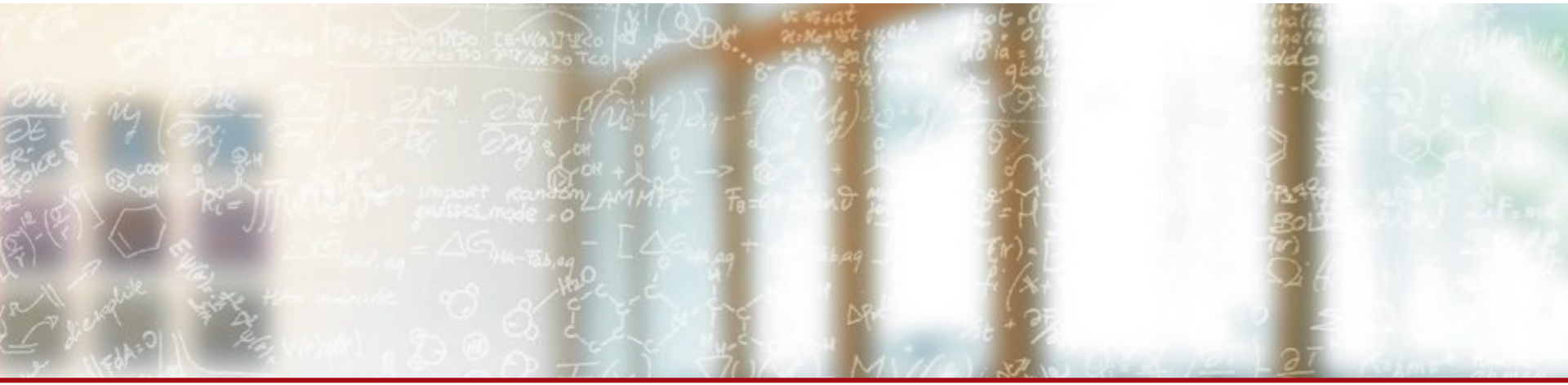
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Developing scientific software for modern high performance computing

Joost VandeVondele, ETHZ, CSCS

June 2019

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Introduction: CSCS & SSL

CSCS : Swiss National Supercomputer Centre

Early adopter of GPU technology for HPC

Flagship : Piz Daint : one of the most powerful supercomputers in the world
5700 GPU accelerated nodes, Aries network, Cray XC50
1800 Dual socket intel nodes, XC40

CSCS is accessible for scientists worldwide via open peer reviewed calls:
Excellent science and efficient use of the GPU accelerated infrastructure
- ‘small’ proposals (<1M node hours) → via national call
- ‘large’ proposals (>=1M node hours) → via PRACE

CSCS : HQ Lugano & ETH Zurich

Lugano

Zurich

Scientific Software and Libraries (SSL) @ CSCS

Develop and modernize libraries and software tools to enable scientific applications to run at scale on different hardware architectures.

30 people, located at ETH Zurich and Lugano

Scientific Software & Libraries, in various areas:

- Weather and Climate
- Neuroscience
- PDEs: E.g. earth sciences
- Material Science
- Linear Algebra
- Programming models

Long term engagements with academia, and related institutions,
Engaged in Swiss & European projects:
HPCN & PASC, NCCR Marvel, CoE MaX, HBP, PRACE

HPC evolution and ecosystem

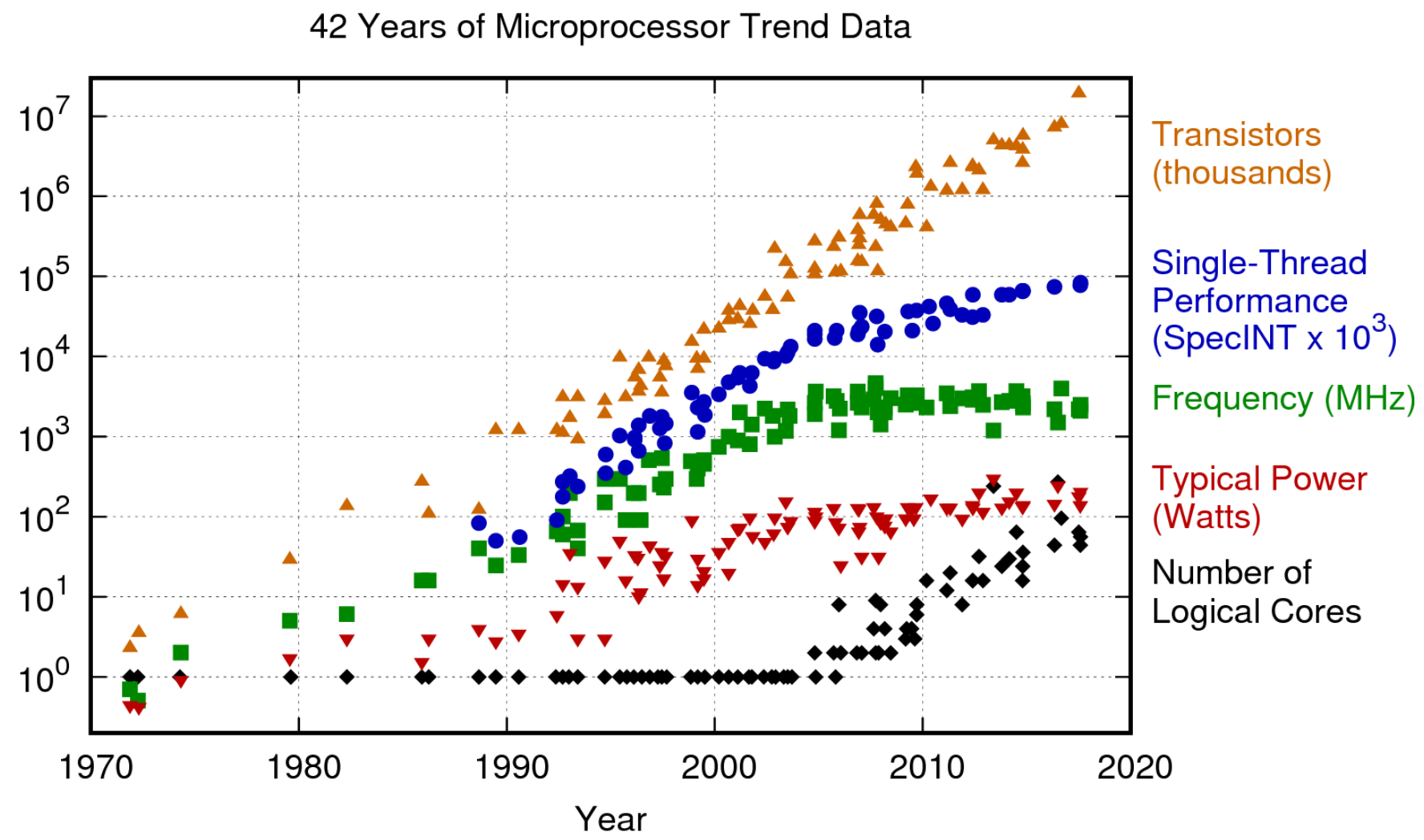
Top 500 number 1 in 1997 and today‘s `equivalent`

Asci Red (1 TF, 1MW in 1997)

Intel® Xeon® Platinum 8180,
>1 TF, 200W, in 2017

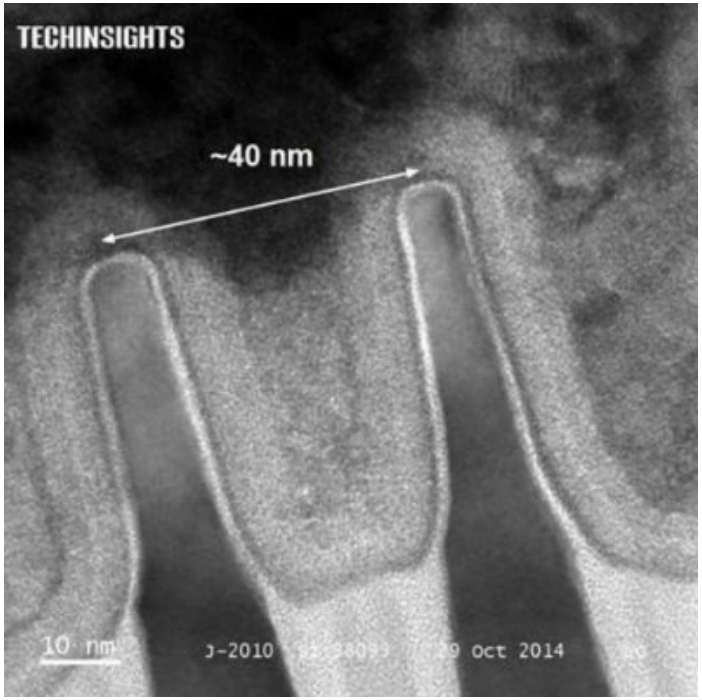
Amazing progress in computing …. but what is the outlook ?

Towards the end of Moore‘s law



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 2010-2017 by K. Rupp

Ultimately physics (and economics) will put the limits..

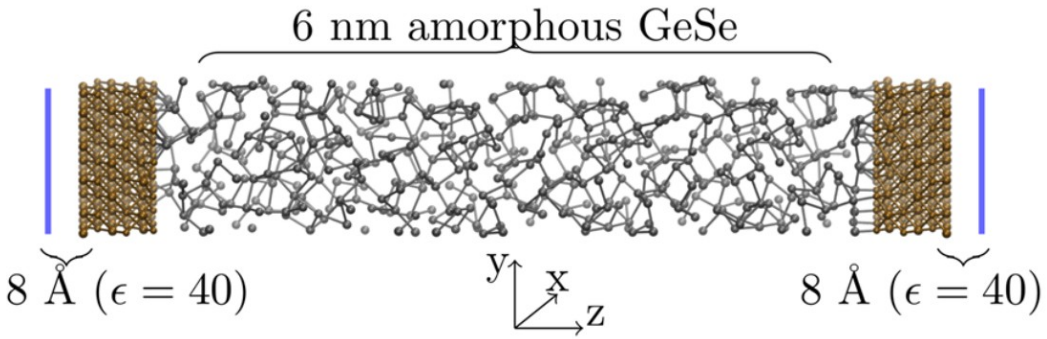


IBM 7nm process (2014)

THE JOURNAL OF CHEMICAL PHYSICS **149**, 124701 (2018)

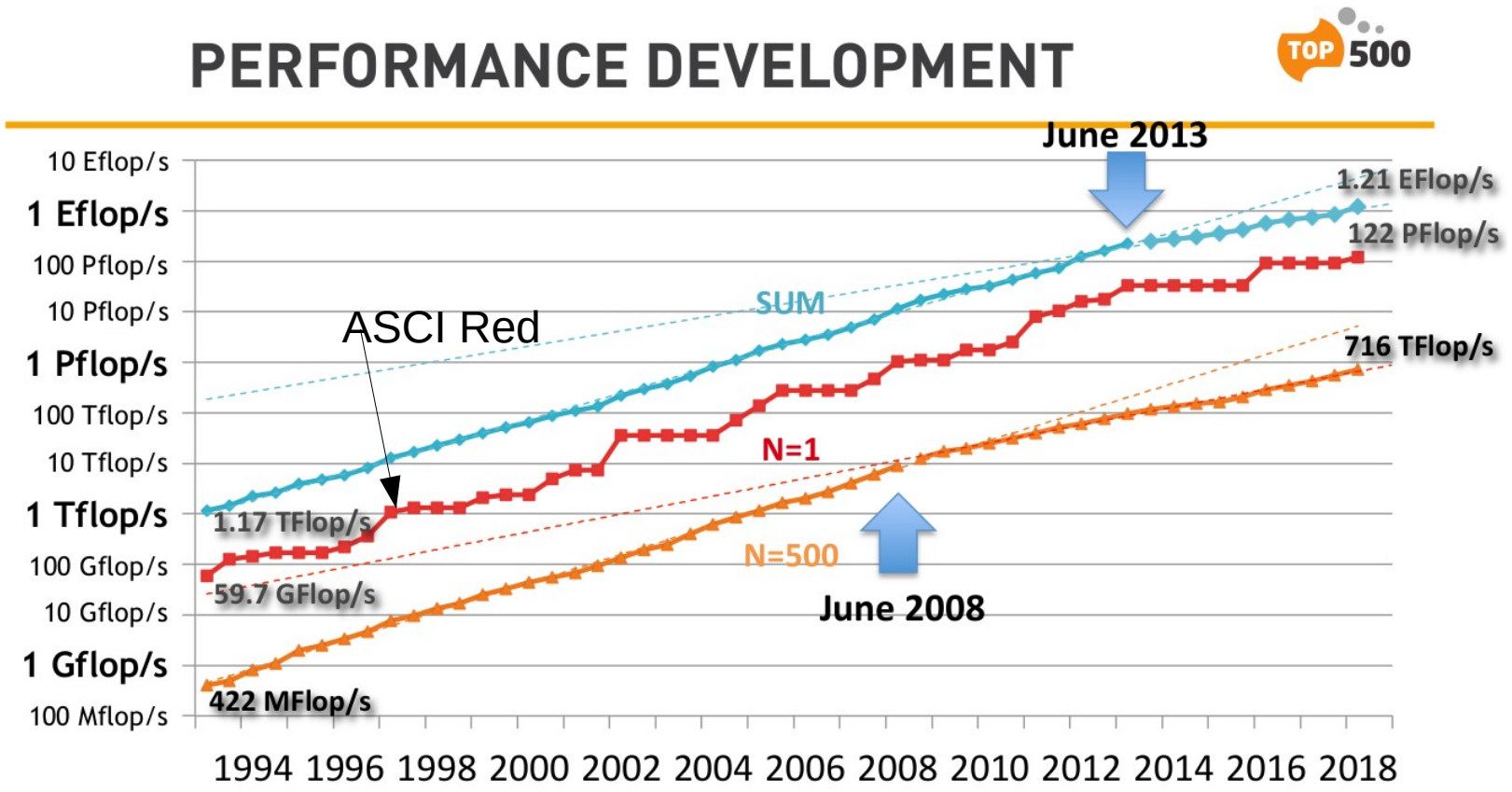
Microcanonical RT-TDDFT simulations of realistically extended devices

