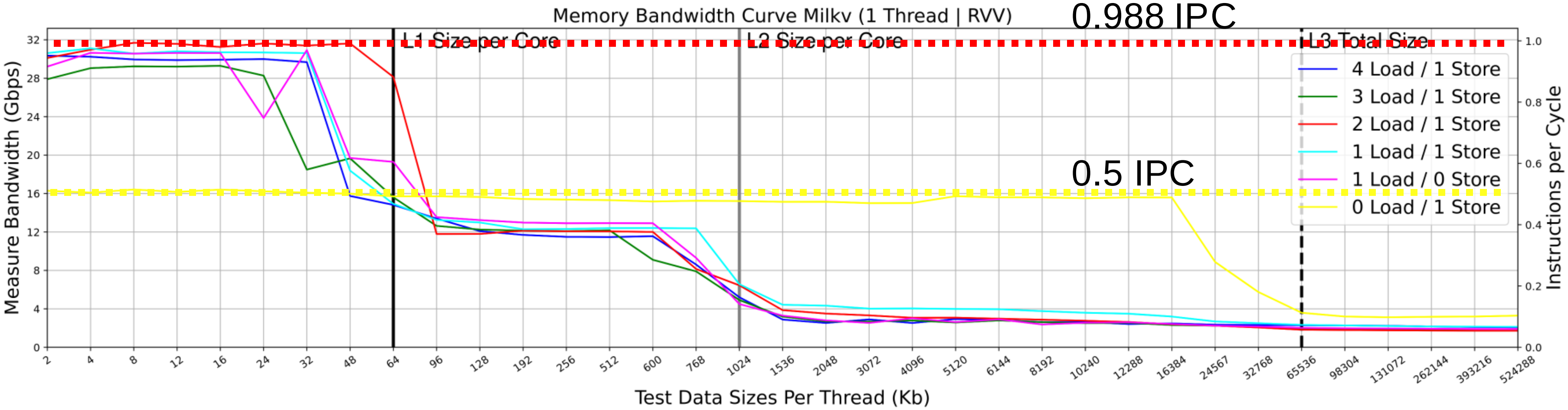
- Best Load/Store Ratio
  - Follows Ld/St unit ratio
  - Loads outperform stores
- Memory Architectural Limits
  - Accurately achieved for L1
  - Progressively harder to reach for lower memory levels



Results - CARM Memory Analysis

- Best Load/Store Ratio
  - Follows Ld/St unit ratio
  - Loads outperform stores

- Memory Architectural Limits
  - Accurately achieved for L1
  - Progressively harder to reach for lower memory levels



Results - CARM Memory Analysis

- Best Load/Store Ratio
  - Follows Ld/St unit ratio
  - Loads outperform stores
- Memory Architectural Limits
  - Accurately achieved for L1
  - Progressively harder to reach for lower memory levels

Deviation / IPC*	Venus	Cara	Armq	Milk-V
L1	-1.54%	-0.093%	-0.01%	-0.01%
L2	+18.57%	-0.01%	0.9 IPC	0.4 IPC
L3	-17%	-16.7%	0.75 IPC	0.1 IPC

