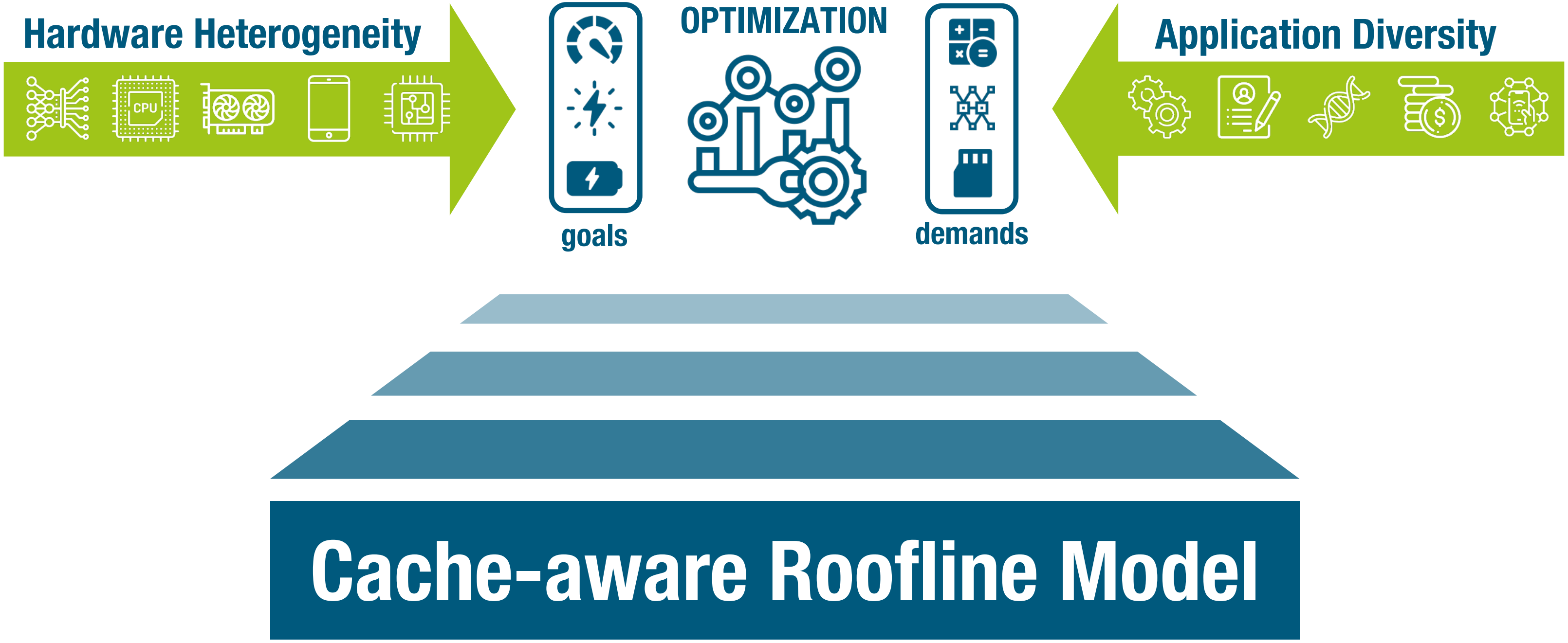
Cache-aware Roofline Model:
Performance, Power and Energy-efficiency

Aleksandar Ilic

Diogo Marques, Frederico Pratas, Leonel Sousa
Rafael Campos, Ricardo Nobre, Sergio Santander-Jiménez

inesc id lisboa 20 YEARS
DEFINING TECHNOLOGY


Hardware Heterogeneity
CPU
OPTIMIZATION
goals
demands
Application Diversity
Cache-aware Roofline Model

PERFORMANCE POWER ENERGY-EFFICIENCY CASE-STUDY

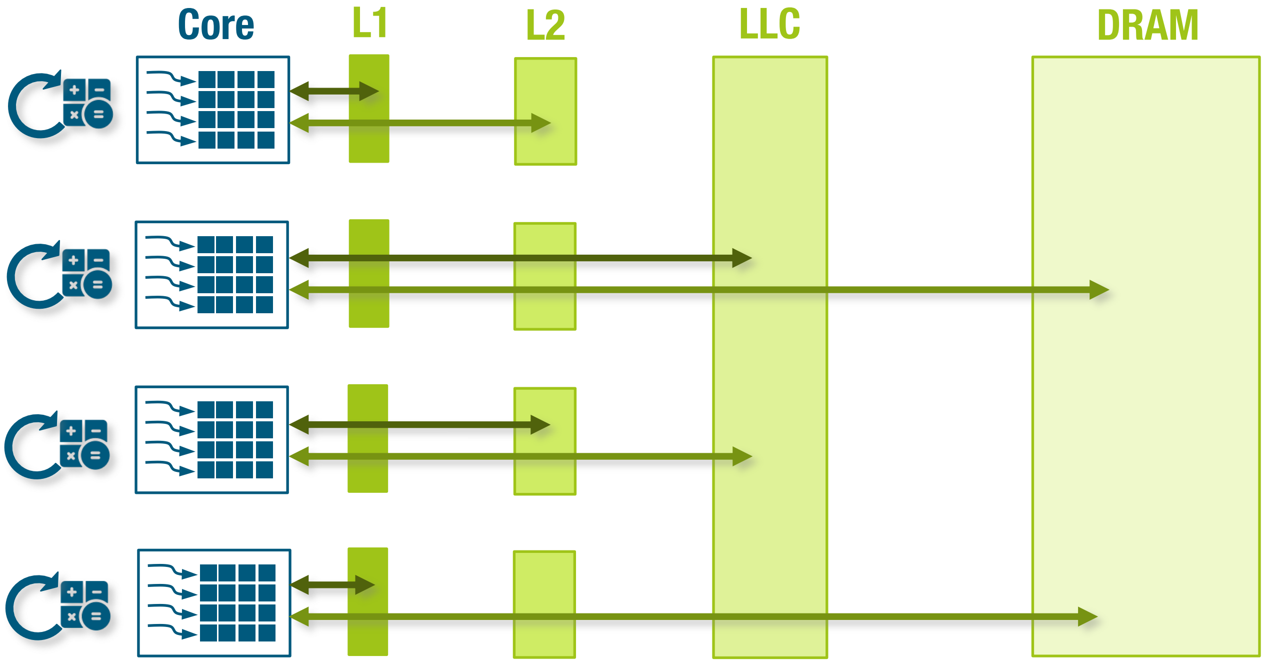
Cache-aware Roofline Model: Outline

PERFORMANCE POWER ENERGY-EFFICIENCY CASE-STUDY

Cache-aware Roofline Model

A. Ilic, F. Pratas and L. Sousa, "Cache-aware Roofline Model: Upgrading the Loft", IEEE Computer Architecture Letters (2014)
D. Marques, A. Ilic, Z. Matveev and L. Sousa, "Application-driven Cache-Aware Roofline Model", Elsevier FGCS (2020)

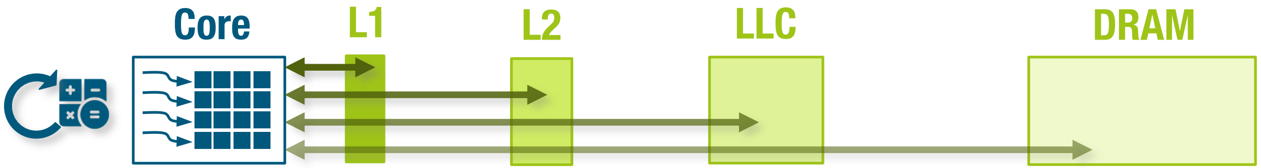
Roofline in a nutshell



Communication overlapped with computation

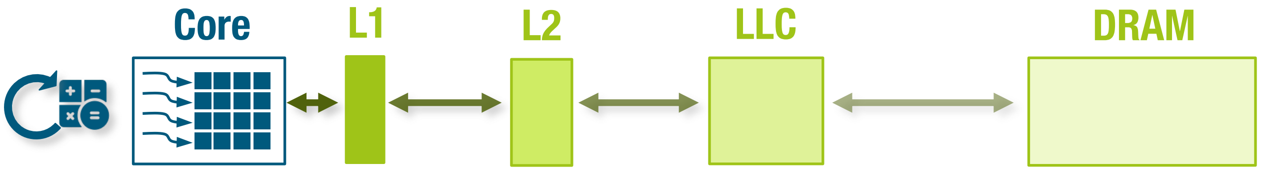
Max performance capped by peak compute throughput or available bandwidth (processor's view)

What is bandwidth?



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

- Obtained via micro-benchmarking



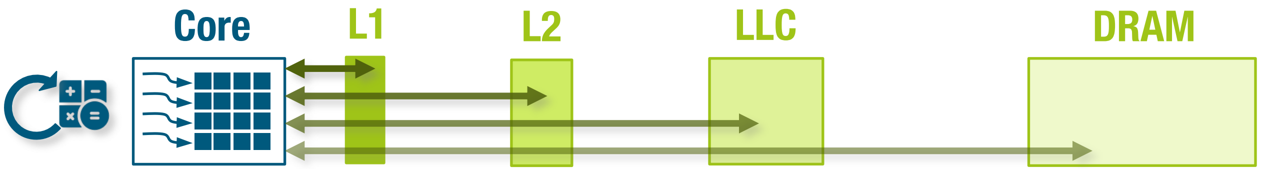
Original Roofline Model (ORM)[2]: Bandwidth between memory levels

- 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)

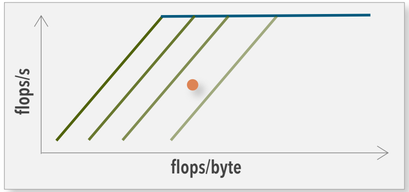
Implications …

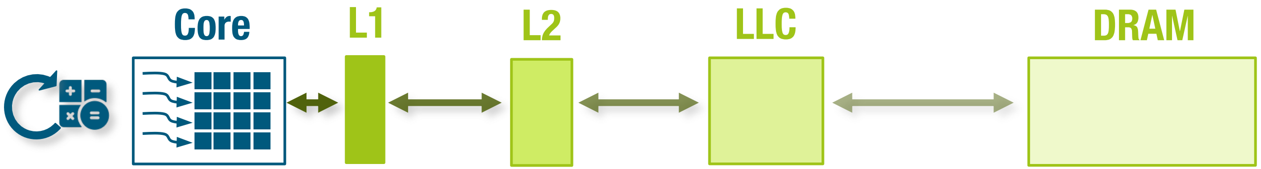


Core L1 L2 LLC DRAM

Cache-aware Roofline Model[1]

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

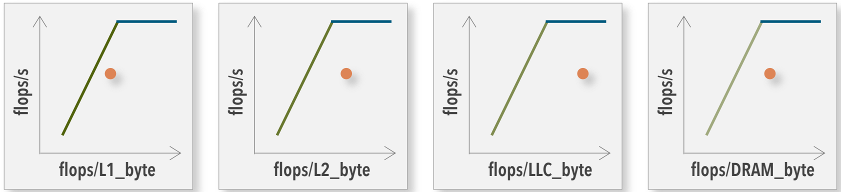




Core L1 L2 LLC DRAM

Original Roofline Model[2]