Samuel Andermatt,[1,a)] Mohammad Hossein Bani-Hashemian,[1] Fabian Ducry,[1] Sascha Brück,[1] Sergiu Clima,[2] Geoffrey Pourtois,[2] Joost VandeVondele,[3] and Mathieu Luisier[1]
[1]*Integrated Systems Laboratory, ETH Zürich, 8092 Zürich, Switzerland*
[2]*IMEC, 75 Kapeldreef, B-3001 Leuven, Belgium*
[3]*Swiss National Supercomputing Centre (CSCS), ETH Zürich, 8093 Zürich, Switzerland*



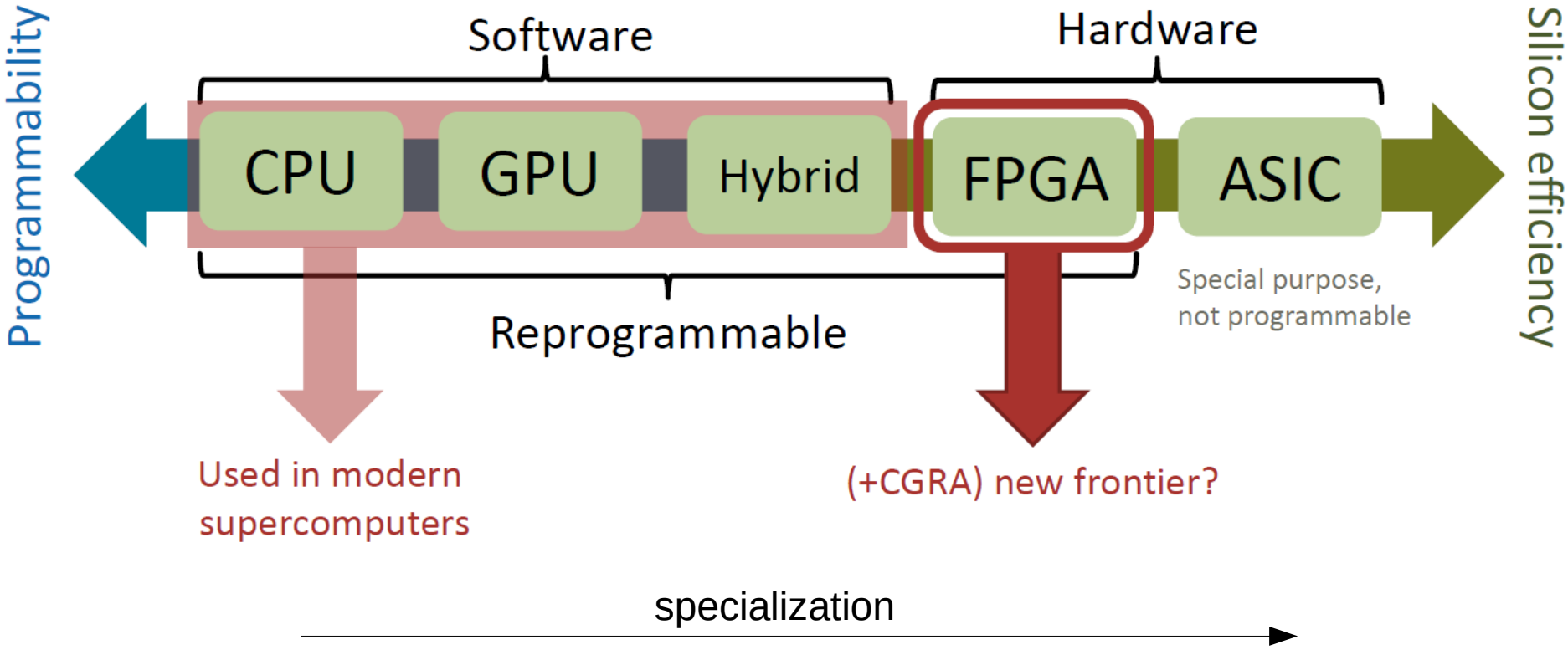
Getting in the explicit atomistic range, every atom counts.

High Performance Linpack (HPL*) / Top500



*A benchmark, with nice consistent data, excellent to stress-test machines… but measure systems by their scientific output.

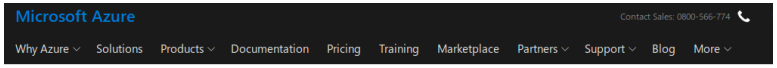
New hardware: ease of programmability vs. efficiency



Extreme specialization is real, the hyperscalers are fast:

Intel Launches FPGA Accelerator Aimed at HPC and HPDA Applications

Michael Feldman | December 20, 2017 12:44 CET



Inside the Microsoft FPGA-based configurable cloud

Microsoft has been deploying FPGAs in every Azure server over the last several years, creating a cloud that can be reconfigured to optimize a diverse set of applications and functions.

Google Announces 8x Faster TPU 3.0 For AI, Machine Learning

By Joel Hruska on May 9, 2018 at 10:30 am | 15 Comments

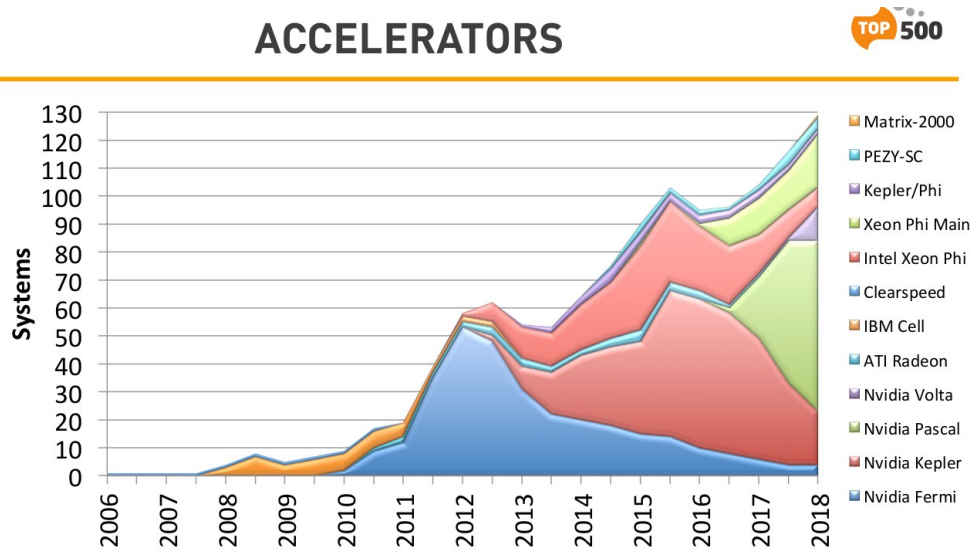
297 SHARES

2. TPU Origin, Architecture, and Implementation

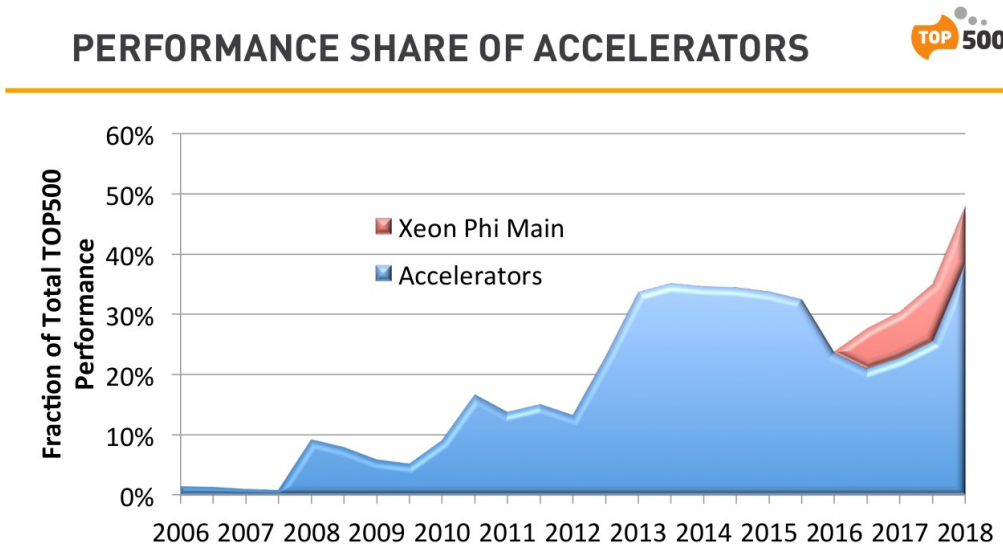
Starting as early as 2006, we discussed deploying GPUs, FPGAs, or custom ASICs in our datacenters. We concluded that the few applications that could run on special hardware could be done virtually for free using the excess capacity of our large datacenters, and it's hard to improve on free. The conversation changed in 2013 when a projection where people use voice search for 3 minutes a day using speech recognition DNNs would require our datacenters to double to meet computation demands, which would be very expensive to satisfy with conventional CPUs. Thus, we started a high-priority project to quickly produce a custom ASIC for inference (and bought off-the-shelf GPUs for training). The goal was to improve cost-performance by 10X over GPUs. Given this mandate, the TPU was designed, verified [Ste15], built, and deployed in datacenters in just 15 months. (Space limits the amount and the level of detail on the TPU in this paper; see [Ros15a], [Ros15b], [Ros15c], [Ros15d], [Tho15], and [You15] for more.)

https://arxiv.org/pdf/1704.04760.pdf

TOP500 Acceleration data



Increasing share of accelerated systems



Nearly 50% of performance from accelerated computing

Example modern accelerator, NVIDIA V100

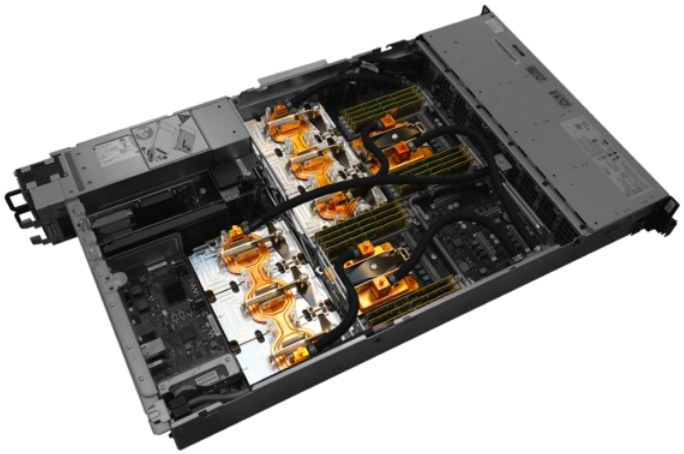
7 TF DP
14 TF SP

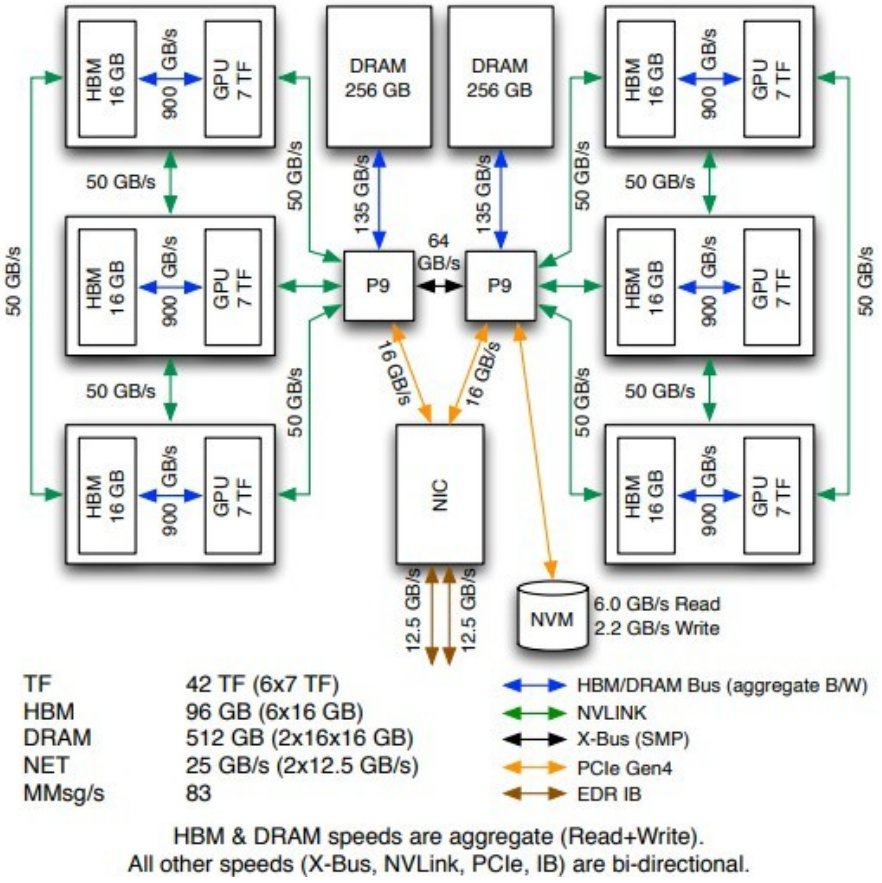
900GB/sec

112 TF Tensor cores

CUDA programming to get access to all features.