*IPC – Instructions per Cycle

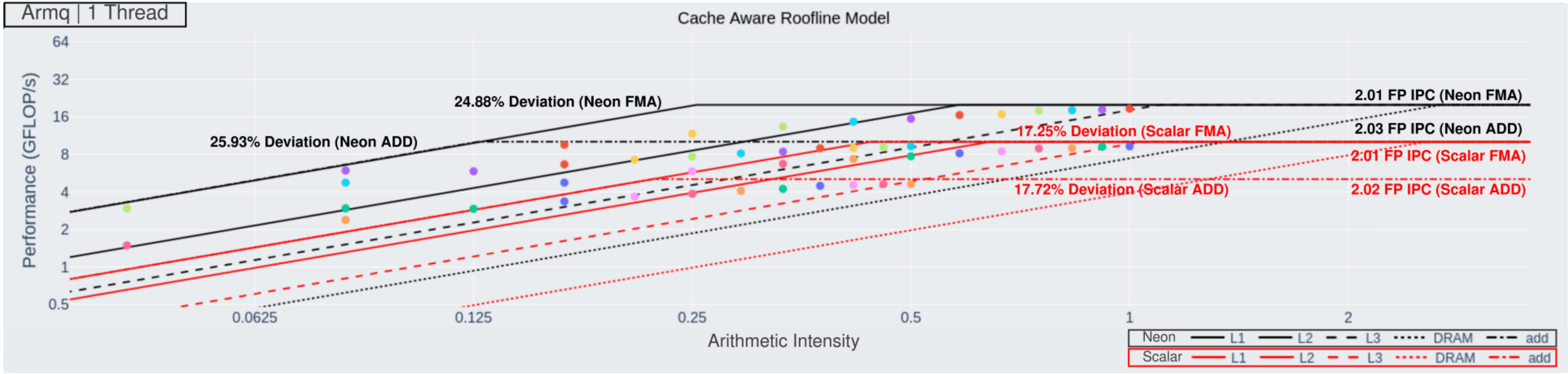
Results - CARM FP Arithmetic Analysis

Architectural Limits

- Accurately achieved
- Intel slows down with wider ISA extensions

Mixed Benchmark Validation

- FMA leads to more deviation
- Wider ISA extensions lead to more deviation



Results - CARM FP Arithmetic Analysis

- Architectual Limits
  - Accurately achieved
  - Intel slows down with wider ISA extensions
- Mixed Benchmark Validation
  - FMA leads to more deviation
  - Wider ISA extensions lead to more deviation

	Intel	AMD	ARM	RISCV
Scalar	-0.78%	-0.15%	+0.54%	+0.66%
Widest ISA	-5.9%	-0.12%	+0.85%	+0.63%

Results - CARM FP Arithmetic Analysis

Architectural Limits

- Accurately achieved
- Intel slows down with wider ISA extensions

Mixed Benchmark Validation

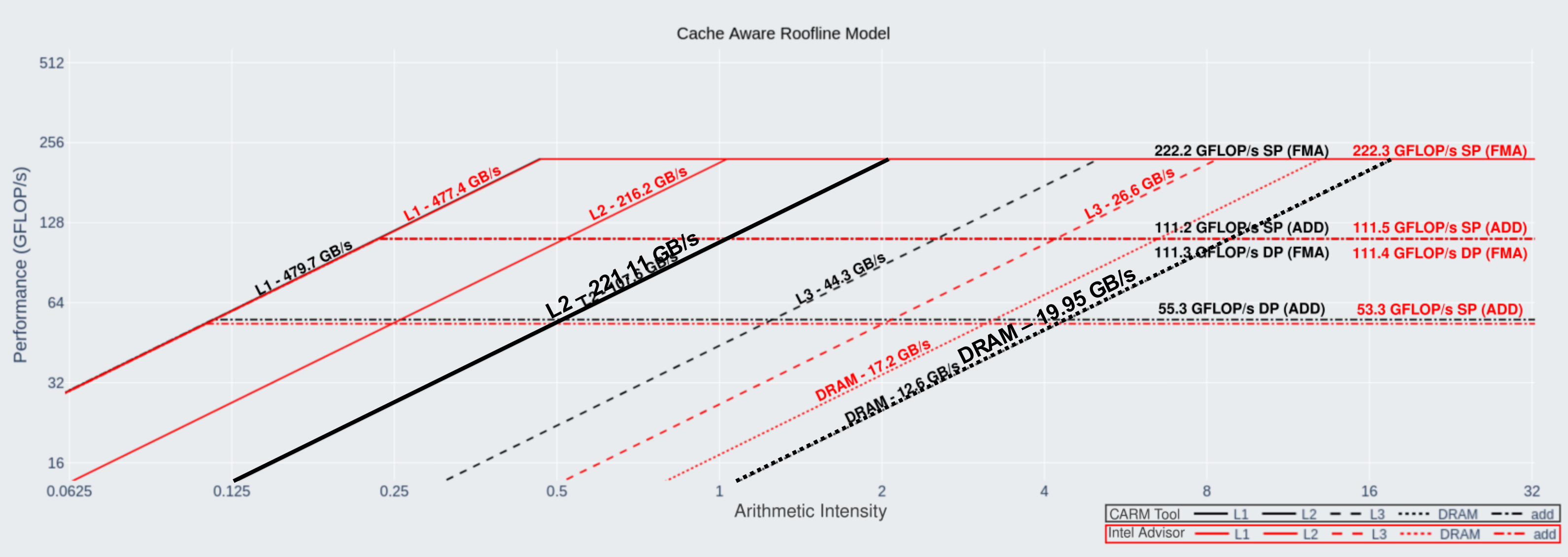
- FMA leads to more deviation
- Wider ISA extensions lead to more deviation

