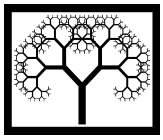
Paper 1



Proceedings of the Eighth International Conference
on Engineering Computational Technology,
B.H.V. Topping, (Editor),
Civil-Comp Press, Stirlingshire, Scotland

Future High Performance Computing Strategies

M.M. Resch
High Performance Computing Center Stuttgart (HLRS)
University of Stuttgart, Germany

Abstract

High performance computing (HPC) has seen a long history of progress over the last six decades. The prospects for an increase in performance over the coming decade are still very good. Systems with a performance in the range of Petaflops become widely available and discussions have started about how to achieve an Exaflop. This paper discusses the further development of high performance computing in the future. It proposes a strategy that shifts the focus of attention from hardware to software. With such a change of paradigm further progress in the field of simulation is more likely than with many of the concepts presented for Exascale computing.

Keywords: high performance computing, hardware-software, exaflop computing

1 Introduction

High performance computing has seen a long history of progress over the last six decades. For a long time Moore's assumption about the doubling of speed every 18 months [1] turned out to be right. When the increase in clock frequency started to level out, parallelism became the driving factor. Parallelism came so far in two waves. The first one started in the late 80s and by the early 90s created a number of interesting concepts. However, most of the advanced concepts with sometimes very large numbers of processors failed, and a number of companies disappeared from the high performance computing arena pretty fast.

The result of this first wave of parallelism was a number of systems that typically provided a moderate number of processors in a single system. For a while the largest systems were hovering around 1000 and up to 10000 processors or cores. The fastest system in the world in 1993 - as presented by the TOP 500 list [2] - was based on 1024 processors. The fastest system in 2003 was based on 5120 processors. The increase was a factor of about 5. We have to consider though that in 2003 the number

one system – the Earth Simulator - was using special fast vector processors. But even if we look at the top 10 systems of the list, we only find an increase in the level of parallelism of about 10. The parallelism of this kind was not easy to master but message passing was a good approach, and software programmers could easily keep track of their thousands of processes.

Since about 2003 we have seen a second phase of parallelism. This was partially driven by the IBM Blue/Gene project [3] which is using a larger number of slower processors. Over the last two to three years graphics processing units (GPUs) - with their hundreds of cores on one card - have further enhanced the trend. The currently valid TOP 500 list of November 2011 shows a system with more than 700.000 cores at the top. This is a factor of 140 over the last 8 years. Looking at the top 10 systems we see a factor of 10. This is comparable to the first phase but the pace is much faster now that we look at a period of 8 years only. For the next 2 years we expect to see about half a million up to a million cores as the average for the fastest 10 systems. So over the decade from 2003 to 2013 we expect to see a factor of 200 in parallelism for the top system and an average factor of between 100 and 200 for an average top 10 system. From a programmers point of view things are getting out of control.

In this paper we first look at the hardware available. We then investigate trends and possible pathways for a further increase in performance. After that we look at software approaches and provide hints of how to handle the huge amount of processes and threads. We finish with some conclusions.

2 Hardware Development

2.1 Current status

The current hardware landscape is dominated by a small number of components that are used by different architectures. With a few exceptions these standard components are integrated to form a large scale system. Speed is guaranteed by maximizing the number of available components. This helps to keep costs under control – since standard components are cheaper than specialized hardware - and provides the application programmer with a familiar hardware and software environment. In the following we look at the various components and try to estimate what might be seen in the future.

2.2 Looking back first

Before we do that, we have a look at some historical developments because this might give us an insight into what we can expect to come next. High performance computing grew out of the primeval soup of computing. Historical accounts [4] claim that Seymour Cray at a certain point in time decided he wanted to build the fastest systems in the world, rather than economically interesting ones. Technically

Seymour Cray was following some basic principles which we still have to consider today. For example the round shape of his first systems was owed to an attempt to reduce distances. The further success of his first company proofed that speed and economic success were possible at the same time. His failure with follow-on projects showed that things did change after a while. As long as the computer was a special purpose instrument for a limited number of users high performance computing was based on special purpose systems with special prices.

