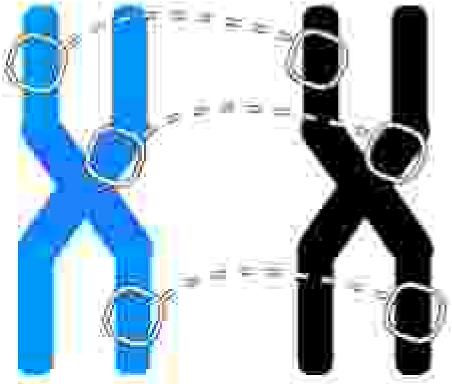
Accelerating epistasis detection on Intel CPUs and discrete GPUs with Intel® Advisor

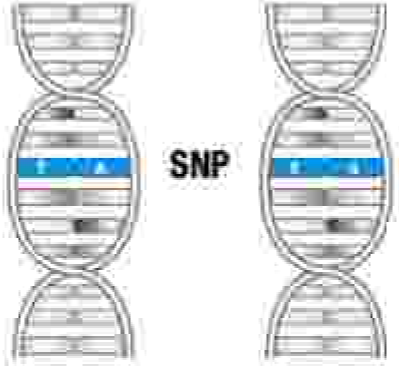
Aleksandar Ilic, Diogo Marques, Rafael Campos and Zakhar Matveev

inesc id lisboa 20 YEARS

DEFINING TECHNOLOGY

Epistasis in a nutshell



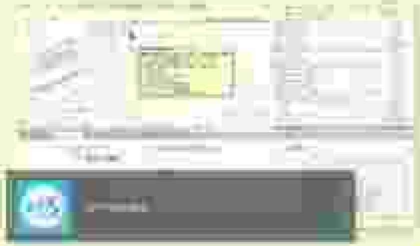


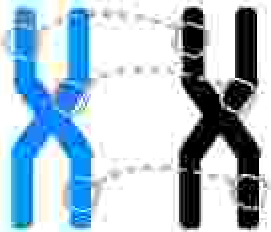
Some SNP interactions may cause life-threating diseases (e.g., Alzheimer, breast cancer)
Discovering which and how many is important, but challenging task!

Boosting Epistasis Detection with oneAPI

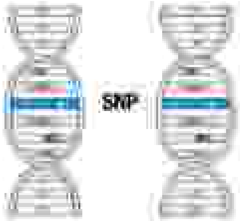
GPU Optimization

CPU Optimization





http://sparcity.eu




Hardware Heterogeneity
OPTIMIZATION
Application Diversity
goals
demands
Cache-aware Roofline
oneAPI
Ad

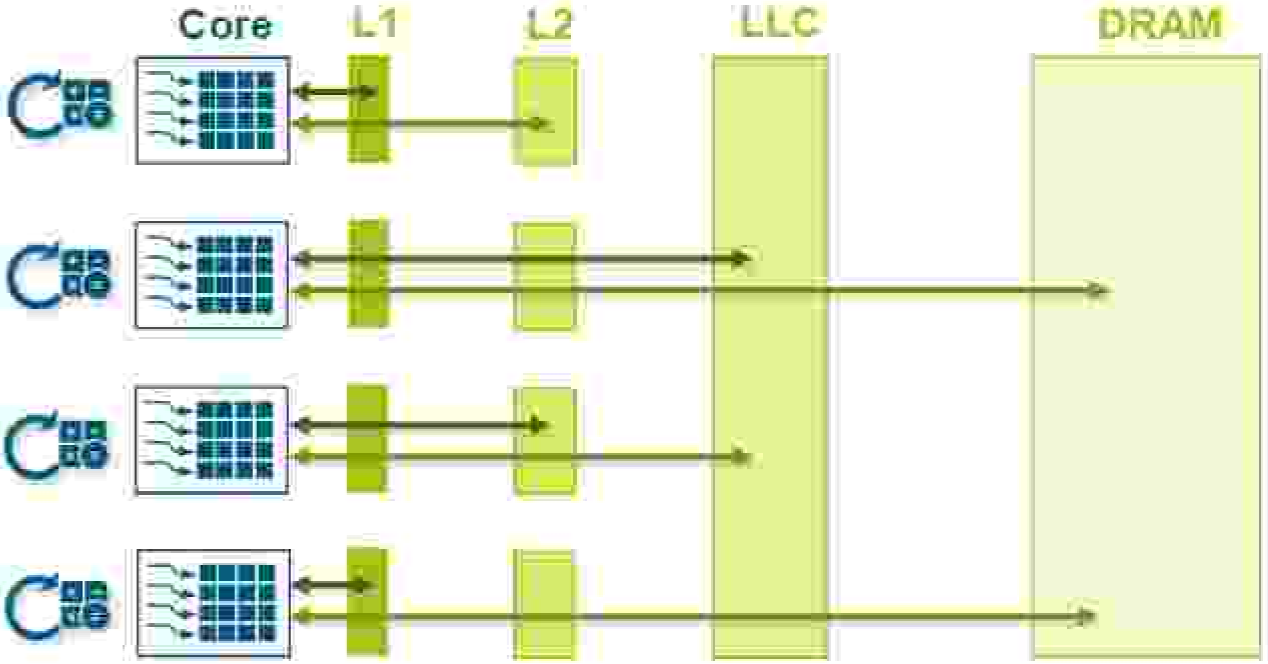
PERFORMANCE POWER ENERGY-EFFICIENCY CASE-STUDY

Outline

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)

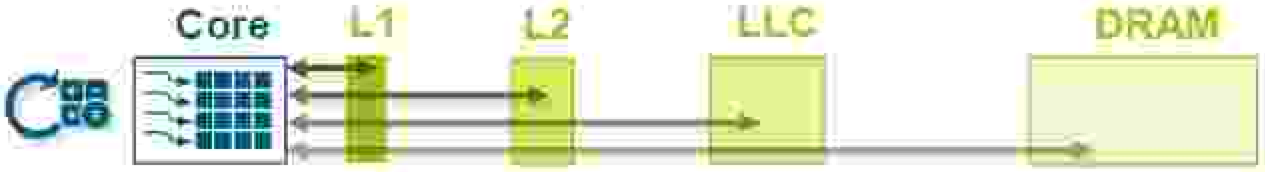
Roofline in a nutshell



Communication overlapped with computation

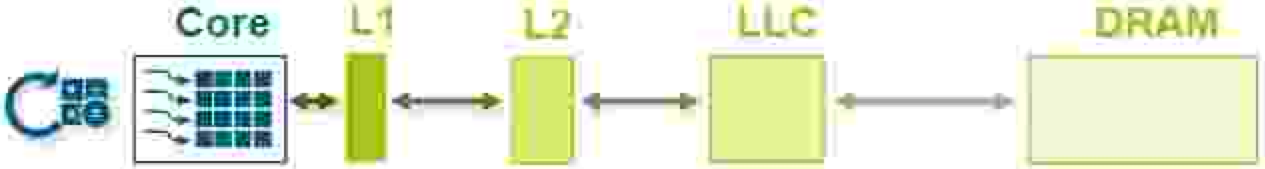
Max performance capped by peak compute throughput or available bandwidth (processor's view)

What is bandwidth?



