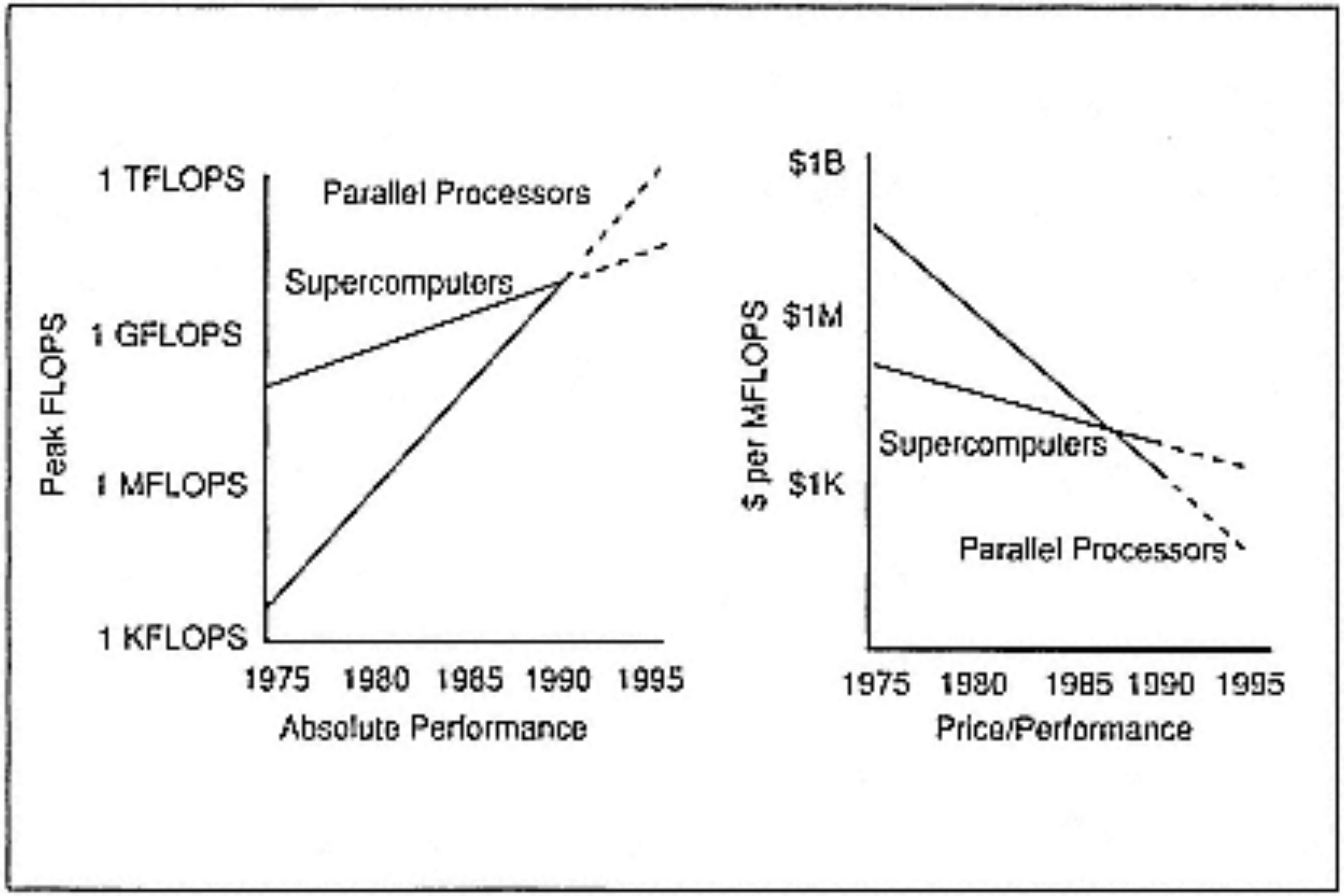
ETH zürich

Dusk of Moore's Law: opportunities for weather and climate modelling?

Thomas C. Schulthess

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

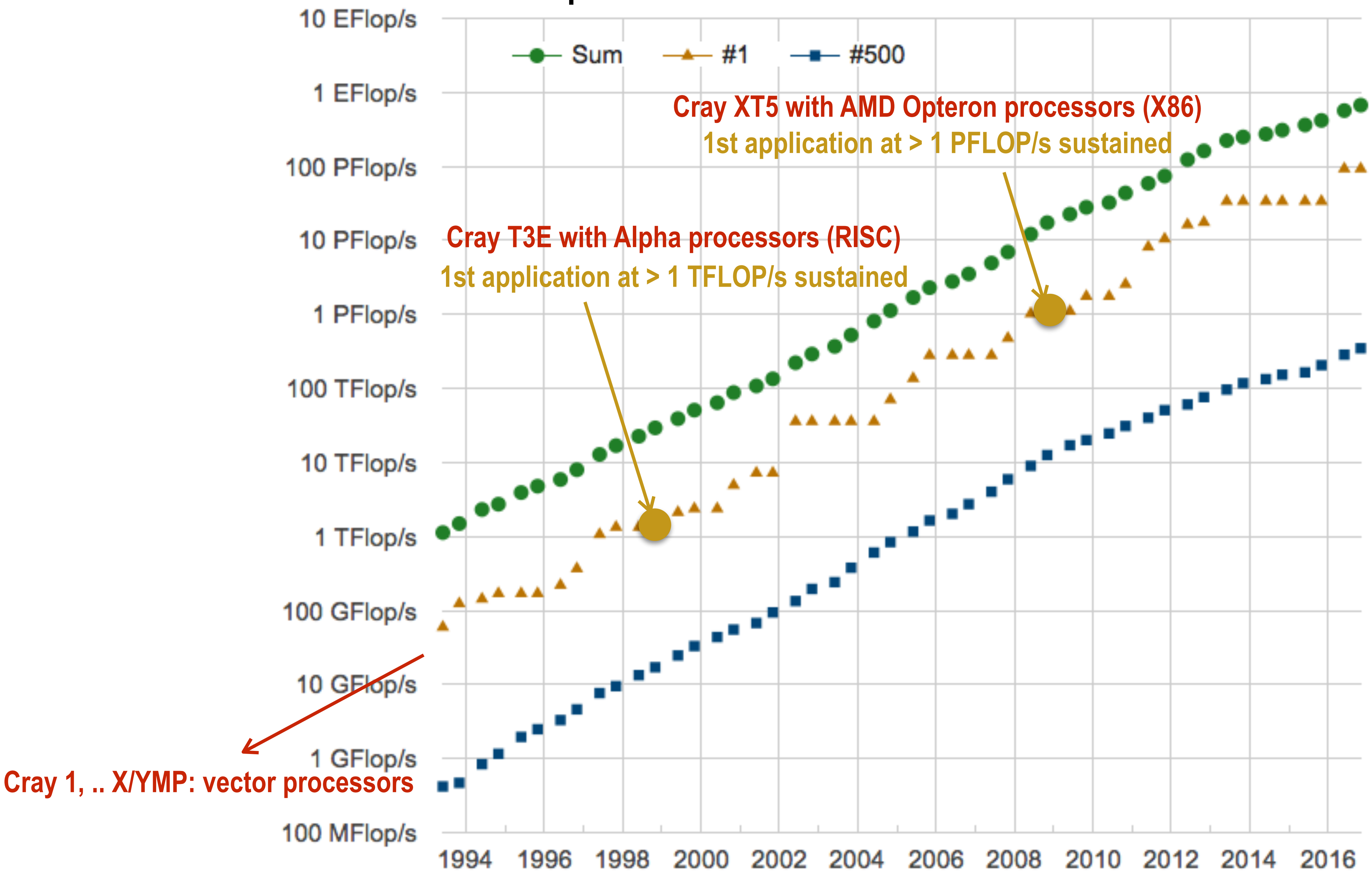
Beginning of change: “Attack of the Killer Micros”



Eugene Brooks (LLNL) @ SC90

The good old days of tera- and petascale computing

Linpack benchmark solves: $Ax = b$

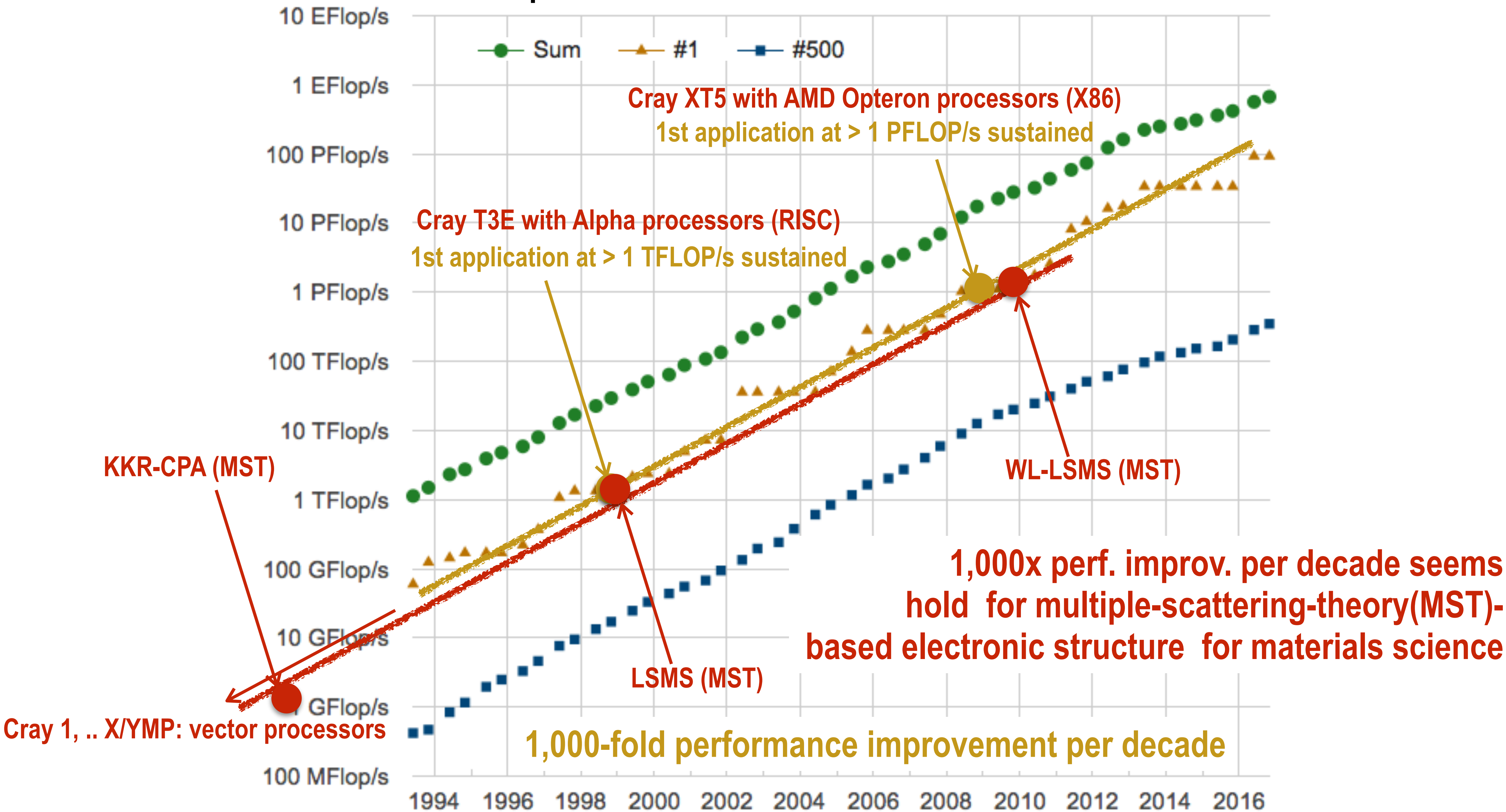


for the historic development of supercomputing performance, see www.top500.org


$500,000,000
$2,000,000,000
$13,000
Source: Andy Keane @ ISC'10

The good old days of tera- and petascale computing

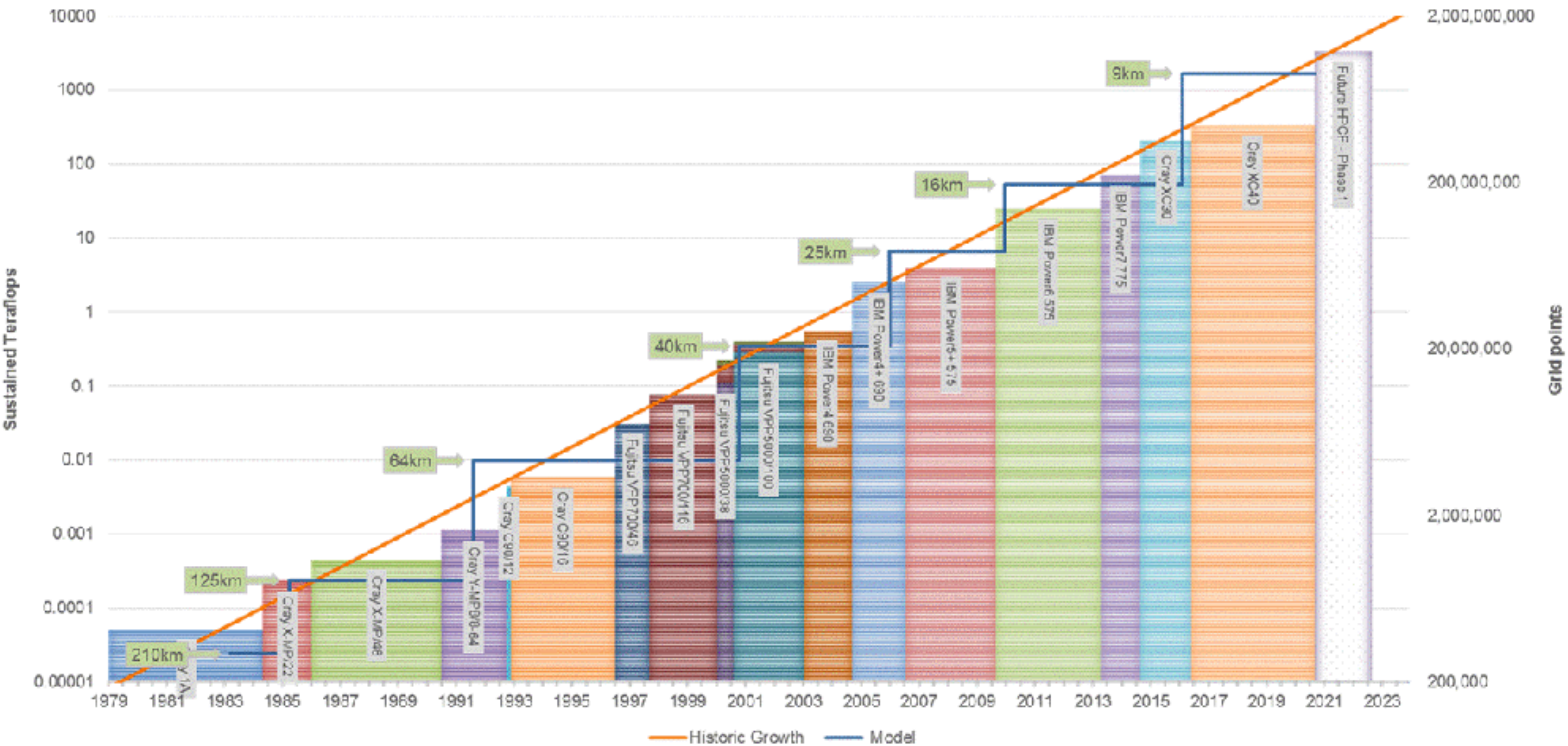
Linpack benchmark solves: $Ax = b$



for the historic development of supercomputing performance, see www.top500.org

ETH zürich

“Only” 100-fold performance improvement in climate codes



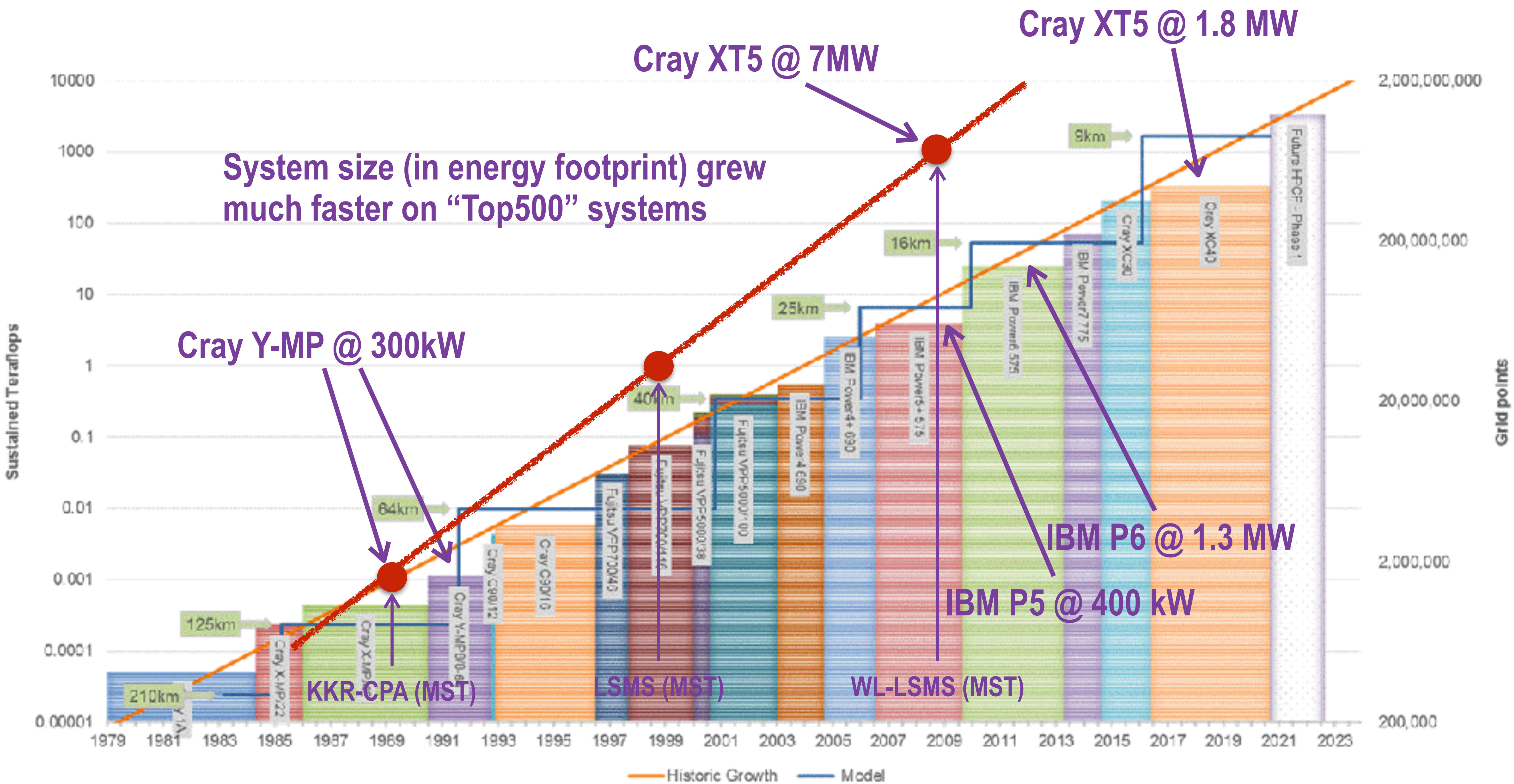
Source: Peter Bauer, ECMWF

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Has the efficiency of weather & climate codes dropped 10-fold every decade?