The advent of the personal computer started to change things. The computer became a ubiquitous instrument. Budgets for computer hardware were gradually moving from special purpose systems to general purpose hardware. Already in 1990 Eugene Brooks coined the word "Killermicros" [5], describing the end of what was considered to be "dinosaur" systems, and their replacement by microprocessor based systems. It did not take too long until Brooks was proven right. The number of systems using specialized processor technology started to dwindle away and by the year 2000 only small "game reserves" were left. By the year 2009 such specialized processors were virtually extinct when NEC announced its withdrawal from the Japanese Next Generation Supercomputing Project. However, recent announcements of NEC suggest that we will see another round of specialized processors in the future.

It is interesting to see that the killermicros did not only make an end to specialized processors in high performance computing. They also started to cannibalize each other. Over a period of about 10 years a number of processor architectures started to disappear from the TOP 500 list. While in the year 2000 the TOP 500 showed 5 different architectures with a substantial share of systems, that same list of the year 2011 shows that 90% of the systems are based on the x86-architecture.

The trend in microprocessor architectures was accompanied by a trend in system architectures. In 1994 Donald Becker and Thomas Sterling started what they called the "Beowulf Project" [6]. As a result of this project - that was building an high performance computing cluster from standard components – clusters became extremely popular in high performance computing. In the year 2000 expectations were high that future high performance computing systems would all be clusters based on "components off the shelf" (COTS). To some extent this has become true. About 80% of the systems listed in the TOP 500 list in November 2011 actually are clusters.

The situation as described already shows that a number of trends shape the landscape of high performance computing. After this short historical review we now have a look at the current situation and try to estimate the future developments in hardware.

2.2.1 Processors

The basis for the top systems in high performance computing is currently many-core processors. The number one system in 2011 – the Japanese K-Computer - relies on

the Fujitsu SPARC64 VIIIfx, a many-core processor with 8 cores [7]. More than 88.000 of these processors are bundled. Other large scale systems are based on the AMD Opteron 6200 processor with 16 cores. In both cases the clock frequency is comparably low. It is 2 GHz for the SPARC processor and 2.3 GHz for the AMD processor.

Another standard building block used in very large scale systems are general purpose graphics processing (GPGPU) units. Based on NVIDIA 2050 cards a high level of peak performance as well as of Linpack performance is made possible. The NVIDIA 2050 comes with 448 cores which again increases the number of cores for the user.

The background of this increase in number of cores is clear. The International Technology Roadmap for Semiconductors [8] indicates that what was suggested by G.E. Moore more than 50 years ago is still valid. The feature size is shrinking and it will keep doing so for a number of years to come. While we cannot increase the clock frequency anymore – basically because of the high leakage that comes with high clock frequencies – we still can substantially increase the number of transistors on a single chip. As a consequence, we are increasing the number of cores on a chip instead of shrinking the chip and increasing the frequency.

The SPARC VIIIfx and the NVIDIA 2050 find themselves on two ends of a spectrum defined in terms of complexity of cores and number of cores. The SPARC VIIIfx processor comes with a rather complex core design. Each of the cores could be described as “fat and fast”. The typical graphic cards like the NVIDIA 2050 come with a very large number of cores. These cores can be described as “slim and slow”.

Finally, we should mention a special purpose system from IBM. The IBM BlueGene is an architecture that is based on the IBM Power PC A2 processor. It comes with a relatively low clock frequency of 1.6 GHz and has 18 cores of which 16 are used for computing. The processor’s architecture is interesting in that it provides one extra core for running an operating system and one extra core as a spare core in case one of the 16 computing cores fails. These two strategies – operating system offloading and hardware support for fault tolerance - are increasingly becoming important. Unfortunately the Power PC A2 is only available in the Blue Gene system for high performance computing. Its market share will hence remain relatively small. It remains to be seen how this architecture will further evolve. Technically the processor can be considered to be closer to the “fat and fast” solution than to the “slim and slow” solution.

