An accelerated Cluster-Architecture for the Exascale

N. Eicker and Th. Lippert

Jülich Supercomputing Centre, Forschungszentrum Jülich, 52425 Jülich Germany, and
Department C, Bergische Universität Wuppertal, 42097 Wuppertal, Germany

Abstract: Clusters are dominating high-performance computing (HPC) today. The success of this architecture is based on the fact that it profits from the improvements provided by main-stream computing well known under the label of Moore's Law. But trying to get to Exascale within this decade might require additional endeavors beyond surfing this technology wave. In order to find a possible direction we review Amdahl's and Gustafson's thoughts on scalability. Based on this analysis we propose an advance architecture combining a Cluster with a so called Booster element comprising of accelerators interconnected by a high-performance fabric. We argue that this architecture provides significant advantages compared to today's accelerated Clusters and might pave the way for Clusters into the era of Exascale computing.

1 Introduction

Clusters are the dominant architecture in HPC. Their big success is based on the ability to profit from advances in main-stream computing. Due to their modular setup it is easy to replace single components improving the overall performance of such systems. For instance, every new generation of CPUs providing more compute-power might replace their predecessor without the necessity to change the high-speed fabric or the complex software-stack at the same time.

Having Petascale system in production today the next challenge is to reach Exascale by the end of the decade. This goal will introduce new difficulties. E.g. most probably using general-purpose CPU will no longer be competitive with more specialized solutions like GPGPUs[1]. Unfortunately, the way GPGPUs are employed today will not lead to scalable architectures. Thus, it will become necessary to review the idea of the Cluster-architecture in HPC in order to make it successful in the future.

To get a better idea in which direction Clusters have to be developed it is necessary to re-investigate the requirements set by application potentially capable to make use of Exascale systems. We will argue that such application will have more than one level of scalability. On the one hand there are highly-scalable parallel kernels having simple communication patterns. On the other hand less-scalable kernels will limit the overall scalability of the application. Typically, the latter will show much more complex communication patterns.

Based on this analysis we will propose a novel HPC-architecture combining a classical Cluster system with a so called Booster system comprising of accelerator-nodes and a

[1] General-purpose Graphical Processing Units – in fact these are highly optimized floating-point units.

high-speed interconnect having a less flexible but very scalable torus-topology. The basic idea is to use the Booster for the highly-scalable kernels of potential Exascale applications. The Cluster part of the system will be reserved for the less-scalable and more complex kernels.

This paper is organized as follows: In the next section we explore the space of HPC-architectures by means of analyzing results taken from the TOP500 list. With that we identify the challenges arising from the goal to reach Exascale by the end of the decade. In section 4 general ideas concerning scalability in HPC are revised. Section 5 presents our proposal for a next generation Cluster system. In the last section we conclude our thoughts and give an outlook on future plans.

2 HPC-Architectures in the TOP500 list

Having collected almost 18 years of statistics the TOP500 list[2] [1] is a powerful tool in order to identify trends in computer-architectures. it shows – starting with very first examples in the late 90's – today Clusters are the dominant architecture in HPC. Their stage was set by commodity processors that became sufficiently powerful in respect to floating-point-performance (e.g. Compaq's Alpha, AMD's Opteron or Intel's Xeon processors), by powerful networks that enabled the connection of such commodity nodes (at that time MyriNet and Quadrics, later Gigabit Ethernet and now superseded by InfiniBand) and last but not least the availability of a complete OpenSource software-stack.

As much as HPC is dominated by Clusters today the prevailing operating-system (OS) in the field is Linux. In fact the big success of Clusters would have been unlikely without the availability of an OpenSource OS. Only the availability of an free and open OS enabled the possibility to adapt this lowest layer of software to the specialized requirements of HPC as including the support dedicated hardware required by Clusters – e.g. low-latency / high-throughput networks. At the same time it cut the software costs significantly.