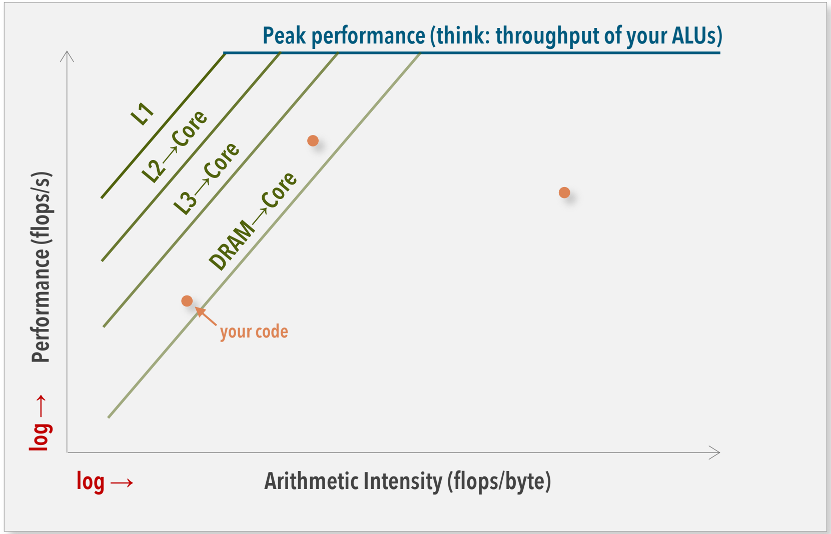
- Several models, several intensities
- Several application "points"



[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



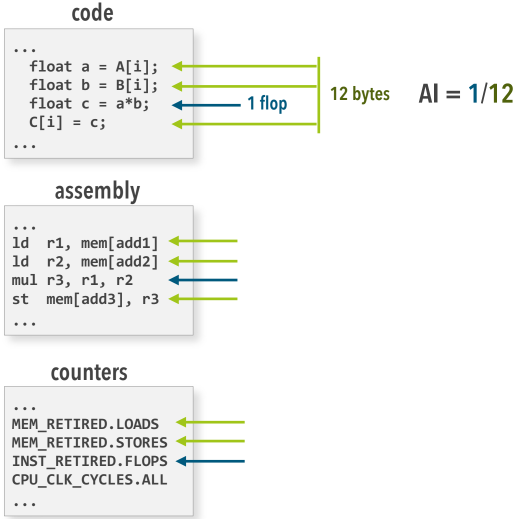
Cache-aware Roofline Model

- 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;
...
```

assembly

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

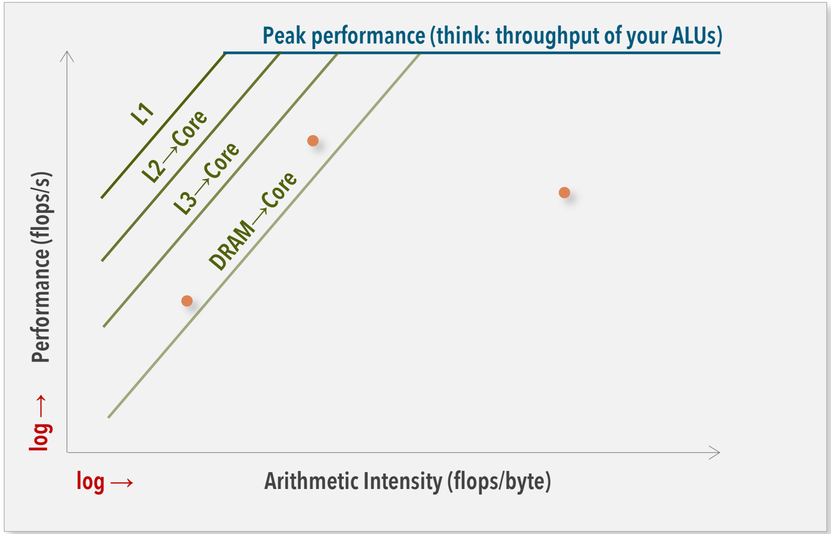
counters

```
...
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)

* We will relax this requirement in the next part of the talk

Implications … bring cool features



Peak performance (think: throughput of your ALUs)
L1
L2→Core
L3→Core
DRAM→Core
log →
Performance (flops/s)
log →
Arithmetic Intensity (flops/byte)

Cache-aware Roofline Model

- Shows absolute <u>architecture</u> maximums

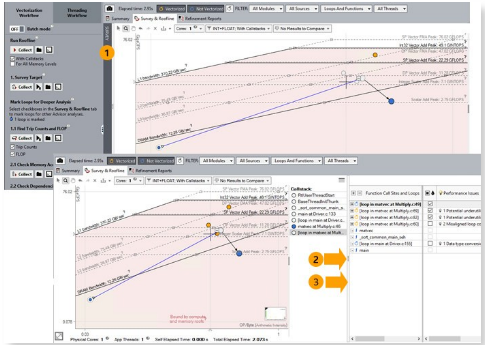
(You can't break them! Can your application exploit them?)

How to "plot" my code?

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

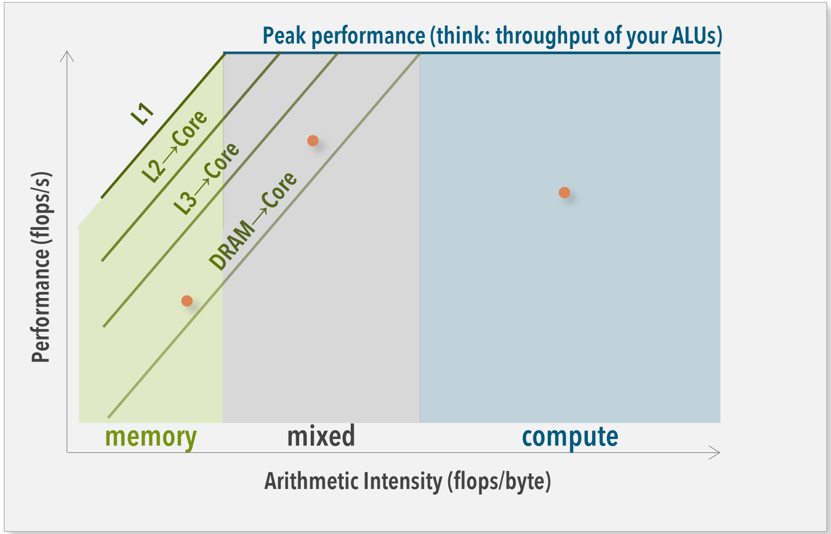
Intel Advisor Roofline feature

- CARM is there since 2017



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

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?

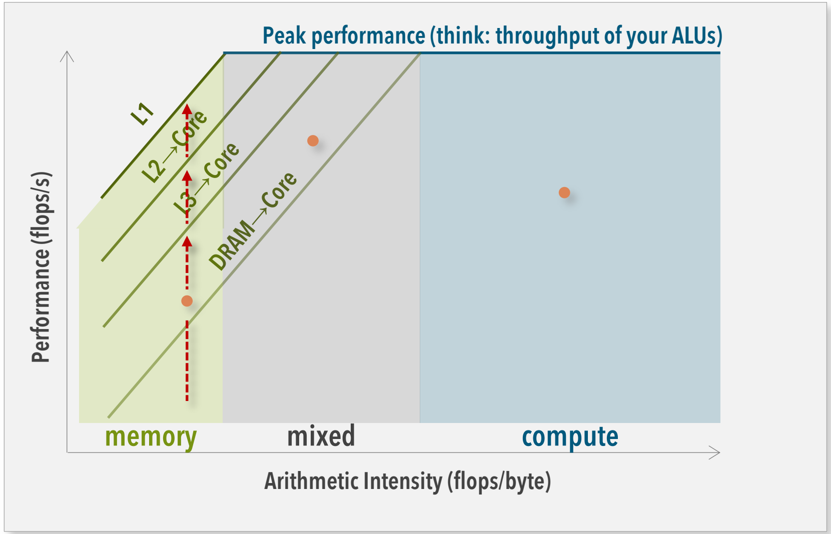
- CARM arithmetic intensity is exactly what you expect it to be!

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)

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?

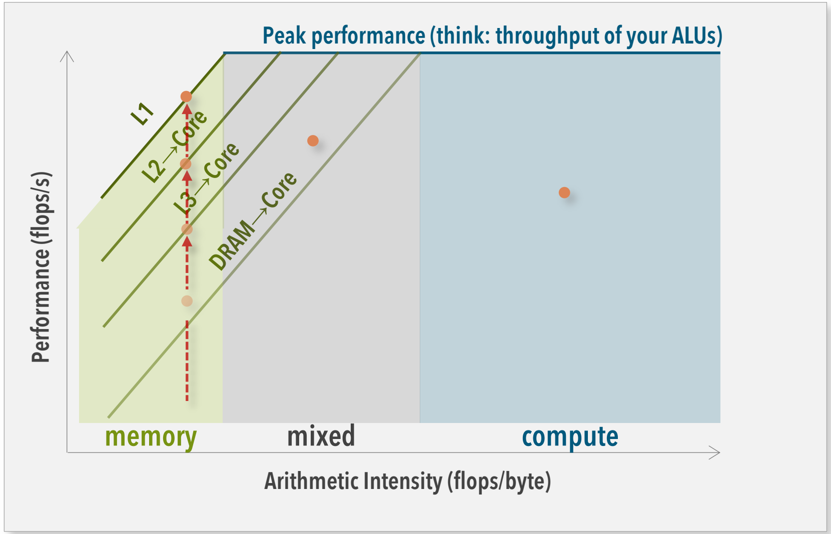
① Detect the boundness region
- What are my expected maximums?
- Provides first optimization hints

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)

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

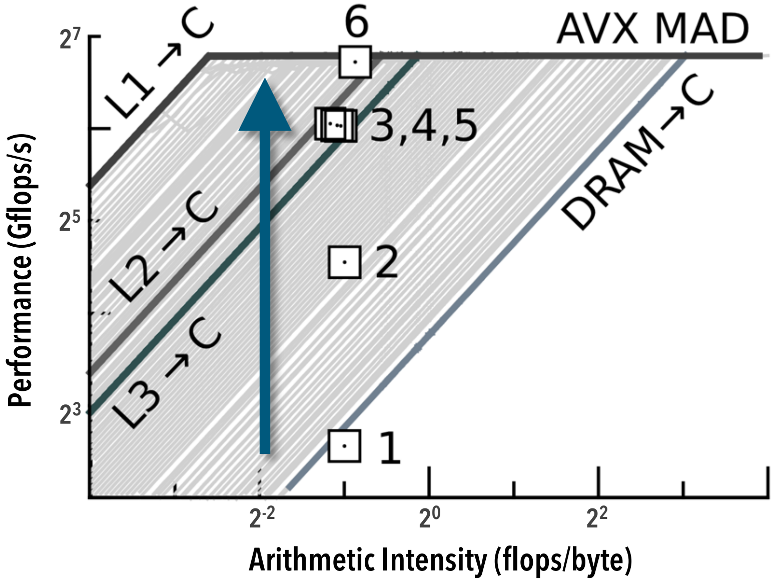
- What are my main bottlenecks? (observe intersected lines)
- Focus your optimization (aim at surpassing the line above)

