How Amdahl’s low restricts supercomputer applications and building ever bigger supercomputers

János Végh
University of Miskolc, Hungary
Department of Mechanical Engineering and Informatics
3515 Miskolc-University Town, Hungary
J.Vegh@uni-miskolc.hu

ABSTRACT
This paper reinterprets Amdahl’s law in terms of execution time and applies this simple model to supercomputing. The systematic discussion results in practical formulas enabling to calculate expected running time using large number of processors from experimental runs using low number of processors, delivers a quantitative measure of computational efficiency of supercomputing applications. Through separating non-parallelizable contribution to fractions according to their origin, Amdahl’s law enables to derive a timeline for supercomputers (quite similar to Moore’s law) and describes why Amdahl’s law limits the size of supercomputers. The paper validates that Amdahl’s 50-years old model (with slight extension) correctly describes the performance limitations of the present supercomputers. Using some simple and reasonable assumptions, the absolute performance bound of supercomputers is concluded, furthermore that serious enhancements are still necessary to achieve the exaFLOPS dream value.

1. INTRODUCTION
Supercomputers do have a quarter of century history for now, see [Top500 (2017)]. The number of processors raised exponentially from the initial just-a-few processors [Dongarra (1992)] to several millions [Fu et al (2016)] and increased their computational performance (as well as electric power consumption) even more impressively. Supercomputers provide their \( R_{\text{max}} \) and \( R_{\text{peak}} \) parameters when running a benchmark using all their processors, but when needing not all available processors of a supercomputer and running programs other than benchmarks, they give not much hints for the effective computational performance for the case in question. Amdahl’s law about parallelly working systems introduces serious limitations on the joint performance, so it is worth to scrutinize how Amdahl’s law can assist in finding out execution time of a program as well as how it affected the operational characteristics of supercomputers. As discussed in [Denning and Lewis (2017)], "Moore’s Law is one small component in an exponentially growing planetary computing ecosystem". Using some simple assumptions, they prove that from more closely Moore’s law is described by an S-curve rather than an infinite exponential. Indeed, from the many components involved one can conclude different saturation points. The fact that no more transistors can increase functionality of the present processors in a reasonable way, caused a saturation about a decade ago (and started the age of multi-cores, because density of transistors is not yet saturated). Reaching the lithographic limits forced only introducing manufacturing tricks, but reaching the size of atoms will surely cause a saturation, so the industry is about to prepare for the post-Moore era (2017).

Supercomputers with ever bigger computational power are planned. As shown below, Amdahl’s law represents another "small component in an exponentially growing planetary computing ecosystem". Shall we also prepare for a post-Amdahl era?

2. AMDAHL’S CLASSIC ANALYSIS
The most commonly known and cited limitation of the parallelization speedup (Amdahl (1967), the so called Amdahl’s law) is based on considering the fact that some parts \( (P) \) of a code can be parallelized, some \( (S) \) must remain sequential. It was early given by Amdahl, although the formula itself was constructed later by his successors. Amdahl only wanted to draw the attention to the fact, that when putting together several single processors, and using Single-Processor Approach (SPA), the available speed gain due to using large-scale computing capabilities has a theoretical upper bound. He also mentioned that data housekeeping (non-payload calculations) causes some overhead, and that the nature of that overhead appears to be sequential.

The classic interpretation implies three essential restrictions, but those restrictions are rarely mentioned (an exception: [Karp and Flatt (1990)]) in the textbooks on parallelization:

- the parallelized parts are of equal length in terms of execution time
- the housekeeping (controlling the parallelization, passing parameters, waiting for termination, exchanging messages, etc.) has no costs in terms of execution time
- the number of parallelizable chunks coincides with the number of available computing resources

Essentially, this is why Amdahl’s law represents a theoretical upper limit for parallelization gain. In Fig the original...
process in the single-processor system comprises the sequential only parts $S_i$, and the parallelizable parts $P_i$. One can also see that the control components $C_i$ (not shown on the left side of the figure) are of the same nature as $S_i$, the non-parallelizable components. This also means that even in the idealistic case when $S_i$ are negligible, $C_i$ will create a bound for parallelization, independently of their origin.

### 2.1 Amdahl’s case under realistic conditions

The realistic case (shown in the right side of Fig1) is, however, that the parallelized parts are not of equal length (even if they contain exactly the same instructions), the hardware operation in modern processors may execute them in considerably different times; for examples see [Vegh et al (2014)] and [Hennessy (2007)], and references cited therein; the operation of hardware accelerators inside a core or the network operation between processors, etc. One can also see that the time required to control parallelization is not negligible and varying. This represents another performance bound.

