High-Performance Computing at Exascale: challenges and benefits

Mathieu Lobet, Matthieu Haefele, Vineet Soni, Patrick Tamain, Julien Derouillat, Arnaud Beck

High-Performance Computing at Exascale: challenges and benefits

Mathieu Lobet, Matthieu Haefele, Vineet Soni, Patrick Tamain, Julien Derouillat, Arnaud Beck

Maison de la Simulation, CEA, CNRS, Université Paris-Sud, UVSQ, Universite Paris-Saclay, F-91191 Gif-sur-Yvette, France
(mathieu.lobet@cea.fr)

SFP Division Plasma

June 14th 2018

Lab and speaker presentation

Maison de la Simulation (Saclay)

Missions:
- Research in HPC, Applied Math, visualization
- Support to the scientific community
- Training (national, Prace)

Involved in national and European projects: EoCoE, Prace

Mathieu Lobet:
Research Engineer in HPC at CEA Saclay

Projects:
- Energy oriented Centre of Excellence for computing application (EoCoE)

Applications:
- SMILEI – High performance Particle-In-Cell code for plasma simulation
- TOKAM3X - Edge turbulence fluid simulations of tokamak plasmas

http://www.maisondelasimulation.fr/

Content

I. The Exascale challenge

II. Future hardware

III. Programming challenges

Conclusion

Content

I. The Exascale challenge

Exascale challenge

To Reach the exaflop computing power (10^{18} flop/s) level with a maximum electricity consumption of 20-40 Mwatts

Main sources of energy consumption:
- Computing units (~50% of the total power consumption on most recent facilities)
- Interconnect
- Memory

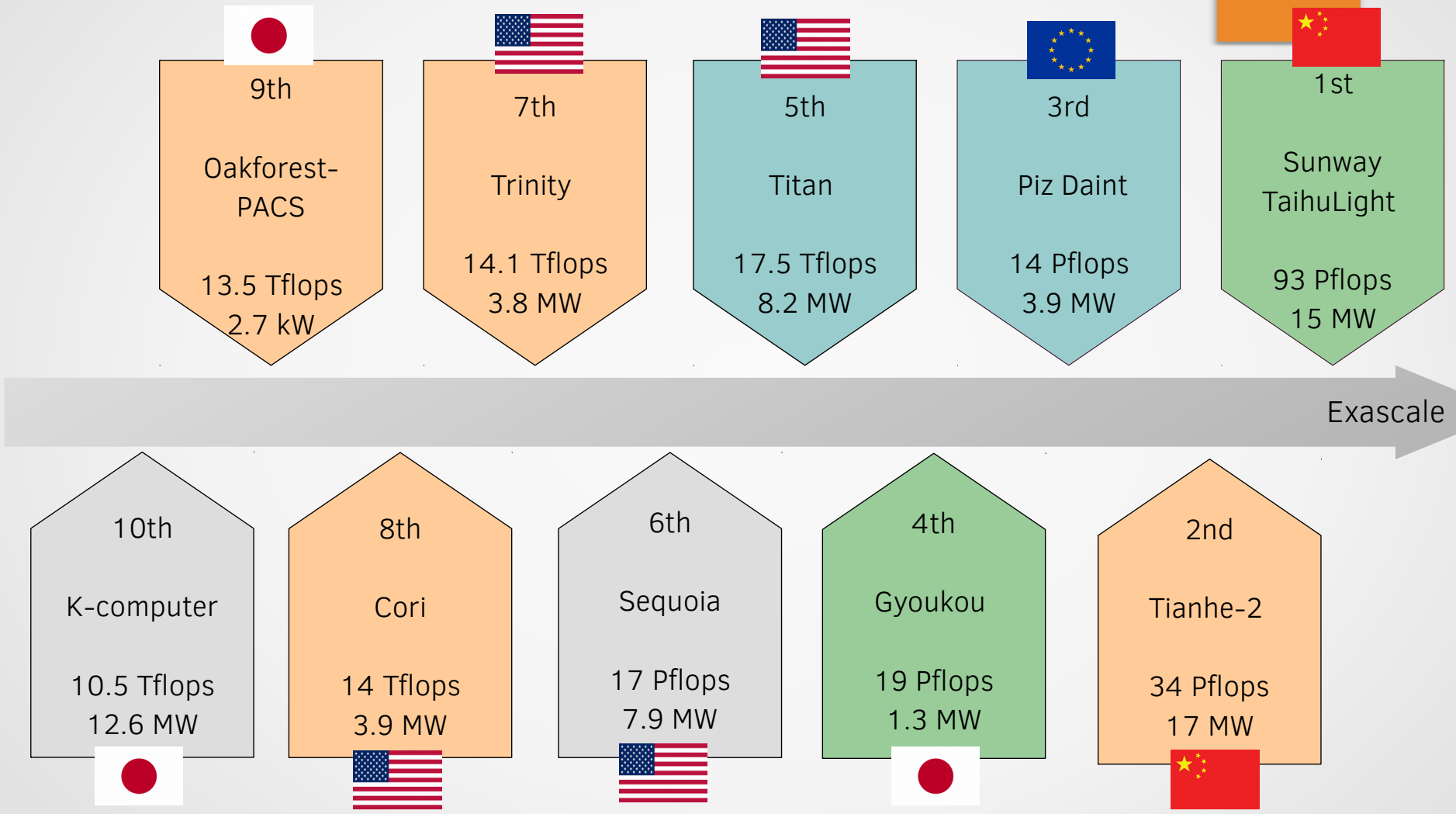
USA – Aurora at Argonne
2021

Europe
2022

China
2019-2020

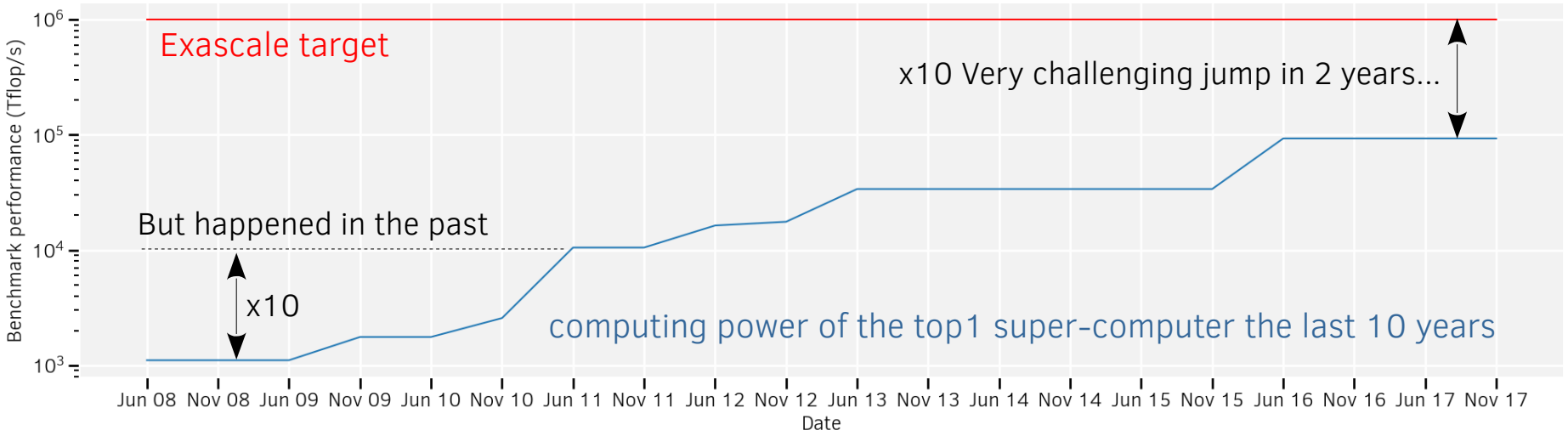
Japan – Post-K computer
2021

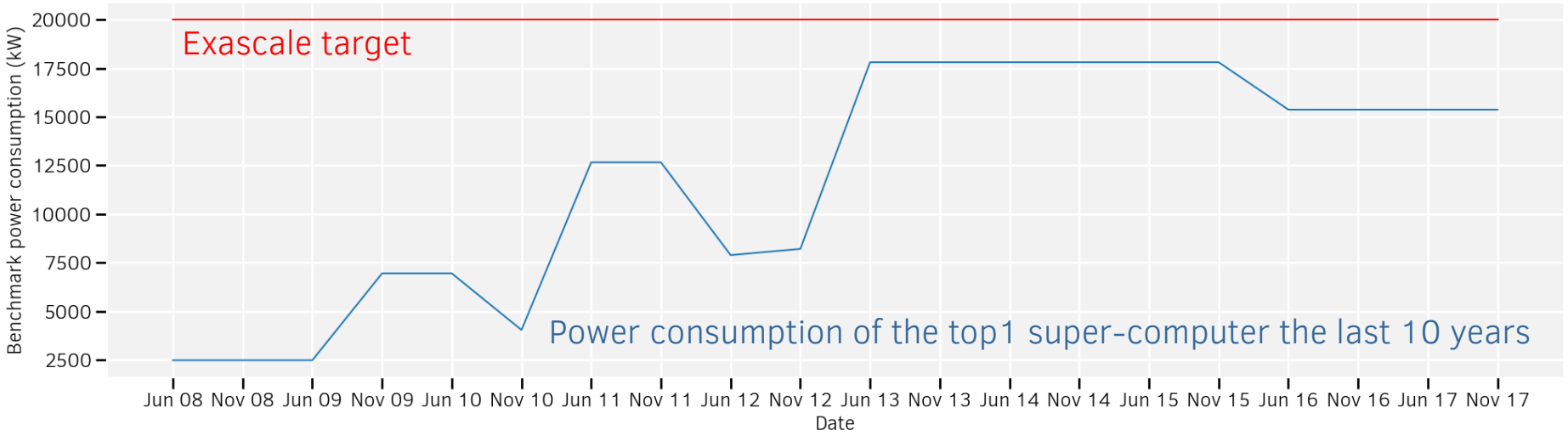
Most powerful supercomputer analysis, Nov 2017



https://www.top500.org/lists/2017/11/

Is this the continuity?