Distributed memory systems like Clusters require a convenient programming paradigm. Therefore, the availability of free, high-quality implementations of the MPI standard was crucial, too. Without them Clusters would not have been usable for practical applications at that time. Here as well the availability of source-code enabled the community to implement the necessary adaptions required by every hardware-innovation in this field.

Since then Cluster architectures took over more than 80% of the market leaving the remainder for MPP systems[3]. This incredible success is caused by the ability to benefit from the enhancements of computer-technology in general. One has to face the fact that this development is not mainly driven by a relatively small market like HPC. Instead, it is pushed by main-stream markets like enterprise-servers, personal computing and gaming. Combined with the ever increasing costs of technology development – designing a new generation of processors requires investments in the range of billions of €– competing

[2] This paper refers to the November 2010 version of this list.

[3] These systems are represented today by IBM's BlueGene systems and the Cray XT-series. One might argue that the latter are Cluster-systems, too.

HPC-architectures were only able to survive in very special niche-markets like the very high end of HPC.

A most crucial feature of HPC-architectures is not the highest performance of a single component but the balance of all building-blocks in the system. E.g. most HPC-systems don't rely on the highest performing CPU available at a given time but instead use a compromise that is more balanced to the available memory- and network-bandwidths, shows higher energy-efficient or optimizes the ratio of price vs. performance.

Another important result derived from the TOP500-list is the development of the available compute-power. Combined with results of historical HPC-systems it shows that HPC-systems gain a factor of 1000 in performance per decade. It is important to face the fact that this development outperforms "Moore's Law", i.e. the observation that semiconductor technology doubles the number of transistors per unit area every 18 month [2]. Thus, in order to achieve this outcome a significant increase of the systems' parallelism beyond sheer accretion of transistors per chip was necessary in HPC. This reflects in the observation that basically all current HPC-architecture are massively parallel.

3 The Exascale (Cluster)-Challenge

Having systems at Petascale today, the community thinks about the next step, i.e. having Exascale systems (10^{18} floating-point operations per second) by the end of the decade. An analysis done by a group of experts two years ago showed [3] that an Exascale-systems using an updated version of today's technology and concepts would run into severe trouble:

- **Power consumption** Projecting today's architectures and concepts onto technology of 2018 shows that an Exascale system would require several 100 MW of power. Of course, it is not acceptable to use a common nuclear power-plant exclusively for running a supercomputer.
- **Resiliency** Projecting the trend of ever increasing number of components in HPC-systems to the end of the decade[4] will lead to millions of components. Combined with today's components' mean-time to failure (MTTF) this will make Exascale systems unusable since the whole system's MTTF will drop into the range of hours or even minutes.
- **Memory hierarchies and I/O** The increasing gap between the development of compute-performance and bandwidth of both, memory and storage, will require additional layers in the memory hierarchy of Exascale systems like higher-level CPU-caches or flash-memory as disk-caches.
- **Concurrency** The growth of levels of parallelism makes it harder for the users to exploit this systems. Additionally, it introduces new requirements on the scalability of Exascale applications as we discuss in the next section.

[4] Already JSC's Petaflop-system JUGENE based on BlueGene/P technology has about 72,000 nodes.

Thus, in order to achieve the ambitious goal of Exascale by the end of the decade one has to hunt for new concepts.

Having this challenges in mind the ability of Clusters to reach Exascale has to be reviewed. In particular, the question has to be raised, if the central concept of Clusters – the utilization of commodity CPUs – is competitive compared to proprietary developments for HPC.

Regarding such competitiveness the yardstick in the next years is set by IBM's efforts leading to the BlueGene/Q system. An analysis of preliminary results [4] create reasonable doubt that commodity CPUs will be sufficient in the future. This is due to their limitations on energy-efficiency and the superior ratio of price vs. performance of the BlueGene/P technology. Both originate in the fact that commodity CPUs carry too much dead-freight required for their general-purpose targets compared to their floating-point capabilities that are the key-capabilities in the field of HPC.