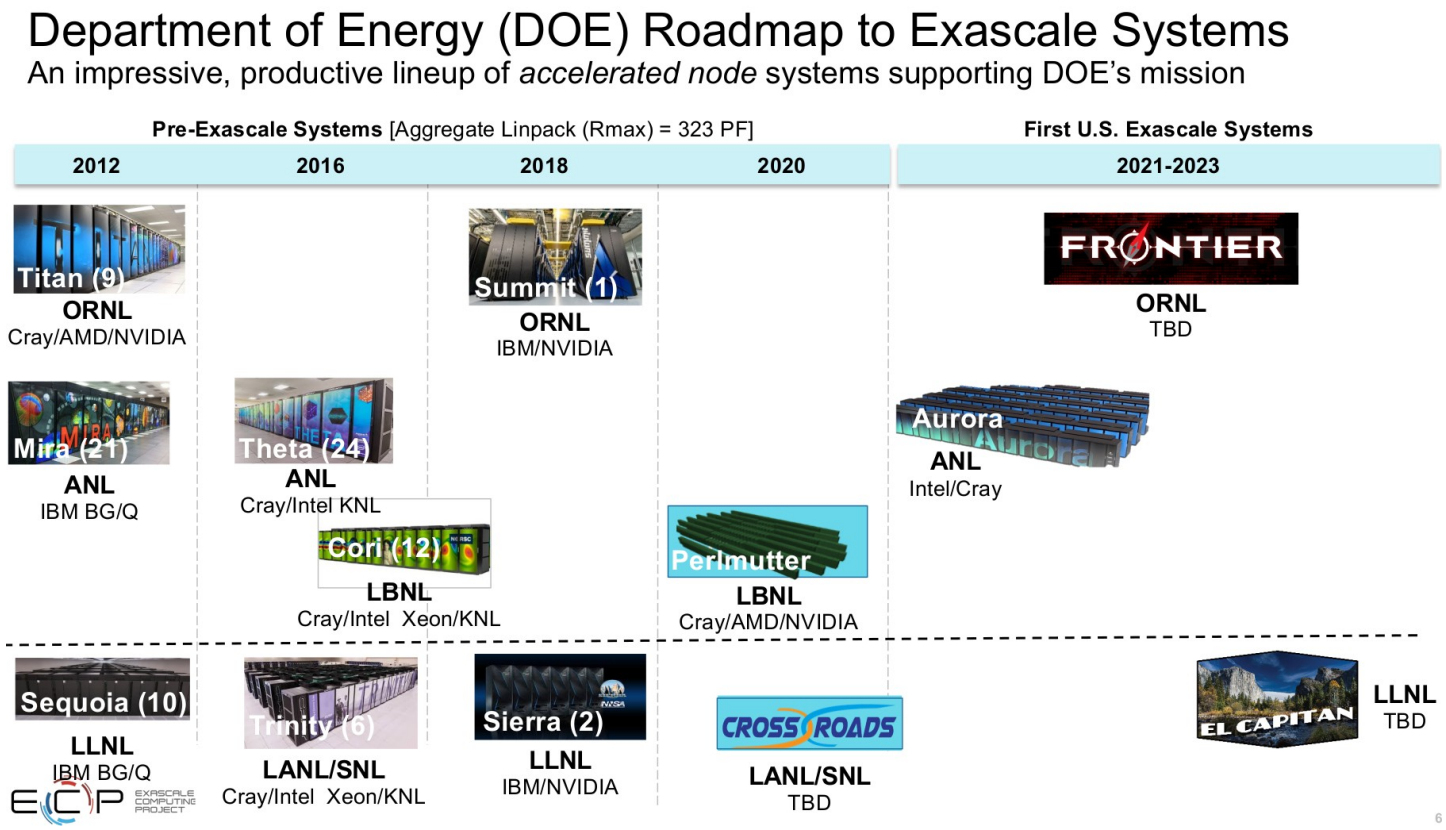
Summit (ORNL, TOP500 #1): modern HPC nodes





Multiple accelerators, many different data movement bandwidths.
Understanding where data is, where it has to go, and when to send it, is key.

US Exascale roadmap



Accelerated systems by various processor vendors (NVIDIA, AMD, Intel)

Asia with two players

Japan

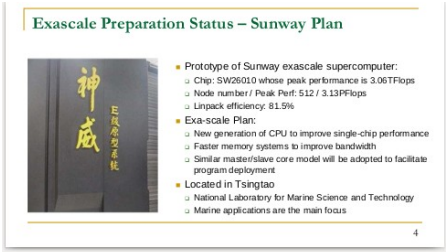
Fugaku

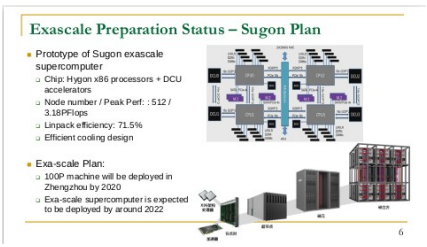
Arm64fx & Post-K (to be renamed)

- **Fujitsu-Riken design A64fx ARM v8.2 (SVE), 48/52 core CPU**
  - *HPC Optimized:* Extremely high package high memory BW (1TByte/s), on-die Tofu-D network BW (~400Gbps), high SVE FLOPS (~3Teraflops), various AI support (FP16, INT8, etc.)
  - Gen purpose CPU – Linux, Windows (Word), other SCs/Clouds
  - Extremely power efficient – > 10x power/perf efficiency for CFD benchmark over current mainstream x86 CPU
- **Largest and fastest supercomputer to be ever built circa 2020**
  - > 150,000 nodes, superseding LLNL Sequoia
  - **> 150 PetaByte/s memory BW**
  - Tofu-D 6D Torus NW, **60 Petabps injection BW (10x global IDC traffic)**
  - 25~30PB NVMe L1 storage
  - ~10,000 endpoint 100Gbps I/O network into Lustre
  - The first 'exascale' machine (not exa64bitflops but in apps perf.)

3

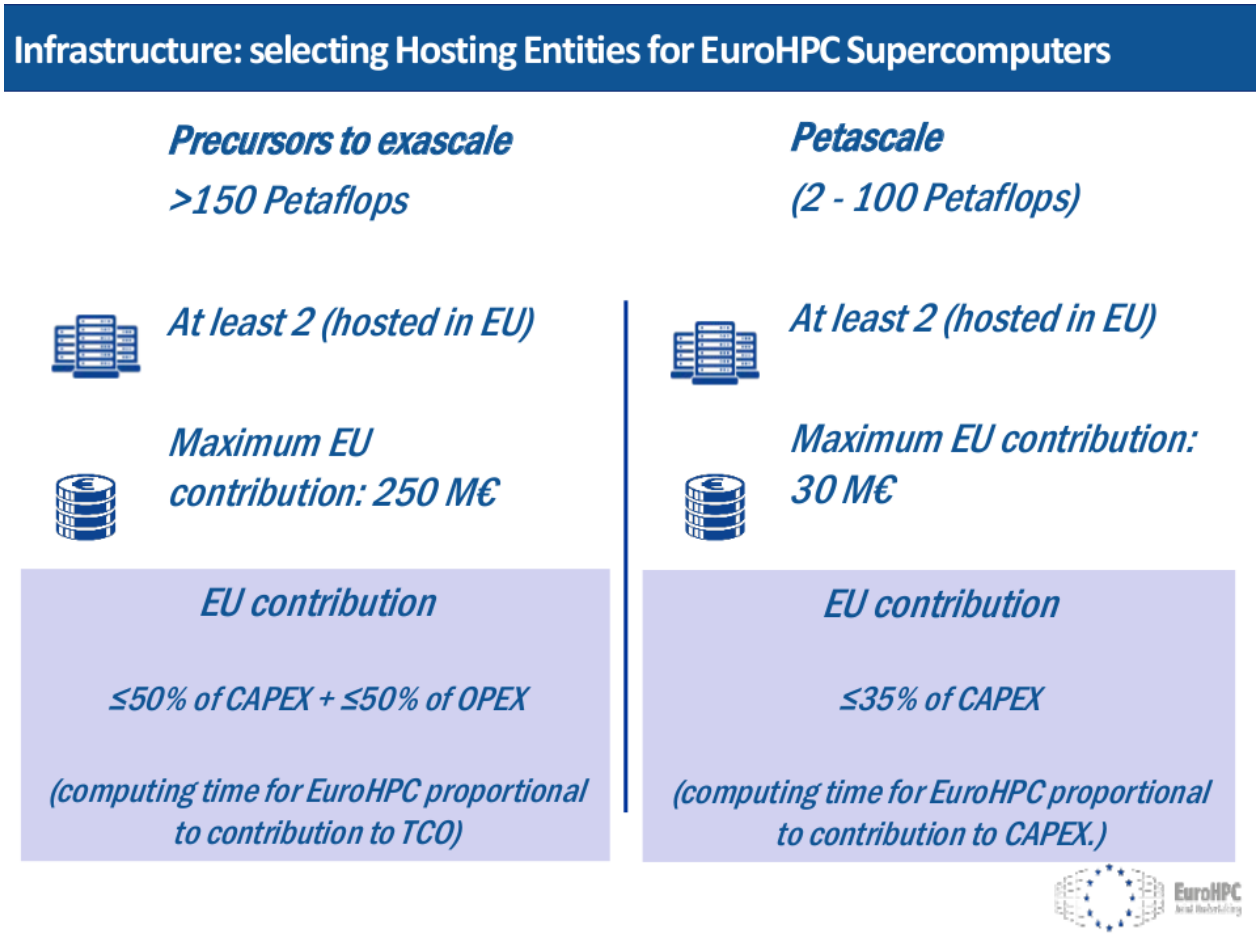
China





Very large node counts, non-x86, some accelerated, prototypes running

European plans

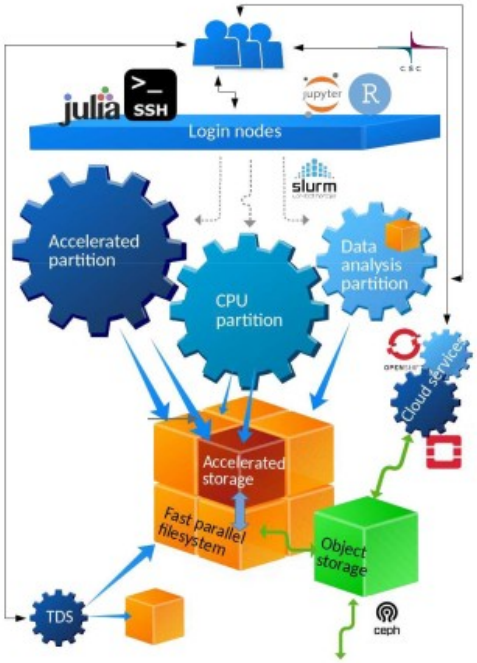


- To be decided: Early June 2019
- To be operational: End of 2020
- Very likely: Accelerated architectures

The LUMI consortium

LUMI: Our vision of the EuroHPC system

- The LUMI system consists of **a pre-exascale (200+ Pflop/s) accelerated supercomputer,** which enables the convergence of **high-performance computing, artificial intelligence,** and **high-performance data analytics.**
- The system will be supplemented by a **number of supporting compute and storage resources,** maximizing its overall value. These will include an additional CPU partition, data analysis partition and cloud resources.
- Maximized **user experience**: interactive supercomputing, easy user interfaces, containerization, pronounced support for data management.

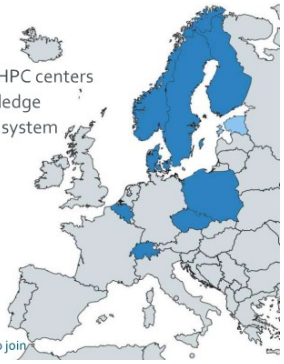


LUMI consortium

- Unique consortium of 9 countries with strong national HPC centers and competence gives an unique opportunity for knowledge transfer and sharing and providing user support for the system
  - We welcome further countries to join the consortium

Finland	50 M€	Norway	4 M€
Belgium	15.5 M€	Poland	5 M€
Czech Republic	5 M€	Sweden	7 M€
Denmark	6 M€	Switzerland	10 M€
Estonia*	2 M€	EU	102.5 M€

The Netherlands has provided an official letter of support to the LUMI consortium.
* Estonia's 2 M€ will be commited during the period for additional states to join



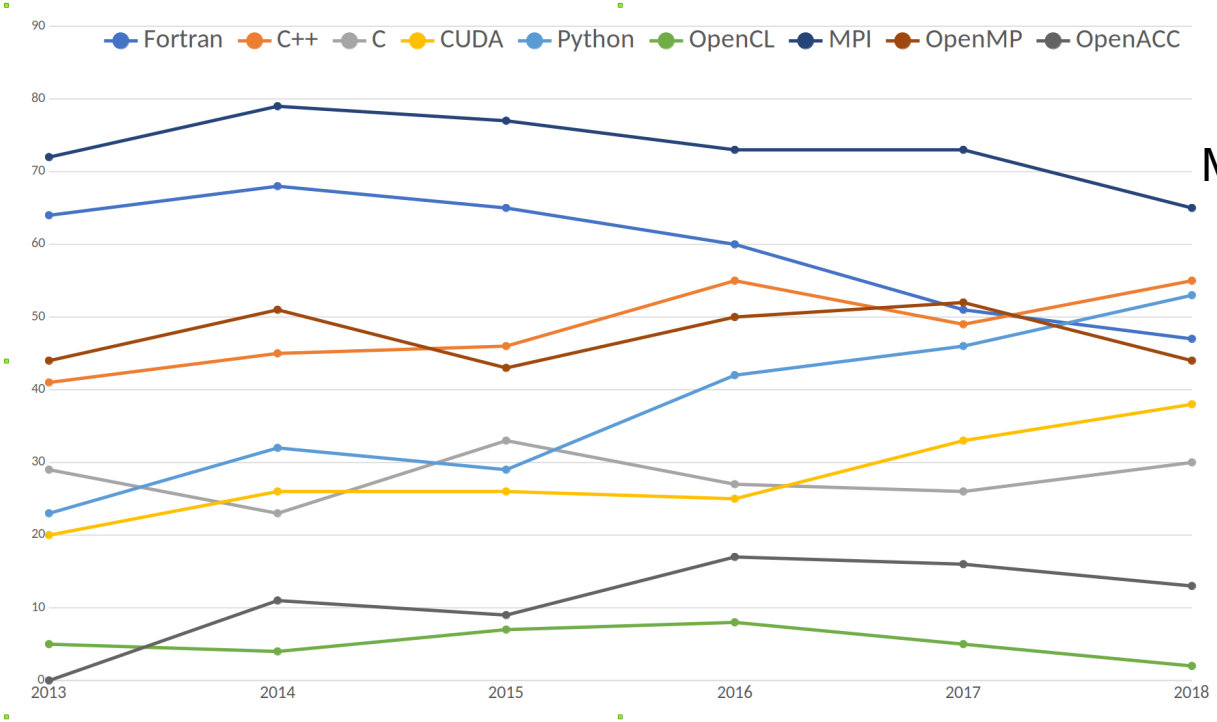
Switzerland and Belgium might soon be partners in one of the largest European supercomputer infrastructures.

