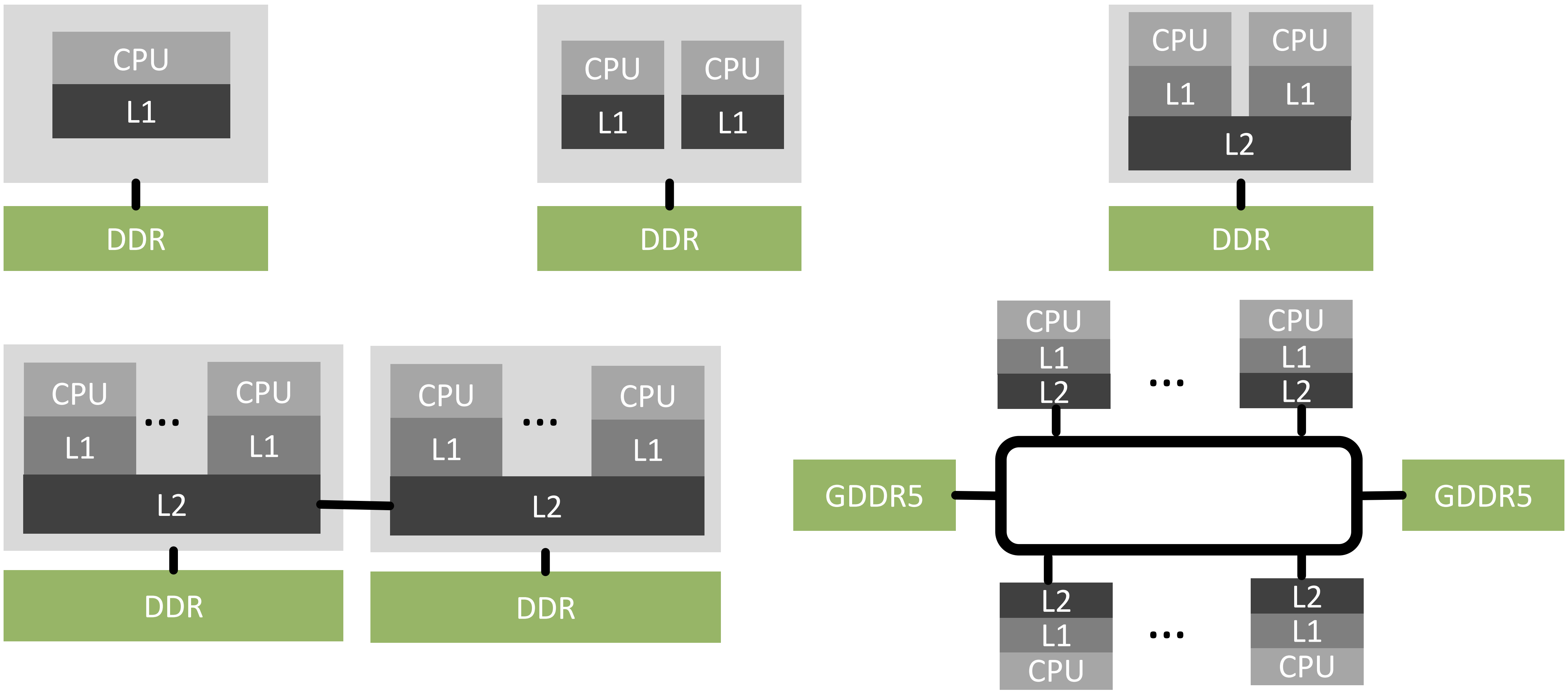
TORSTEN HOEFLER, SABELA RAMOS, CARLOS OSUNA, FELIX THALER, STEFAN MOOSBRUGGER, OLIVER FUHRER

Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL and the COSMO Weather Code

SPCL

Microarchitectures are becoming more and more complex



How to optimize codes for these complex architectures?

- **Performance engineering**: ***"encompasses the set of roles, skills, activities, practices, tools, and deliverables applied at every phase of the systems development life cycle which ensures that a solution will be designed, implemented, and operationally supported to meet the non-functional requirements for performance (such as throughput, latency, or memory usage)."***

- **Manually profile codes and tune them to the given architecture**
  - Requires highly-skilled performance engineers
  - Need familiarity with

    NUMA (topology, bandwidths etc.)

    Caches (associativity, sizes etc.)

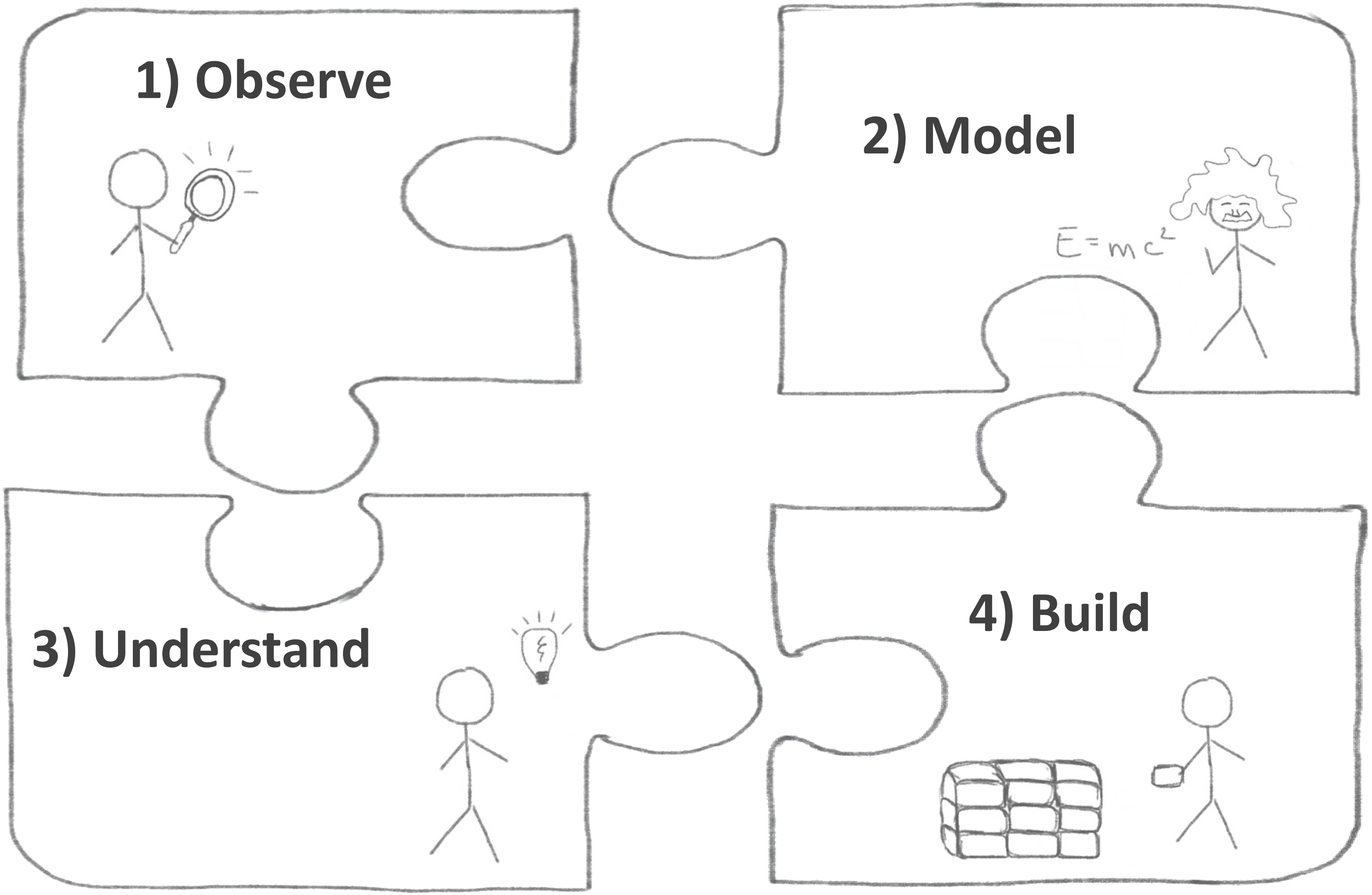
    Microarchitecture (number of outstanding loads etc.)

Trust me, I'm an engineer!

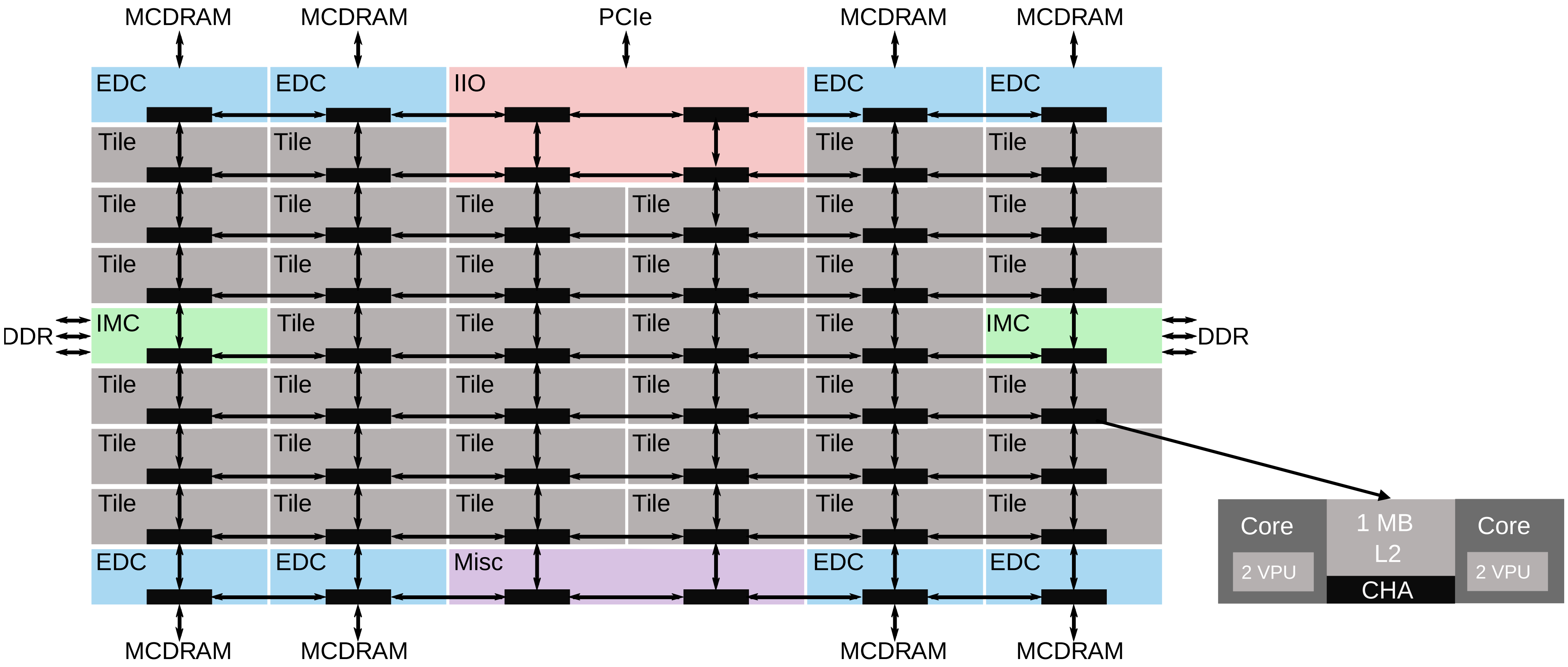
<title>code ninja</title>

An engineering example – Tacoma Narrows Bridge

Scientific Performance Engineering

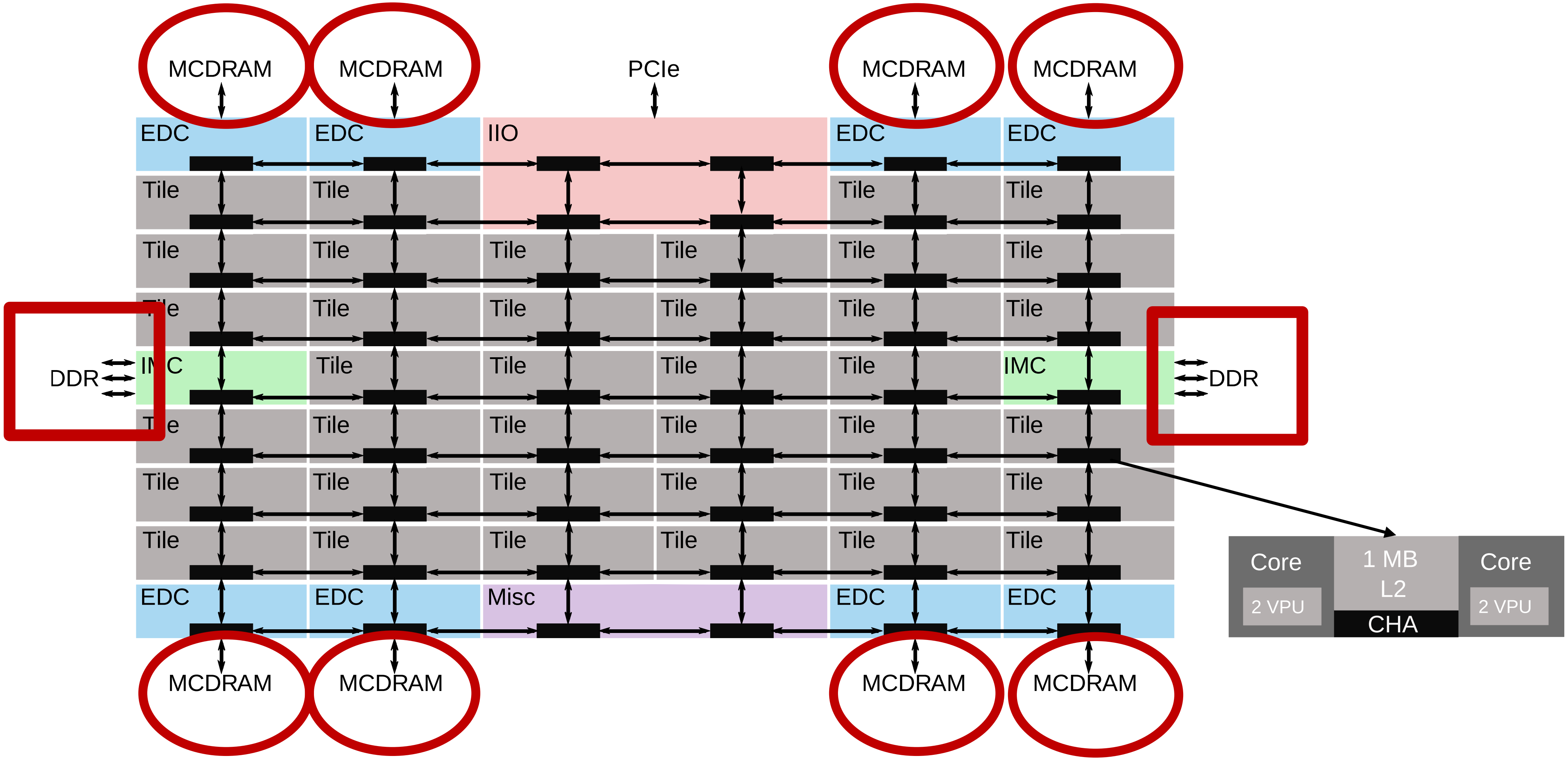


Modeling by example: KNL Architecture (mesh)



