Some issues about exascale computing

About this document

- This document is intended to show/describe some issues about future exascale HPC system.
- Many issues are really oversimplified, this document is done just to give a view of the main issues about exascale computing
- There are many other issues under the carpet… ☺

(trivial) Definitions

- ✓Exa → 10^18 (i.e. 1’000’000’000’000’000’000)
- ✓Peta → 10^15 (i.e. 1’000’000’000’000’000)
- ✓Tera → 10^12 (i.e. 1’000’000’000’000)
- ✓Giga → 10^9 (i.e. 1’000’000’000)
- ✓Mega → 10^6 (i.e. 1’000’000)
- ✓Kilo → 10^3 (i.e. 1’000)

- ✓floating point Unit (FPU) = units devoted to perform floating point operations

1. What is EXASCALE computing?
2. Physical constraint for Exascaling
3. Performance & Coding issues: a simple example
4. Some issues for next EXASCALE system
5. Some issues for an EXASCALE-aware system
6. CFD4Exascale

What is EXASCALE

- With EXASCALE we intend a system able to perform, at least as theoretical peak, one or more EXAFlops.
- 1 EXAFlops system is a system able to deliver 10^18 Floating point operations for second.
  - ✓It is supposed to be operative around 2020
  - ✓Power consumption should be lower than 30 MW

The questions are:

✓How real world code (like OpenFoam) will perform on this kind of systems?

✓Which modification has to be done to exploit performance?

How to compute floating point operations?

- To obtain the theoretical (peak) value of Flops obtained from a single CPU we must multiply
  - ✓Clock Frequency
  - ✓Number of computational (physical) core in the CPU
  - ✓Number of floating point operations that can be delivered for each clock cycle from every core
- Example (Intel Phi KNL)
  - ✓Clock Frequency= 1.4 GHz
  - ✓Number of computational (physical) core in the CPU = 68
  - ✓Number of floating point operation that can be delivered for each clock cycle from every core = 64 (Single precision) or 32 (Double precision)
  - ✓1’400’000’000*68*32 = 3 Tflops (Peak Perfromance for Double Precision)

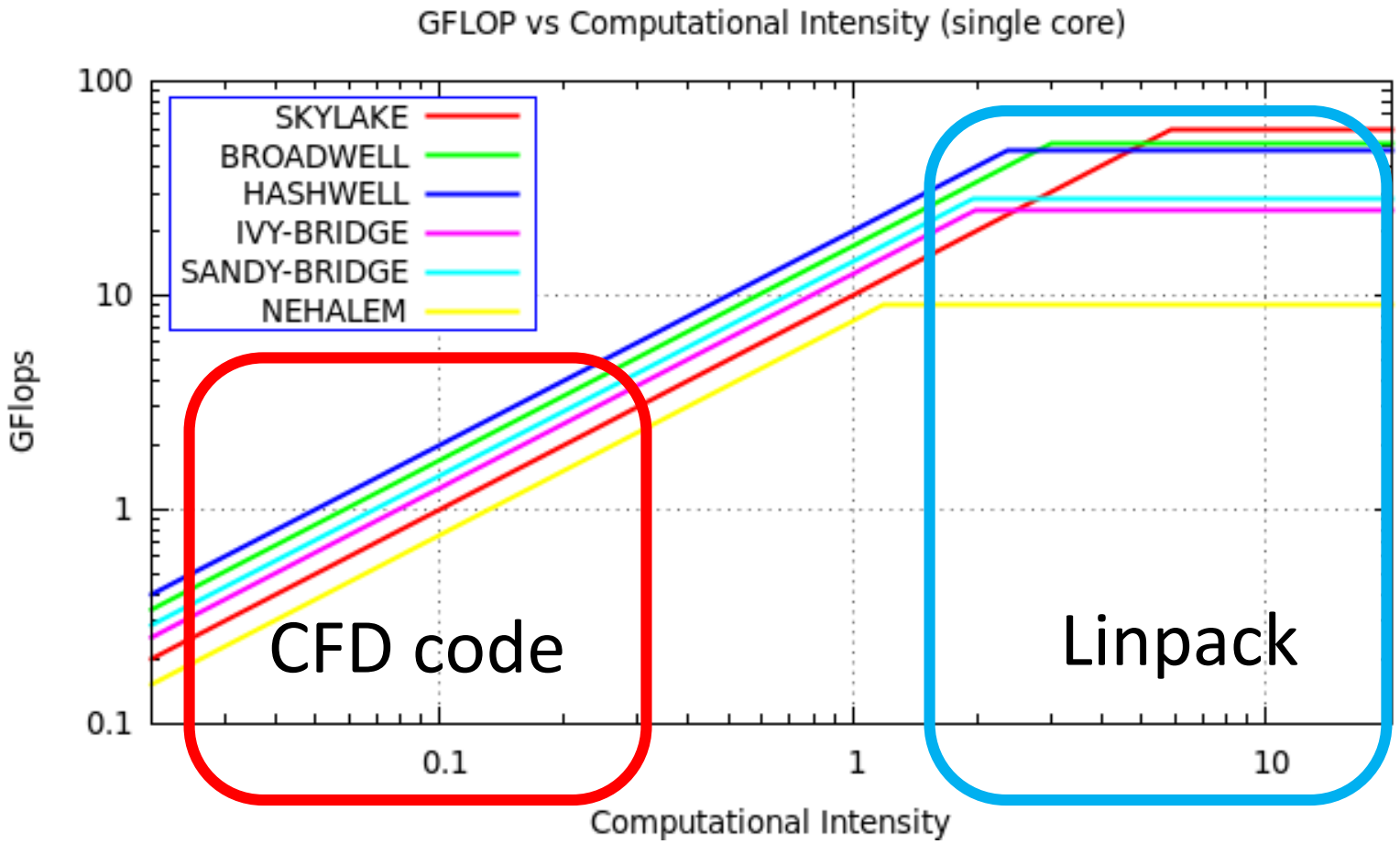
CAVEAT about FLOPs?

