Lowain Processors

Luděk Kučera

October 11, 2017

The Exascale Dream

On September 28, 2008, the U.S. agency DARPA (Defense Advanced Research Projects Agency) published a document called
"ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems" (http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf),
where a goal of building a supercomputer capable of performing 10^{18} floating point double precision arithmetic operations per second (1 ExaFlop/s) was stated with the year 2016 as the target. Now, in 2017, the most powerful supercomputer in the U.S.A. is still below 30 PFlop/s.

China entered the exascale game by upgrading Tianhe (Milky Way) computer in 2013: Tianhe-2 performance (as of June 2013) was 54.9 PFlop/s peak and 33.86 PFlop/s Linpack, thus being slightly more than twice as powerful as the second computer in the Top500 list, Titan (U.S.A.).

The next step forward was made by China in 2016 by Sunway TaihuLight with 125 PFlop/s peak, 93 PFlop/s Linpack, the first computer with the peak computing power over 100 PFlop/s.

At the moment of writing this document (October 2017), Sunway TaihuLight remains to be the most powerful supercomputer of the planet, followed by Tianhe-2 that has just recently been upgraded to 95 PFlop/s peak. Third place (by Linpack performance) is occupied by Piz Daint (Cray XC50, installed in Switzerland, 25.3 PFlop/s peak, 19.59 PFlop/s Linpack) that pushed Titan back to the 4-th place in June 2017.

Not too much is known about Chinese exascale plans, some experts speculate that the first exascale computer could be Tianhe-3 sometime in 2020.

On 29 July 2015, U.S. President Obama signed an executive order creating a National Strategic Computing Initiative (NSCI). As a part of NSCI, the Exascale Computing Project (ECP), a collaborative effort of two U.S. Department of Energy organizations - the Office of Science (DOE-SC) and the National Nuclear Security Administration (NNSA), aims at building an exascale supercomputer in early 2020's.

The U.S.A. are preparing to get back the HPC leadership by two systems built under U.S. DOE CORAL pre-exascale initiative that should be operational soon (2018?): Sierra (120-150 PFlop/s), Summit (about 200 PFlop/s), both based on IBM Power9 chips and Nvidia Volta architecture GPU's.

Another U.S. project within the CORAL pre-exascale initiative, Aurora will be based on Intel chips. The original plans (180 PFlop/s, delivery in 2018, based on Intel 10 nm third generation "Knights Hill" Xeon Phi processors) have just recently been radically changed (one can even say the original plan was cancelled and replaced by another one). The new Aurora with the expected delivery in 2021 would be the first U.S. exascale supercomputer with the computing power 1000 PFlop/s.

The change of plans might be connected with a slow-down in Intel's transition to 10 nm technology; it is also possible that it has been found that 10 nm is not enough for the exascale, and the plan has been postponed with the prospect of 7 nm process in 2020. The DOE "rebaseline" review also announces that the new Aurora will be "exciting with many novel technology choices that can change the way computing is done", without specifying the "novel technology choices". It is speculated that this might mean the future Intel's silicon photonics and 3D XPoint memory.

The European Exascale Dream

On March 23, 2017, as a part of celebrations of the 60-th aniversary of the Treaties of Rome, Europe entered into the exascale race by adoppting a declaration, signed by ministers of 7 EU countries (FR, DE, IT, L, NL, P, SP)
(see the reference "EuroHPC declaration"
in https://ec.europa.eu/digital-single-market/en/news/eu-ministers-commit-digitising-europe-high-performance-computing-power),
that calls, among others, for

- "The procurement processes for the acquisition of two world-class pre-exascale supercomputers preferably starting on 2019-2020, and two world-class full exascale supercomputers preferably starting on 2022-2023", and for
- "The development of high-quality competitive European technology, its optimisation through a co-design approach and its integration in at least one of the two exascale supercomputers".

The Declaration also "invites all Member States and Associated Countries to join EuroHPC".

While the use of terms like "world-class", "preferably", "high-quality", "competitive" might seem funny in this context, it is not easy at all to understand what the Declaration says. Especially the sequences "The procurement processes for the acquisition" and "high-quality competitive European technology" are very vague.

I will understand that, in this context, "acquisition" does not mean "buying" (outside Europe or from a subsidiary or a representation of a non-European company), but making, building or manufacturing in Europe.

I wil also understand that, in the context of supercomputing, "technology" to be integrated into at least on exascale supercomputer is "processors". Another system that is so important that its source is reported when describing a supercomputer is an interconnect fabric. However, it is common to report either just processors or processors+interconnect when describing the technology that has been used (not just the interconnect).

"European technology" might also be a complicated term in the same sense as asking whether Nvidia GPU's are U.S. technology, being fabricated by TSMC.

In this way, I will understand the Declaration so that it describes

- the minimal goal: an exascale supercomputer that is built in EU, and uses parts (processors etc.) bought outside EU
- the maximal goal: an exascale supercomputer that is built in EU, and uses processors and an interconnect fabric designed and manufactured in EU as well

In some sense, the Declaration goals are very simple: neither an exascale supercomputer can be built by a consortium of universities and supercomputing centers, nor (within given time limits) a new company can be created to fulfill such tasks.

In the U.S.A., supercomputers like the future Summit, Sierra, Aurora, and existing Titan, Sequoia, Cori, and others will be or were made under a contract of federal agencies with private companies like Cray, IBM, Intel, Nvidia. Programs like Exascale Computing Project support the contracts by providing results of research by academic and federal institutions.

In Europe, the only company that is able to build "pre-exascale supercomputers preferably on 2019-2020, and full exascale supercomputers on 2022-2023" is the French company Bull (a subsidiary of Atos). Similarly as U.S. Cray, Bull builds supercomputers, has its own interconnect product, but does not make silicon chips.