2.2.2 Networks

In the field of internal communication networks we have seen a variety of solutions in the past [10]. For a while several solutions were competing in the field of cluster computing. With the advent of Infiniband [11] the situation has changed. The new technology has practically replaced all other special solutions in the cluster market.

Of the 50 top systems in the TOP 500 23 are clusters based on Infiniband. The interesting finding is that 26 of the TOP 50 system are using some kind of proprietary network. Only one system is still based on Gigabit Ethernet (ranked number 42 in November 2011).

When we turn the pages of the TOP 500 list we find that starting around a ranking of 150 the number of Gigabit Ethernet installations substantially increases. This indicates that for high end systems Infiniband is a must. This is also supported when looking at the level of sustained Linpack performance in the list. The typical Gigabit Ethernet systems achieve about 50 to 60 % of peak performance for the Linpack benchmark. For an Infiniband system this ratio is typically in the range between 45 to 85 %. The big variation indicates that Infiniband is used to build a variety of network topologies.

2.2.3 Architectures

Looking at architectures we find that clusters dominate the TOP 500 list. About 80% of the fastest systems in the world are in that group. The rest of about 20% is based on an MPP architecture approach. Even though clusters are the biggest group we look at the MPP architectures. They seem to be outdated but keep a constant share of about 20% over the last 8 years, while other types of architectures have disappeared. What is more interesting: in terms of performance MPPs have a much larger share that is in the range of 40%. This is because MPPs can typically be found in the upper part of the TOP 500. So, when talking about real high performance computers we find that MPP and clusters are two competing technologies at equal footing.

One of the reasons for a renaissance of the MPPs is the IBM Blue Gene architecture. Originally the concept was based on a relatively light-weight processor. The new Blue Gene/Q has a relatively strong processor but comes with a lower clock frequency than comparable systems. The network is proprietary and provides a 5D-Torus.

In general, one of the main features of MPPs systems seems to be the better network connectivity. The basic performance numbers (latency and bandwidth) for MPI are typically comparable to what Infiniband can offer. However, the better network connectivity seems to increase the level of sustained performance. Analysing the fastest 50 systems in the world, we see that proprietary interconnects on the average achieve 78% of sustained performance for the Linpack benchmark. Infiniband based systems achieve about 74%. This is not a big difference.

A further investigation of the list shows that low sustained performance is caused by the usage of graphics cards. Such cards provide a high level of peak performance but typically do not work well in terms of sustained Linpack performance. The average sustained Linpack performance of the top 50 systems is 76%. The average of the 6 systems that make use of NVIDIA cards is 51% only. This is a clear indication that such systems do not show satisfactory sustained performance for the classical high performance computing applications.

What is further interesting is the evaluation of network architectures when we eliminate the NVIDIA results. Without these systems both Infiniband based systems and proprietary systems show an average of 80% of Linpack performance. The maximum for proprietary networks is 93%, for Infiniband it is 89%. The minimum for proprietary networks is 72%, and for Infiniband it is 59%. However, the relatively low minimum for Infiniband is the exception to the rule.

2.3 Potential paths forward

The last twenty years have shown that changes in high performance computing happen all the time and that predictions are difficult to make. However, there are a number of findings.

The number of cores in a high performance computer will further increase. There is currently no way of avoiding a situation in which millions of cores form the compute backbone of a high performance computer system. We may see solutions where the large number of cores is hidden from the user. However, this is most likely going to happen based on some kind of software solution. Most likely we are going to see compilers that support a high level of parallelism in a single node – whatever the term "node" is going to mean in the coming years.

Given the actual lack of advantage for proprietary networks one has to expect that Infiniband – or a follow-on technology – is going to gain more ground also in the top 50 list. Economic considerations might lead to an end of proprietary network development in much the same way that they have caused a substantial reduction in processor architectures available for high performance computing.