- Peak Performance cannot be really achieved: in implicit way we mean that:
  - ✓All O.S interactions are removed
  - ✓All data needed for operation are available with no latency

- Real world application can get only few percentage of peak performance, as reference:
  - Linpack (e.g. Matrix-Matrix multiplication) → around 60-80% of peak performance
  - CFD core → usually less than 10% of peak performance

More real performance model: roofline

- Performance ordered with respect to computational intensity:
  - ✓CI =#flops/#byte
  - ✓CI > 1 FLOPs limited
  - ✓CI < 1 BW limited



1. What is EXASCALE computing?
2. Physical constraint for Exascaling
3. Performance & Coding issues: a simple example
4. Some issues for next EXASCALE system
5. Some issues for an EXASCALE-aware system
6. CFD4Exascale

Moore Law: theory

- “Number of transistors for the same size doubles every 18 Month”

- Stated in mid-sixties, still valid now
- It means that the improvements in manufacturing technology allows to increase the circuit density but
  - ✓Take care of power consumption
- Now CPU has more than billions (> 1’000’000’000) of transistors

Moore Low: Implementation

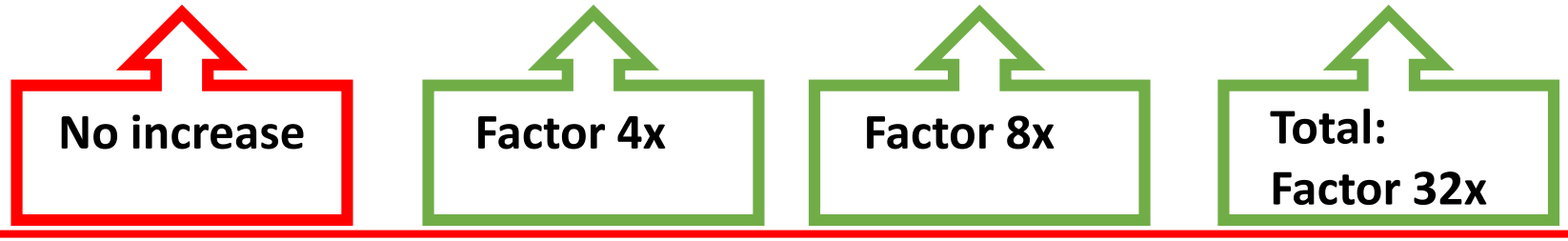
- Up to 2006
  - ✓Dennard scaling: “as transistors get smaller their power density stays constant, so that the power use stays in proportion with area: both voltage and current scale (downward) with length”.
  - ✓You can raise clock frequencies from one generation to the next without significantly increasing overall circuit power consumption
  - ✓ 2005–2007 Dennard scaling appears to have broken down: power consumption explodes!