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Software development

Evolution of used Languages / Tools at CSCS



MPI

Python

C++

CUDA

Fortran

OpenACC (COSMO, VASP), openCL

- C++ or Python > Fortran in 2018
- Python more than doubled
- CUDA increases steadily, OMP stagnant
- Limited openCL, openACC

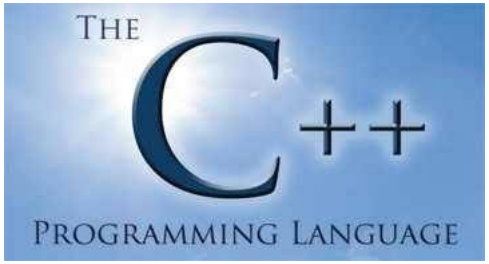
All students know MPI, python,
CSE students += C++ & Cuda

Adopt the tools of the big players in IT

Think of what happened to the vector computers when the cheap microprocessors came out!

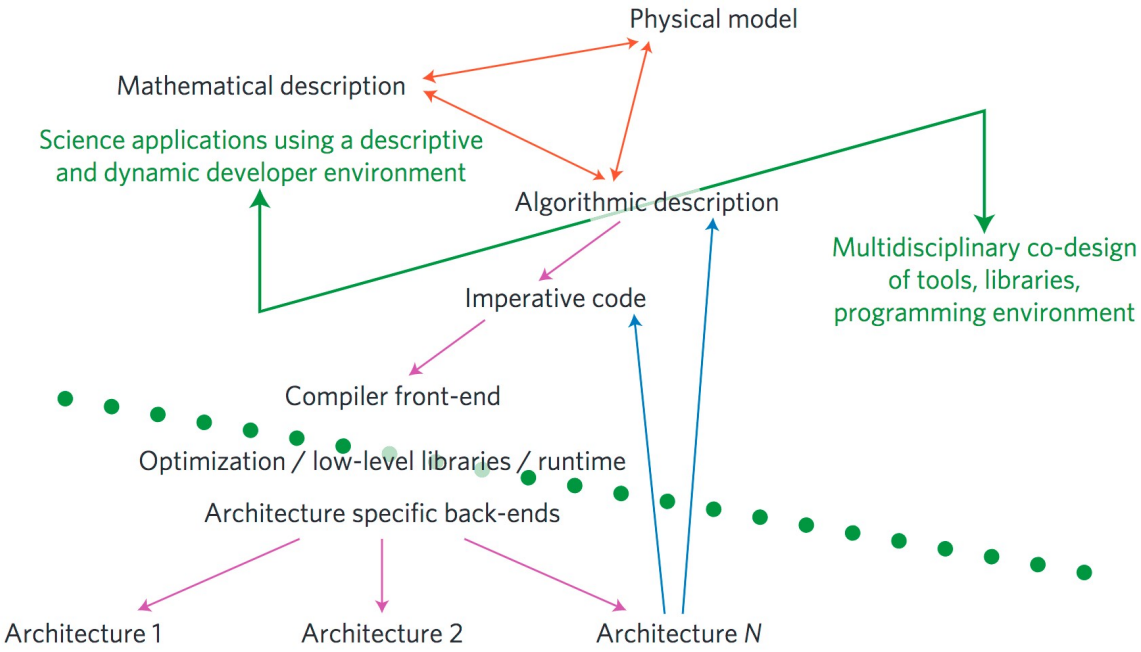
~ 1995 - 2000 use of commodity hardware in HPC

~ 2015 – 2025 use of commodity software in HPC



It will be increasingly more difficult for traditional HPC tools (Fortran / OMP / etc) to follow the rapid evolution. Students will have different background and interests.

Software development with separation of concerns



Separate software in performance portable backends and rapid prototyping frontends

Requires a change in how the community writes and maintains software

Thomas C. Schulthess

Nature Physics, 11 (5): 369-373, London: Nature Publ. Group, 2015.

Programming for portability ?

Challenging engineering, not just adopting a single programming model

- Data layout
- Iteration ordering
- Maintainability
- Recompute vs store - load

Separate what vs how, and specialize ‘how’ as needed

- I need to store a 3D array
  → create a storage (how: use tiles, padding, etc).
- I need a triple loop over the data
  → use an iterator (how: blocked, unrolled, fused, ...)

The optimal solution will be hardware dependent. Increasingly data motion, not compute will be the key.

Tools and strategies

Adoption of C++ and python as languages
- C++ : good performance, large ecosystem, generic, strongly-typed
- python : rapid prototyping, large ecosystem, versatile, interactive

Use of vendor specific extensions as needed, but well separated
- right now : CUDA
- future: ROCM, OpenCL, SYCL, …

C++ ecosystem provides useful tools for reuse e.g.
- Kokkos
- TBB
- HPX

Adoption of (embedded) domain specific languages (eDSL), and domain specific libraries (reuse).

!$OMP (target??)/pragmas

Even if the maintainability of OMP were not an issue, performance is:

- What when my data layout needs to change to make good use of the hardware

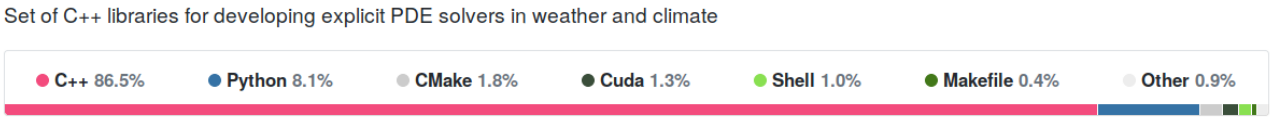
- OMP doesn't give me access to 'fancy' stuff
  - __ldg
  - __shared__
  - FP16 Tensor Cores
  - JIT compilation
  - numa-awareness?

- OMP (and pragmas) are not well integrated in the language, e.g. how can we put a pragma on a loop that is not explicit (C++ TMP)

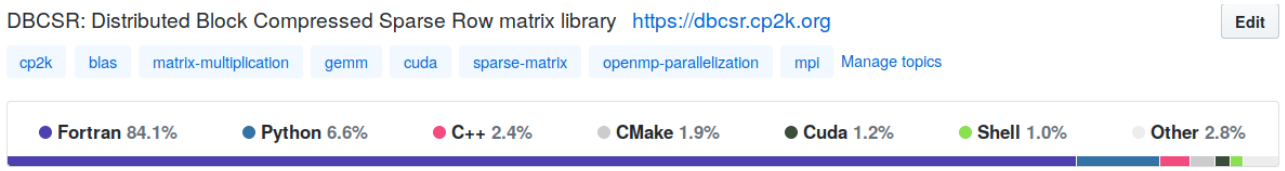
For both performance & maintainability OMP is not optimal.
(but it benefits from being easy for the first 80% of the road to HPC)

Your CUDA code is not portable…… but maybe just a small part of the whole

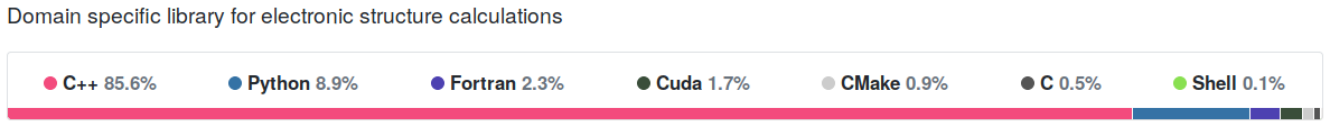
GridTools as a DSL, enables a production-quality weather model to run on GPUs:



DBCSR: CUDA enabled sparse matrix multiplication backend for CP2K and others



SIRIUS: CUDA enabled backend for PW DFT (QE, CP2K, Exciting, ...)



- less than 2.0% CUDA code, typically less than Cmake :-)
- Moving SIRIUS to ROCm took a new hire (Msc level) ~2 months

GPU enabling is often introducing good software engineering practices

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Weather and Climate: GridTools

NWP with COSMO: Initial Objective and Result

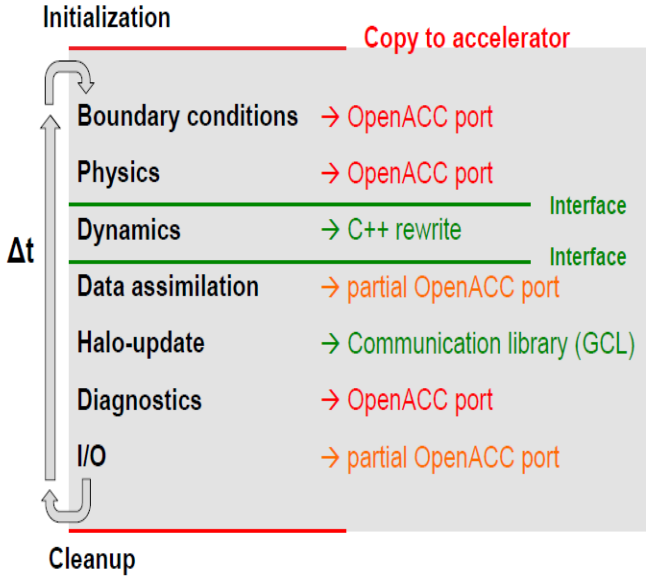
MeteoSwiss was the first national weather service to have GPU based operational forecast. This was made possible by Stella/GridTools as performance portable library.

- Weather forecasts in CH: 40x improvement in 3 years
- Same budget

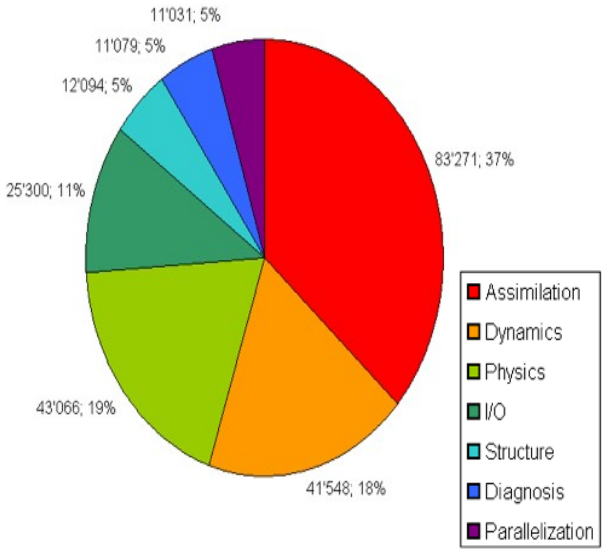
- Rewrite code (Fortran -> C++): 1.7x
  - Change in software architecture
- Mathematical improvements (e.g., single prec): 2.8x
- Change in architecture (GPUs): 2.3x
- Moore's Law: 2.8x
- More "processors": 1.3x

COSMO overview

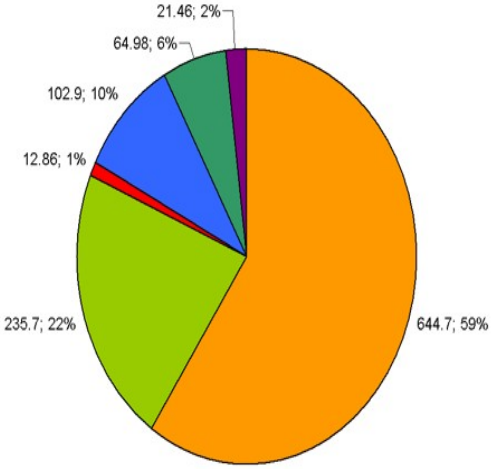
- Nonhydrostatic regional weather and climate model on a lat-lon grid
- Monolithic Fortran90 code
- About 250’000 lines of code



Lines of code (F90)



Runtime



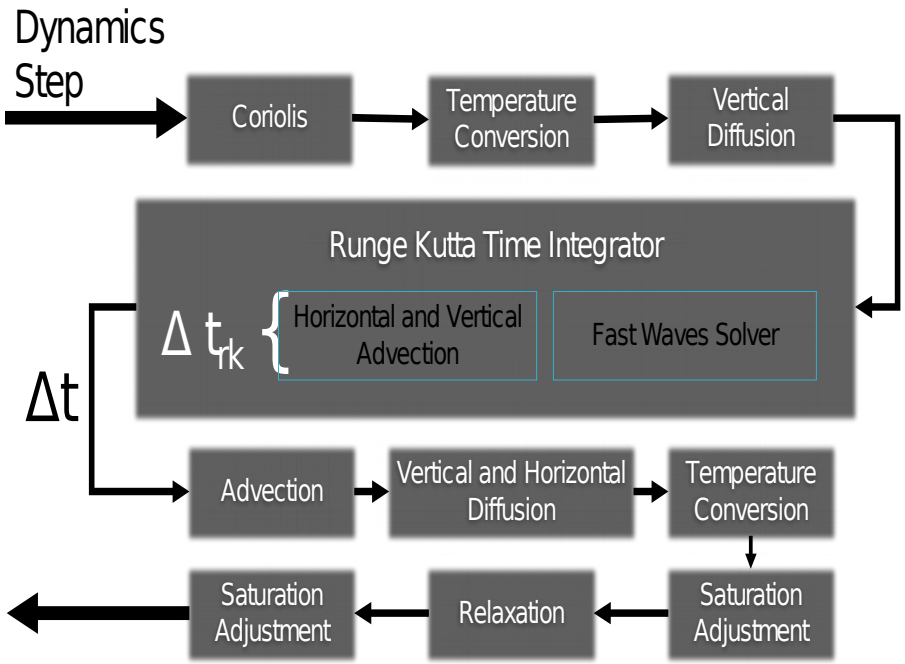
based on 2 km operational model of MeteoSwiss until 2016