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…

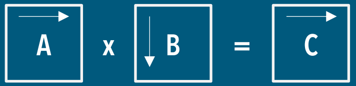
A. Ilic, F. Pratas and L. Sousa, "Cache-aware Roofline Model: Upgrading the Loft", IEEE Computer Architecture Letters (2014)
D. Marques, et.al., "Performance analysis with Cache-aware Roofline Model in Intel Advisor", HPCS (2017)

Matrix Multiplication

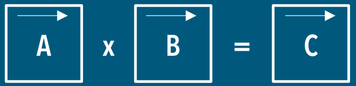


All codes AVX vectorized!*

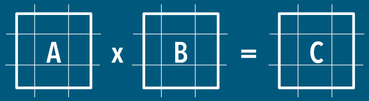
[1] Basic implementation (row major)



[2] Transposed B (improved mem. access)



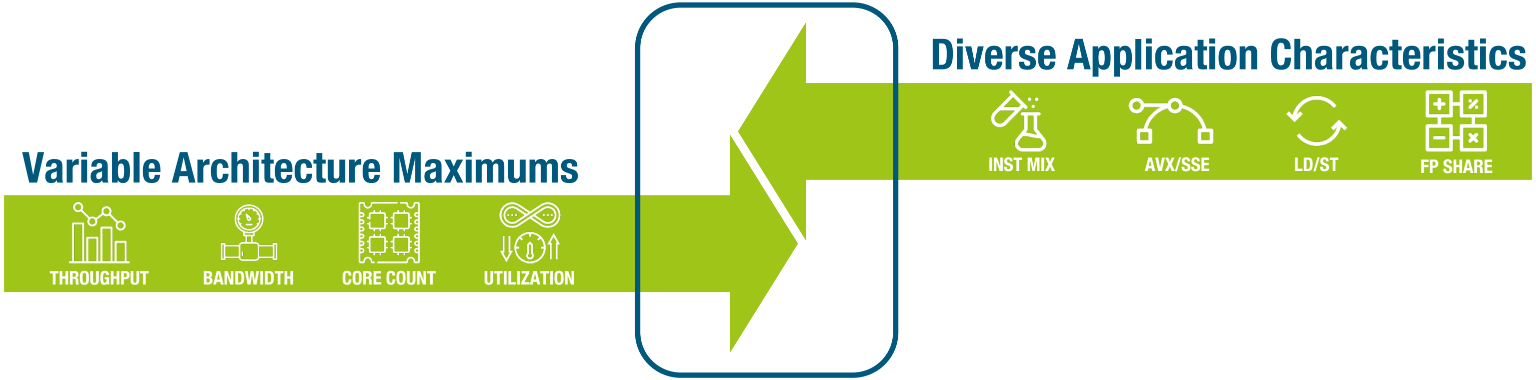
[3,4,5] Cache blocking: L3, L2, L1



[6] Intel MKL

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

* A. Ilic, F. Pratas and L. Sousa, “Beyond the Roofline: Cache-Aware Power and Energy-Efficiency Modeling for Multi-Cores”, IEEE Trans. on Computers (2017)



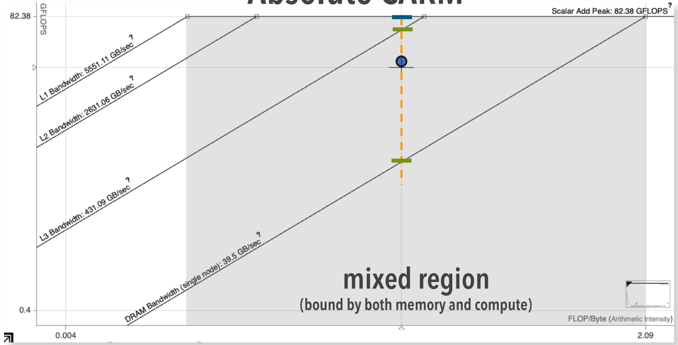
Application-driven CARM

(scaling rooflines to meet application demands)

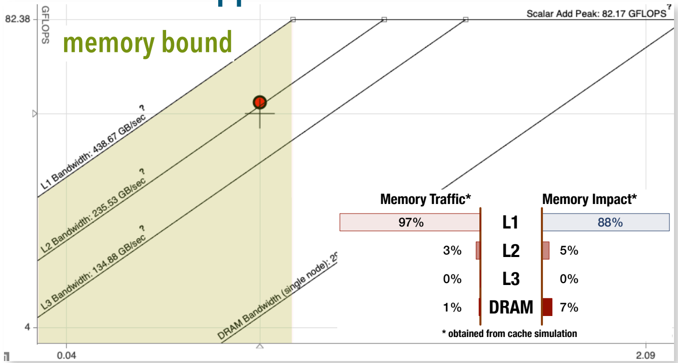
D. Marques, A. Ilic, Z. Matveev and L. Sousa, “Application-driven Cache-Aware Roofline Model”, Elsevier FGCS (2020)

ISO-3DFD: Quite optimized 3D stencil (scalar)

Absolute CARM



Application-driven CARM



CARM characterization cheat-sheet

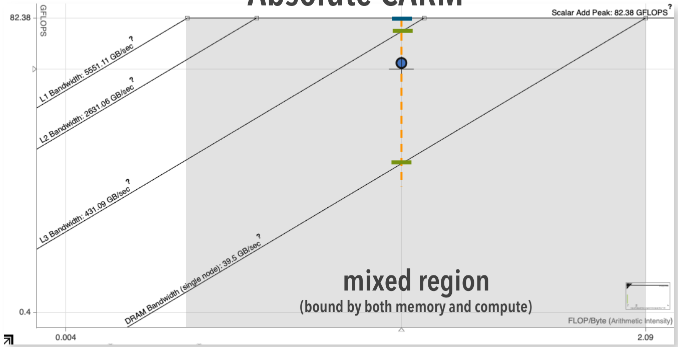
	Absolute	Application-driven
region	mixed	memory
max. perf.	compute (add)	memory (L1)
bottleneck	memory/compute	memory
optimize	everything	memory (or nothing)

Application-driven CARM

- models architecture maximums exploitable by your application
- improves characterization and hints (bottlenecks, optimization)
- provides consistent characterization during optimization process

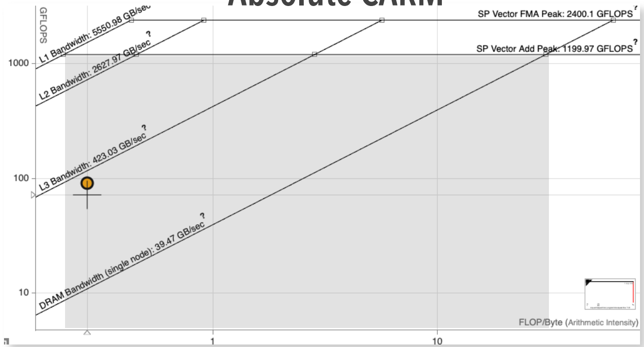
ISO-3DFD: Scalar (left) vs. AVX512 (right)

Absolute CARM

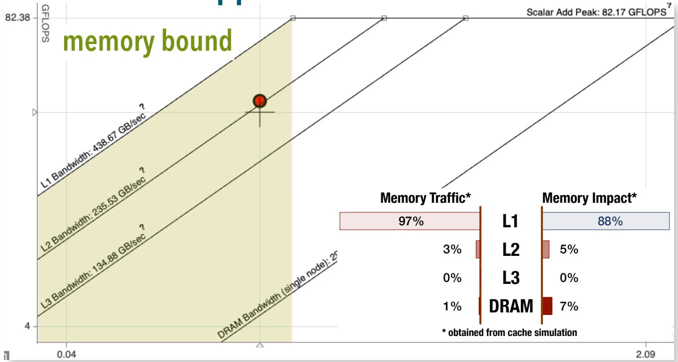


mixed region
(bound by both memory and compute)

Absolute CARM



Application-driven CARM

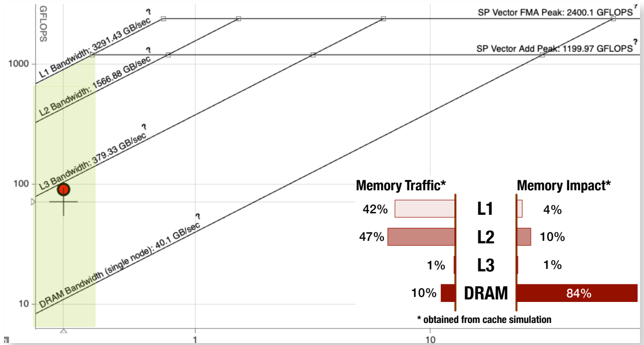


memory bound

Memory Traffic*		Memory Impact*
97%	L1	88%
3%	L2	5%
0%	L3	0%
1%	DRAM	7%

* obtained from cache simulation

Application-driven CARM



Memory Traffic*		Memory Impact*
42%	L1	4%
47%	L2	10%
1%	L3	1%
10%	DRAM	84%

* obtained from cache simulation

PERFORMANCE POWER ENERGY-EFFICIENCY CASE-STUDY

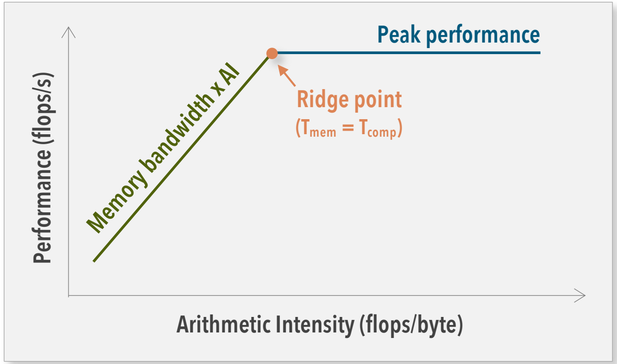
Cache-aware Roofline Model

A. Ilic, F. Pratas and L. Sousa, “Beyond the Roofline: Cache-Aware Power and Energy-Efficiency Modeling for Multi-Cores”, IEEE Trans. on Computers (2017)

CARM: Power Consumption

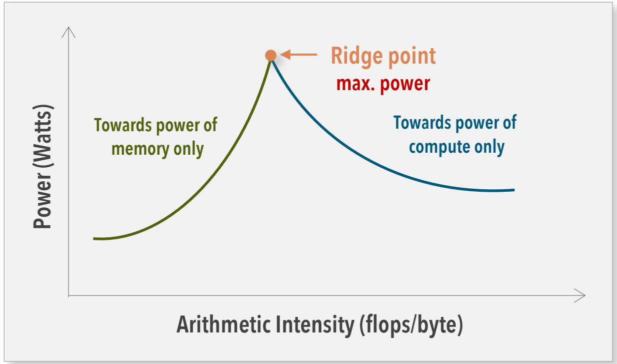
Performance CARM

- Contributions of comps and mops overlapped (in time)



Power CARM

- Contributions of comps and mops superposed (average power)


Ridge point
max. power
Towards power of memory only
Towards power of compute only
Power (Watts)
Arithmetic Intensity (flops/byte)