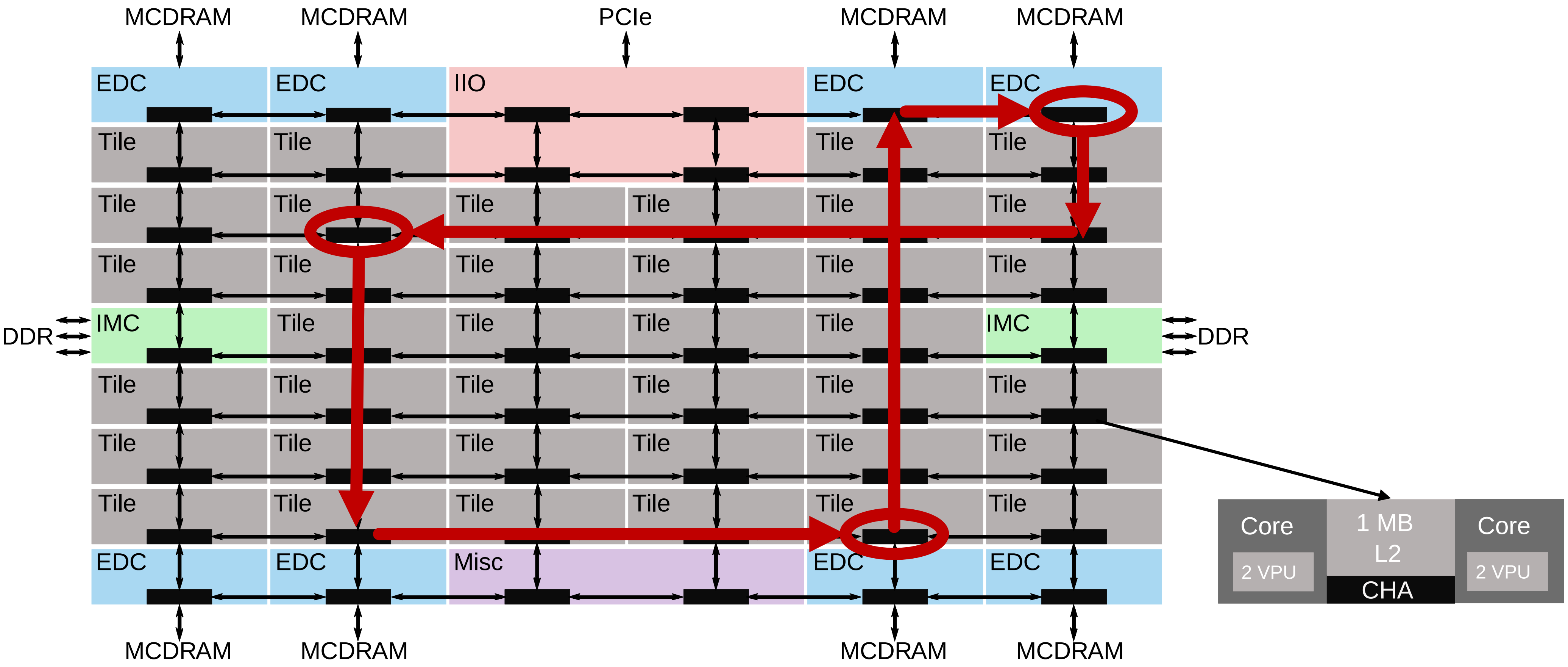
S. Ramos and T. Hoefler: Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL, IPDPS’17

KNL Architecture (memory: Flat & Cache)



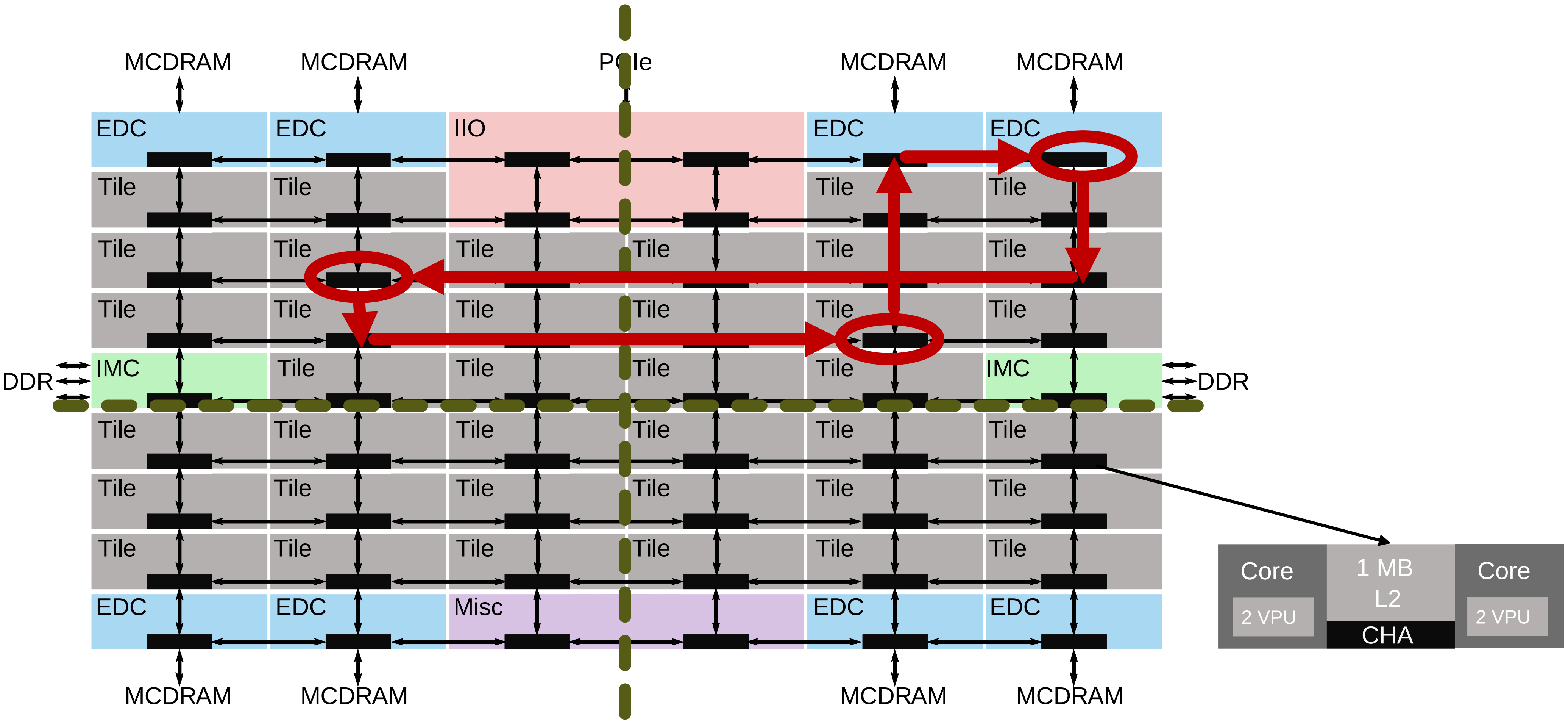
S. Ramos and T. Hoefler: Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL, IPDPS'17

KNL Architecture (all to all mode)



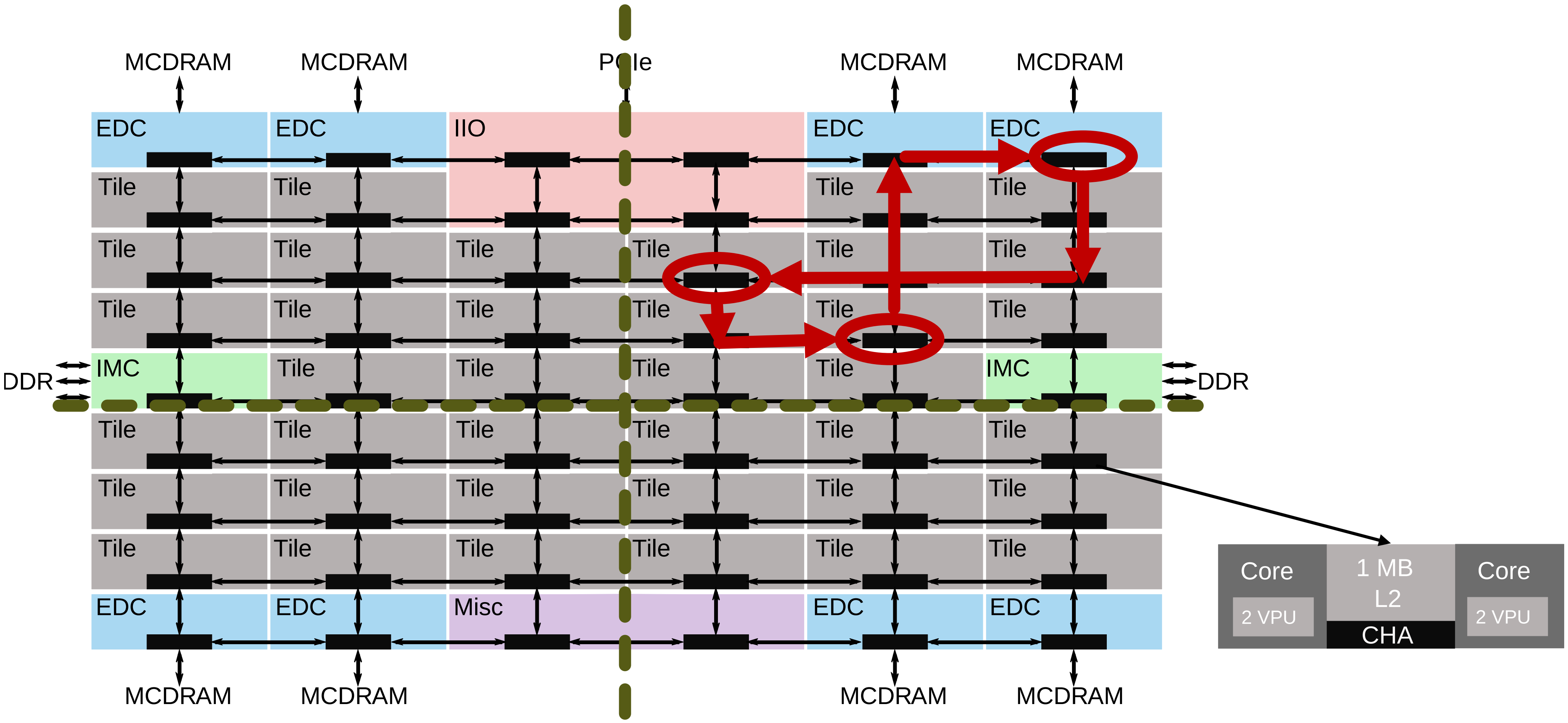
S. Ramos and T. Hoefler: Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL, IPDPS'17

KNL Architecture (Quadrant or Hemisphere)



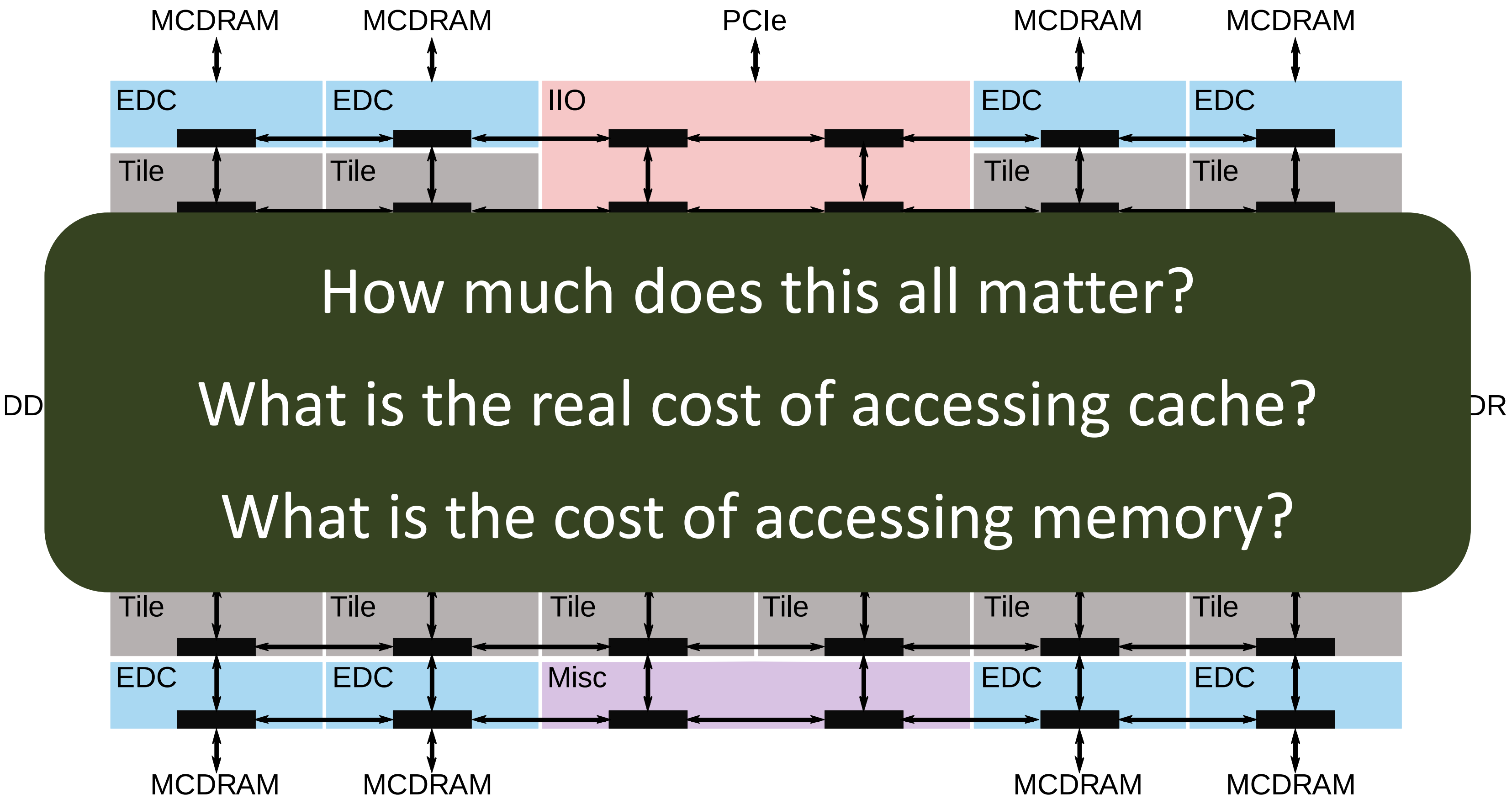
S. Ramos and T. Hoefler: Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL, IPDPS'17

KNL Architecture (SNC-4 or SNC-2)



S. Ramos and T. Hoefler: Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL, IPDPS'17

KNL Architecture



How much does this all matter?

What is the real cost of accessing cache?

What is the cost of accessing memory?

S. Ramos and T. Hoefler: Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL, IPDPS’17

Step 1: Understand core-to-core transfers – MESIF cache coherence

Write back overhead

Location: only 5-15% difference

Contention effects?

That is curious!

Step 2: Understand core-to-memory transfers – DRAM and MCDRAM

MCDRAM 20% slower!

MCDRAM 4-6x faster!