- From 2006 to now
  - ✓Keep clock frequency stable (or with a slight decrease)
  - ✓Increase # computational unit (i.e. CORE) → Core level parallelism
  - ✓Increase # floating point units per core →Instruction level parallelism
  - ✓Increase complexity of floating point units (i.e. vectorization) → Instruction level parallelism
  - ✓Side effect: to exploit performance huge data parallelism/concurrency is requested!!!!

Moore Low: Implementation/2

- Some figures about CPU evolution (Intel CPU@CINECA, period 2010/17)

CPU (codename)	Clock Frequency	Number of core	Flops cycle (DP)	Peak Perf.
Xeon E5645 (Westmere)	2.4 GHz	2x6	4	115 GFlops
Xeon E5-2687W0 (Sandy Bridge)	3.1 GHz	2x8	8	396 GFlops
Xeon E5-2670v2 (Ivy Bridge)	2.5 GHz	2x10	8	400 GFlops
Xeon E5-2630v3 (Hashwell)	2.4 GHz	2x8	16 (AVX-256bit)	614 GFlops
Xeon E5-2697v4 (Broadwell)	2.3 GHz	2x18	16 (AVX-256bit)	1325 GFlops
Xeon Platinum (Skylake)	2.1 GHz	2x24	32 (AVX-512bit)	3225 GFlops



1. What is EXASCALE computing?
2. Physical constraint for Exascaling
3. Performance & Coding issues: a simple example
4. Some issues for next EXASCALE system
5. Some issues for an EXASCALE-aware system
6. CFD4Exascale

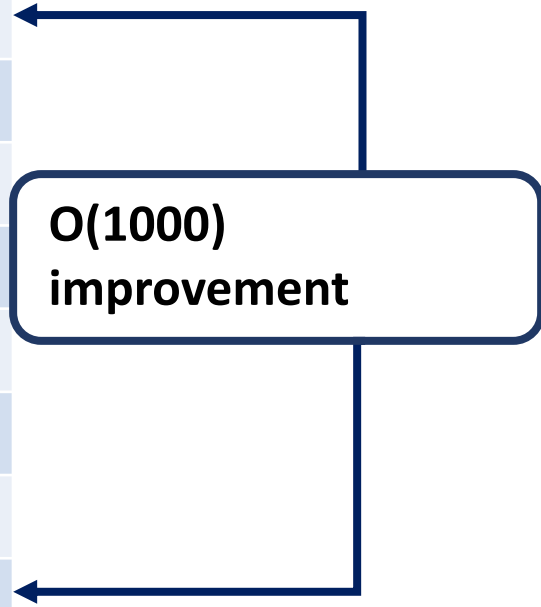
Single CPU performance: users side

- A single CPU is a parallel systems with many core (up to O(100))
- Serial/Single core performance is going to be meaningless
- Parallel programming is mandatory to exploit CPU performance
- Programming style can affect seriously performance
- Different way of writing a program, even if correct from numerical point of view can seriously affect performance, of more than one order of magnitude (e.g. loop cache friendly/unfriendly)

Single core performance: users side

- Time to perform simple matrix-matrix product, size = 4’096, Intel SKL, serial performance
- Theoretical peak performance: 67.2 GFlops

Compiler Options	Loop cache friendly	GFlops	Peak (%)
-O0	yes	0.38	0.6%
-O1	No	0.06	0.1%
-O1	Yes	2.08	3.1%
-O2 (default option)	Yes	2.87	4.3%
-O3	Yes	9.34	14%
-xCORE-AVX512	yes	3.14	4.7%
-O3 -mtune=skylake	yes	9.22	14%
-O3 -xCORE-AVX512	yes	13.5	20%
Using intel MKL (i.e. BLAS)	-	56.7	84%



Single node (2xCPU) performance: users side

- Time to perform simple matrix-matrix product, size = 16’384,Intel SKL, parallel performance