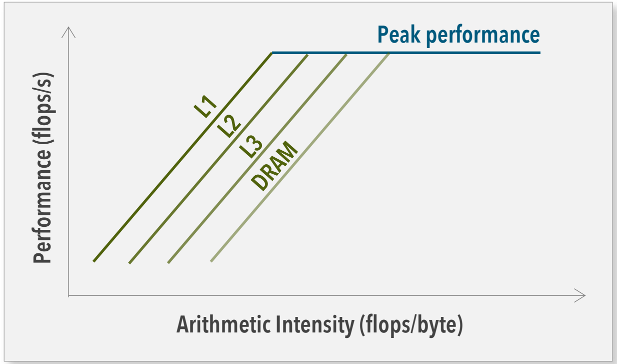
comps = compute operations
mops = memory operations

A. Ilic, F. Pratas and L. Sousa, "Beyond the Roofline: Cache-Aware Power and Energy-Efficiency Modeling for Multi-Cores", IEEE Trans. on Computers (2017)

CARM: Power Consumption

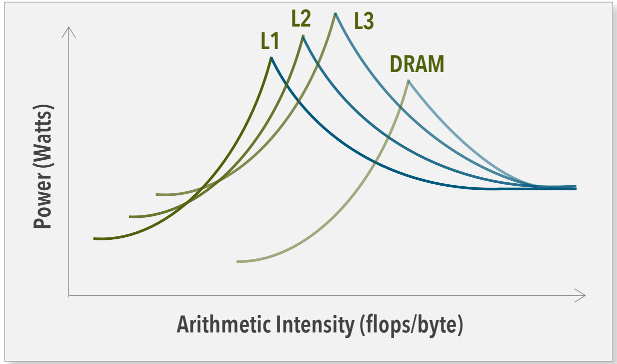
Performance CARM

- Contributions of comps and mops overlapped (in time)



Power CARM: Cores

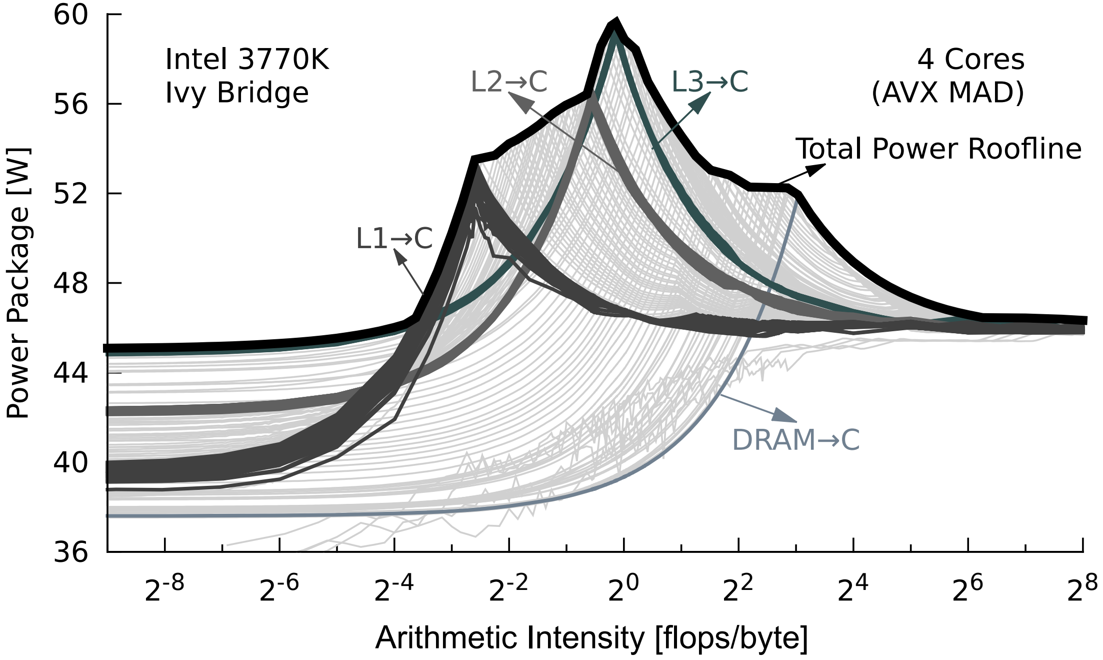
- Contributions of comps and mops superposed (average power)



comps = compute operations
mops = memory operations

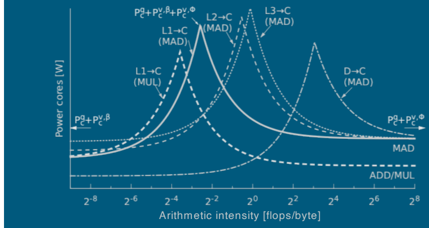
A. Ilic, F. Pratas and L. Sousa, “Beyond the Roofline: Cache-Aware Power and Energy-Efficiency Modeling for Multi-Cores”, IEEE Trans. on Computers (2017)

Total Power CARM: Defining envelope

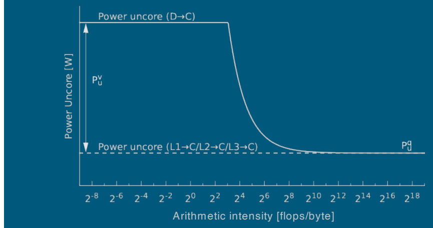


CARM for different RAPL domains

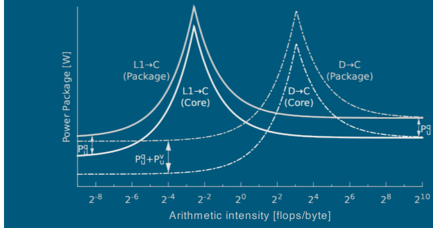
Cores



Uncore

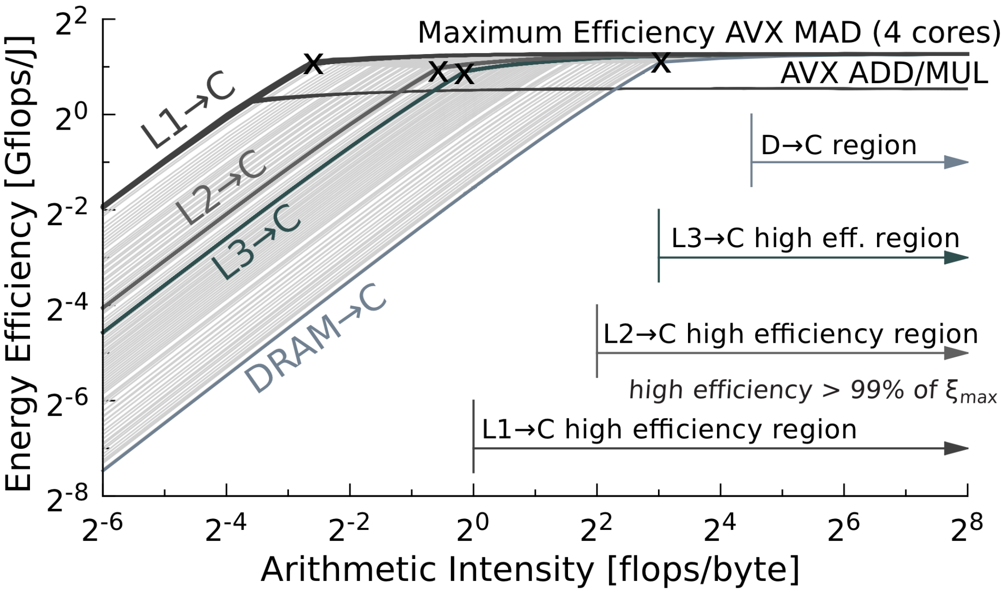


Package



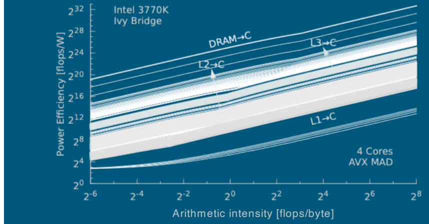
A. Ilic, F. Pratas and L. Sousa, “Beyond the Roofline: Cache-Aware Power and Energy-Efficiency Modeling for Multi-Cores”, IEEE Trans. on Computers (2017)

Energy-efficiency CARM

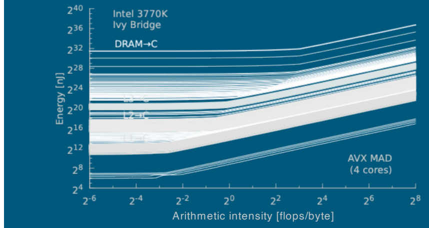


Maximum efficiency for infinite arithmetic intensity!

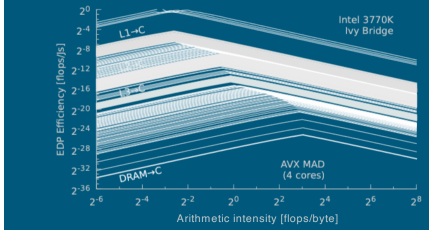
Power-efficiency CARM [flops/W]



Energy CARM [Joule]

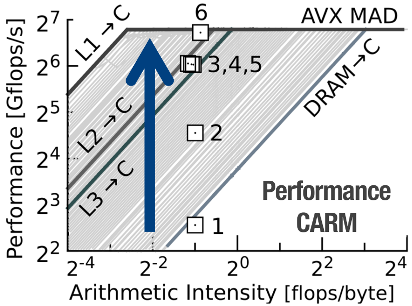


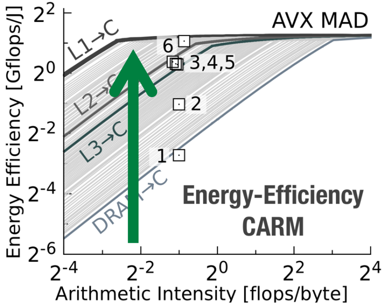
EDP-Efficiency CARM [flops/Js]

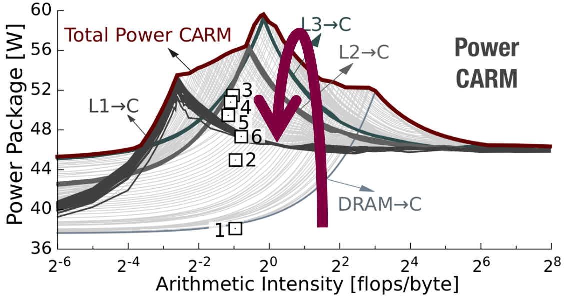


A. Ilic, F. Pratas and L. Sousa, "Beyond the Roofline: Cache-Aware Power and Energy-Efficiency Modeling for Multi-Cores", IEEE Trans. on Computers (2017)

Matrix Multiplication

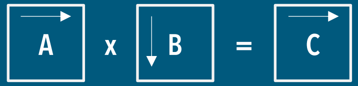




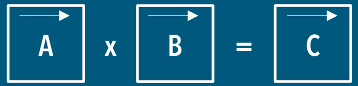


All codes AVX vectorized!*

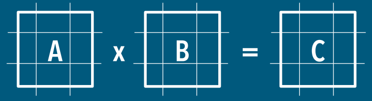
[1] Basic implementation (row major)



[2] Transposed B (improved mem. access)