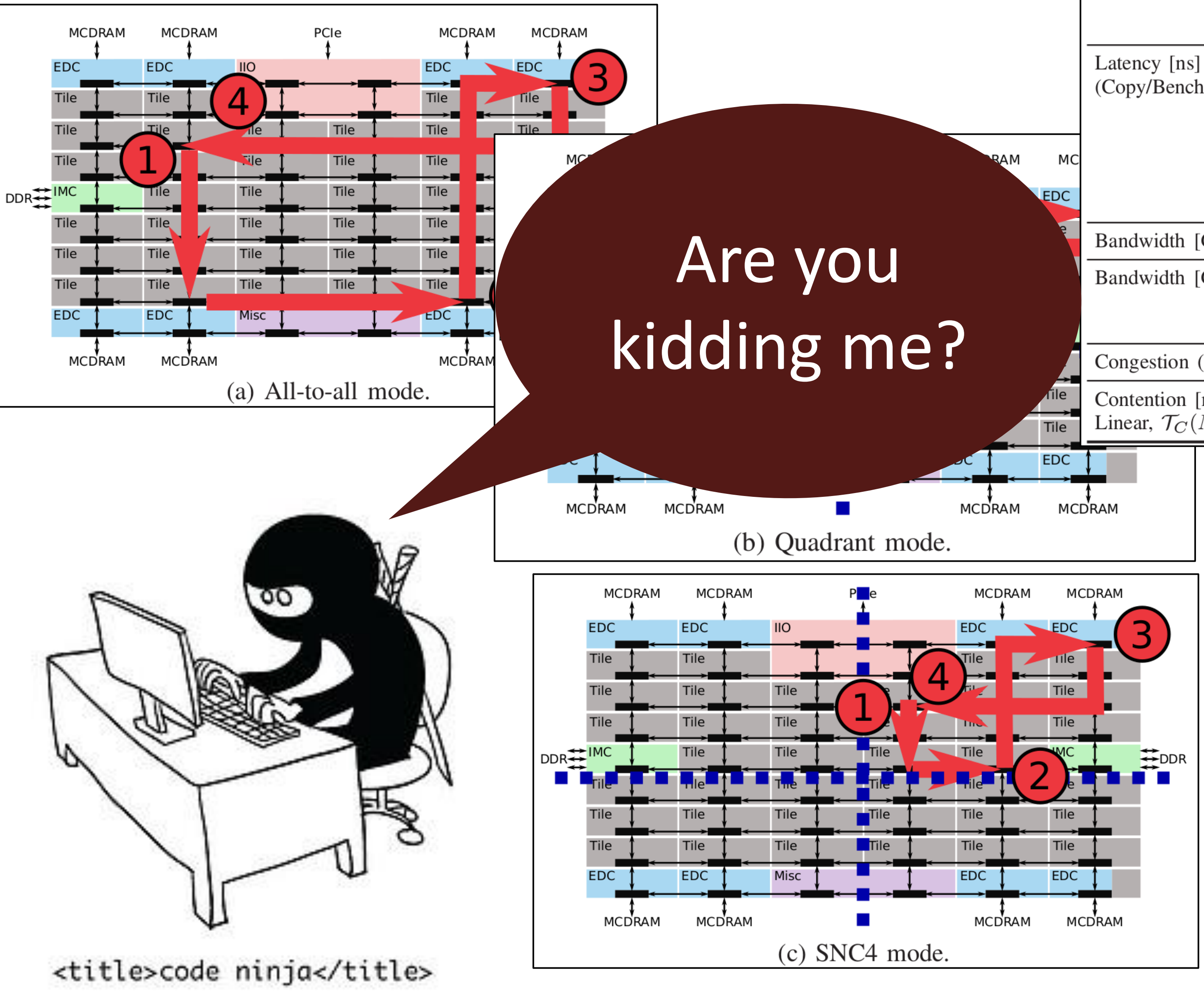
Need to read **and** write for full bandwidth

Cache mode >20% slower

Bandwidth suffers a bit

All values are medians within 10% of the 95% nonparametric CI, cf. TH, RB: “Scientific Benchmarking of Parallel Computing Systems”, SC16

Performance engineers optimize your code!



(a) All-to-all mode.

(b) Quadrant mode.

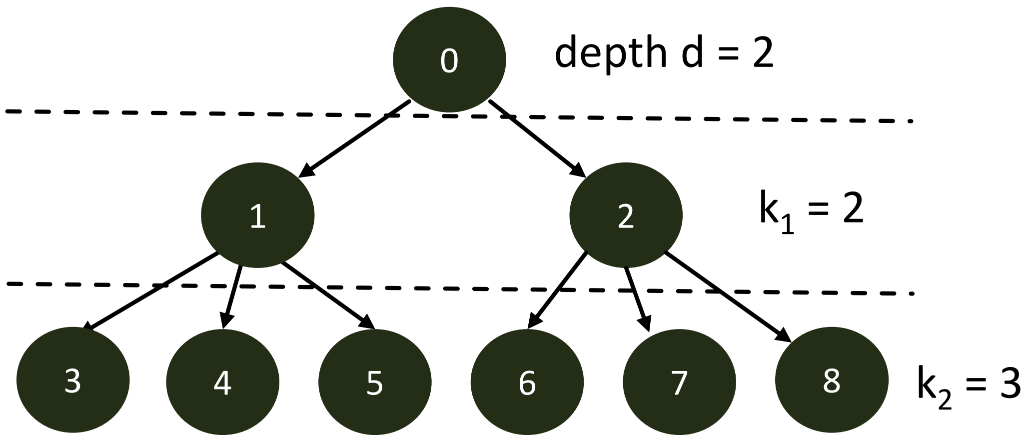
(c) SNC4 mode.

		Software NUMA		Software UMA		
		SNC4	SNC2	QUAD	HEM	A2A
Latency [ns] (Copy/BenchIT)	Local (L1)	3.8	3.8	3.8	3.8	3.8
	Tile (L2)	34 (M) 17 (E) 14 (S,F)	34 (M) 18 (E) 14 (S,F)	34 (M) 18 (E) 14 (S,F)	34 (M) 18 (E) 14 (S,F)	34 (M) 18 (E) 14 (S,F)
	Remote	107-122 (M) 98-114 (E) 96-118 (S,F)	111-125 (M) 104-117 (E) 104-118 (S,F)	119 (M) 116 (E) 107-117 (S,F)	120 (M) 116 (E) 107-117 (S,F)	122 (M) 116 (E) 109-117 (S,F)
Bandwidth [GB/s] (Read)		2.5	2.5	2.5	2.5	2.5
Bandwidth [GB/s] (Copy)	Tile	6.7 (M) 7.6 (E)	6.7 (M) 6.7 (E)	7.5 (M) 9.2 (E)	7.4 (M) 9.2 (E)	7.5 (M) 9.2 (E)
	Remote	7.7	6.7	7.5	7.5	7.5
Congestion (P2P pairs)		None				
Contention [ns] (1:N copy)	α	200	200	200	200	200
Linear, $\mathcal{T}_C(N) = \alpha + \beta \cdot N$	β	34	34	34	34	34

			Software NUMA		Software UMA		
			SNC4	SNC2	QUAD	HEM	A2A
Flat Mode	Latency [ns] (BenchIT)	DRAM	130-140	134-146	140	140	139
		MCDRAM	160-175	160-170	167	167	168
	Bandwidth [GB/s] (Copy NT / STREAM Copy)	DRAM	69 / 77	69 / 77	70 / 77	71 / 77	71 / 77
		MCDRAM	342 / 418	333 / 388	333 / 415	315 / 372	306 / 359
	Bandwidth [GB/s] (Read)	DRAM	71	71	77	77	77
		MCDRAM	243	288	314	314	314
	Bandwidth [GB/s] (Write)	DRAM	33	34	36	36	36
		MCDRAM	147	163	171	165	161
	Bandwidth [GB/s] (Triad NT / STREAM Triad)	DRAM	71 / 82	71 / 82	74 / 82	73 / 82	73 / 82
		MCDRAM	371 / 448	347 / 441	340 / 441	332 / 434	325 / 427
Cache Mode	Latency [ns] (BenchIT)		158-178	161-171	166	168	172
	Bandwidth [GB/s] (Copy NT / STREAM Copy)		150 / 252	130 / 252	175 / 255	134 / 237	132 / 233
	Bandwidth [GB/s] (Read)		87	95	124	128	118
	Bandwidth [GB/s] (Write)		56	56	72	72	68
	Bandwidth [GB/s] (Triad NT / STREAM Triad)		296 / 292	246 / 294	296 / 309	273 / 274	264 / 269

A principled approach to designing cache-to-cache broadcast algorithms

Multi-ary tree example



Tree depth

Level size

$$\mathcal{T}_{tree} = \sum_{i=1}^{d} \mathcal{T}_C(k_i) = \sum_{i=1}^{d} (c \cdot k_i + b)$$

Tree cost

$$= \sum_{i=1}^{d} (R_R + R_L + c \cdot (k_i - 1))$$

$$\mathcal{T}_{sbcast} = \min_{d,k_i} \left(\mathcal{T}_{fw} + \sum_{i=1}^{d} (c \cdot k_i + b) + \sum_{i=1}^{d} \mathcal{T}_{nb}(k_i + 1) \right)$$

Reached threads

$$N \leq 1 + \sum_{i=1}^{d} \prod_{j=1}^{i} k_j, \quad \forall i < j, k_i \leq k_j$$

S. Ramos, TH: *“Modeling Communication in Cache-Coherent SMP Systems - A Case-Study with Xeon Phi “, ACM HPDC’13*

S. Ramos, TH: *“Cache line aware optimizations for ccNUMA systems (IEEE TPDS’17).*

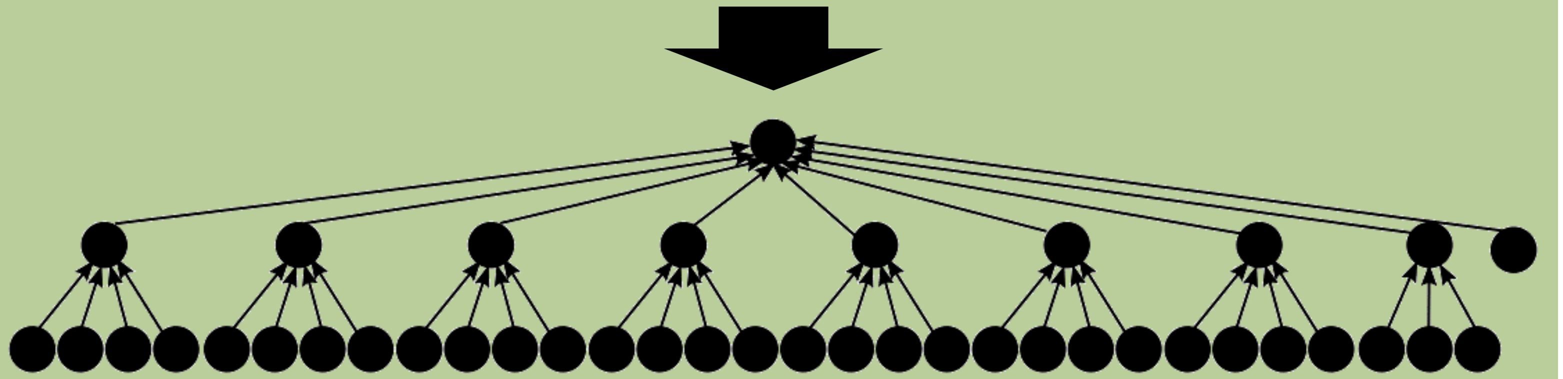
Model-driven performance engineering for broadcast

Binary or binomial trees?

Oh! But sure I can do that.

$$\mathcal{T}_{sbcast} = \min_{d,k_i}\left(\mathcal{T}_{fw} + \sum_{i=1}^{d}(c \cdot k_i + b) + \sum_{i=1}^{d}\mathcal{T}_{nb}(k_i + 1)\right)$$

$$N \leq 1 + \sum_{i=1}^{d}\prod_{j=1}^{i} k_j, \quad \forall i < j, k_i \leq k_j$$



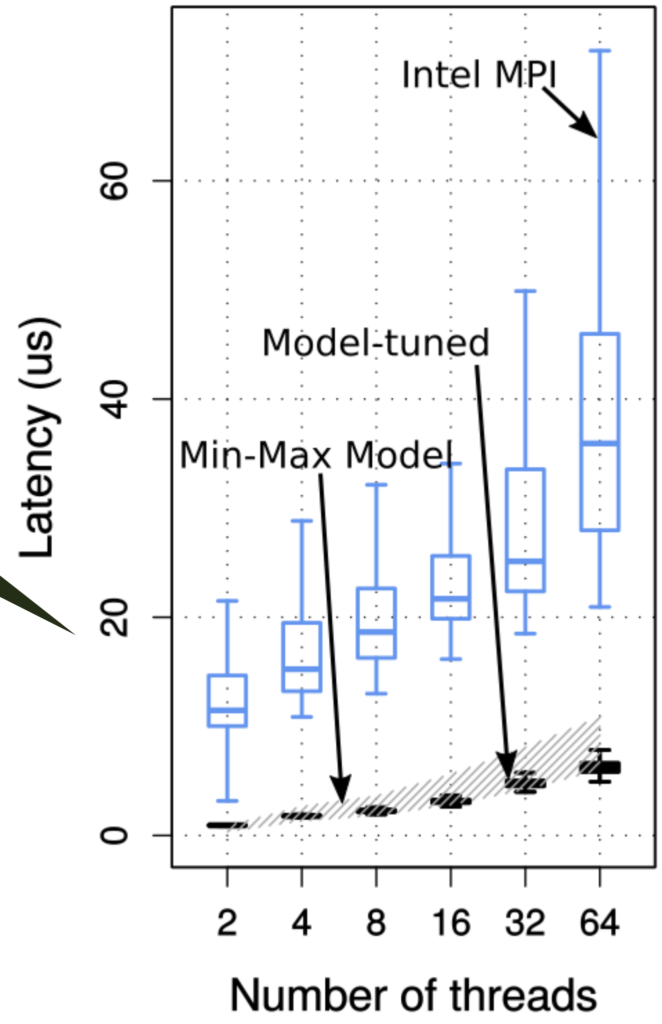
Model-driven performance engineering for broadcast

Binary or binomial trees?

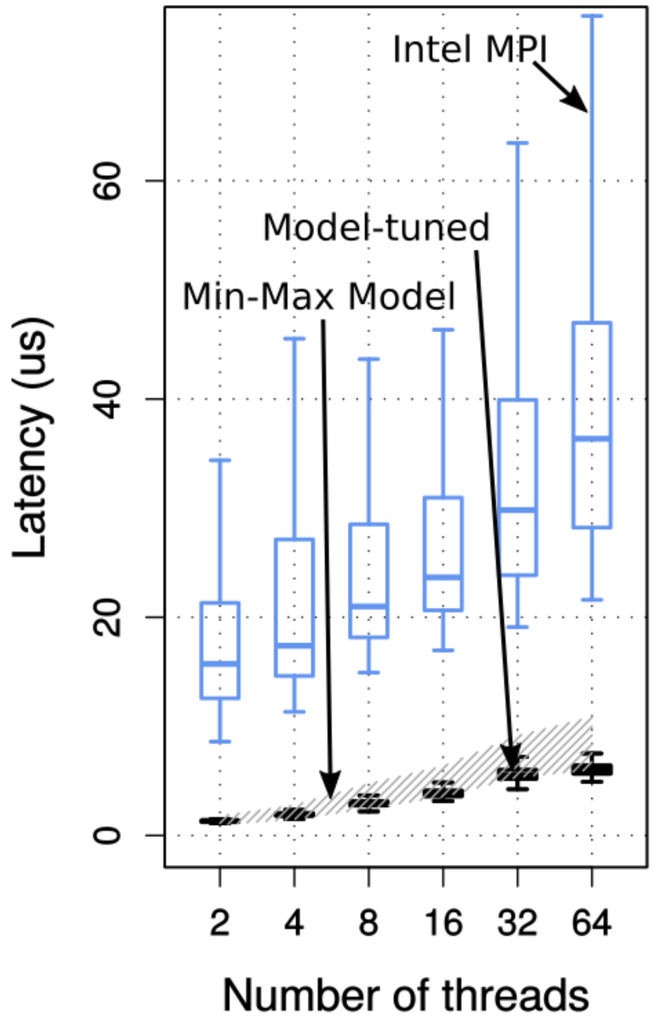
13x faster, lower variance

Oh! But sure I can do that.