Parallelization	Num threads	GFLOPs	Peak (%)
Hand made OpenMP	**8**	**66**	**2.4 %**
Intel MKL (i.e. BLAS)	**8**	**443**	**14 %**
Hand made OpenMP	**16**	**123**	**3.8 %**
Intel MKL (i.e. BLAS)	**16**	**869**	**27 %**
Hand made OpenMP	**32**	**200**	**6.2 %**
Intel MKL (i.e. BLAS)	**32**	**1470**	**46 %**
Hand made OpenMP	**48**	**242**	**7.5 %**
Intel MKL (i.e. BLAS)	**48**	**1631**	**51 %**

- Simple OpenMP (no blocking/unrolling…) could be not enough…

Single node performance: size matters!!

- Time to perform simple matrix-matrix product, Intel SKL, parallel performance using intel MKL

Size	Num threads	GFLOPs	Peak (%)
1024	48	91.6	2.8 %
4096	48	192	6.0 %
8192	48	1267	39 %
16384	48	1669	52 %
32768	48	1985	62 %

- To exploit performance suitable problem size are needed to fulfill Instruction level parallelism

1. What is EXASCALE computing?
2. Physical constraint for Exascaling
3. Performance & Coding issues: a simple example
4. Some issues for next EXASCALE system
5. Some issues for an EXASCALE-aware system
6. CFD4Exascale

Way to build EXASCALE/1

- One single CPU with EXA-Hz frequency is not feasible ☹
- Power dissipation of all system MUST be below 30MW
- For power dissipation reason frequency should be around 1/2 GHz
  - ✓with 1FLop/core we need 10^9 core → too much
  - ✓O(10-100) core level parallelism needed

- Today technology (e.g. intel Xeon Phi)
  - → Up to 68 core per CPU
  - → Up to 32 floating point operation per core

  - → We need to integrate O(10^6) CPU

Way to build EXASCALE/2

- A simple table: for 1 Exascale system we have:
  - M nodes each with N Peak TFlops

Single node performance (Tflop)	Number of Nodes
1	1’000’000
4	250’000
8	125’000
25	40’000
50	20’000
100	10’000
200	5’000

NOW!!!

Way to build EXASCALE/3

- No clear recipe so far

1. Many O(100’000) nodes: serious network issues
2. Using accelerators (e.g. GPGPU): needs to rewrite the code
3. Who knows?

All issues (programming, HW, algorithmic,….) has to be explored to achieve the results

Unfortunatly the silver bullet doesn’t exist ☹

Main EXASCALE feature

- MTTI=mean time to Interrupt

Systems	2009	2011	2015	2018
System Peak Flops/s	2 Peta	20 Peta	100-200 Peta	1 Exa
System Memory	0.3 PB	1 PB	5 PB	10 PB
Node Performance	125 GF	200 GF	400 GF	1-10 TF
Node Memory BW	25 GB/s	40 GB/s	100 GB/s	200-400 GB/s
Node Concurrency	12	32	O(100)	O(1000)
Interconnect BW	1.5 GB/s	10 GB/s	25 GB/s	50 GB/s
System Size (Nodes)	18,700	100,000	500,000	O(Million)
Total Concurrency	225,000	3 Million	50 Million	O(Billion)
Storage	15 PB	30 PB	150 PB	300 PB
I/O	0.2 TB/s	2 TB/s	10 TB/s	20 TB/s
MTTI	Days	Days	Days	O(1Day)
Power	6 MW	~10 MW	~10 MW	~20 MW

1. What is EXASCALE computing?
2. Physical constraint for Exascaling
3. Performance & Coding issues: a simple example
4. Some issues for next EXASCALE system
5. Some issues for an EXASCALE-aware system
6. CFD4Exascale

Writing an exascale-aware code

- Very hard target
- Different skill needed
  - ✓Numerical Analysts
  - ✓Computer Scientists
  - ✓OS/Compiler/Networking specialist
  - ✓HW experts
  - ✓End Users
- Different HW/configurations to play with
- Different tool to use
  - Profilers, Compilers, Debuggers, Simulators….