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

- Obtained via micro-benchmarking

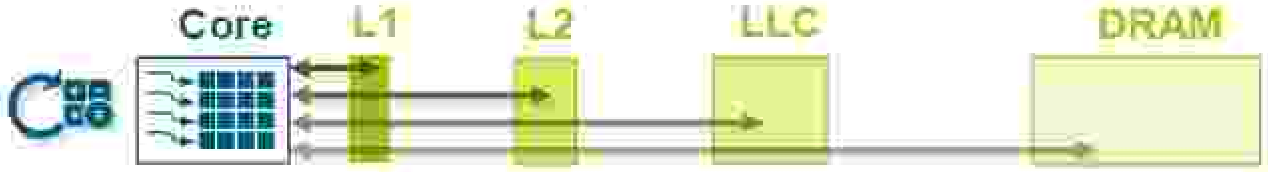


Original Roofline Model (ORM)[2]: Bandwidth between memory levels

- Can be obtained from data-sheets

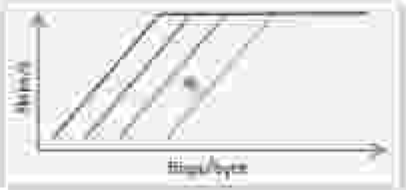
[1] A. Ilic, F. Pratas and L. Sousa, "Cache-aware Roofline Model: Upgrading the Loft", IEEE Computer Architecture Letters (2014)
[2] S. Williams, A. Waterman, D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures", Commun. ACM (2009)

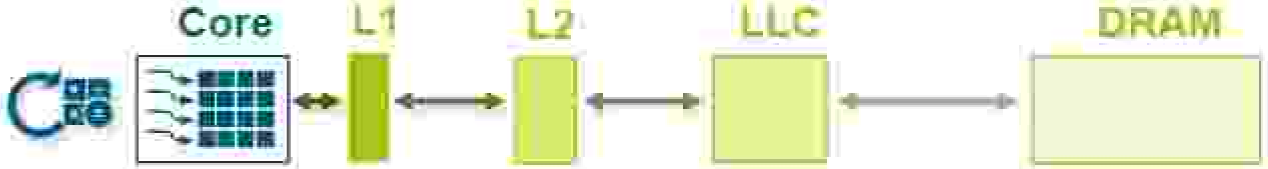
Implications …



Cache-aware Roofline Model[1]

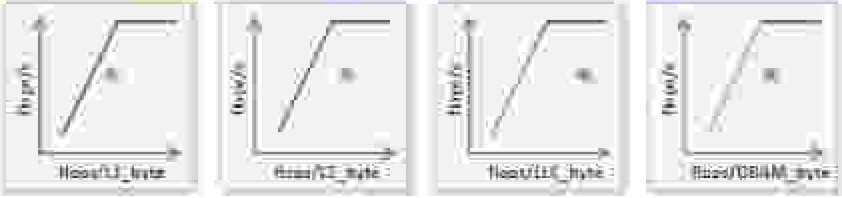
- One model, one arithmetic intensity
- One application "point"





Original Roofline Model[2]

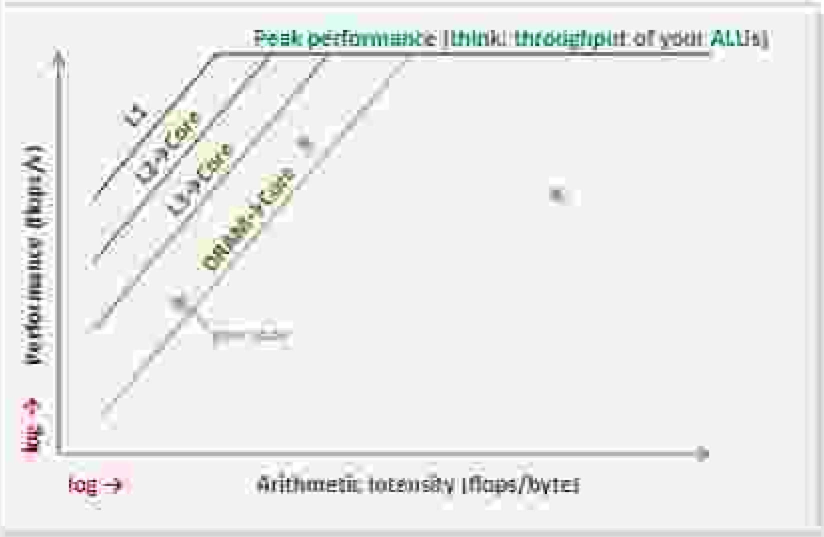
- Several models, several intensities
- Several application "points"



[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



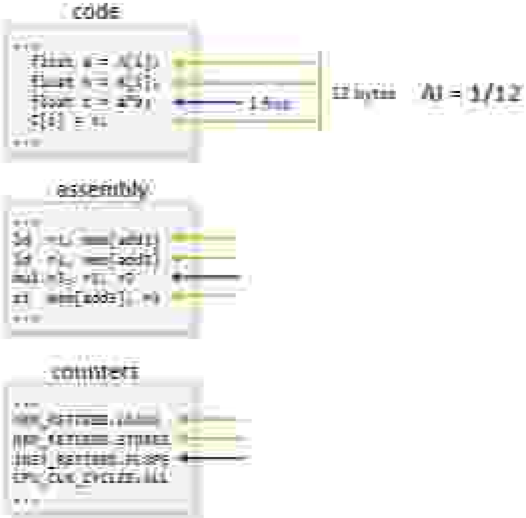
Cache-aware Roofline Model

- Shows absolute architecture maximums*

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

How to "plot" my code?

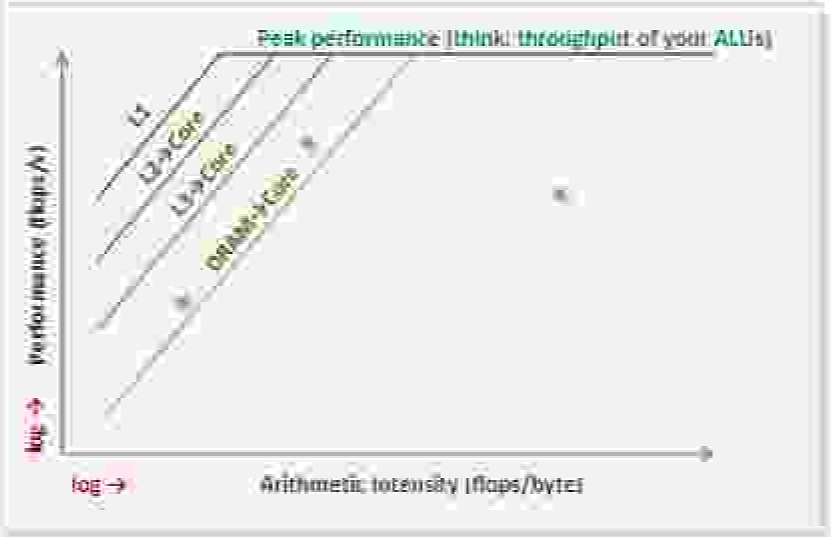
- CARM arithmetic intensity is exactly what you expect it to be!



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)

*We relaxed this requirement in our FCGS paper (2020)

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?

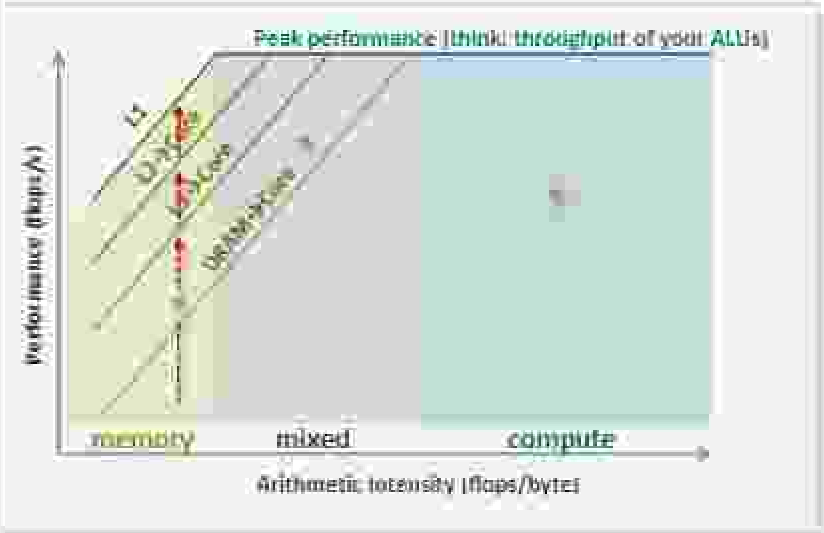
- CARM arithmetic intensity is exactly what you expect it to be!

Intel Advisor Roofline feature

- CARM is there since 2017

 | 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?