The figure also calls the attention to the fact that the static correspondence between program chunks and processing units can be very inefficient: all assigned processing units must wait the delayed unit and also some capacity is lost if the number of computing resources exceeds the number of the parallelized chunks. It is much worse, if the number of the processing units is larger than that of parallelized threads: the processor must organize several "rounds" for the remaining threads, with all disadvantages of the duty of synchronization [Yavits (2014)] [David et al (2013)]. Also, the longer code chunks $P_i$ are, the higher is the chance to waste computing capacity of the processing units which already finished their task. Note that here the programmer (the person or the compiler) has to organize the job, in the spirit of Single Processor Approach (SPA): at some point the single processor splits the execution, transmits the necessary parameters to some other processing units, starts their processing, then waits the termination of the started processings. The real-life programs show sequential-parallel behavior, with variable degree of parallelization [Yavits (2014)] and even the apparently massively parallel algorithms change their behavior during processing [Pingali (2011)]. All these make Amdahl’s original mode non-applicable.

### 2.2 Factors affecting parallelism

Usually, Amdahl’s law is expressed with the formula

$$S^{-1} = (1 - \alpha) + \alpha/k$$

(1)

where $k$ is the number of parallelized code fragments, $\alpha$ is the ratio of the parallelizable part to the total, $S$ is the measurable speedup. The assumption can be visualized that (assuming many processors) in $\alpha$ fraction of the running time processors are processing data, in $(1-\alpha)$ fraction they are waiting (all but one). That is $\alpha$ describes how much, in average, the processors are utilized. Having those data, the resulting speedup can be estimated.

A general misconception (introduced by followers of Amdahl) is to assume that Amdahl’s law is valid for software only and that $\alpha$ contains something like ratio of numbers of the corresponding instructions. Actually, Amdahl’s law is valid for any partly parallelizable activity (including computer unrelated ones) and $\alpha$ is given as the ratio of the time spent with parallelizable activity to the total time.
For a system under test, where $\alpha$ is not a priori known, one can derive from the measurable speedup $S$ an effective parallelization factor as

$$\alpha_{eff} = \frac{k - S}{S - 1}$$

(2)

Obviously, this is not more than $\alpha$ expressed in terms of $S$ and $k$ from Equ. (1). So, for the classical case, $\alpha = \alpha_{eff}$; which simply means that in ideal case the actually measurable effective parallelization achieves the theoretically possible one. In other words, $\alpha$ describes a system the architecture of which is completely known, while $\alpha_{eff}$ characterizes a system the performance of which is known from experiments. Again in other words, $\alpha$ is the theoretical upper limit, which can hardly be achieved, while $\alpha_{eff}$ is the experimental actual value, which describes the complex architecture and the actual conditions.

$\alpha_{eff}$ can then be used to refer back to Amdahl’s classical assumption even in the realistic case when parallelized chunks have different lengths and the overhead to organize parallelization is not negligible. Speedup $S$ can be measured and $\alpha_{eff}$ can be utilized to characterize the measurement setup and conditions, how much from the theoretically possible maximum parallelization is realized.

Note that in the case of real tasks a kind of Sequential/Parallel Execution Model [Yavits (2014)] shall be applied, which cannot use the simple picture reflected by $\alpha$, but $\alpha_{eff}$ gives a good merit of the degree of parallelization for the duration of the execution of the process on the given hardware configuration, and can be compared to the results of the technology-dependent parametrized formulas. Numerically $(1 - \alpha_{eff})$ equals with the $f$ value, established theoretically by [Karp and Flatt (1990)]. In the case of supercomputer HW, it can measure what fraction of running time spend processors with payload activity.

### 2.3 Applying $\alpha_{eff}$ to Amdahl’s model

With our notations, in the classical Amdahl case on the left side in Fig. 1

$$S = \frac{\sum_i S_i + \sum_i P_i}{\sum_i S_i + \max_i P_i} = 2$$

(3)

and

$$\alpha = \alpha_{eff} = \frac{\sum_i P_i}{\sum_i S_i + \sum_i P_i} = 3/4$$

(4)

Now we can compare the effective parallelization in the two cases shown in Fig. 1. In the realistical case $S = 10/7$, which results in

$$\alpha_{eff} = \frac{3 \times 10/7 - 1}{2 \times 10/7} = 0.45$$

(5)

The overhead and the different duration of the parallelized parts reduced the effective parallelization drastically relative to the theoretically achievable value. Fig 2 gives a feeling on the effect of the computer system behaviour on the effective parallelization. The middle region (marked by balls) is mentioned by Amdahl as typical range of overhead. The asterisk in the figure shows the “working point” corresponding to the values used in Fig 1.