ETH zürich

Floating points efficiency dropped from 50% on Cray Y-MP to 5% on today's Cray XC (10x in 2 decades)

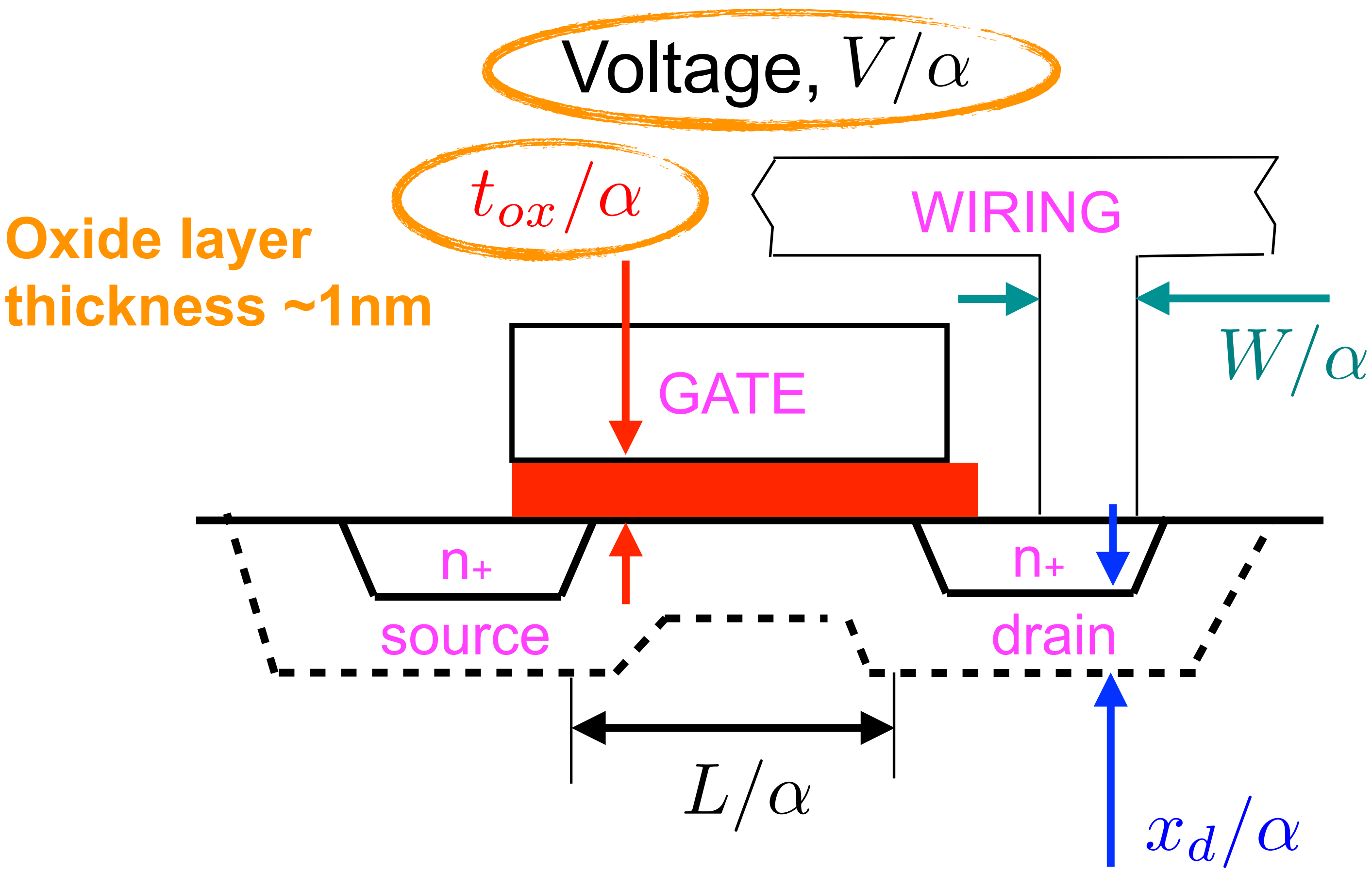


Source: Peter Bauer, ECMWF

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

The end of Dennard Scaling

Robert H. Dennard (1974)



p substrate, doping $\alpha^* N_A$

Source: Ronald Luijten, IBM-ZRL

SCALING

~~Voltage:~~ $\cancel{V/\alpha}$

~~Oxide:~~ $\cancel{t_{ox}/\alpha}$

Wire width: W/α

Gate Width: L/α

Diffusion: x_d/α

Substrate: $\alpha^* N_A$

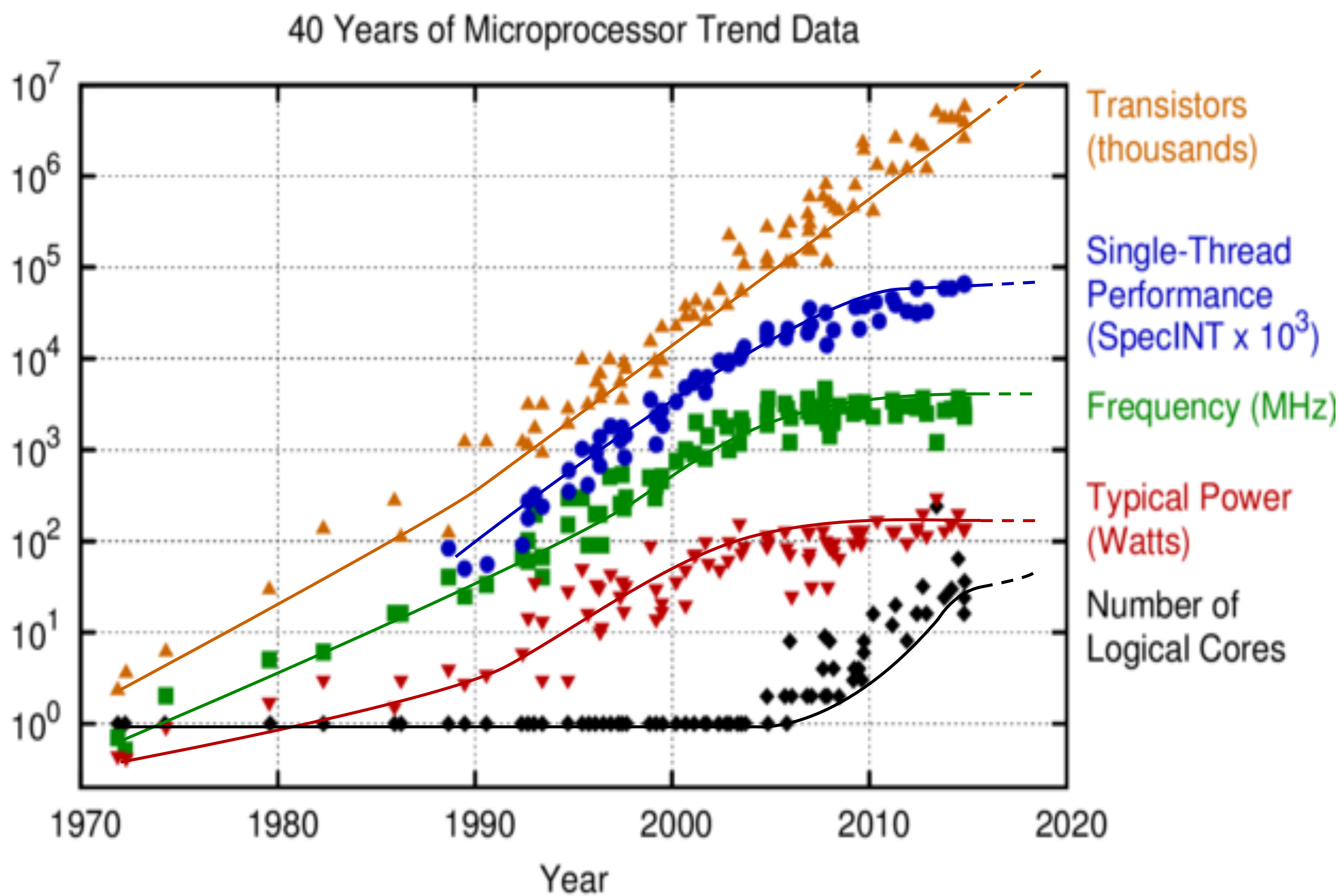
CONSEQUENCE:

Higher density: $\sim \alpha^2$

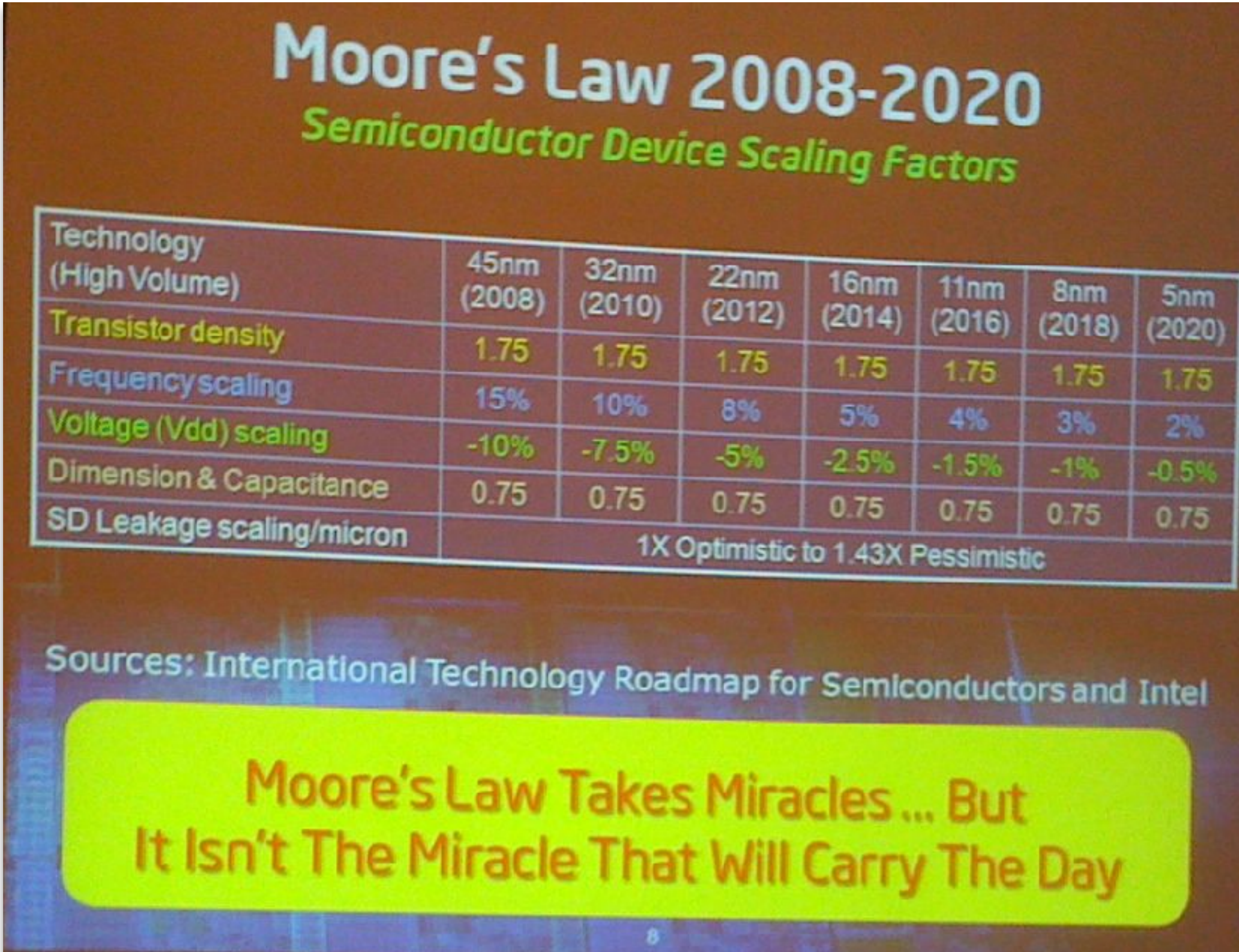
Higher speed: $\sim \alpha$

Power/ckt: $\sim 1/\alpha^2$

~~Power density:~~ $\cancel{\sim \text{constant}}$



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp



Moore's Law 2008-2020

Semiconductor Device Scaling Factors

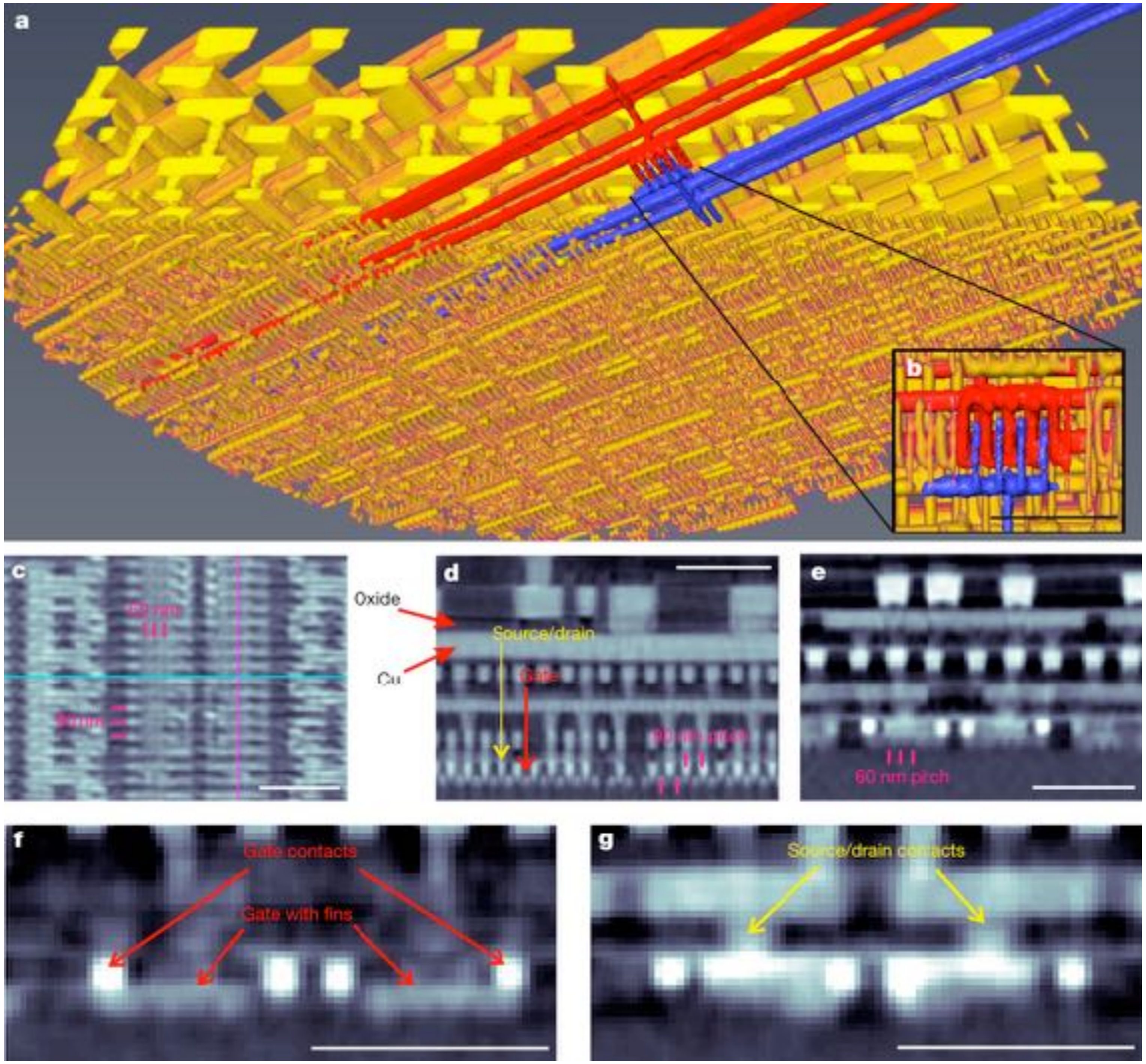
Technology (High Volume)	45nm (2008)	32nm (2010)	22nm (2012)	16nm (2014)	11nm (2016)	8nm (2018)	5nm (2020)
Transistor density	1.75	1.75	1.75	1.75	1.75	1.75	1.75
Frequency scaling	15%	10%	8%	5%	4%	3%	2%
Voltage (Vdd) scaling	-10%	-7.5%	-5%	-2.5%	-1.5%	-1%	-0.5%
Dimension & Capacitance	0.75	0.75	0.75	0.75	0.75	0.75	0.75
SD Leakage scaling/micron	1X Optimistic to 1.43X Pessimistic						

Sources: International Technology Roadmap for Semiconductors and Intel

Moore's Law Takes Miracles … But
It Isn't The Miracle That Will Carry The Day

Source: Rajeeb Hazra's (HPC@Intel) talk at SOS14, March 2010

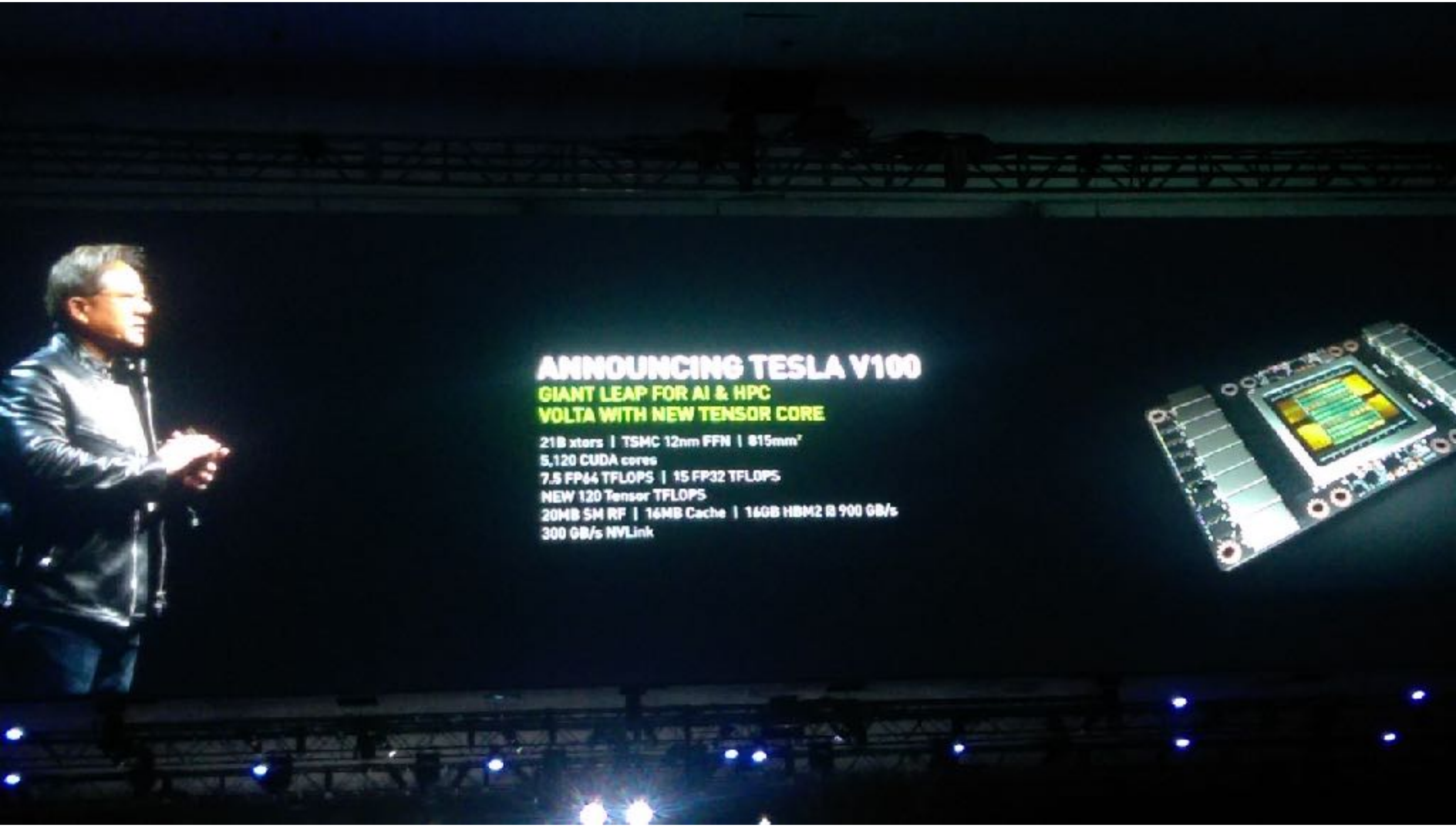
PXCT imaging of Intel processor



M Holler *et al.* *Nature* **543,** 402–406 (2017) doi:10.1038/nature21698


NVIDIA DGX-1
WORLD'S FIRST
DEEP LEARNING SUPERCOMPUTER
170TF | "250 servers in-a-box" | nvidia.com/dgx1
$129,000
for 8 GPUs, or $16k a piece


$500,000,000
$2,000,000,000
$13,000
Source: Andy Keane @ ISC'10


ANNOUNCING TESLA V100
GIANT LEAP FOR AI & HPC
VOLTA WITH NEW TENSOR CORE
21B xtors | TSMC 12nm FFN | 815mm²
5,120 CUDA cores
7.5 FP64 TFLOPS | 15 FP32 TFLOPS
NEW 120 Tensor TFLOPS
20MB SM RF | 16MB Cache | 16GB HBM2 @ 900 GB/s
300 GB/s NVLink

Porting codes to GPUs, Xeon (Phi), ARM, etc.