<title>code ninja</title>



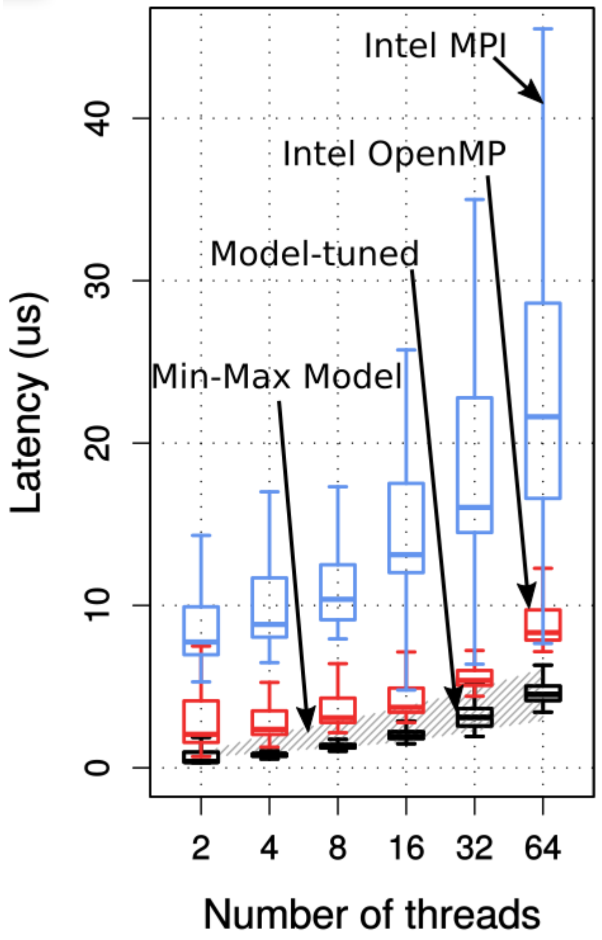
(a) Filling Tiles.



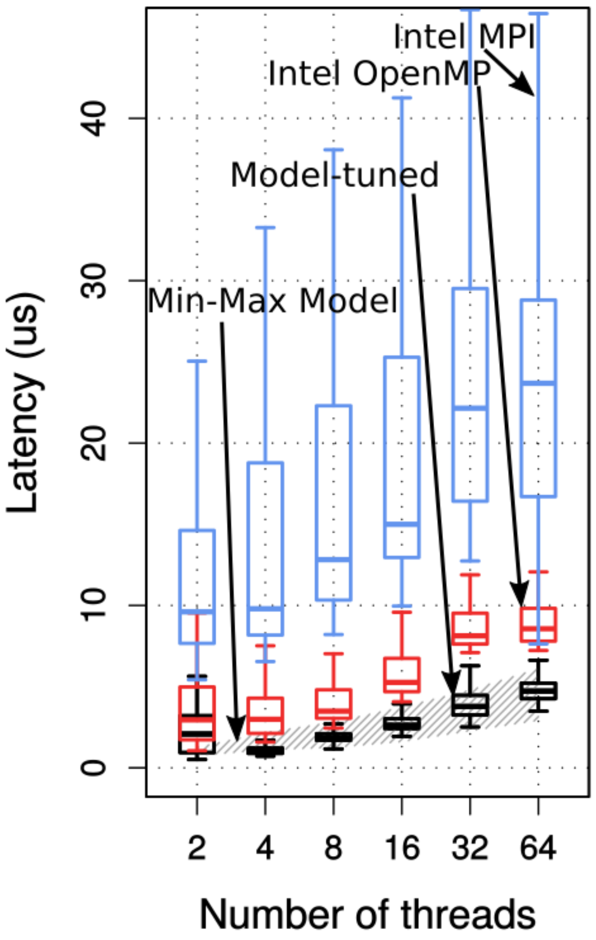
(b) Scatter.

S. Ramos and T. Hoefler: Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL, IPDPS'17

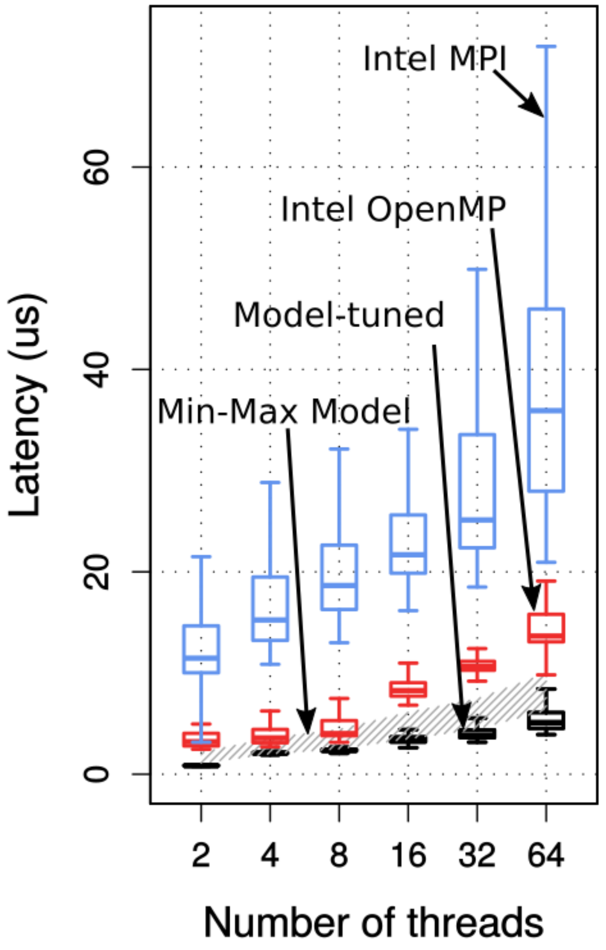
Easy to generalize to similar algorithms



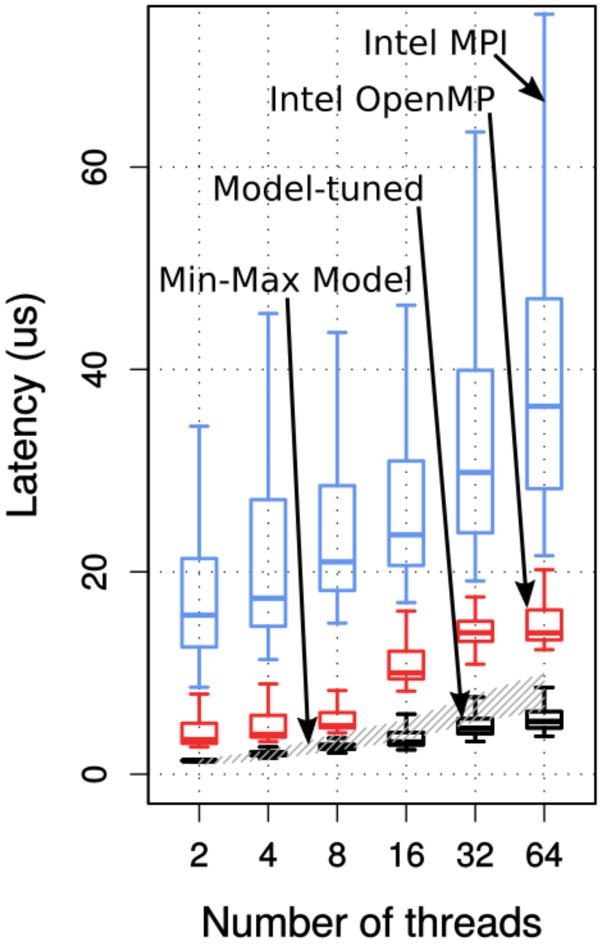
(a) Filling Tiles.



(b) Scatter.



(a) Filling Tiles.



(b) Scatter.

Barrier (7x faster than OpenMP)

Reduce (5x faster then OpenMP)

S. Ramos and T. Hoefler: Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL, IPDPS'17

What about real applications?