One can see that effective parallelization drops quickly with both increasing overhead and sequential parts of the program. This fact draws the attention to the idea that through decreasing either the control time or the sequential-only fraction of the code (or both), and utilizing the otherwise wasted processing capacity, a serious gain in the effective parallelization can be reached.

### 3. EFFICIENCY OF PARALLELIZATION

The distinguished constituent in Amdahl’s classic analysis is the parallelizable payload fraction $\alpha$, all the rest (including wait time, communication, system contribution and any other non-payload activity) goes into the “sequential-only” fraction according to this extremely simple model.

When using several processors, one of them makes the sequential-only calculation, the others are waiting (use the same amount of time). So, when calculating the speedup, one calculates

$$S = \frac{(1 - \alpha) + \alpha}{(1 - \alpha) + \alpha + \frac{k}{k(1 - \alpha) + \alpha}} = \frac{k}{k(1 - \alpha) + \alpha}$$

(6)

hence the efficiency $E$ (how speedup scales with the number of processors)

$$E = \frac{S}{k} = \frac{1}{k(1 - \alpha) + \alpha}$$

(7)

The importance of producing more efficient parallelization manifests from the very beginnings, see Table 1. It could be observed that supercomputers having an order of magnitude higher number of processors should have an order of magnitude lower value of $(1 - \alpha_{eff})$ to produce reasonable efficiency. Ironically enough, the higher number of cores is accompanied with lower efficiency, but at

$^3$This quantity is almost exclusively used to describe computing performance of multi-processor systems. In the case of supercomputers, $\frac{R_{peak}}{R_{peak}}$ is provided, which is identical with $E$. 

$$E$$

(7)
Comparing supercomputers having different architectures, manufacturers, number of processors, technological age, etc. is not easy at all. In order to make different architectures comparable, with as low distortion as possible, the same benchmark program HP L benchmark software at the beginning of the supercomputer age

| Computer Model | N proc | Efficiency | $1 - \alpha_{eff}$ |
|----------------|--------|------------|-------------------|
| Cray Y-MP C90 | 16     | 0.69       | 2.995E-02         |
| NEC SX-3      | 2      | 0.91       | 9.890E-02         |
| Cray Y-MP/8   | 8      | 0.87       | 2.135E-02         |
| Fujitsu AP 1000 | 512     | 0.29       | 4.791E-03         |
| IBM 3090/600S VF | 6      | 0.94       | 1.277E-02         |
| Intel Delta   | 512    | 0.03       | 6.327E-02         |
| Alliant FX/2800-200 | 14     | 0.79       | 2.045E-02         |
| NCUBE/2       | 1024   | 0.12       | 7.168E-03         |
| Convex C3240  | 4      | 0.95       | 1.754E-02         |
| Parsytec FT-400 | 400   | 0.55       | 2.051E-03         |

Table 1: Performance of Various Computers Using Standard Linear Equations (HPL) Software at the beginning of the supercomputer age

In supercomputers, the "(apparently) sequential part" is technically of different origin – and as will be shown below, can be separated to contributions of different nature –, but has the same effect on \(1 - \alpha_{eff}\) in the frame of the classic model. To scrutinize the possible dependency of \(1 - \alpha_{eff}\) on time, data covering 25 years of the "supercomputer age" [Top500 (2017)], has been analyzed. The ratio of data \(R_{max}\) and \(R_{peak}\) provided \(E\), and using Equ. (2), \(1 - \alpha_{eff}\) has been calculated in function of time and ranking, see Fig. [3]. It looks like \(1 - \alpha_{eff}\) changes in an exponential-like way, both with the time and with the ranking in a given year.

In Fig. [5] \(1 - \alpha_{eff}\) values for the top 3 supercomputers

3This of course assumes that \(\alpha_{eff}\) is independent of the number of processors, which seems to be valid at this level of approximation.
Figure 4: The Top500 supercomputer parallelization efficiency. The $(1 - \alpha_{eff})$ parameter for the past 25 years and the (by $R_{max}$) first 25 computers. Data derived using the HPL benchmark.
Figure 5: The trend of the development of $(1 - \alpha_{eff})$ in the past 25 years, based on the first three (by $R_{max}$) and the first (by $(1 - \alpha \ast eff)$) in the year in question. Data derived using the HPL benchmark.
ers (ranked by $R_{\text{Max}}$) are displayed in function of time. The figure also contains the diagram of the best (ranked by $(1 - \alpha_{\text{eff}})$) computer in the year, which confirms that high computing performance strongly correlates with the efficiency of parallelization. Both methods lead to the same trend: the two regression lines shown in the figure are calculated for the #1 by $R_{\text{Max}}$ and by $(1 - \alpha)$, respectively. It looks like this development path (independently of technology, manufacturer, number and type of processors, etc.) shows a semi-logarithmic behavior over a quarter century. It is able to forecast the expected behavior of performance in the coming years, in the same way as the Moore observation does for single processors.