Porting goals

- Single source code
- Performance portable: optimal on all supported architectures
- **Separation of concerns**

The dynamical core

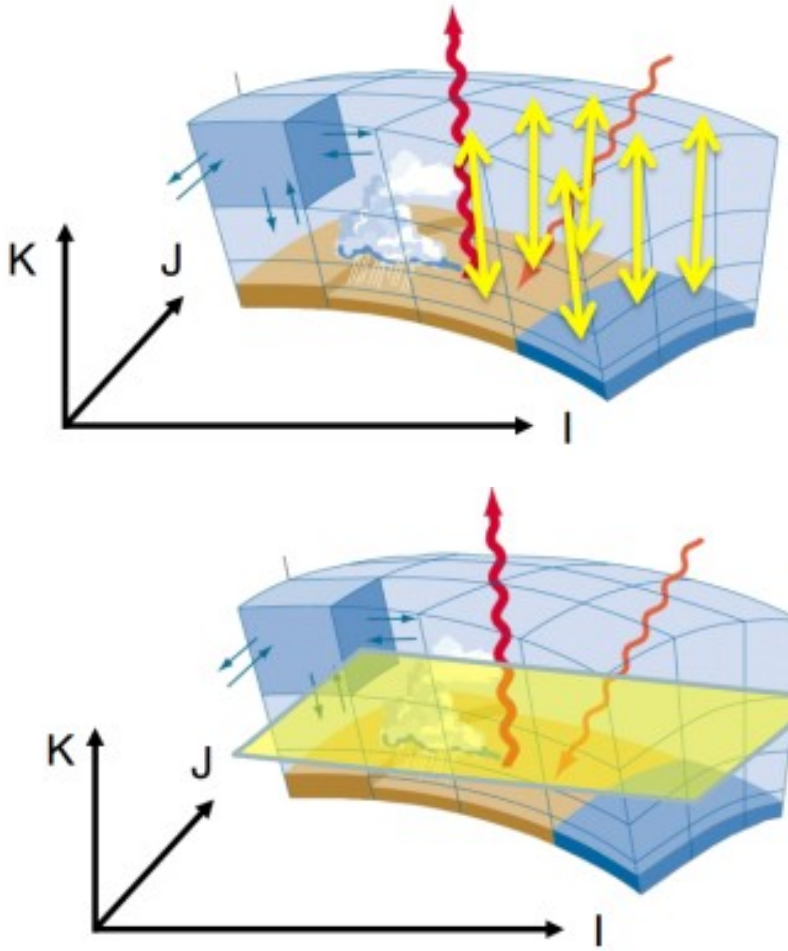
- 39 stencils
- 50-60 fields in each timestep



The challenge … real world complexity

- Portable efficient weather/climate applications
  - Explicit methods
  - Many data-fields
  - Many equations per time step
  - Implicit in the vertical
  - Mostly memory bound stencils
- Many users
  - Many architectures

Few approaches make it into an operational model !

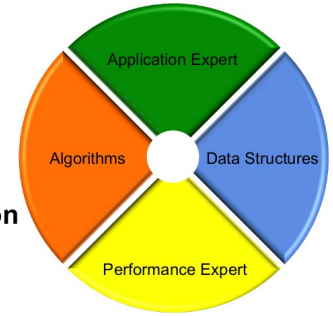


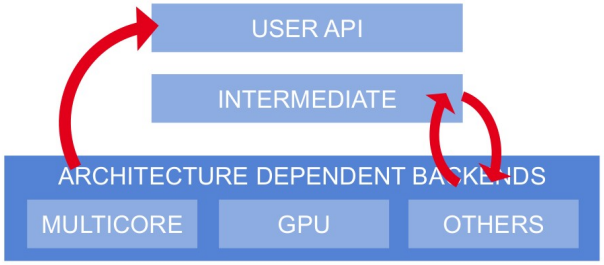
Weather and Climate : the GridTools Framework

Introduce a domain specific language to express stencil computations.

Design Principles

- Separation of concerns between
  - **Algorithms** and **data structures**
  - **User interface** and **efficient implementation**
    - Backend knows the architecture
    - User queries the backend (statically)
    - Backend assumes best conditions
- Change a single line to move to another machine
  - And recompile





Separation between datastructures and algorithms can be obtained with generic programming.

(C++ templates and STL typical example)

Performance by e.g. loop fusion, data layout (tiling, alignment), re-use of optimal code, hardware specific backends.

Example DSL code

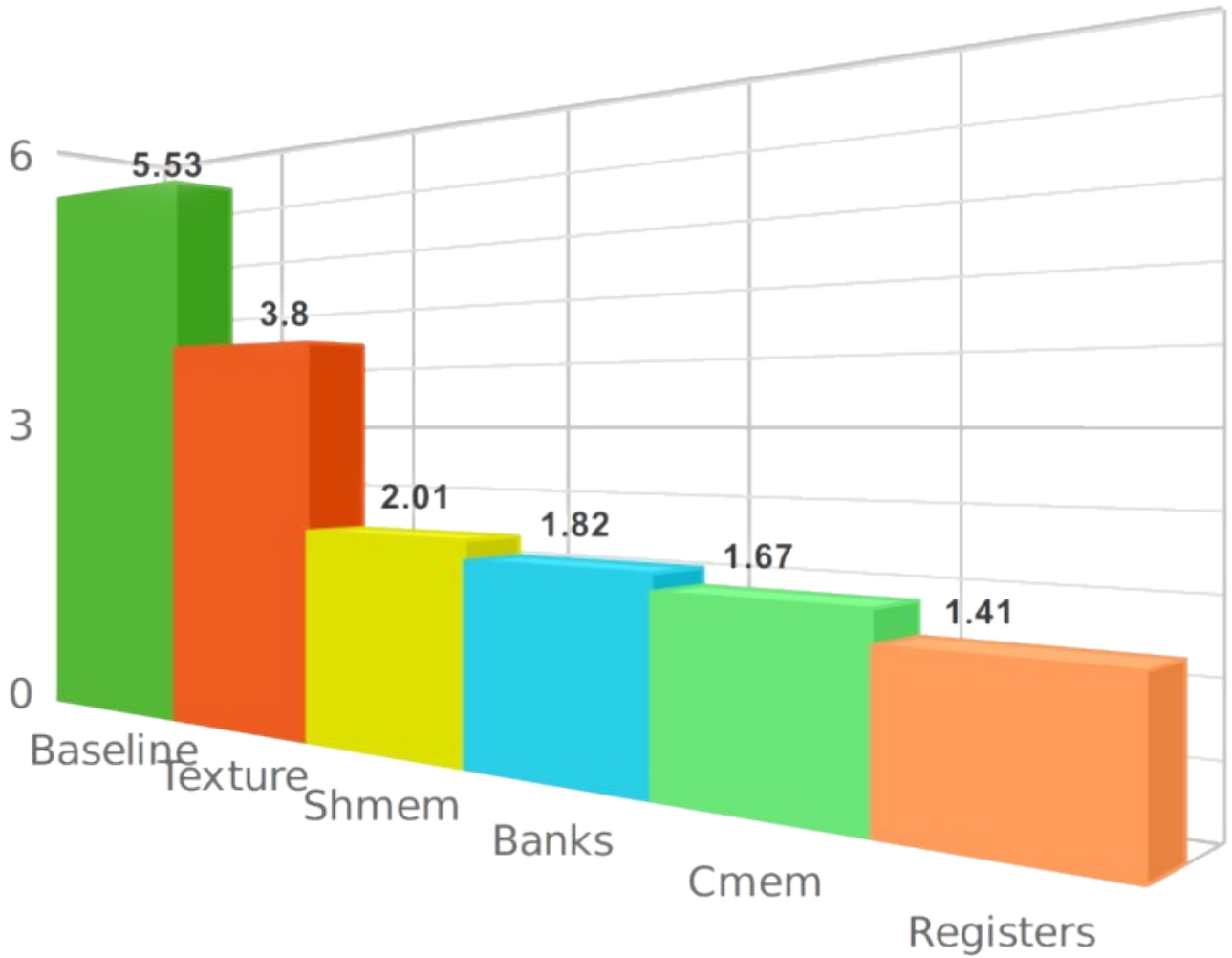
```
struct laplacian {
  using in = in_accessor<0, extent<-1, 1, -1, 1>>;
  using lap = inout_accessor<1>;
  using param_list = make_param_list<in, lap>;

  template <typename Evaluation>
  GT_FUNCTION static void apply(Evaluation const &eval) {
    eval(lap(i, j, k)) = -4. * eval(in(i, j, k))
                         + eval(in(i + 1, j, k))
                         + eval(in(i, j + 1, k))
                         + eval(in(i - 1, j, k))
                         + eval(in(i, j - 1, k));
  }
};
```

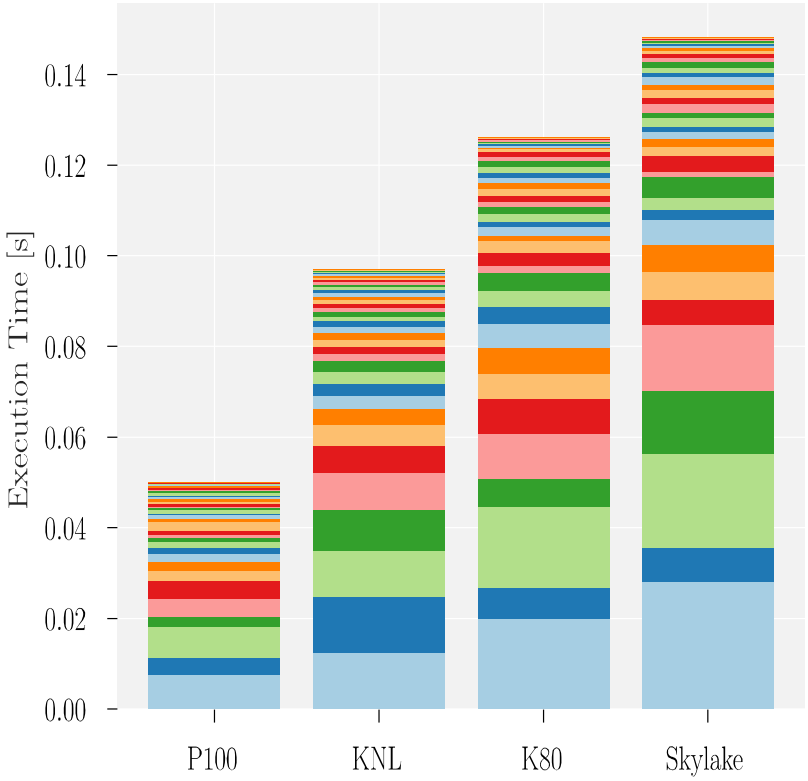
```
using arg_in = arg<0, data_store_t>;
using arg_lap = tmp_arg<1, data_store_t>;
using arg_out = arg<2, data_store_t>;
auto horizontal_diffusion = make_computation<backend_t>(
    my_grid,
    make_multistage(
        execute::parallel(),
        make_stage<laplacian>(arg_in(), arg_lap()),
        make_stage<laplacian>(arg_lap(), arg_out())
        ));
horizontal_diffusion.run(arg_in{} = in, arg_out{} = out);
```

What, not how!

