IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. VII (Feb. 2014), PP 36-43
www.iosrjournals.org

A review of High Performance Computing

G.Sravanthi[1] , B.Grace[2] , V.kamakshamma[3]
[1 2 3]*Department of Computer Science, PBR Visvodaya Institute of Science And Technology, India*

Abstract: *Current high performance computing (HPC) applications are found in many consumers, industrial and research fields. There is a great deal more to remote sensing data than meets the eye, and extracting that information turns out to be a major computational challenges, so the high performance computing (HPC) infrastructure such as MPP (massive parallel processing), clusters, distributed networks or specialized hardware devices are used to provide important architectural developments to accelerate the computations related with information extraction in remote sensing. The focus of HPC has shifted towards enabling the transparent and most efficient utilization of a wide range of capabilities made available over networks. In this paper we review the fundamentals of High Performance Computing (HPC) in a way which is easy to understand and sketch the way to standard computers and supercomputers work, as well as discuss distributed computing and essential aspects to take into account when running scientific calculations in computers.*
Keywords: *High Performance Computing, scientific supercomputing, simulation, computer architecture, MPP, distributed computing, clusters, parallel calculations.*

I. Introduction

High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business. the high-performance computing community has long advocated composing individual computing resources in an attempt to provide higher quality of service (in terms of processing time, size of data store, bandwidth/latency, remote instrument access, special algorithm integration, and so on, for example)

HPC, as a discipline, is largely concerned with software design and implementation issues associated with achieving maximal performance for a given algorithm; the choice of the algorithm itself — for example, an iterative solver for a system of linear equations — is more properly the focus of computational science than of HPC. For the most part, the issues that most affect HPC are:

- Instruction-level parallelism [1]. [2]. [3], [4], [5], [6],
- The storage hierarchy [1], [3], [4], [6],
- High performance compilers [1], [3], [5], [6],
- Shared memory parallelism (e.g., OpenMP) [1], [3], [6],
- Distributed parallelism (e.g., MPI) [1], [2].
- Scientific libraries [5], [6],
- I/O [6],
- Visualization [2].

In addition, the past several years have seen the rise of the issue of remote, heterogeneous (i.e., Grid-based) computing. The pedagogical challenge for the discipline of HPC is to find means of expressing these basic concepts in a manner that is approachable by scientists and engineers who have strong mathematical and scientific backgrounds but modest software development experience. Thus, teaching strategies for HPC require minimal jargon and maximal intuitiveness.

Science and technology plays an important role in improving the quality of life. Many products have been designed to extend human capability and perception in various ways. Numerous scientific and engineering problems must be solved before a very complex product can be designed. This approach used to solve science and engineering problems. Traditionally, scientific and engineering problems can be studied by conducting extensive experiments and analysis . Although it is unavoidable, this approach is very costly and time consuming .The emerging of computer technology 50 years ago introduced tremendous changes to scientific study . Instead of spending precious time and expense conducting real experiments, problems such as weather forecasting, structural analysis, molecular dynamics, large scale financial and economic systems can be described using a mathematical model, then simulated using a computer. Afterwards, when more understanding are obtained via an analysis and visualization, experiments can be conducted to verify the simulation results or collect more information to fine-tune the model. The clear advantages of this second approach are faster turn-around time and much less cost. However, the computing power needs to solve these kind of problems is enormous. This is the rational for recent emerging of *Computational Science* which is the study of techniques and tools to tackle compute-intensive applications.

High Performance Computing (or HPC) is an emerging discipline concerning the solving of large scale, compute intensive mathematical problems using High Performance Computer Systems. Numerous applications in various fields depend on high performance computing. The examples of these application are chemical process simulation, numerical weather prediction, finite element analysis, molecular dynamics, quantum chromo dynamic, air pollution control, and seismic migration.