Content

II. Future Hardware

X86 (Intel, AMD) and Power CPU (central processing unit)

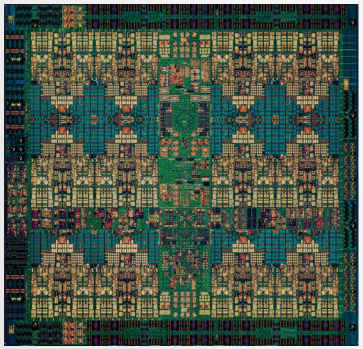
For Intel and AMD:

► Historically, complex processing units originately dedicated to multiple different purposes

► In personnal computers, a lot of silicon dedicated to complex instruction sets to be performant in major cases (operating system, web, office software, video...)

► Disavantage in HPC, results in a bad energy efficiency

► For few years now, specific processors for server and HPC purposes such as Intel Xeon and Intel Xeon Phi that integrates smaller and simpler energy efficient cores with low frequency and larger vector units.

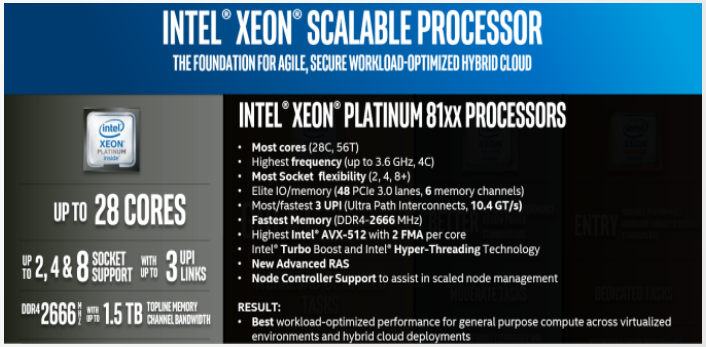


IBM Power9

Up to 32 cores, up to 4 Ghz, SMT4 – SMT8, 190 W

AMD Epyc 7000 series

32 cores, base 2.2 Ghz, 180W



Intel Xeon Skylake

28 cores, base 1.5 Hhz, 200W, AVX512

www.microway.com/knowledge-center-articles/detailed-specifications-of-the-skylake-sp-intel-xeon-processor-scalable-family-cpus/
www.anandtech.com/show/11544/intel-skylake-ep-vs-amd-epyc-7000-cpu-battle-of-the-decade/7
https://www.nextplatform.com/2017/12/05/power9-to-the-people/

X86 (Intel, AMD) and Power CPU (central processing unit)

Pros:

▲ Easy to program

▲ Easy to debug

▲ Cross-plateform

Cons:

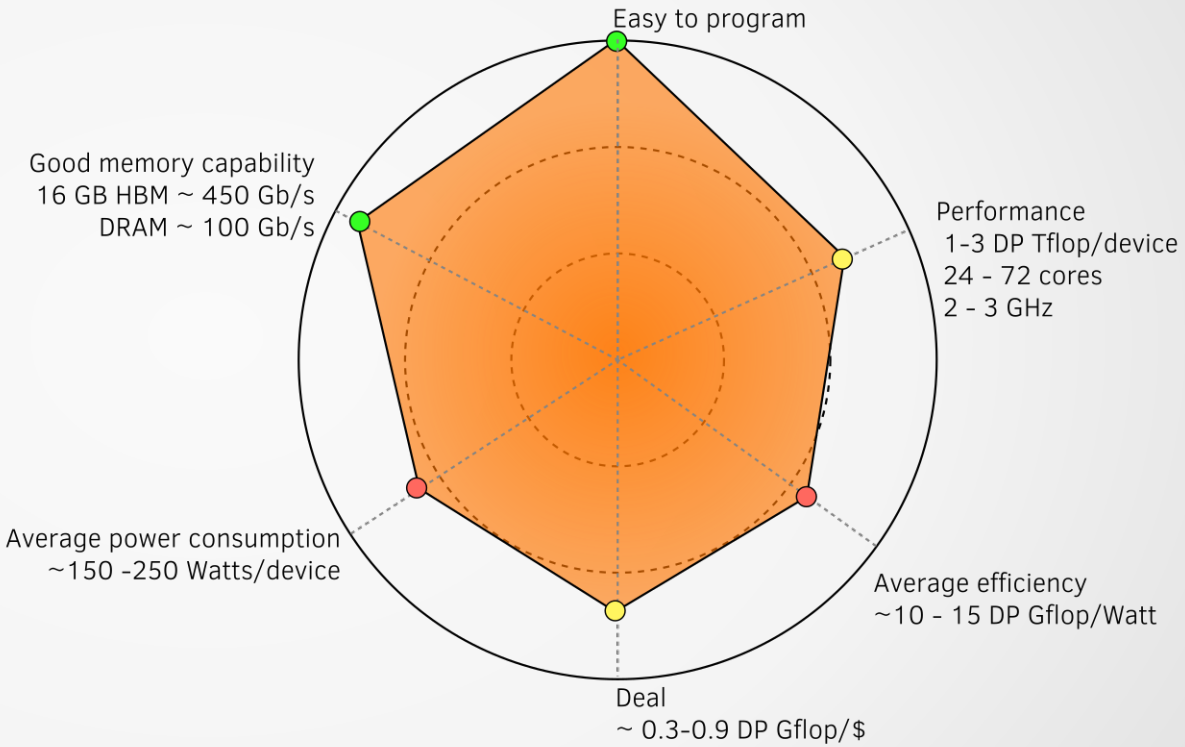
▼ For the moment, not suitable for Exascale standalone due to high energy consumption

Configuration for HPC:

- Host
- Accelerators for Xeon Phi

Programming models available:

- C, C++, Fortran
- Vendor compilers
- LLVM, GNU
- MPI, OpenMP, OpenCL



Based on Xeon Skylake and Xeon Phi KNL characteristics

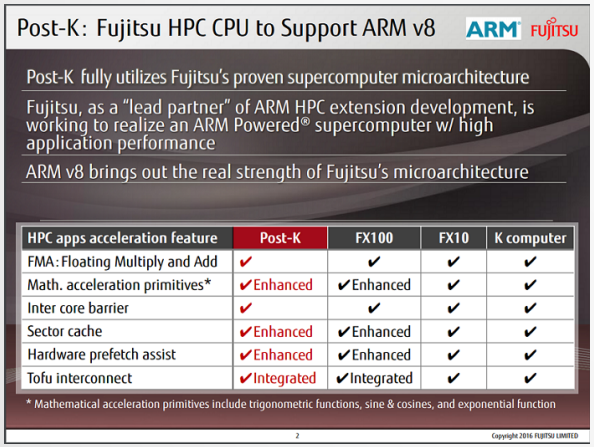
www.microway.com/knowledge-center-articles/detailed-specifications-of-the-skylake-sp-intel-xeon-processor-scalable-family-cpus/
www.anandtech.com/show/11544/intel-skylake-ep-vs-amd-epyc-7000-cpu-battle-of-the-decade/7
https://www.nextplatform.com/2017/12/05/power9-to-the-people/

RISC-based processors such as ARM

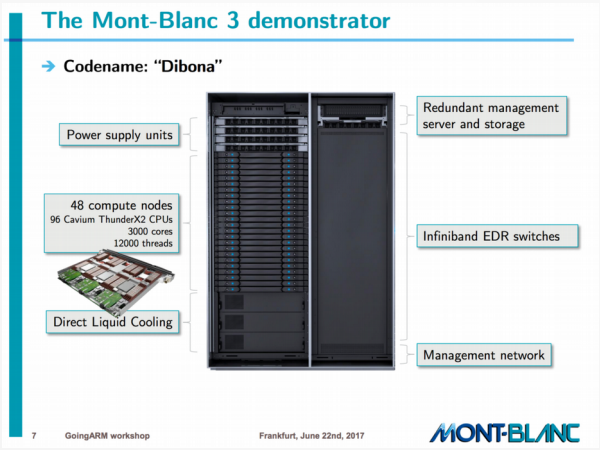
Historically, ARM processor uses the light and less complex RISC architecture (Reduced Instruction Set Computing) that does not contain complex instruction sets enabling low energy consumption per core, less silicon and cheaper chips.

ARM was initially designed for embebded and mobile applications.

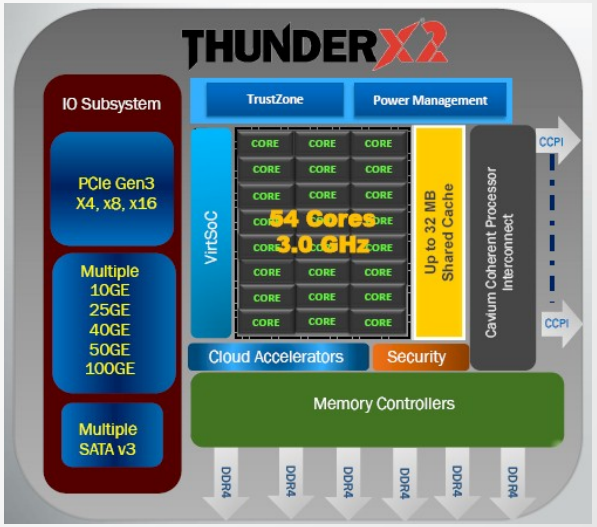
Europe (Mont-Blanc, CEA, Bull) and Japan (RIKEN) are now pushing the development of ARM super-computers.