[3,4,5] Cache blocking: L3, L2, L1



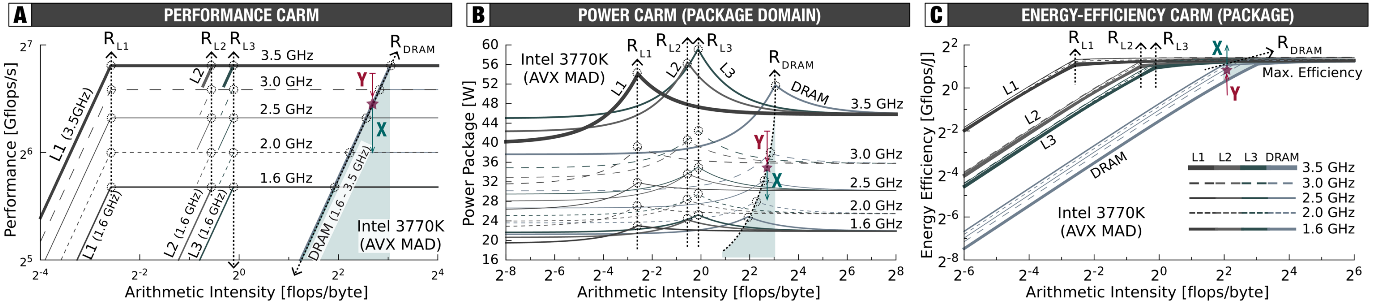
[6] Intel MKL

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

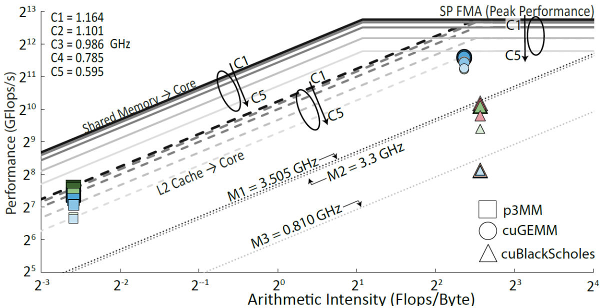
* A. Ilic, F. Pratas and L. Sousa, "Beyond the Roofline: Cache-Aware Power and Energy-Efficiency Modeling for Multi-Cores", IEEE Trans. on Computers (2017)

Cache-aware Roofline Model: Extensions

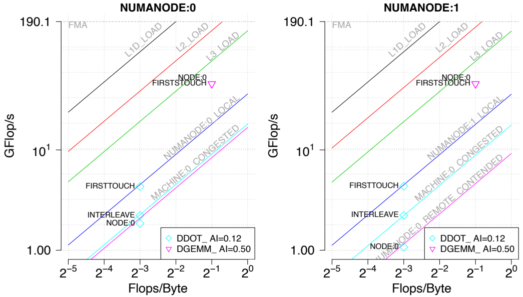
CARM-based DVFS analysis



GPU CARM: Performance, Power, DVFS



NUMA CARM: Multi-socket, KNL



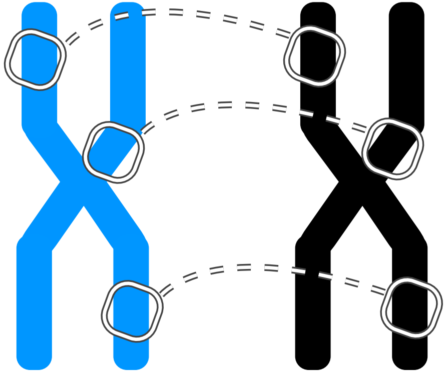
A. Ilic, F. Pratas, L. Sousa, “Beyond the Roofline: Cache-Aware Power and Energy-Efficiency Modeling for Multi-Cores”, IEEE Trans. on Computers (2017)
A. Lopes, F. Pratas, L. Sousa, A. Ilic, “Exploring GPU performance, power and energy-efficiency bounds with Cache-aware Roofline Modeling”, ISPASS (2017)
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)

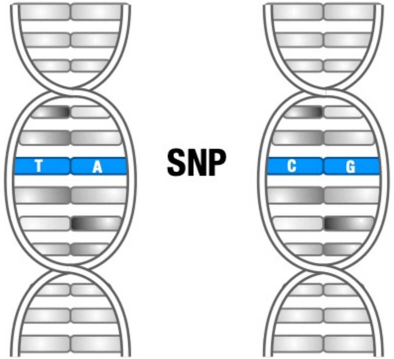
PERFORMANCE POWER ENERGY-EFFICIENCY CASE-STUDY

Epistasis Detection: CARM-driven Optimization

R. Nobre, A. Ilic, S. Santander-Jiménez, L. Sousa, "Exploring the Binary Precision Capabilities of Tensor Cores for Epistasis Detection", IPDPS (2020)
R. Campos, D. Marques, S. Santander-Jiménez, L. Sousa, A. Ilic, "Heterogeneous CPU+ iGPU Processing for Efficient Epistasis Detection", EuroPar (2020)

Epistasis in a nutshell



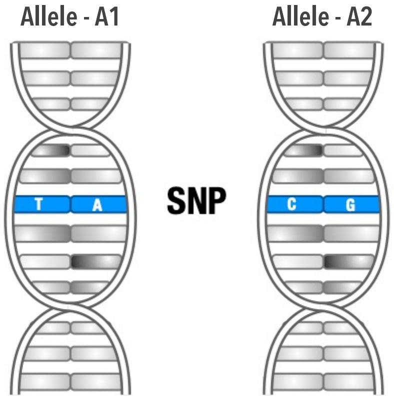


Some SNP interactions may cause life-threating diseases (e.g., Alzheimer, breast cancer)
Discovering which and how many is important, but challenging task!

Short Bio Recap: Codifying your genotype

Genotype	A1	A2	
0	◆	◆	Homozygous Major
1	◆	◇	Heterozygous
2	◇	◇	Homozygous Minor

◆ dominant allele
◇ recessive allele

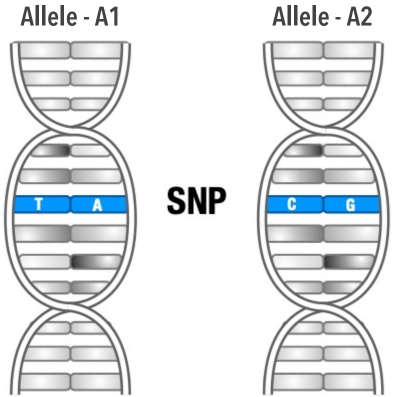


Binarizing your genotype

SNP X	P0	Genotype	A1	A2	
X0	0	0	◆	◆	Homozygous Major
X1	1	1	◆	◇	Heterozygous
X2	0	2	◇	◇	Homozygous Minor
phenotype	0				

◆ dominant allele
◇ recessive allele

Think: Patient 0 (**P0**) with genotype 1 does not have disease (control)

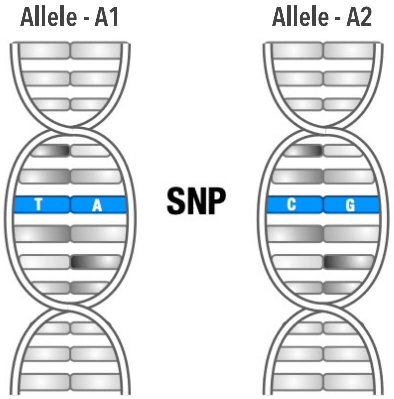


Binarizing your genotype

SNP X	P0	P1	Genotype	A1	A2	
X0	0	0	0	◆	◆	Homozygous Major
X1	1	0	1	◆	◇	Heterozygous
X2	0	1	2	◇	◇	Homozygous Minor
phenotype	0	1				

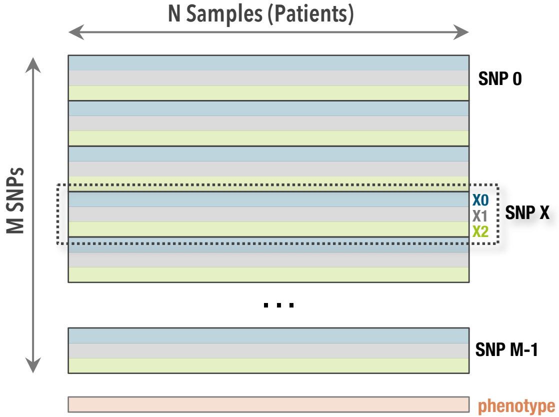
◆ dominant allele
◇ recessive allele

Think: Patient 1 (**P1**) with genotype 2 has disease (case)



Dataset structure

SNP X	P0	P1	P2	P3	P4	P5	...	PN
X0	0	0	0	1	0	0	...	1
X1	1	0	1	0	0	1	...	0
X2	0	1	0	0	1	0	...	0
phenotype	0	1	1	1	0	0	...	1

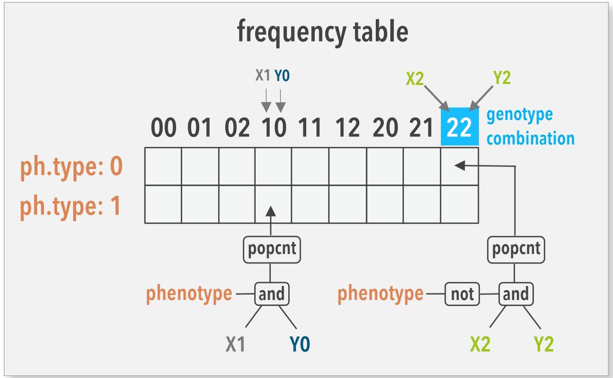


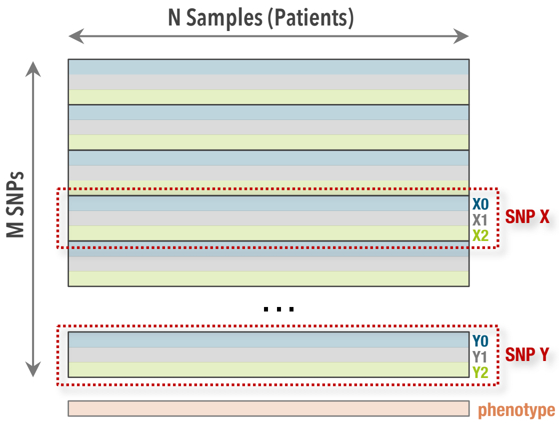
Dataset structure

Our dataset: 10 040 SNPs x 104 448 samples

2-way Epistasis Detection: Pair-wise interaction

Pair-wise interaction: SNPs (X,Y)





Dataset structure

Our dataset: 10 040 SNPs x 104 448 samples

Search space: All SNP combinations

(0,1)	(0,2)	(0,3)	(0,4)	(...)	(0,M-1)
	(1,2)	(1,3)	(1,4)	(...)	(1,M-1)
		(2,3)	(2,4)	(...)	(2,M-1)
			(3,4)	(...)	(3,M-1)
				(...)	
				(...)	(M-2,M-1)