Deviation	Venus	Cara	Armq	Milk-V
Mixed Scalar Add	-3.44%	-1.11%	-17.72%	-4.55%
Mixed Scalar FMA	-5.34%	-10.27%	-17.25%	-8.29%
Mixed Widest ISA Add	-3.44%	-0.16%	-25.93%	-10.45%
Mixed Widest ISA FMA	-24.83%	-9.14%	-24.88%	-10.57%

Results – CARM Comparison

- Comparison with ERT
  - Higher L1 bandwidth and GFLOPS achieved
  - Lower level discrepancies due to imprecise cache size values

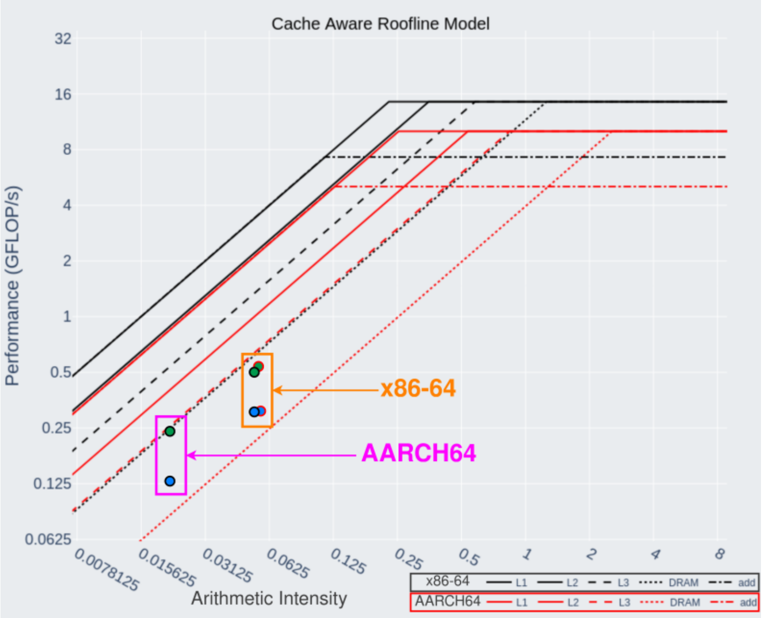
- Comparison with Intel Advisor
  - Similar L1 bandwidth and GFLOPS achieved
  - Lower level discrepancies due to ld/st ratio variations



Results – Application Analysis

- Cross-Architecture SpMV Analysis
  - Using the Eigen library
  - SpMV performance comparison

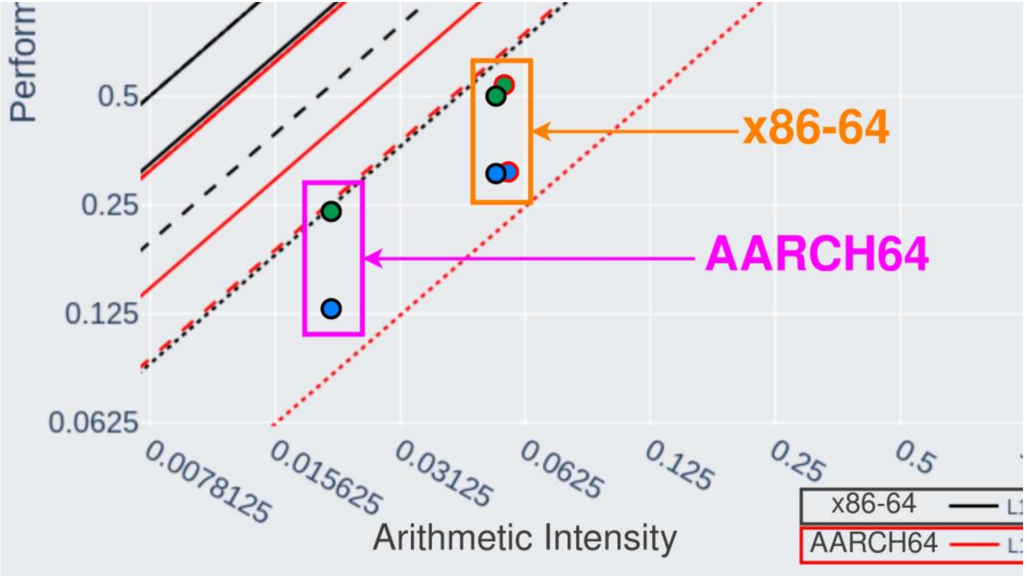
- SpMV Performance
  - RCM Re-Ordering improves performance