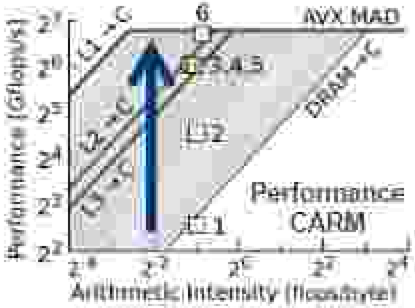
① **Detect the boundness region**
- What are my expected maximums?
- Provides first optimization hints

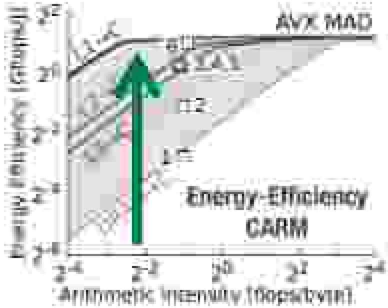
② **Draw an imaginary vertical line**
- What are my main bottlenecks? (observe intersected lines)
- Focus your optimization (aim at surpassing the line above)

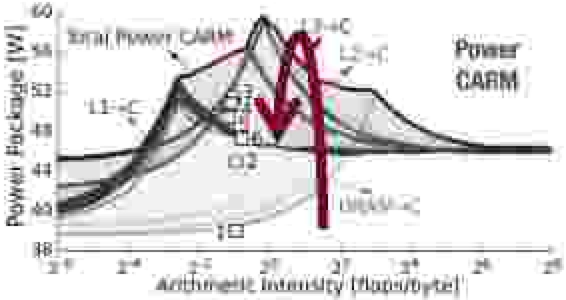
③ **Optimize your code: Break above roofs!**
- You should move up (as your performance improves)
- Unless you restructure the code, or your compiler decides so…

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)

Matrix Multiplication

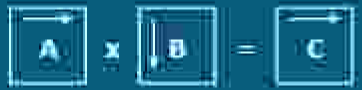




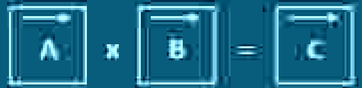


All codes AVX vectorized!*

[1] Basic implementation (row major)

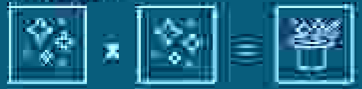


[2] Transposed B (improved mem. access)



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

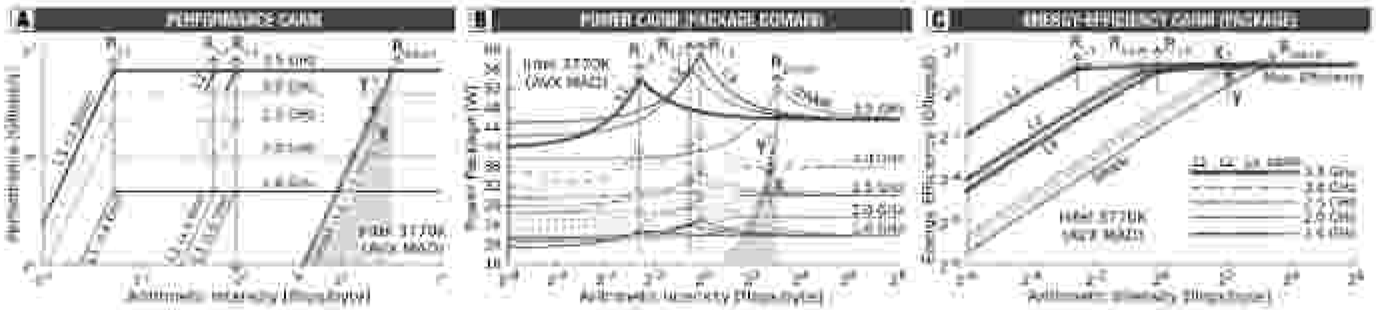
[6] Intel MKL



* 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

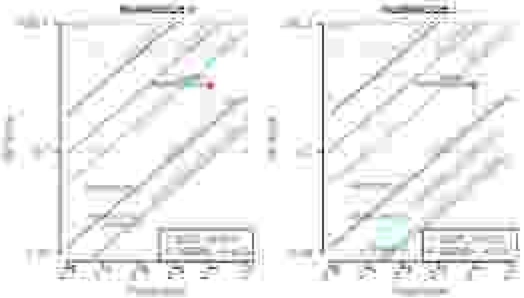
Cache-aware Roofline Model: Extensions

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

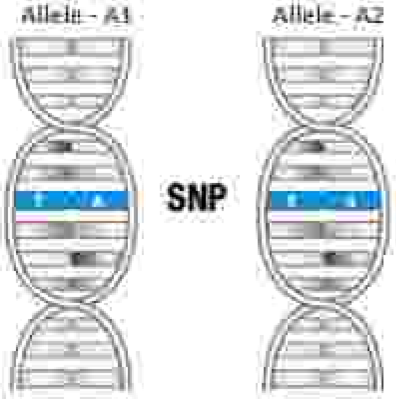
Epistasis Detection

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)

Binarizing your genotype

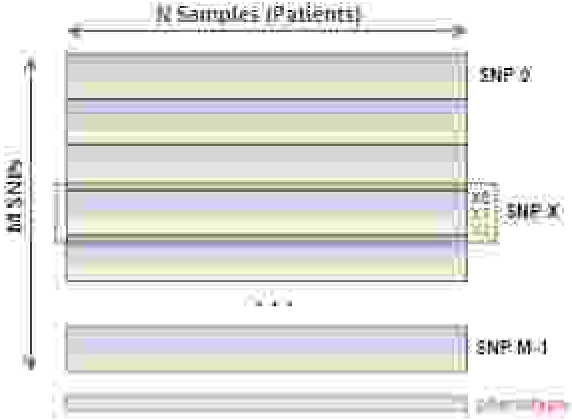
SNP X	P0	P1	Genotype	A1	A2	
X0	0	0	0			Homozygous Major
X1	1	0	1			Heterozygous
X2	0	1	2			Homozygous Minor
phenotype	0	1				

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

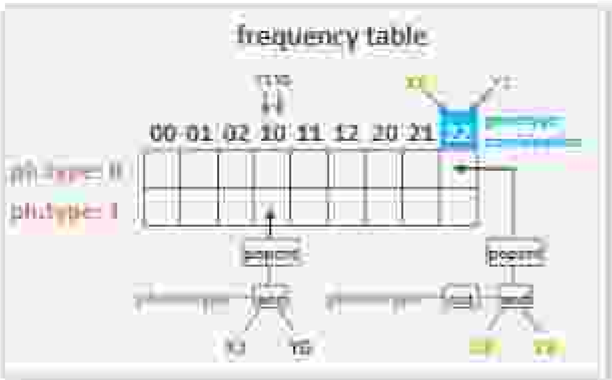


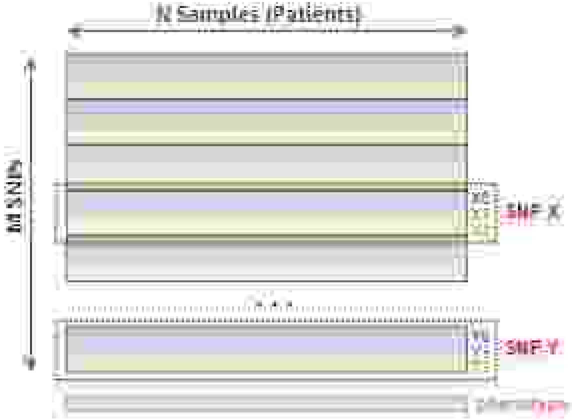
Dataset structure

Our dataset: 10 048 SNPs x 104 448 samples

2-way Epistasis Detection: Pair-wise interaction

Pair-wise interaction: SNPs (X,Y)

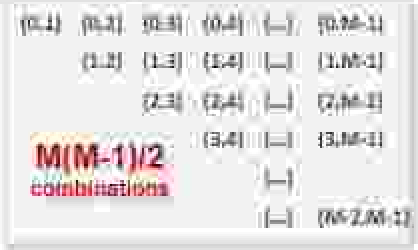




Dataset structure

Our dataset: 10 048 SNPs x 104 448 samples

Search space: All SNP combinations



Our dataset: 50 476 128 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+GPU Processing for Efficient Epistasis Detection", EuroPar (2020)

Epistasis Detection: GPU Optimization

Results obtained with Intel® Advisor | Platform: Intel® HD Graphics P630 GPU (@1.15 GHz)