The architectural solution does not play a very important role here. As Fig. 3 depicts, both major architectural technologies result $(1 - \alpha_{\text{eff}})$ in the same order of magnitude, and $(1 - \alpha_{\text{eff}})$ raises with ranking of the computer, so some other factors may decide on ranking.

One might think it is another appearance of Moore’s law. However, consider that when calculating $R_{\text{Peak}}$, single-processor performance (clock frequency and component density, etc.) consequences of the Moore-observation are removed. What remains: how perfectly single-processors can work together.

For the first look it might be surprising to look for any dependency of $(1 - \alpha_{\text{eff}})$ on year of construction. Consider, however, that it represents the non-parallelizable fraction of execution time, and the need for higher performance requires to increase the number of processors comprised. The key is Eq. (7): $R_{\text{Peak}}$ increases linearly with $k$, and efficiency decreases according to $\frac{1}{1 - \alpha_{\text{eff}}}$. Supercomputers need a large number of processors, so the only way to put them together in an efficient and economic way, is to decrease $(1 - \alpha_{\text{eff}})$. Moore’s law assures that in the same year a very similar SPA technology is used by all manufacturers, so that computers from different manufacturers can be compared. Amdahl's law assures, that the only way to build many-processor system with higher $R_{\text{Max}}$ from the same single-processor components, is to reduce $(1 - \alpha_{\text{eff}})$.

### 4.3 Benchmarking supercomputers

Benchmarks, utilized to derive numerical parameters for supercomputers, are specialized programs, which run on the HW/OS environment provided by supercomputer under test. One can use benchmarks for different goals. Two typical fields of utilization: to describe the environment the supercomputer application runs on, and to guess how quickly an application will run on a given supercomputer.

To understand the operation of benchmarking, one needs to extend Amdahl’s original model in such a way, that the non-parallelizable (i.e. apparently sequential) part comprises contributions from HW, OS and SW. Among this, SW represents what was assumed by Amdahl as the total sequential fraction. As will be demonstrated in the discussion below, in the age of Amdahl, the other two contributions could be neglected compared to the SW contribution.

Obviously, the (apparently) sequential fraction $(1 - \alpha_{\text{eff}})$ cannot distinguish between (at least apparently) sequential processing time contributions of different origin, even the SW (including OS) and HW contributions cannot be separated. Real-life supercomputer applications and the ones used for benchmarking differ very much in their contribution to the non-parallelizable fraction. Different benchmarks provide different contributions to the non-parallelizable fraction of the execution time, so comparing results derived using different benchmarks shall be done with maximum care. Since the efficiency depends heavily on the number of cores, different configurations shall be compared using the same benchmark and same number of processors (or same $R_{\text{Peak}}$). Utilizing benchmarks without considering this restriction would result in absolutely different rankings, both for HPL and HPCG benchmarks.

If the goal is to characterize the supercomputer’s $HW + OS$ system itself, a benchmark program should distort $HW + OS$ contribution as little as possible, i.e. SW contribution must be much lower than $HW + OS$ contribution. In the case of supercomputers, benchmark HPL is used for this goal since the beginning of the supercomputer age. The mathematical behavior of HPL enables to minimize SW contribution. The $HW + OS$ contribution can also be lowered in such a way, as designers of Sunway do: they use 256+4 cores per processors, and 4 cores are dedicated to assist OS functionality [Dongarra (2016)], thus reducing the duty of OS to deal with cores by two orders of magnitude. In this way the resulting non-parallelizable fraction can be attributed mostly to $HW$ implementation.

If the goal is to estimate the expectable behavior of an application, the benchmark program should imitate the structure of the application. In the case of supercomputers, a couple of years ago HPCG benchmark has been introduced for this goal, since “HPCG is designed to exercise computational and data access patterns that more closely match a different and broad set of important applications, and to give incentive to computer system designers to invest in capabilities that will have impact on the collective performance of these applications” [7th HPCG Performance List (2017)]. However, its utilization can be misleading: the ranking is only valid for the HPCG application, and only utilizing that number of processors.

### 4.4 Validating the $\alpha_{\text{eff}}$ model for supercomputers

To validate the the $\alpha_{\text{eff}}$ model for supercomputers, one can compare the parameters and ranking derived using the HPL and HPCG [7th HPCG Performance List (2017)] benchmarks, see Table 2. For the items in the table the HW/OS environment (and so: the corresponding contributions to the non-parallelizable time) is the same, the difference is caused by the benchmark program structure. The differences in efficiency values delivered by the two benchmarks clearly show that the efficiency differs by two orders of magnitude. The $(1 - \alpha_{\text{eff}})$ values give the explanation: the non-parallelizable fractions are 2-3 orders of magnitude higher when measured using HPCG than using HPL.