As of July 2017 (see www.top500.org), the most powerful supercomputer built by Bull was Mistral (at DKRZ - Deutsches Klimarechenzentrum) with 3.96 PFlop/s peak and 3.01 PFlop/s Linpack, and there are 9 further Bull computers in Top100 (for Czech reader - Salomon in IT4I Ostrava has 2 PFlop/s peak and 1.46 PFlop/s Linpack, about one half of Mistral). Thus, the Declaration calls for building two computers that are *250-times (!)* more powerful than Mistral within 6 years. Even though Bull is working on new and larger projects, this is a really challenging goal for Bull (and Europe).

Concerning the maximal goal, the only EU company in processor design has been ARM, based in Cambridge, UK. However, ARM is now brexiting, being, moreover, sold to Japanese group SoftBank ... (Hermann Hauser, one of the founders of Acorn Computers, Ltd., the principal root of ARM, said about the deal: "This is a sad day for ARM and a sad day for technology in the UK" - my comment: and for Europe as well). There is no other company in Europe having actually experience with the top level processor design.

Full-EU processor making is even less obvious. The recent state-of-the-art, the 12-16 nm process, is mastered just by Intel, TSMC, and Samsung, and the U.S. DOE "rebaseline" report suggests that even 10 nm might not be enough for a straightforward path to exascale.

The most advanced European chip maker is the Franco-Italian ST Microelectronics (even though it is registered in Netherlands with headquarters in Geneva), who mastered making 12" silicon wafers and 32 nm chips. However, the FinFET technology used for 22 - 12 nm process is quite different from 28/32 nm, and it does not seem likely that chips based on 32 nm process could be used to build an exascale computer.

Not all EU initiatives of the past finished by full success, (e.g. [1]), and the maximal goal does not seem to be sufficiently supported by European industrial environment.

This is, by the way, reflected by the panel discussion of leading European HPC authorities during the March 23 Digital Day, youtubed at
https://www.youtube.com/watch?v=2y5LWCZLluc&list=PLyMUk47rPuqozitX6N52khWAgEFqkgZ8n
(video length 1:58:11)
and reported, e.g., in
https://www.top500.org/news/europe-sets-exascale-supercomputing-goal-amid-divisions-on-how-to-achieve-it/.

As two extreme opinions, I can quote:

Mateo Valeto, director of Barcelona Computing Center, said: "If we don't develop hardware, we will always be in the second division" (remark: he didn't point out who in Europe will develop the exascale hardware).

On the other hand, Thomas Schulthess, the director of Swiss National Supercomputing Center (actually hosting the 3rd most powerfull supercomputer Piz Daint) thinks that the best path to maintain scientific leadership is "rather than focusing on building an exascale system, we should focus on the scientific questions that we want to solve, and use technology to solve that problem" (i.e., why to focus on something we are not able to make).

Supercomputer performance: peak, Linpack, and HPCG

The easiest way of assessment of the computing power of a supercomputer is to add computing performances of its processor chips. This measure is called the (theoretical) peak power; the plot of peak powers of Top10 supercomputers (but ordered by their Linpack performance) is as follows

[1]Lisbon Strategic Goal 2000: "The (European) Union has today (March 23, 2000) set itself a new strategic goal for the next decade: to become the most competitive and dynamic knowledge-based economy in the world" (see www.europarl.europa.eu/summits/lis1_en.htm, Art. I.5)

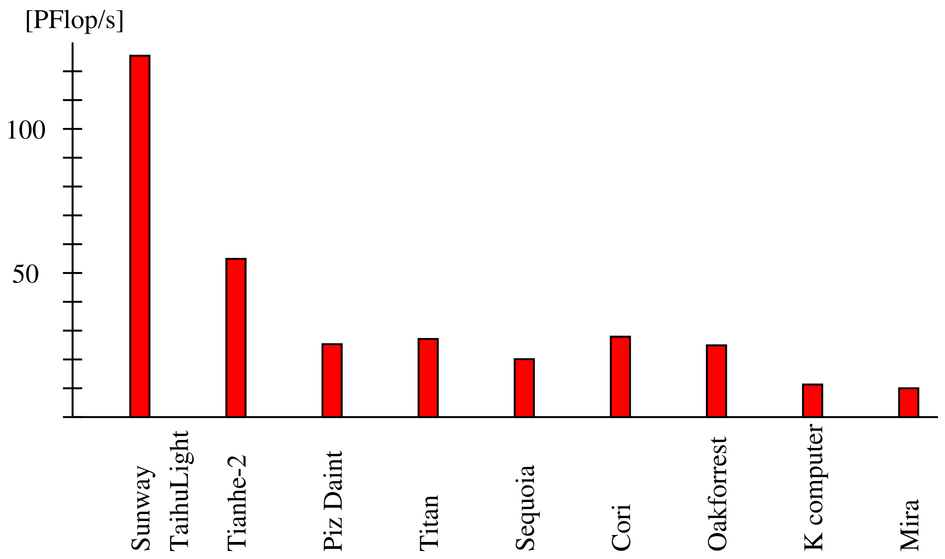

Figure 1: Peak computing power, Top10

The peak power does not reflect performance of a computer in practical computations, where communication and data exchange among processors play an important role. More realistic assessment is obtained if the performance is measured when a supercomputer solves a practical problem that involves both computation and communication. Since large problems solved on supercomputers are very often based on Linear Algebra, especially large systems of linear equations, and since Gaussian elimination is the fundamental method of solving linear systems, Jack Dongarra et al. developped a benchmark that is a part of the Linpack package, and represents an implementation of the Gaussian elimination. Since 1979, when the first Linpack manual had been published, the benchmark became the most frequent tool for ranking supercomputers and measuring their computing power.

The following plot repeats the peak performances of Top10 supercomputers (red) with their Linpack performances (green).