Since commodity processors will not be sufficient one has to strive for a new working-horse in HPC systems. A good candidate are GPGPUs providing an order of magnitude higher floating-point performance compared to commodity CPUs today at the cost of limited usability. They represent the current end-point of a long development of accelerators that might be used as co-processors enhancing the capabilities of general purpose processors significantly.

In fact, 40% of the top 10 systems in the current TOP500 list are already equipped with accelerator-cards. Nevertheless, the dominating architecture of accelerated Clusters[5] suffers from severe limitation concerning ballance and, thus, scalability. This is due to the fact that the accelerators are neither directly connected to the Cluster node's main memory nor capable to directly send or receive data from their local memory. In the end, data is forced to be copied back and forth between main-memory and the memory residing on the accelerator card using up the scarce resource of bandwidth on the node's system bus. At the same time latency for doing communication between the actual compute elements – that is the accelerator – is significantly increased due to the required detour through main-memory.

A good way out of this dilemmas would be to have a Cluster of accelerators. In this concept the node is consisting of an accelerator only – accompanied by some memory and a high-speed interconnect – saving the commodity CPU. In particular, programs running on the accelerator-cores shall be capable to initiate communication operations without the support of some commodity processor. A good example for this concept is the QPACE system [6] that was ranked #1 of the GREEN500 list [5] of the most energy-efficient HPC-systems in the world.

Nevertheless, this concept has limitations, too. Besides the problem of finding accelerators that are capable to run autonomously they might be not flexible enough to drive a high-speed interconnect efficiently. Furthermore, the gain introduced by the direct connection between the compute-element and the interconnect-fabric might be wasted by the fact that accelerators – as highly dedicated devices – suffer when running general purpose codes.

[5] We will distinguish accelerated Clusters, i.e. classical Clusters comprising of nodes equipped with accelerators, and Clusters of accelerators.

Thus, a radically new concept is required for Clusters systems that shall be competitive at Exascale.

4 Considerations on Scalability

Talking about Exascale it is crucial to discuss the effects of parallelism on scalability. A first result on this field is known as Amdahl's Law [7] stating that the scalability of a parallel program is inherently limited by the sequential part of this application.

Assume that the run-time of the program's part that is parallelizable is given by $p \cdot t$ on a single processor. Thus, the sequential part will take $(1-p)t$. If this program is executed on a computer providing N processors, the amount of time taken by the parallelizable part reduces to $p \cdot t/N$. By definition, the run-time of the sequential part will not be reduced at all by additional processors. The speedup S of the program on a parallel machine is given by the ratio of the execution-time on this machine T_N compared to the run-time on a serial machine T_S, i.e.:

$$S = \frac{T_S}{T_N} = \frac{(1-p)+p}{(1-p)+\frac{p}{N}} = \frac{1}{(1-p)+\frac{p}{N}} \tag{1}$$

In practice this means that the parallel speedup is limited by the sequential part of a given program. E.g. for a program with a sequential part of just 1% the speedup will be limited to 100, even if throwing in $N = \infty$ processors. Having said that the fact that some applications are able to scale on BlueGene-systems with $\mathcal{O}(100000)$ processors shows that these seem to have a vanishing sequential part.

In fact, the behavior of such parallel applications is better described by Gustafson's Law [8]. In Amdahl's Law it is assumed that putting a program on a larger machine will leave the problem to solve untouched. In reality this is not the case but usually the problem-size is scaled with the capabilities of the machine. E.g. if a scientist wants to solve a problem in molecular-dynamics, the availability of a machine twice as fast will typically not lead to solving the same problem in half the time but to tackle a problem twice as big in the same time or to do a more detailed simulation that requires to double the amount of operations in the same time . This is mostly due to the fact that simulations are not able to handle realistic systems today. Instead, it is required to make a compromise by reducing the size of a simulation or neglecting details expensive to calculate.

Therefore, in Gustafson's Law the amount of work done on a system N times more capable than the serial one is $w = (1-p) + pN$. Overall this leads to a practical speedup of

$$S = \frac{(1-p)+Np}{(1-p)+p} = (1-p)+Np \tag{2}$$

assuming that executing the sequential part will take the same amount of time on the parallel system as on the serial one. It is obvious that for a large number of processors the speedup is dominated by N.