Post K-computer, a Japanee candidate for Exascale, will be equipped of ARMv8 with SVE ISA. (http://www.fujitsu.com/global/Images/post-k-supercomputer-overview.pdf)



European Bull DIBONA prototype uses Calvium ThunderX2 ARM cards in the framework of the European Mont-Blanc project. (http://montblanc-project.eu/prototypes)



ARM based server Calvium ThunderX2 used in DIBONA. (www.nextplatform.com/2016/06/03/next-generation-thunderx2-arm-targets-skylake-xeons/)

http://laruche.it/2014/07/05/lqt-cpu-arm-x86-differences-549851
https://www.androidauthority.com/arms-secret-recipe-for-power-efficient-processing-409850/
https://www.servethehome.com/cavium-thunderx2-review-benchmarks-real-arm-server-option/

RISC-based processors such as ARM

Pros:

▲ Few code modifications from x86

▲ Server-oriented ARM SoC cheaper than Intel x86 CPUs

▲ Sell as licence: possibility to manufacture in Europe

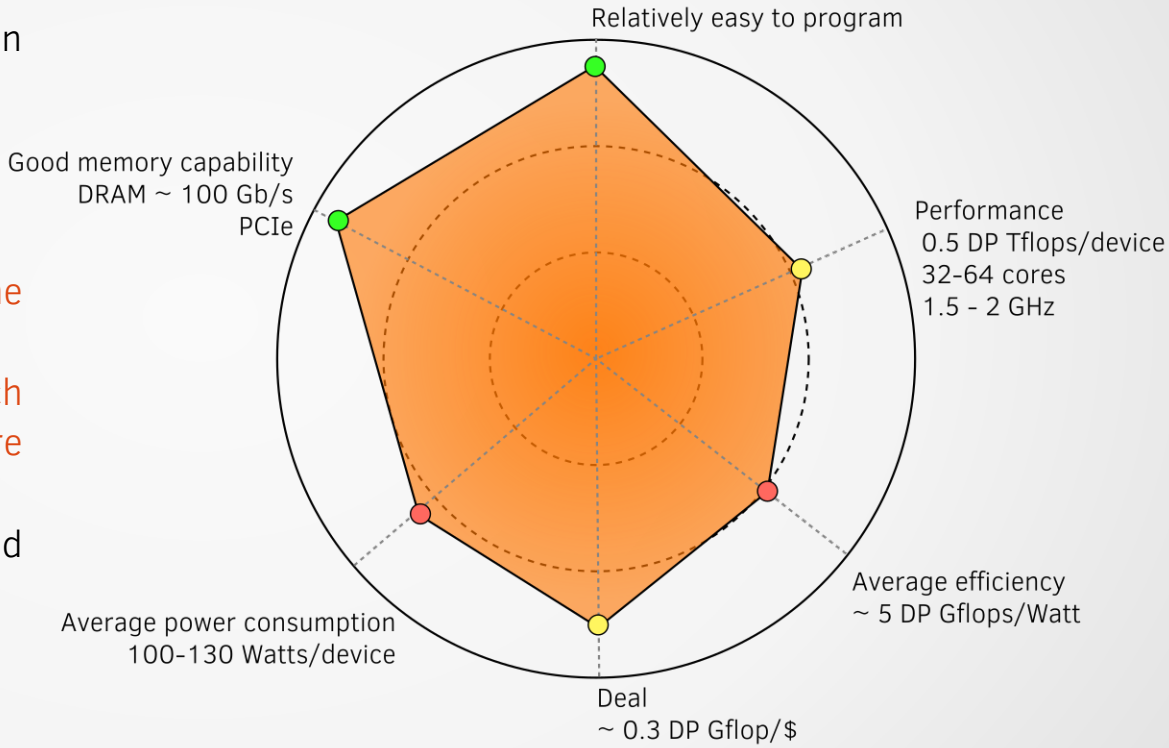
▲ Easy to debug

Cons:

▼ Not really more efficient than x86 for the moment if strong scalable vector usage

▼ Server-oriented ARM TDP tends to reach Intel x86 CPUs due to increased core complexity for HPC

▼ More optimization let to the compilers and the developers to get performance



Based on Cavium, Ampere, Centric 2400 and Phytium Mars characteristics

Configuration for HPC:

- Host
- Replace the x86 CPUs

Programming model available:

- Same as for x86 CPU architecture (C, C++, Fortran, LLVM, GNU, MPI, OpenMP...)

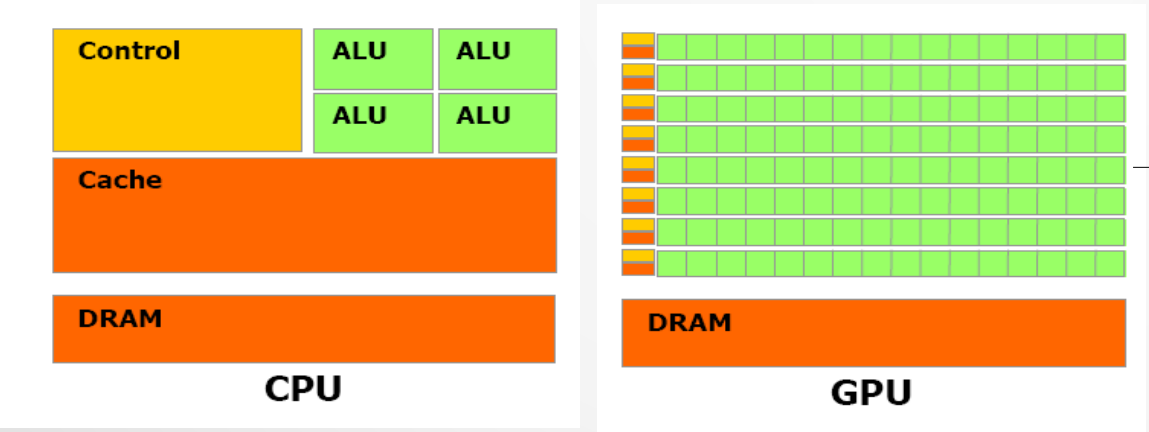
https://www.nextplatform.com/2017/06/30/momentum-building-arm-hpc/ ; https://www.nextplatform.com/2016/09/01/details-emerge-chinas-64-core-arm-chip/

GPGPU (General Purpose Graphic Processing Units)

Historically, GPUs are used for complex graphical rendering in PC.
► need a lot of small simple computing units working in parallel

In the early 2000's, GPUs start to overcome CPU computational power and the gap is still increasing today.
► appeared to be excellent candidate for HPC

Today, GPUs are massively parallel and vector devices composed of a lot of very simple and small cores (> 1000) contrary to CPUs composed of a small number of more complex cores.
► How to parallelize an algorithm is a different way of thinking



Nvidia Tesla P100 Streaming Multiprocessor (SM) architecture

https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf
https://devblogs.nvidia.com/inside-pascal/
http://www.info2.uqam.ca/~makarenkov_v/BIF7002/Rapport_Arhjoul.htm

GPGPU (General Purpose Graphic Processing Units)

Pros:

▲ Good performance in a single device (Desktop HPC application)

▲ Relatively easy to debug

▲ More and more easy to program

Cons:

▼ Important adaptation efforts depending on the algorithm

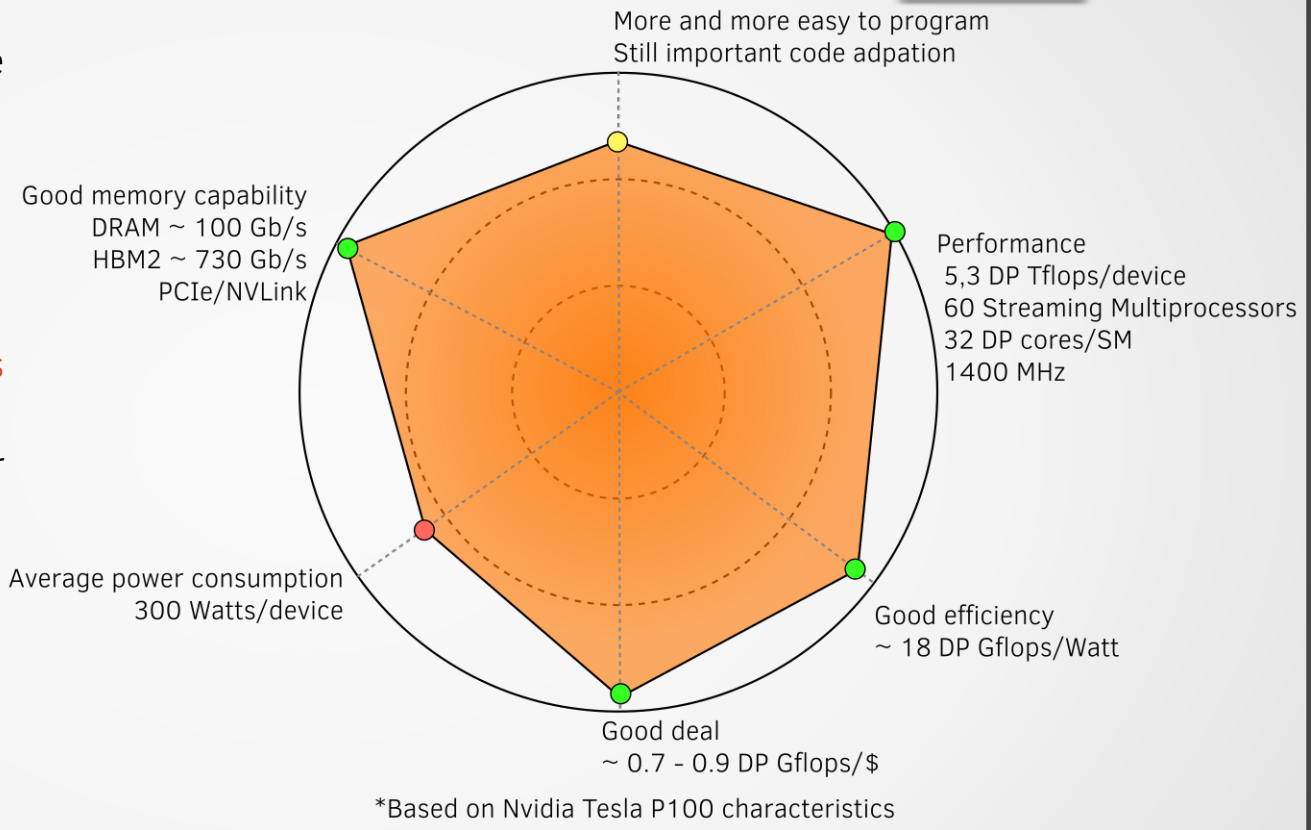
▼ Important optimization efforts for decent performance

Configuration for HPC:

- Accelerator

Programming models available:

- CUDA, OpenCL, OpenACC, OpenMP



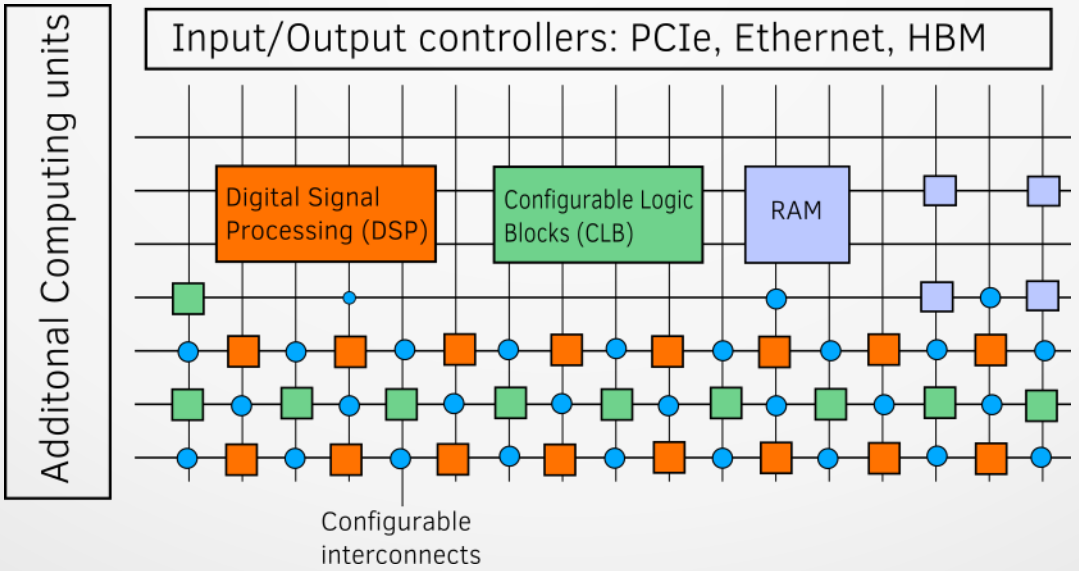
https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf
https://devblogs.nvidia.com/inside-pascal/
http://www.info2.uqam.ca/~makarenkov_v/BIF7002/Rapport_Arhjoul.htm

FPGA - Field Programmable Gate Arrays

FPGA an integrated circuit designed to be configured: an array of programmable logic blocks and reconfigurable interconnects.

The FPGA architecture provides the flexibility to create a massive array of application-specific ALUs that enable both instruction and data-level parallelism.

Very high energy efficiency because of low frequency, a lot of silicon dedicated to the computation, unsued blocs does not consume energy.



https://science.energy.gov/~/media/ascr/ascac/pdf/meetings/201612/Finkel_FPGA_ascac.pdf
https://www.xilinx.com/support/documentation/white_papers/wp375_HPC_Using_FPGAs.pdf
https://www.altera.com/products/fpga/stratix-series/stratix-10/overview.html

FPGA - Field Programmable Gate Arrays

Pros:

▲ Best energy efficiency (flop/Watt) on the market

▲ Versatile and adaptable to many kernels

Cons:

▼ Difficult to program today

▼ Low frequency (~100 MHz) ►pipelining optimization and vectorization

▼ Long compilation (several hours)

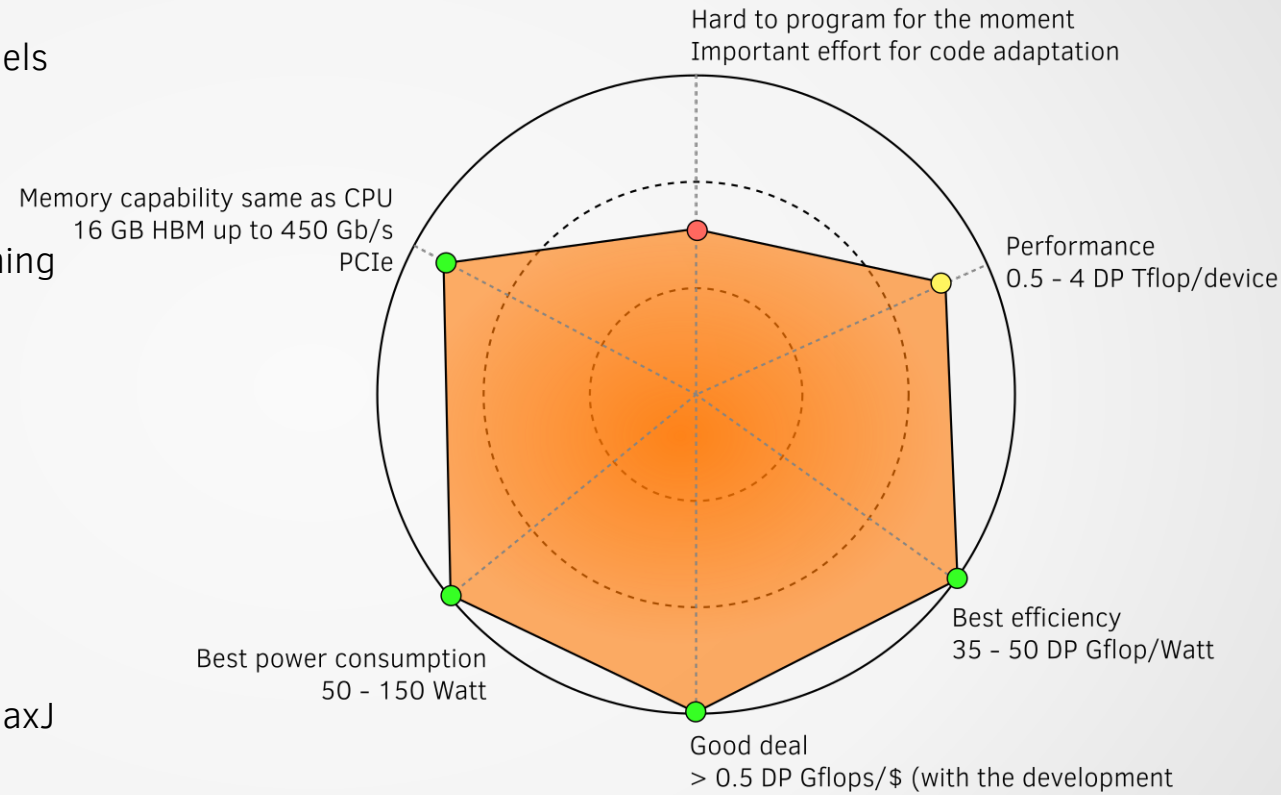
▼ Debugging ? Emulators ?