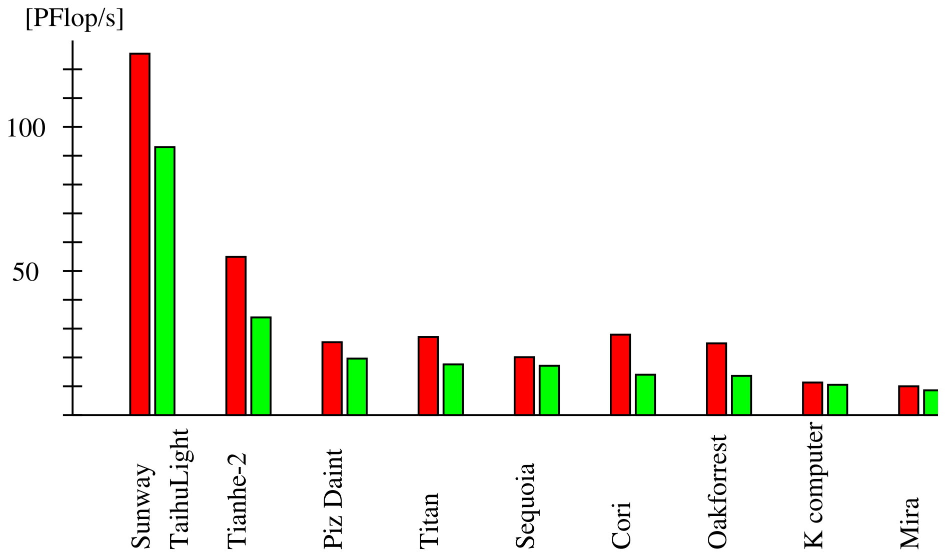

Figure 2: Peak and Linpack computing power, Top10

The Linpack benchmark is popular not only because it is a practical problem, but also because Linpack results correlate well with the peak power, being usually 50-75 % of the peak. One can even say that Linpack is a way of selling the peak power values. The Linpack is the basis for the most popular supercomputer ranking, Top500, and it is assumed that a true exascale supercomputer will be the one with *Linpack* performance of 1 EFlop/s.

However, there has been and there is growing feeling among many researchers that the Gaussian elimination (Linpack) is not any more the best measure of the computing performance, see Appendix for a more detailed discussion. This is why, in 2014, Jack Dongarra, the principal co-author of Linpack, and his collegues M. Heroux and P. Luszczek, came with an alternative benchmark HPCG (High Performance Conjugate Gradient) that is a standardized implementation of the Conjugate Gradient Method, which (either directly or in a form of its generalizations, e.g., different Krylov subspace methods) is an iterative method of solving sparse systems of linear equations used for simulation of physical processes described by PDE's, e.g., in the fluid mechanics (meteorology and weather forecasting, aerodynamic studies and virtual wind tunnels, hydrodynamics and oceanology, etc.), mechanical strength and deformation (simulated car crashes), heat conduction and equilibria, combustion and explosion mechanics, and many other applications.

The following plot shows the performance of the Top10 supercomputers (June 2017) in the HPCG benchmark (blue), compared with the peak (red) and Linpack (green) performance, all data in PFlop/s.

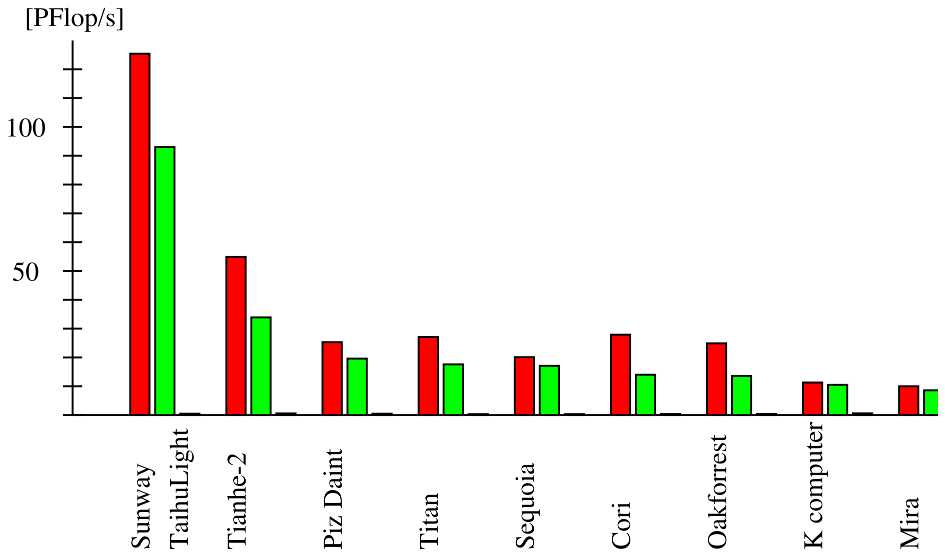

Figure 3: Peak, Linpack, and HPCG computing power, Top10

Well, perhaps you do not see the blue columns - HPCG results are so low that we are hardly able to see the corresponding "columns". The next plot shows the HPCG results (June 2017) magnified about 100-times in the vertical direction:

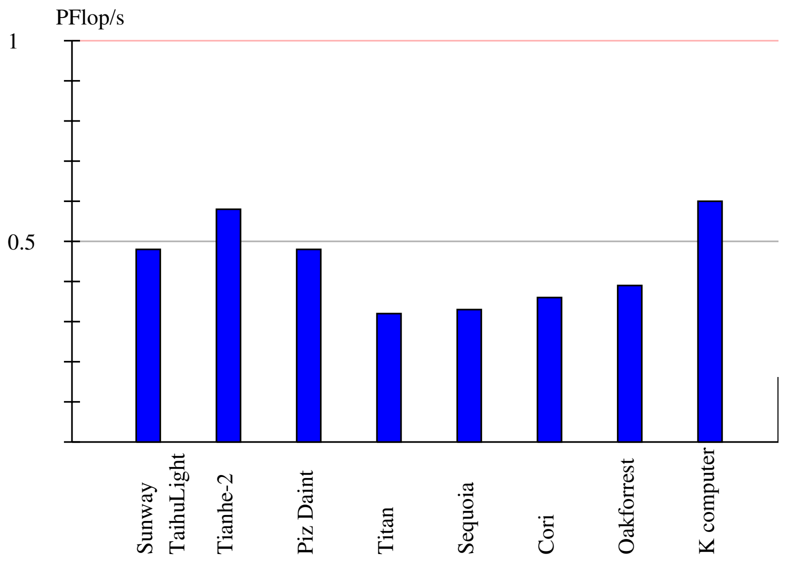

Figure 4: HPCG computing power, Top10

The following observations are immediate:

- Compared to the peak and Linpack performance, the HPCG results are worse than poor - no one of the systems with the peak performance in the range 10-125 PFlop/s is able to run at more than about 0.6 PFlop/s (see the red horizontal line representing one petaflops barrier). We dream about exascale, but the practical computation is not yet in the petascale era.
- HPCG results are not in correlation with the peak/Linpack performance - they are comparable for systems having the peak power different by one order of magnitude

The following plot is even more interesting: it shows the HPCG performance expressed as a fraction of the peak power (i.e., the practical "efficiency" of a supercomputer):

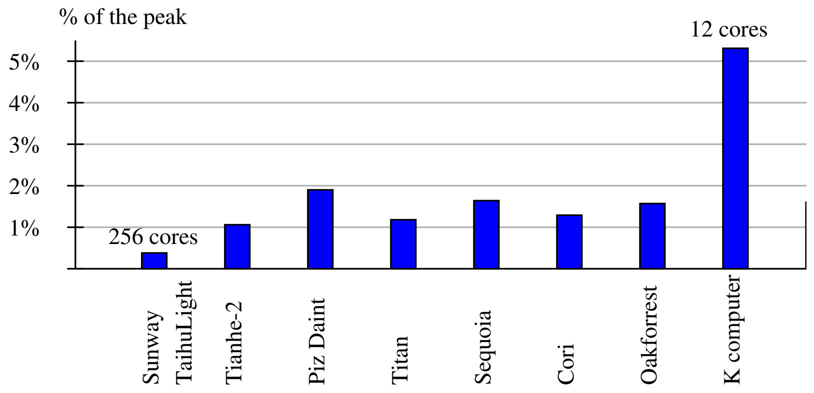

Figure 5: HPCG efficiency, Top10

Again, the following observations are immediate:

- The efficiency graph presents even better view of the poor HPCG performance of the Top10 supercomputers: with notable exception of the Japanese K, they are using less than 2 % of their peak power.
- it seems that the HPCG efficiency is inversely proportional to the number of cores - among Top10, by far the best efficiency has the old Japanese K with only 8 cores per socket, while Sunway TaihuLight with 260-core processor Shen Wei is by far the worst; see also the next figure.

When looking at the data of supercomputers that appear both in the Top500 (www.top500.org/lists/2017/06/) and the HPCG lists (www.hpcg-benchmark.org/custom/index.html?lid=155&slid=291) (altogether 63 computers), we can find 2 systems with the efficiency over 5.0 %, 1 over 4.0 %, 6 in the range 3.0-3.99, and 10 between 2.0 and 2.99 %, and 7 below 1%, while the remaining 37 systems are between 1 and 2 %.

The next plot shows the HPCG efficiency related to the number of cores per processor socket for the intersection of the Top500 and the HPCG lists.

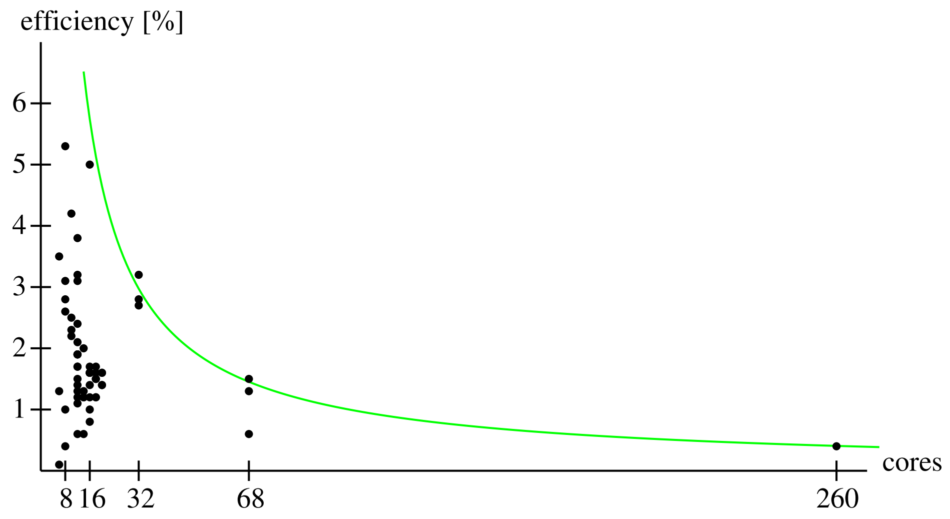

Figure 6: HPCG active cores, Top10

As suggested by the figure, it seems that there is an upper bound (the green curve) to the HPCG efficiency; most of the computers are well *below* the green curve (mostly small systems with simpler processors having 6-18 cores), but no supercomputer is substantially above the curve. Taking into account small number of supercomputers that are near to the green curve, we have to be careful when drawing conclusion, but the conjecture that the efficiency decreases with the number of cores per socket seems to be well supported. Another support for the sonjecture follows from more detailed analysis of the reasons of poor HPCG efficiency, see the next section of this report.