Successive Backend Optimizations



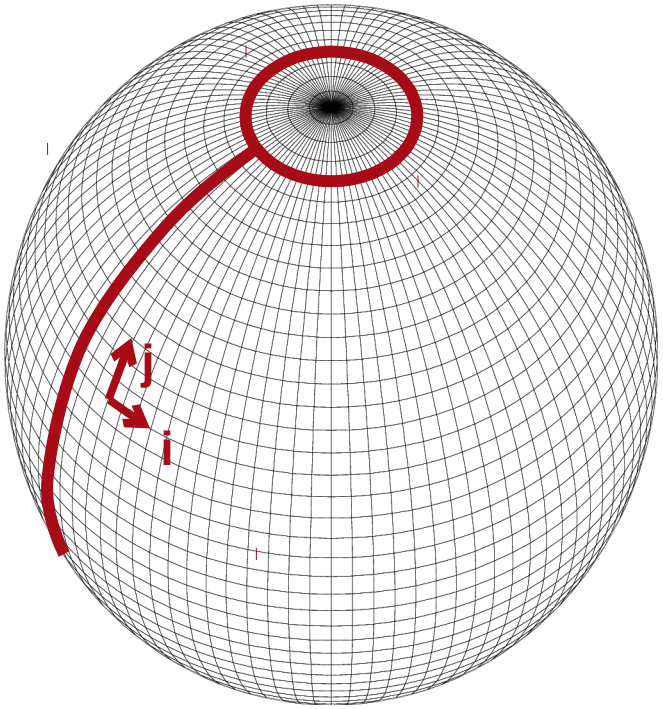
KNL Experiments: Dycore on KNL

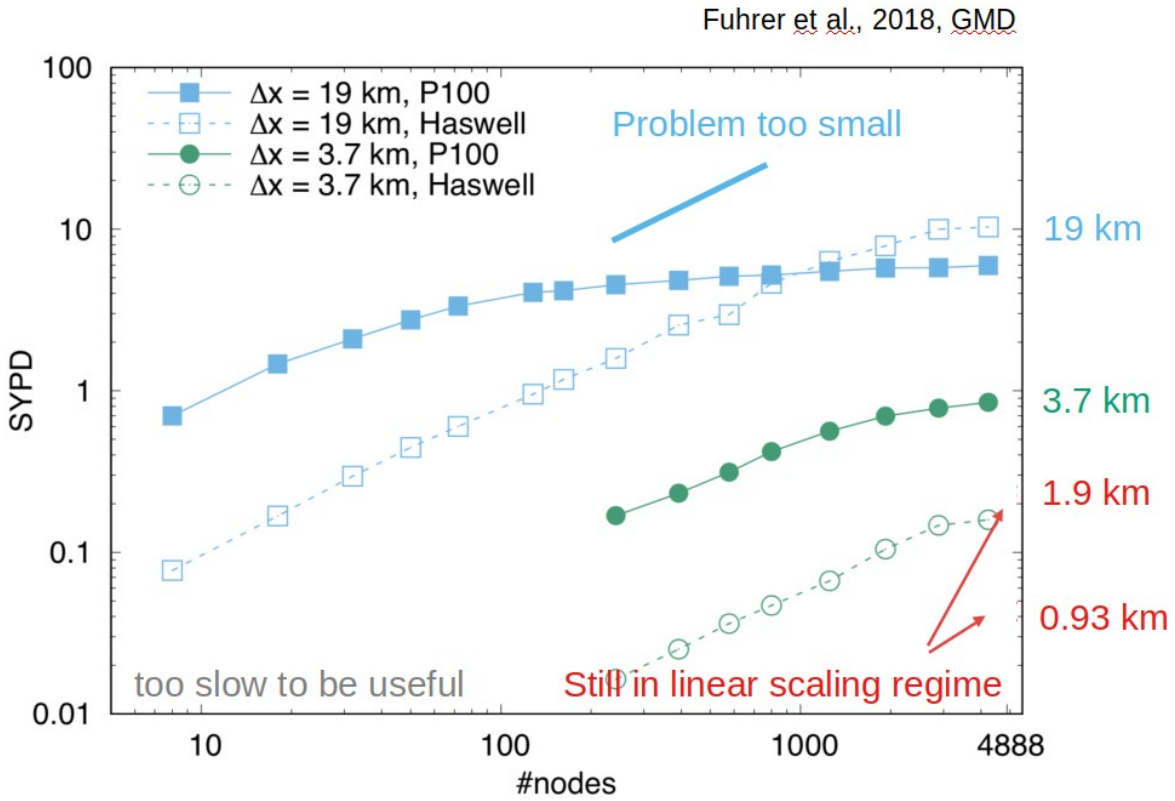


- Same user code for KNL and GPU
- KNL 2 times slower than P100
- Limitations of Performance Portability
  - Compiler issues, softw. prefetching
  - Fusion style
- Quite nice:
  - On par with Fortran implementation
  - 10× faster than the old CPU backend on KNL

CSCS Exascale target :

Global climate simulation at 1km resolution & 1 SYPD





Ambitious scientific goal, needs 60 – 100x speedup. Will require further progress in software and hardware.

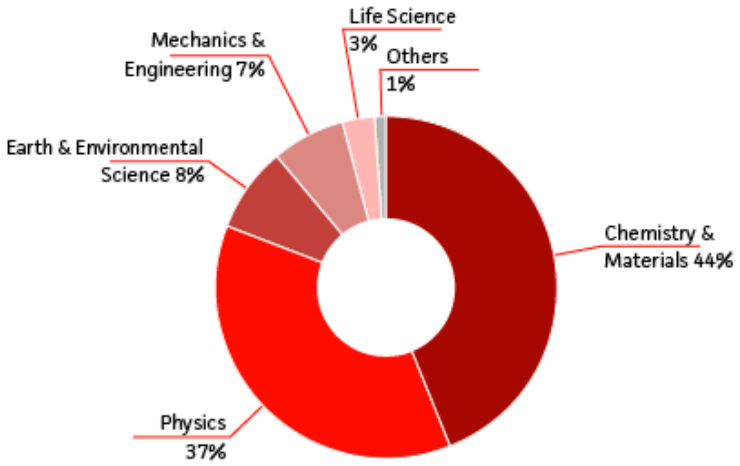
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Computational Materials / Chemistry: CP2K

Quantum chemistry & CP2K

Swiss Supercomputing (CSCS)



Solving Quantum Chemistry equations (=Chemistry and Materials) amounts to roughly 44% percent of supercomputer usage in Switzerland.

CP2K is a code for such calculations, Production ready and widely used:

www.cp2k.org

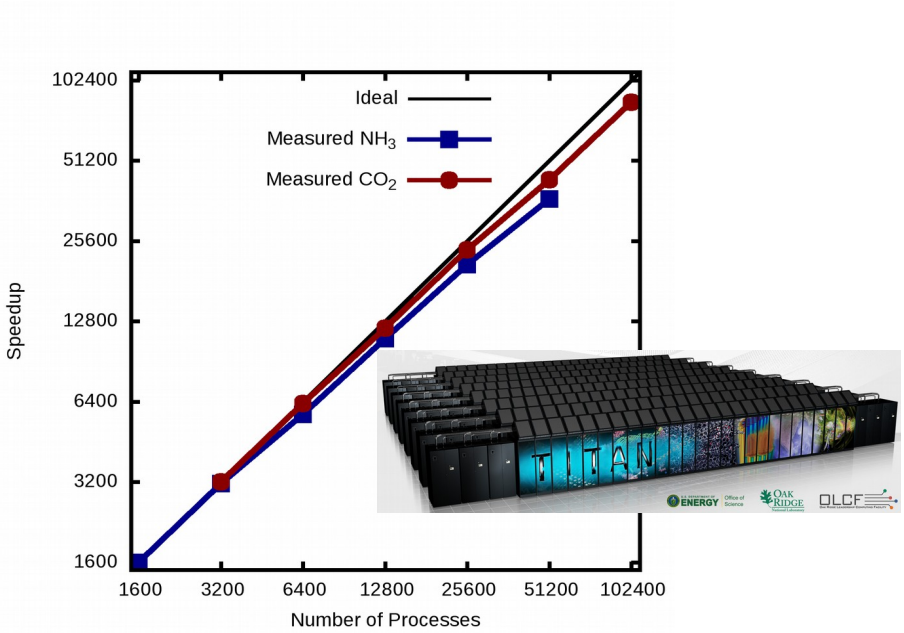
CP2K: algorithms & implementation

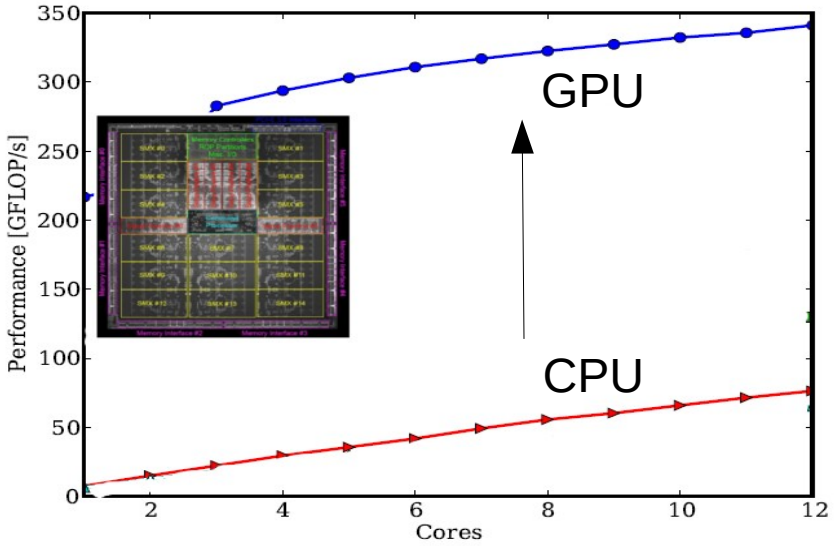
Research & co-design: Hardware vendors & scientists look together for the best solution (both soft- and hardware)

How can we program

for 10'000 – 1'000'000 cores ?

for 'emerging' architectures ?





Could save 300 MWh/yr for our group.

M. Del Ben, J. Hutter, J. VandeVondele, J. Chem. Theory Comput 8, 4177-4188 (2012)
Schütt O; Messmer P; Hutter J; VandeVondele J ISBN 9781118661789

Linear scaling DFT: 1M quite feasible

Linear Scaling Self-Consistent Field Calculations with Millions of Atoms in the Condensed Phase

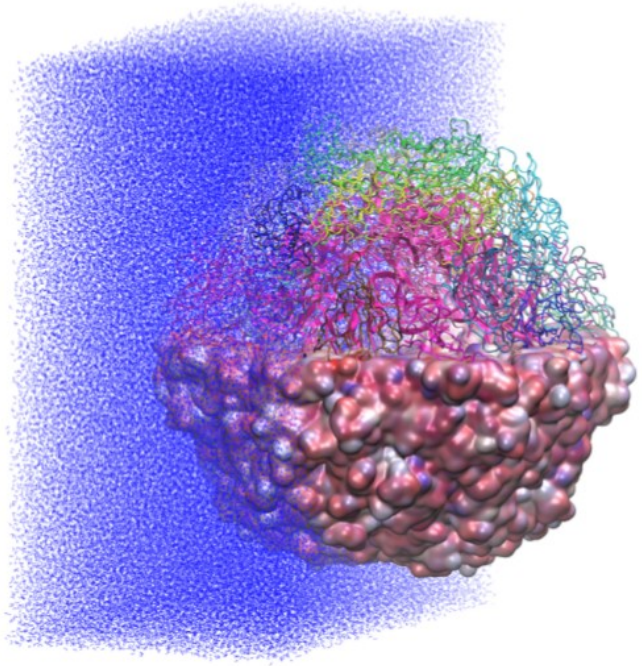
Joost VandeVondele,*,† Urban Borštnik,‡,§ and Jürg Hutter‡

†Department of Materials, ETH Zurich, Wolfgang-Pauli-Strasse 27, 8093 Zurich, Switzerland
‡Physical Chemistry Institute, University of Zurich, Winterthurerstrasse 190, CH-8057 Zurich, Switzerland

Combining Linear-Scaling DFT with Subsystem DFT in Born–Oppenheimer and Ehrenfest Molecular Dynamics Simulations: From Molecules to a Virus in Solution

Samuel Andermatt,† Jinwoong Cha,† Florian Schiffmann,†,‡ and Joost VandeVondele*,†

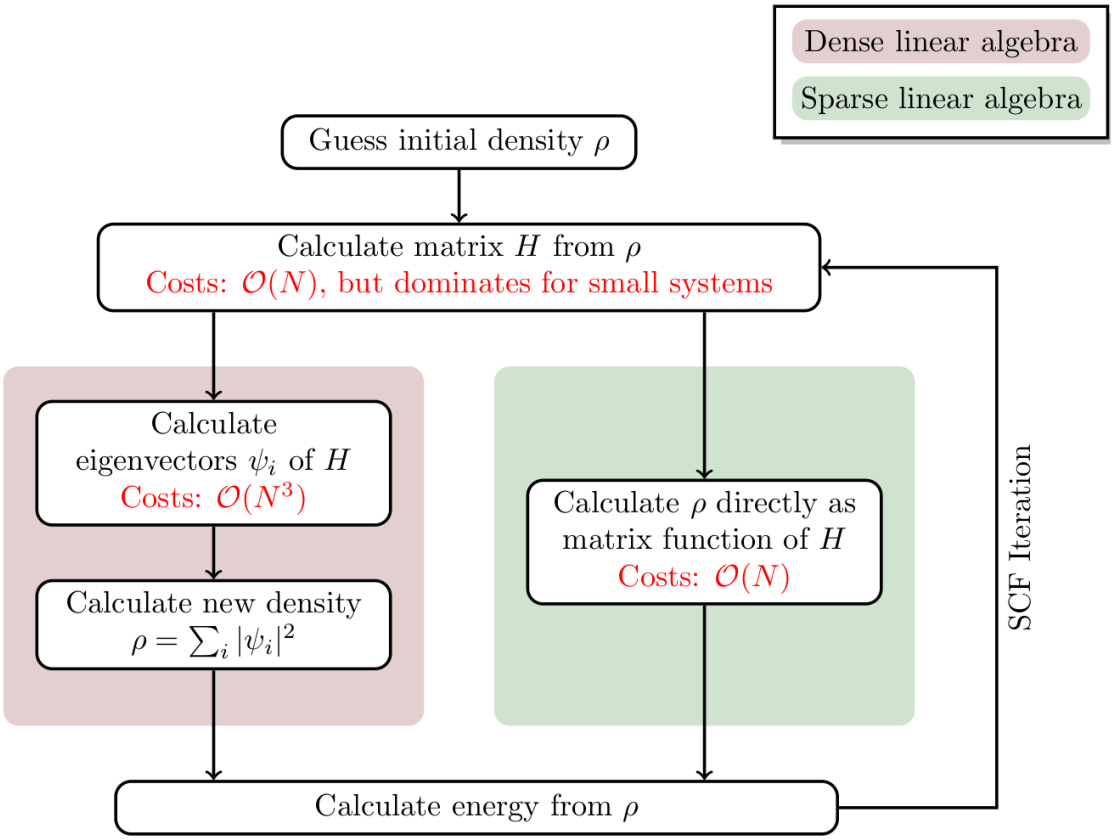
†Department of Materials, ETH Zürich, Zürich, Switzerland
‡Centre of Policy Studies, Victoria University, Melbourne, Australia