DPC++ setup

Host code

```
uint* dev_data = (uint*) cl::sycl::malloc_shared(M * N * 3 * sizeof(uint), device, context);
uint* dev_pben = (uint*) cl::sycl::malloc_shared(N * sizeof(uint), device, context);
float* dev_kernel = (float*) cl::sycl::malloc_shared(M * M * sizeof(float), device, context);

for(x = 0; x < M * N * 3; x++)
   dev_data[x] = host_data[x];
for(x = 0; x < N; x++)
   dev_pben[x] = host_pben[x];

cl::sycl::range<2> global_range(M, M);
cl::sycl::range<2> local_range(WORK_GROUP_SIZE, 1);
queue.submit([&](cl::sycl::handler& h)
{
   h.parallel_for<class kernel_xxx>(cl::sycl::nd_range<2>(global_range, local_range),
                          [=](cl::sycl::nd_item<2> it)
   });
      // ...
```

Create USM buffers

Transfer data from host to device

Launch kernel to execute on device

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

The kernel is launched for M*M work-items

Work-items without a valid combination will not do work

DPC++ setup

Device code

```
sycl::range<2> global_range(M, M);
sycl::range<2> local_range([illegible], 1);
queue.submit([&](sycl::handler& h)
{
    h.parallel_for<class [illegible]>(sycl::nd_range<2>(global_range, local_range),
                                      [=](sycl::nd_item<2> id)
    {
        i = id.get_global_id(0);
        j = id.get_global_id(1);

        if(j > i)
        {
            [illegible]
```

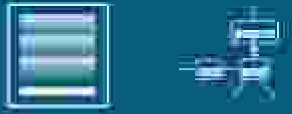
Kernel defined as lambda function

Obtain SNPs from work-item index

(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)

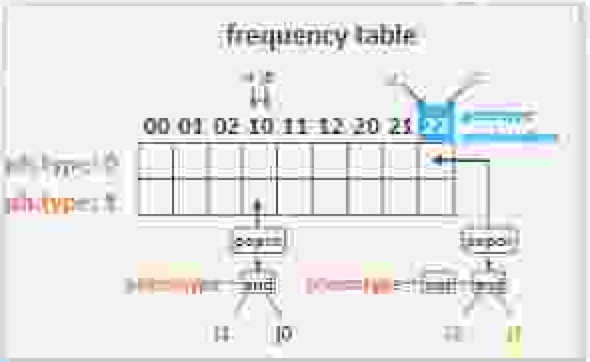
The kernel is launched for M*M work-items

Work-items without a valid combination will not do work



DPC++ implementation

Pair-wise interaction: SNPs (i,j)



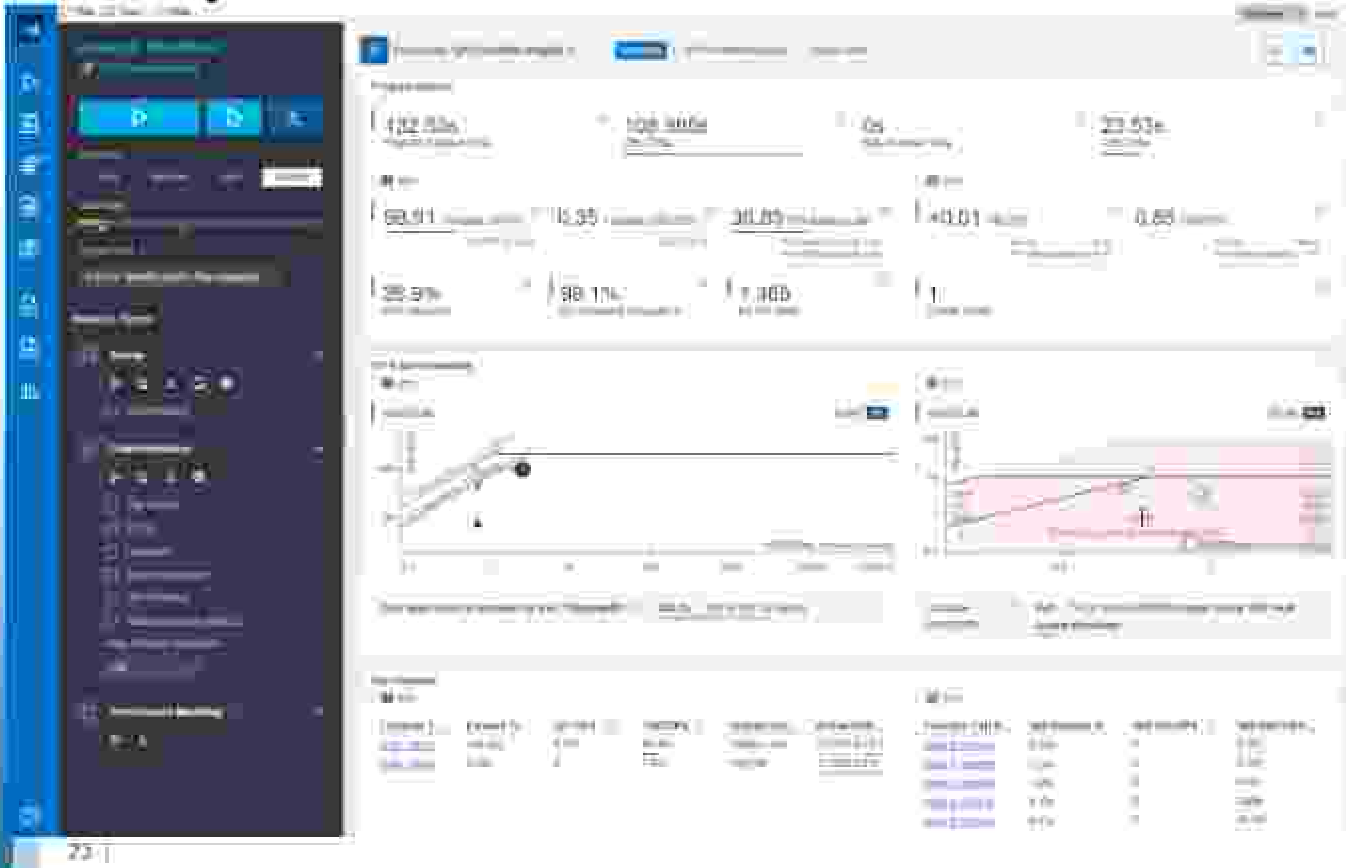
Device code

```
[illegible]
```

The frequency table is filled for SNPs i and j by going through all samples, disposed in columns