But the most striking fact that follows from the HPCH results is the following: the product of the HPCG efficiency and the number of cores can be understood as the average number of cores per socket that are doing useful work when the computer runs the HPCG benchmark. The following table lists all entries of the intersection of the Top500 and HPCG lists that have the average number of active cores higher than 0.4 when running HPCG benchmark:

T500	CG	Computer	Peak	Linpack	HPCG	Eff	C	Active
1	3	Sunway TaihuLight	125.4	93.015	0.4808	0.4	260	1.04
34	15	SORA-MA, Fujitsu	3.5	3.157	0.1102	3.2	32	1.024
7	5	Oakforest-PACS, Fujitsu	24.9	13.555	0.3855	1.5	68	1.02
55	24	Plasma Simulator, Fujitsu	2.6	2.376	0.0732	2.8	32	0.896
6	6	Cori, Cray	27.9	13.832	0.3554	1.3	68	0.884
39	19	ITC Nagoya, Fujitsu	3.2	2.910	0.0865	2.7	32	0.864
123	34	Oakleaf-FX, Fujitsu	1.1	1.043	0.0565	5.0	16	0.8
80	27	JURECA, T-Platforms	1.7	1.425	0.0683	3.8	12	0.456
8	1	K computer, Fujitsu	11.3	10.510	0.6027	5.3	8	0.424
90	32	iDataPlex DX360M4, IBM	1.5	1.283	0.0615	4.2	10	0.42
14	26	Marconi, Lenovo	10.8	6.223	0.0686	0.6	68	0.408

where
- T500 is the Top500 rank
- CG is the HPCG rank
- Peak, Linpack and HPCG are the corresponding performances in PFlop/s,
- Eff, the HPCG efficiency, is the ratio of the HPCG and the Peak performances,
- C is the number of cores per socket, and
- Active is the HPCG average number of the active cores per socket

The last table makes us even to think about practical use of multicore and many-core architectures.

Extrapolating the data, we can expect that the future exascale supercomputer will have (unless a quite new architectonical approach is adopted) the HPCG efficiency of 0.5-1 % or, in other words, the effective HPCG computing power 5-10 PFlop/s.

Why HPCG results are so bad?

There are three key components of the Conjugate Gradient algorithm: the dot (scalar) product of two vectors, multiplying of a large sparse matrix and a vector, and preconditioning (which is essentially solving a system of linear equations with a special sparse triangular matrix by back substitution). All three problems have very low arithmetic intensity (the average number of arithmetic operations that can be performed for one byte of data moved from the memory to a processor), also called the flop:byte ratio.

The dot product ($\sum_{i=0}^{n-1} x_i y_i$) has arithmetic intensity 0.125: moving x_i and y_i into a processor for some i allow us to perform one multiplication and one addition (usually combined into one FMA - fused multiply add - operation). When working with double precision floating point numbers, moving x_i and y_i operates with 2×8 bytes, and $2/16 = 0.125$.

When performing a matrix-vector product (with DP elements), then even if we succeed to store the vector in a cache, each 8 byte element of the matrix moved to the processor allows us to perform just two arithmetic operations: multiplying the matrix element by certain vector element, and add the product to an accumulator. Two flops per at least 8 bytes give the arithmetic intensity at most 0.25.

The analysis of the preconditioning is more complex, but the results are similar.

Let us now check how many arithmetic operations could be performed when the top level processors are running HPCG benchmark. The first column of the next table lists three of the top processors: Intel Xeon Phi v200 (“Knights Landing”), Nvidia Tesla V100 with the Volta GV100 GPU, and the Chinese processor Shwn Wei 26010 of Sunway TaihuLight supercomputer. The Xeon Phi processor has two memory systems: it has the on package “near” MCDRAM memory of the capacity 16 GB and the memory bandwidth about 480 GB/s, and it can be connected to at most 384 GB of DDR4 DRAM by memory channels of the bandwidth about 120 GB/s. The Tesla V100 accelerator comes with 16 GB of fast HBM2 DRAM with the memory bandwidts up to 900 GB/s, the information about possible high volume lower speed DDR DRAM memory is not available. The Shen Wei has just “slow” memory channel of the bandwidth about 120 GB/s.

Assuming that we are using the processors to solve a problem of the arithmetic intensity 0.25 (i.e., 1 Flop per 4 bytes, like a matrix-vector product), the third column shows how many Flop’s can be executed, taking into account the product of the arithmetic intensity of the problem and the memory bandwidth of the processor. The fourth colum is the peak computing power of the corresponding processor. It is clear that, when running HPCG, the memory interface brings only a small fraction of the data flow that would be needed to keep all cores of the processor busy all the time. The last column, the upper bound to the HPCG efficiency, is the ratio of the previous two columns.

Processor	Memory bandwidth [GB/s]	Enough data for [GFlop/s]	Computing power (peak) [GFlop/s]	Efficiency
Xeon Phi v200 (fast memory)	600	150	3000	5 %
Xeon Phi v200 (DDR4 DRAM)	120	30	3000	1 %
Tesla V100	900	225	7800	2.9 %
Shen Wei 26010	120	30	3000	1 %

The table fully explains the poor behavior of the recent supercomputers running the HPCG benchmark. It is important to realize that high bandwidth memories like MCDRAM of “Knights Landing” and HBM2 of Tesla V100 are not too big, in both cases 16 GB (and perhaps 32 GB in the future), while typical supercomputer nodes have often well over 100 GB of memory per processor. Since both HPCG rules and practical consideration forces us usually using (almost) all available memory, the average memory bandwidth in such a case would be substantially lower than 600/900 GB/s of the table, resulting in the efficiency around 1 %.

There is another possible source of low usage of the computing power of processors, namely the communication among processor nodes that also might cause idle states of processing units. However, at least in the case of simpler problems like HPCG, the interconnect bottleneck is not too severe and the memory bandwidth is the principial source of the HPCG inefficiency. See, e.g., a simplified analysis in Luděk Kučera: On architecture for the future *petascale* computing, to appear in the proceedings of ParCo'17 conference, see also http://kam.mff.cuni.cz/~ludek/Parco.pdf.

How frequent is HPCG-like behavior

Supercomputer behavior in the two benchmarks described above could hardly be more different: the Linpack performance is close to and well correlated with the peak power[2], while HPCG performance is a very small fraction of the peak only a does not seem to be correlated with the peak at all. The principal question now is:

How frequent are programs that use supercomputer computing power as inefficiently as the HPCG benchmark?, or, perhaps better,

How much supercomputer time is spent by running programs that behave as inefficiently as the HPCG benchmark?

In other words, is the HPCG benchmark just a rare particular case that does not reflect the typical behavior of practical supercomputer programs, or (as intended by Dongarra et al.) it represents the usual behavior of most (or at least large part) of supercomputer programs?