This helps to understand the supercomputer development timeline: in HPL approach, all contributions are decreased as much as possible. In HPCG approach the SW contribution dominates: (all values $(1 - \alpha_{\text{eff}})_{\text{HPCG}}$ are nearly equal, except those where $(1 - \alpha_{\text{eff}})_{\text{HPL}}$ is an order of magnitude higher because of the high $HW$ contribution), and actually, the rest can be neglected. According to Eqn. (7), the increased $(1 - \alpha_{\text{eff}})$ causes considerable differences in

---

3 This separation cannot be strict: for example [Fu et al (2016)] utilizes 256+4 cores, where 4 cores assist system functionality at processor level [Dongarra (2016)].
the efficiency. For example, efficiencies of 'Cori' and 'Oakforest' differ by less than 10 % when measured using HPL and by more than 100 % when measured by HPCG.

Benchmarked supercomputers provide different $R_{Max}$, depending on value of $(1 - \alpha_{eff})$ of the benchmark. Fig. 6 displays some typical results, using Taihulight as example. (For other supercomputers the graphs are similar, except that the same $R_{Peak}$ is produced using processors with different single-processor performance, and so because of the different number of processors, the efficiency also changes.)

The two bubbles on the figure mark positions of measured $R_{Max}$ values, when using HPL and HPCG benchmarks, for Taihulight and K computer, respectively. As noted above, in the case of HPCG contribution of SW "pulls down" the payload performance much stronger in the case of Taihulight having much smaller $(1 - \alpha_{eff})$, than in the case of K computer.

As shown, building even bigger supercomputers has only sense for HPL class applications. HPCG class applications enjoy just marginally better payload performance than they would do on the same architecture equipped with two orders of magnitude less processors. Fig. 6 also helps to find out the optimum size of supercomputer for a specific application.

In the light of this discussion, results derived utilizing HPCG as benchmark program, gives no direct information about the hardware/software environment of the supercomputer: the contribution due to the program structure (the measuring device) is at least two orders of magnitude higher than the contribution due to the hardware/software system (the device under test), so the provided ‘ranking’ is the result of a kind of round-off effects, see the two rankings in Table 2 and also in Fig. 6. (BTW: $(1 - \alpha_{eff})$ strongly correlates with ranking, both for HPL and HPCG.) Adopting HPCG as benchmark enables hardware designers to utilize less expensive (and at the same time: less performable) architectural solutions, and the resulting HW will not be able to run HPL benchmark at some reasonable speed and/or efficiency, because in that case the high HW contribution will dominate. In the case of HPCG, however, an order of magnitude higher OS contribution or HW contribution makes no difference, and this benchmark does not honour developments in this direction, so the development of supercomputers will follow a different direction when directed by HPCG as benchmark. Building supercomputers about 1 % efficiency should not be a reasonable goal.

Just note that some dedicated measurements would enable to provide better estimations for the OS and HW contributions: making several dry (i.e. using the correct time but making no action) system handling calls the slope of the (by the present model, linear) dependency of $(1 - \alpha_{eff})$ on the number of system calls would provide the contribution of OS, while the intercept delivers the (HW+SW) fraction. Through changing the payload execution time, the SW contribution can similarly be estimated, which finally enables to estimate also the HW contribution, which would be a real merit of the HW.

### 4.5 Applications

In the case of benchmark programs $(1 - \alpha_{eff})$ is much lower, than in the case of real-life programs. (Amdahl (1967) estimated the non-parallelizable part to be above 20 %), i.e. about 2-3 orders of magnitude higher than even that of the HPCG benchmark. Because of this, the efficiency of real-life programs decreases even more strongly with the number of processors, so it is really worth to consider how many processors will provide optimum price/performance, if the requirement for absolute processing time enables it.

---

**Table 2: Comparing HPL and HPCG benchmark results of some TOP10 supercomputers**

| Computer Model | HPL Efficiency $(1 - \alpha_{eff})$ | HPL Rank | HPCG Efficiency $(1 - \alpha_{eff})$ | HPCG Rank |
|----------------|-------------------------------------|----------|-------------------------------------|-----------|
| Taihulight     | 0.742 3.273E-08 3.121e-5 4        |          | 0.932 1.040E-07 8                  |           |
| Tianhe-2       | 0.853 2.191E-07 9                  |          | 0.853 2.191E-07 9                  |           |
| Titan          | 0.649 1.409E-06 4                  |          | 0.016 2.043e-4 8                   |           |
| Sequalo        | 0.503 1.590E-06 6                  |          | 0.013 1.220e-4 8                   |           |
| K computer     | 0.503 2.000E-07 9                  |          | 0.012 1.220e-4 8                   |           |
| Trinity        | 0.731 2.212E-06 10                 |          | 0.016 2.043e-4 8                   |           |

