Towards Exascale Computing

Prof. Houssain Kettani

Professor of Computer Science,
Florida Polytechnic University, USA
hkettani@flpoly.org

***Abstract* – Due to hardware limitations, parallel computing became an integral part of our lives that it is hard to imagine a device that is not using multiprocessor power, including smartphones. What started as a hardware solution to physical limitation, prompted software engineers to adopt to parallelism. This in turn, compelled theoretical computer scientists to develop theoretical solutions to algorithms design and analysis to provide a solution that is parallel oriented rather than a serial oriented one, using divide-and-conquer algorithm design technique.Due to the exponential growth in high performance computing, the fastest computer in the world is projected to run at one Exaflop/s by 2020, or 10^{18} operation/second, ushering the era of exascale computing. As we move towards exascale computing and beyond, we need to keep in mind that the power of humanity is not in the powerful machines we develop, but remains in our intelligence and ability to develop solutions to problems at the basic level.**

I. INTRODUCTION

In 1985, the fastest computer in the world reached one Gigaflop/s, or one billion (10^9) calculation per second. By 1997, the speed reached oneTeraFlop/s or one trillion (10^{12}), then onePetaFlop/s or one quadrillion (10^{15}) by 2008. By 2017, the fastest computer in the world performedalmost one hundred PetaFlop/s and many hand-held devices including smart phones are faster than the fastest supercomputer in the 1980s. The oneExaFlop/s mark, or one quintillion (10^{18}) is expected to be reached in 2020. Thus, in the past thirty years, advances in high performance computing have increased the performance by million times, and decreased the volume of the machine by similar order.In addition, current hand-held devices such as smartphones have performance that rivals those machines of the 1980s.

The website Top500.org assembles and maintains a list of the 500 most powerful computer systems. The list has been compiled twice a year since June 1993 and the performance development and projectionare depicted in Fig. 1 as posted in [1]. The middle line is for the fastest machine, the bottom is the bottom of the list (500), while the top line is the aggregated performance of all 500 machines. The figure shows that the performance of the fastest computer in the world increased from 60 GF in 1993 to 93 PF or 93,000,000 GF in 2016. This amounts to an average speed doubling approximately every eleven months. In other words, the span of years to move from one class (1000^n) to the next class (1000^{n+1}) is around twelve years.Accordingly, a performance at the zettascale (10^{21}) and yottascale (10^{24}) is projected to be achieved by 2033 and 2045, respectively.

Currently, the fastest supercomputer has close to eleven million cores and consumes over 15 MW (Mega or million Watts). It is like 150,000 light bulbs of 100W on at the same time. It is more than a million times faster than a personal computer. So, one second of computing using the fastest supercomputer is equivalent to almost two weeks using a PC, while one hour is equivalent to over a century on a PC! However, such humongous machines present huge complexity in operation, maintenance, protection, etc. This remains an active area of interdisciplinary research for continuous improvement in speed, efficiency, hardware and software development as well as algorithms design and analysis to advance the state of the art of parallel computing. Advances in high-performance computing allowed humans to solve problems that were impossible to solve few years before, including weather (earth and space) forecast, gene permutations, hurricane tracking, asteroids/comets tracking, spying, etc.

In the sequel, we analyze these advances in high performance computing in the context of parallel computing and theoretical analysis. We argue that advances in technology cannot sidestep theoretical breakthroughs. Accordingly, while some problems need to be theoretically prepared to be solved in parallel, other problems cannot be naively solved even using the fastest supercomputer.

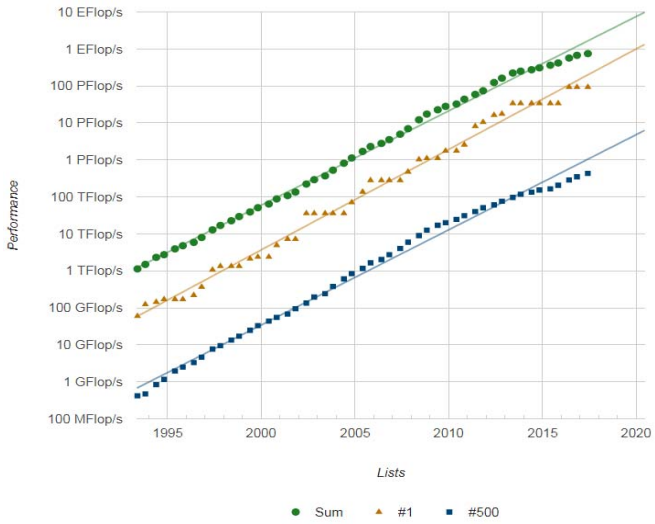

Fig. 1.Performance development since 1993 and its projection to 2020.

978-1-5090-3012-5/17/$31.00 ©2017 IEEE

II. THEORETICAL FRAMEWORK

The first equation that we encounter in the design and analysis of algorithms is

$$T(n) \approx c_{op}C(n), \quad (1)$$

where$T(n)$ the running time of the algorithm with input size $n$and $C(n)$ is the number of times the algorithm's basic operation is executed, while c_{op} is the execution time for the basic operation [2]. The importance of this equation is that we can compare algorithms irrelevant of era or technology by comparing their basic operation counts. For example, we can analyze Euclid's algorithm for finding the greatest common divisor, which was developed more than 2300 years ago and has count $C_1(n)$, and compare it with a newly developed algorithm with count $C_2(n)$.

For less efficient algorithms, the basic operation count $C(n)$, can be exponential or factorial, which makes the solution infeasible for reasonable values of n even if advanced computing technology is used. Classical examples are brute force solutions to knapsack and traveling-salesman problems. The former is in $O(2^n)$, while the latter is in $O(n!)$. Such class of algorithms can only be solved for small input size n and advances in technology do not help much.