CUDA (C / C++ / Fortran)

```
__global__ void add_pw_ekin_gpu_kernel(int num_gvec__,
                                       double alpha__,
                                       double const* pw_ekin__,
                                       cuDoubleComplex const* phi__,
                                       cuDoubleComplex const* vphi__,
                                       cuDoubleComplex* hphi__)
{
    int ig = blockIdx.x * blockDim.x + threadIdx.x;
    if (ig < num_gvec__) {
        cuDoubleComplex z1 = cuCadd(vphi__[ig], make_cuDoubleComplex(alph
                                                                     alph
        hphi__[ig] = cuCadd(hphi__[ig], z1);
    }
}
```

OpenCL

```
__kernel void vector_add(const int n, __global float *a, __global float *b, __global float *c) {
```

OpenACC

```
acc = 0
!$acc parallel present(x)
!$acc loop reduction(+:acc)
do i = 1, N
    acc = acc + x(i) * x(i)
enddo
!$acc end parallel
call mpi_allreduce(acc, accglobal, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD, err)
```

OpenMP 4.x

```
 data map(tofrom: x[0:n],y[0:n])
rget
#pragma omp for
for (int i = 0; i < n; i++)
    y[i] += a * x[i];
}
```

Architectural diversity is here to stay, because it is a consequence of the dusk of CMOS scaling (Moore's Law)

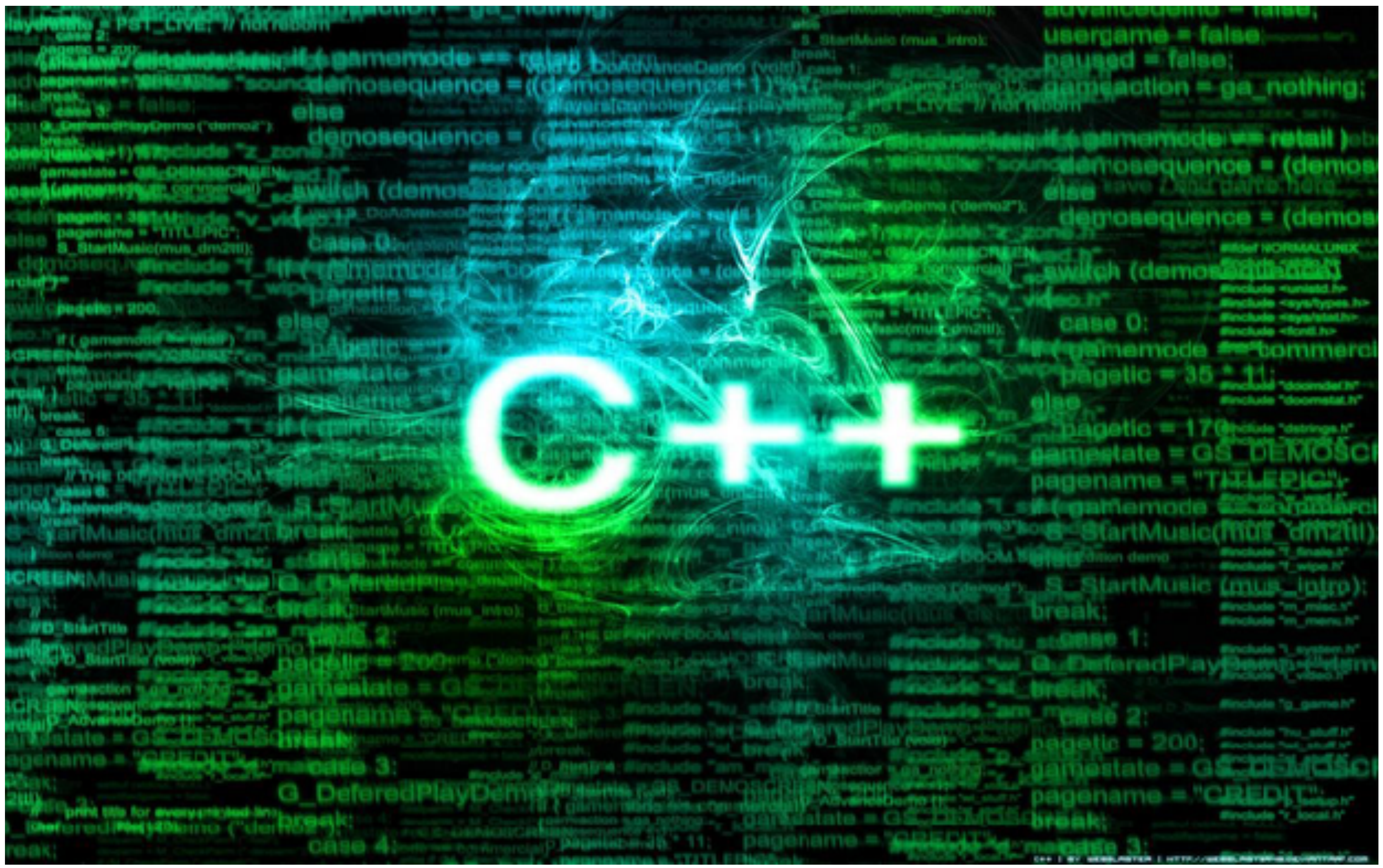
What are the implications?

Complexity in software is one,
but we don't understand all implications

Physics of the computer matters more than ever

The good news

C++ standard is evolving quickly and implementations follow!



C++ 11, 14, (HPX-3/Kokkos), … 17, 20, …

Who will pay for the implementation of Fortran, OpenACC, OpenMP, …?

The top ranking programming languages in 2017
spectrum.ieee.org

Language Types (click to hide)

Web | Mobile | Enterprise | Embedded

Language Rank	Types	Spectrum Ranking
1. Python	Web, Enterprise	100.0
2. C	Mobile, Enterprise, Embedded	99.7
3. Java	Web, Mobile, Enterprise	99.5
4. C++	Mobile, Enterprise, Embedded	97.1
5. C#	Web, Mobile, Enterprise	87.7
6. R	Enterprise	87.7
7. JavaScript	Web, Mobile	85.6
8. PHP	Web	81.2
9. Go	Web, Enterprise	75.1
10. Swift	Mobile, Enterprise	73.7

ETH zürich

The top ranking programming languages in 2017

spectrum.ieee.org

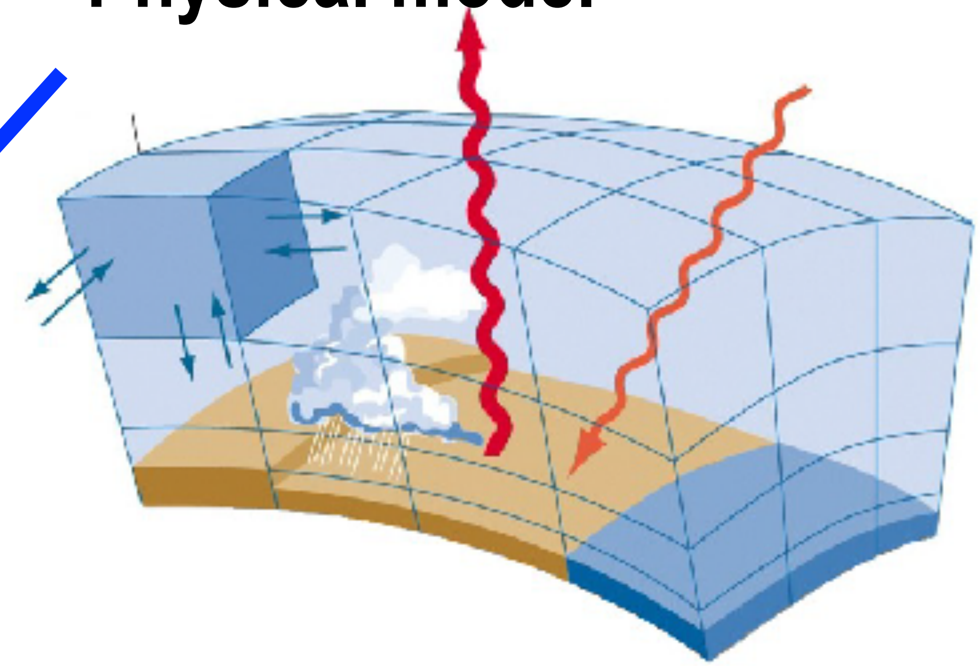
Language Types (click to hide)

Web | Mobile | Enterprise | Embedded

Language Rank	Types	Spectrum Ranking
1. Python	Web, Enterprise	100.0
2. C	Mobile, Enterprise, Embedded	100.0
3. Java	Web, Mobile, Enterprise	99.4
4. C++	Mobile, Enterprise, Embedded	96.9
5. C#	Web, Mobile, Enterprise	88.6
6. R	Enterprise	88.1
7. JavaScript	Web, Mobile	85.3
8. PHP	Web	81.1
9. Go	Web, Enterprise	75.7
10. Swift	Mobile, Enterprise	74.3
11. Arduino	Embedded	72.4
12. Ruby	Web, Enterprise	72.0
13. Assembly	Embedded	71.7
14. Matlab	Enterprise	68.6
15. Scala	Web, Mobile	68.0
16. HTML	Web	67.1
17. Shell	Enterprise	65.0
18. Perl	Web, Enterprise	57.0
19. Visual Basic	Enterprise	54.3
20. Cuda	Enterprise	52.6
28. Fortran	Enterprise	40.3

ETH zürich

Physical model



Wind $\rho\dot{\mathbf{v}} = -\nabla p + \rho\mathbf{g} - 2\Omega\times(\rho\mathbf{v}) + \mathbf{F}$

Pressure $\dot{p} = -(c_{pd}/c_{vd})\,p\nabla\cdot\mathbf{v} + (c_{pd}/c_{vd}-1)Q_h$

Temperature $\rho c_{pd}\dot{T} = \dot{p} + Q_h$

Water $\rho\dot{q}^{v} = -\nabla\cdot\mathbf{F}^{v} - (I^{l} + I^{f})$

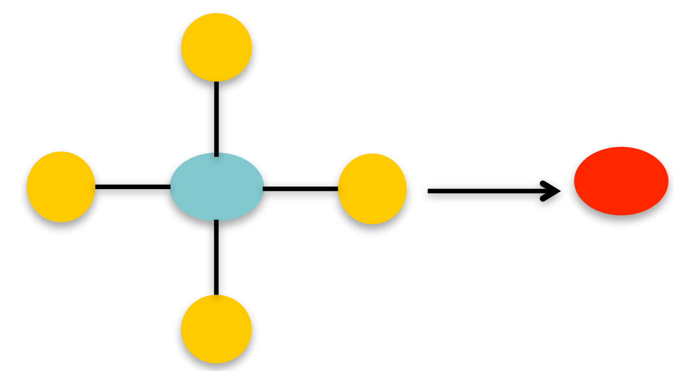
$\rho\dot{q}^{l,f} = \nabla\cdot(\mathbf{P}^{l,f} + \mathbf{F}^{l,f}) + I^{l,f}$

Density $\rho = p[R_d\,(1+(R_v/R_d-1)\,q^{v} - q^{l} - q^{f})\,T]^{-1}$

Mathematical description

Domain science & applied mathematics

Algorithmic description



```
lap(i,j,k) = -4.0 * data(i,j,k) +
    data(i+1,j,k) + data(i-1,j,k) +
    data(i,j+1,k) + data(i,j-1,k);
```

Imperative code

Compilation

Computer

Computer engineering

Schulthess, Nature Physics, vol 11, 369-373 (2015)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

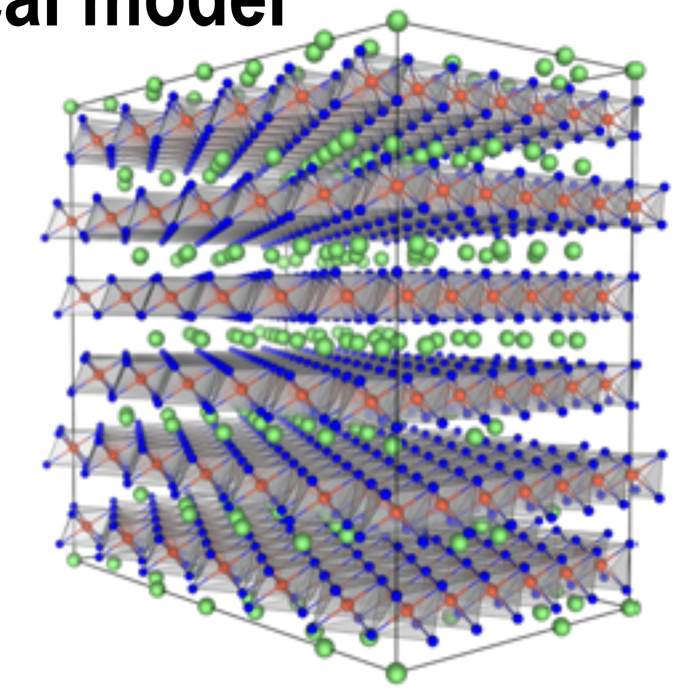
DFT ground state

Eigen-value problem $\hat{H}\Psi_{i\mathbf{k}}(\mathbf{r}) = \epsilon_{i\mathbf{k}}\Psi_{i\mathbf{k}}(\mathbf{r})$

Charge density $\rho_{\sigma\sigma'}(\mathbf{r}) = \sum_{i\mathbf{k}}^{occ} \Psi^{*\sigma'}_{i\mathbf{k}}(\mathbf{r})\Psi^{\sigma}_{i\mathbf{k}}(\mathbf{r})$

Effective potential $v^{eff}(\mathbf{r}) = v^{eff}[\rho_{\sigma\sigma'}(\mathbf{r})](\mathbf{r})$
and magnetic field $\mathbf{B}^{eff}(\mathbf{r}) = \mathbf{B}^{eff}[\rho_{\sigma\sigma'}(\mathbf{r})](\mathbf{r})$

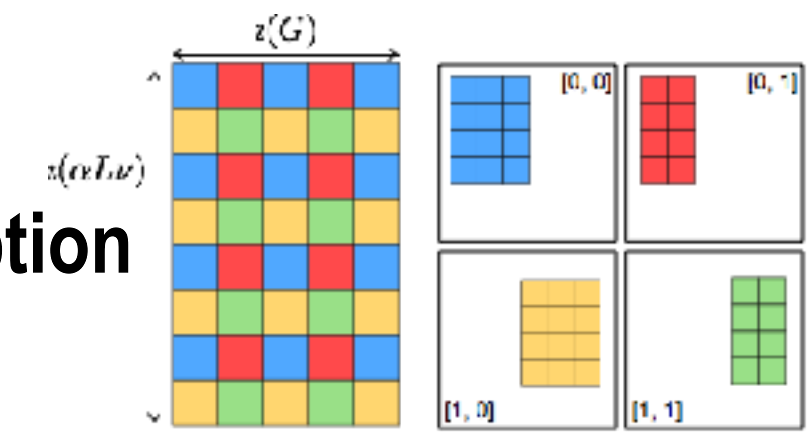
Physical model



Mathematical description

Domain science & applied mathematics

Algorithmic description



Imperative code

Compilation

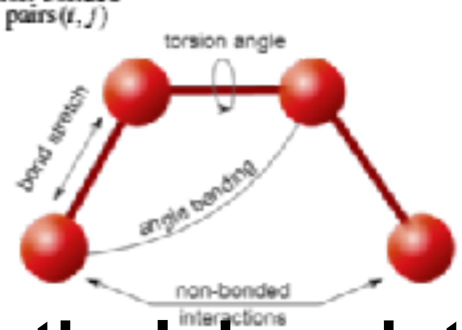
Computer

Computer engineering

Schulthess, Nature Physics, vol 11, 369-373 (2015)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

$$V(r) = \sum_{\text{bonds}} k_b(b-b_0)^2 + \sum_{\text{angles}} k_\theta(\theta-\theta_0)^2 + \sum_{\text{dihedrals}} k_\phi(1+\cos(n\phi-\phi_0)) + \sum_{\text{impropers}} k_\psi(\psi-\psi_0)^2 + \sum_{\substack{\text{non-bonded}\\ \text{pairs}(i,j)}} 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] + \sum_{\substack{\text{non-bonded}\\ \text{pairs}(i,j)}} \frac{q_i q_j}{\varepsilon_D r_{ij}}$$



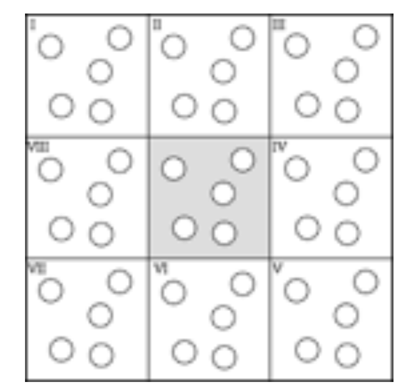
Physical model

Mathematical description

Domain science & applied mathematics

```
Cray PAT API "11 loop init mtran" in UKCA_COAGWITHNUCL
      call pat_region_begin(11,'11 loop init mtran',pat_stat)
!
!!!       DO IMODE=1,NMODES
!!!        NDOLD(:,IMODE)=ND(:,IMODE)
!!!        DO ICP=1,NCP
!!!         MDOLD(:,IMODE,ICP)=MD(:,IMODE,ICP)
!!!         DO JMODE=1,NMODES
!!!          MTRAN(:,IMODE,JMODE,ICP)=0.0
!!!         ENDDO
!!!        ENDDO
!!!       ENDDO
!
! replace triple loops above with F90 array syntax (let
compiler decide)
      NDOLD=ND
      MDOLD=MD
      MTRAN=0.0
      call pat_region_end(11,pat_stat)
```

Algorithmic description



Imperative code

Compilation

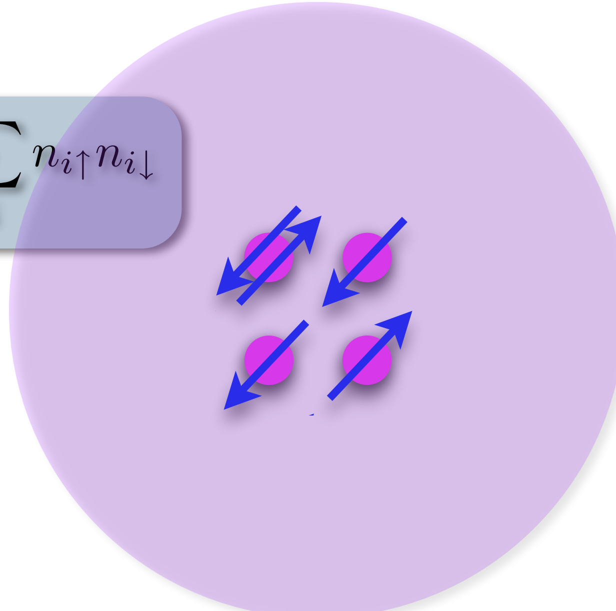
Computer

Computer engineering

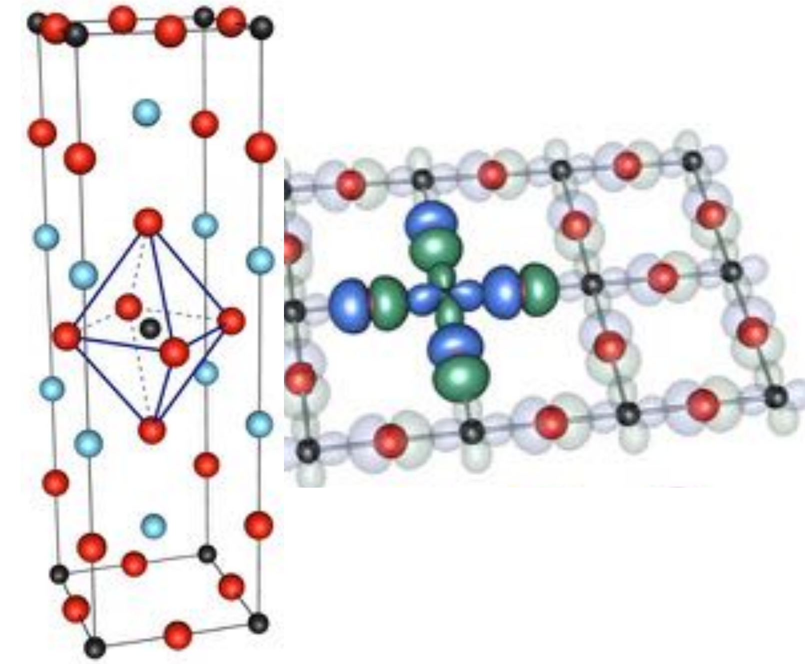
Schulthess, Nature Physics, vol 11, 369-373 (2015)

ETH zürich

$$\mathcal{H} = -t \sum_{\langle ij \rangle, \sigma} c^{\dagger}_{i\sigma} c_{j\sigma} + U \sum_{i} n_{i\uparrow} n_{i\downarrow}$$