Figure 6: Correlation of ranking and $\alpha_{eff}$, derived using $HP$ and HPCG.
$R_{\text{Max}}$ of Taihulight supercomputer in function of $R_{\text{Peak}}$ with different benchmarks

Figure 7: $R_{\text{Max}}$ performance in function of their peak performance $R_{\text{Peak}}$, at different $1 - \alpha_{\text{eff}}()$ values.
Supercomputer applications must be tuned to be scalable with the number of processors. For this goal $\alpha_{eff}$ can be utilized excellently. Before and after some change in the program structure the execution time can be measured using two different number of processors, and from those times $\alpha_{eff}$ can be derived. On the other side, using $\frac{R_{Max}}{R_{eff}}$ of a particular supercomputer (identical with $E$ in Equ. (4)) one can conclude the efficiency (i.e. execution time) of a particular application using that supercomputer. Practically this would be the goal of HPCG.

5. BOUNDS ON COMPUTING GROWTH

"The nature of this overhead (in parallelism) appears to be sequential so that it is unlikely to be amenable to parallel processing techniques. Overhead alone would then place an upper limit on throughput ... even if the housekeeping were done in a separate processor [Amdahl (1967)].

5.1 A new exponential law of computing growth?

In a recently published paper [Denning and Lewis (2017)] the authors have pointed out that in real systems the curves like Moore’s law describing component decreasing (so, maybe also the development of $\alpha_{eff}$ with time, see Fig. 5) will sooner or later saturate. Their analysis reveals that the exponential nature of the growth is the result of an interplay of many factors, and also that such growth is better described by a “logistic curve” which saturates after reaching some point.

Moore’s law is formulated for several dependencies. In that case a saturation point already reached: since cca. 2005 no more transistors can be added to a CPU in a reasonable way. For the number of transistors in a chip Moore’s law still persists, but not any more for the number of transistors in a processor. Similar turning point was experienced with the manufacturing technology (lithographic size) and is expected to occur with the atomic nature of technological materials. The saturation value, however, is not yet known and is different for the different reasons.

5.2 Limitations of building larger supercomputers

Fig. 5 shows up a behavior very similar to that of the Moore’s law, (it looks like that $(1 - \alpha_{eff})$ decreases year-by-year by a factor of cca. 1.5). Some reasons can also be seen why also this behavior is not without limitations. Using some reasonable assumptions about the different contributions to the non-parallelizable fraction mentioned above, their order of magnitude can be estimated, and also some saturation values can be forecasted.

For calculating bounds based on our extended model data published by [Dongarra (2016)] are used. The 13,298 seconds benchmark runtime on the 1.45 GHz processors means $2 \times 10^{13}$ clock periods. The absolutely necessary non-parallelizable activity is to start and stop the calculation. If starting and stopping a zero-sized supercomputer without OS could be done in 2 clock periods, then the absolute limit for $(1 - \alpha_{eff})$ would be $10^{-13}$.

If one considers a cca. 100 meter sized computer having 1 GHz cores, the signal round trip time is cca. $10^{-6}$ seconds, or $10^{5}$ clock periods, and a network message can be estimated to be of length $10^{-5}$ seconds (including operating time of HW), or $10^{5}$ clock periods. So, the absolute limit for $(1 - \alpha_{eff})$ of a computer with realistic size, but no operating system is $10^{-9}$.

We need to use, however, an operating system. If one considers context change with its consumed $10^{9}$ cycles, the absolute limit is cca. $10^{-9}$, on a zero-sized supercomputer. In addition, millions of cores must be manipulated through the system call, which contribution increases linearly with the number of cores and contribution from OS can be dominant at high number of cores. This is why designers of Sunway dedicated 4 cores per processor [Fu et al (2016)] to reduce this dependence by two orders of magnitude. As discussed, the application itself produces some non-payload activity, which can be assumed also to be at least in the range of $10^{3}$- $10^{6}$ clock cycles. It is probably a realistic estimation, that contributions of HW, OS and SW (the application itself) can sum up to at least $10^{7}$ clock cycles, resulting in a $10^{-8}$ absolute limit for $(1 - \alpha_{eff})$. As it can be concluded from results of benchmarks HPCG and HPL, the SW contribution dominates, so choosing the right benchmark is decisive.