Application Area	Computation Tasks and Expected Results
Magnetic recording technology	To study magnetostatic and exchange interaction to reduce noise in high-density disk
Rational drug design	To develop drug that cure cancer or AIDs by blocking the action of HIV protease
High-speed civil transport	To develop supersonic jet through computational fluid dynamics running on super computer
Catalysis	Computer modeling of biomimetic catalysts to analyze enzymatic reaction in manufacturing process.
Fuel combustion	Designing better engine model via chemical kinetics calculations to reveal fluid mechanical effects.
Ocean modeling	Large scale simulation of ocean activities and heat exchange with atmospherically flows
Ozone depletion	To study chemical and dynamical mechanisms controlling the ozone depletion process.
Digital anatomy	Real Time medical imaging , MRI
Air pollution	Simulated air quality model
Protein structure design	3-D structural study of protein formation using MPP
Image understanding	Real-time image rendering
Technology linking research to education	Scientific and Engineering Education using computer simulation.

Table 1 Grand Challenge problems

- There are a lot of benefits from high performance computing such as
  - Cost effective
  - Simulate physically challenging experiments
  - Extract meaningful information from massive data

US High-Performance Computing and Communication program had identified some substantially important applications that demand high performance computing [7]. These so called Grand Challenge Applications are listed in Table 1.

II. Hardware Basics

The basic scheme of a single computer is simple; it is sketched in fig. 1. A computer contains memory, which is a set of physical devices to store information. The memory contains both data to use in the calculations, and coded instructions to be provided to the Control Unit so that the calculations can be performed. The Control Unit controls the data flow and the operations that are to be performed in the Arithmetic Logic Unit (ALU). The ALU performs both arithmetic operations on numbers (like addition and subtraction) and logic operations (like AND, OR, XOR, etc.) on the binary digits (bits) of the stored variables. Computers also include a clock, which operates at a given frequency (the clock frequency or clock-rate). The clock-rate determines the number of maximum operations performed per second: An arithmetic or logic operation (as well as each stage an operation is divided into) takes at least one clock cycle. The Control Unit and the Arithmetic Logic Unit together form the CPU. The basic computer device also includes an interface which enables its interaction with the human user (input/output). High performance computers are essentially formed by the accumulation of CPUs linked in a smart way.

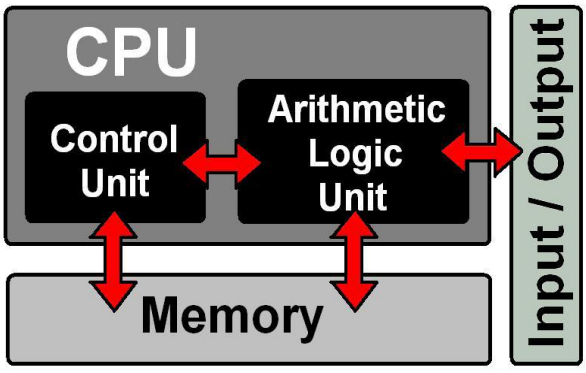

Figure 1: Scheme of the basic parts of a computer [8]. The red arrows indicate information flow

In fig. 1, red arrows indicate information flow. This flow can be physically handled by different devices (the network). One or several CPUs together with some communication devices can be set on a thin layer semiconductor with electronic circuits (i.e., on a chip), to form a processor or microprocessor. The CPU interacts with the outside world via the input/output interface. This interface enables, for example, the system to be managed by the human user (e.g., a keyboard is an input device, and a monitor is an output device).

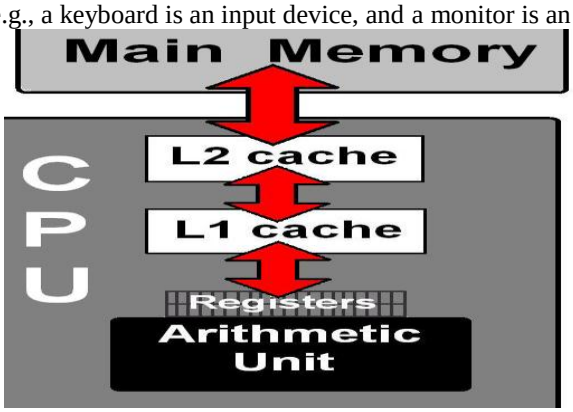

Figure 2: *Scheme of the flow of information (red arrows) in a computer to and from its Arithmetic Unit through a hierarchy of memories [8]. The cache can have more levels (not necessarily two). Usually, the closer a device is to the Arithmetic Unit, the faster the information transferring, but the lower the storage capacity (see fig. 3). The disk, as well as eventual external memories like tape libraries, are not directly connected to the CPU. Some architectures also include direct access paths which connect the registers with the main memory*