Physical model

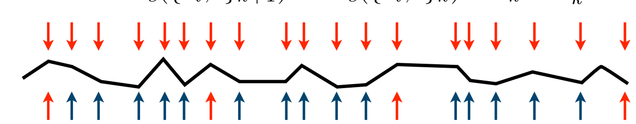


Mathematical description

Domain science & applied mathematics

Algorithmic description

$$\mathbf{G}_c(\{s_i, l\}_{k+1}) = \mathbf{G}_c(\{s_i, l\}_k) + \mathbf{a}_k \times \mathbf{b}^t_k$$



$$\mathbf{G}_c(\{s_i, l\}_{k+1}) = \mathbf{G}_c(\{s_i, l\}_0) + [\mathbf{a}_0|\mathbf{a}_1|...|\mathbf{a}_k] \times [\mathbf{b}_0|\mathbf{b}_1|...|\mathbf{b}_k]^t$$

Imperative code

Compilation

Computer

Computer engineering

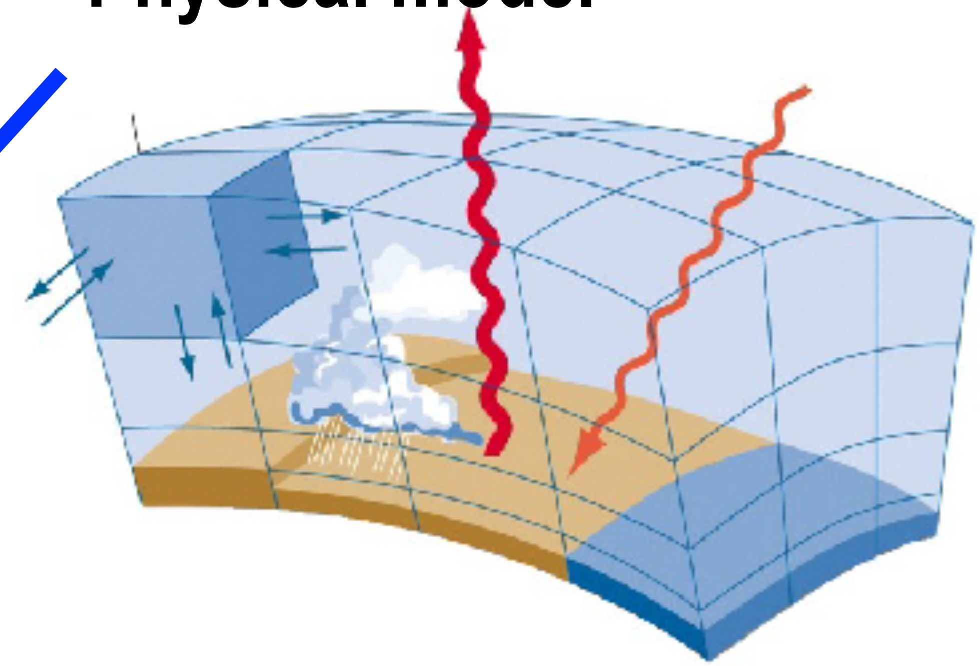
Schulthess, Nature Physics, vol 11, 369-373 (2015)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

$$
\begin{aligned}
\text{Wind } \rho\dot{\mathbf{v}} &= -\nabla p + \rho\mathbf{g} - 2\Omega\times(\rho\mathbf{v}) + \mathbf{F} \\
\text{Pressure } \dot{p} &= -(c_{pd}/c_{vd})\, p\nabla\cdot\mathbf{v} + (c_{pd}/c_{vd}-1)Q_h \\
\text{Temperature } \rho c_{pd}\,\dot{T} &= \dot{p}+Q_h \\
\text{Water } \rho\dot{q}^{v} &= -\nabla\cdot\mathbf{F}^{v} - (I^{l}+I^{f}) \\
\rho\dot{q}^{l,f} &= \nabla\cdot(\mathbf{P}^{l,f} + \mathbf{F}^{l,f}) + I^{l,f} \\
\text{Density } \rho &= p[R_d\,(1+(R_v/R_d-1)\,q^{v}-q^{l}-q^{f})\,T]^{-1}
\end{aligned}
$$

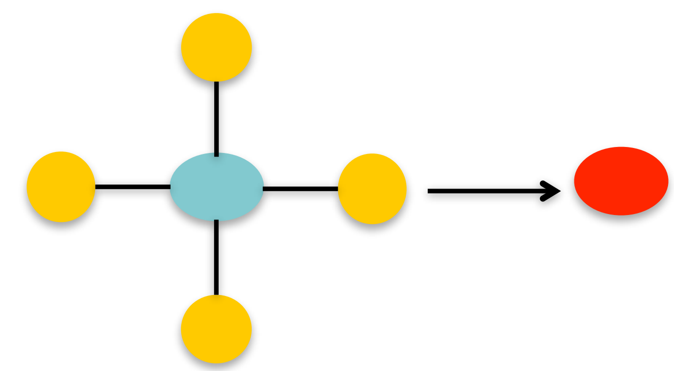
Physical model



Mathematical description

Domain science & applied mathematics

Algorithmic description



```
lap(i,j,k) = -4.0 * data(i,j,k) +
    data(i+1,j,k) + data(i-1,j,k) +
    data(i,j+1,k) + data(i,j-1,k);
```

Imperative code

Compilation

Computer

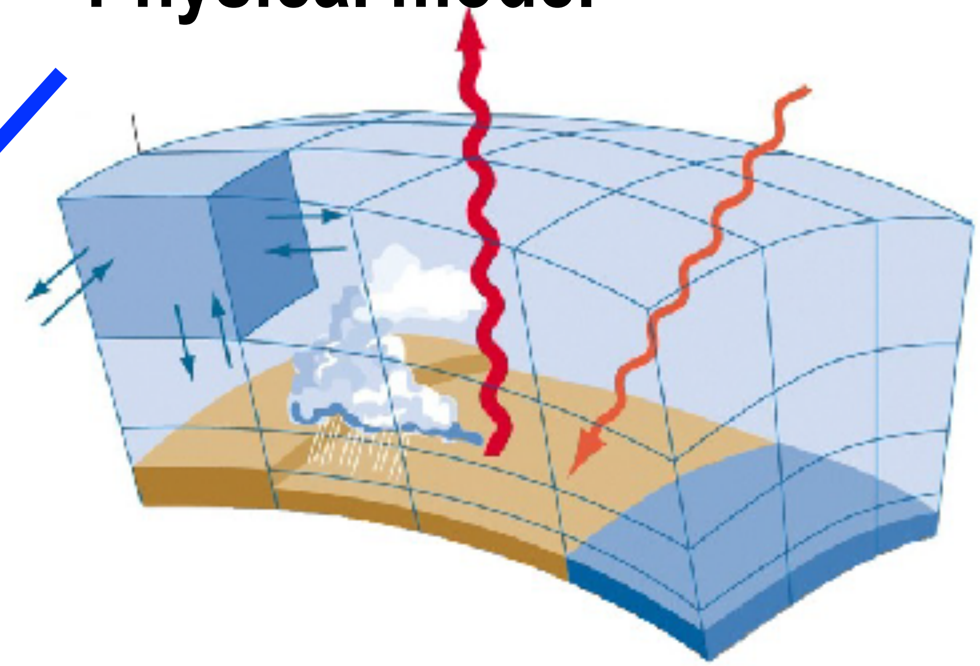
Computer engineering

Schulthess, Nature Physics, vol 11, 369-373 (2015)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Physical model



Wind $\rho\dot{\mathbf{v}} = -\nabla p + \rho\mathbf{g} - 2\Omega\times(\rho\mathbf{v}) + \mathbf{F}$

Pressure $\dot{p} = -(c_{pd}/c_{vd})\, p\nabla\cdot\mathbf{v} + (c_{pd}/c_{vd}-1)Q_h$

Temperature $\rho c_{pd}\dot{T} = \dot{p} + Q_h$

Water $\rho\dot{q}^{v} = -\nabla\cdot\mathbf{F}^{v} - (I^{l} + I^{f})$

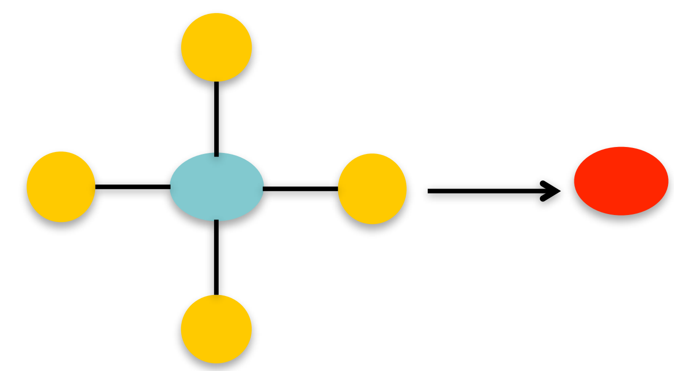
$\rho\dot{q}^{l,f} = \nabla\cdot(\mathbf{P}^{l,f} + \mathbf{F}^{l,f}) + I^{l,f}$

Density $\rho = p[R_d\,(1+(R_v/R_d-1)\,q^{v} - q^{l} - q^{f})\,T]^{-1}$

Mathematical description

Domain science & applied mathematics

Algorithmic description



```
lap(i,j,k) = -4.0 * data(i,j,k) +
    data(i+1,j,k) + data(i-1,j,k) +
    data(i,j+1,k) + data(i,j-1,k);
```

Imperative code

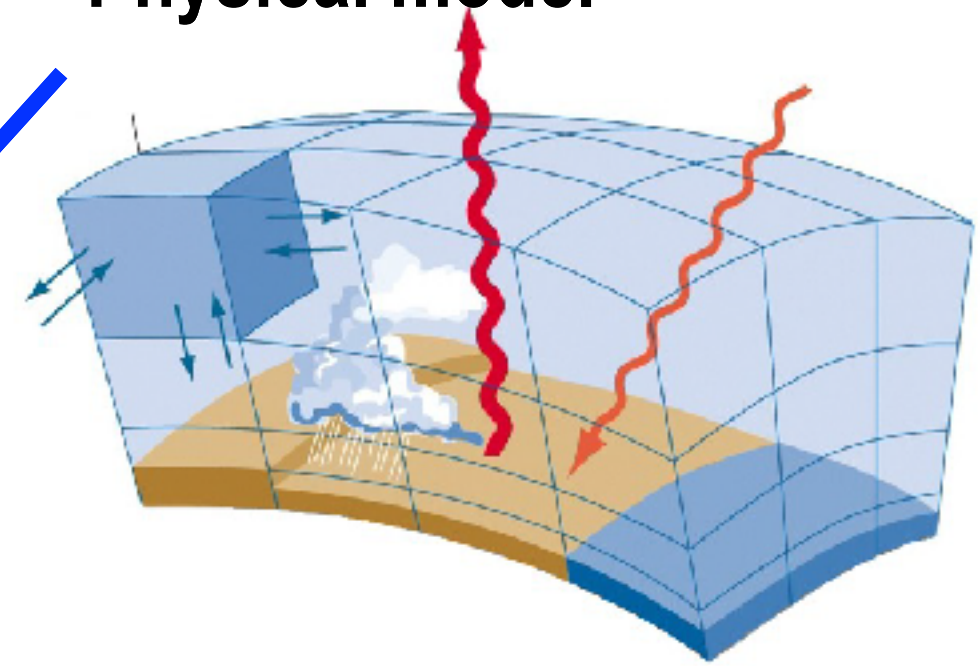
Compilation

Computer engineering

Schulthess, Nature Physics, vol 11, 369-373 (2015)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Physical model



$$\text{Wind } \rho\dot{\mathbf{v}} = -\nabla p + \rho\mathbf{g} - 2\Omega\times(\rho\mathbf{v}) + \mathbf{F}$$

$$\text{Pressure } \dot{p} = -(c_{pd}/c_{vd})\, p\nabla\cdot\mathbf{v} + (c_{pd}/c_{vd}-1)Q_h$$

$$\text{Temperature } \rho c_{pd}\dot{T} = \dot{p} + Q_h$$

$$\text{Water } \rho\dot{q}^v = -\nabla\cdot\mathbf{F}^v - (I^l + I^f)$$

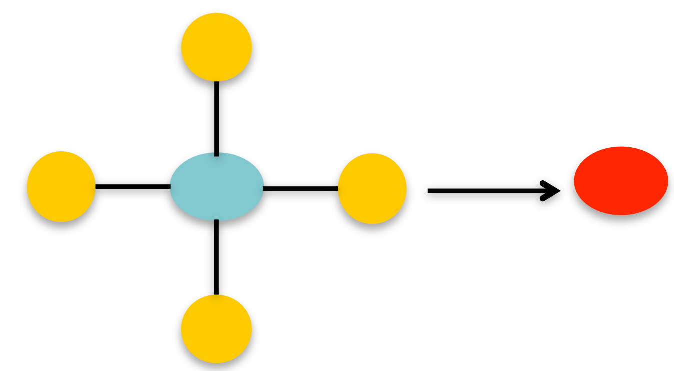
$$\rho\dot{q}^{l,f} = \nabla\cdot(\mathbf{P}^{l,f} + \mathbf{F}^{l,f}) + I^{l,f}$$

$$\text{Density } \rho = p[R_d\,(1+(R_v/R_d-1)\,q^v - q^l - q^f)\,T]^{-1}$$

Mathematical description

Domain science & applied mathematics

Algorithmic description



```
lap(i,j,k) = -4.0 * data(i,j,k) +
    data(i+1,j,k) + data(i-1,j,k) +
    data(i,j+1,k) + data(i,j-1,k);
```

Imperative code

Compilation

Computer engineering

Schulthess, Nature Physics, vol 11, 369-373 (2015)

Wind $\rho\dot{\mathbf{v}} = -\nabla p + \rho\mathbf{g} - 2\Omega\times(\rho\mathbf{v}) + \mathbf{F}$

Pressure $\dot{p} = -(c_{pd}/c_{vd})\, p\nabla\cdot\mathbf{v} + (c_{pd}/c_{vd}-1)Q_h$

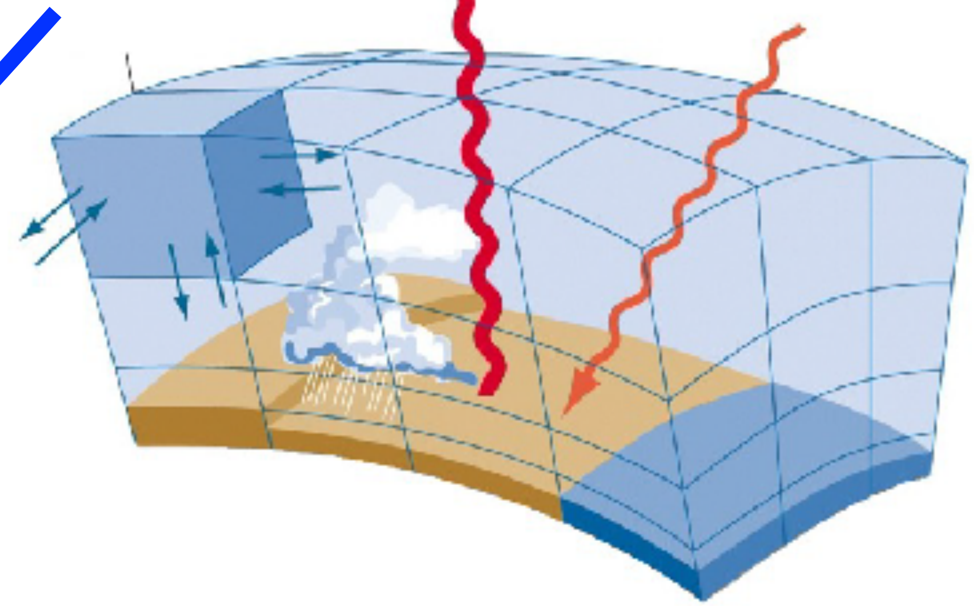
Temperature $\rho c_{pd}\dot{T} = \dot{p} + Q_h$

Water $\rho\dot{q}^{v} = -\nabla\cdot\mathbf{F}^{v} - (I^{l}+I^{f})$

$\rho\dot{q}^{l,f} = \nabla\cdot(\mathbf{P}^{l,f}+\mathbf{F}^{l,f}) + I^{l,f}$

Density $\rho = p[R_d\,(1+(R_v/R_d-1)\,q^{v} - q^{l} - q^{f})\,T]^{-1}$

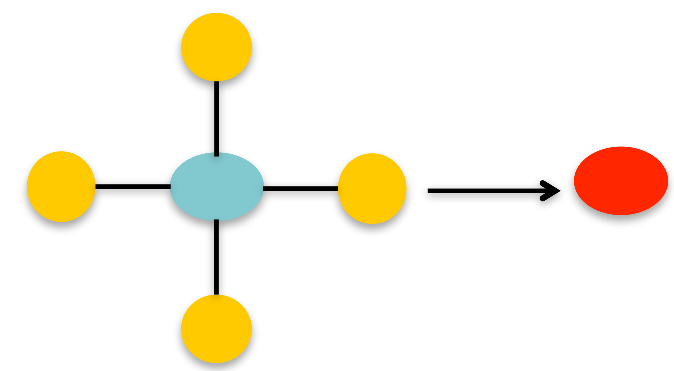
Physical model



Mathematical description

Domain science & applied mathematics

Algorithmic description



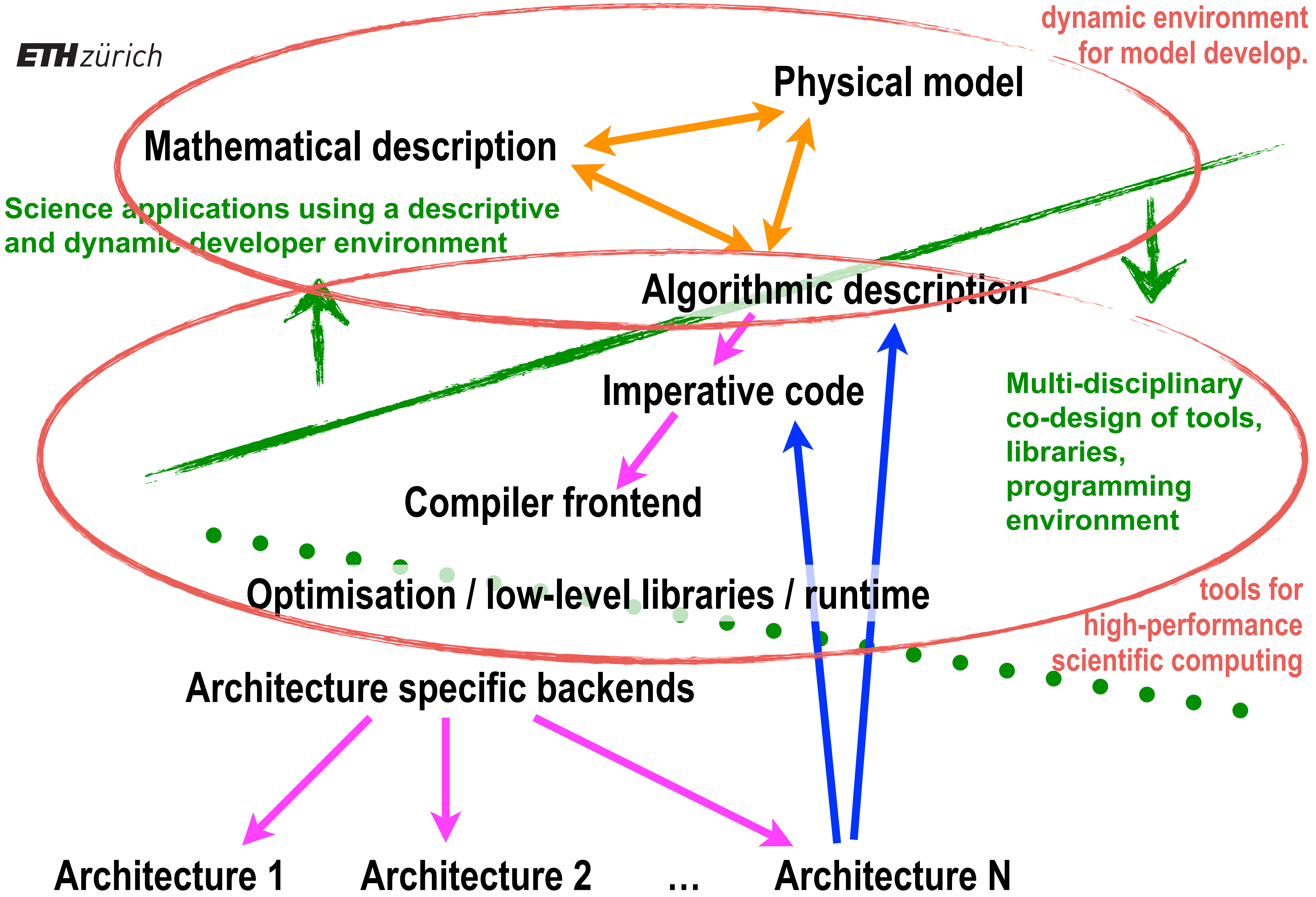
```
lap(i,j,k) = -4.0 * data(i,j,k) +
    data(i+1,j,k) + data(i-1,j,k) +
    data(i,j+1,k) + data(i,j-1,k);
```