Although it is a very rough estimation, it is worth to compare it to the value $3.3 \times 10^{-5}$, calculated from data published by [Dongarra (2016)] for the Sunway supercomputer [Fu et al (2016)]. Absolutely the same order of magnitude. As it is known from textbooks, according to Amdahl’s law, the available maximum speedup (the apparent computing throughput) is given by $\frac{T_{Max}}{T_{eff}}$, i.e. for the derived limiting value is about $10^{8}$. This should be multiplied with the computing throughput of a typical processor used in supercomputers, typically 10 GFLOPS. This results in $10^{18}$ FLOPS, i.e. about the "dream limit", targeted by several supercomputer building teams. This analysis provokes the question: how much is it realistic to plan building even larger SPA supercomputers?

5.3 Forecasts for exaFLOPS supercomputers

From TOP500 data the efficiency values in function of
Table 3 displays the results in tragically low efficiency. The pre-last column of efficiency values, the nature of efficiency also changes: while on HP CG benchmark to contradict to it.)

One can recall Table 1: putting more processors in an architecture, without making efforts for enhancing αeff results in tragically low efficiency. Probably, there is not much sense to build supercomputers with efficiency below one percent, but the fact that benchmark HP CG is accepted as a new metric for ranking supercomputer systems, seems to contradict to it.)

Figure 9: Comparing efficiencies of selected TOP10 (as of 2017 July) supercomputers in function of their peak performance \( R_{\text{Peak}} \), for the HPL and HPGC benchmarks. The actual \( R_{\text{Peak}} \) values are denoted by a bubble.

Table 3: \( R_{\text{Peak}} / \text{RM}\alpha \) of present TOP10 supercomputer architectures upgraded with more cores to provide 1 exaFLOPS, or the \((1 - \alpha_{eff})\) to be achieved to keep their present efficiency. Data are derived using the HPL benchmark.

\[ \frac{R_{\text{Peak}}}{\text{RM}\alpha} \]

\( \frac{R_{\text{Peak}}}{\text{RM}\alpha} \) can be calculated, see Fig. 8 (i.e. virtually the number of processors is changed for the different configurations). Since the efficiency values differ by orders of magnitude, the behavior of efficiencies changes drastically between the two benchmarks. The reported (measured) efficiency values are marked by bubbles on the figures.

One can recall Table 1: putting more processors in an architecture, without making efforts for enhancing \( \alpha_{eff} \) resulted in tragically low efficiency. The pre-last column of Table 8 displays the \( R_{\text{Peak}} / \text{RM}\alpha \) value for the first ten as of 2017 July supercomputers on the TOP500 list, if they were upgraded with more cores to provide 1 exaFLOPS. (For the trend of HPL efficiency see Fig. 8. Probably, there is not much sense to build supercomputers with efficiency below one percent, but the fact that benchmark HPCG is accepted as a new metric for ranking supercomputer systems, seems to contradict to it.)

Fig. 9 also underlines the importance of benchmarking: benchmark HPCG produces not only drastically lower efficiency values, the nature of efficiency also changes: while on the left side increasing \( R_{\text{Peak}} \) by two orders of magnitude triggers less than two orders of magnitude decrease in efficiency, on the right side the efficiency decreases more than two orders of magnitude. Similarly to the case shown in Table 1, putting more processors in an architecture, without making efforts for enhancing \( \alpha_{eff} \) results in tragically low efficiency.

Building exaFLOPS supercomputers for applications akin HPCG is nonsense: the more processors, the less efficiency, thanks to the two orders of magnitude higher \((1 - \alpha_{eff})\) value. Supercomputers running HPCG will provide just a few percent efficiency, and provide \( \text{RM}\alpha \) about 0.1 exaFLOPS. To achieve 1 exaFLOPS \( \text{RM}\alpha \), a few times \( 10^{-9} \) value of \((1 - \alpha_{eff})\) should be achieved.

Fig. 10 shows the absolute computing performance \( \text{RM}\alpha \) for some of the TOP10 supercomputers. When calculating the diagram lines, virtually the number of processors were varied. The actual value is denoted by a bubble. As expected, \( \text{RM}\alpha \) for Taihulight seems to saturate around \( .35 \) exaFLOPS, the rest of supercomputers at much lower values.

One can also calculate from reverting Equ. 7 what enhancement in \((1 - \alpha_{eff})\) would be necessary for those configurations to achieve 1 exaFLOPS and at the same time to keep their present \( \frac{R_{\text{Peak}}}{\text{RM}\alpha} \) keep their present efficiency, see the last column of Table 8. For deriving the results, benchmark HPL was used. As from Fig. 8 can be concluded, for the future supercomputers achieving \( \frac{R_{\text{Peak}}}{\text{RM}\alpha} \) around 0.73 can be expected, the development will have to target reducing \((1 - \alpha_{eff})\).