3 Software Strategies

Given the rapid increase in complexity, discussions about software development have spread across the world. One such activity is the International Exascale Software Project (IESP) [12] that brings together researchers from the US, Europa and Asia to discuss not only technical questions but also matters of funding with respect to software for Exascale software. The main issues discussed concerning software refer to:

- Co-design as a way to overcome the gap between hardware and software development
- Programming models and languages as a way to bridge the gap between hardware on the one hand and the software developer on the other hand
- Models to integrate hybrid architectures that use a mix of CPUs and GPUS or even accelerators.

In the following we discuss some of these issues and try to highlight the potential benefit of new concepts and the direction into which we are headed in Exascale computing software.

3.1 Co-Design

In the discussions about the development of Exascale systems co-design plays an important role. By developing software and hardware at the same time one expects to overcome the asynchronies of the two technical development paths. Theoretically this is a reasonable approach. However, the asynchronies remain. The basic fact is that changes in hardware come in steps of 2-4 years. This is about the time horizon for the renewal of a system and an upgrade of existing hardware technology. We are still living in the age of rapidly changing technology when it comes to computers.

Software changes happen at a different speed. A short investigation of system software and application software shows that both types of software follow very similar patterns. First, there is an idea or a basic algorithm. In a second step there is some prototype implementation. Over a time of 3-5 years the software starts to mature. A renewal cycle for basic software typically lasts for about 20 years. Looking at the history of operating systems one might assume even longer cycles. If we take UNIX and Linux together as being based on very similar fundamental ideas, one would say that the life span is in the range of 20-40 years. Similar time frames can be found for application software. Basic concepts get implemented in prototype software. Over time these packages mature and become available to a wider user community. Overall the process of maturing a software approach takes at least about twice as much time as the change in hardware architecture and potentially even longer.

It remains to be seen whether a co-design approach can speed up the software development process. It is certainly going to be beneficial if software developers are very early on involved in the hardware development process. On the other hand, operating system development at large hardware providers for a long time was done in-house such that an internal co-design was already taking place at high performance computing vendors. Nothing indicates that such an in-house co-design was able to overcome the gap between fast hardware development and slow software development.

3.2 Programming Models

For a long time parallel programming languages were considered to offer a solution to the problem of parallelization. With the advent of MPI and OpenMP these parallel programming languages started to lose ground. OpenMP quickly became the method of choice to program shared memory parallel systems. Given its simple constructs it became quite popular. However, with the increasing number of processors large scale systems turned into distributed memory systems.

In a sense the hardware architectures outgrew shared memory. The method of choice for such distributed memory systems was – and still is – MPI. The explicit exchange of messages is especially attractive for users that still fully understand the underlying algorithms and concepts of their parallel programs. However, increasingly the complexity of MPI implementations starts to limit users. In some cases large scale systems are unable to provide enough buffer space in order for MPI to effectively use thousands of processors.

One method of overcoming the problem of very large numbers of processes is to make use of the underlying architectural concepts. Most architectures are based on shared memory nodes. A typical system today might have several thousands of such nodes. Each of the nodes may have up to 32 cores. To avoid a very large number of MPI processes one may use a hybrid approach. For the 32 cores of each shared memory node a single MPI process is used. The parallelism of shared memory is exploited by using OpenMP directives. MPI communication is only used between the nodes. By using such a hybrid method the number of MPI processes can be kept relatively low even when a large number of cores are used. In the future the number of cores in a node may substantially increase and go up to thousands. It remains to be seen whether a hybrid approach can also handle such extremely fat nodes.

A new approach is the usage of Partitioned Global Address Space (PGAS) languages. Languages like co-array Fortran and UPC propose to hide the actual communication from the user and hence make programming much easier. By only extending existing languages like Fortran and C there might be a chance that these languages are adapted rather fast and that programming constructs become part of the underlying standard language. However, so far the take up is rather slow.

3.3 Compilers

When considering hybrid architectures we already saw that a combination of OpenMP and MPI might be a good solution for the programming challenge. However, handling two programming models in a single code is difficult. An option to avoid such an approach is a parallelizing compiler. The concept is well known and was applied very successfully in the Hitachi SR8000 system back in 2000. The compiler automatically parallelizes over a given number of cores that share their memory. MPI communication is done by one core only. The compiler gives full support for such a model.