Imperative code

Compilation

Computer engineering

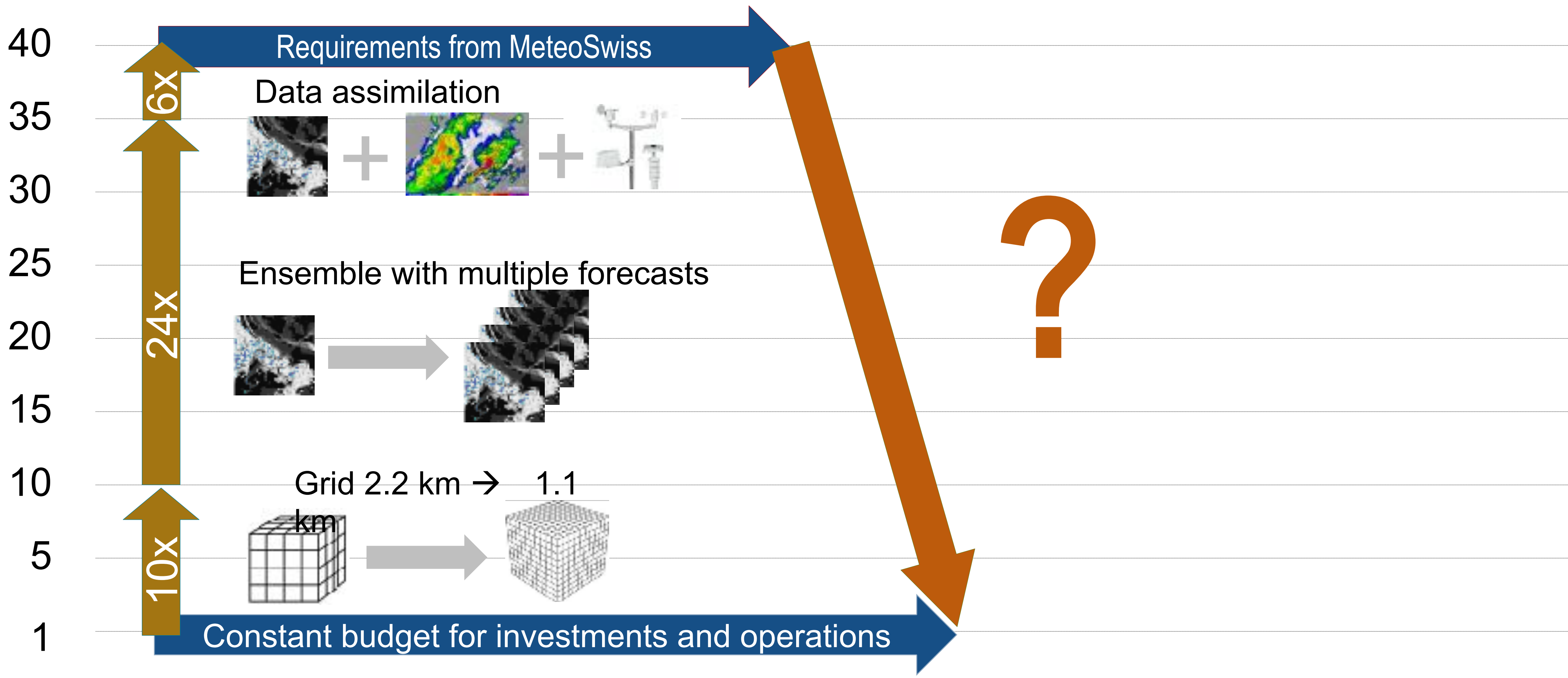
Schulthess, Nature Physics, vol 11, 369-373 (2015)



Schulthess, Nature Physics, vol 11, 369-373 (2015)

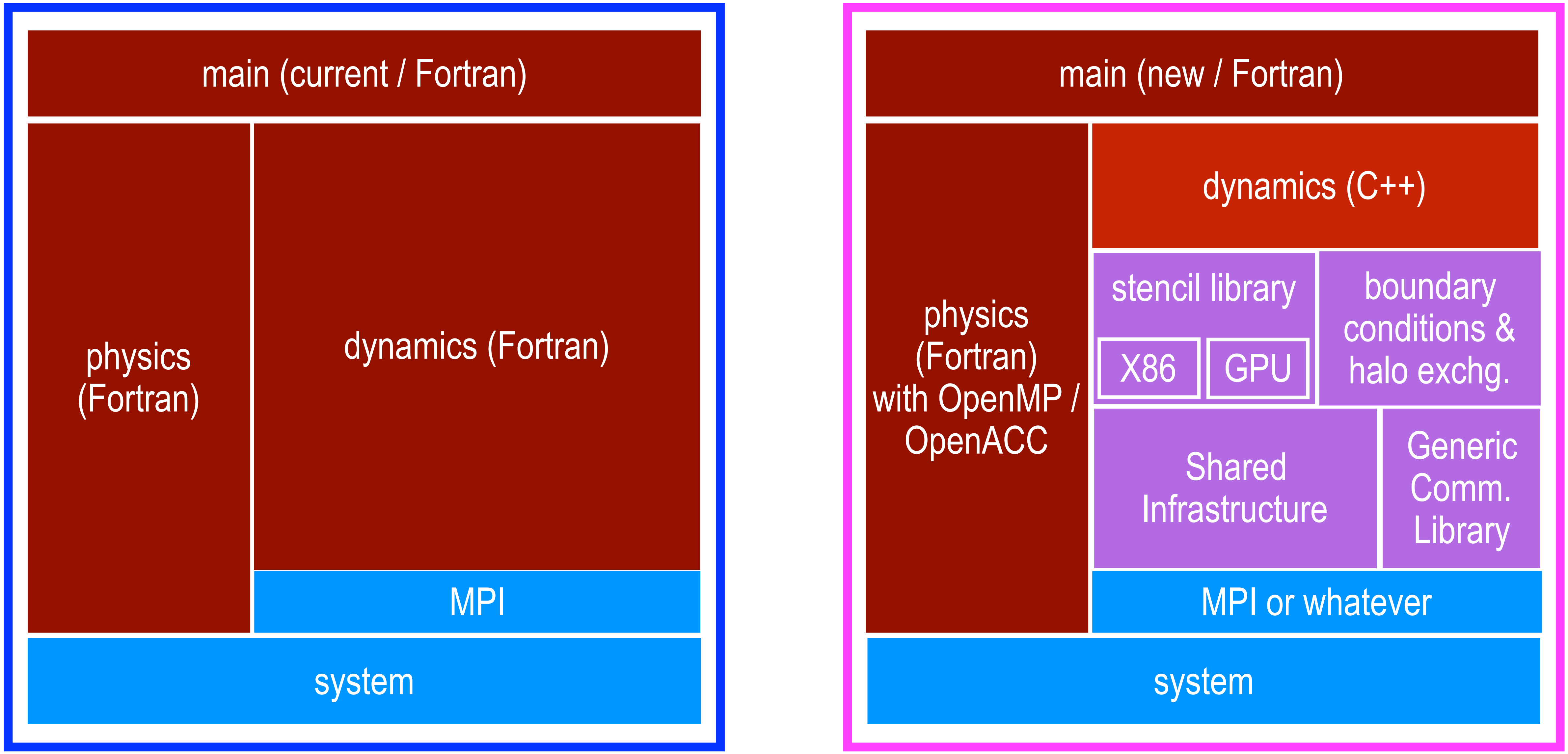
ETH zürich

MeteoSwiss' performance ambitions in 2013



We need a 40x improvement between 2012 and 2015 at constant cost

COSMO: old and new (refactored) code

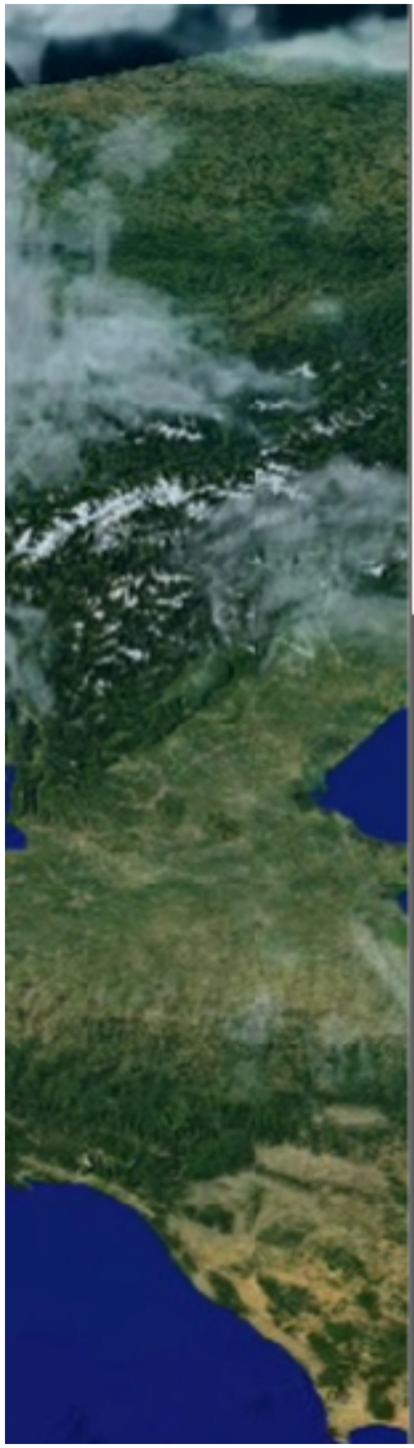


HPCwire

September 15, 2015

Today’s Outlook: GPU-accelerated Weather Forecasting

John Russell



MeteoSwiss New Weather Supercomputer

World’s First GPU-Accelerated Weather Forecasting System

2x Racks

48 CPUs

192 Tesla K80 GPUs

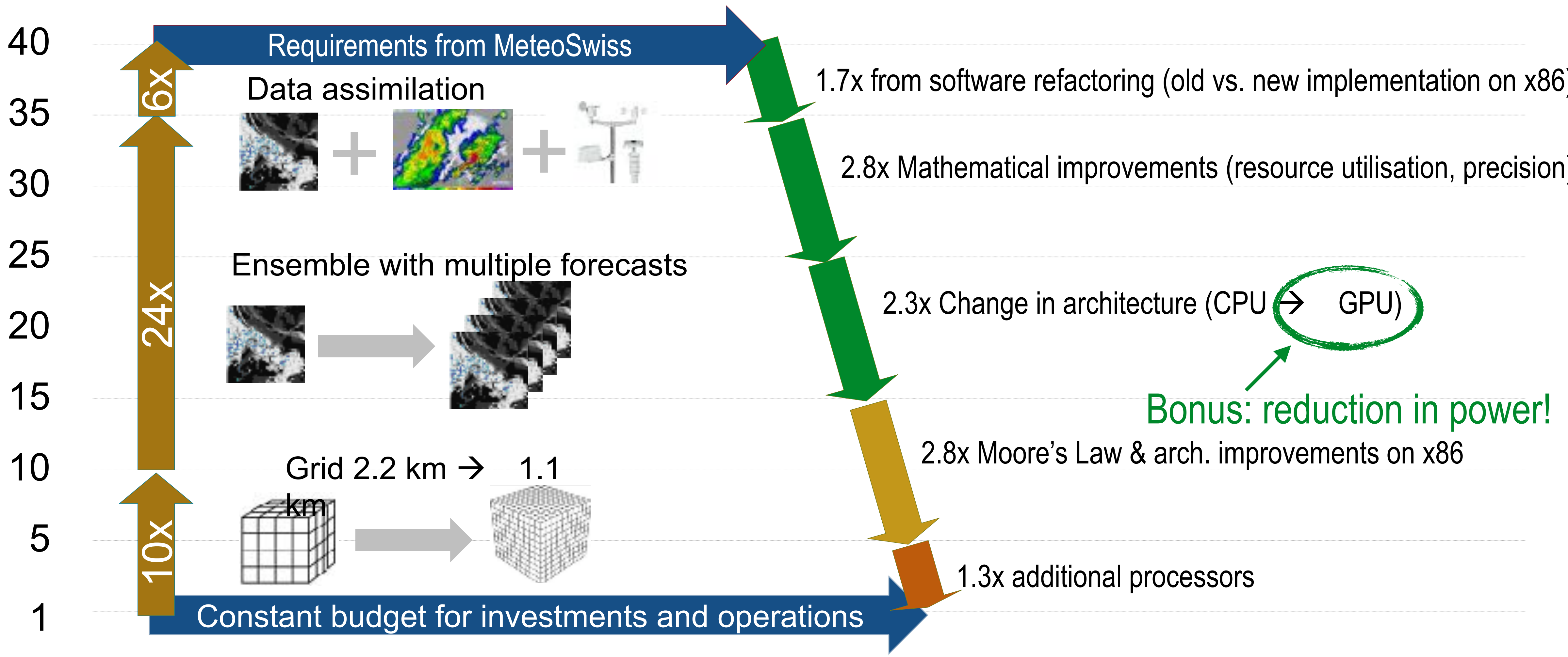
> 90% of FLOPS from GPUs

Operational in 2016

5 NVIDIA

Where the factor 40 improvement came from

Investment in software allowed mathematical improvements and change in architecture



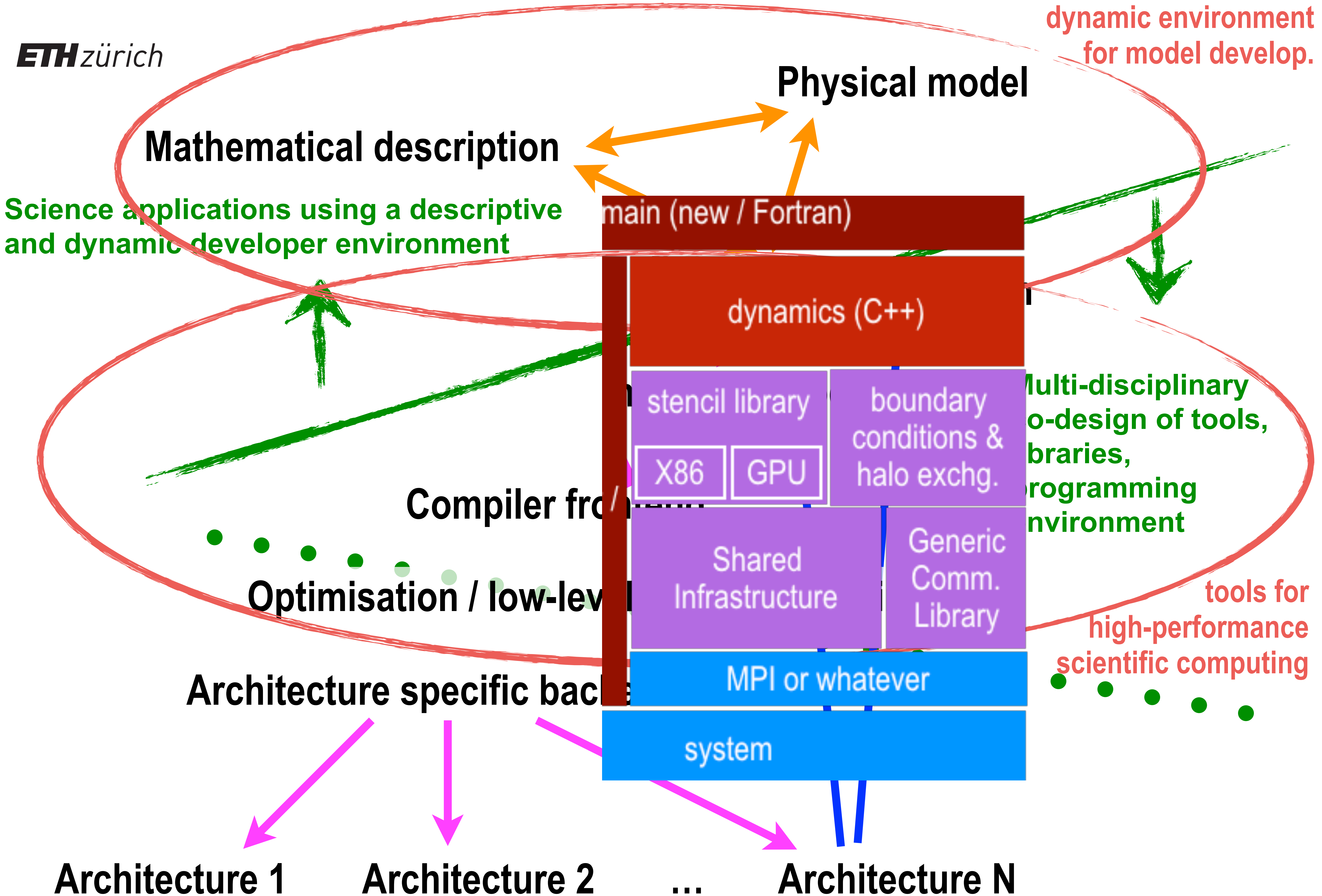
There is no silver bullet!

Setting a new baseline for atmospheric simulations

The state-of the art implementation of COSMO running at most weather services on multi-core hardware.

~10x

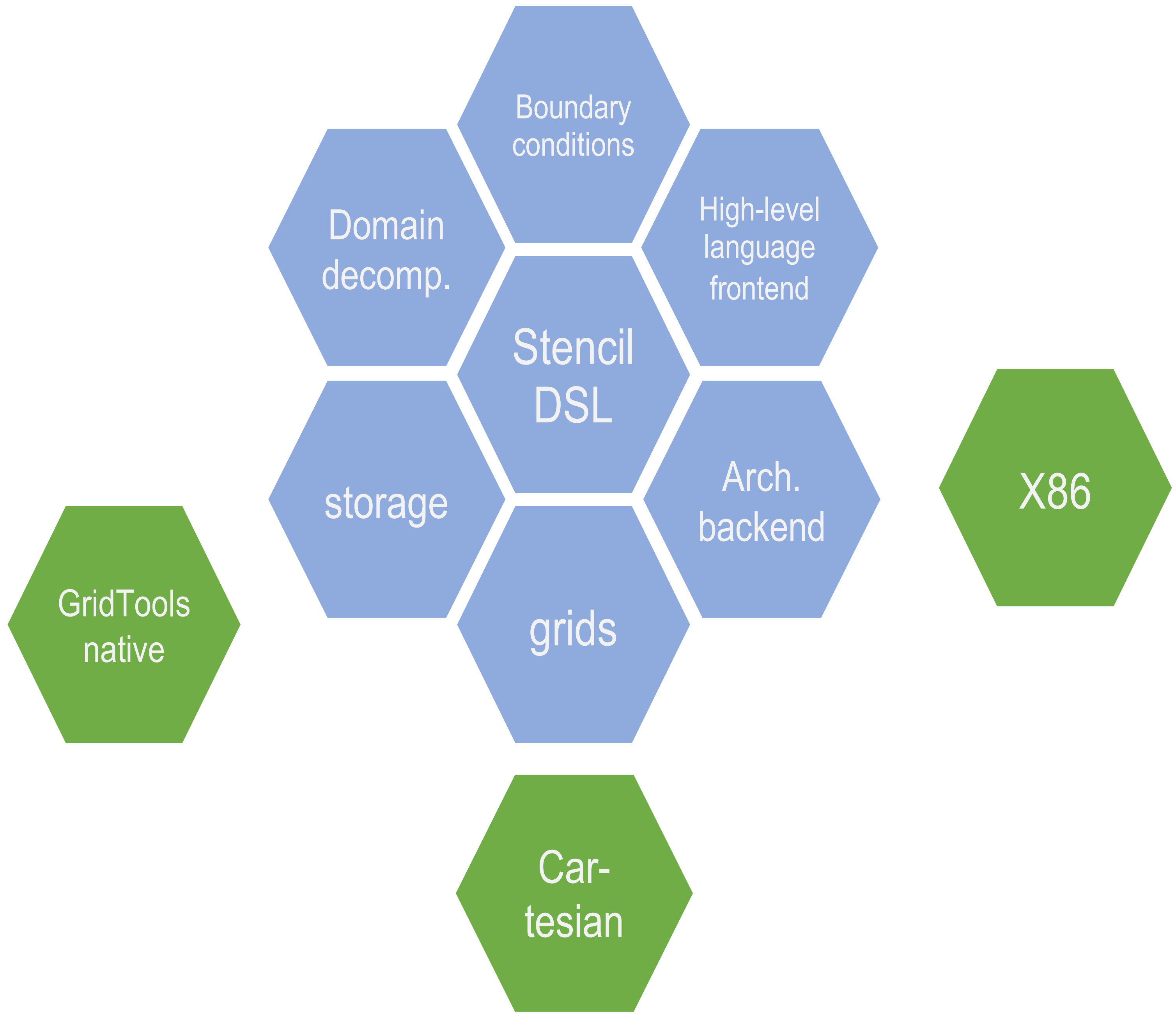
The refactored version of COSMO running at MeteoSwiss on multi-core or GPU accelerated hardware.