Results – Application Analysis

- Cross-Architecture SpMV Analysis
  - Using the Eigen library
  - SpMV performance comparison

- SpMV Performance
  - RCM Re-Ordering improves performance

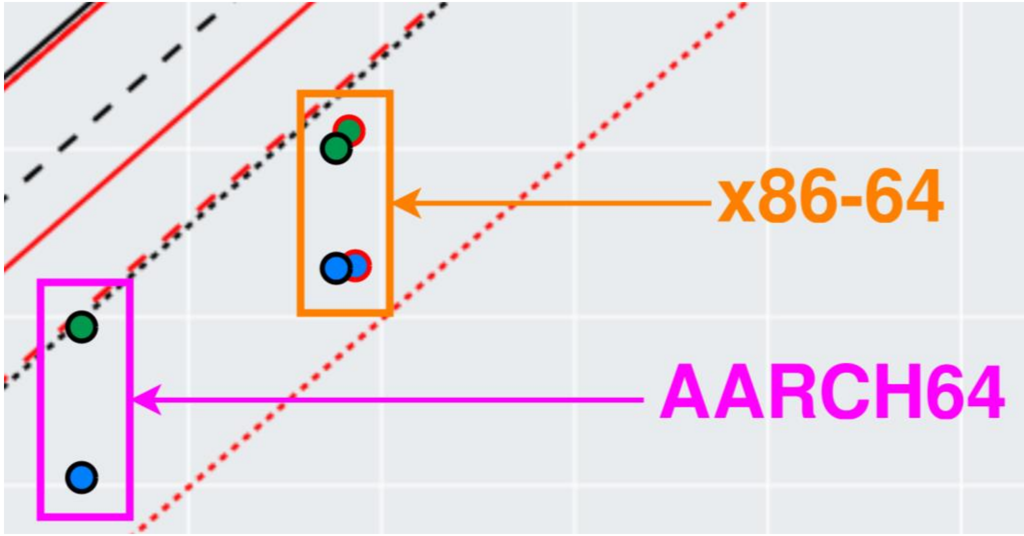


● DBI - Original Matrix ● DBI - RCM Matrix

Results – Application Analysis

- Cross-Architecture SpMV Analysis
  - Using the Eigen library
  - SpMV performance comparison

- SpMV Performance
  - RCM Re-Ordering improves performance



● DBI - Original Matrix ● DBI - RCM Matrix ● PMU - Original Matrix ● PMU - RCM Matrix

CONTENTS

- State of the Art
- The CARM Tool
- Results
- **Conclusion**

Conclusion

- Current collaborations
  - CERN, SYCLOPS, POP3, BSC
- Complete collaborations
  - SparCity – SuperTwin Live-CARM
- CARM Tool / CHAMP Hub Github and Paper