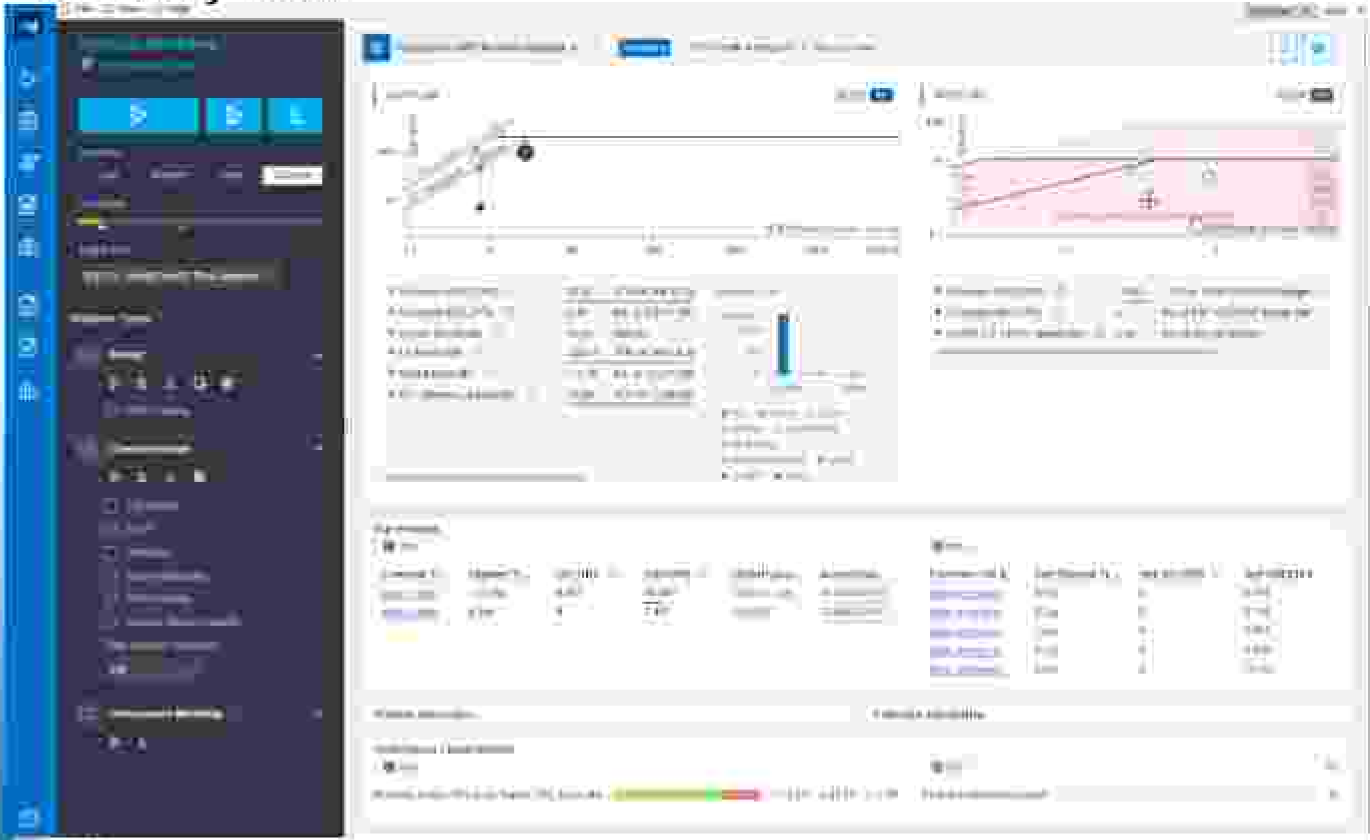
GPU Roofline in Advisor

Summary View:



GPU Roofline in Advisor

Summary View:



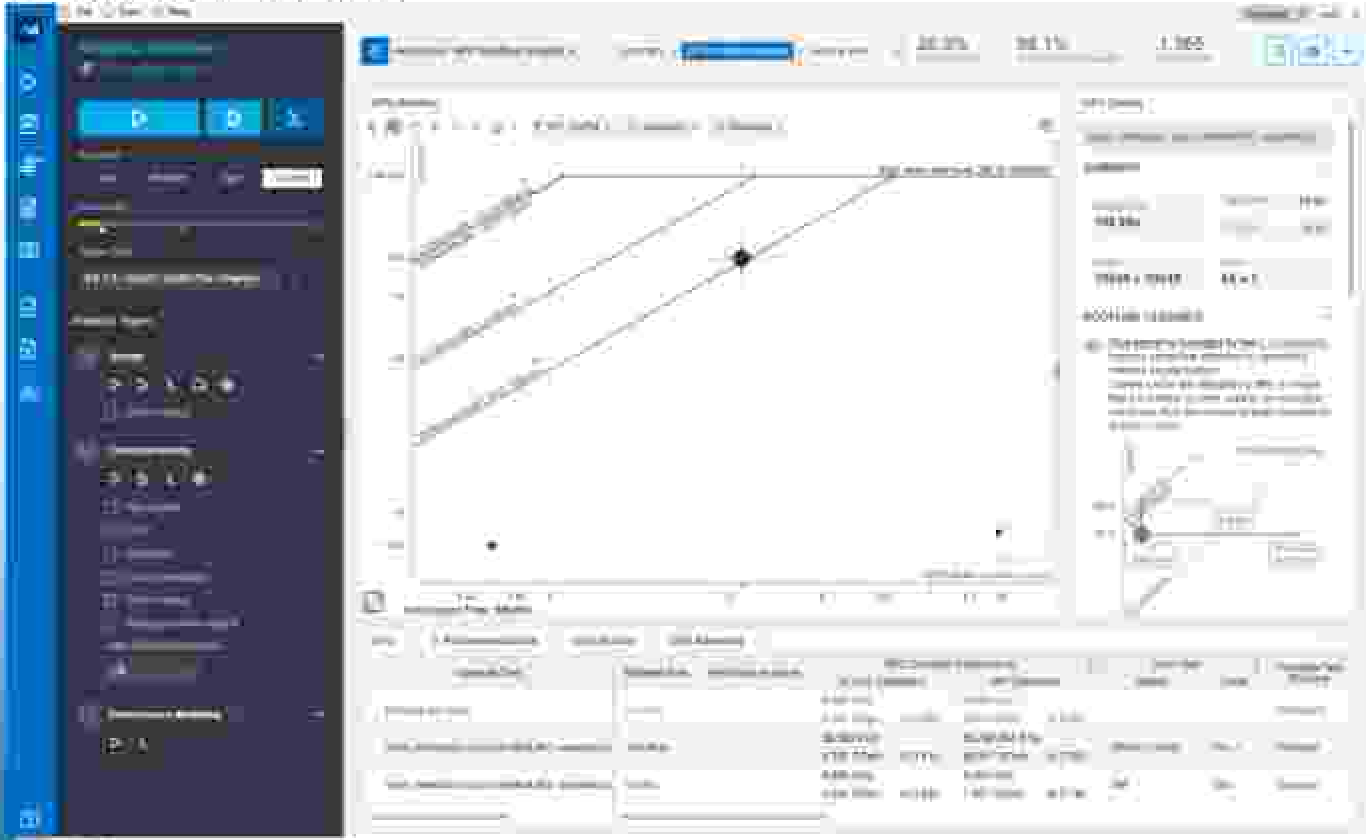
inesc id lisboa

GPU Roofline in Advisor

GPU Optimization

- Three Genotypes + Phenotype

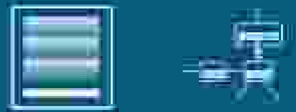
GPU Roofline view:



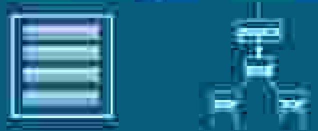
Advisor in action…

Restructuring our algorithm…

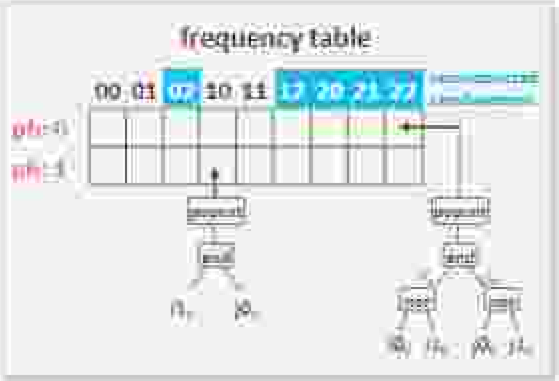
- Three Genotypes + Phenotype

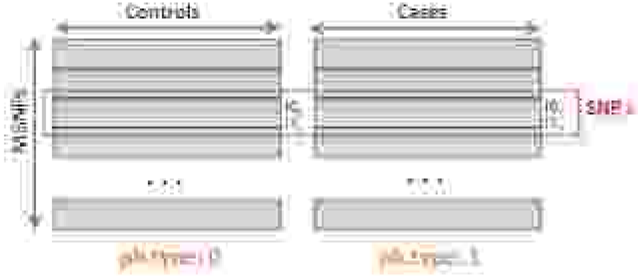


- Two Genotypes, No Phenotype



Pair-wise interaction: SNPs (i,j)





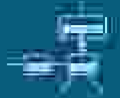
"New" Dataset structure
(removed: phenotype and genotype 2)

Reducing memory transfers!