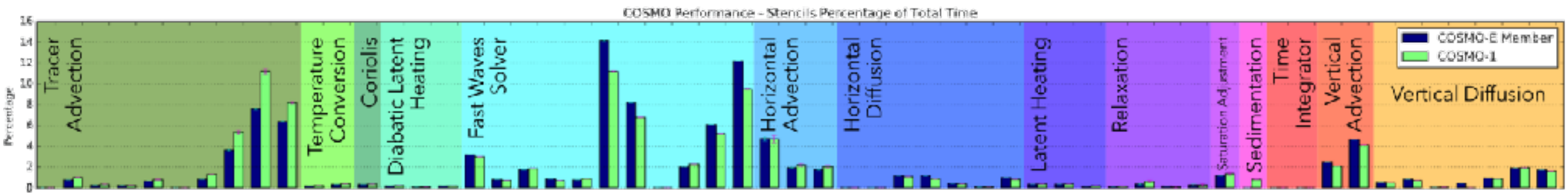


Schulthess, Nature Physics, vol 11, 369-373 (2015)

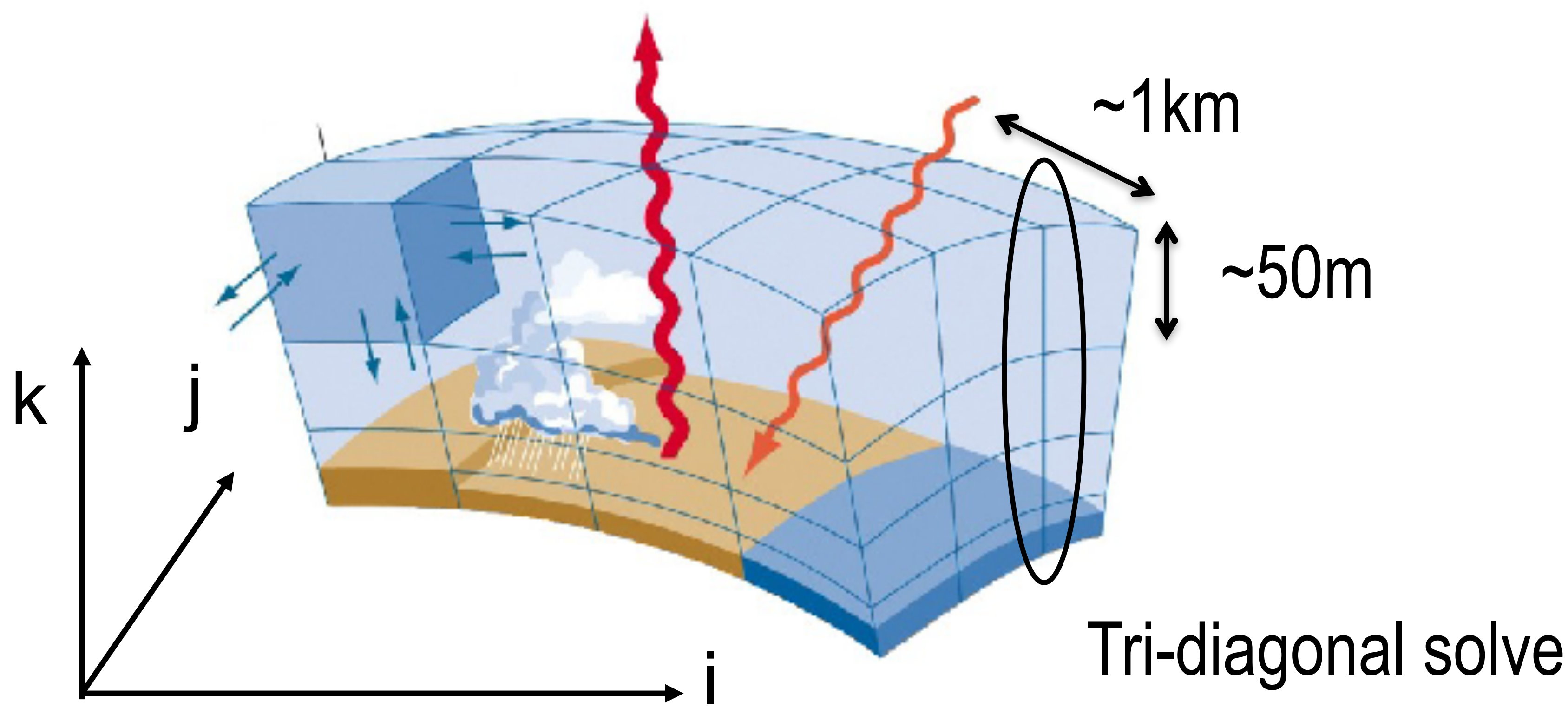
GridTools Framework

COSMO @ MeteoSwiss





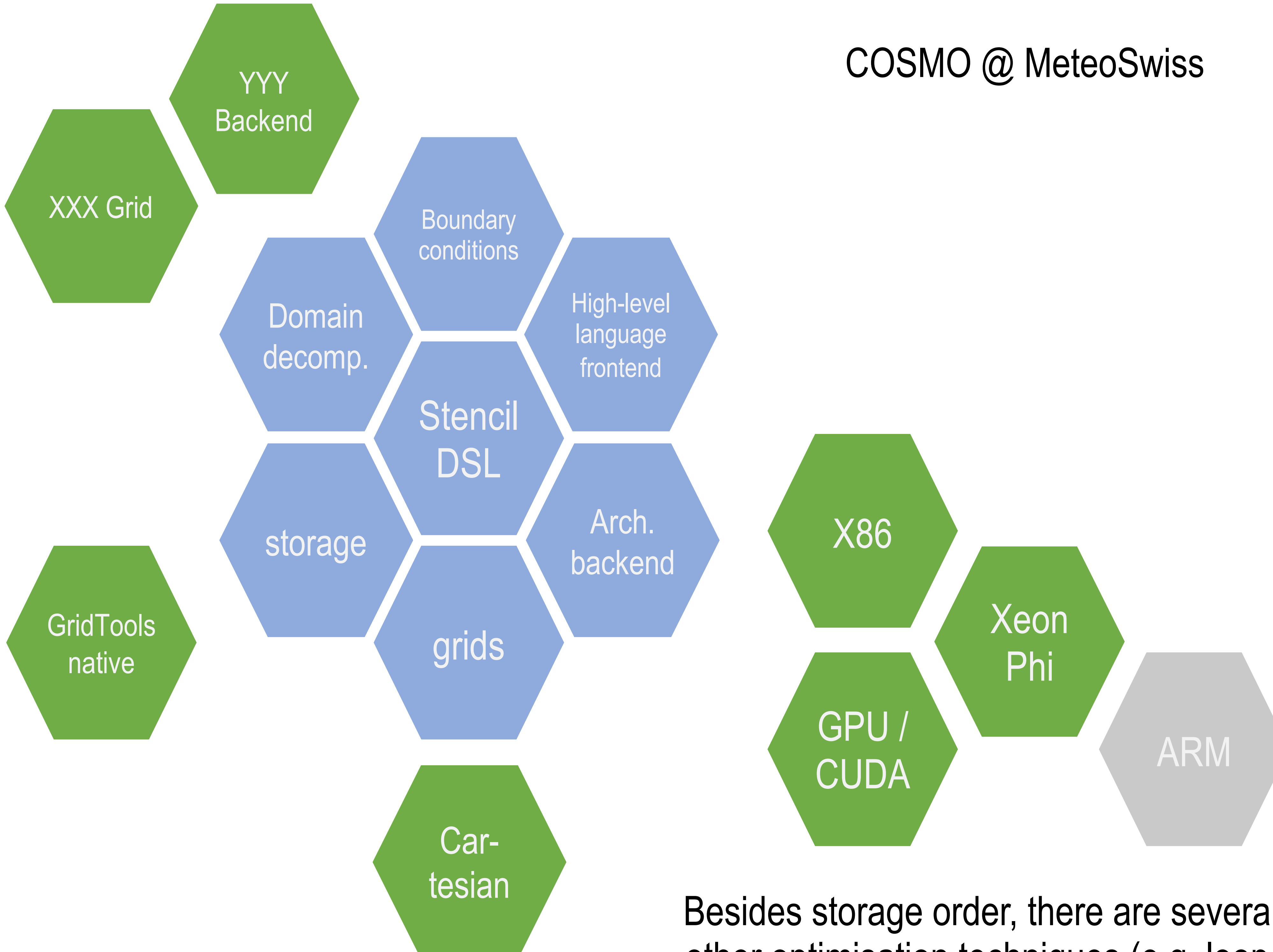
No performance hotspots



Store fields in (i,j,k) or (k,i,j) order?

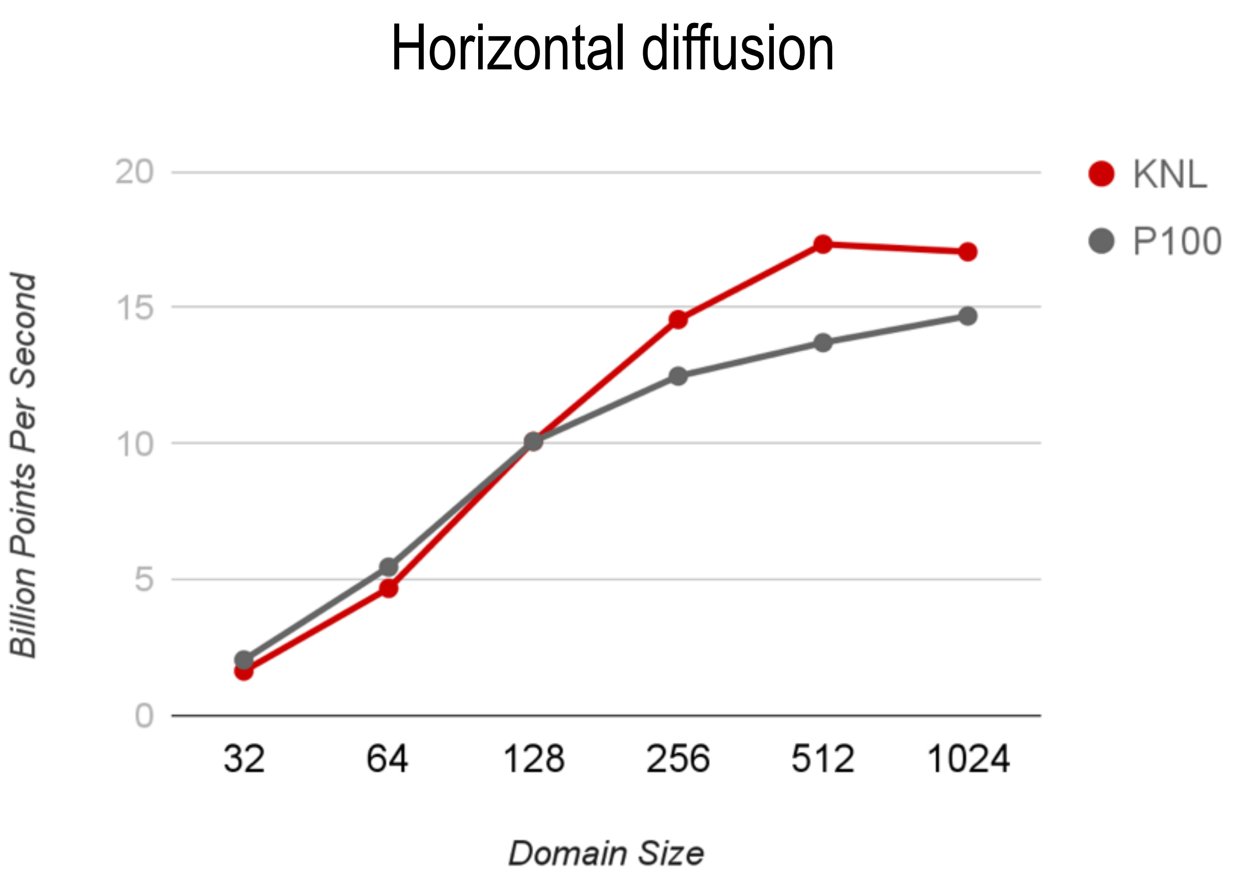
This depends on the architecture

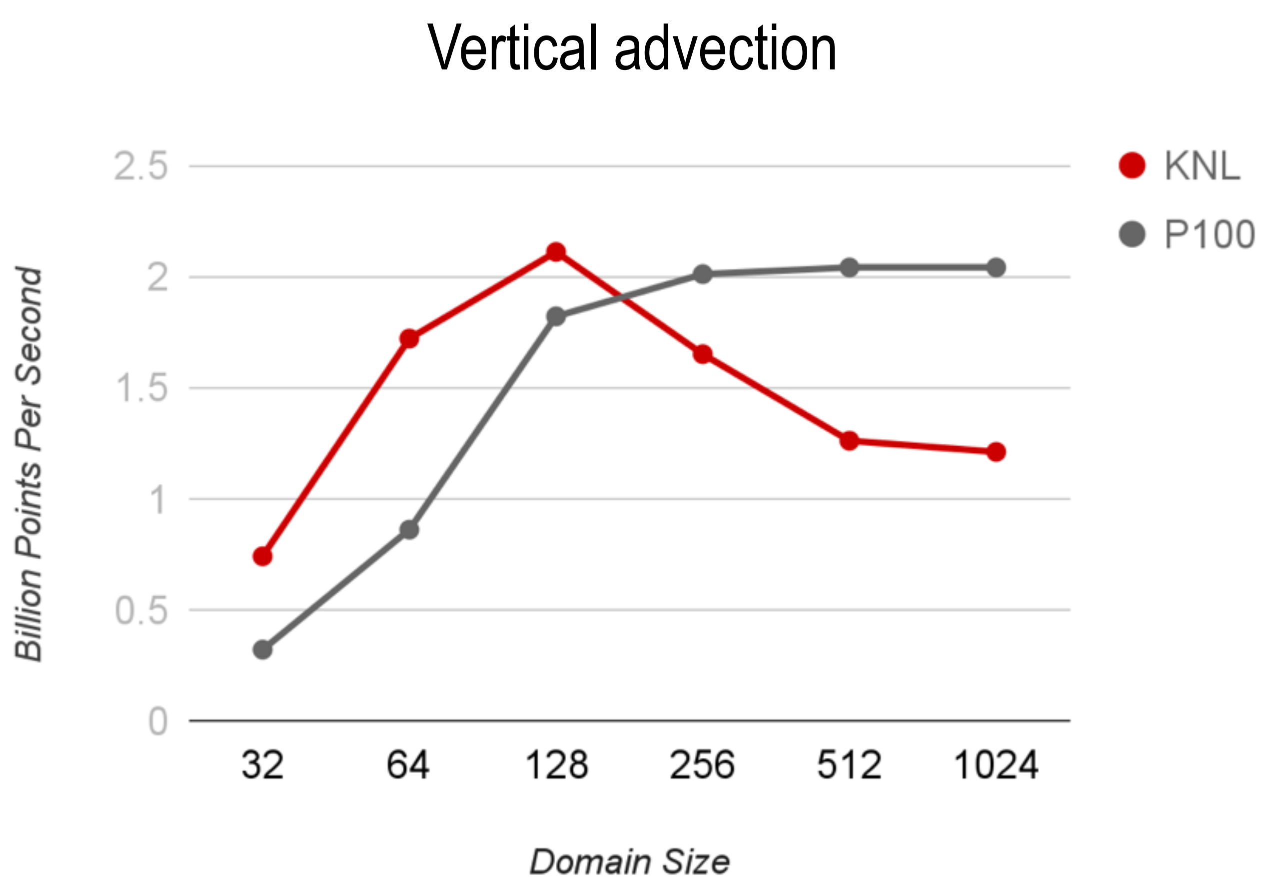
COSMO @ MeteoSwiss



Besides storage order, there are several other optimisation techniques (e.g. loop/stencil fusion) where details depend on architecture specifics