Of course, Gustafson is only right, if it is possible to implement the parallel part of the problem in a scalable way. In practice there are several caveats. E.g. any collective communication operation in a parallel system like a global sum or a broadcast will inherently introduce costs beyond $\mathcal{O}(1)$ and is, thus, not scalable. Additionally it turns out to be expensive to build really scalable fabrics of high-speed interconnects comprising full flexibility on communication-patterns at reasonable prices[6]. All this might limit the scalability of the parallel portion of an application significantly.

Thus, a different viewpoint to the question on how to reach Exascale might be taken by looking at applications and their inherent scalability. Since HPC-systems undoubtedly will be massively parallel by the end of the decade with an even higher degree of parallelism than today, application's scalability will play a crucial role in order to exploit the computational power of such systems. Analyzing JSC's application portfolios one finds basically two classes of applications.

1. Highly-scalable codes using regular communication patterns. These are the codes that are able to exploit JSC's BlueGene/P system.
2. Less-scalable but significantly more complex codes. Most often these codes require complicated communication patterns. Their requirements constrain these applications to Clusters.

A more detailed view on the second class unveils that among them are several applications that have highly scalable kernels, too. I.e. in principle they should be able to also exploit BlueGene-type of machines. Nevertheless, in analogy to the serial work in Amdahl's Law their scalability is limited by the least scalable kernel.

In respect to Exascale systems it would be highly desirable to run applications of the second class on such systems. In fact, due to the higher degree of parallelism in such systems, even codes still in the first class today might be shifted into the second class. Because of that this less scalable class is expected to grow in the future. A further growth is expected due to the trend that more and more aspects of a given scientific question are included into corresponding simulations. This will make codes more complex and, thus, might limit scalability. Last but not least problems completely out of range today due to their complexity might become feasible at the availability of Exascale-systems. Most probably they are expected to fall into the second class, too.

5 Proposal for a future Cluster-Architecture

Putting all together a future Cluster-Architecture requires different ingredients in order to provide enough flexibility to run complex applications at the Exascale. Our concept as sketched in figure 1 foresees a Cluster element comprising of nodes (CN) connected by a highly flexible switched network labeled as InfiniBand here. It is accompanied by a so-called Booster element. The Booster element is build out of Booster nodes (BN). Each

[6] Full crossbar-switches today are limited to several dozens ports; the trick to use fat-tree topologies introduces an overhead of $\mathcal{O}(p^2)$ for p ports [9].

BN hosts an accelerator capable to run its own operating-system autonomously. The BNs are interconnected by a highly scalable torus-network sketched as red lines in the figure. Booster Interfaces (BI) connect the Booster to the Cluster's fabric.

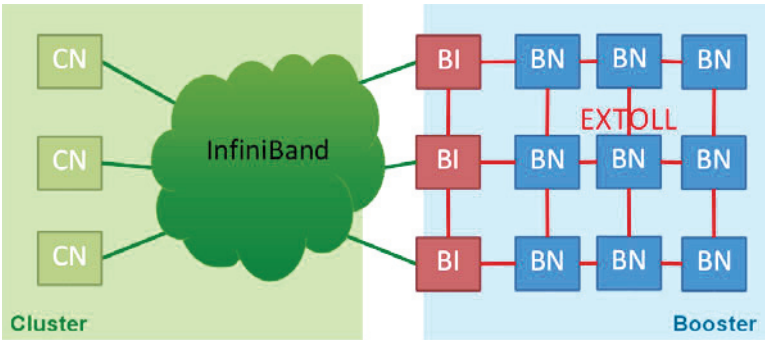

Figure 1: Sketch of the proposed architecture. For simplicity, the Booster's 3D-torus is depicted as a 2D mesh.