It follows from the previous section that, for supercomputers with the recent top level processors, low arithmetic intensity immediately implies very inefficient use of the computing power.

The arithmetic intensity is precisely known just for certain very simple problems, e.g., 0.125 for a dot (scalar) product of vectors, 0.25 for a matrix-vector product, but results that precisely determine arithmetic intensity of more complex problems are very rare.

However, it is my understanding that all programs, approximating PDE's describing physical processes in 3D by methods of Linear Algebra, are base on matrices of incidence of grids that are used to discretize the physical space, and such grids have very local structure, i.e., any grid node has only a small number of neighbors, are based on manipulation with sparse metrices that has typically very low arithmetic intensity.

One support for this opinion can be found at the folowing figure, which has appeared in many papers and reports, and which I traced to a report of Lawrence Livermore Lab about 10 years ago:

[2]The average of the ratio Linpack/Peak is 72% for Top10 and 66 % for Top500

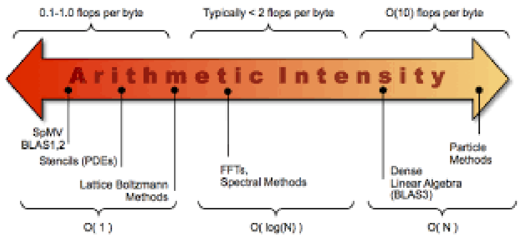

Figure 7: Arithmetic intensity of problems

The figure explicitly shows that simulations based on solving PDE's have typically low arithmetic intensity (the figure does not indicate more precise values). My ParCo article (http://kam.mff.cuni.cz/~ludek/Parco.pdf) also references several of many papers, complaining about low arithmetic intensity of problems in sparse Linear Algebra.

Currently, I am cooperating with Department of Meteorology MFF UK trying to assess the arithmetic intensity of WRF model (Weather Research and Forecasting). We conjecture that the computational kernel of the model has low arithmetic intensity as well - and meteorology oriented programs are very frequent users of the supercomputer time.

The impact of my opinion is, of course, negligible. However, during the ParCo conference mentioned above, I met with Jack Dongarra; his opinion has, I am sure, very heavy impact. I asked him explicitly about his estimation of how much supercomputer time is spent, say, in the U.S., running problems that are as inefficient as the HPCG benchmark. And he answered:
"*I would say most of the time.*"
And he added:
"*If someone could do better ...*".

In his view, thousands of hours of supercomputing time are spent by routine simulation programs in car and aerospace industry, defense research and other fields and behave in a way very similar to HPCG benchmark.

Exascale equivalent supercomputer

Motto: *Technology always develops from the primitive, via the complicated, to the simple.*
Attributed to Antoine de Saint-Exupéry in the English Web, even though I found the original citation neither in the French Web not in Saint-Exupéry's texts

The facts introduced in the previous questions induce a natural question: why to follow the way to a supercomputer with 1000 PFlop/s peak, but the practical performance of about 5-10 PFlop/s (i.e., 0.5-1 % of the peak)? Wouldn't it be better to build directly a cheaper and simpler machine able to run HPCG and other "most-of-the-time" programs at the same speed as the potential exascale supercomputer, no matter what is its theoretical computing power?

The vision of the present paper is a simple and relatively cheap 10-15 PFlop/s peak computer, built about the same time as the first (Chinese or U.S.) true exascale computer sometime in 2020-2022 which would prove to be *computationally equivalent* (=as fast as) the the exascale machine when running not only HPCG benchmark, but also a large range of practical programs in meteorology, aerodynamic, mechanic and explosion simulation, and similar fields. I will call such a computer *exascale equivalent*[3].

A big advantage of the notion of exascale equivalent supercomputer is that, as I will try to argue in Appendix B, such a system could be based on full-European 32 nm processors (thus fulfilling the second goal of the March 23 (2017) Rome declaration), while the latest news about the projected U.S. Aurora supercomputer might suggest that even 10 nm process would not give processors powerfull enough for Exascale (and it is generally assumed that the present 14 nm is not sufficient).

Appendix B describes an architectonical outline of a potential 32 nm processor with a relatively low peak power 400 GFlop/s that, however, cold be used up to 100 % even when running a program with arithmetic intensity as low as 0.25 (e.g., sparse matrix operations)[4]. About 1500 such processors would be sufficient to win June 2017 HPCG list, and about 12,500-25,000 such "slow" but cheap and simple EU-made chips would give an exascale equivalent supercomputer[5]. Nevertheless, I have to recall that "simple" is used with respect to the silicon logic of the processor, while the memory interface of such processors need to be enhanced in a way that is on the edge or slightly beyond the present state of the art.

Let me finish by quoting Mr. Roberto Saigri, Eurotech CEO who, during the panel discussion at Digital Day in Rome on On March 23, 2017, compared EuroHPC to the US Apollo program:

"They had a dream and we need another dream for high perfomance computing".

During 45 years since Apollo 17, no other men flew to Moon, but the near space is full of meteorological, communication, and other types of satelites and permanently populated by at least 2-3 men or women.

The true Exascale is an Apollo-like dream,
our dream is making anonymous low-flying exascale-equivalent working horses.

[3]Let me recall that an exascale equivalent computer is not meant as a *universal* computer to run all kinds of program. It is specialized to execute codes that, I (and Jack Dongarra) believe, use most of the global supercomputer time, but there are problems and programs of high arithmetic complexity (e.g., Linpack), for which the notion of exascale equivalence is not appropriate. It is only a question whether such arithmetically complex problems are sufficiently frequent to justify building an extremely expensive true exascale computer.

[4]Note that Intel Knights Landing has effective computing power about 150 GFlop/s for this arithmetic intensity, the newest Nvidia Volta GPU offers not more than 225 GFlop/s.

[5]Note that about 110,000 of the newest 12 nm Nvidia Volta GPU's would be needed to get 1 EFlop/s peak.