The data moves to and from the ALU through a memory hierarchy following the pattern displayed in fig. 2. Information can be stored in numerous devices which exist for that purpose (the memory), each having a different maximum amount of bytes for storage (size) and different bandwidth (maximum rate for information transfer). They also have distinct latencies (the latency is the amount of time between a request to the memory, and the time when its reply takes place). For example, if **y** bytes of information are to be transferred from a memory device which has a latency of l seconds, and a bandwidth of **b** bytes/second, then the minimal time required for the information to be delivered will be

$$\mathbf{t = l + y/b}$$

Not only do the different kinds of memory have latency and a bandwidth, but also the network does. Latencies and bandwidths have a major influence on a computer's performance, especially in parallel machines. In fig. 2 we can see the scheme of connection of an arithmetic unit with several types of memories. Commonly, closer connections are with memories with lower latency and higher bandwidth, though smaller size.

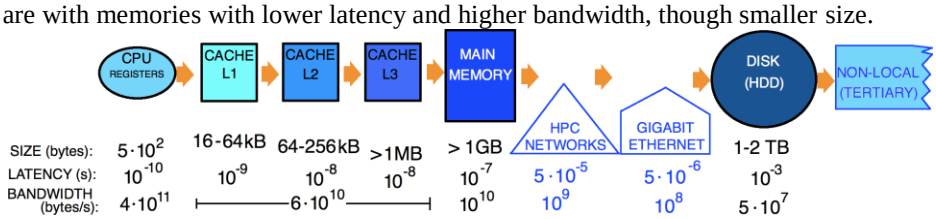

Figure 3: *Hierarchy of memories in a computer [7, 9, 10], with typical values of their latency, memory size and bandwidth. The lower the delay in data transfer (latency), the higher the bandwidth and the lower the size of the memory. Blue borders indicate elements typical in supercomputers, while black borders indicate elements which are present both in supercomputers and in personal computers. White shapes do not represent memories, but networking devices for interconnection of nodes in supercomputers.*

A scheme of the memory hierarchy, including their present-day typical sizes, bandwidths and latencies, is displayed in fig. 3.The information which is expected to be used immediately by the CPU is stored in its registers, which have a very low latency, but can only store a small amount of information. At present, typical CPUs have between 16 and 128 user-visible registers [8]. The next level in the hierarchy of memories (after the registers) is the level of cache memories. Typically, there exist three different levels within this cache memory, which are usually denoted with L1, L2, L3. When the CPU needs some data, it first checks whether or not they are stored in the cache. Efforts in circuit integration are specifically aimed at increasing the storage capabilities of caches, in order to reduce the time to access the information the CPU requires. After caches, the next level in the memory hierarchy is the main memory (which is sometimes called RAM, although this acronym refers to a specific type of technology).

The disk (hard disk drive, or HDD) has larger storage space, but higher latency and lower bandwidth. Although access to the disk is the slowest if compared to access to other memories, it has much more capability of storing information permanently [11] (external memories, such as CD-ROMs, pen drives, etc. excepted). The first disk memory was developed by IBM in 1956. This first device was able to store 2 kilobits/in2, while disks

manufactured today can store data at densities of 0.25 Terabits/in2 [12]. In the last decades, the space of data volumes is doubling each year or even faster [13, 14]. It is worth mentioning that the increase in disk memory capabilities has been boosted by the discovery of giant magneto resistance [15, 16]. This phenomenon makes it possible to manufacture MRAM memories which store information (bits) in magnetic layers [17], resulting in storage capabilities larger than those of previous technologies.