For example, for $C(n) = 2^n$ and $n = 100$, we have $C(100) \approx 10^{30}$. Then, using a theoretical Exascale supercomputer, in which $c_{op} = 10^{-18}$, we obtain $T(100) = 10^{12}$ seconds or 32,000 years!Thus, using the fastest computer in the world in the next decade, cannot solve a brute-force algorithm for the traveling-salesman problem with one hundred cities. This is a clear example that technology does not sidestep theoretical analysis of algorithms.

III. PARALLEL COMPUTING

In parallel computing, we apply divide-and-conquer design strategy which starts with a problem of size n, reduces it to m sub-problems of size n/m that are sent to m processors, then the solution to the original problem is obtained from the solution to these m sub-problems.

Hardware engineers were able to make processors faster and smaller until the start of this century, when heat generation started melting materials and causing malfunction of processors. A smart and cost effective solution to this physical limitation was to stuff several cores in one processor: two for duo, four for quad, etc. Thus, parallel computing became ubiquitous, and even hand-held devices such as smartphones are becoming increasingly multicore. So for instance, instead of having a one-core processor running at 1GHz, we have two-cores each one runs at 1GHz in one processor and market the product as running at 2GHz.

It turned out that hardware engineers shifted the problem to software engineers as the software and operating systems were run in serial instead of parallel. Thus, the maximum a serial program can achieve is the power of one of the cores and not the advertisedoverall power of the processor. This prompted software engineers to upgrade software and operating system to take advantage of this parallelism. Hence these systems need to operate on several cores simultaneously to maximize the use of the hardware capability and achieve the advertised performance capabilities.

IV. LIMITATIONS

Applying a serial algorithm on a parallel machine will not help in achieving the maximum performance at such machines. However, solving a problem in parallel instead of serial may not be trivial. In fact, some problems may not be solvable in parallel. A classic example of this is pregnancy: we cannot expect a baby in one month by giving nine women one month each, at least with current science and technology!

Other problems like building a bridge or a tall building for instance, we cannot finish the tenth floor without building the preceding ones. Yet, the closest to parallelism we got in these examples is by building each floor or bridge segment separately then assembling theses segment or blocks together to obtain the final product: a building or a bridge. For example, Broad Sustainable Building Company has completed in nineteen days a 57-storey tower in the Chinese city of Changshan, The tower contains 800 apartments and office space for 4,500 people. The company also proposed to build Sky City in ninety days. Which is a 200-storey skyscraper that would be the tallest in the world. The company, which specializes in eco-friendly, pre-fabricated builds, managed the amazing speed of construction by pre-building blocks in factories, then stacking and screwing them together on site – a bit like giant Lego blocks [3].

Another example of a classic serial problem that has been proven to be solved in parallelis base number conversion.The classic algorithm to convert from decimal to base b, is to repeatedly divide the integer part of d by b and record the remainders. Then, starting with the first recorded one, we write down these remainders from right to left. Thus, we obtain the integer part of the corresponding base number b. This algorithm is referred to as repeated division. As a consequence, when converting from decimal to binary using the classic algorithm, we only know the value of the hundredth bit after obtaining the values of the previous ninety-nine bits. In [4], a parallel algorithm was outlined based on the following theorem that was proved therein.

THEOREM.*Let $d \geq 0$ bea real number and let $b \geq 2$ be an integer. Let d, written in base b, be given by*

$$d = (d_n d_{n-1} \ldots d_0 . d_{-1} \ldots d_{-m+1} d_{-m})_b \quad (2)$$

wheren and m are positive integers and, for $-m \leq j \leq n$, the digitsd_j of the expansion belong to the set $\{0, 1, \ldots, b-1\}$. Then, for each j,

$$dj = \lfloor db^{-j} \rfloor - b \lfloor db^{-j-1} \rfloor \quad (3)$$

In short, this theorem makes it possible to find each bit without the need to know other bits. Consequently, we can develop a parallel algorithm by which cohorts of bits are sent to different processors then the overall solution is obtained from the output of these processors, see [4] for more details. This is a proof of concept for a conversion form serial to parallel of a problem that is historically known to have only a serial solution.

V. CONCLUDING REMARKS

While our extreme multi-core technology is taking us to exascale computing, young researchers should not be tempted to ignore computing essentials such as efficiency of algorithms and theoretical analysis. There are numerous examples in which a naïve approach of solving a problem makes the solution practically not feasible, yet a different look at the problem makes it solvable in a reasonable amount of time on a personal computer. The physical limitation that hardware engineers encountered has been passed to software engineers through the multi-core solution. The software engineers, whoin turn have passed the problem to theoretical computer scientists and applied mathematicians to develop parallel solutions to problems that are known to be serial. As we move towards exascale computing and beyond, we need to keep in mind that the power of humanity is not in the powerful machines we develop, but remains in our intelligence and ability to develop solutions to problems at the basic level.

REFERENCES

[1] Top 500 The List. Retrieved from https://www.top500.org/

[2] A.V. Levitin. 2012. Introduction to the design and analysis of algorithms (3rd ed.). Addison Wesley. ISBN 9780132316811.

[3] CityMetric. 2015, March,11. A Chinese company built a 57-storey tower in 19 days.Retrieved from http://www.citymetric.com/skylines/chinese-company-built-57-storey-tower-19-days-830

[4] H. Kettani. 2006. On the conversion between number systems. IEEE transactions on circuits and systems II, Vol. 53, No. 11, 1255-1258.