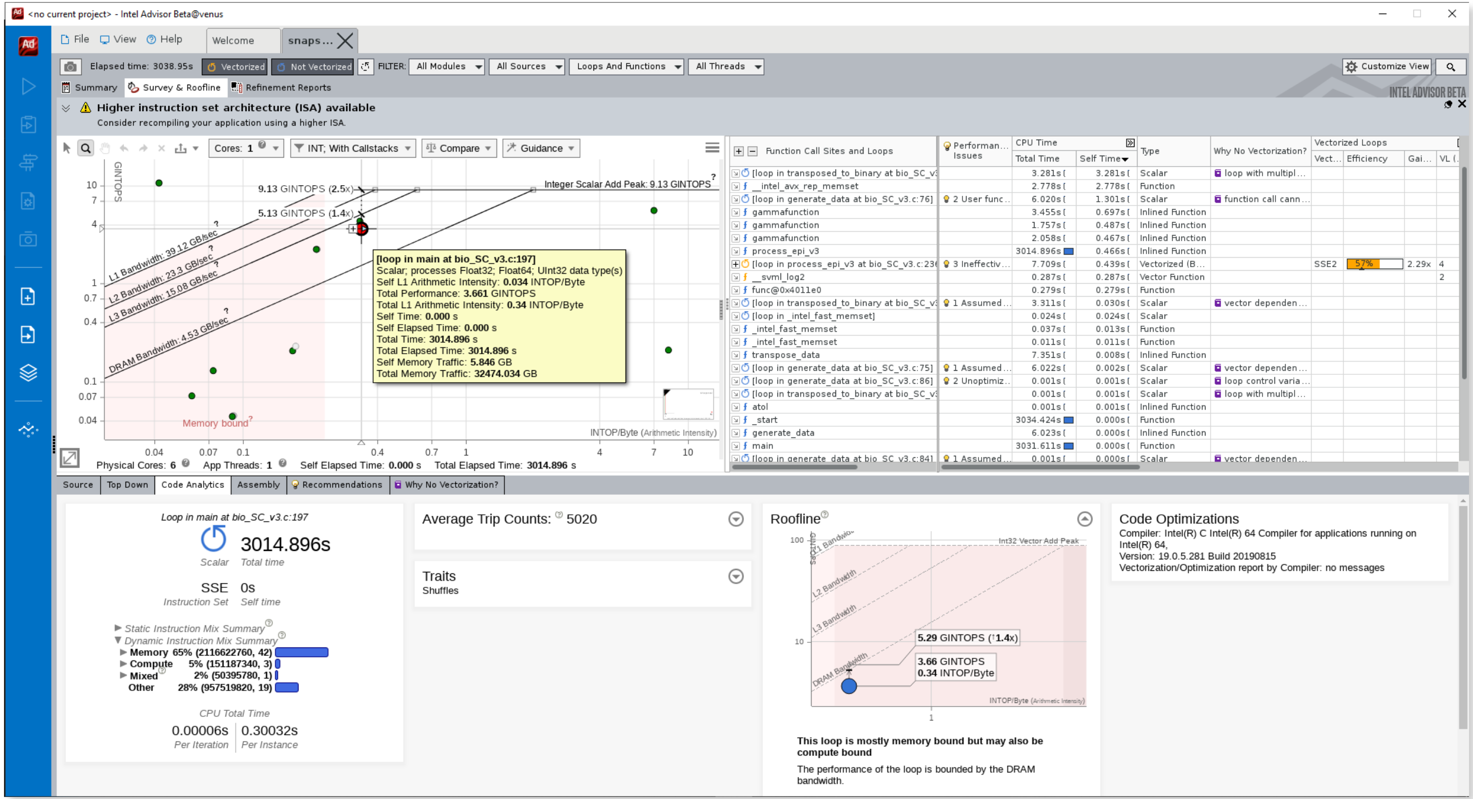
M(M-1)/2 combinations

Our dataset: 50 395 780 combinations

Each frequency table evaluated with Bayesian K2 score
Epistasis: Minimum K2 score among all combinations!

R. Nobre, A. Ilic, S. Santander-Jiménez, L. Sousa, “Exploring the Binary Precision Capabilities of Tensor Cores for Epistasis Detection”, IPDPS (2020)
R. Campos, D. Marques, S. Santander-Jiménez, L. Sousa, A. Ilic, “Heterogeneous CPU+ iGPU Processing for Efficient Epistasis Detection”, EuroPar (2020)

Cache-aware Roofline Model in Intel® Advisor



Higher instruction set architecture (ISA) available

Consider recompiling your application using a higher ISA.

[loop in main at bio_SC_v3.c:197]
Scalar; processes Float32; Float64; UInt32 data type(s)
Self L1 Arithmetic Intensity: **0.034** INTOP/Byte
Total Performance: **3.661** GINTOPS
Total L1 Arithmetic Intensity: **0.34** INTOP/Byte
Self Time: **0.000** s
Self Elapsed Time: **0.000** s
Total Time: **3014.896** s
Total Elapsed Time: **3014.896** s
Self Memory Traffic: **5.846** GB
Total Memory Traffic: **32474.034** GB

Physical Cores: **6** App Threads: **1** Self Elapsed Time: **0.000** s Total Elapsed Time: **3014.896** s

Function Call Sites and Loops	Performan... Issues	CPU Time		Type	Why No Vectorization?	Vectorized Loops			
		Total Time	Self Time			Vect...	Efficiency	Gai...	VL (...
[loop in transposed_to_binary at bio_SC_v3		3.281s	3.281s	Scalar	loop with multipl...				
__intel_avx_rep_memset		2.778s	2.778s	Function					
[loop in generate_data at bio_SC_v3.c:76]	2 User func...	6.020s	1.301s	Scalar	function call cann...				
gammafunction		3.455s	0.697s	Inlined Function					
gammafunction		1.757s	0.487s	Inlined Function					
gammafunction		2.058s	0.467s	Inlined Function					
process_epi_v3		3014.896s	0.466s	Inlined Function					
[loop in process_epi_v3 at bio_SC_v3.c:23	3 Ineffectiv...	7.709s	0.439s	Vectorized (B...		SSE2	57%	2.29x	4
__svml_log2		0.287s	0.287s	Vector Function					2
func@0x4011e0		0.279s	0.279s	Function					
[loop in transposed_to_binary at bio_SC_v3	1 Assumed...	3.311s	0.030s	Scalar	vector dependen...				
[loop in _intel_fast_memset]		0.024s	0.024s	Scalar					
_intel_fast_memset		0.037s	0.013s	Function					
_intel_fast_memset		0.011s	0.011s	Function					
transpose_data		7.351s	0.008s	Inlined Function					
[loop in generate_data at bio_SC_v3.c:75]	1 Assumed...	6.022s	0.002s	Scalar	vector dependen...				
[loop in generate_data at bio_SC_v3.c:86]	2 Unoptimiz...	0.001s	0.001s	Scalar	loop control varia...				
[loop in transposed_to_binary at bio_SC_v3		0.001s	0.001s	Scalar	loop with multipl...				
atol		0.001s	0.001s	Inlined Function					
_start		3034.424s	0.000s	Function					
generate_data		6.023s	0.000s	Inlined Function					
main		3031.611s	0.000s	Function					
[loop in generate_data at bio_SC_v3.c:84]	1 Assumed...	0.001s	0.000s	Scalar	vector dependen...				

Loop in main at bio_SC_v3.c:197

3014.896s
Scalar *Total time*

SSE 0s
Instruction Set *Self time*

▶ *Static Instruction Mix Summary*
▼ *Dynamic Instruction Mix Summary*

▶ **Memory**	**65% (2116622760, 42)**
▶ **Compute**	**5% (151187340, 3)**
▶ **Mixed**	**2% (50395780, 1)**
Other	**28% (957519820, 19)**

CPU Total Time

0.00006s *Per Iteration* | 0.30032s *Per Instance*

Average Trip Counts: 5020

Traits
Shuffles

Roofline

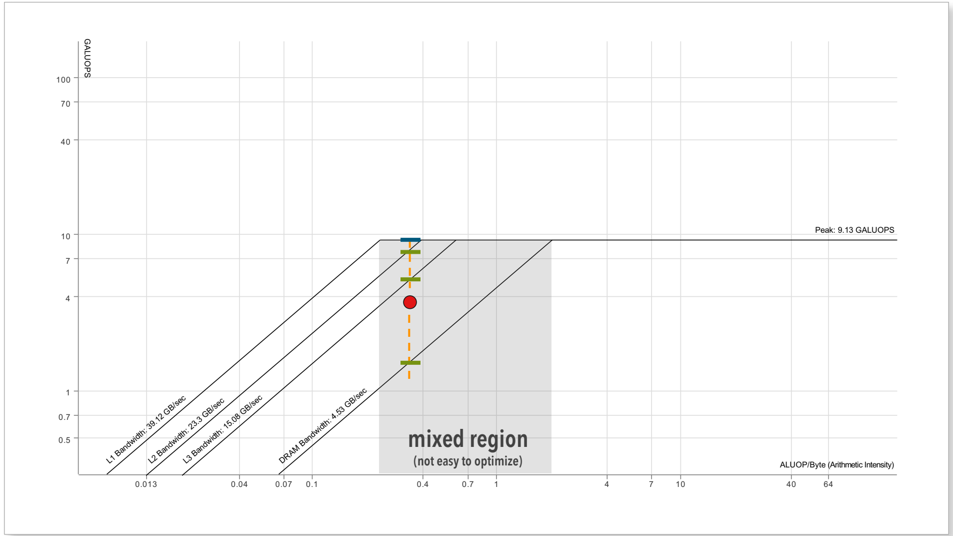
This loop is mostly memory bound but may also be compute bound

The performance of the loop is bounded by the DRAM bandwidth.

Code Optimizations
Compiler: Intel(R) C Intel(R) 64 Compiler for applications running on Intel(R) 64,
Version: 19.0.5.281 Build 20190815
Vectorization/Optimization report by Compiler: no messages

Let’s CARMify it!

Three Genotypes + Phenotype

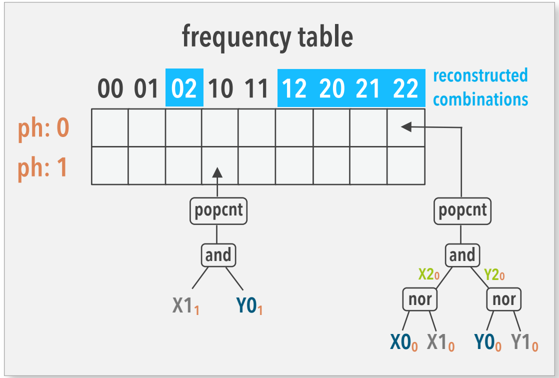


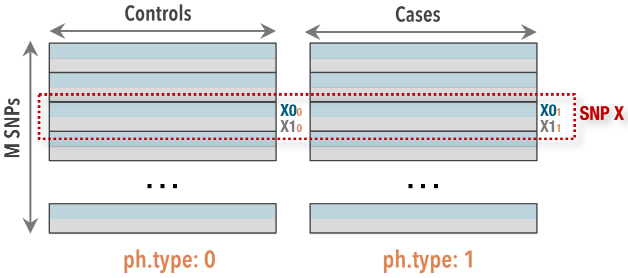
Mixed region, but seems memory bound
Let’s be smart: Restructure our algorithm!