The increase in memory size of devices such as caches, RAMs and disks is quite useful for scientific simulation, because much of the information of the tackled complex systems has to be stored frequently during the calculation process, which makes memory an important limiting factor for in silico scientific calculations. Both the amount of available memory (in disk) and the speed to access information in all levels of the hierarchy imply major limitations to scientific calculations. Data storage is reported to be a big energy consumer; moreover, its power intake tends to grow because storage requirements are increasing over and over, and disks are faster and faster [18] . The low speed to access the information on disks is another drawback of the current technology. I/O (input and output to disk) bandwidth has not advanced as much as storage capacity. As stated in [13]: 'In the decade 1995-2005, while capacity has grown more than 100-fold, storage bandwidth has improved only about 10-fold'. External1 devices to store information can be considered the last level in the hierarchy of memories. These devices can be CD or DVD disks, USB flash drives, or different technologies. Massive storage devices, such as tape libraries are often used in supercomputers. Sometimes, the words primary memory for registers, cache and main memory, secondary memory for hard disks and tertiary memory for non-local memories are used.

III. Super Computer

Because of remarkable advances in computer technology, scientists now have a new problem solving tool, a supercomputer. Supercomputers supported by high powered graphics workstations are extremely valuable resources for a wide range of scientific investigations. High performance computing is becoming increasingly more important to many scientific and engineering disciplines, so it is important for educators to prepare future generations to be ready for that growing demand.

A new supercomputer being deployed this month in the U.S. is using solid-state drive storage(SSD) as an alternative to DRAM and hard drives, which could help speed up internal data transfers because SSDs are lightning fast. If this is close, or as fast as the RAM, maybe we could take the RAM out and replace it with an SSD. Because it is non-volatile and has great storage, it could bring better performance and saving more power (since sleeping will be considered as shutting down because there is no RAM to power up) and speed up booting times.

The supercomputer, called Catalyst, will be deployed at Lawrence Livermore National Laboratory in Livermore, Calif. Built by the U.S. Department of Energy, Cray and Intel, the supercomputer delivers a peak performance of 150 teraflops and will be available for use starting later this month.

Catalyst has 281TB of total SSD storage and is a giant computing cluster broken into 324 computing units, called "nodes" by LLNL. Each computing unit has two 12-core Xeon E5-2695v2 processors, totaling 7,776 CPU cores for the supercomputer. Each node has 128GB of DRAM, while 304 nodes have 800GB of solid-state drive storage. Additionally, 12 nodes have 3.2TB of solid-state drive storage for use across computing units.

The supercomputer is built around the Lustre file system, which helps break bottlenecks and improves internal throughput in distributed computing systems.

The super computer performance is measured in number of floating point operations per second (FLOPS) that the machine is able to perform. The overall performance of the super computer is nowhere near that of the world's fastest supercomputer, Tianhe-2, which delivers a peak performance of 54.9 petaflops, but the implementation of solid-state drives as an alternative to both volatile DRAM and hard drives sets Catalyst apart.

"As processors get faster with every generation, the bottleneck gets more acute," -said Mark Seager, chief technology officer for Technical Computing Group at Intel.

The throughput in the supercomputer is 512GB per second, which is equal to that of Sequoia, the third-fastest supercomputer in the world, which is also at LLNL, Seager said. Sequoia delivers peak performance of 20 petaflops. Intel's 910 series SSDs with 800GB of storage are being used in Catalyst. The SSDs are plugged into PCI-Express 2.0 slots, the same used for graphics cards and other high-bandwidth peripherals.