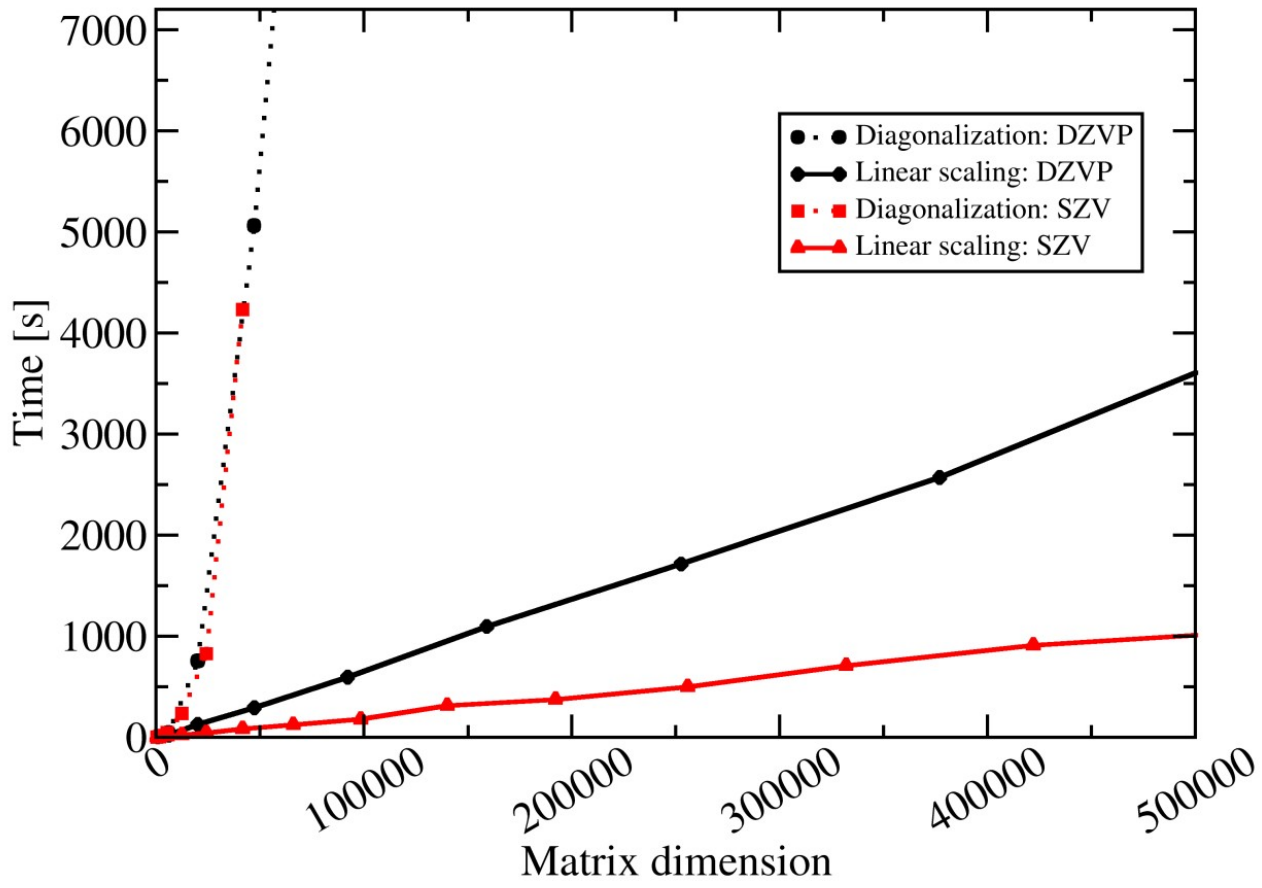
Linear scaling (GGA) DFT.. now well possible for large systems.
Even 1M atoms not too hard any more.
Lots of work getting the basic algorithms (SpMM) to perform.

Figure 11. Visualization of the electrostatic properties of the viral capsid as computed with DFT. The solvent (blue sticks), protein, and RNA (ribbons) are shown in part, while the surface, which corresponds to the viral capsid, is colored according to the computed charges of its constituent atoms. The system contains somewhat more than one million atoms. Visualized using VMD.[81]

DFT a quick refresh



Linear scaling DFT, performance comparison



Algorithms underlying linear scaling DFT

Many algorithms that exploit sparsity in P, H, and S for non-metallic systems.
One example:

$$PS = \frac{1}{2}\left(1 - sign\left(S^{-1}H - \mu I\right)\right)$$

$$X_{n+1} = \frac{1}{2}X_n(3I - X_n^2).$$

$$X_0 = cA \quad \longrightarrow \quad X_\infty = \mathrm{sign}(A)$$

Newton Schulz iteration, requires only matrix multiplications

DBCSR : a sparse matrix library

https://www.cp2k.org/dbcsr
https://github.com/cp2k/dbcsr

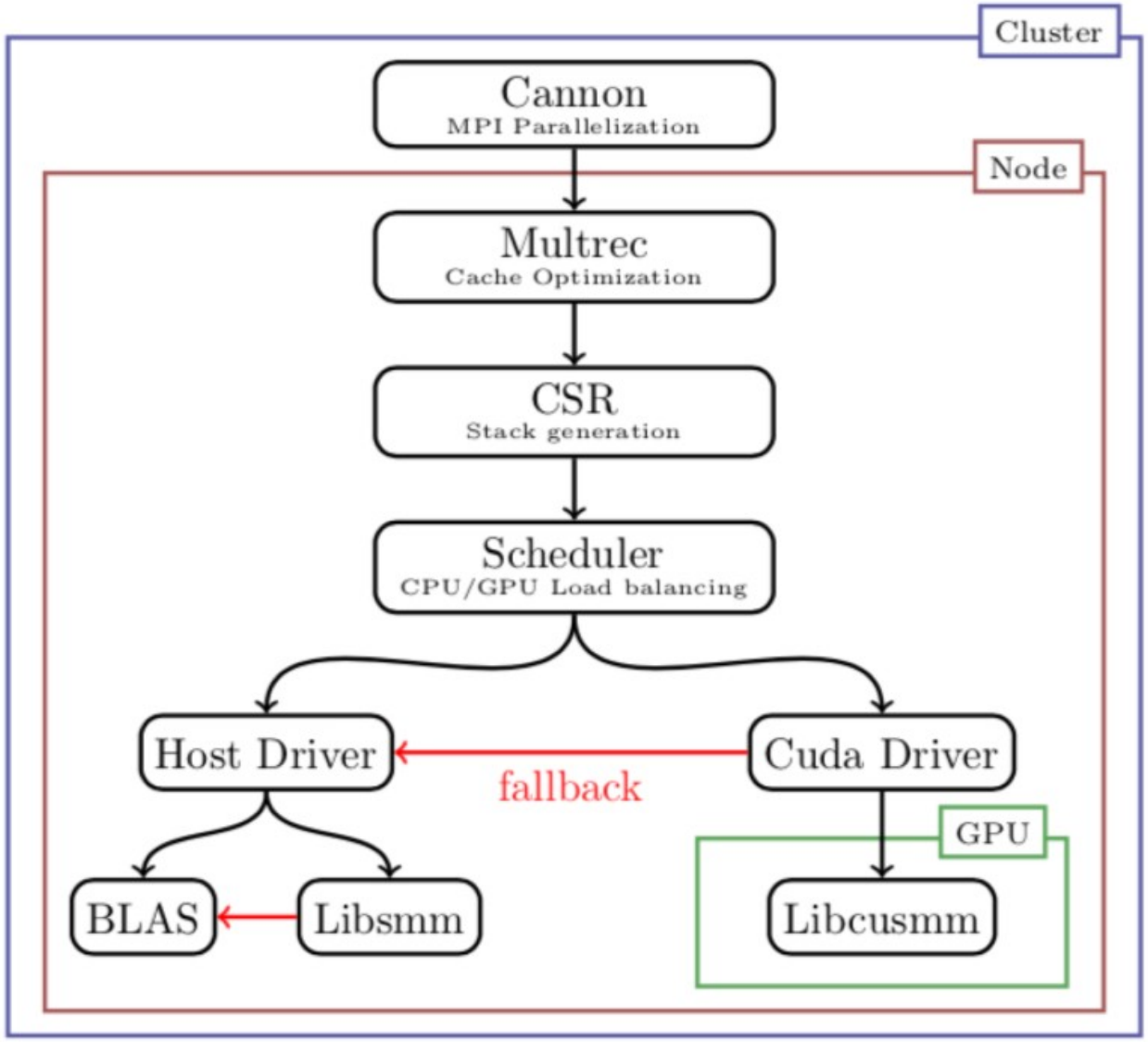
Optimized for the science case: 10000s of non-zeros per row.
The dense limit as important as the sparse limit.

Main operation parallel matrix – matrix multiply.
MPI parallel, OMP parallel, CUDA accelerated

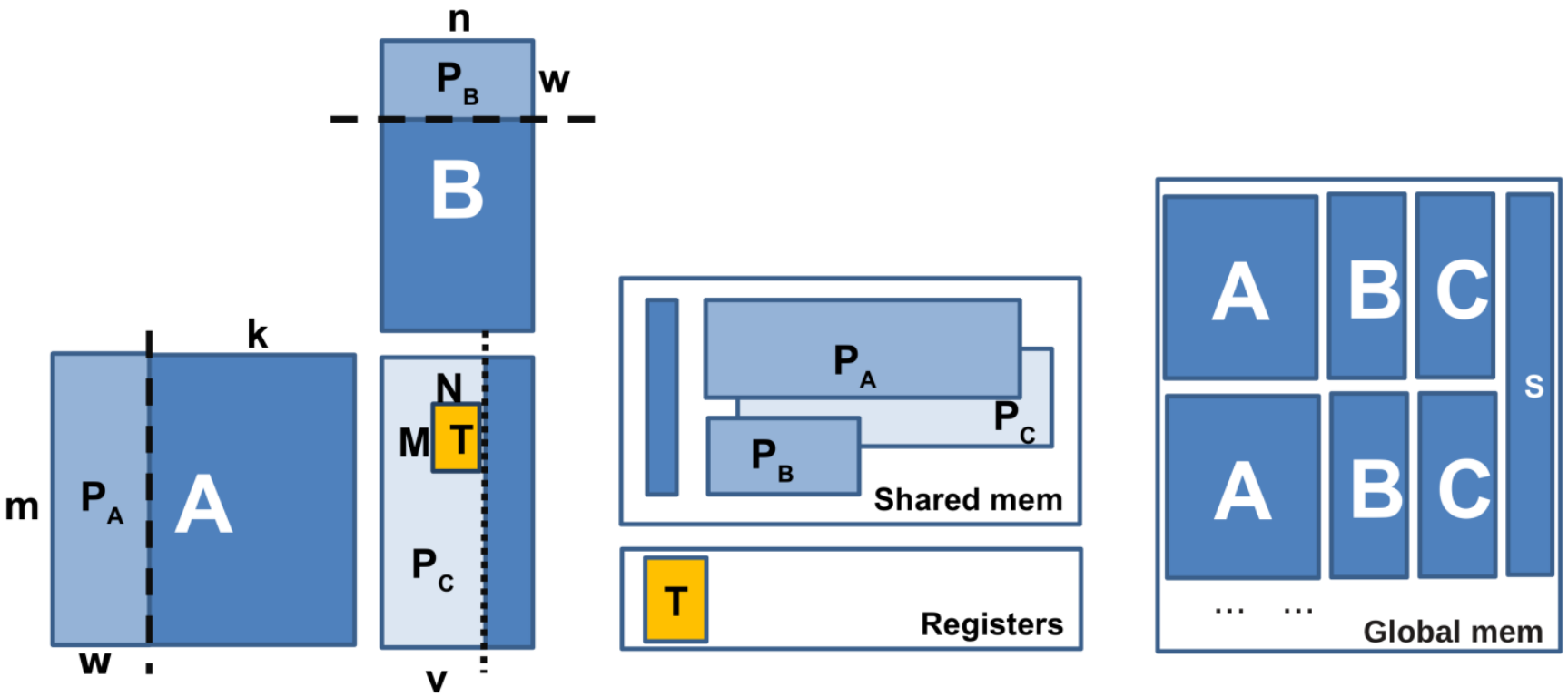
Available as a standalone library, to share
> 20 person-years of development effort

GPU-Accelerated Sparse Matrix–Matrix Multiplication for Linear Scaling Density Functional Theory : https://doi.org/10.1002/9781118670712.ch8