Configuration for HPC:

- as an accelerator

Programming model available:

- Vendor specific languages (MaxJ language for Maxeler)
- OpenCL with C/C++/Fortran (pushed by Intel)
- OpenMP/OpenACC (not mature)
- VHDL (higher difficulty)

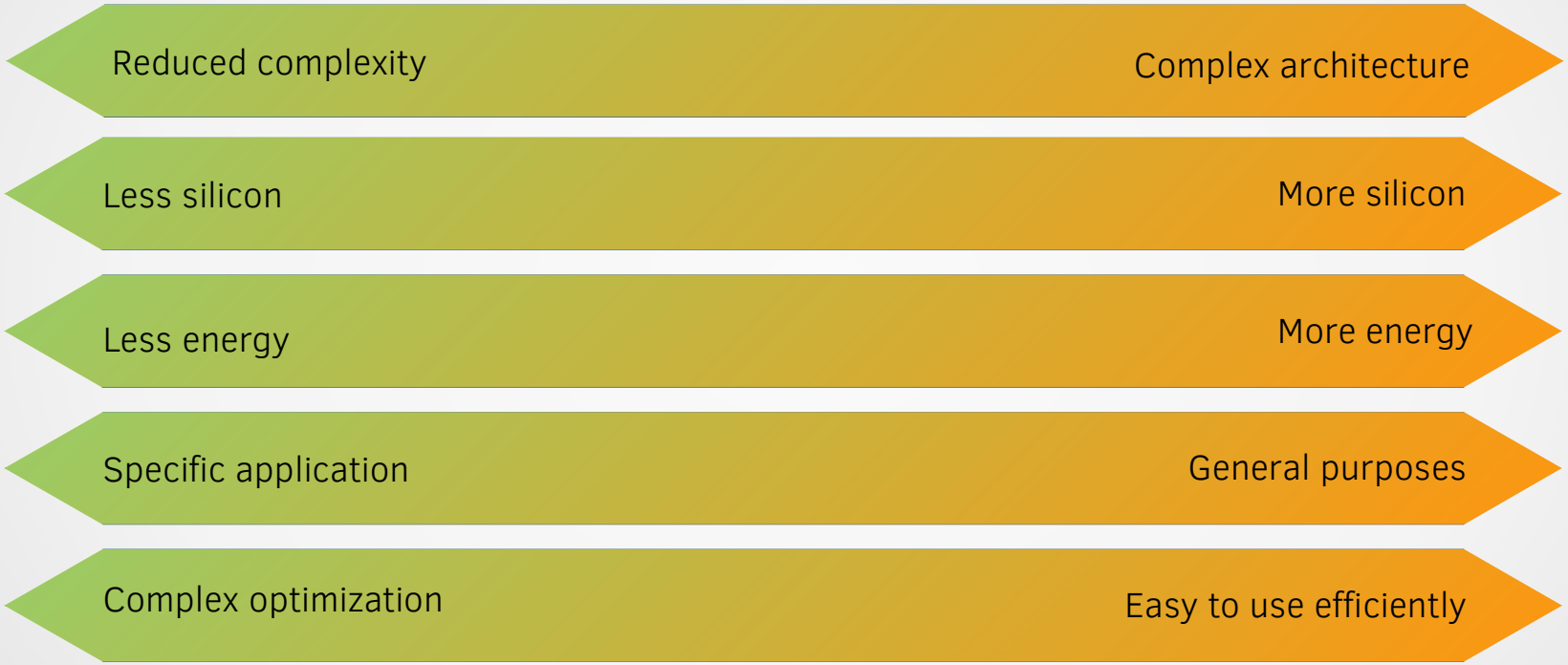


Based on Altera Stratix 10 and Xilinx Virtex Ultrascale+ characteristics

https://science.energy.gov/~/media/ascr/ascac/pdf/meetings/201612/Finkel_FPGA_ascac.pdf
https://www.xilinx.com/support/documentation/white_papers/wp375_HPC_Using_FPGAs.pdf
https://www.altera.com/products/fpga/stratix-series/stratix-10/overview.html ; https://indico.math.cnrs.fr/event/3435/material/slides/0.pdf

Processing unit general view

No miracle so far...



Programmed FPGA | GPGPU | Xeon Phi Power | Xeon SP ARMv8 | Xeon ARM | Pentium AMD Bullldozer

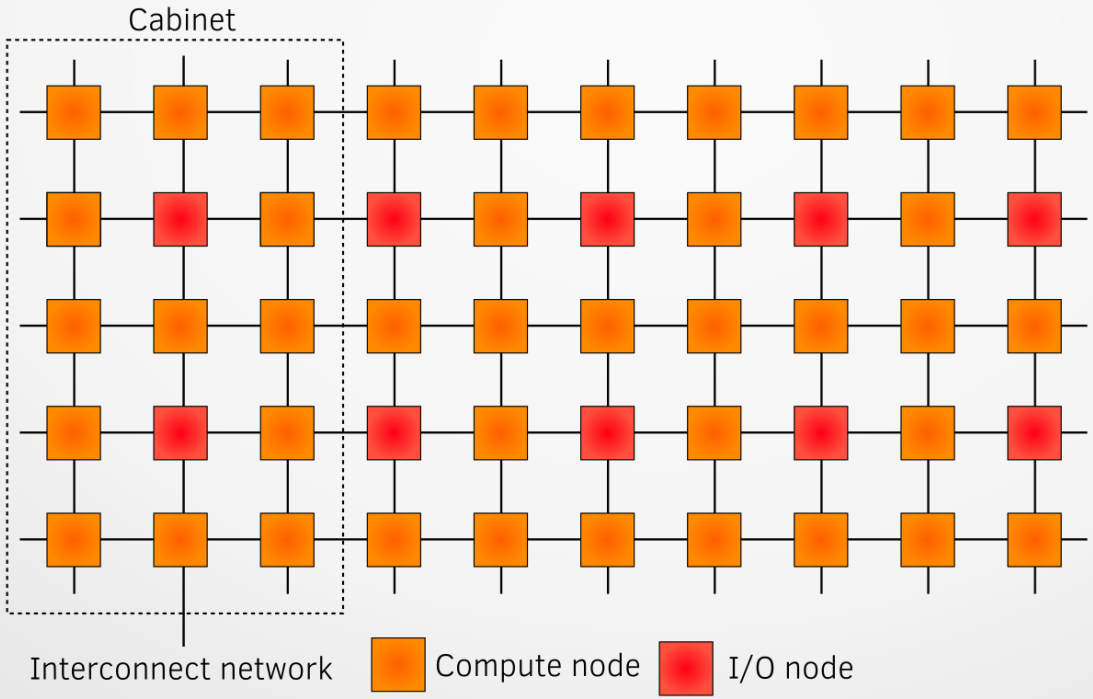
Hardware trends: multi-level massive parallelism

Exascale supercomputers will be composed of more than 100 000 interconnected nodes (several millions cores).

Level 1 – distributed parallelism between node

↔ MPI still the considered communication model for developers

▲ Complex software stack required to handle huge traffic and failure (losseless compression, resilence, huge pages, RMA...)



https://www.nextplatform.com/2018/03/19/argonne-hints-at-future-architecture-of-aurora-exascale-system/

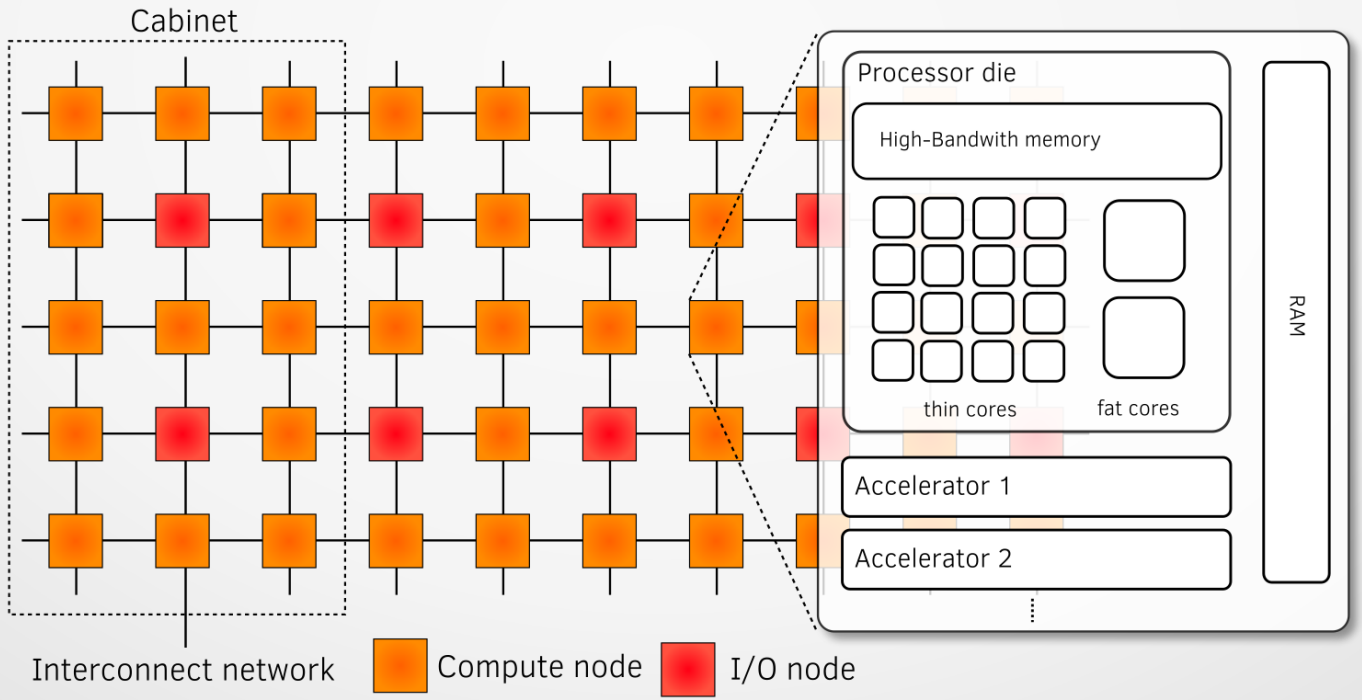
Hardware trends: multi-level massive parallelism

Each node composed of several sockets of several tens of cores/threads potentially connected to 1 or more accelerators

Level 2 – inner-node parallelism

↔ Shared parallelism with non-uniform memory access (NUMA) between cores or sockets