Faster solid-state drives are increasingly replacing hard drives in servers to improve data access rates. SSDs are also being used in some servers as cache, or short-term storage, where data is temporarily stored for quicker processing. For instance, Facebook replaced DRAM with flash memory in a prototype server called McDipper, and is also using SSDs for long-term cold storage. Though SSDs are more expensive than hard drives, observers say SSDs are poised for widespread enterprise adoption as they consume less energy and are becoming more reliable. SSDs are also smaller and can provide more storage in fewer servers. Samsung in

August announced faster V-NAND flash storage chips that could be 10 times more durable than current flash storage. With faster SSD storage, Catalyst is adept at solving "big data" problems, such as in bioinformatics, analytics and natural language processing, LLNL said in a statement. LLNL has developed a system so memory arrays are mapped directly to DRAM and SSDs, which helps in faster processing of serial applications like gene sequencing. Seager said Catalyst is partly an experiment in new supercomputer designs as the nature of applications and hardware changes.

IV. Massive Parallel Processing

MPP (massively parallel processing) is the coordinated processing of a program by multiple processor s that work on different parts of the program, with each processor using its own operating system and memory . Typically, MPP processors communicate using some messaging interface. In some implementations, up to 200 or more processors can work on the same application. An "interconnect" arrangement of data paths allows messages to be sent between processors. Typically, the setup for MPP is more complicated, requiring thought about how to partition a common database among processors and how to assign work among the processors. An MPP system is also known as a "loosely coupled" or "shared nothing" system.

MPP is a complicated process requiring a certain database functions to be shared between all involved processors. Messages are exchanged between processors via an interconnection of data paths during MPP. MPP is typically found in applications like decision support systems and data warehouses. Supercomputers are also examples of MPP architecture.

V. Distributed Computing

In recent times, other solutions for HPC, such as grid computing, cloud computing and volunteer computing, have become very popular. These three more modern ways of computing are said to be ways of distributed computing. Distributed computing is based on the concept of doing high performance calculations using geographically distant machines, which has been enabled with the advent of the internet and high-speed networks. Computers participating in a given problem can lie thousands of kilometers away from each other, but they can share information through the internet.

Grid computing [22] uses geographically distant nodes to solve a given problem simultaneously in a cooperative way. grid computing capabilities are usually managed by a given organization, and the computational resources (mediated by physical machinery) which support the calculations are provided by different supporting institutions and organizations, which can be companies, research groups, laboratories, universities, etc. A management committee distributes the computational capabilities at every moment among the requests of different groups of users. The groups of people taking part in grid computing projects for solving problems are usually called virtual organizations since these groups are frequently heterogeneous, being formed by many people from different organizations which are geographically distributed.

The essential aspect of a virtual organization is that it is formed for a specific project. Virtual organizations can act either as producers or consumers of resources (or both). Various virtual organizations involved in a grid computing project are mutually accountable; i.e., if one misbehaves, the others can cease sharing resources with it.

Since many computer cores throughout the world are working together, much computing power can be accumulated, which enables solving many problems whose solution may not be feasible even in the most powerful supercomputing clusters. This generates vast amounts of data, which spurs the creation of large collective databases [13]. Large grid computer facilities are often used by a large number of users, which helps to match the demand of the computational resources with their availability. It is also worth noticing that grid computing projects commonly operate under open-source software standards, which eases the development of software applications and the cooperation among groups.

A popular software package to manage grids is the Globus Toolkit, including the GRAM software as a tool for the users. Grid facilities, as well as cluster computers, frequently run in Linux operating systems. grid computing has been successful in numerous research fields, such as drug design, bimolecular simulation, engineering and computation for industry, Chemistry, Geology (e.g. earthquake simulations) or meteorology [22, 23]. It also play recently, another kind of distributed computing, volunteer computing, has become a useful tool for scientific purposes. volunteer computing consists of using the computation power of machines which were neither devised nor purchased to do scientific calculations, but for use in daily life. Common PCs and laptops connected to the internet, like those in millions of homes, can perform calculations to solve scientific problems. It is only necessary that the owner of the computer agrees and installs the appropriate software (this is the reason why this kind of computing is called volunteer computing). As stated in [24], there are hundreds of millions of idle PCs potentially available for use every moment, and the majority of them are strongly underused. Moreover, while the complexity and the network effciency have increased following their own

Moore's Laws, the number of computer users has increased at even a higher rate during the last decades, which makes the potential capabilities of volunteer computing huge [24].