DBCSR code architecture

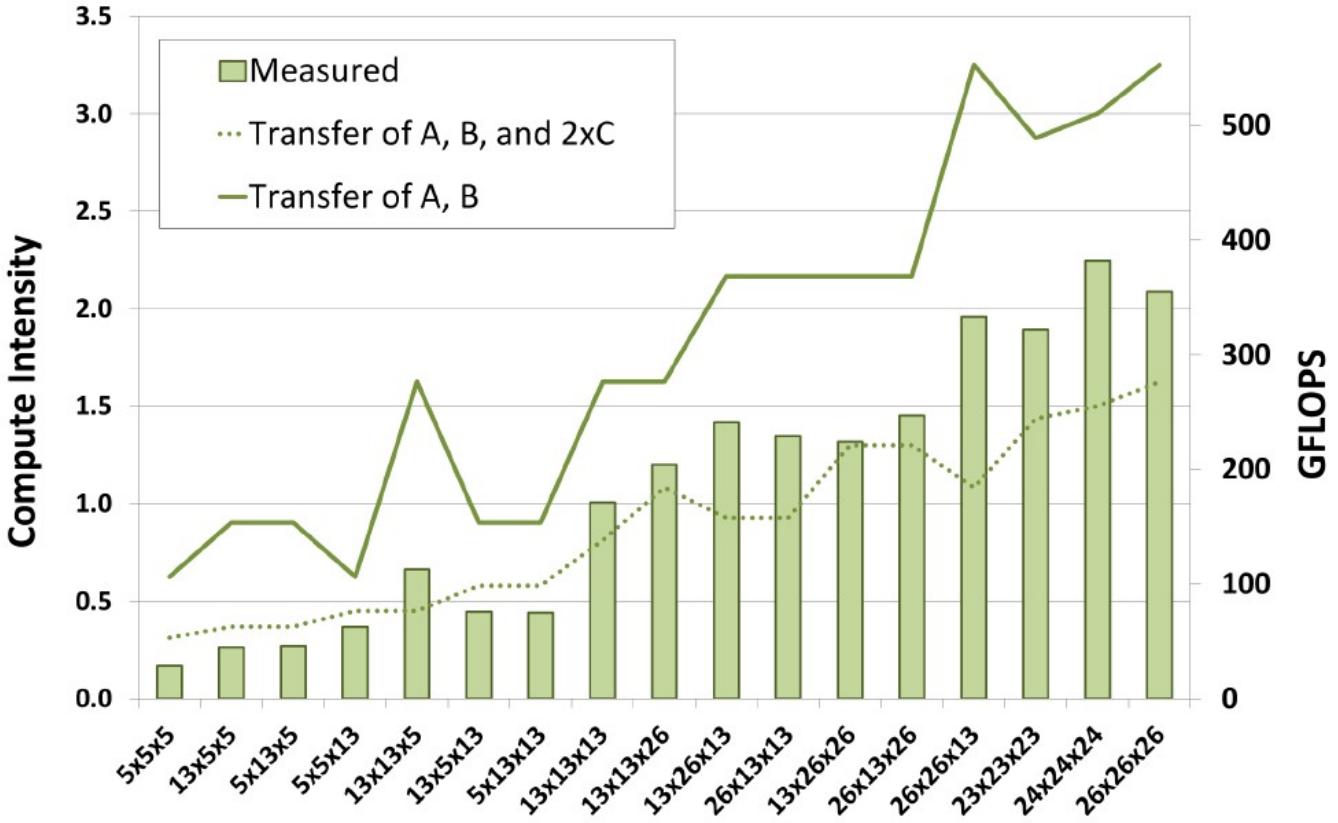


Small matrix multiplication kernels for C += A x B



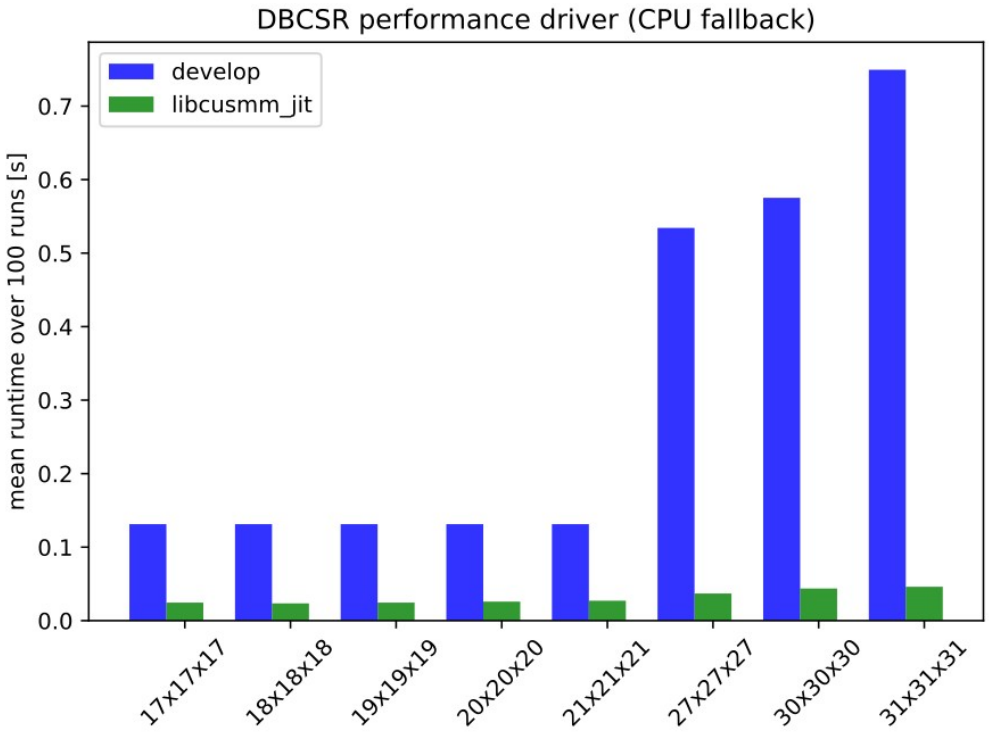
Carefully orchestrate the memory access pattern,
Requires finding multiple parameters for each m,n and k

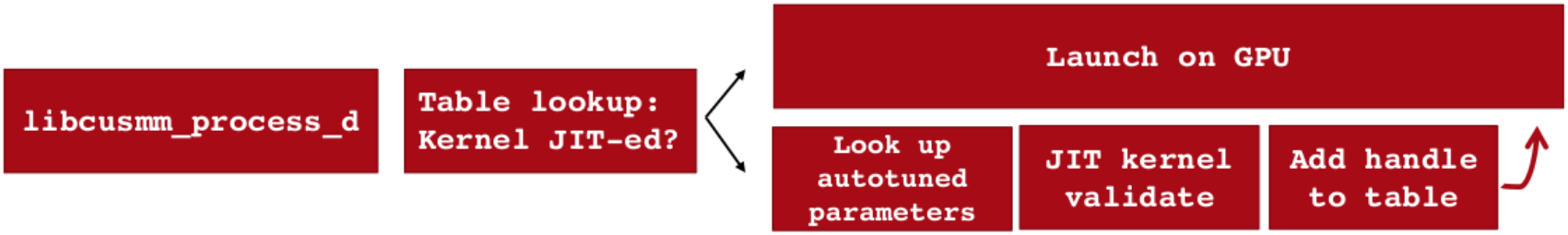
LibCUSMM performance



Measured performance against a roofline model based on memory transfer

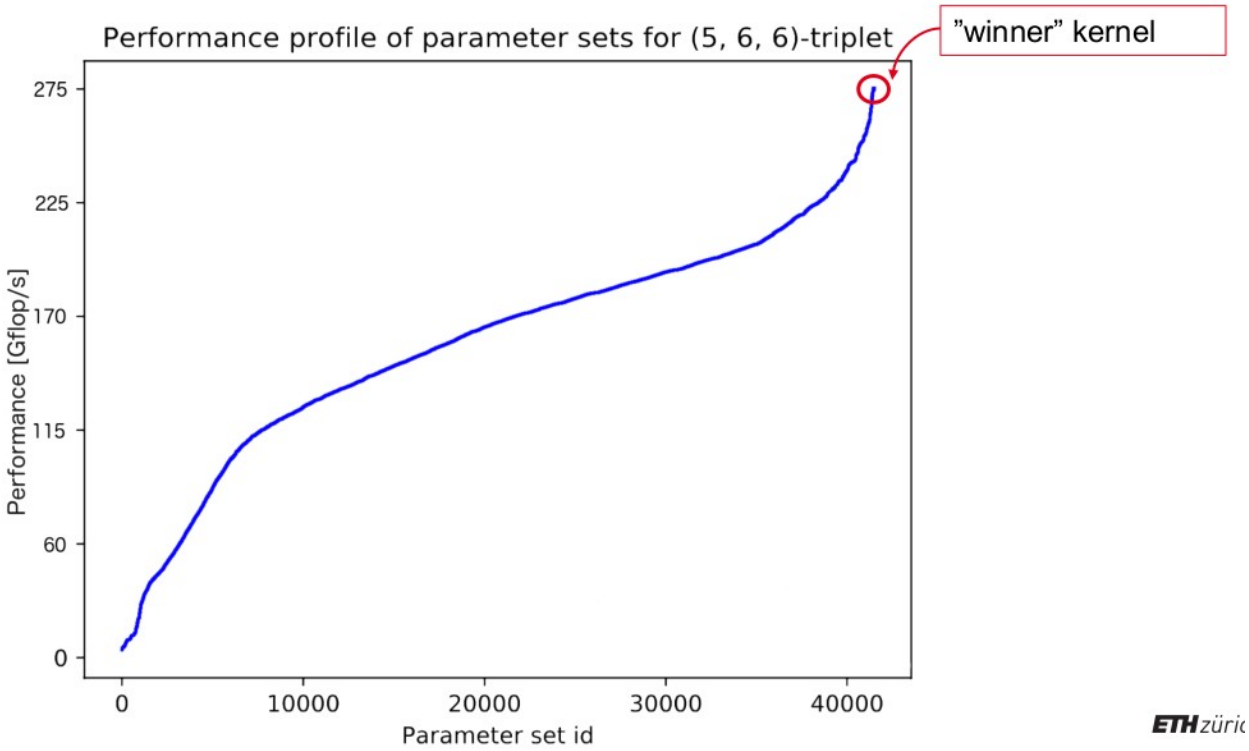
JIT compilation to avoid GPU vs CPU fall-back for kernels





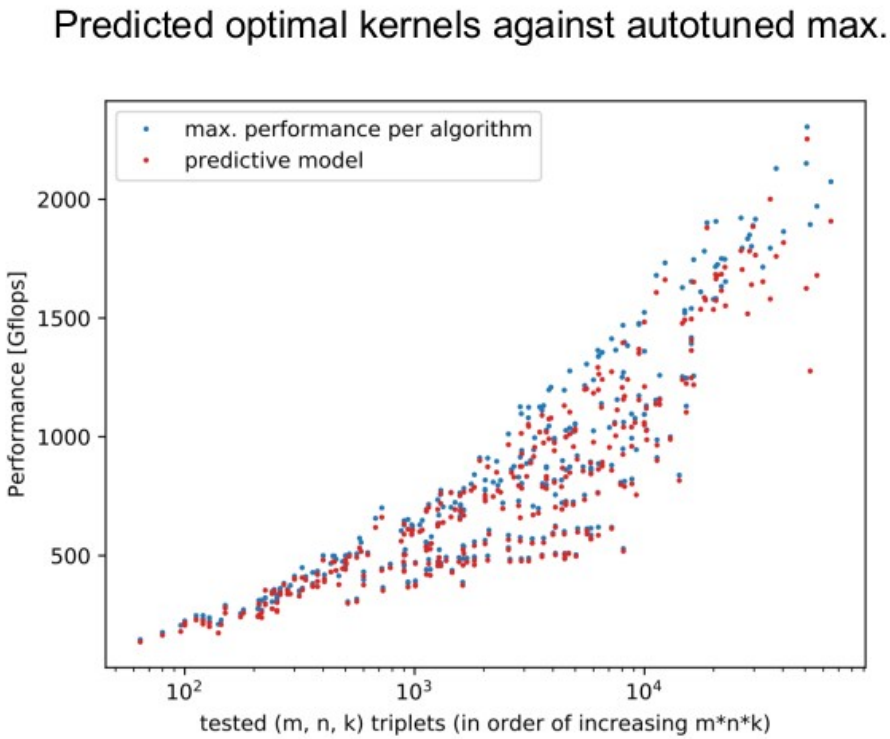
Kernels only… full code Daint node vs node typically 2-3x speedup

Picking optimal kernel parameters



- Many 10000s of variants for each (m,n,k)
- Only few 'winning' kernels
- autotuning possible, but only for a subset of (m, n, k)

Predicting optimal parameters using machine learning



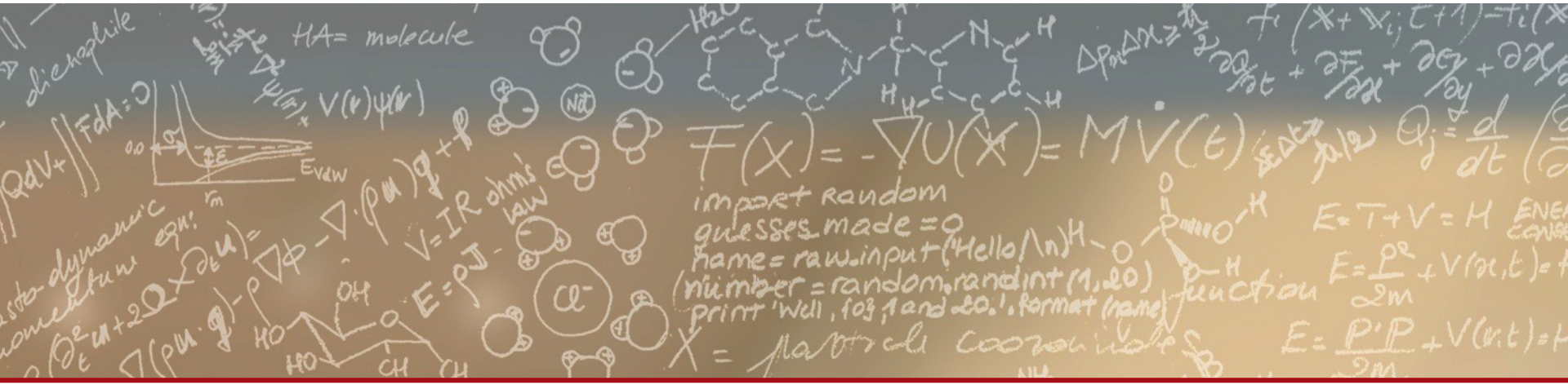
- Regression decisions trees
- Near perfect results (<5% error typical)
- Outperforms ~25% expert baseline results

Conclusions

Conclusions & Acknowledgements

- HPC remains in quick evolution
- GPU acceleration will dominate in Europe and US for the next 5-7 years
- Invest in software: good engineering & separation of concerns needed
- Successful demonstration in
  - Weather and Climate : GridTools
  - Sparse Matrix Multiplication : DBCSR

Thanks to the GridTools and DBCSR teams, at
CSCS, MeteoSwiss, University of Zurich, ETH Zurich !



Thank you for your attention.