Writing an exascale-aware code/1

- Explore different parallel-paradigm
  - ✓Fat node (O(100) core): hybrid parallelization must be exploited
  - ✓External accelerator
    - GPGPU (e.g. NVIDIA Pascal)
    - FPGA
- Size matters
  - ✓Define correct test case
  - ✓Deep profiling to find hotspot
  - ✓Introduce (light) profiling in the core order to guide optimization

Writing an exascale-aware code/2

- Exascale system will need high level parallelism:
- An order of 1’000’000 concurrent operations is the target
  - ✓Are the HW system powerful to feed the single NODE?
  - ✓Are the numerical algorithm suitable for this system?

Writing an exascale-aware code/3

- I/O, pre/post processing
- Exascale means problem O(10^9) or even more grid-points: it means PB of data
  - ✓It must be handled in correct way to be effective. Parallel I/O is mandatory
  - ✓Pre/post processing must be parallel too, no enough memory on a single node!!!
  - ✓Domain decomposition could be time-consuming and error-prone. Inefficient domain decomposition over O(100000) tasks could lead to very low performance
  - ✓ High flexibility for domain decomposition: the user can change the number of task easily: the number of available nodes can vary significantly

Writing an exascale-aware code/3

- Not only a single big simulation has to be performed but a complete workflow
- Fine tuning will be different for different simulation: This means
  - ✓Spread Knowledge about code/optimization
  - ✓Spread Knowledge about HW feature & issues
  - ✓Spread Knowledge about pre/post processing tools

→ Deployment

→ Dissemination

Writing an exascale-aware code/4

- To be successful all the player involved must share their knowledge & expertise
- …..

1. What is EXASCALE computing?
2. Physical constraint for Exascaling
3. Performance & Coding issues: a simple example
4. Some issues for next EXASCALE system
5. Some issues for an EXASCALE-aware system
6. CFD4Exascale

CFD4Exascale proposal

- ✓CFD4exascale focuses on the technologies necessary to transition CFD from its current peta-scale performance point towards exascale deployment. CFD is widely used in industry for design, analysis and research in engineering and safety. This proposal supports the Strategic Research Agenda proposed by ETP4HPC, the European Technology Platform for High-Performance Computing
- ✓The consortium gathers acknowledged experts in Europe from the widest CFD spectrum. It receives worldwide active interest, placing European technology at the heart of the drive for HPC deployment. We combine algorithm specialists, code release authority, supercomputing centres, application specialists and major industrial end-users with the need and means to exploit the technologies under assessment.
- ✓The open-source and freely available common assessment CFD platform OpenFOAM®, widely used by tens-of-thousands of engineers, will provide the basis for CFD4exascale activities, ensuring open, free and public deployment and exploitation during and after conclusion of the project.

CFD4Exascale: partners

- ESI-OCFD
- CINECA
- HLRS
- Wikki
- University of Darmstadt (TuD)
- CFD-Berlin
- E4
- University of Zagreb (FSB)
- Politecnico di Milano (PoliMI)

CFD4Exascale: Work Package (& Lead)

- ✓ **WP1: Management, ESI-OCFD**
- ✓ **WP2: Demonstrate current peta-scale performance Lead, Cineca (+HLRS, TuD, OCFD)**
- ✓ **WP3: Methods for extreme parallelism Lead, FBS (+CFDB, PoliMi, Wikki, OCFD)**
- ✓ **WP4: Data management, comms/storage/retrieval Lead, HLRS (+E4, Cineca, Wikki, TuD)**
- ✓ **WP5: Verification, Validation, Uncertainty Quantification Lead, CFDB (+OCFD, Wikki)**
- ✓ **WP6: Software Stewardship, Test performance and scaling on many-core configurations Lead, TuD (+Cineca, HLRS and E4)**
- ✓ **WP7: Deployment; application test of run-ready software on many-core configurations Lead, Wikki (+All)**
- ✓ **WP8: Exploitation and Dissemination Lead, ESI-OCFD**

links

- https://en.wikipedia.org/wiki/FLOPS
- http://www.dolbeau.name/dolbeau/publications/peak.pdf
- https://en.wikipedia.org/wiki/Roofline_model
- https://en.wikipedia.org/wiki/Moore%27s_law
- https://en.wikipedia.org/wiki/Dennard_scaling


CFD4exascale