Volunteer computing projects often rely on the BOINC open-source software16, which is also appropriate for grid computing. Although grid computing and volunteer computing share some features, there is a key difference between them. grid computing is commonly symmetric while volunteer computing is commonly asymmetric. This is, in the former, one organization can borrow resources one day, and supply them the next; in the latter, contributors (particular computer owners) commonly just provide resources to the project.

Apart from grid computing and volunteer computing, cloud computing [25,26, 27] also supplies computational capabilities for scientific calculations by connecting to remote machines via the internet. This is performed by powerful computers that companies dedicate for this purpose (usually for a fee). In cloud computing, a set of virtual servers work together to satisfy user requests, enabling interactive feedback and taking advantage of the available computing capabilities to maximize their use. Cloud computing has some advantages with respect to other ways of computing. For example, it enables the user immediate access to computational resources without the need to obtain approval from an allocations committee and the service can be provided without human interaction with the service provider. Cloud computing enables the use of software without the need for purchasing a licence or installing it, and the user does not need to have strong qualifications in software or infrastructure management. Cloud computing can be classified into three models [27]:

- ***Software as a service (SaaS):*** The user can run the available software, but he cannot install new programs or configure the operating system.
- ***Platform as a service (PaaS):*** The user can install new programs, but he cannot act on the operating system.
- ***Infrastructure as a service (IaaS):*** The user is enabled to configure the infrastructure;

User can install new software, configure the operating system, the network, etc. At present, several companies offer cloud computing resources at competitive prices. Downloading vast amounts of data generated from calculations done in the cloud, however, is customarily relatively expensive.

VI. Accuracy And Efficiency

When performing scientific calculations, both software developers and software users should try to avoid some important issues related to methodology, which are commonly related to accuracy and execution time. We can call accuracy the similarity between the result of a given calculation and the hypothetical result that would be obtained if we were able to perform the same calculation without any numerical error. When calculating physical or chemical quantities, the accuracy is essential, because a lack of accuracy makes results unreliable. The accuracy can be lowered by many sources of error that exist for calculations performed in computers.

Apart from the accuracy, the other main limiting factor in computer simulation is the execution time. Nowadays, we do know equations which describe small scale phenomena quite well, but their solution for complex systems is cumbersome, and often unaffordable. The numerical complexity of the solution of simulation problems usually increases with the size or complexity of the system tackled. This numerical complexity can be measured with the number of required operations. Some examples of this can be

- If one wants to add N arbitrary numbers, then the number of operations required will be N -1.
- Solving a linear system of equations Ax = b, being A an N * N dense matrix, requires of the order of N^3 operations using the typical Gaussian elimination scheme.
- The simplest implementations of the Hartree-Fock method to find the ground state of the electronic Schrödinger Hamiltonian require a number of operations which is proportional to N^4, being N the number of basis functions used.
- A naive approach to calculate an estimation of the partition function of a system depending on N coordinates, and sampling m different values for each, takes of the order of m^N operations (exponential growth).

In all these examples the size of the system is proportional to a number N, and the increase of N leads to an increase of the numerical complexity of the solution of the problem. In doing any calculation, we want its result to be ready within a given time; systems beyond a given size will be unaffordable. This scaling of the methods can sometimes be reduced by doing a number of approximations consisting of neglecting part of the information involved in the problem and expecting it will not have a major influence on results [20].

The considerations about simulation time are more complex if parallel programs are run, instead of serial programs. Parallel programs distribute the workload in several computing threads, each of which is run in a different computing unit. When executing a parallel program, it is customary to measure its efficiency with

- The total execution time (wall clock time) which is required for a given task tNc (which is a function of the number of cores working on it, Nc).
- speedup, which is defined as the quotient $SN_c := t1=tN_c$ This is, the time that the task would last if run in one core divided by the time it lasts when run in N_c cores.