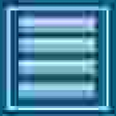
 | R. Nobre, A. Ilic, S. Santander-Jiménez, L. Sousa, "Exploring the Binary Precision Capabilities of Tensor Cores for Epistasis Detection", IPDPS [2020

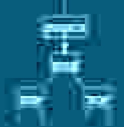
GPU Optimization

- Three Genotypes + Phenotype



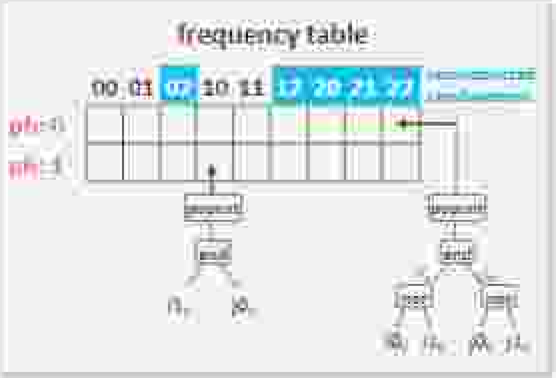
- Two Genotypes, No Phenotype





DPC++ implementation

Pair-wise interaction: SNPs (i,j)

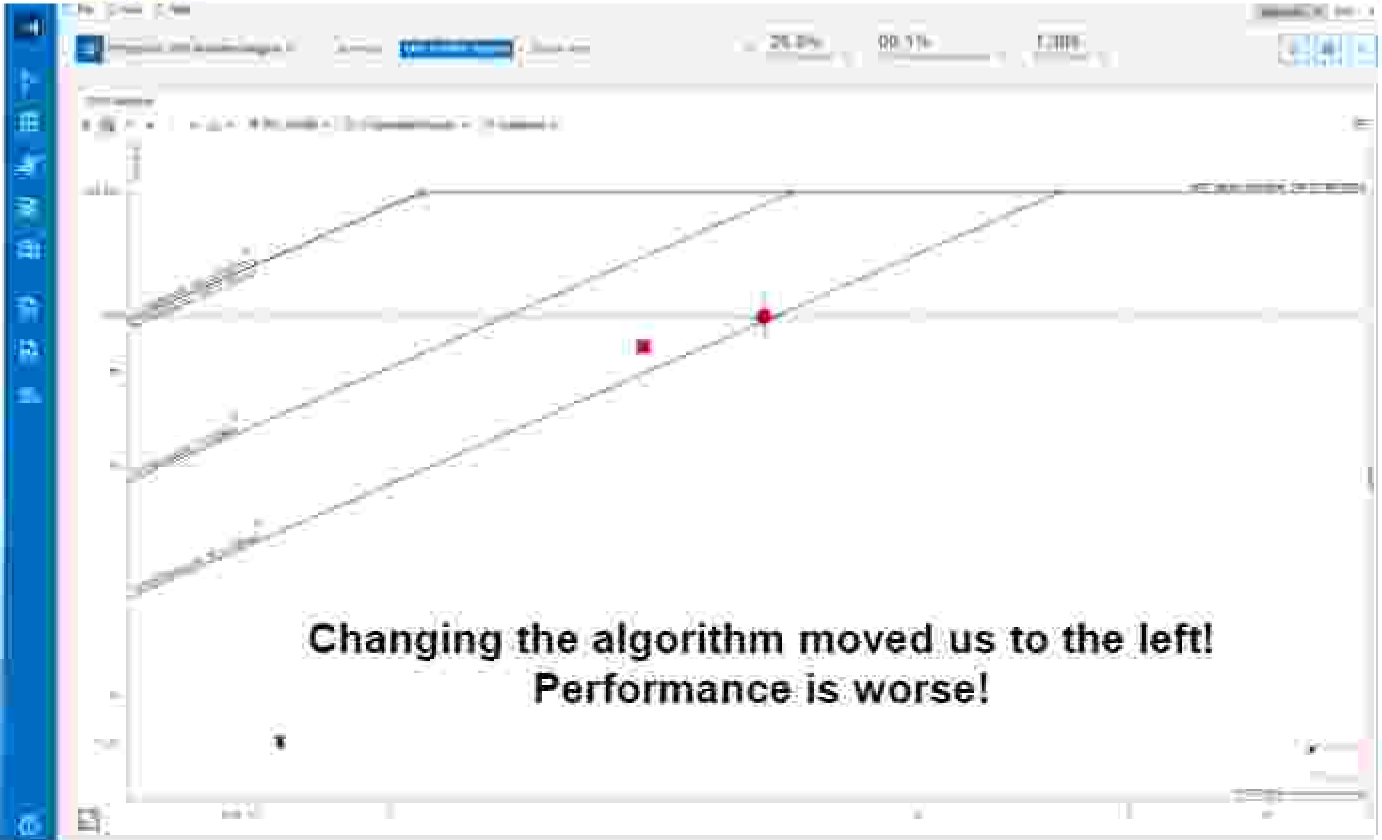


Device code

The frequency table is filled for SNPs i and j is filled separately for each phenotype

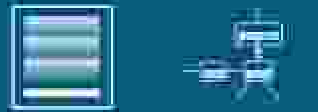
 | R. Nobre, A. Ilic, S. Santander-Jiménez, L. Sousa, "Exploring the Binary Precision Capabilities of Tensor Cores for Epistasis Detection", IPDPS (2020

Advisor in action…

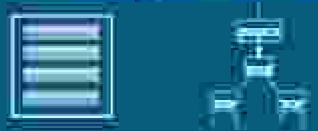


GPU Optimization

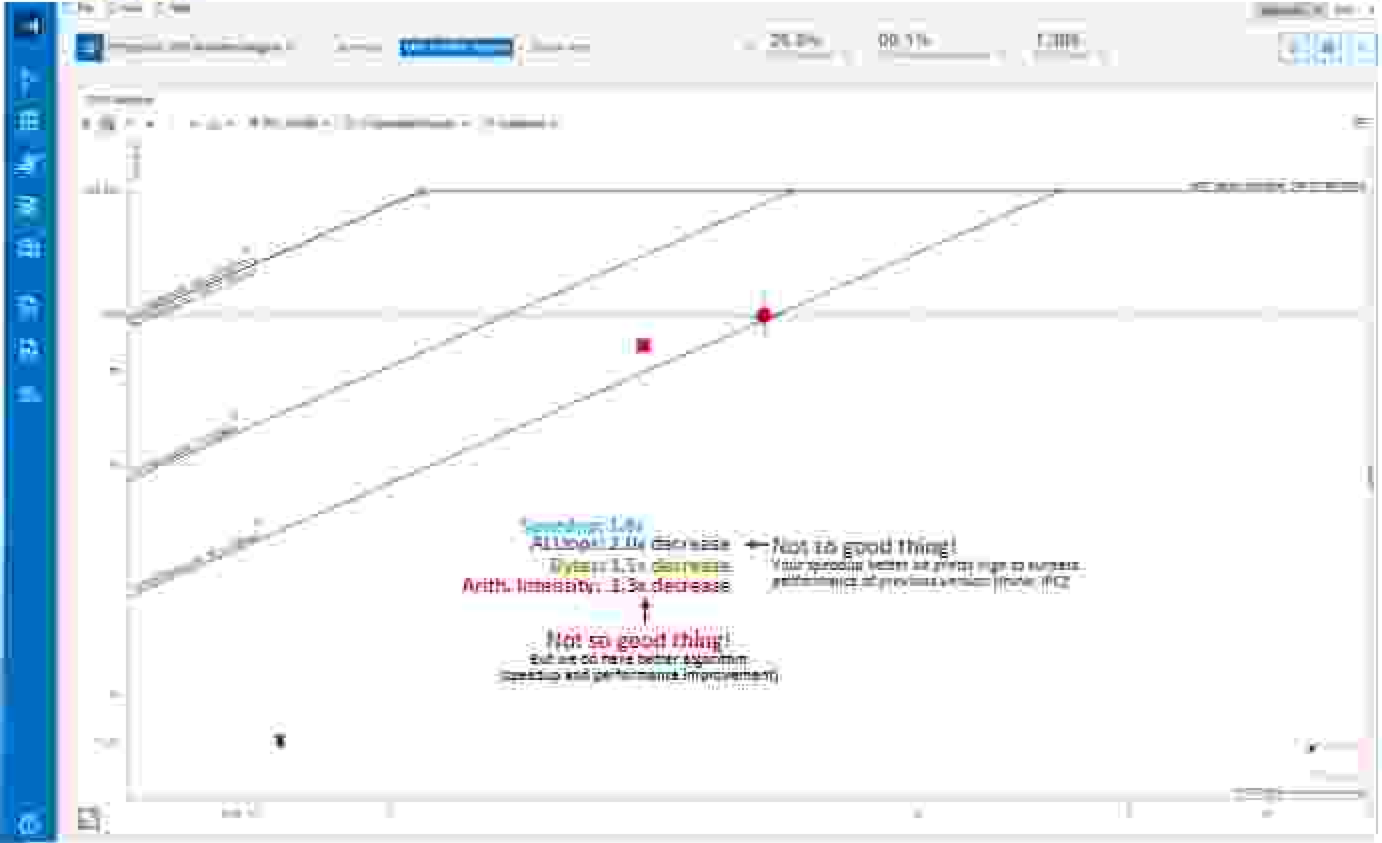
- Three Genotypes + Phenotype



- Two Genotypes, No Phenotype

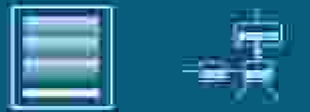


Advisor in action…

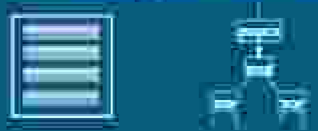


GPU Optimization

- Three Genotypes + Phenotype



- Two Genotypes, No Phenotype



inesc id lisboa

Let's continue optimizing...