Appendix A
Direct and indirect solving of sparse systems of linear equations

In computer simulations of physical processes, partial differential equations describing a given physical process are solved using discrete approximations in essentially the following way: a discrete grid is embedded into the physical space, and we are interested in values of certain physical quantity in the grid nodes. Partial derivatives in the grid nodes are approximated by linear functionals depending on the value in a given node and its near neighbors. In this way we get a system of linear equations, where non-zero elements of the rows of the matrix of the system correspond to neighbors of the given node of the grid. It is clear that the number of neighbors is usually small (quite often 7, 19, or 27 for 3D grids due to using of 7-, 19-, or 27-point stencil).

Imagine that we are solving a system of linear equations that would fit into 1 PB of memory of a supercomputer (a typical value for the recent Top10 systems). Even in the case of 27 non-zeroes per a row of the system matrix, one row represent 27×8 bytes of data - I will suppose that one row occupies (at most) 256 bytes. Thus, the memory of the supercomputer makes it possible to store a sparse matrix with about 4×10^{12} rows.

If such a system is solved by Gaussian elimination, we eventually reduce the system matrix to a *triangular* matrix. Unfortunately, even if we start with a very sparse matrix, the resulting matrix of the Gaussian elimination is usually very dense, and such a matrix would have $0.5 \times (4 \times 10^{12})^2 = 8 \times 10^{24}$ non-zero items - too much for recent supercomputers. The largest matrix that could be solved by a direct method would have the order of about 2.8×10^6.

If $a > 0$, the solution of an equation $ax = b$ can be found as the minimum of the function $f(x) = \frac{1}{2}ax^2 - bx$, because $f' = ax - b$, and the minimum of f occurs if $f' = 0$.

Similarly, if $\mathbf{A}$ is positive definite matrix, then the solution of $\mathbf{Ax} = \mathbf{b}$ is the vector $\mathbf{x}$, which minimizes the functional $\frac{1}{2}\mathbf{x^T A x} - \mathbf{Ax}$. The minimum of the functional can be found with almost no extra memory, using, e.g., Steepest Descent Method or, preferably, Conjugate Gradient Method (CGM) or another Krylov subspace method. This is why CGM and similar methods are often used in HPC simulations.

Appendix B
Example: A case study of a 196 GFlop/s lowain processor

The term *lowain* processor will be used to refer to a processor designed so that it can fully use its peak computing power even when running a problem of *low arithmetic intensity.* As it will be clear from the following text, the present state of art allow us to build a lowain processor with the computing power of several hundreds of GFlop/s. As an example of feasibility of such a design, the following architecture is presented.

Let us study a potential lowain processor with the target arithmetic intensity 0.25 and the memory bandwidth 1.6 TB/s. The value of the bandwidth is substantially beyond the state of the art; the highest memory bandwidth has presently Nvidia Tesla V100 - about 900 GB/s. However, it is conceivable that such a chip could be made in the near future. The values give the effective computing power for the given arithmetic intensity equal to 400 GFlop/s.

Let us assume that the instruction flow would be pipelined so that one FMA (fused multiply add) operation is performed per cycle and core. Since FMA is counted as 2 arithmetic operations, one core performs 4 GFlop/s. The last value means that we would need 100 cores to get 400 GFlop/s.

Thus, I assume that the processor has a matrix of $10 \times 10 = 100$ scalar cores, which gives the computing power (at 2 GHz clock) equal to $2 \times 2 \times 100 =$ 400 GFlop/s. It will be convenient if the cores work as vector units for lower precisions (two single precision or four half precision operations like FMA per cycle), but this feature is out of the scope of our problem.

Nvidia 12 nm Volta GV100 GPU has about 800 mm^2 die and more than 2500 FP64 cores, so one FP64 core occupies about 0.32 mm^2. Let us assume that the lowain processor is fabricated by 32 nm process that is mastered in Europe (ST Microelectronics). Extrapolating one Volta core area to 32 nm would give $0.32 \times (32/12)^2 = 2.3$ mm^2 = (1.5 mm)^2.

It is therefore feasible to assume that one core would occupy the area $1.5 \times 1.5 = 2.25$ mm^2. The 10×10 matrix of cores would occupy 15 mm $\times$ 15 mm $\approx$ 225 mm^2. I will suppose a processor die of the size 20 mm $\times$ 20 mm = 324 mm^2, about 56% occupied by the core matrix, the remaining 44 % given to memory buffers and controllers.

Recall that the necessary memory bandwidth needed for 400 GFlop/s at the arithmetic intensity 0.25 is 1600 GB/s = 12800 Gbit/s.

One possibility of getting high memory bandwidth is to have processor cores on the same die as the memory. However, one processor has usually 10's or more frequently 100's of GB of memory, while standard DRAM chips have at most 2 GB, and, moreover, the manufacture process is different for logic silicon and for DRAMs.

We will investigate different bandwidth of data lines, connecting memory chip(s) with the corresponding processors: 1.6 Gbit/s (slower DRAM data line), 2.4 Gbit/s (faster DRAM data line), 3.2 Gbit/s (very fast DRAM data line), 32 Gbit/s fast serial links (see, e.g., Virtex UltraScale+ XCVU13P offers 128 GTY transceivers 32.75 Gbit/s, page 11 of www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf):

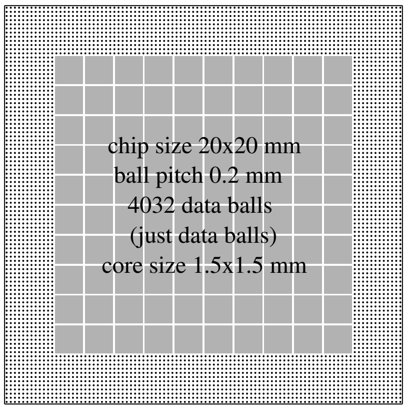

Figure 8: Lowain processor ball array

Data Line Bandwidth [Gbit/s]	Number of Needed Data Lines
1.6	8000
2.4	5400
3.2	4000
32	400

Taking into account the Xilinx chip, one could believe that obtaining the necessary memory bandwidth by building 400 fast serial links on a chip is technically feasible. However, necessary transceivers (both within the chip and outside on the board) would complicate the design, and hence “slow” data lines connecting the chip directly to DRAMs might be an easier solution.