- The quotient S_{Nc} / N_c (sometimes called the efficiency factor)

For a given problem of constant size, Amdahl's Law [28] states that if p is the fraction of the problem which can be run in parallel, and therefore s: = (1-p) is the minimum fraction which must be run in serial, then the maximum speedup that can be achieved by using N_c cores is

$$S_{N_c}^{max} = \frac{N_c}{N_c(1-p)+p} . \quad (1)$$

This expression has an horizontal asymptote in $(1-p)^{-1}$. The speedup can be increased by increasing the total time required by the fraction of the problem which can be run in parallel, which can usually be achieved by increasing the size of the simulated system (for example, increasing its number of atoms). Commonly, p is not constant, but increasing as the size of the problem increases. Let us consider a variable-size problem which requires a time of $T(s + pN^a_c)$ to be solved in serial. In this expression, T is the total time required for solving the problem of a given size in serial, and the exponent α is a given positive number. If the part that can be parallelized is indeed parallelized (assuming optimal scaling in the $N_c$computing units), the time required by the execution in parallel will be $T(s + pN_c^{\alpha-1})$. Applying that s + p = 1 in the ratio of serial and parallel times for a variable-size problem for $0 < \alpha < 1$, becomes:

$$S_v := \frac{s + (1-s)N_c{}^{\alpha}}{s + (1-s)N_c^{\alpha-1}} , \quad (2)$$

where S v means speedup factor for variable size algorithms. If α = 0 (i.e., if the problem size does not increase with Nc) the expression above equals the Amdahl's Law (1). In the limit of high N_c, eq. (2) becomes

$$\mathbf{S_v = 1 + \frac{P}{S} N^{\alpha}} \quad (3)$$

In the case of α = 1 (linear scaling of the size of the problem with the number of computing units solving it), the ratio of serial and parallel time is

$$\mathbf{S_v^G = S + (1-S)N_c = (1-P) + PN_c} \quad (4)$$

for the ideal parallelization situation. Equation (4) is called the Gustafson's Law [8] and states that the speedup for solving a problem can be increased by increasing the size of its parallelizable part. Considerations such as the ones underlying Amdahl's and Gustafson's Laws can be useful for scientific software developers, in order to increase the efficiency of their codes. Parallelization characteristics of algorithms, however, are commonly much harder to derive than these laws.

References

[1] K. Dowd and C. Severance. *High Performance Computing*. O'Reilly & Associates, Inc., Sebastopol CA, 2nd edition, 1998.

[2] L. D. Fosdick, E. R. Jessup, C. J. C. Schauble, and G. Domik. *An Introduction to High-Performance Scientific Computing*. The MIT Press, Cambridge MA,1996.

[3] R. Gerber. *The Software Optimization Cookbook: High-performance Recipesfor the Intel Architecture*. Intel Press, United States, 2002.

[4] S. Goedecker and A. Hoisie. *Performance Optimization of Numerically Intensive Codes*. Society for Industrial and Applied Mathematics, Philadelphia PA, 2001.

[5] W. Triebel, J. Bissell, and R. Booth. *Programming Itanium-based Systems*.Intel Press, United States, 2001.

[6] K. R. Wadleigh and I. L. Crawford. *Software Optimization for High Performance Computing*. Prentice Hall PTR, New Jersey, 2000

[7] Hwang, K. *Advance Computer Architecture*: *Parallelism, Scalability, Programmability*, McGraw-Hill, New York, 1993.

[8] G. Hager and G. Wellein, *Introduction to High Performance Computing for Scientists and Engineers*, CRC Press - Taylor & Francis Group, 1st edition, 2011.

[9] J. L. Hennessy and D. A. Patterson, Computer Architecture, A Quantitative Approach, Morgan Kaufmann - Elsevier, San Mateo, CA, 5th edition, 2012.

[10] Microprocessor report 25,10 (2011) .