Exploring Intel Xeon Phi (KNL) and NVIDIA’s P100





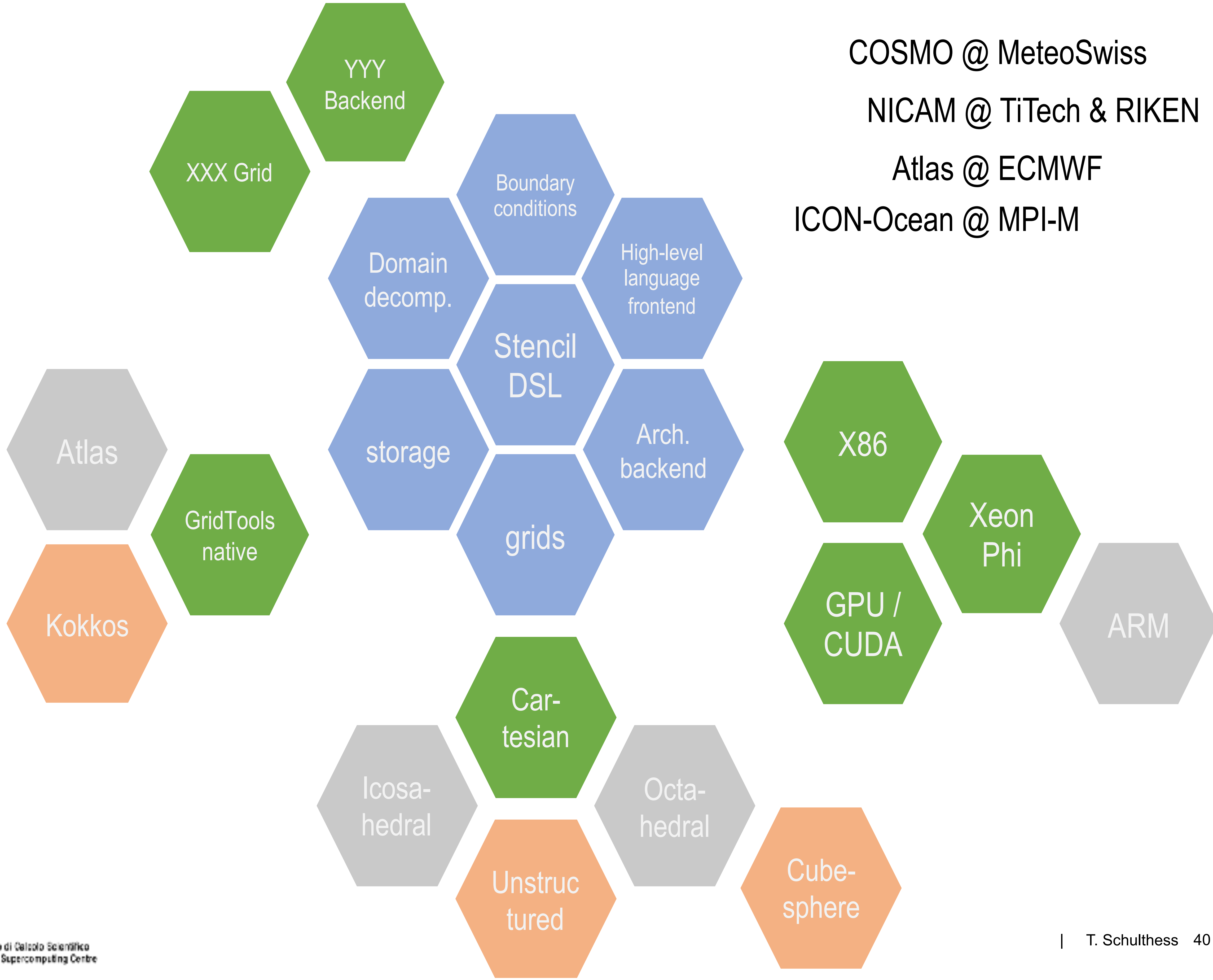
Source: Felix Thaler, CSCS

COSMO @ MeteoSwiss

NICAM @ TiTech & RIKEN

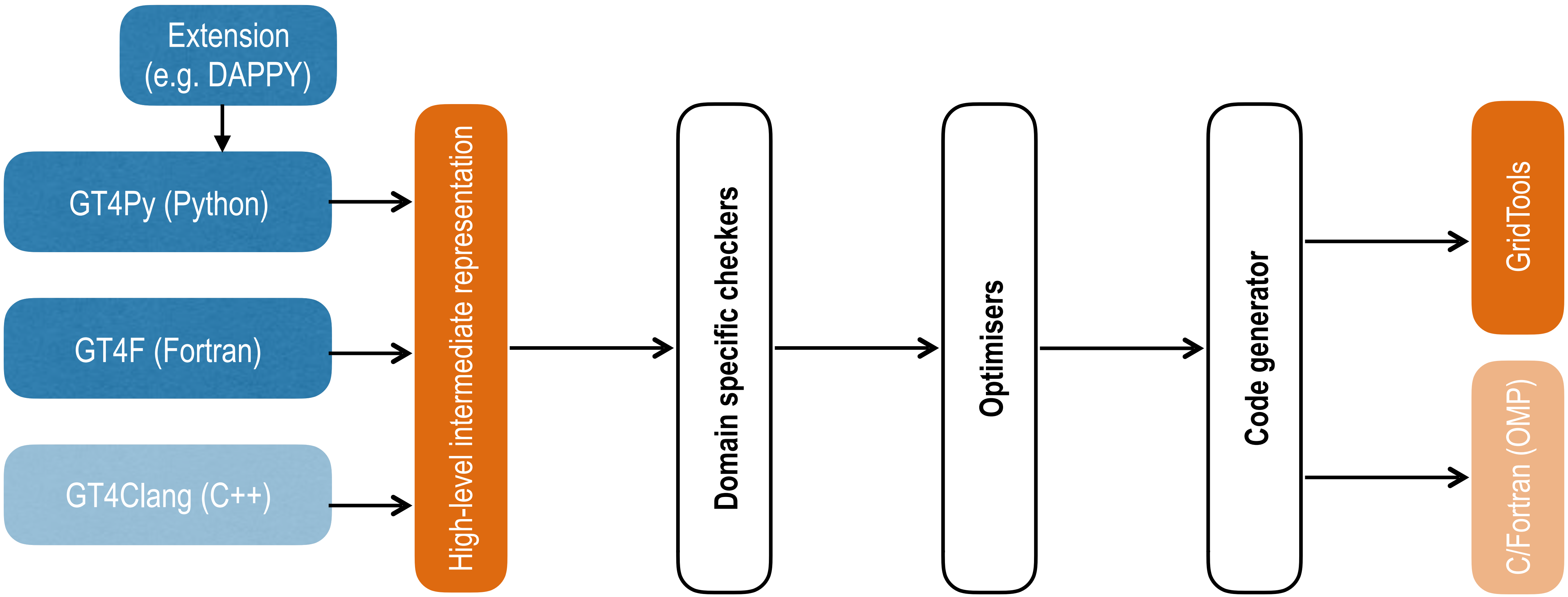
Atlas @ ECMWF

ICON-Ocean @ MPI-M



ETH zürich

Toolchain

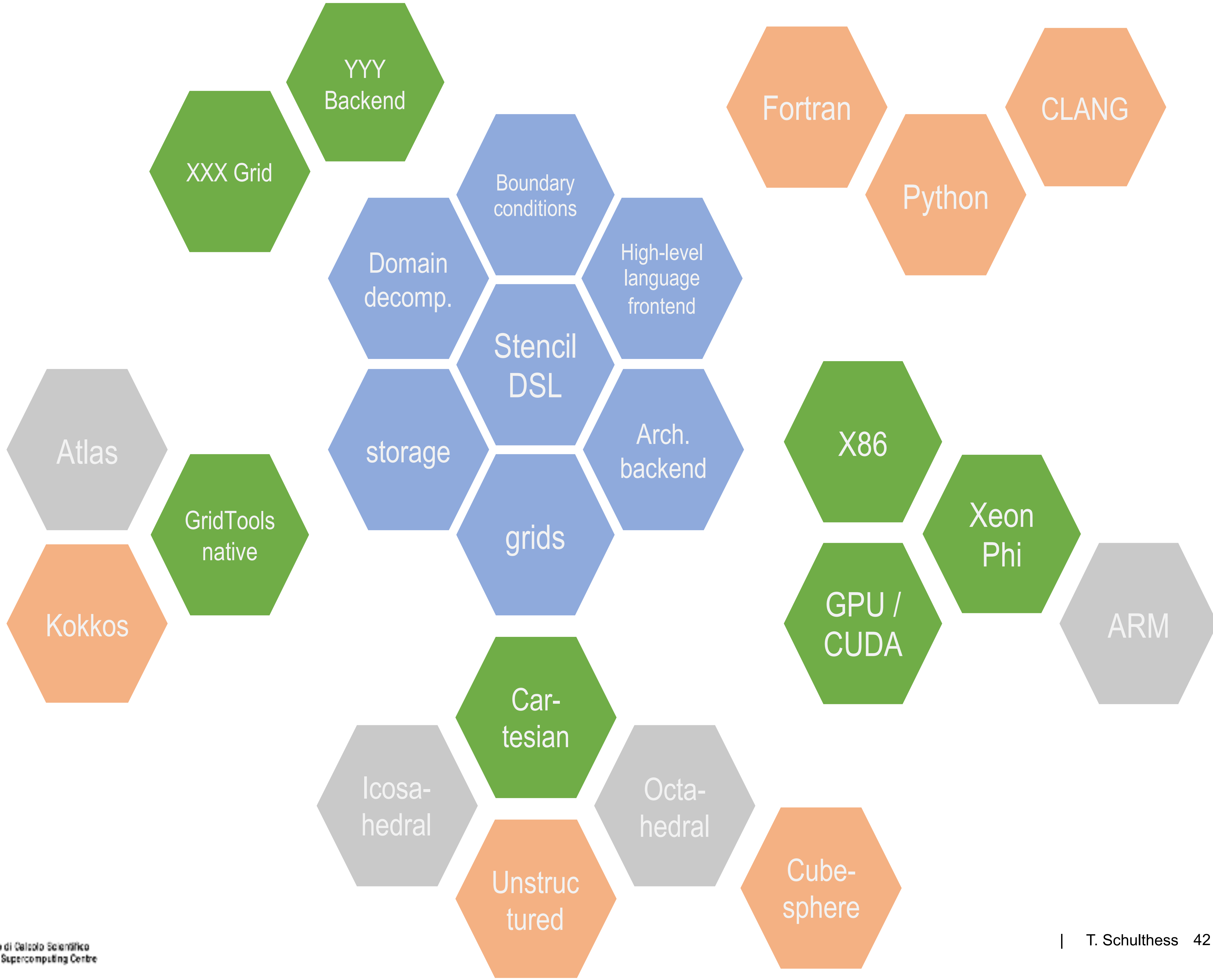


- read before write
- missing boundary update
- data dependency race conditions
- out of bounds stencil access

- Software managed caches
- Full vertical parallelisation
- Stage fusion
- Data locality exploit
- Strong/weak scaling optimiser

- Native C/C++/Fortan
- Optimised GridTools Generator (C++)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre


YYY Backend
XXX Grid
Fortran
CLANG
Python
Boundary conditions
Domain decomp.
High-level language frontend
Stencil DSL
storage
Arch. backend
grids
Atlas
GridTools native
Kokkos
X86
Xeon Phi
GPU / CUDA
ARM
Car-tesian
Icosa-hedral
Octa-hedral
Unstruc tured
Cube-sphere

The GridTools Team

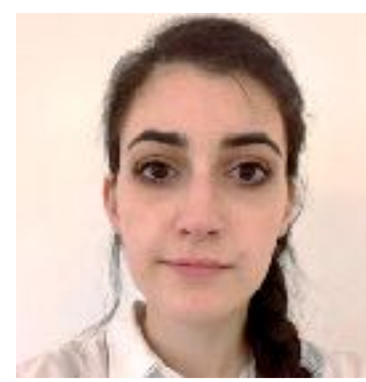
Anton Afanasyev
Software Architect

Hannes Vogt
Dycore Lead, CUDA
Scrum Master

Carlos Osuna
Other Grids, CUDA
Product Owner

Mauro Bianco
Stencil Composition
Technical Lead

Chritopher Bignamini
Dycore (no full time)



Nora Abi Akar
ARM Backend

Felix Thaler
KNL Backend

Stefan Moosbrugger
KNL, Storage

What exactly is "exascale" computing?

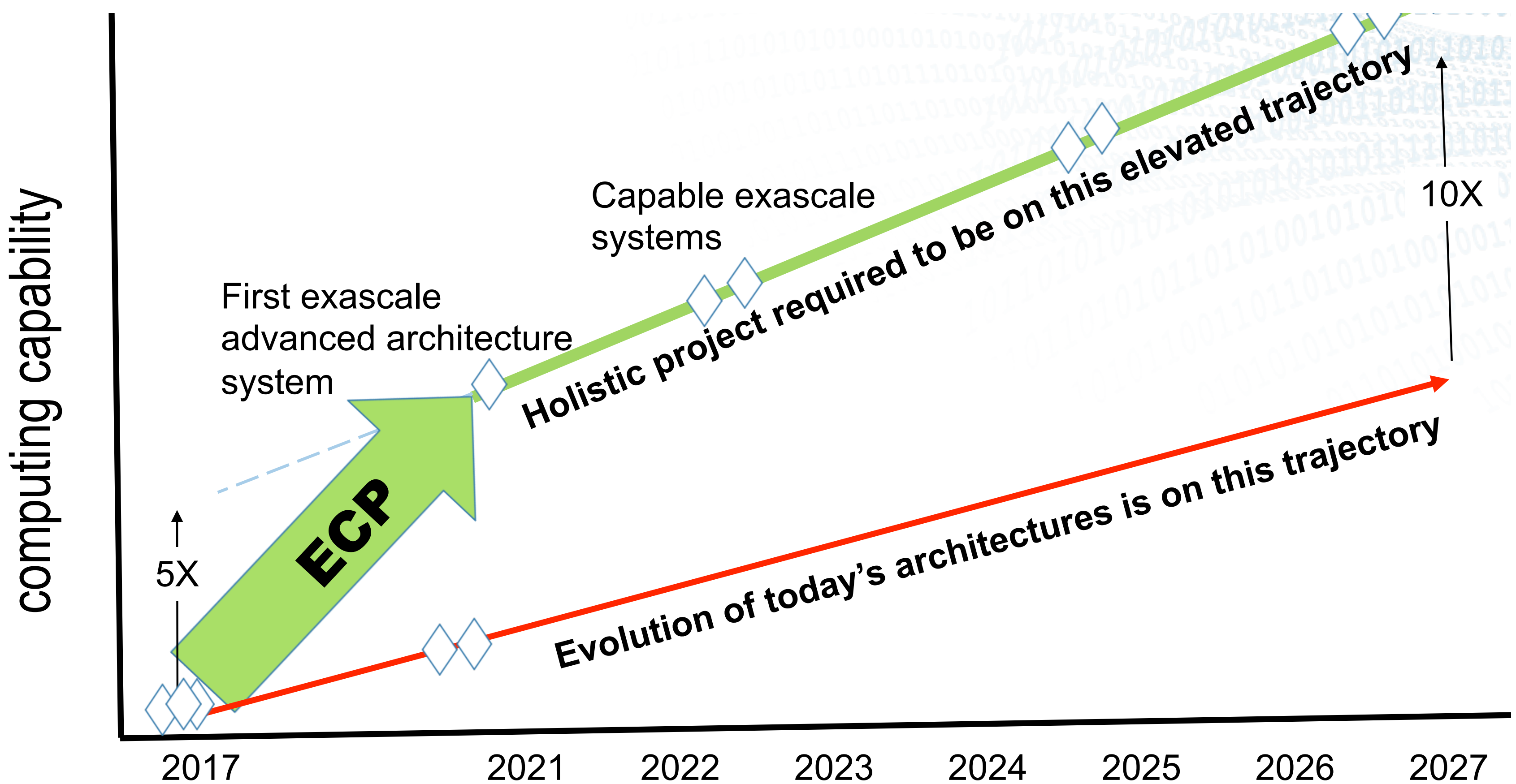
Paul Messina, US-DoE

EXASCALE COMPUTING PROJECT

Exascale is not just ~10^3 PF

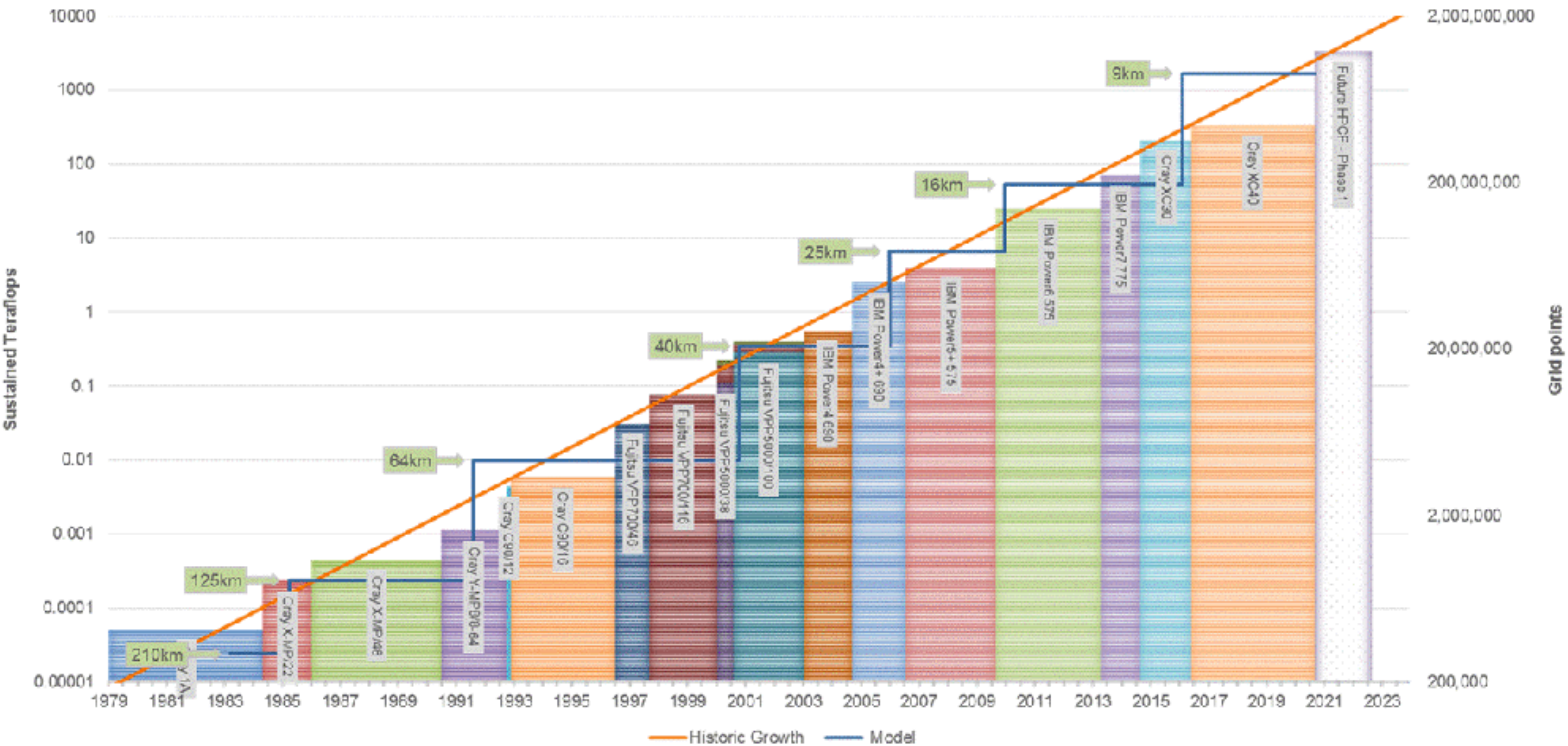
A capable exascale computing system requires an entire computational ecosystem that:

- Delivers 50x the performance of today's ~20 PF systems, supporting application that deliver high-fidelity solutions in less time and address problems of greater complexity
- Operates in a power envelope of 20-30 MW
- Is sufficiently resilient (perceived full rate: <= 1/week
- Includes a software stack that supports a broad spectrum of applications and workloads



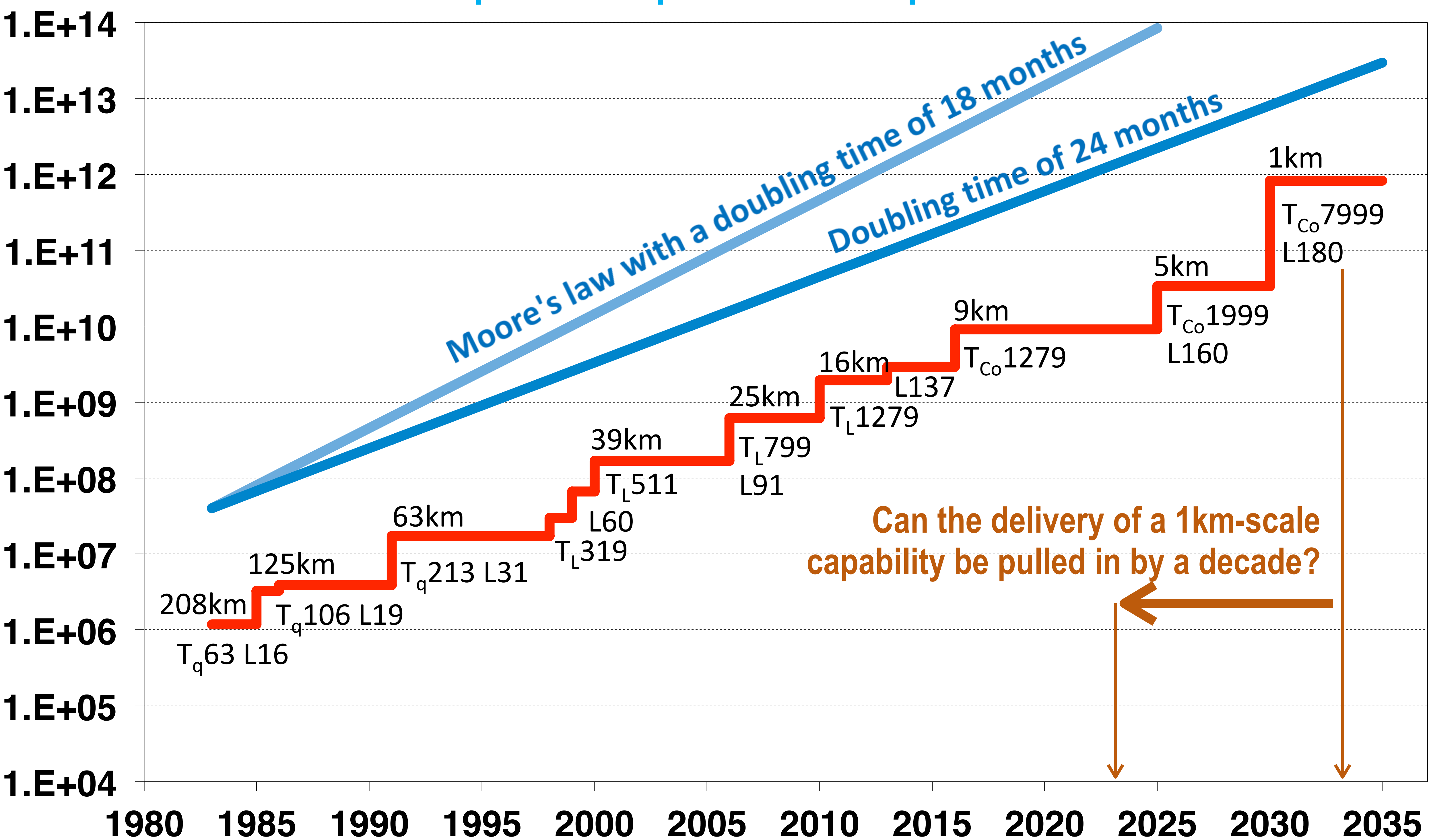
Transition to higher trajectory with advanced architecture

But what is the goal for exascale computing, and the baseline?