Results obtained with a “special version“ of Intel® Advisor | Platform: Intel® i7-8700K (3.7GHz) with HT/Prefetching/TurboBoost disabled, single core

Increase arithmetic intensity

Pair-wise interaction: SNPs (X,Y)

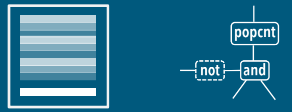




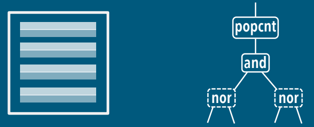
“New” Dataset structure
(removed: phenotype and genotype 2)

Reducing memory transfers!
Boosting our arithmetic intensity!

○ **Three Genotypes + Phenotype**

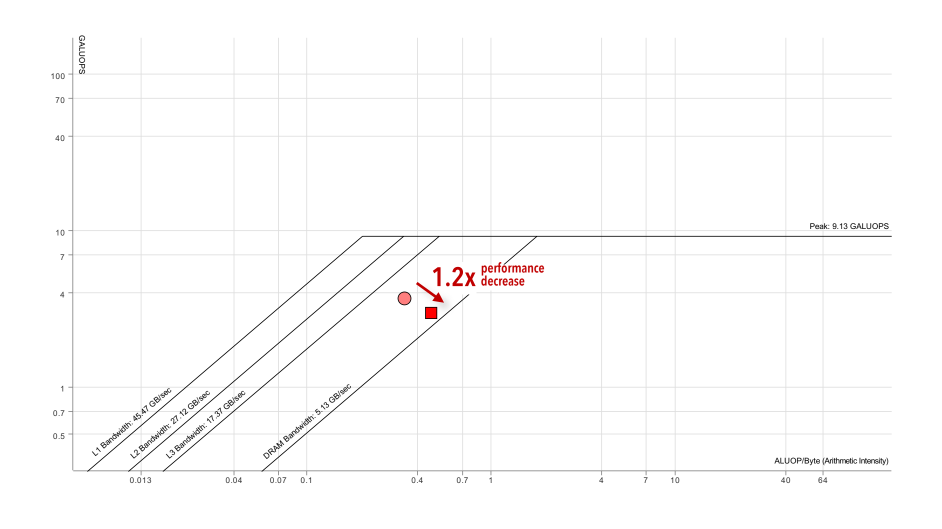


□ **Two Genotypes, No Phenotype**



R. Nobre, A. Ilic, S. Santander-Jiménez, L. Sousa, “Exploring the Binary Precision Capabilities of Tensor Cores for Epistasis Detection”, IPDPS (2020)

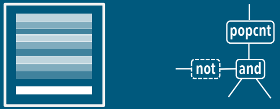
Let’s CARMify it (again)!



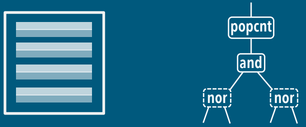
1.2x performance decrease

Wait! Being smart decreases performance!
How come?!

● Three Genotypes + Phenotype

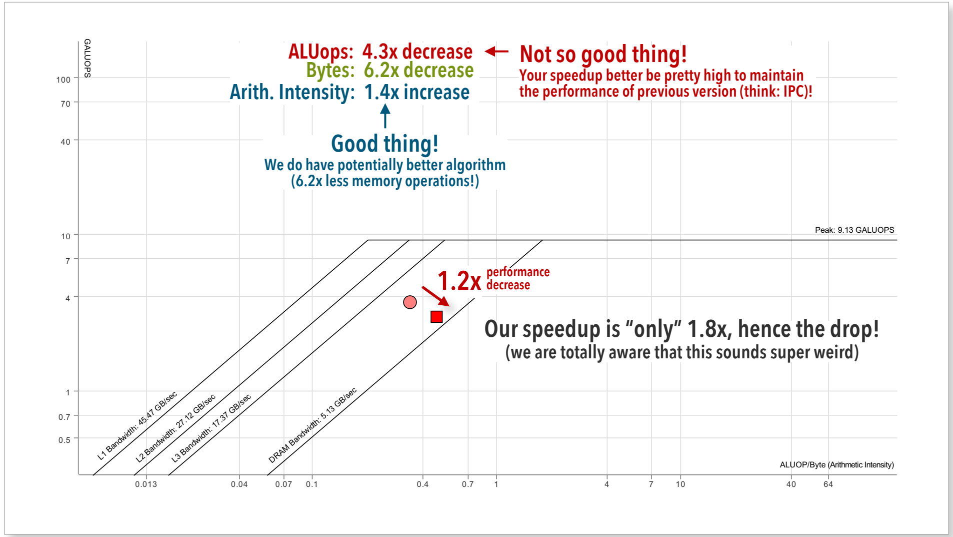


■ Two Genotypes, No Phenotype



Results obtained with a “special version“ of Intel® Advisor | Platform: Intel® i7-8700K (3.7GHz) with HT/Prefetching/TurboBoost disabled, single core

Let’s CARMify it (again)!



ALUops: 4.3x decrease
Bytes: 6.2x decrease
Arith. Intensity: 1.4x increase

Not so good thing!
Your speedup better be pretty high to maintain the performance of previous version (think: IPC)!

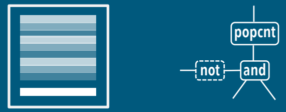
Good thing!
We do have potentially better algorithm (6.2x less memory operations!)

1.2x performance decrease

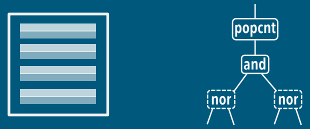
Our speedup is “only” 1.8x, hence the drop!
(we are totally aware that this sounds super weird)

Wait! Being smart decreases performance!
How come?!

● Three Genotypes + Phenotype

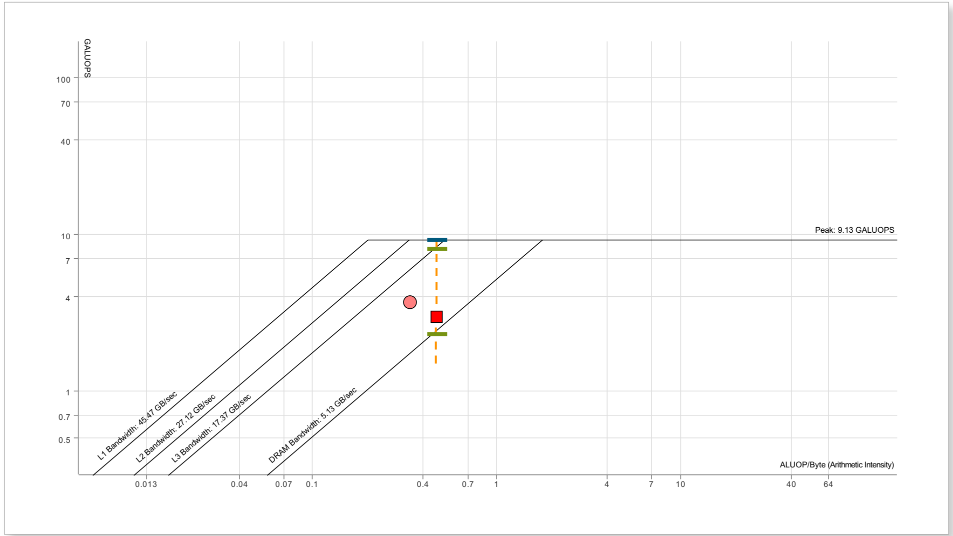


■ Two Genotypes, No Phenotype

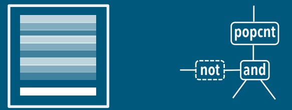


Results obtained with a “special version“ of Intel® Advisor | Platform: Intel® i7-8700K (3.7GHz) with HT/Prefetching/TurboBoost disabled, single core

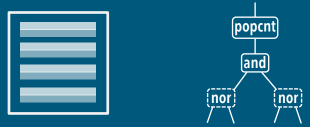
Let’s continue optimizing…



● Three Genotypes + Phenotype



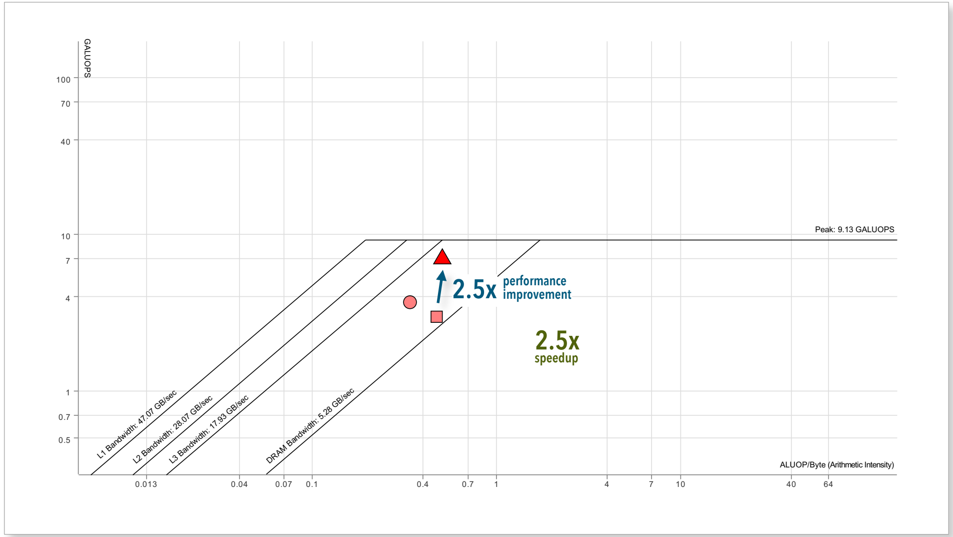
■ Two Genotypes, No Phenotype



CARM and perf. decrease may suggest memory issues!
Let’s “tile” our dataset for caches!

Results obtained with a “special version” of Intel® Advisor | Platform: Intel® i7-8700K (3.7GHz) with HT/Prefetching/TurboBoost disabled, single core

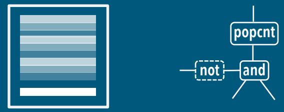
Improvements, at last!!!



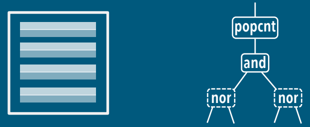
Tiling worked!

We now have both: performance increase and speedup!