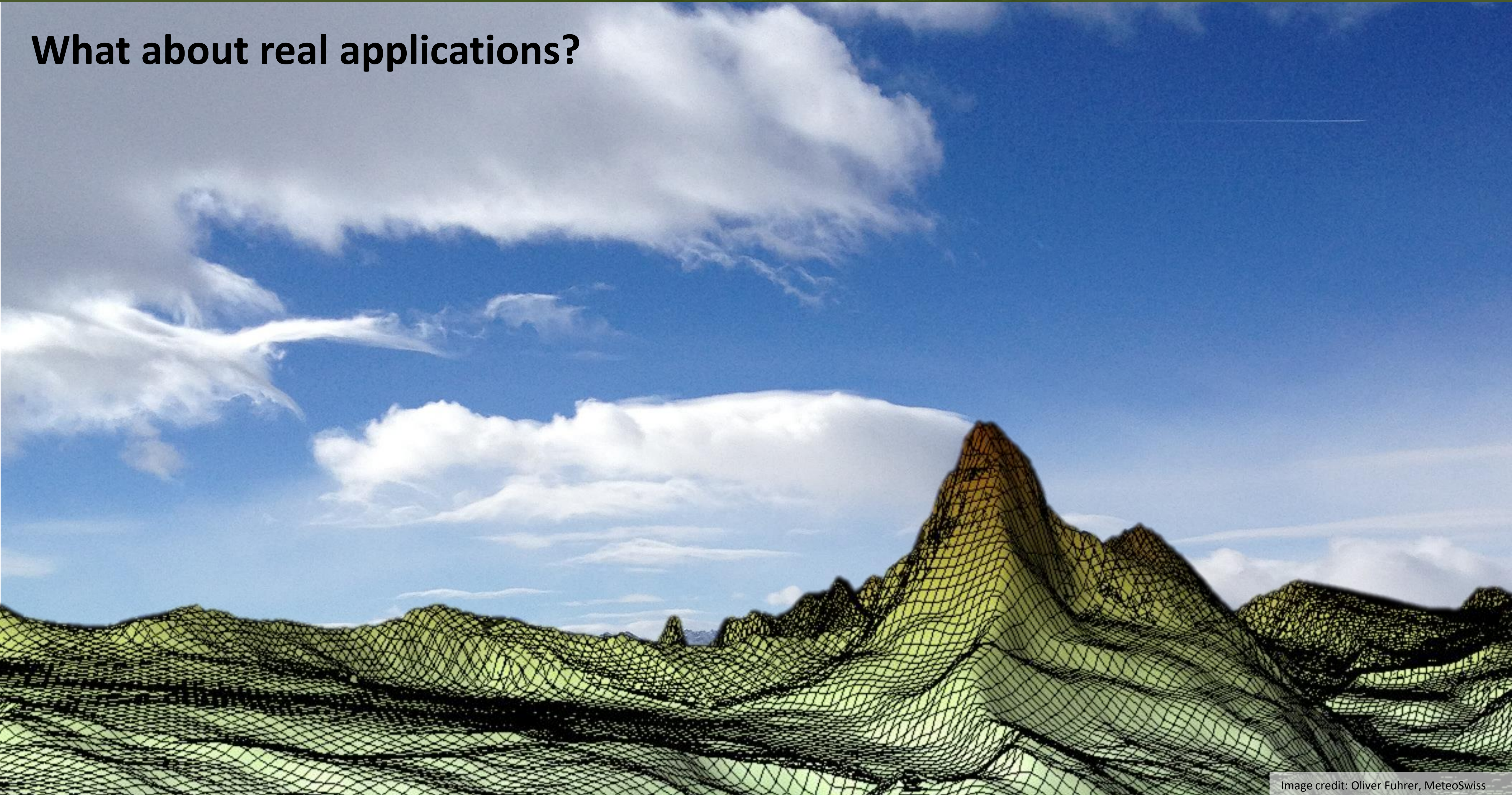


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

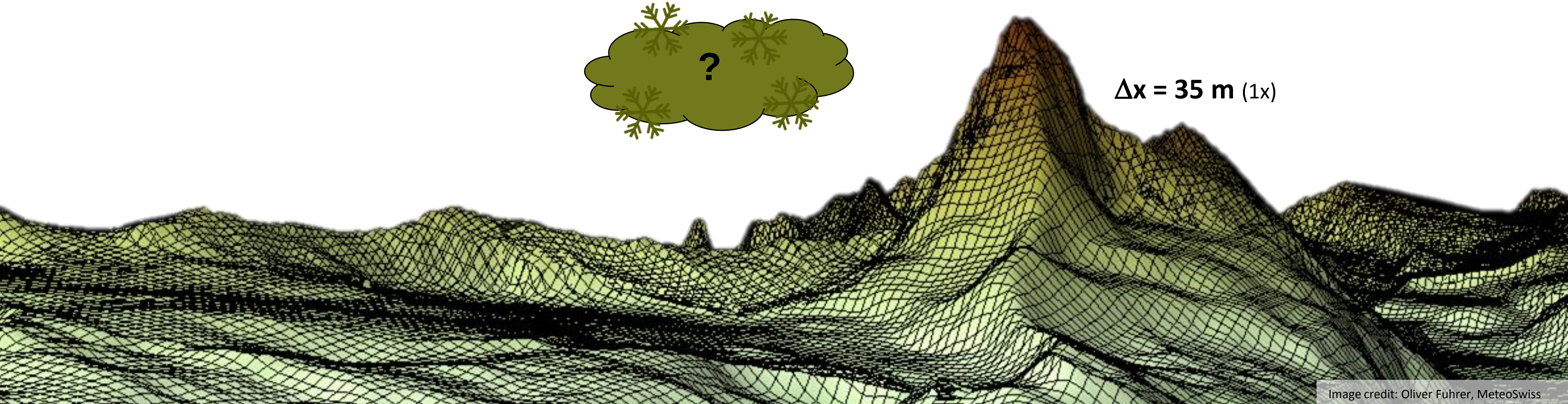


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

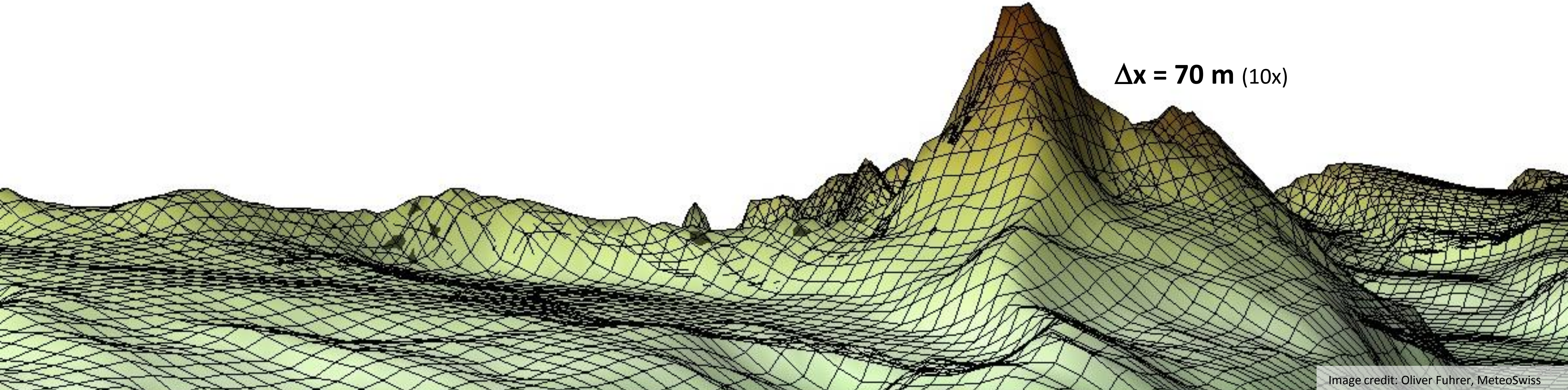


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

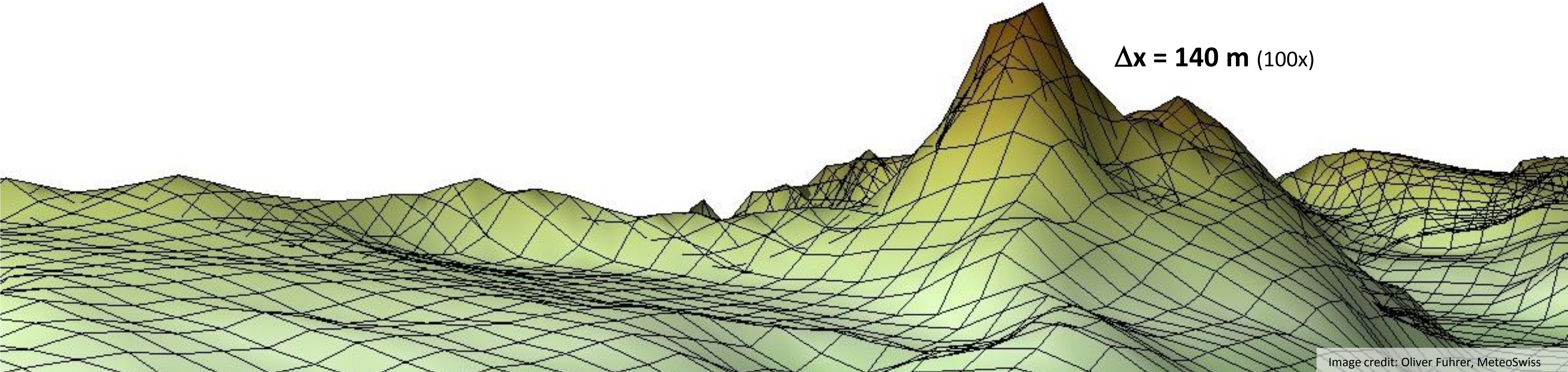


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

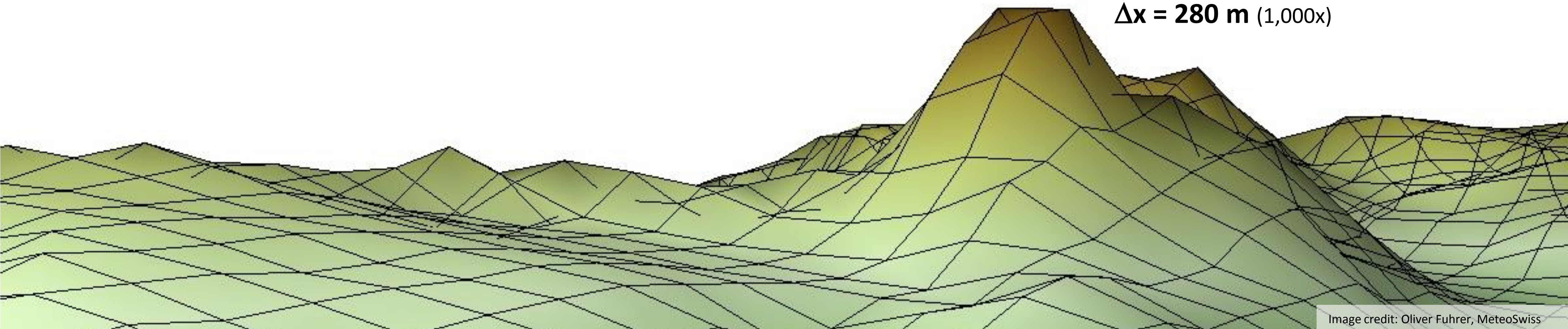


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

Δx = 550 m (10,000x)

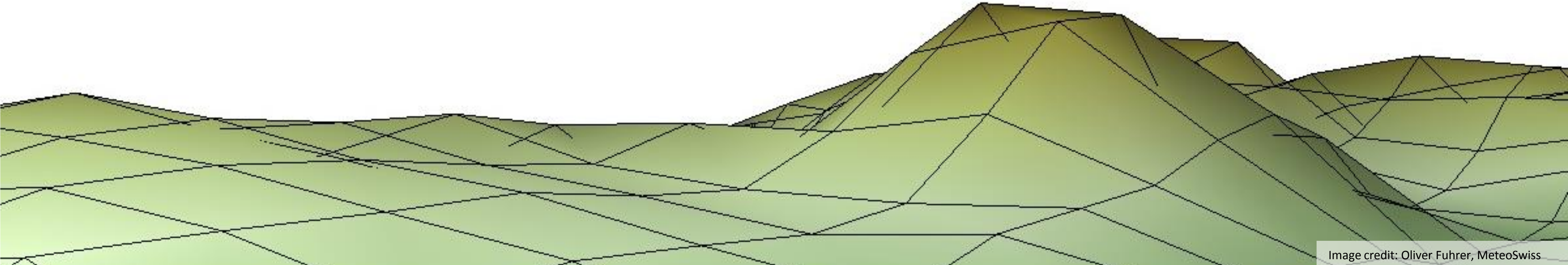


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

Operational model of MeteoSwiss today!

Δx = 1100 m (100,000x)

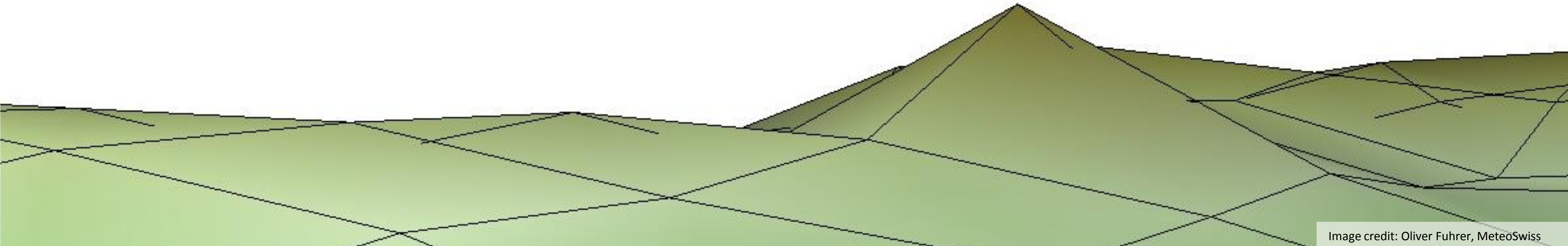
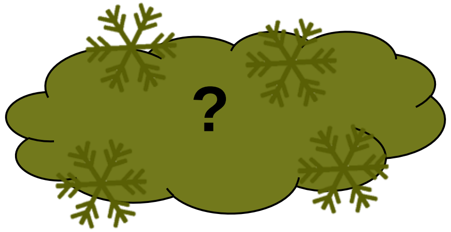


Image credit: Oliver Fuhrer, MeteoSwiss

A factor **2x** in resolution roughly corresponds to a factor **10x** compute

Operational model of MeteoSwiss before 2016!



Δx = 2200 m (1,000,000x)

We’re a factor of **100,000** away!

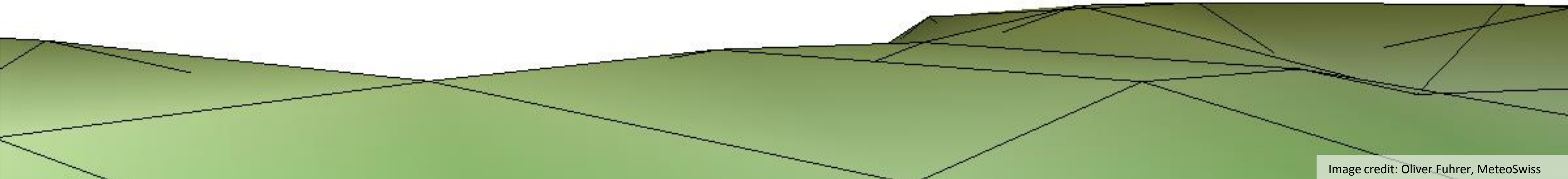
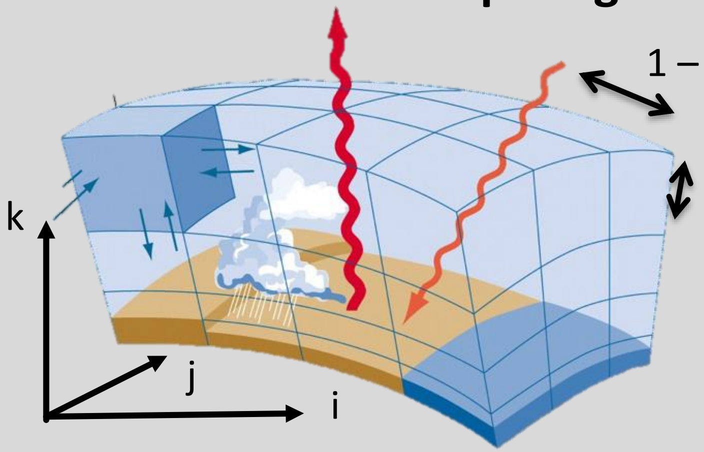


Image credit: Oliver Fuhrer, MeteoSwiss

Basic Atmospheric Equations

Wind	$\rho \frac{d\mathbf{v}}{dt} = -\nabla p + \rho\mathbf{g} - 2\mathbf{\Omega} \times (\rho\mathbf{v}) - \nabla \cdot (\underline{\mathbf{T}})$
Pressure	$\frac{dp}{dt} = -(c_{pd}/c_{vd}) p \nabla \cdot \mathbf{v} + (c_{pd}/c_{vd} - 1) Q_h$
Temperature	$\rho c_{pd} \frac{dT}{dt} = \frac{dp}{dt} + Q_h$
Water	$\rho \frac{dq^v}{dt} = -\nabla \cdot \mathbf{F}^v - (I^l + I^f)$
	$\rho \frac{dq^{l,f}}{dt} = -\nabla \cdot (\mathbf{P}^{l,f} + \mathbf{F}^{l,f}) + I^{l,f}$
Density	$\rho = p\{R_d(1 + (R_v/R_d - 1)q^v - q^l - q^f)T\}^{-1}$

Discretized on a compute grid

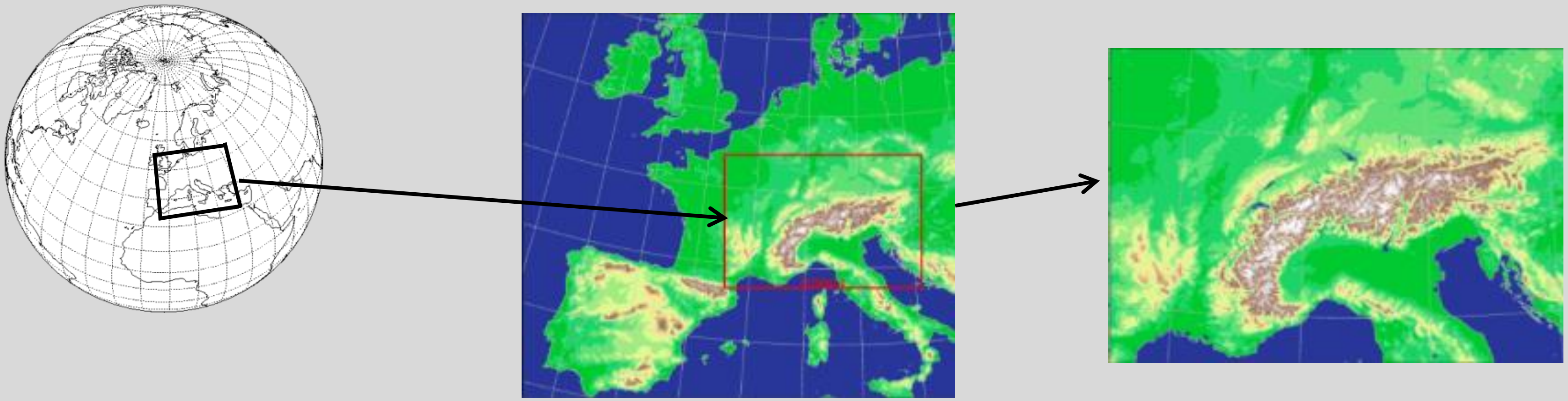



```
u(i,j,k),
v(i,j,k), w(i,j,k),
p(i,j,k), t(i,j,k),
qv(i,j,k), qc(i,j,k),
qi(i,j,k), rho(i,j,k)
```

ECMWF-Model
16 km Grid
2 x per day
10 days prediction

COSMO-7
6.6 km Grid
3 x per day
72 h prediction

COSMO-1
1.1 km Grid
7 x pro day 33 h prediction
1 x pro day 45 h prediction

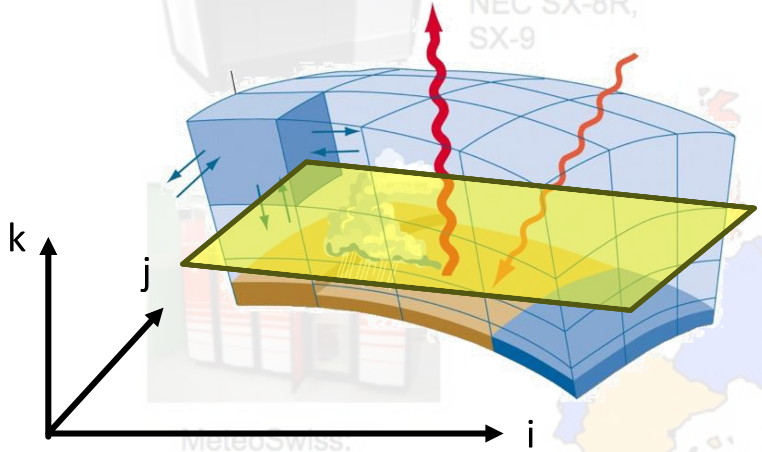


Slide adopted from T. Schulthess

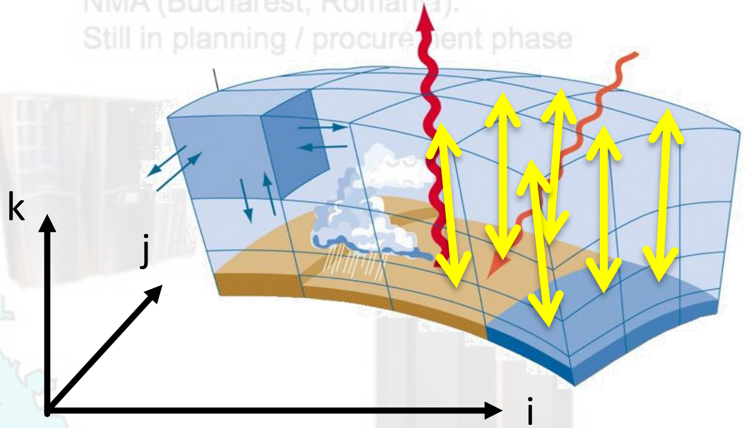
The COSMO Code – 300k SLOC Fortran

Two main algorithmic motifs in dynamical core

Finite Difference Stencils



Tridiagonal Solvers




```
do j = 1, niter
  do i = 1, nwork
    c(i) = a(i) + b(i) * ( a(i+1) - 2.0d0*a(i) + a(i-1) )
  end do
end do
```