[11] W. A. de Jong, E. Bylaska, N. Govind, C. L. Janssen, K. Kowalski, T. Muller, I. M. B. Nielsen, H. J. J. van Dam, V. Veryazov, and R. Lindh, *Utilizing high performance computing for chemistry: parallel computational chemistry*, Phys. Chem. Chem. Phys. 12 (2010) 6896–6920.

[12] R. Wood, Future hard disk drive systems, *Journal of Magnetism and Magnetic Materials 321 (2009) 555 - 561, Current Perspectives: Perpendicular Recording.*

[13] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber, *Scientific data management in the coming decade*, SIGMOD Rec. 34 (2005) 34–41.

[14] J. Larus, *Spending Moore's dividend*, Commun. ACM 52 (2009) 62–69.

[15] M. N. Baibich, J. M. Broto, A. Fert, F. N. Van Dau, F. Petroff, P. Etienne, G. Creuzet, A. Friederich, and J. Chazelas, Giant Magnetoresistance of (001)Fe/(001)*Cr Magnetic Superlattices, Phys. Rev. Lett. 61 (1988) 2472– 2475.*

[16] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, in *A novel architecture of the 3D stacked MRAM L2 cache for CMPs in IEEE 15th International Symposium on High Performance Computer Architecture*, pp. 239–249, IEEE, 2009.

[17] Y. Jin and K. Li, *An optimal multimedia object allocation solution in multi power mode storage systems, Concurrency and Computation*: Practice and Experience 22 (2010) 1852–1873.

[18] R. Kumar, D. Tullsen, N. Jouppi, and P. Ranganathan, Heterogeneous chip multiprocessors, Computer 38 (2005) 32–38
[19] R. Kumar, D. Tullsen, N. Jouppi, and P. Ranganathan, Heterogeneous chip multiprocessors, Computer 38 (2005) 32–38.
[20] T. C. Germann, K. Kadau, and S. Swaminarayan, 369 Tflop/s molecular dynamics simulations on the petaflop hybrid supercomputer "Roadrunner", Concurrency and Computation: Practice and Experience 21 (2009) 2143–2159.
[21] Buttari, J. Dongarra, J. Kurzak, J. Langou, P. Luszczek, and S. Tomov,The impact of multicore on math software, in Proceedings of the 8th international conference on Applied parallel computing: state of the art in scientific computing, PARA'06, pp. 1–10, Berlin, Heidelberg, 2007, Springer-Verlag.
[22] B. Wilkinson, Grid Computing: Techniques and Applications, Chapman & Hall/CRC Computational Science, 1st edition, 2009.
[23] M.-A. Thyveetil, P. V. Coveney, H. C. Greenwell, and J. L. Suter, Computer Simulation Study of the Structural Stability and Materials Properties of DNAIntercalated Layered Double Hydroxides, J. Am. Chem. Soc. 130 (2008) 4742-4756.
[24] V. S. Pande, I. Baker, J. Chapman, S. P. Elmer, S. Khaliq, S. M. Larson, Y. M. Rhee, M. R. Shirts, C. D. Snow, E. J. Sorin, and B. Zagrovic, Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing, Biopolymers 68 (2003) 91–109.
[25] D. Kondo, B. Javadi, P.Malecot, F. Cappello, and D. P. Anderson, Cost-benefit analysis of Cloud Computing versus desktop grids, Parallel and Distributed Processing Symposium, International 0 (2009) 1-12.
[26] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, On the Use of Cloud Computing for Scientific Workflows, in eScience, 2008. eScience '08. IEEE Fourth International Conference on, pp. 640 –645, 2008.
[27] I. Foster, Y. Zhao, I. Raicu, and S. Lu, Cloud Computing and Grid Computing 360-Degree Compared, Grid Computing Environments Workshop, 2008. GCE'08 (2009) 1–10.
[28] G. M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in Proceedings of the April 18-20, 1967, spring joint computer conference, AFIPS '67 (Spring), pp. 483–485, New York, NY, USA, 1967, ACM.