■▪ Heterogeneous computing with different accelerators, memory space and bandwidth



https://www.nextplatform.com/2018/03/19/argonne-hints-at-future-architecture-of-aurora-exascale-system/

Hardware trends: heterogeneity

Frontal nodes

Role: OS
Processing units: Server x86 (Xeon), Power, ARM

Compute node

Host

RAM

- Multi-core x86, Power, ARM CPUs
- Many-core x86, Power, ARM CPUs
- Skinny-Fat x86
- Big.LITTLE ARM

Role: OS, computation, post-processing, software
Programming model: MPI, OpenMP…

PCIe, NVLINK…

Unified memory

Accelerators

- Co-processors (Many-core), PEZY, MATRIX-2000
- GPGPU
- FPGA

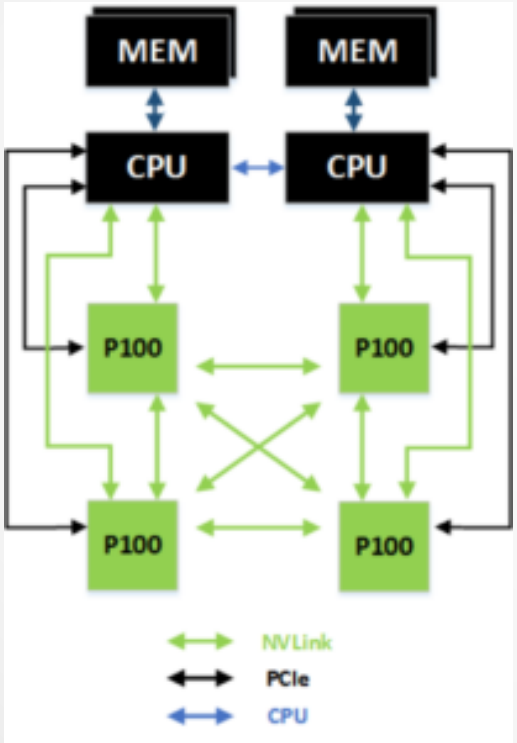
HBM

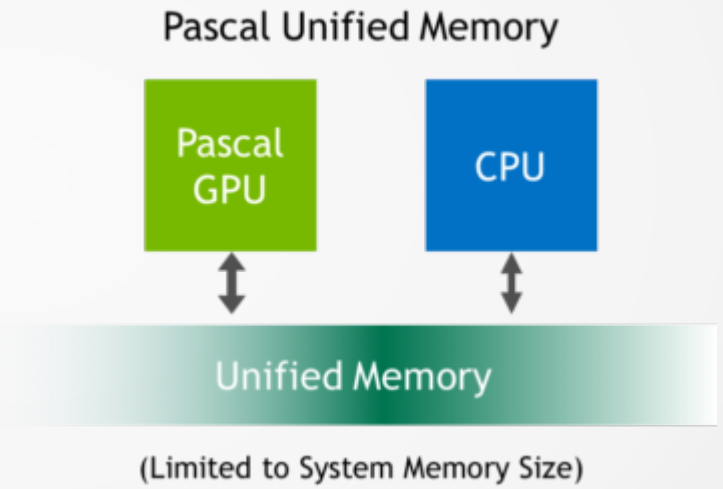
Role: computation, post-processing, rendering
Programming model: OpenCL, CUDA, OpenACC, OpenMP…

I/O nodes

Role: OS, conputation, post-processing, software, IO treatment
Programming model: MPI, IO stack

High-speed interconnect: openPOWER example





Hardware trends: vectorization

All computing devices can be seen as a vector machine

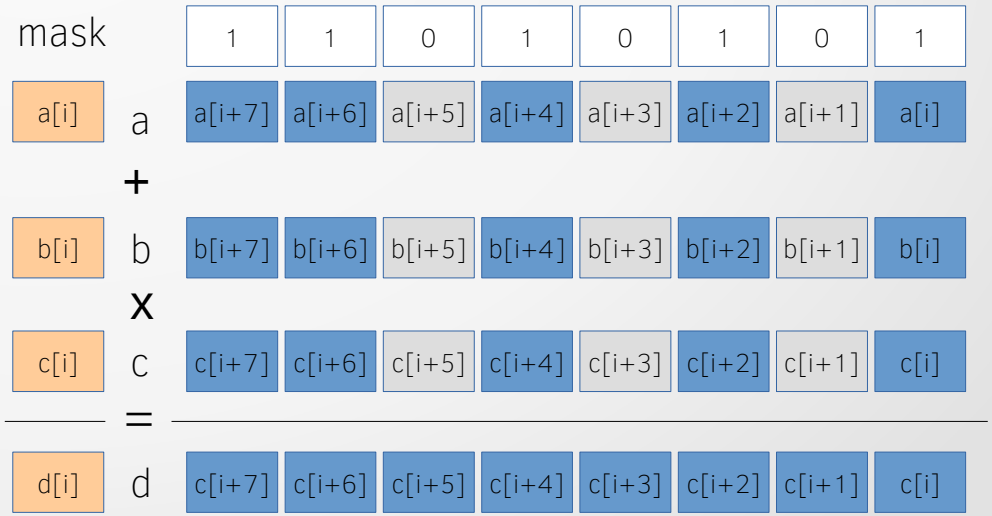
Level 3 – vectorization
- Capability to compute simultaneously a full vector of data with the same set of operations

CPU: Single instruction multiple data (SIMD) vectorization

GPU: vectorization through the large number of cores working simultaneously within a SM: Single instruction multiple thread vectorization (SIMT)

FPGA: vectorization through kernel encoding replica: several pipelines of the same kernel working simutaneously

For i from 1 to N :
    If (condition on i) :
        d[i] = a[i] + b[i]*c[i]



Single Instruction Multiple Data SIMD vision as in x86 Intel

Hardware trends: vectorization

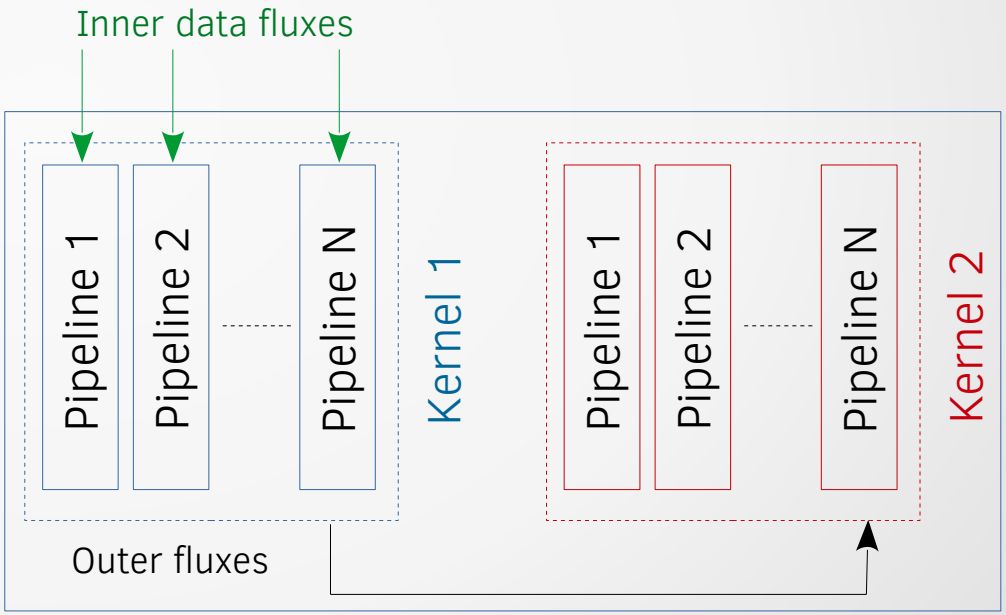
All computing devices can be seen as vector machines

Level 3 – vectorization
- Capability to compute simultaneously a full vector of data with the same set of operations

CPU: Single instruction multiple data (SIMD) vectorization

GPU: vectorization through the large number of cores working simultaneously within a SM: Single instruction multiple thread vectorization (SIMT)

FPGA: vectorization through kernel encoding replica: several pipelines of the same kernel working simutaneously