Sustained Teraflops
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
0.00001
Grid points
2,000,000,000
200,000,000
20,000,000
2,000,000
200,000
1979
1981
1983
1985
1987
1989
1991
1993
1995
1997
1999
2001
2003
2005
2007
2009
2011
2013
2015
2017
2019
2021
2023
210km
125km
64km
40km
25km
16km
9km
Cray X-MP/22
Cray X-MP/48
Cray Y-MP/8-64
Cray C90/12
Cray C90/16
Fujitsu VPP700/46
Fujitsu VPP700/116
Fujitsu VPP5000/38
Fujitsu VPP5000/100
IBM Power-4 690
IBM Power4+ 690
IBM Power5+ 575
IBM Power6 575
IBM Power7 775
Cray XC30
Cray XC40
Future HPCF - Phase 1
Historic Growth
Model

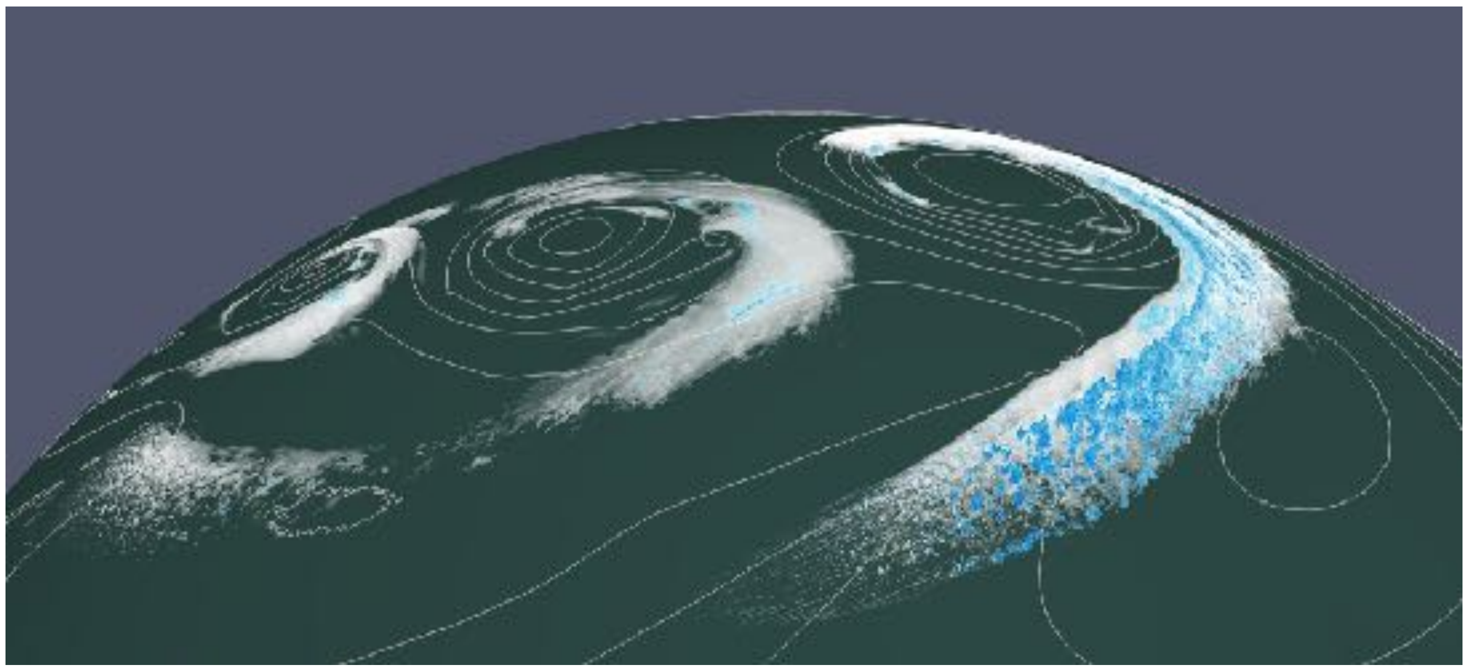
Computational power drives spatial resolution

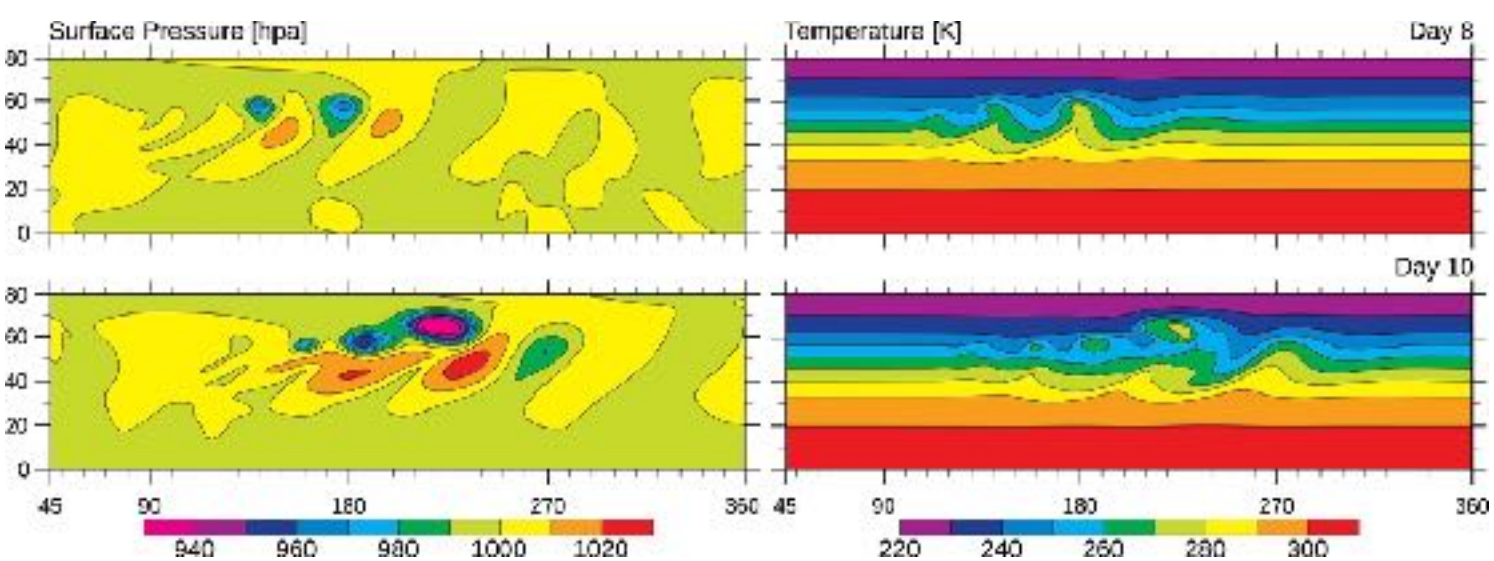


Let’s assume for a moment we can build on the CSCS-MCH experience

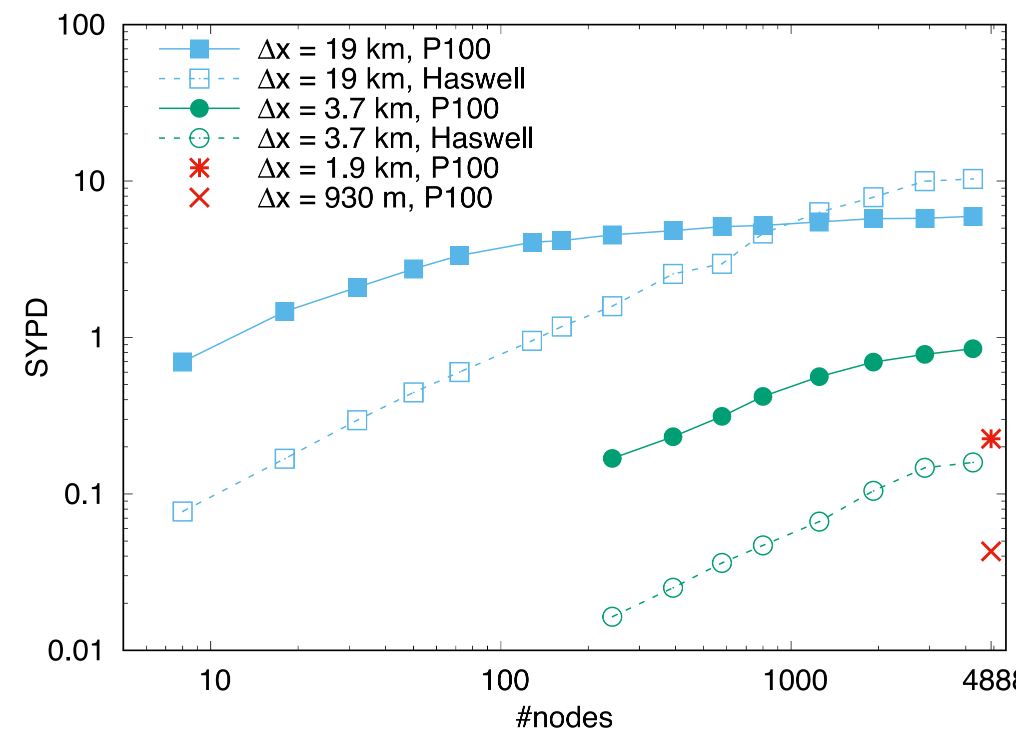
Near-global climate simulation at 1km resolution: establishing a performance baseline on 4888 GPUs with COSMO 5.0

Fuhrer et al., Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2017-230, published 2018





Metric: simulated years per wall-clock day



$\langle \Delta x \rangle$	#nodes	Δt [s]	SYPD	MWh/SY	gridpoints
930 m	4,888	6	0.043	596	3.46×10^{10}
1.9 km	4,888	12	0.23	97.8	8.64×10^{9}
47 km	18	300	9.6	0.099	1.39×10^{7}

(c) Time compression (SYPD) and energy cost (MWh/SY) for three moist simulations. At 930 m grid spacing obtained with a full 10d simulation, at 1.9 km from 1,000 steps, and at 47 km from 100 steps

2.5x faster than Yang et al.'s 2016 Gordon Bell winner run on TaihuLight!

"Exascale" goal for global weather and climate runs

Horizontal resolution	1 km (globally quasi-uniform)
Vertical resolution	180 levels (surface to ~100 km)
Time resolution	0.5 minutes
Coupled	Land-surface/ocean/ocean-waves/sea-ice
Atmosphere	Non-hydrostatic
Precision	Single or mixed precision
Compute rate	1 SYPD (simulated years per wall-clock day)

The baseline for COSMO-global and IFS

	Near-global COSMO [Fuh2018]		Global IFS [Wed2009]	
	Value	Shortfall	Value	Shortfall
Horizontal resolution	0.93 km (non-uniform)	0.81x	1.25 km	1.56x
Vertical resolution	60 levels (surface to 25 km)	3x	62 levels (surface to 40 km)	3x
Time resolution	6 s (split-explicit with sub-stepping)*	-	120 s (semi-implicit)	4x
Coupled	No	1.2x	No	1.2x
Atmosphere	Non-hydrostatic	-	Non-hydrostatic	-
Precision	Double	0.6x	Single	-
Compute rate	0.043 SYPD	23x	0.088 SYPD	11x
Other (e.g. physics, ...)	microphysics	1.5x	Full physics	-
Total shortfall		**60x**		**247x**

Memory use efficiency

Fuhrer et al., Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2017-230, published 2018

Necessary data transfers

0.88

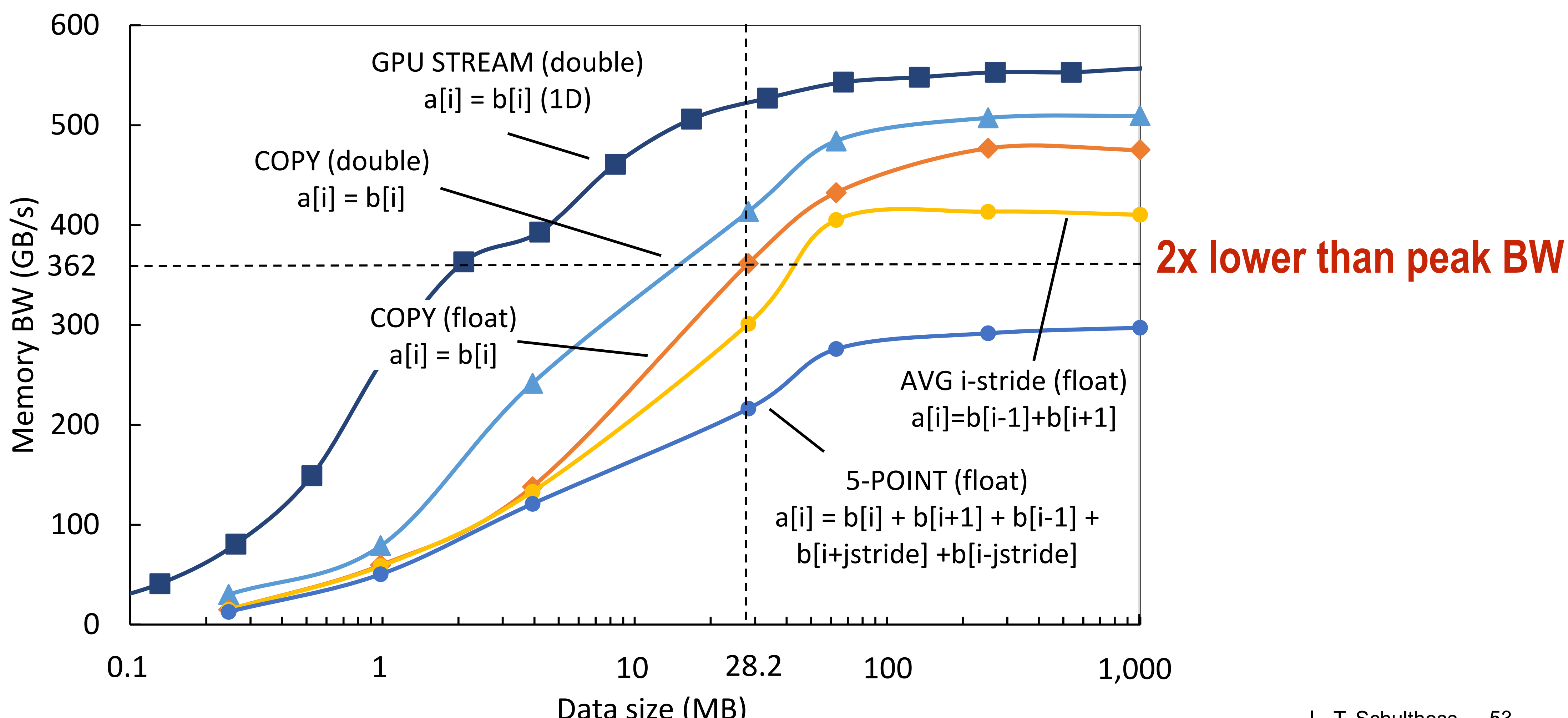
Achieved BW

$$MUE = \text{I/O efficiency} \cdot \text{BW efficiency} = \frac{Q}{D}\frac{B}{\hat{B}} = 0.67$$

Actual data transfers

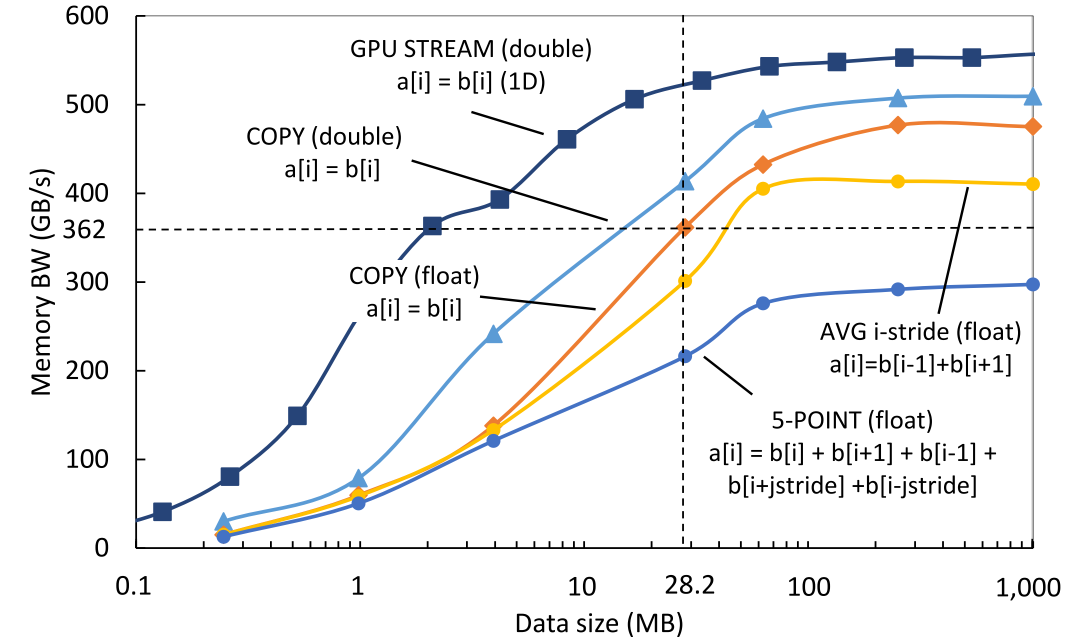
0.76

Max achievable BW



ETH zürich

How realistic is it to overcome 65-fold shortfall of a grid-based implementation like COSMO-global?



1. Icosahedral grid (ICON) vs. Lat-long/Cartesian grid (COSMO)

    2x fewer grid-columns

    Time step of 10 ms instead of 5 ms

    4x

2. Improving BW efficiency

    Improve BW efficiency and peak BW (results on Volta show this is realistic)

    1.5x

3. Weak scaling

    4x possible in COSMO, but we reduced available parallelism by factor 2

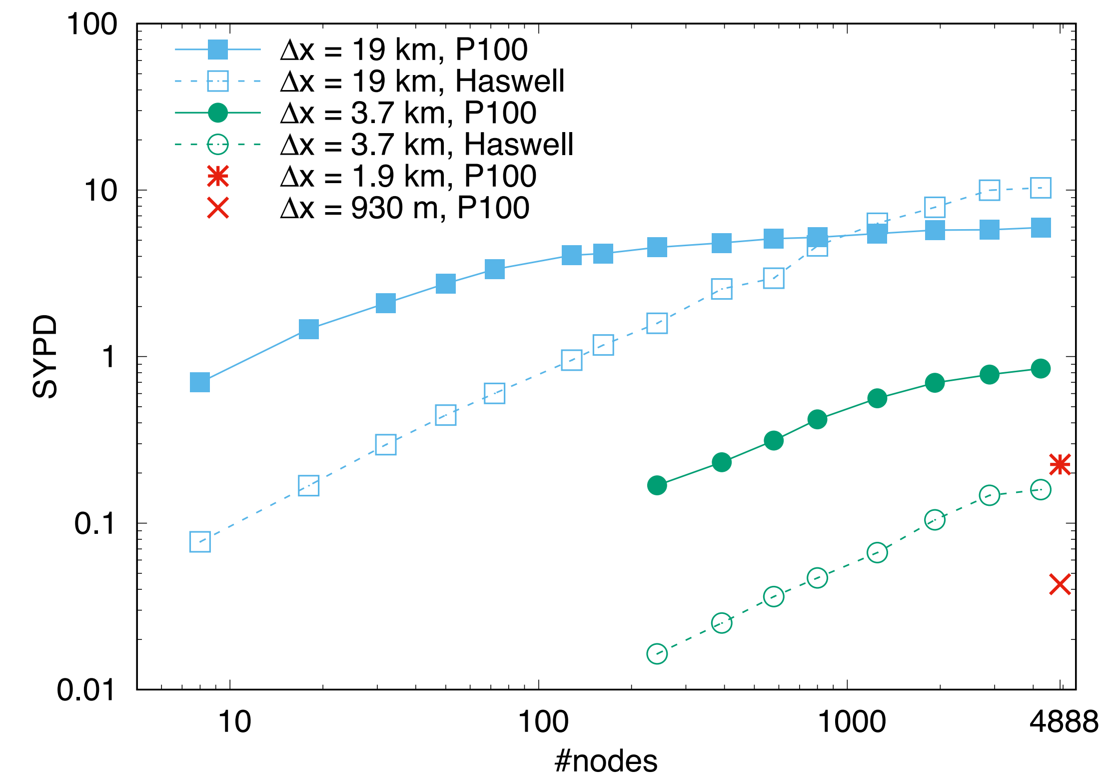
    2x

4. Remaining reduction in shortfall

    5x

    Numerical algorithms (larger time steps)

    Further improved processors / memory



But we don’t want to increase the footprint of the 2021 system beyond “Piz Daint”

The main conclusions

- Change is nothing new to HPC, nor is the reluctance to adapt to change
    - “Killer micros”, memory wall, end of Dennard Scaling and multi-core, GPU
- CMOS scaling tapering due to constraints in device physics and fabrication
- Architectural improvements & diversity seem a good option to improve performance
- New opportunities for materials science and device physics?
- Fundamental challenge to software / application development
- Domain specific libraries and frameworks are a way out
    - GridTools framework with successful demonstration to COSMO @ MeteoSwiss
- “Exascale” computing, if properly defined and pursued, could give us ~1km scale horizontal resolution in simulation with good throughput

Great motivations to clean up our software stack!

ETH zürich

Collaborators


Tim Palmer (U. of Oxford)


Bjorn Stevens (MPI-M)


Peter Bauer (ECMWF)


Oliver Fuhrer (MeteoSwiss)


Nils Wedi (ECMWF)


Torsten Hoefler (ETH Zurich)


Christoph Schar (ETH Zurich)