Stencil computations (oh no, another stencil talk)

Motivation:

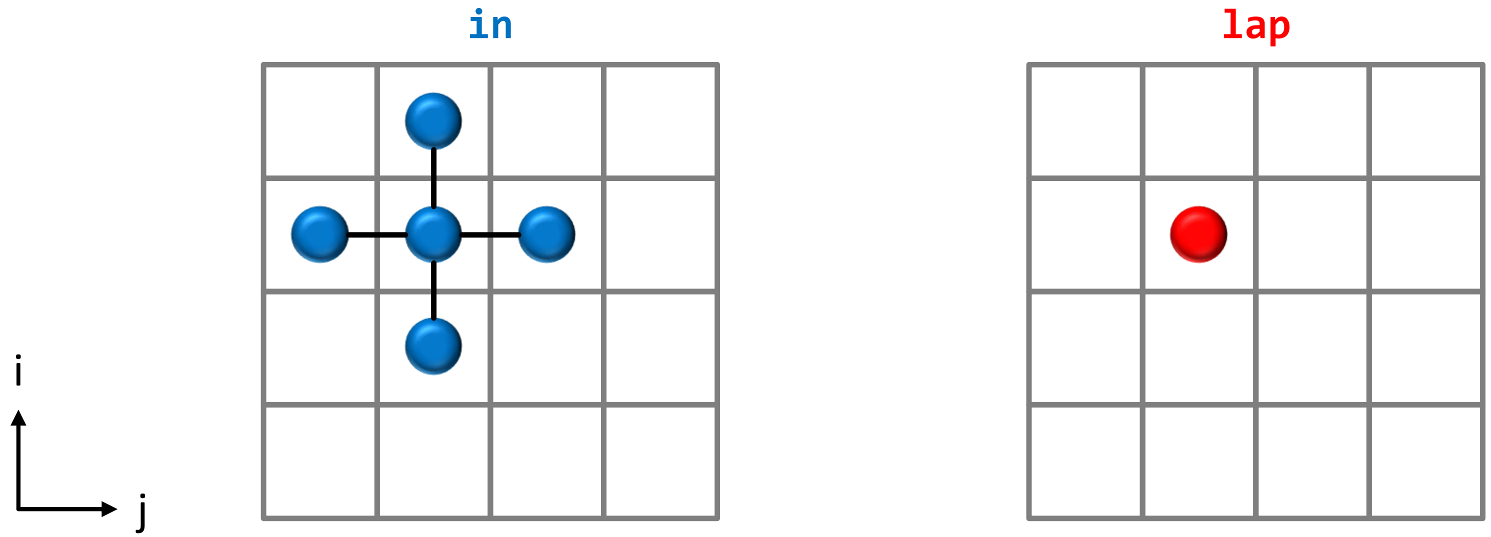
- **Important algorithmic motif (e.g., finite difference method)**

due to the typically low arithmetic intensity stencil computations are often memory bandwidth limited!

Definition:

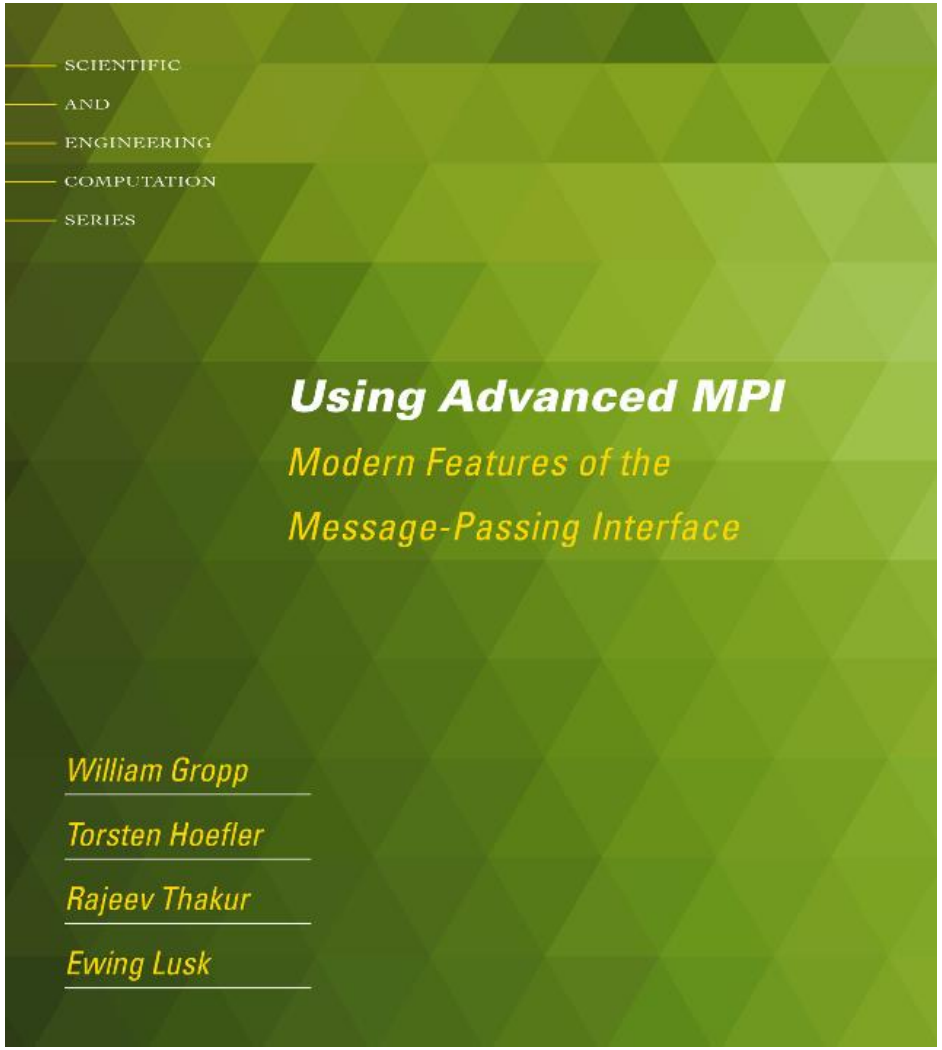
- **Element-wise computation on a regular grid using a fixed neighborhood**
- **Typically working on multiple input fields and writing a single output field**

```
lap(i,j) = -4.0 * in(i,j) + in(i-1,j) + in(i+1,j) + in(i,j-1) + in(i,j+1)
```



How to tune such stencils (most other stencil talks)

- **LOTS of related work!**
  - Compiler-based (e.g., Polyhedral such as PLUTO [1])
  - Auto-tuning (e.g., PATUS [2])
  - Manual model-based tuning (e.g., Datta et al. [3])
  - … essentially every micro-benchmark or tutorial, e.g.:
- **Common features**
  - Vectorization tricks (data layout)
  - Advanced communication (e.g., MPI neighbor colls)
  - Tiling in time, space (diamond etc.)
  - Pipelining
- **Much of that work DOES NOT compose well with complex <u>stencil programs</u> in weather/climate**



[1]: Uday Bondhugula, A. Hartono, J. Ramanujan, P. Sadayappan. *A Practical Automatic Polyhedral Parallelizer and Locality Optimizer*, PLDI'08
[2]: Matthias Christen, et al.: *PATUS: A Code Generation and Autotuning Framework for Parallel Iterative Stencil Computations …, IPDPS'11*
[3]: Kaushik Datta, et al., *Optimization and Performance Modeling of Stencil Computations on Modern Microprocessors,* SIAM review

What is a "complex stencil program"? (this stencil talk)

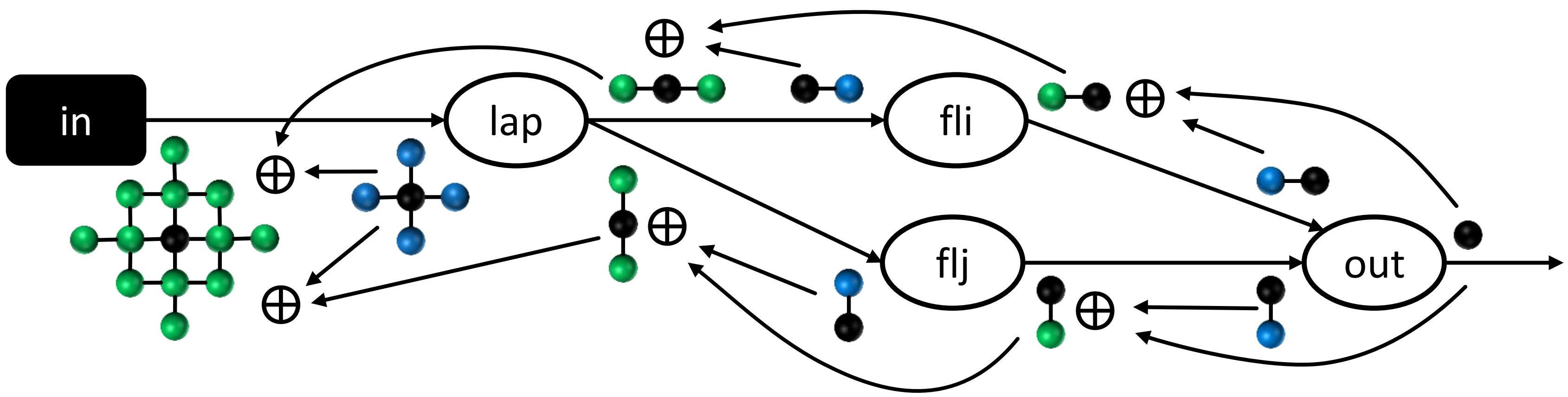
E.g., the COSMO weather code

- **is a regional climate model used by 7 national weather services**
- **contains hundreds of different complex stencils**

Modeling stencils formally:

- **Represent stencils as DAGs**
  - Model stencil as nodes, data dependencies as edges

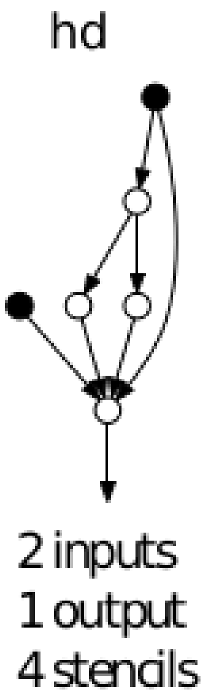
simplified horizontal diffusion example



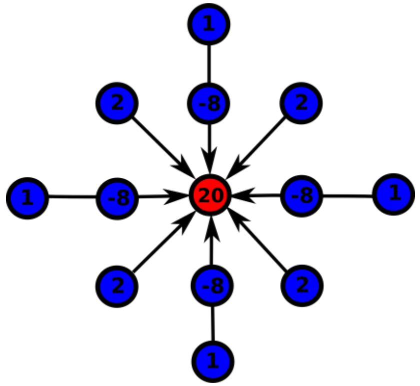
$$a \oplus b = \{a' + b' \mid a' \in a, b' \in b\}$$

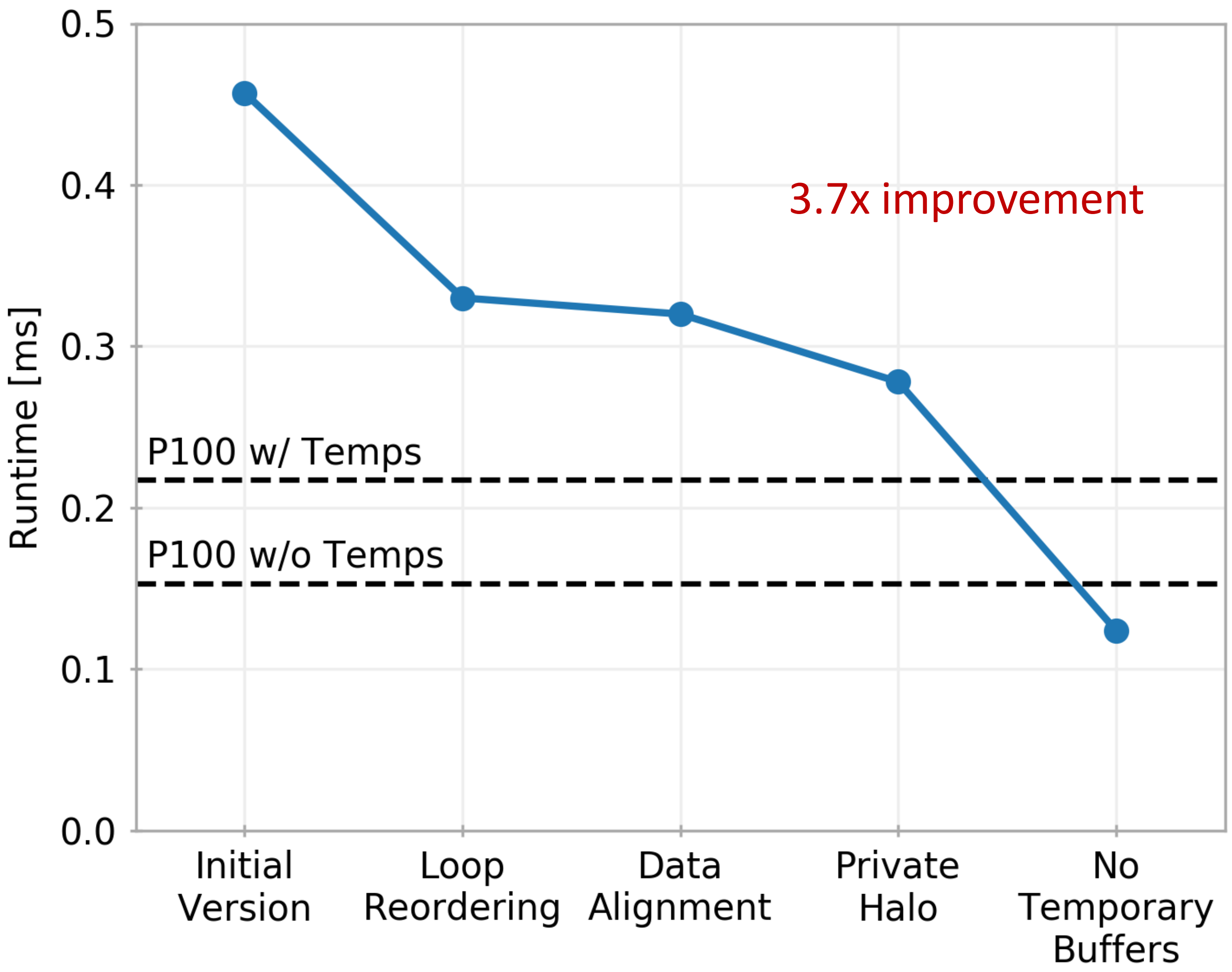
T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS'15