In the light of the analysis above the primary candidates to deliver 1 exaFLOPS are the supercomputers which are able to produce presently \((1 - \alpha_{eff})\) around \( 10^{-8} \). To achieve
Figure 10: $R_{Max}$ performance of selected TOP10 (as of 2017 July) supercomputers in function of their peak performance $R_{Peak}$, for the HPL benchmark. The actual $R_{Peak}$ values are denoted by a bubble.
considerably lower \((1 - \alpha_{\text{eff}})\), all contributions mentioned in our model must be considerably lowered. The physical size (mainly due to need of cooling) can hardly be decreased, although 3D arrangements can help also here. The present layering of computing stack requires context change, and starting/preparing/stopping an application is inevitable and the linearly increasing contribution of OS due to the large number of cores must be eliminated. Lowering only one of these contributions is useless. That is, at least the physical size (the speed of the light) really puts an upper bound to supercomputers performance.

At least if utilizing Single Processor Approach persist. Maybe it is really time to introduce using \[\text{Cooperating cores (2016)}\] and \[\text{Reasonable layering (2017)}\]?

5.4 Is perfectness the common reason?

The number of similarities between the present law depicted in Fig. 5 and Moore’s law suggests to find some common reason. An interesting idea is that in both cases a kind of “perfectness” is approached. The goal to be achieved in the case of Moore’s law is infinitesimally small component size, in the case of parallelization infinitesimally small non-parallelizable part. Both dependencies show exponential behavior, and as presented, both of them will behave (sooner or later) as a logistic curve. These laws are able to forecast the expected behavior of performance in the coming years, and serious consideration must be given to their scope of validity.

6. CONCLUSION

Amdahl’s law (published in a quarter century before supercomputing was born) seems to bound supercomputer applications and supercomputer architecture itself. The introduced \(\alpha_{\text{eff}}\) parameter, actually: the (at least apparently) sequential fraction, can be used to describe architecture and operation of supercomputers built in Single Processor Approach, and can effectively assist both supercomputer application makers and supercomputer constructors.

The simple model describes all experiences acquired by the supercomputer community. The model provides surprisingly good numerical values for all published performance data, although dedicated measurements are needed to pinpoint role, size and interaction of the components in the suggested model. The paper validates that Amdahl’s 50-years old model (with slight extension) correctly describes the performance limitations of the present supercomputers. Using some simple and reasonable assumptions, the absolute performance bound of supercomputers was concluded, furthermore that serious enhancements are still necessary to achieve the exaFLOPS dream value.

Acknowledgement

The author kindly acknowledged support from National Research, Development and Innovation Office – NKFIH, grant no. K 125547.

7. REFERENCES

The Top 500 supercomputers

https://www.top500.org/

HPCG Benchmark

http://www.hpcg-benchmark.org/custom/index.html?lid=1548&lid=282

Dongarra J Performance of Various Computers Using Standard Linear Equations Software TN 37996-1301

Fu Haohuan and 18 authors The Sunway TaihuLight supercomputer: system and applications, Science China Information Sciences 59 (2016)1-16.

Denning P.J. and Lewis T.G. (2017) Exponential Laws of Computing Growth, Comm. of the ACM 59(2016)1-16.

IEEE, Architectures for the Post-Moore Era

https://www.computer.org/micro/2016/09/12/architectures-for-the-post-moore-era-call-for-papers/

Amdahl, G. M. (1967) Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities, AFIPS Conference Proceedings 30(1967)483-485.

Karp A.H. and Flatt H.P. (1990) Measuring Parallel Processor Performance, Commun. ACM 33(1990)539-543.

Végh J, Kicsák Á., Bagoly Z. and Molnár P. (1967) An alternative implementation for accelerating some functions of operating system, ICSOFT-EA 2014 - Proceedings of the 9th International Conference on Software Engineering and Applications (2014)494-499.

Hennessy, J.L. and Patterson D. A. (2007) Computer Architecture: A Quantitative Approach, Morgan Kaufmann Publishers, 2007.

Yavits L., Morad A. and Ginosar R. (2014) The effect of communication and synchronization on Amdahl’s law in multicore systems, Parallel Computing 40(2014)1-16.

David T, Guerraoui R. and Trigoniakis, V. (2013) Everything you always wanted to know about synchronization but were afraid to ask, Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP ’13) (2013)33-48.

Pingali K and 11 authors (2014) The Tao of Parallelism in Algorithms, SIGPLAN Not. 46(2011)12-25.

Dongarra J Report on the Sunway TaihuLight System UT-EECS-16-742

Végh J (2016) A cycle accurate simulator for Explicitly Many-Processor Approach (EMPA) computer https://github.com/jvegh/EMPAthY86

Végh J (2017), Do we need cross layering activities or reasonable layering in computing systems? IEEE Design&Test, submitted