The advantage of such an approach is that MPI codes can easily be reused. Furthermore, programmers do not have to mix OpenMP and MPI. Programming remains relatively easy. The potential disadvantage is that auto-parallelizing compilers may not be able to find an optimum solution for the parallelization problem for one shared-memory node. Practical experience for shared-memory parallelization using OpenMP showed that automatic parallelization is done well for up to 8 or 16 threads. Beyond that number it may be better to use explicit MPI programming – even on shared memory machines.

With modern processors we have to expect shared memory nodes with core numbers between 16 and 64 and with GPGPUs it may well be thousands of cores. So we may lose performance when using automatic parallelization through compilers. On the other hand, very large MPI codes – with one process per core instead of one process per node – may become increasingly inefficient when the number of cores used increases.

3.4 Algorithms

A prerequisite for sustained performance on large scale parallel systems is the suitability of algorithms for the underlying architecture. Over the last decades of high performance computing algorithms were based on the von-Neumann-architecture and only a few new concepts were developed that specifically tackle parallel architectures. With the number of cores growing to 100.000 and more this situation will have to change.

So, the ability to exploit a high level of parallelism depends highly on the underlying mathematical and computational algorithms. In [13] G.R. Liu describes desired features for algorithms for Exascale computing. He condenses his findings to four concepts that need to be considered.

- Extremely high parallelism
- Minimal data communication
- Locality
- Simplicity

3.4.1 Extremely high parallelism

Extremely high parallelism is obviously necessary. An application needs to be able to make use of a large number of cores. For many types of applications this should not be a problem. Typical finite element or finite volume codes use tens of millions or even billions of elements. For a system with 100.000 cores this implies that each core would work on several hundreds or even thousands of elements. The challenge is going to find a reasonable way to achieve load balancing of ever more complex meshes. If such a load balancing can be achieved there should be enough computational load for each core. Similar calculations can be done for particle based codes. As long as the number of particles is large enough very high levels of parallelism can be achieved. One consequence of the need for extremely well balanced loads of work is that mesh based methods will have to return back to simple meshes. Unstructured or even adaptive methods may turn out to be mathematically interesting but practically unfeasible for high performance computing.

A general finding would hence be that as long as we increase the size of the problem together with the number of cores, we may expect reasonable computational performance. The benefit of faster systems would be in being able to solve larger prob-

lems. On the other hand we have to see that we will not be able to speed up the solution of smaller problems.

3.4.2 Minimal data communication

Data communication is always costly in parallel systems. Driving the recommendation to the extreme one would therefore request: do not communicate.

There are two reasons why we want to minimize data communication. First, it is time consuming and hence reduces the sustained performance of an application. Second, it consumes a lot of electrical power. Reduction of data communication hence results in a substantial reduction of power consumption. Given that one of the key problems of future systems will be electrical power, the reduction of data communication is a key factor.

With respect to traditional algorithms two approaches could be followed. On the one hand, algorithms could become more loosely coupled. The key questions then are stability and convergence of the method. On the other hand, one could aim at replacing communication by computation. By increasing the computational efforts we could reduce the need for communication.

In any case, reducing data communication is a much tighter restriction from the point of view of algorithms than massive parallelism.

3.4.3 Locality

Locality partially is a corollary of the second requirement (avoid communication). But locality has two sides to it. On the one hand locality at the macro-level allows reducing data communication between processes. On the other hand locality has also to be seen at the micro-level. Data locality can help to keep data in cache and hence avoid costly access to main memory. The consequence of this finding is that we need to go back to traditional concepts of computing like vector computing. It is hence not surprising that modern processor architectures increasingly come with vector like hardware features.

On the other hand, this requirement helps us to understand that the basis for sustained performance in a large scale system is the performance that we can achieve on a single processor. Speedup does not have any meaning if the single processor performance is low. Hence, the first step in optimizing any parallel program is the optimization of the single processor performance.