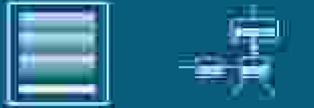
Transposed Dataset
(inverted SNPs and samples)

Device code

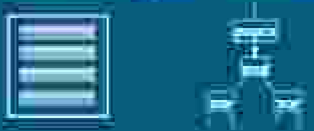
Improving memory accesses by dataset transposition
Coalescing of memory accesses by the work-items

GPU Optimization

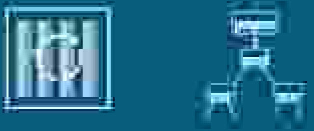
- Three Genotypes + Phenotype



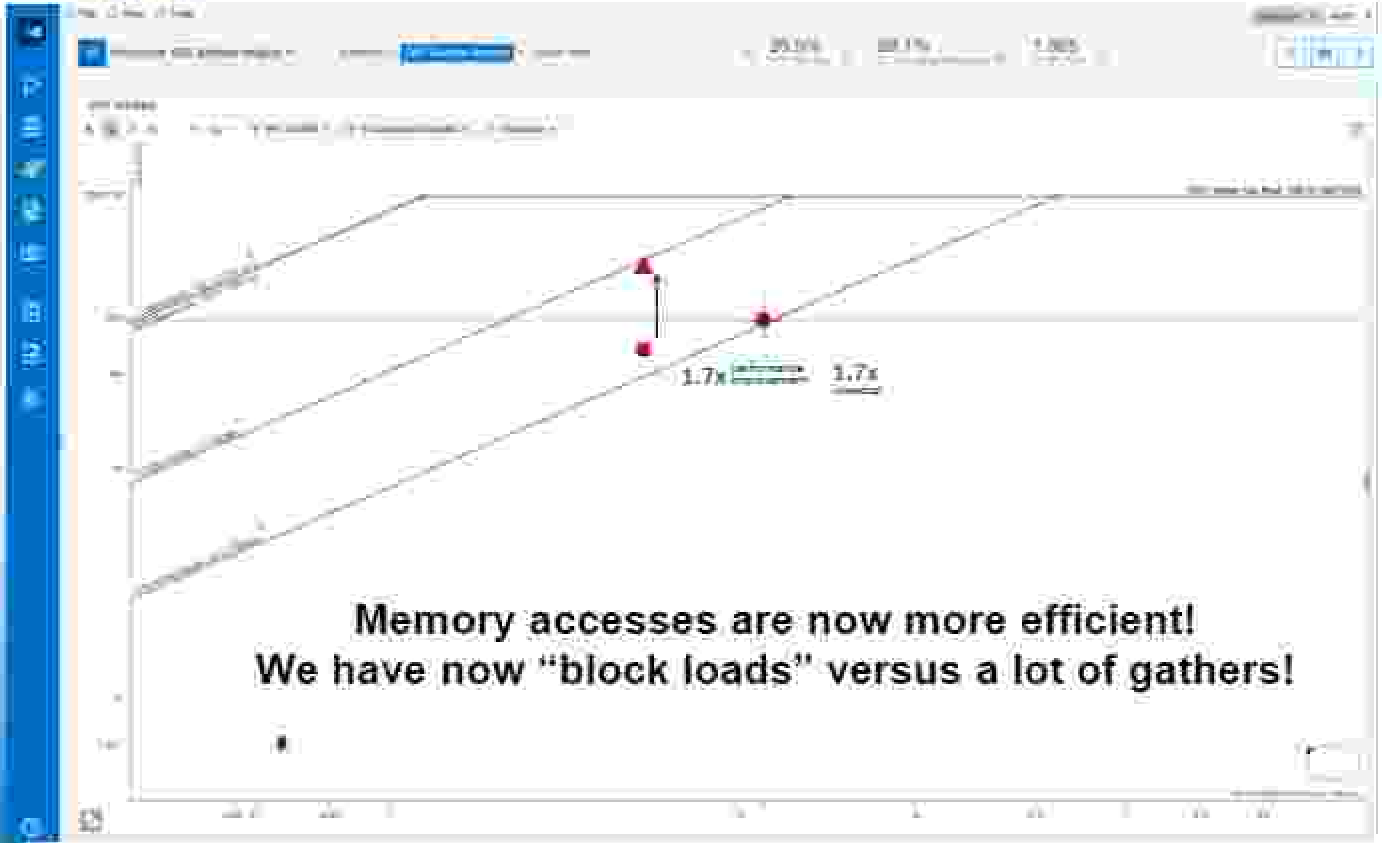
- Two Genotypes, No Phenotype



- Transposed Dataset

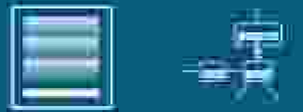


Advisor in action…

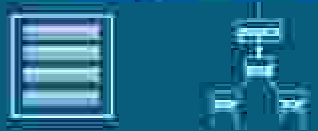


GPU Optimization

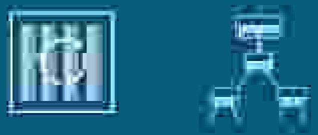
- Three Genotypes + Phenotype



- Two Genotypes, No Phenotype



- Transposed Dataset



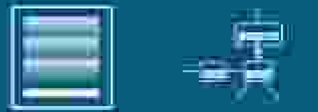
Advisor in action…

This allows to perform disease detection 1.7x faster!

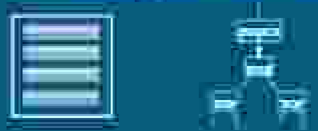
How can we further improve this result?

GPU Optimization

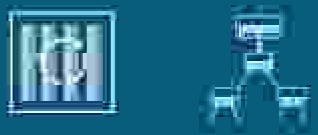
Three Genotypes + Phenotype



Two Genotypes, No Phenotype



Transposed Dataset



inesc id lisboa

Data set tiling

Device code

```
// phenotype zero
SNPi = &data_zeros[(i_block * N_zeros + (i - i_block)) * 2];
SNPj = &data_zeros[(j_block * N_zeros + (j - j_block)) * 2];
for(p = 0; p < 2 * block_snp * N_zeros; p += 2 * B_SNP)
{
    di2 = ~(SNPi[p] | SNPi[p + 1]);
    dj2 = ~(SNPj[p] | SNPj[p + 1]);

    ft[0] += cl::sycl::popcount(SNPi[p] & SNPj[p]);
    ft[1] += cl::sycl::popcount(SNPi[p] & SNPj[p + 1]);
    ft[2] += cl::sycl::popcount(SNPi[p] & dj2);
    ft[3] += cl::sycl::popcount(SNPi[p + 1] & SNPj[p]);
    ft[4] += cl::sycl::popcount(SNPi[p + 1] & SNPj[p + 1]);
    ft[5] += cl::sycl::popcount(SNPi[p + 1] & dj2);
    ft[6] += cl::sycl::popcount(di2 & SNPj[p]);
    ft[7] += cl::sycl::popcount(di2 & SNPj[p + 1]);
    ft[8] += cl::sycl::popcount(di2 & dj2);
}
// phenotype one
...
```