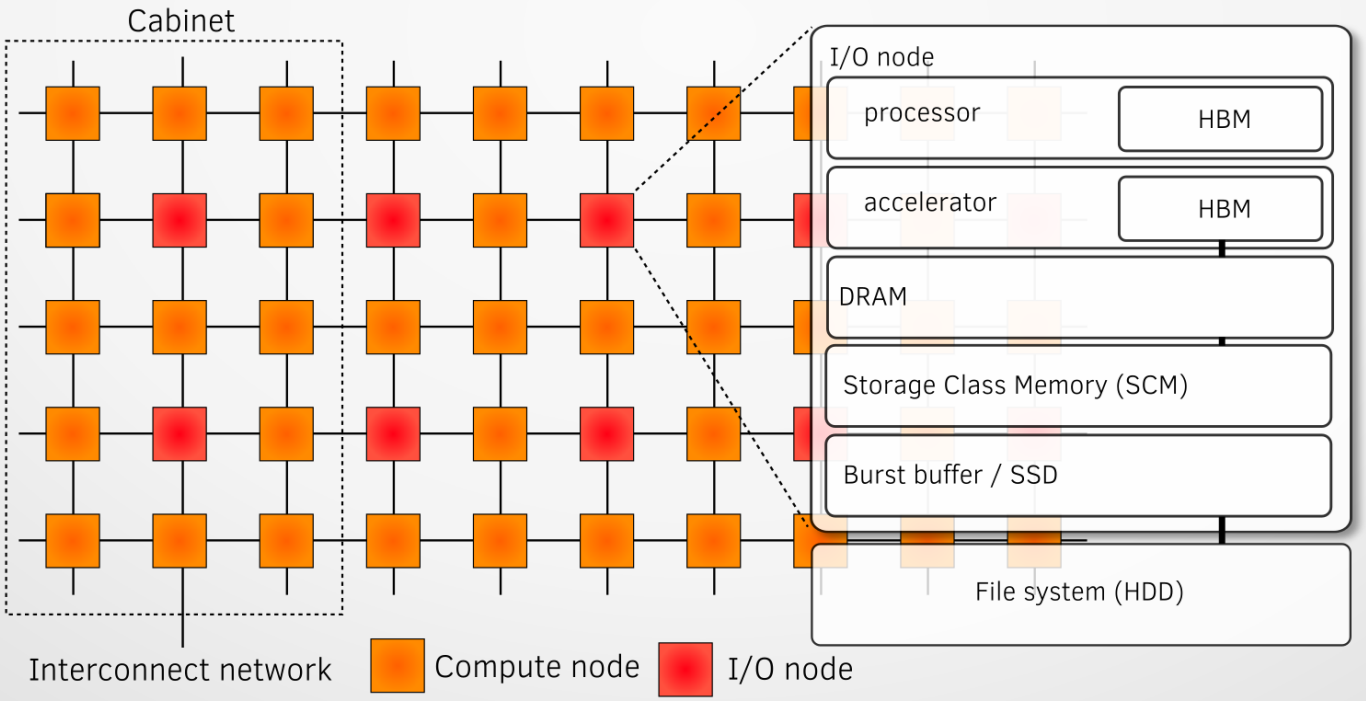


FPGA pipeline vectorization by the multiplication of the a pipeline from the same kernel

Hardware trends: I/O nodes

I/O nodes are specialized in fast input / ouput with a complex storage hierarchie

DRAM: volatile, highest bandwidth (~100Gb/s), lowest capacity (~10 Gb)
Storage Class Memory (SCM): volatile, lower bandwith than DRAM (~10 Gb/s) but higher capacity (~100 Gb)
SSD: also referred to as burst buffer, permanent, lower bandwith (~Gb/s) than SCM but higher capacity (~500 Gb)
HDD: permanent, lowest bandwidth (~100 Mb/s), highest capacity (~ Tb)



https://www.nextplatform.com/2018/03/19/argonne-hints-at-future-architecture-of-aurora-exascale-system/

Content

III. Programming challenges

Presentation of the codes

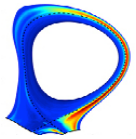
Smilei

LULI, LLR, CEA Saclay

Smilei is a Particle-In-Cell code for plasma simulation. Open-source, collaborative, user-friendly and designed for high performances on CPU super-computers, it is applied to a wide range of physics studies: from relativistic laser-plasma interaction to astrophysics.

- Python + C++
- MPI + OpenMP
- HDF5 + OpenPMD

⌂ http://www.maisondelasimulation.fr/smilei/



Tokam3X

CEA Cadarache

The TOKAM3X code simulates both limited and diverted plasma turbulence in full torus Tokamak geometry including the open field lines of the Scrape-off Layer (SOL) and the edge closed field lines region in the vicinity of the separatrix.

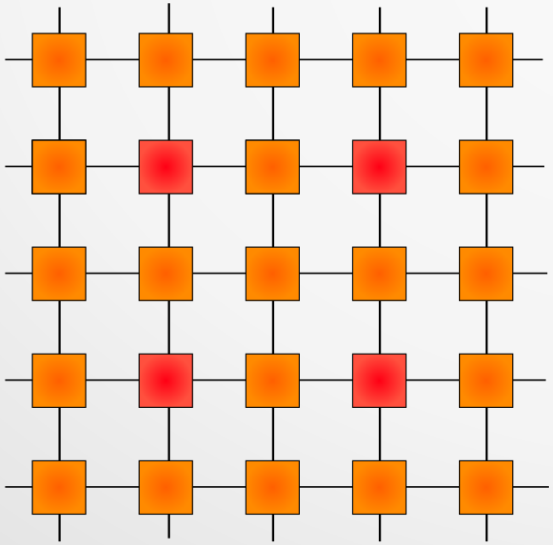
- Fortran
- MPI + OpenMP
- HDF5 sequential

Programming challenges – multi-level massive parallelism

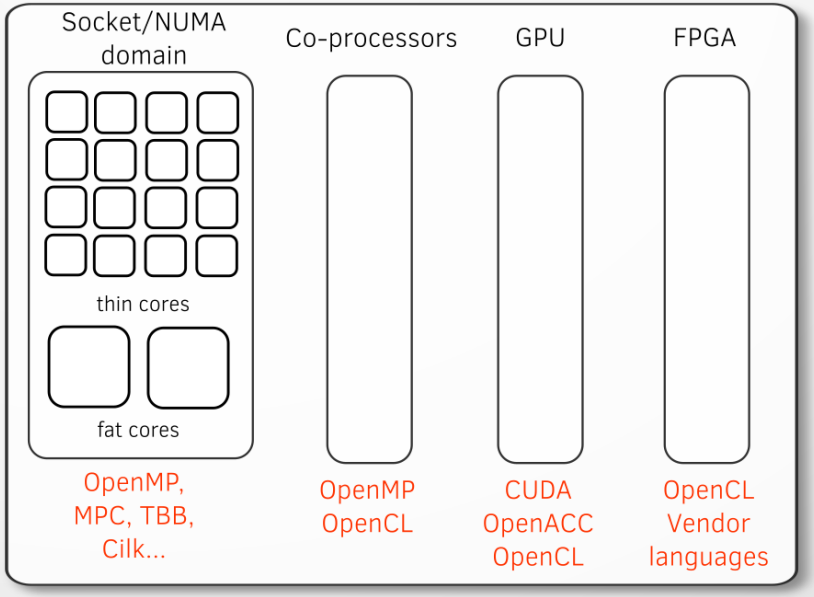
Time spent in communication will overcome computational time (almost the case for SMILEI)
Network capacity evolves less rapidly than computation power

Scalability:

- MPI + X (OpenMP, Open ACC...)
- Limit data movements, avoid global communications, use shared memory within a node
- Use non-blocking comunication with computation
- Dedicate cores to the communications, diagnostic processing, I/O
- Use accelerator in symmetric mode to exploit the full node computational power



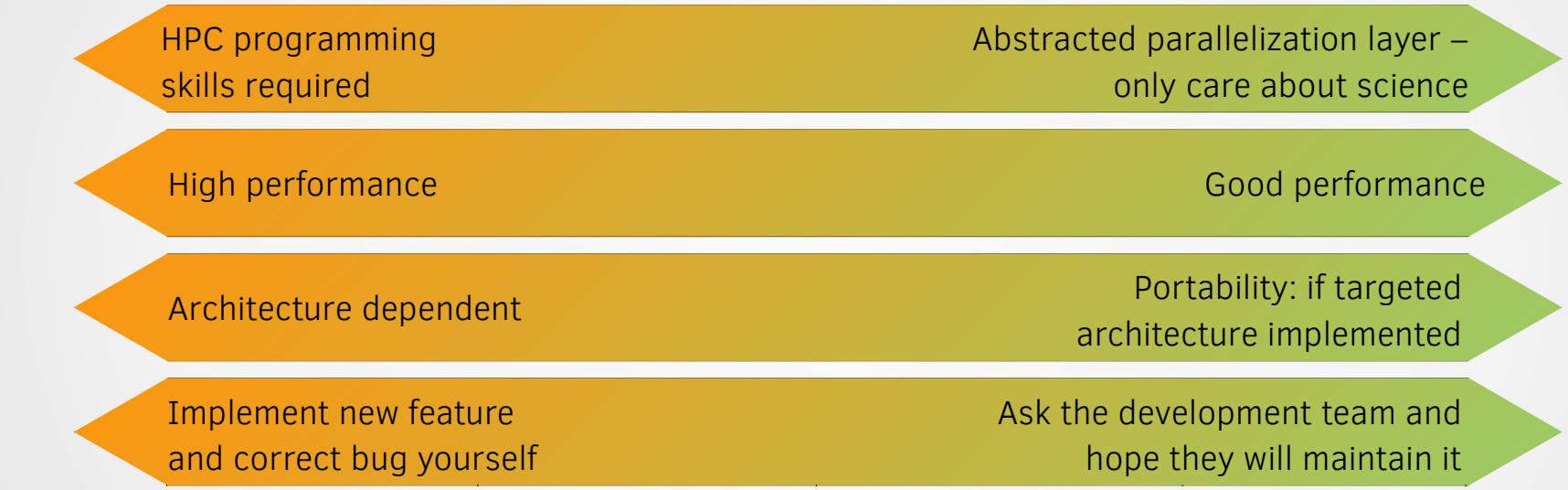
MPI, PGAS, MPC
Inter-node parallelism



Intra-node parallelism

Programming challenges – architecture dependency

How to deal with architecture multiplication?