Horizontal Diffusion Stencil Program tuned to Xeon Phi KNL



4th order horizontal diffusion: $-\alpha\nabla^2(\nabla^2 u)$



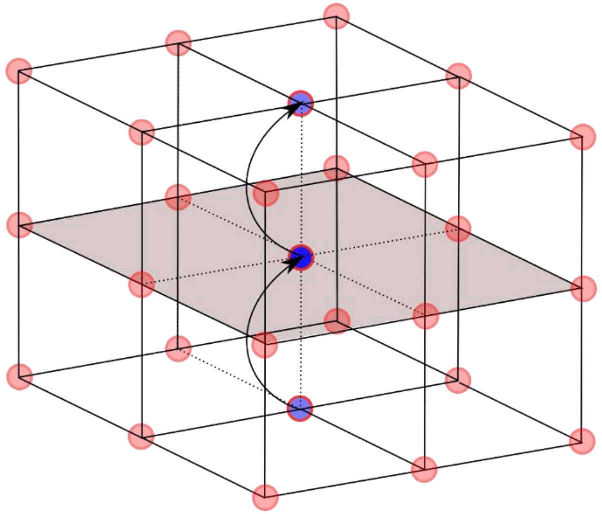


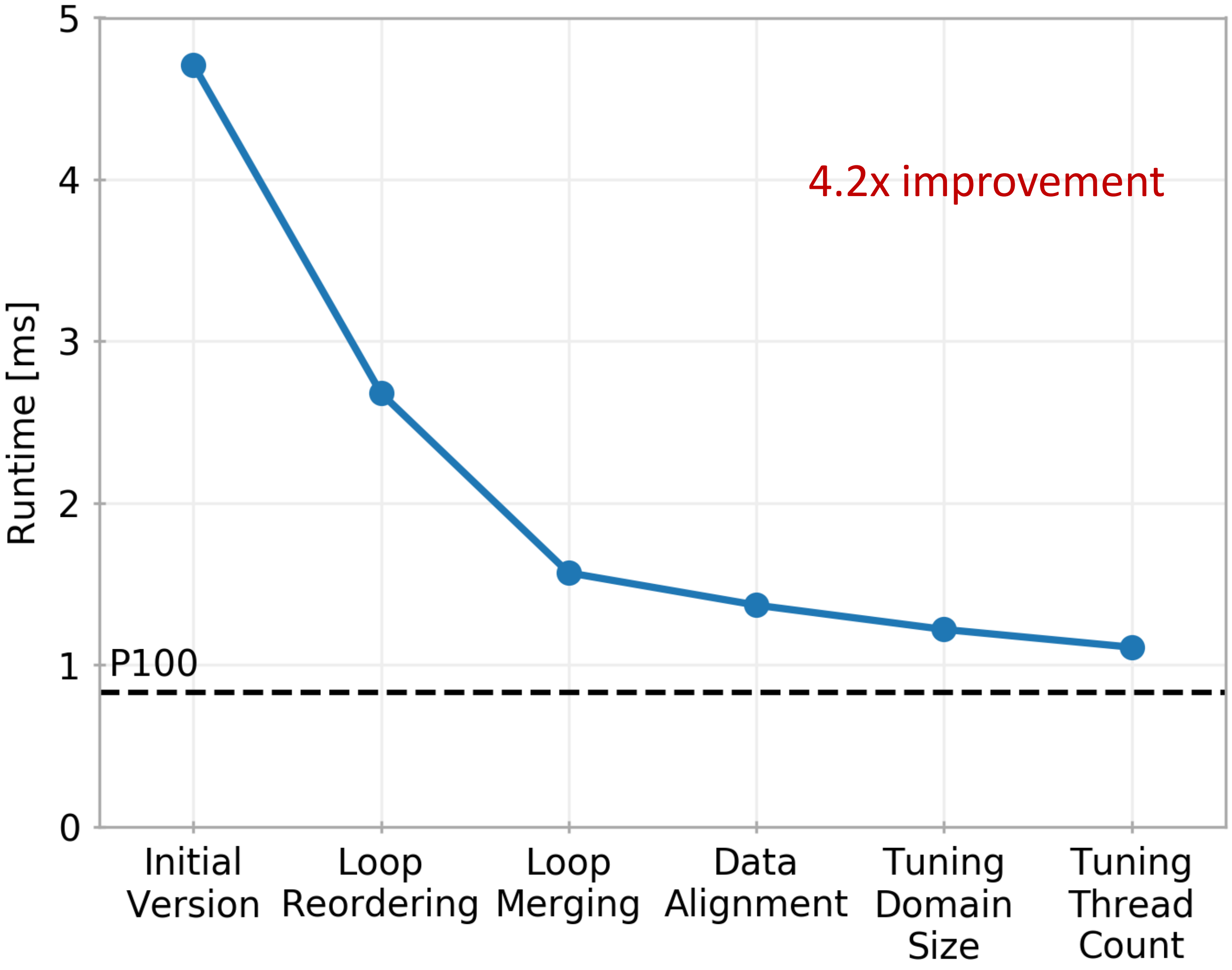
Work performed at the Intel Parallel Computing Center at ETH Zurich

Vertical Advection Stencil Program tuned to Xeon Phi KNL

Vertical stencil-like operations, derived from
Iterations of tridiagonal solvers (Thomas algorithm)

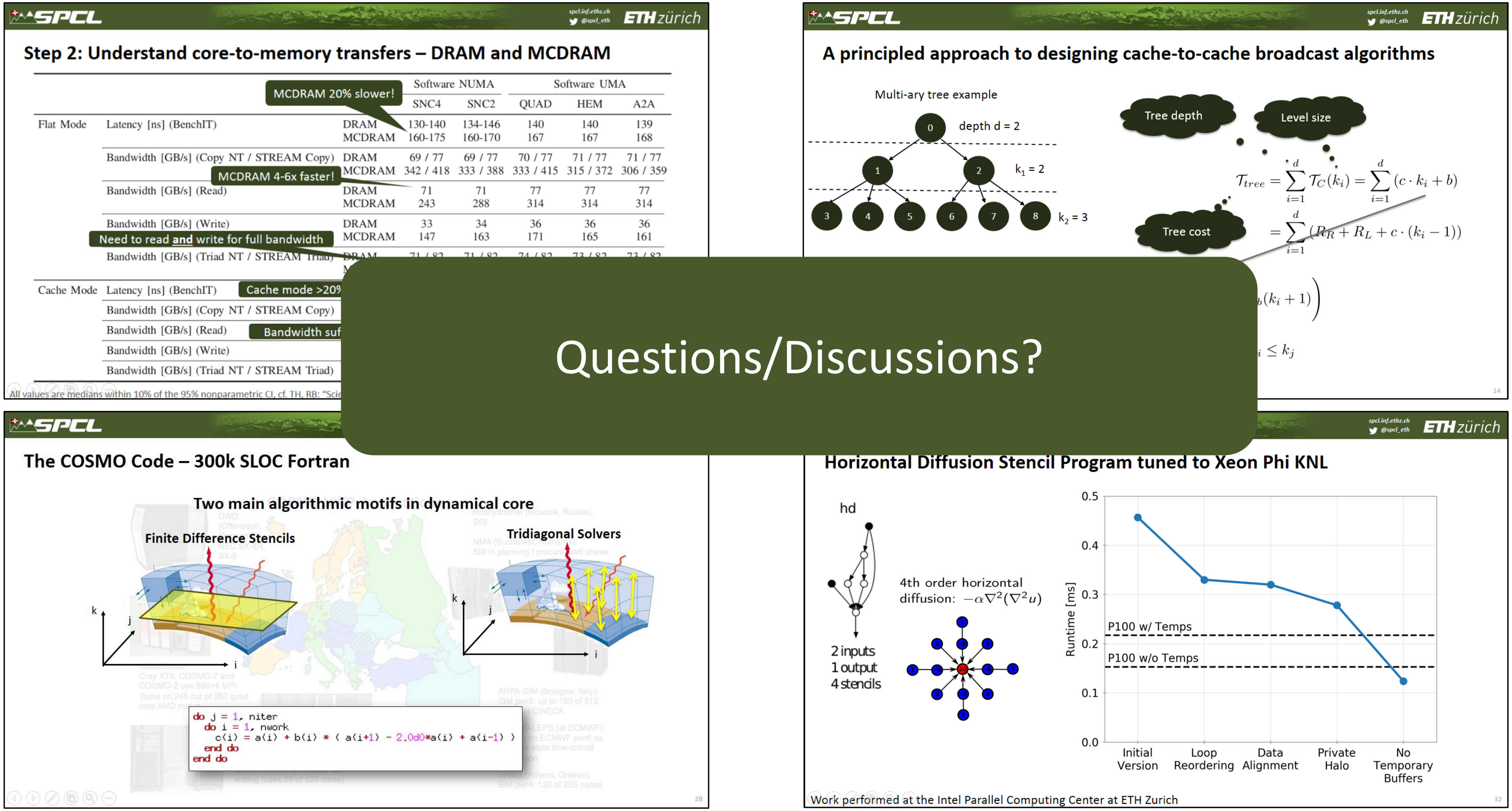
$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & c_{n-1} \\ & & \ddots & a_n & b_n \\ 0 & & & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}$$





Work performed at the Intel Parallel Computing Center at ETH Zurich

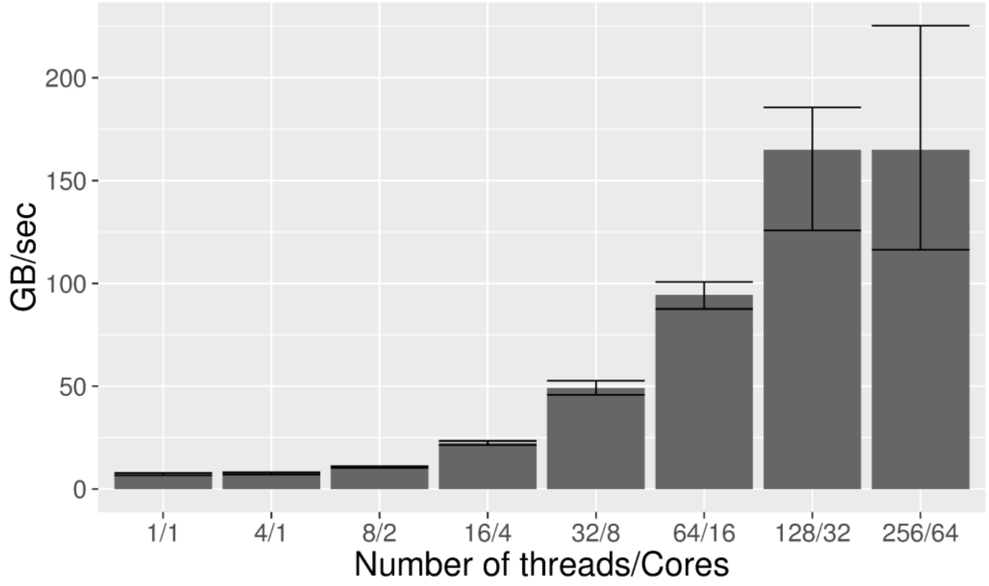
Scientific performance engineering for complex memory systems



Backup

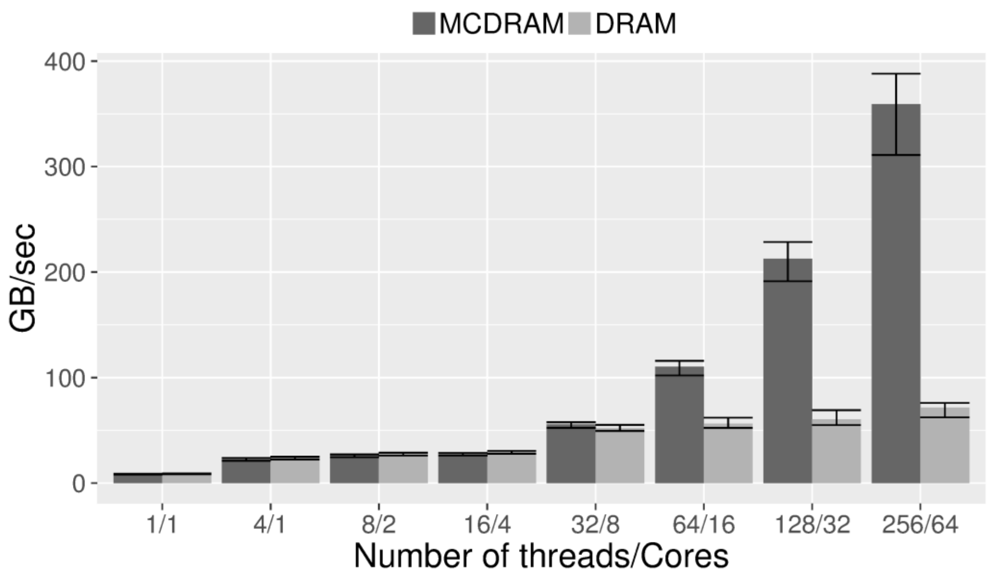
Sorting in complex memories

Triad
Cache Mode
SNC4



Check! Let's do something "Big Data"!

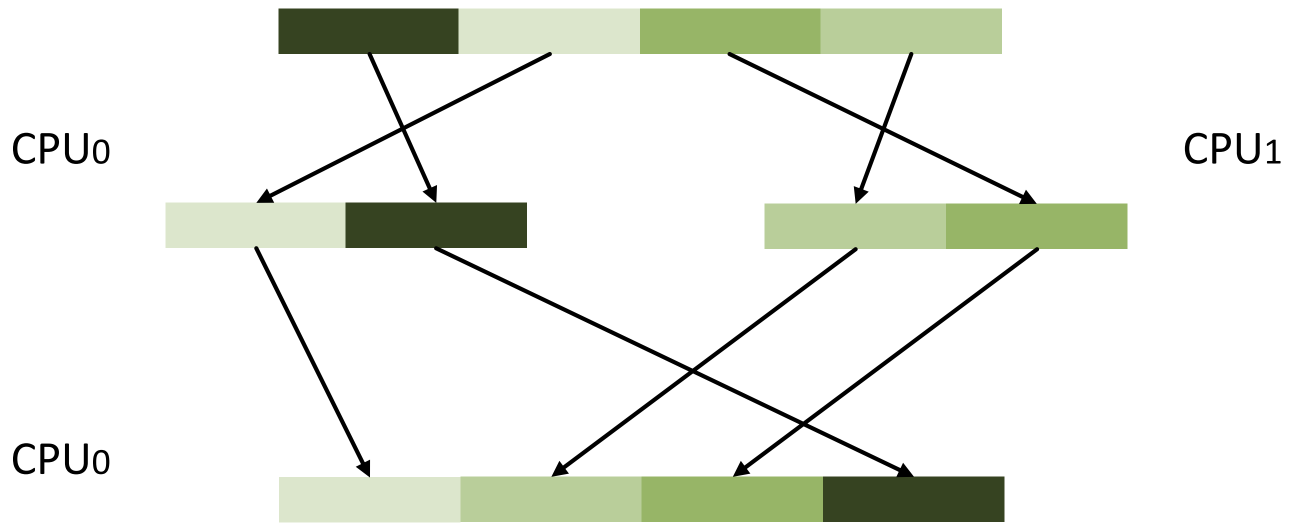
Triad
Flat Mode
SNC4



We'll need a parallel sort on all cores, right?