One possibility is that the bottom part of the processor holds a ball array of either $64 \times 64 = 4096$ balls (for 3.2 GHz) or $74 \times 74 = 5476$ balls (for 2.4 GHz) or $90 \times 90 = 8100$ balls (for 1.6 GHz). Assuming the footpring of the die 20 mm $\times$ 20 mm, this would correspond to the ball pitch 0.3 mm (for 3.2 GHz) or slightly more than 0.2 mm (for 1.6 GHz), which is essentially within state of the art: e.g., ST Microelectronics makes chips with a (small) ball array with the pitch 0.2 mm; the problem for the lowain processor would be the number of balls. The table summarizes the values:

Bandwidth [Gbit/s]	Lines	Ball Array	Pitch
1.6	8000	90×90	0.22 mm
2.4	5400	74×74	0.27 mm
3.2	4000	64×64	0.31 mm

Another possibility is that that the ball array is just below the circumference of the chip, see Fig. 8 for 3.2 GHz lines (where gray squares represent cores):

Finally, let us consider a solution inspired by Intel’s EMIB (Embedded Memory Interface Bridge), where the die is placed on an interposer, and data lines are located on the boundary of the die square:

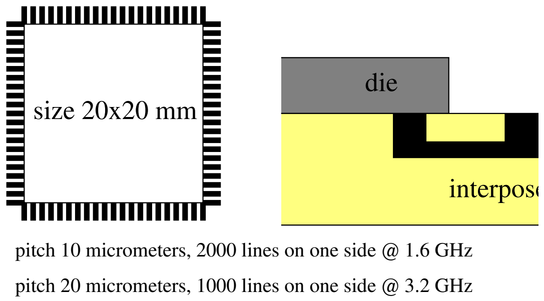
size 20x20 mm
die
interpos
pitch 10 micrometers, 2000 lines on one side @ 1.6 GHz
pitch 20 micrometers, 1000 lines on one side @ 3.2 GHz

Figure 9: Lowain processor memory bridge

The boundary of a 20 mm $\times$ 20 mm die has the length 80 mm, and therefore we would need the pitch 10 μm for 8000 data lines at 1.6 GHz, or the pitch 15 μm for 5400 data lines at 3.2 GHz, or the pitch 20 μm for 4000 data lines at 3.2 GHz.

The Intel patent document U.S. 9,240,377 explicitly mentiones such data lines with the pitch 14 μm running at 1.6 GHz, which gives hopes that the desired memory bridge could be constructed, see the table

Bandwidth [Gbit/s]	Lines	Pitch
1.6	8000	10 μm
2.4	5400	15 μm
3.2	4000	20 μm
1.6	U.S. 9240377	14 μm

2 GB 3.2 GHz DDR4 DRAM chips can now be taken as the top of the state of the art. If 4000 data lines of the 3.2 GHz memory interface of the lowain chip is connected with 4000/16 = 250 DDR4 DRAM 2GB chips of the x16 architecture, we will get a processing node with the computing power 400 GFlop/s (for arithmetic intensity ≥ 0.25) and DRAM capacity 500 GB. About 2500 such chips would be sufficient to overcome 1 PFlop/s HPCG barrier, and 12500-25000 chips to build an EXEQ (exascale equivalent) supercomputer.

As already pointed above, about one half of the lowain processor die would be occupied by memory controller and buffers. The reason is simple: very large number of the memory-processor interface data lines and memory chips around a processor would result in high and non-uniform length of the data lines which in turn leads to long and non-uniform latencies.

The present mechanism of caching data and pre-fetching them to a cache from external memory chips is sophisticated, but rigid and it is often quite difficult (or impossible) to adapt it so that data are delivered to arithmetic units "just in time" for processing.

I am sure that a smooth data flow could be guaranteed only by a user programmable memory controller that controls the memory-processor data flow. The controller runs under supervision of a user program (co-algorithm) that cooperates with the main code that is executed in the processor to obtain intelligent prefetch of the data that are based on the user's understanding of the main algorithm.

The present solution, where user can only give prefetch hints to the compiler, hoping that the optimized compiler will choose an appropriate method, seems to be too rigid to guarantee the best prefetch strategy that is needed for efficient use of the memory-processor interface[6].

Memory controller design is a subject of forthcomming research and more detailed explanation would be outside of the scope of the present paper. I am sure that that the memory controller architecture and algorithms represent the critical part of the lowain processor design that would require most of the research activity.[7]

It is clear that many variations could be designes by extending the parameters and architecture of the lowain chip memory interface to the limits of the present technology.

Finally, let us summarize the parameters of one possible embodiment of a 400 GFlop/s lowain processor (the die area about one half of that of Nvidia Volta GV100 and about 57 % of Intel KNL, 32 nm process instead of 12-14 nm, the logic occupying just about 56 % of the chip area):

CMOS process	32 nm
Die size	20 mm × 20 mm
Die area occupied by the cores	15 mm × 15 mm
Die area occupied by the cores	56 %
Number of scalar cores	100
Processor clock frequency	2 GHz
Computing power	400 GFlop/s
Computing power for arithmetic intensity 0.25	400 GFlop/s
Memory bandwidth	1600 GB/s = 12800 Gbit/s
Memory data line frequency	3.2 GHz
Number of data lines	4000
Variant 1: ball array pitch (see Fig. 8)	0.2 mm
Variant 2: memory bridge pitch (see Fig. 9)	20 μm
Chips to win June'17 HPCG	1500
Chips for 1 PFlop/s HPCG	2500
Chips for exascale equivalence	12500-25000

[6]Supported by my experience with programming Intel KNL's at Cori supercomputer

[7]I have submitted a GAČR research proposal directed to memory controller architectures and prefetch co-algorithm design.