Low level API: MPI, CUDA, OpenCL

Directives: OpenMP, OpenACC

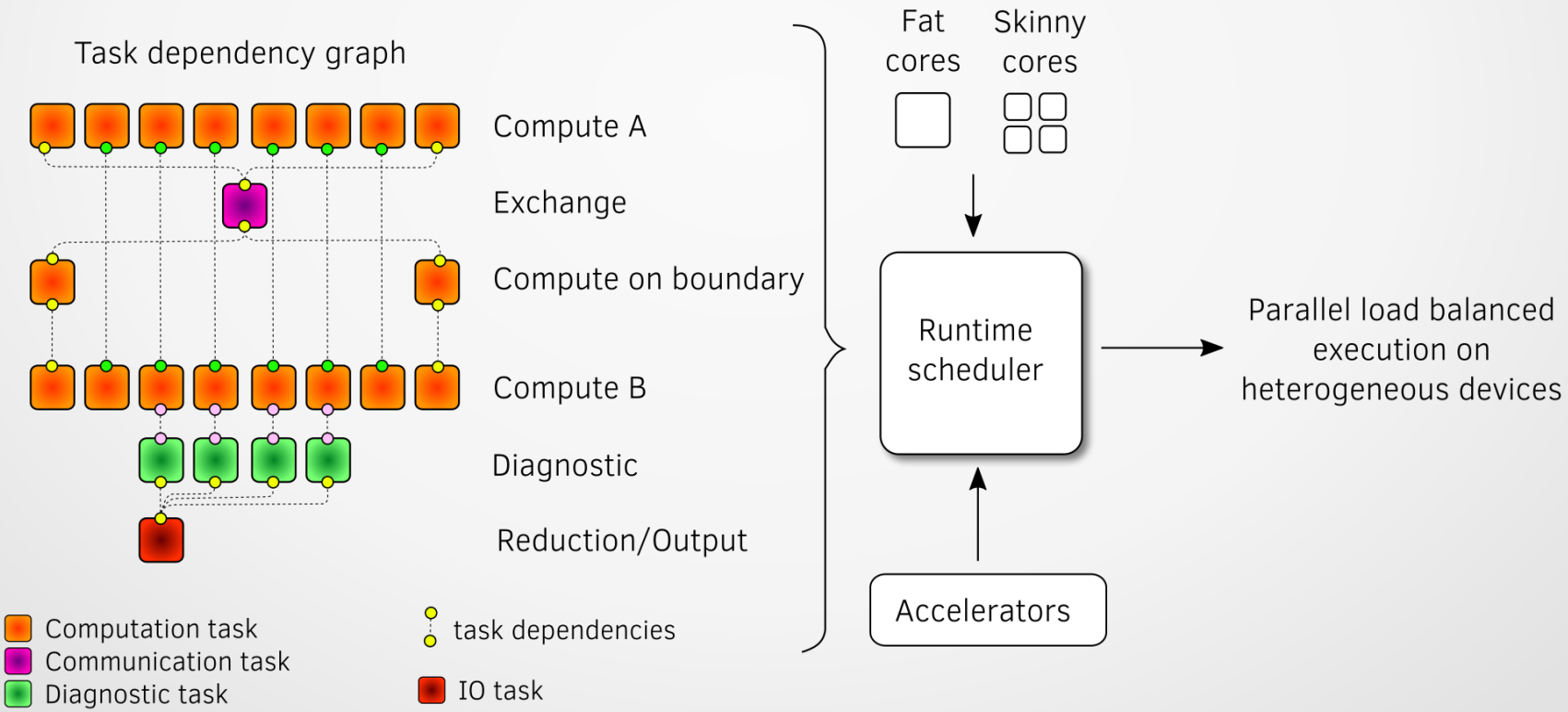
Libraries/ Abstraction: Kokkos (Data) Alpaka

Unified parallel: PGAS, Chapel, X10, MPC

Domain Specific languages (DSL): Nabla

Programming challenges – Task-based approach

► An application can be divided temporally and spatially into inter-dependent tasks of different natures (computation, communication, IO)
► Using a smart runtime scheduler, tasks can then be run concurrently/asynchonously in parallel on a large number of cores/ on several architectures.
► To use a task-based model can speedup the code by overlaping IO and communication with computation.
► Factor of performances: grain size (fine, coarse), scheduler, dependency graph
► OpenMP task, StarPU, libKOMP...

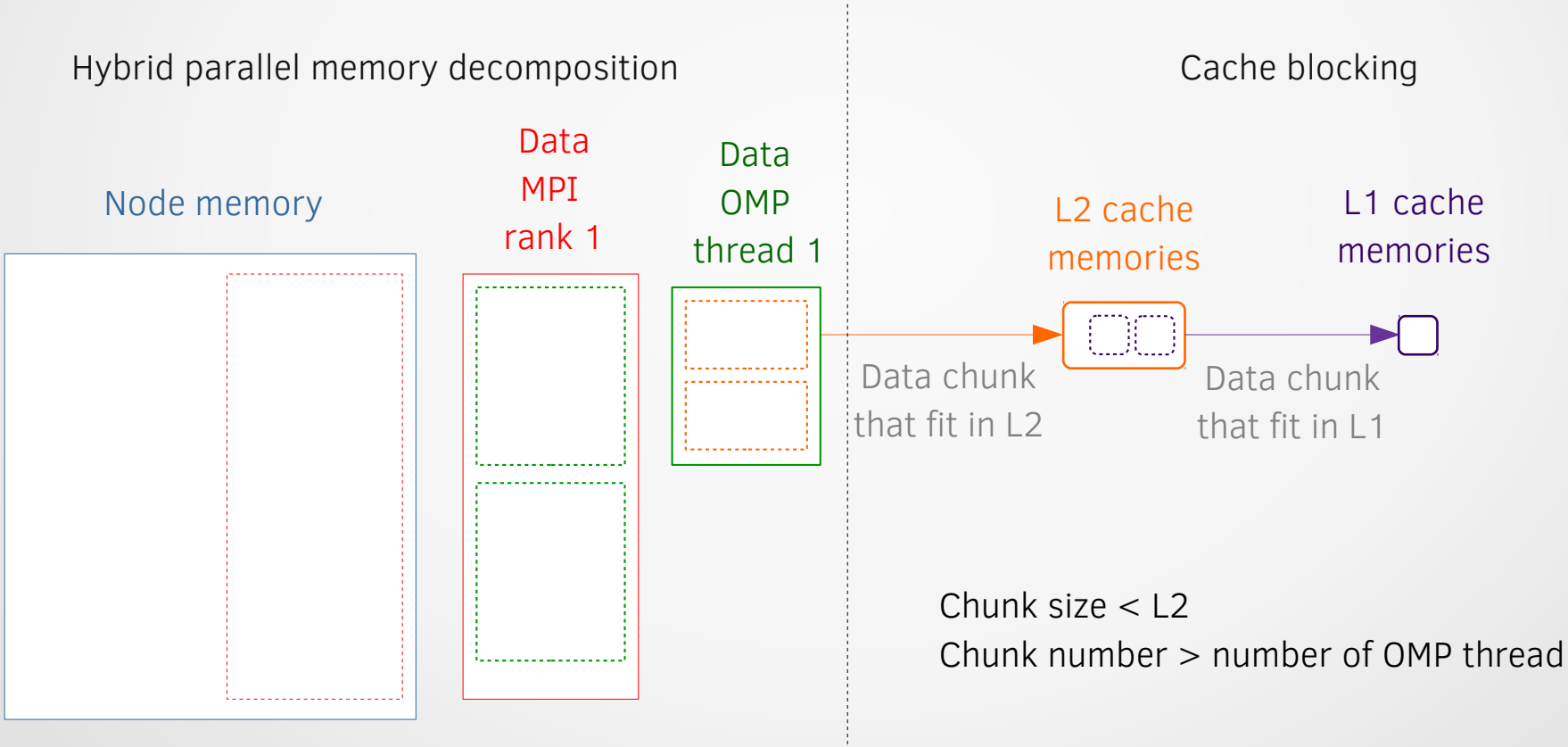


https://hal.inria.fr/hal-00796253

Programming challenges – cache blocking + vectorization

Decomposition of the workload into chunk that fit the targeted level of cache :

- ► Diminish number of cache misses / reuse data in cache
- ► Reduce memory bandwidth pressure : improve performance of memory bound algorithms



Programming challenges – IOs

- Dealing with specific IO nodes
- Dealing with complex IO hardware stack
- Scalable and concurrent reading/writing process
- Portability / Flexibility
- Checkpointinig for fault tolerance
- Storage capacity increases less rapidly than computational power: data treatment on the fly, in-situ visualization, in-situ analysis via machine learning

Basic	Parallel	Portability	Output + postprocessing	Flexibility	Checkpointing	In-situ	IO abstraction
POSIX	MPI-IO	HDF5 NetCFD	XIOS	ADIOS SIONLib	FTI	FlowVR VisIt	PDI

Low-level → High-abstraction / many capabilities

Additional challenging aspects not covered here

► Code output precision (how to tune precision to speed-up the code)

► Failure prediction and correction

► Code optimization to reduce energy-consumption (not just a question of computer design)

Conclusion

► No breakthrough expected, forthcoming HPC hardware is the continuity or adapated from other domains such as the mobile world

► Scalability on millions of core will be challenging: multi-level parallelism, communication limitation, communication/computation/diagnostic overlapping, task model

► Core optimization: smart use of cache levels, vectorization

► Heterogenity will be challenging: concurrent use of resources to reach peak performance (ovelapping, task...)

► Disk memory will be limited and IO stack more complex: parallele use of IO nodes, dedicated resources to diagnostics and IOs, asynchronous treatment taking advantage of intermediate storages

►Focusing on a specific architecture can be dangerous to run efficiently on forthcoming machines

►Exascale optimization efforts useful for current architecture as well