<title>code ninja</title>

Memory model: Bitonic Mergesort



- Slices of 16 elements go through a bitonic network.
- Communication: CPU0 accesses data from local and remote caches.
- Synchronization: CPU0 waits for CPU1.
- Memory accesses: latency vs. bandwidth.

Modeling Bitonic Sorting

L1 access cost

memory access cost

$$C_{L1}(n) = [\log_2(n) - 1]2n \cdot \mathrm{cost}_{L1} + 2n \cdot \mathrm{cost}_{mem}$$

elements in L1

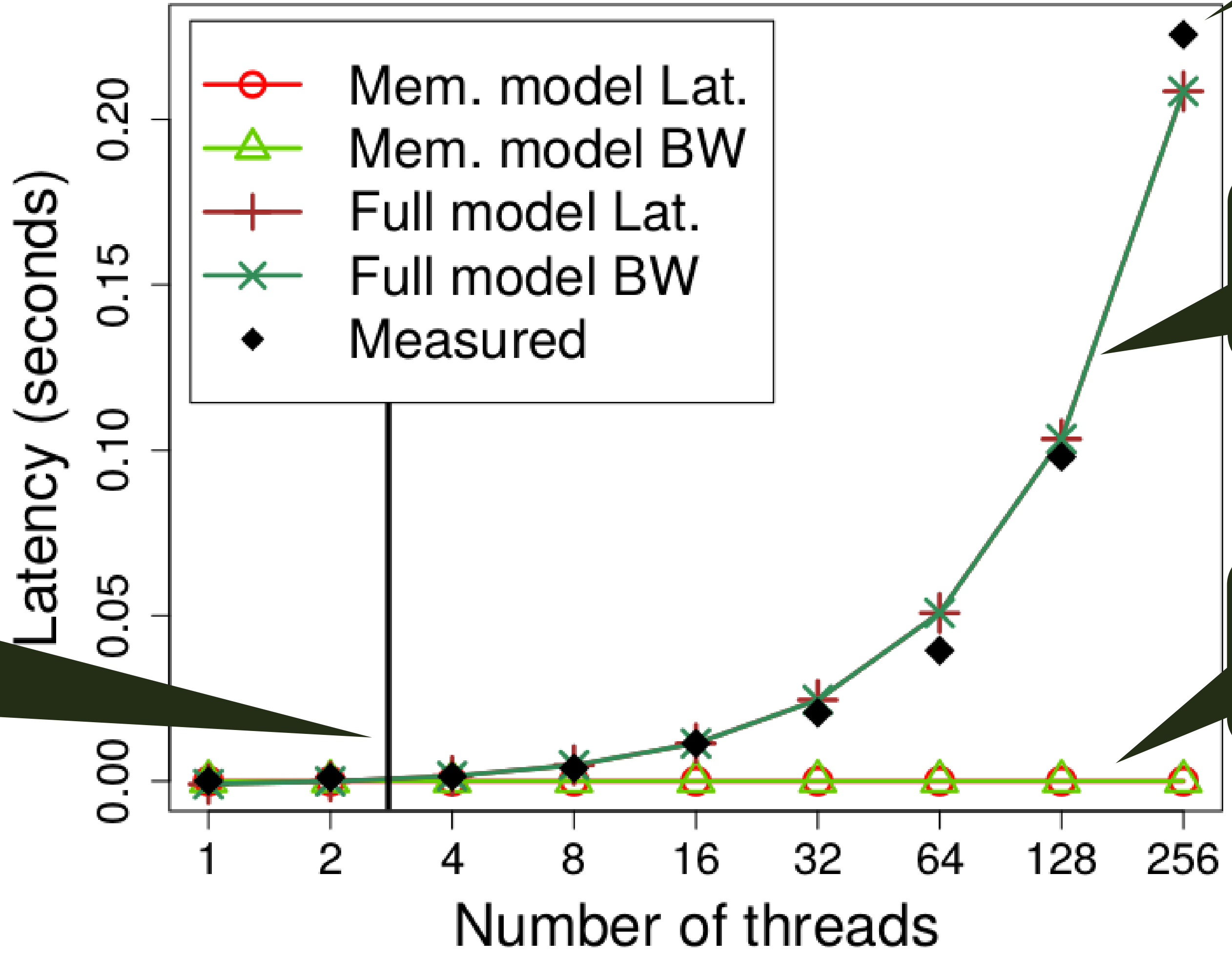
L2 access cost

$$C_{L2}(n) = \frac{n}{n_{L1}} C_{L1}(n_{L1}) + \\ + [\log_2(n) - \log_2(n_{L1})]2n \cdot \mathrm{cost}_{L2}$$

elements in L2

$$C_{mem}(n) = \frac{n}{n_{L2}} C_{L2}(n_{L2}) + \\ + [\log_2(n) - \log_2(n_{L2})]2n \cdot \mathrm{cost}_{mem}$$

Bitonic Sort of 1 kiB



(a) Sorting 1 KB of integers.

measurement

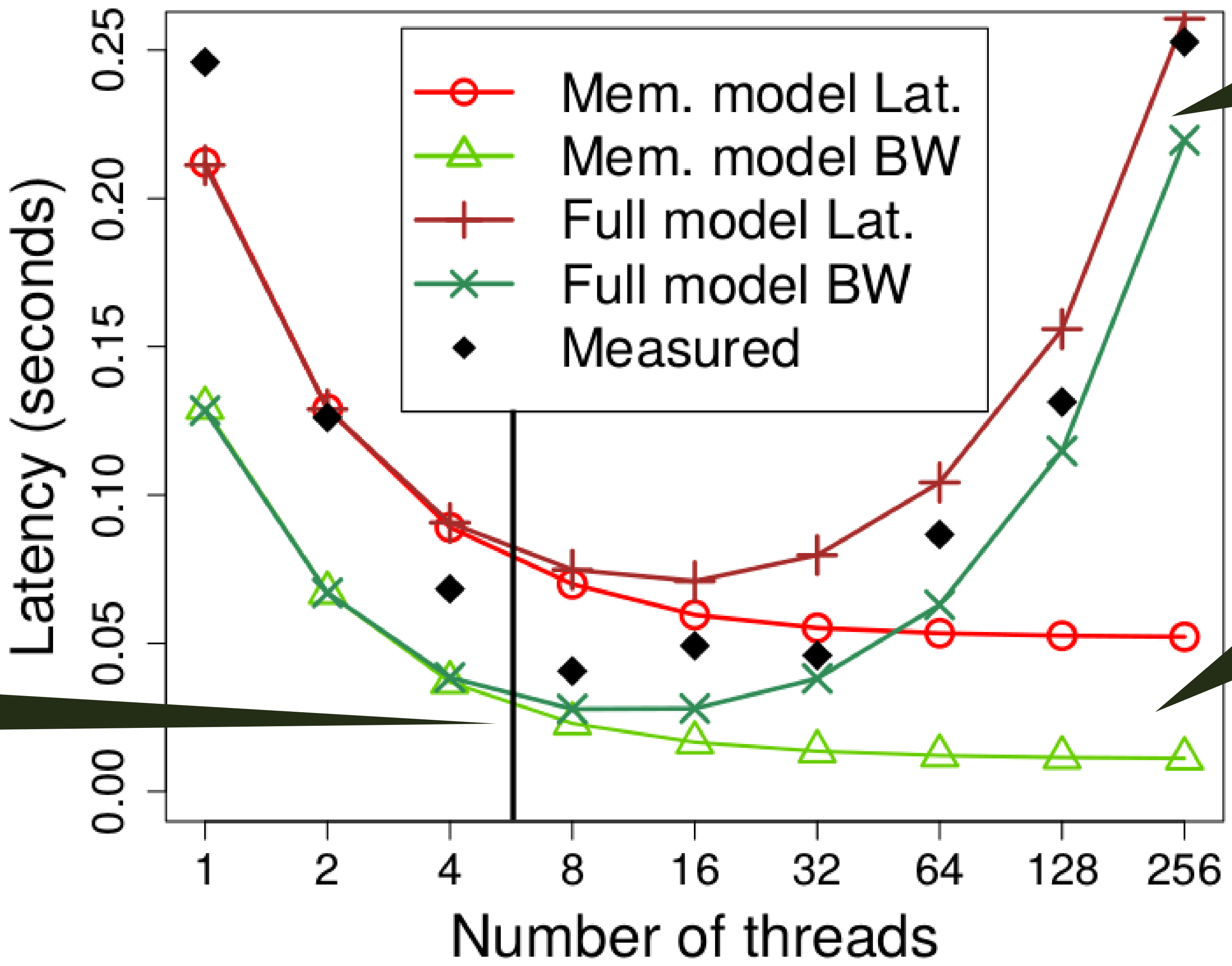
model including synchronization cost

model (just) memory costs

Synchronization outweights memory costs for small data!

So don't parallelize too much!

Bitonic Sort of 4 MiB

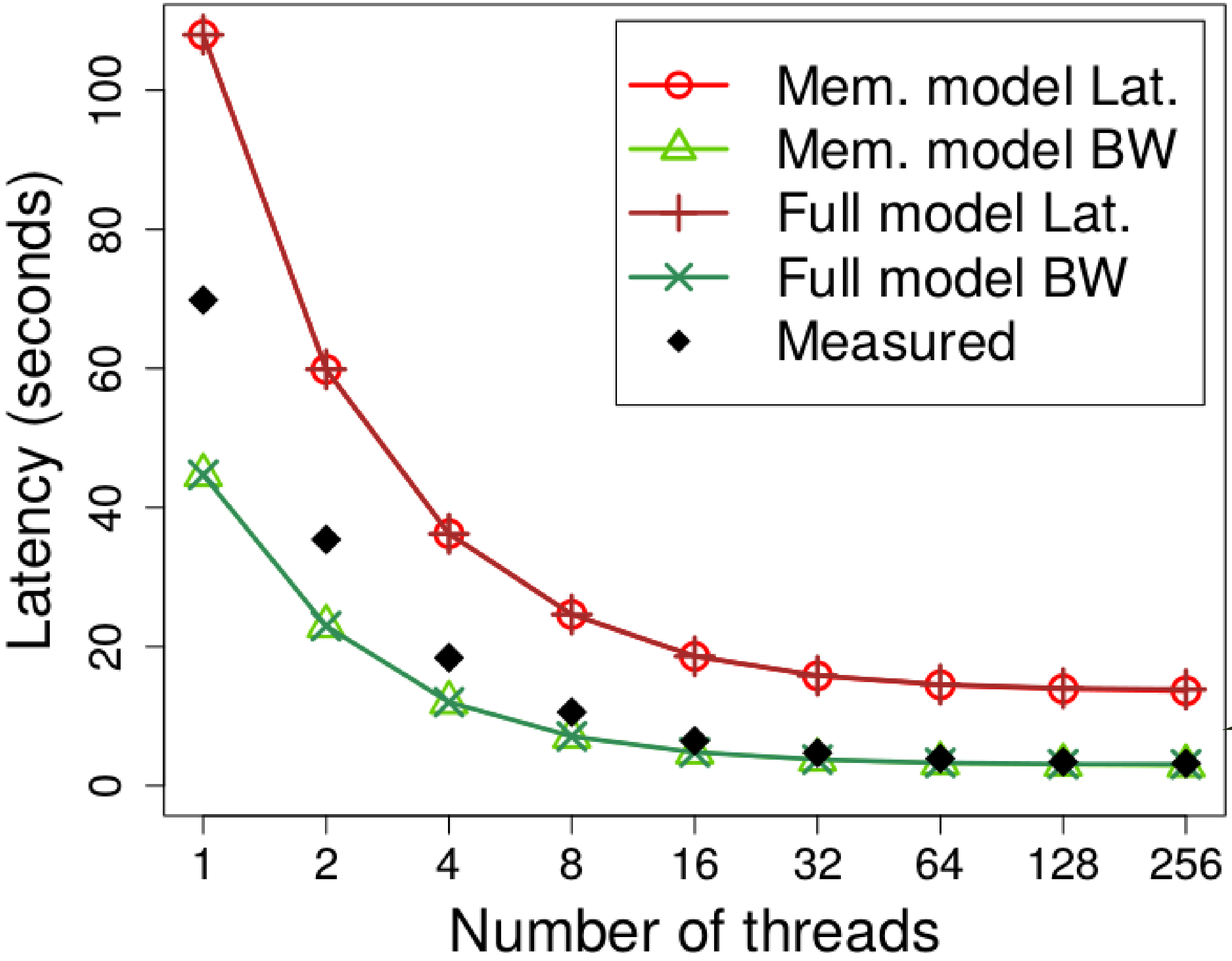


model including synchronization cost

model (just) memory costs

"parallelization boundary"

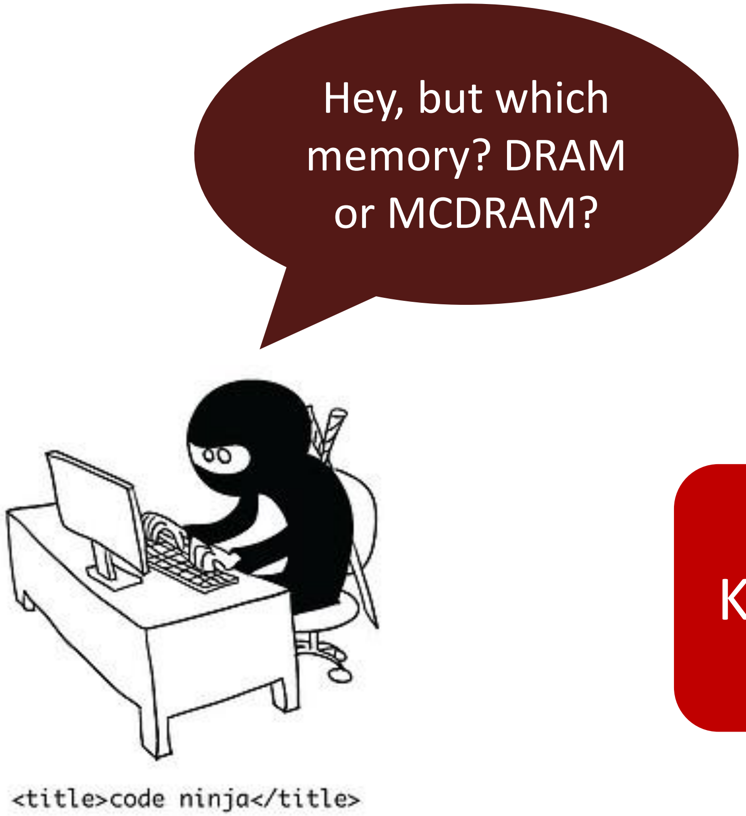
Bitonic Sort of 1 GiB



Always use all cores!

synchronization negligible

The most surprising result last …



The model (and practical measurements) indicate that it does not matter.

Thesis: the higher bandwidth of MCDRAM did not help due to the higher latency ($\log^2 n$ depth).

Disclaimer: this is NOT the best sorting algorithm for Xeon Phi KNL. It is the best we found with limited effort. We suspect that a combination of algorithms will perform best.