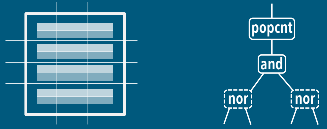
● Three Genotypes + Phenotype



■ Two Genotypes, No Phenotype

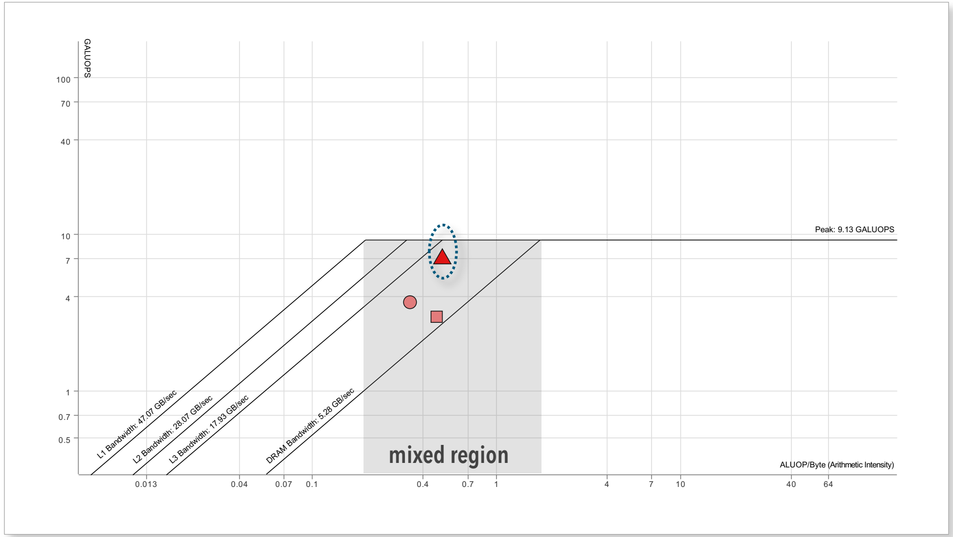


▲ Cache tiling



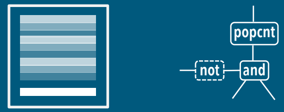
Results obtained with a “special version“ of Intel® Advisor | Platform: Intel® i7-8700K (3.7GHz) with HT/Prefetching/TurboBoost disabled, single core

Improvements, at last!!!

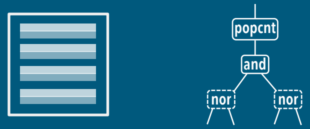


Mixed region, but close to “compute” roof!
Let’s vectorize!

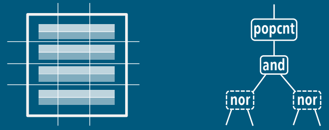
● Three Genotypes + Phenotype



■ Two Genotypes, No Phenotype

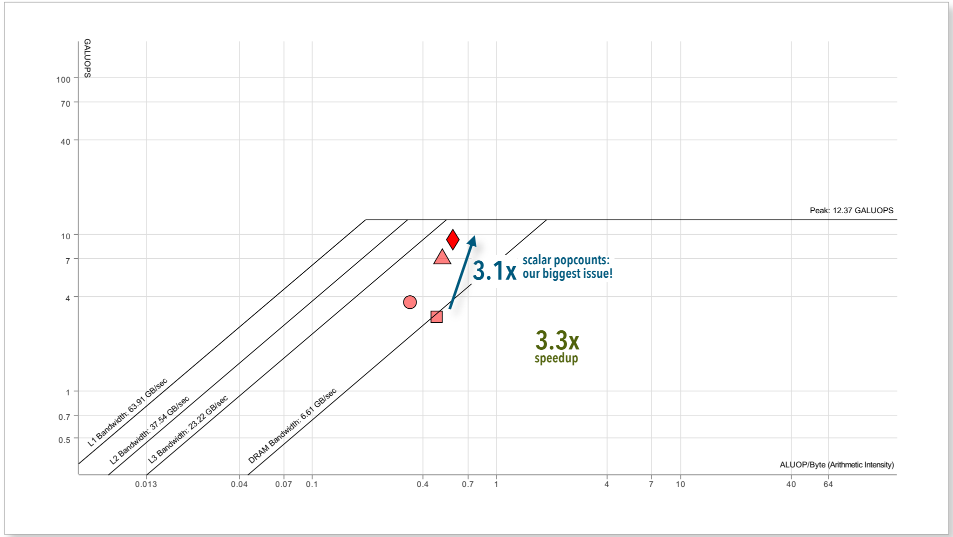


▲ Cache tiling



Results obtained with a “special version” of Intel® Advisor | Platform: Intel® i7-8700K (3.7GHz) with HT/Prefetching/TurboBoost disabled, single core

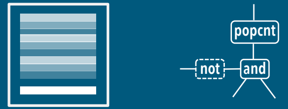
CARM in action …



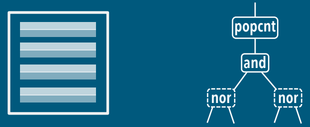
3.1x scalar popcounts: our biggest issue!

3.3x speedup

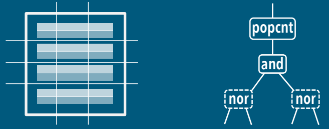
● Three Genotypes + Phenotype



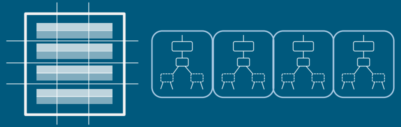
■ Two Genotypes, No Phenotype



▲ Cache tiling



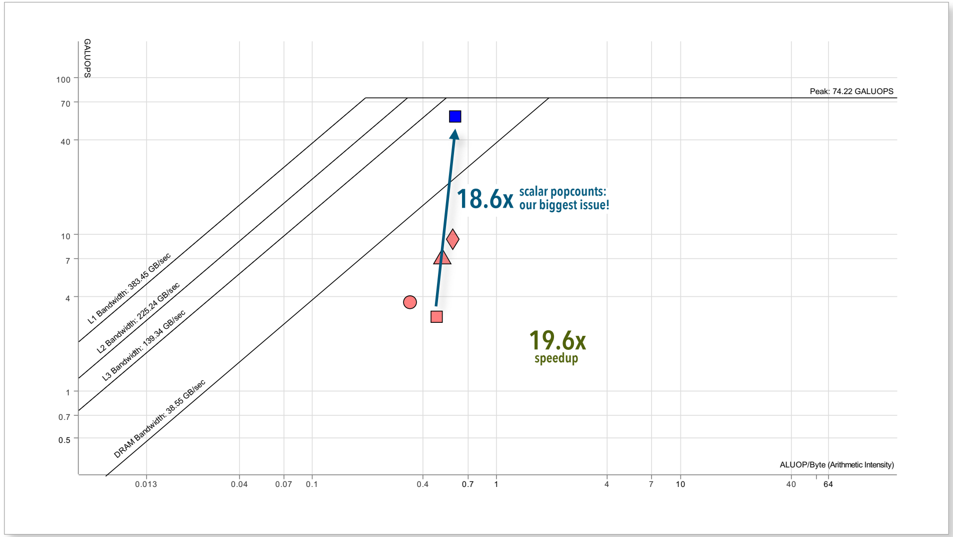
◆ AVX2 Vectorization



Let’s multi-thread it!

Results obtained with a “special version“ of Intel® Advisor | Platform: Intel® i7-8700K (3.7GHz) with HT/Prefetching/TurboBoost disabled, single core

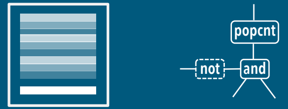
CARM in action …



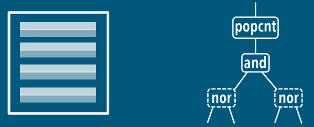
18.6x scalar popcounts: our biggest issue!

19.6x speedup

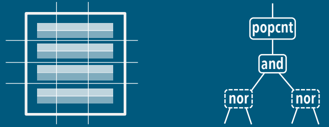
● Three Genotypes + Phenotype



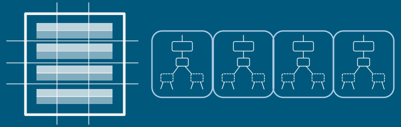
■ Two Genotypes, No Phenotype



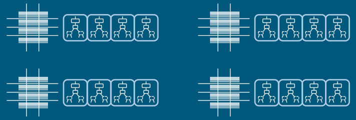
▲ Cache tiling



◆ AVX2 Vectorization

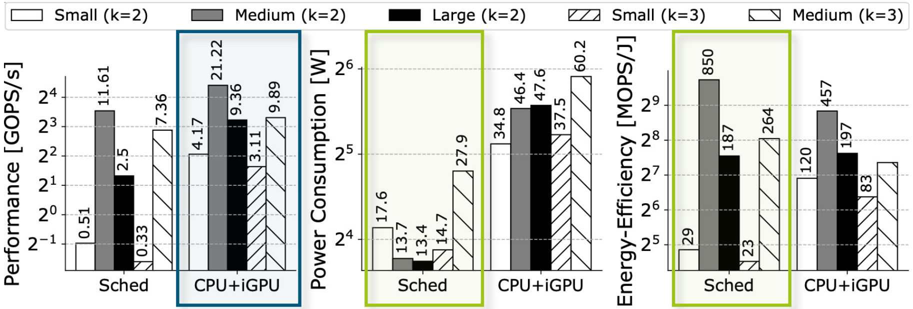


□ Multi-core



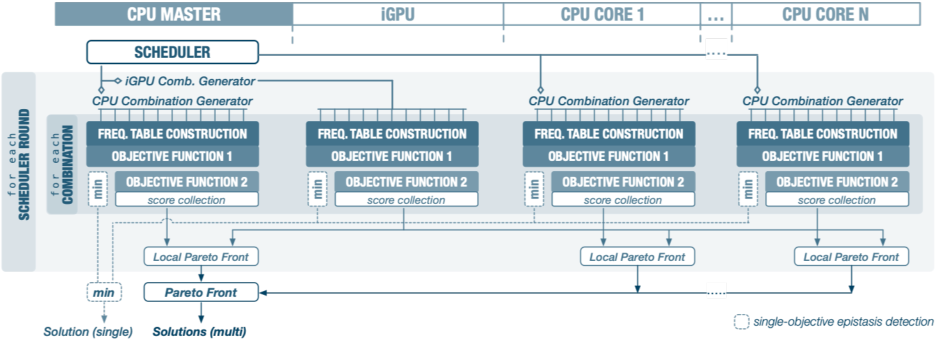
Results obtained with a "special version" of Intel® Advisor | Platform: Intel® i7-8700K (3.7GHz) with HT/Prefetching/TurboBoost disabled, six cores

Epistasis Detection on Intel CPU+iGPU



Best Performance

Best Power and Energy-Efficiency



| R. Campos, D. Marques, S. Santander-Jiménez, L. Sousa, A. Ilic, “Heterogeneous CPU+ iGPU Processing for Efficient Epistasis Detection”, EuroPar (2020)

PERFORMANCE

POWER

ENERGY-EFFICIENCY

CASE-STUDY

Cache-aware Roofline Model: Conclusions

D. Marques, A. Ilic, Z. Matveev and L. Sousa, "Application-driven Cache-Aware Roofline Model", Elsevier FGCS (2020)
R. Nobre, A. Ilic, S. Santander-Jiménez, L. Sousa, "Exploring the Binary Precision Capabilities of Tensor Cores for Epistasis Detection", IPDPS (2020)
R. Campos, D. Marques, S. Santander-Jiménez, L. Sousa, A. Ilic, "Heterogeneous CPU+ iGPU Processing for Efficient Epistasis Detection", EuroPar (2020)
A. Ilic, F. Pratas and 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)
A. Ilic, F. Pratas and L. Sousa, "Cache-aware Roofline Model: Upgrading the Loft", IEEE Computer Architecture Letters (2014)

Thank you!

inesc id
lisboa
20
YEARS
DEFINING TECHNOLOGY

FCT
Fundação
para a Ciência
e a Tecnologia

EUROPEAN UNION
European Regional Development Fund

Supported by Intel Corporation and Portuguese national funds through FCT under projects UIDB/50021/2020 and PTDC/CCI-COM/31901/2017 (HiPErBio).