3.4.4 Simplicity

Simplicity could be seen as a corollary of the first two requirements. The first benefit of simplicity is ease of compilation. Achieving a substantial level of sustained performance on the single processor will require for the compiler to better “under-

stand" the structure of the code. This requires the algorithms to be simple. On the other hand simplicity typically results in a simple replication of parallel tasks. Using such a simple replication allows to fulfil the first requirement of massive parallelism.

4 Conclusion

The further increase in performance in computing is driven exclusively by the concept of parallelism. Other concepts are not to be expected in the near future. From the point of view of an application programmer the hardware to be exploited is a typically very homogeneous set of hundreds of thousands or millions of cores.

The support for programming of such large scale systems is rather limited. Compilers can mainly exploit single processors. Automatic parallelization is only possible for a very limited number of processors. Concepts like OpenMP are limited in scope and cannot be extended to very large scale systems. Tools for automatic parallelization are also limited in scope.

Programming models and programing languages seem to provide the best approach for the programming of large scale systems. MPI is well established as a model. It requires improvement and potentially simplifications to be of practical use for a very large number of processors or cores. Partitioned Global Address Space languages (PGAS) show some potential to at least support parallel programming efforts. However, they are not yet fully integrated into the standards of Fortran and C. It seems to be necessary to follow up on these developments and evaluate how far they can get us in real world applications.

Algorithms and models are typically not well suited for very large scale systems. For a number of traditional applications the outlook is not very positive. Especially in the field of engineering traditional methods are challenged by the low speed of communication on the one hand and the high level of parallelism on the other hand. An increase in number of elements is not always a solution to the problem. Size of problem is not in every case interesting for the end user. Increasingly we are confronted with applications that are sufficiently fine discretized but would require a higher sustained performance in order to get solutions within a reasonable timeframe.

In summary the prospects for high performance computing simulations are mixed. We can expect to see a further increase in performance. This will allow us to tackle new research questions and improve existing concepts. In order to harness the full potential of new hardware we need to rethink our algorithms. Both mathematical methods and computational approaches will have to be changed to be successful in the new era of high performance computing.

References

[1] G.E. Moore, Cramming more components onto integrated circuits, Electronics, 38(8), 114-117, 1965.

[2] TOP 500 List www.top500.org (10.3.2012)

[3] Blue/Gene

[4] Charles J. Murray, The Supermen – The Story of Seymour Cray and the Technical Wizards behind the Supercomputer, John Wiley & Sons, 1997

[5] http://www.websters-online-dictionary.org/definitions/KILLER+MICRO (10.3.2012)

[6] T. Sterling, D. Savarese, B. Fryxell, K. Olson, D.J. Becker, Communication Overhead for Space Science Applications on the Beowulf Parallel Workstation, Proceedings of High Performance Distributed Computing (HPDC-4), 1995

[7] Takumi Maruyama, Tsuyoshi Motokurumada, Kuniki Morita, Naozumi Aoki, Past, Present, and Future of SPARC64 Processors, FUJITSU Sci. Tech. J., Vol. 47, No. 2, pp. 130 – 135 (April 2011)

[8] ITRS, International Technology Roadmap for Semiconductors 2011 Edition http://www.itrs.net/Links/2011ITRS/Home2011.htm

[9] IBM Corporation, IBM System Blue Gene/Q, IBM Corporation, 2011

[10] M.M. Resch, "Trends in Architectures and Methods for High Performance Computing Simulation", in B.H.V. Topping, P. Iványi, (Editors), "Parallel, Distributed and Grid Computing for Engineering", Saxe-Coburg Publications, Stirlingshire, UK, Chapter 3, pp 37-48, 2009. doi:10.4203/csets.21.3

[11] http://www.infinibandta.org/ (10.3.2012)

[12] www.exascale.org (10.3.2012)

[13] G. R. Liu, On Future Computational Methods for Exascale Computer, iacm expressions, 30, 8 – 10, December 2011