Conclusion – Current Collaborations

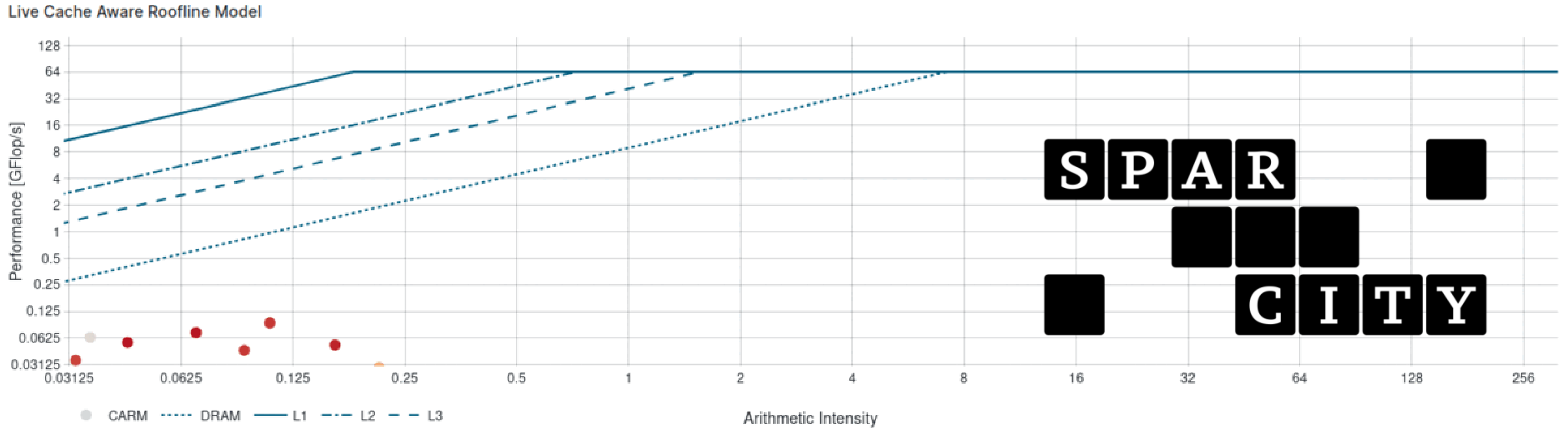
- CERN – Adaptive Perf Tool
- BSC – Paraver Tool
- SYCLOPS – SYCL DB
- POP3 – Application Profiling

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Conclusion – Complete Collaborations

- Sparcity – SuperTwin
  - CARM Tool source code shipped with the tool
  - SuperTwin interfaces with the CARM Tool to get CARM results

- Live-CARM
  - Based on live performance counter data from SuperTwin
  - Application and benchmark analysis was conducted

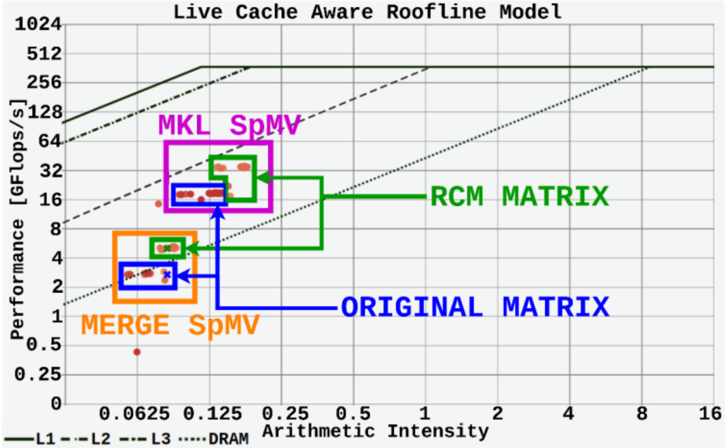

Live Cache Aware Roofline Model

TÉCNICO LISBOA

Conclusion – Complete Collaborations

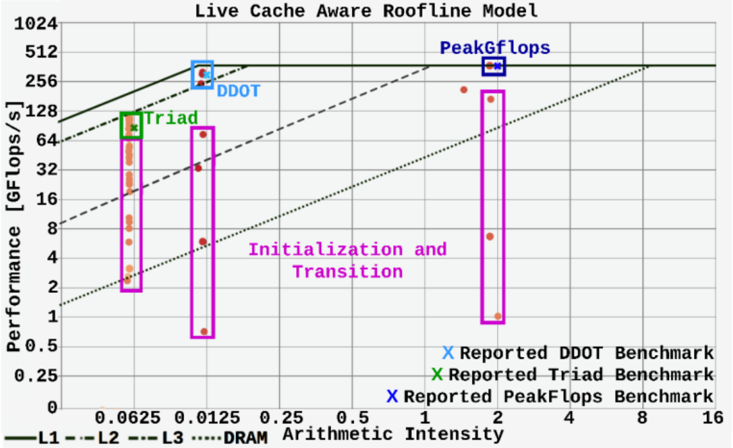
Sparcity – SuperTwin

- CARM Tool source code shipped with the tool
- SuperTwin interfaces with the CARM Tool to get CARM results



Live-CARM

- Based on live performance counter data from SuperTwin
- Application and benchmark analysis was conducted



CARM Tool – Available on Github

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

- The CARM Tool's paper is accepted for publication in IISWC24

- 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

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

TÉCNICO LISBOA

Thank You

Any questions?

José Morgado
Leonel Sousa
Aleksandar Ilic

28th POP Seminar
5 September 2024