The data set is iterated by rows of size 2 * B_{SNP}

The 3rd genotype is computed

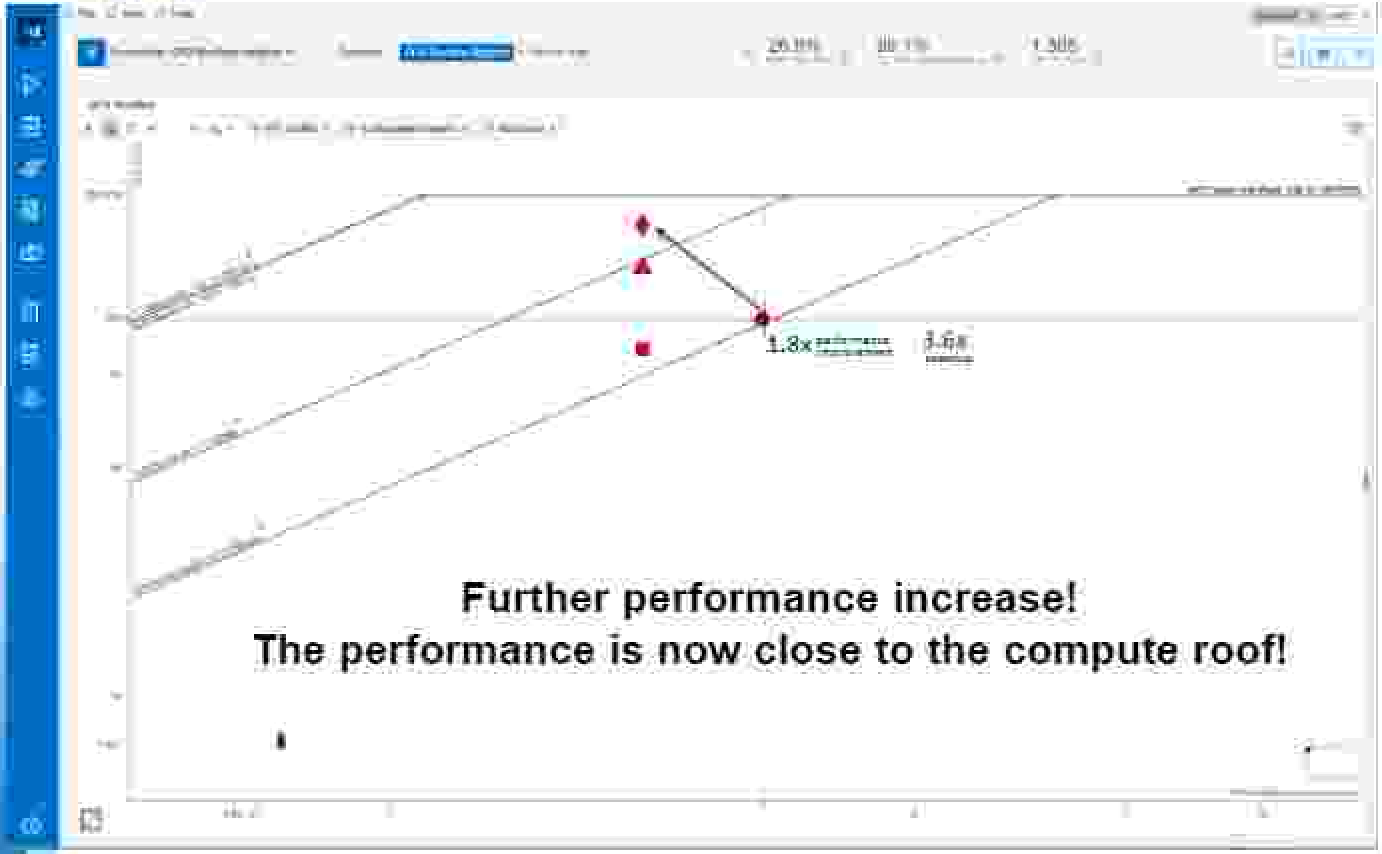
The frequency table is filled using 2 genotypes

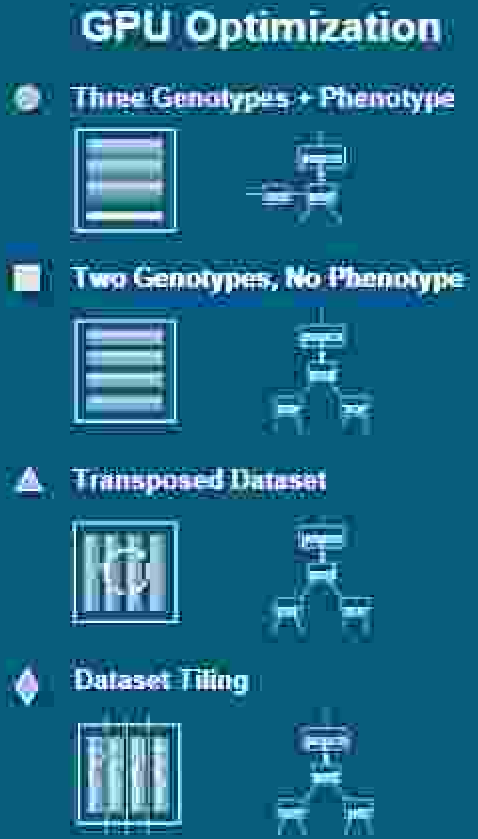
Tiling our dataset to squeeze the maximums!
Using a tile size of B_{SNP} we can maintain constant access stride

GPU Optimization

- Three Genotypes + Phenotype
- Two Genotypes, No Phenotype
- Transposed Dataset
- Dataset Tiling

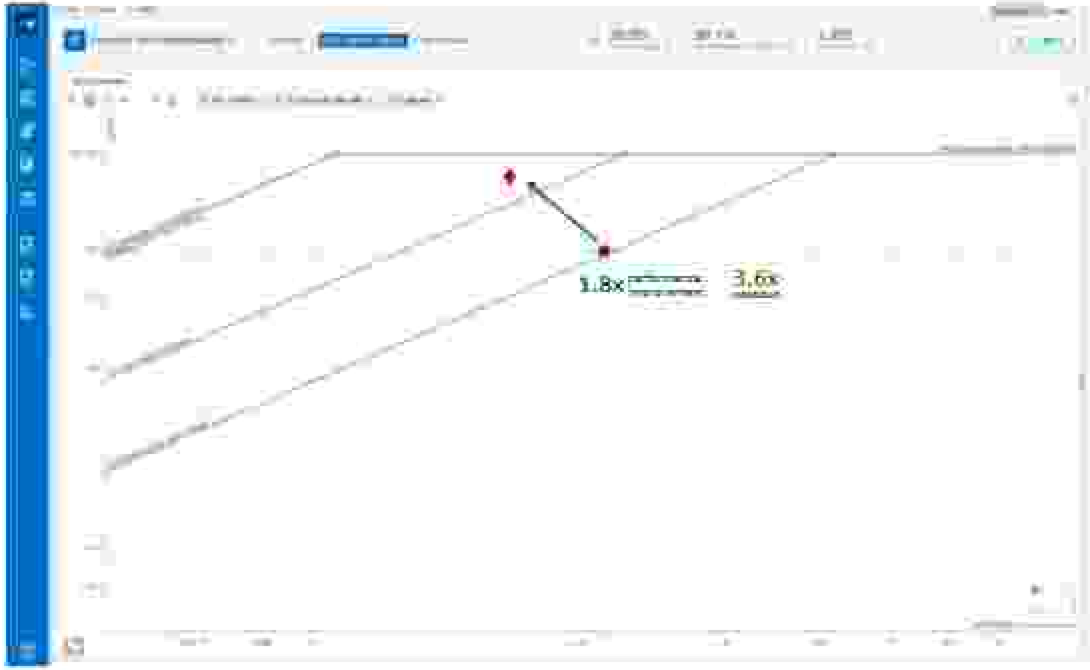
Advisor in action…





Conclusions

Conclusions



The insights of Intel Advisor can allow to vastly improve performance

This allows to perform epistasis detection 3.6x faster!

These techniques can be applied to applications in CPU or GPU

PERFORMANCE POWER ENERGY-EFFICIENCY CASE-STUDY

Acknowledgements

Special thanks to Sugata Tiorewala, the Intel Advisor team, and Igor Vorobtsov

Intel DevMesh Project – [illegible]
HiPerBio Repository – https://github.com/hiperbio

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, 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, Portuguese FCT national funds and EuroHPC Joint Undertaking under projects UIDB/50021/2020, PTDC/CCI-COM/31901/2017 (HiPErBio), and grant agreement no. 956213 (Sparcity)