The proposed architecture provides several advantages:

- The Cluster element allows to run complex kernels. There is no restriction to run parallel kernels with complicated communication-patterns or highly irregular codes as on todays highly-scalable MPP machines.
- The Booster element is able to run highly-scalable kernels in the most energy-efficient way. The limitations of today's accelerated Clusters are avoided by enabling the compute-elements – i.e. the accelerators in the BNs – to do direct communication to remote BNs.
- It cannot be expected that the ratio between the amount of work to be executed on the commodity CPU and the accelerator is fixed between different applications or even between different kernels of one applications. The proposed architecture supports this fact by virtualizing the accelerator. The accelerator is no longer part of the CN and assigning CNs and BNs might be done dynamically.
- At the same time the dynamical assignment of CNs and BNs improves resiliency. Faulty accelerator do not affect CNs and vice versa.

In addition to described advantages, the architecture supports the user in employing the high degree of parallelism in future machines. Today the type of kernels to be offloaded onto an accelerators is very limited. E.g. accelerators typically do not allow to do communication within this kernels efficiently. In contrast, the proposed architecture allows more complex kernels to be offloaded to a Booster. These kernels might even include communication, as long as the patterns are regular enough and don't swamp the torus topology.

In this context the Booster might be seen as highly-scalable system on its own. Thinking of the highly-scalable codes that are able to exploit BlueGene or QPACE today, it should be possible to run them on the Booster alone.

In the other extreme BNs might be assigned to single CNs and used in the same fashion as in today's accelerated Clusters. Both use-cases – and anything in between – are possible without modification of the hardware concept but are just implemented by means of configuration on the level of system- and application software.

To reflect all this features, the architecture was named dynamical Exascale evaluation platform (DEEP).

5.1 Coping with Amdahl's Law

The Cluster Booster Architecture offloads kernels with high scalability ($\mathcal{O}(N)$ concurrency) onto the Booster while leaving kernels with limited scalability ($\mathcal{O}(k)$ concurrency) on the Cluster element. Let's assume a highly idealized situation where the code fraction of the $\mathcal{O}(N)$ concurrency is H and the fraction of the $\mathcal{O}(k)$ concurrency is $1 - H$. Given a number h of cores on the Booster and a number of f cores on the Cluster, and assuming the Cluster's multi-core units to be a factor c times faster than the Booster's many-core units, Amdahl's Law predicts the speedup S of the DEEP Cluster Booster Architecture:

$$S = \frac{1}{\frac{1-H}{fc} + \frac{H}{h}} \tag{3}$$

For a 10-PFlop/s Booster with $h = 500000$, a Cluster with $f = 10000$, an effective performance difference of a factor of 4 between multi-core and many-core units and a favorable concurrent $\mathcal{O}(N)$ code fraction of $H = 0.95$, the speedup S would reach a value of 320000 as compared with a single core on the Booster. This corresponds to an effectiveness of 64%. It is evident that the efficiency strongly depends on the value of H. Since H is approaching 1 in a weak scaling scenario as advocated by Gustafson one can be optimistic to evade severe performance degradation that would be expected for $\mathcal{O}(k)$ concurrent kernels on the Booster. Note that the communication between cluster and Booster was not taken into account in this consideration. The optimal balance between h and f is specific for a given application. The DEEP architectural concept allows adjusting this balance dynamically. The recipe to achieve scalability of the DEEP grand challenge applications is to maximize H in the breakdown of the code into kernels while achieving data traffic as minimal as possible or even hiding the Cluster Booster communication behind computation.

In summary, we expect that the proposed Cluster Booster Architecture will demonstrate that it can not only reach the Exaflop/s level but – more important – that applications can achieve unprecedented speed-ups and increases in performance.

5.2 The Cluster-Booster Software-stack

In order to benefit from the Booster element of the DEEP architecture the user is forced to identify highly-scalable kernels capable to be offloaded. In a way this is similar to the

identification of kernels to be offloaded into the GPGPUs of accelerated Clusters today. Nevertheless, for the Cluster-Booster architecture this should be easier since the Booster provides more flexibility through its internal communication capabilities.

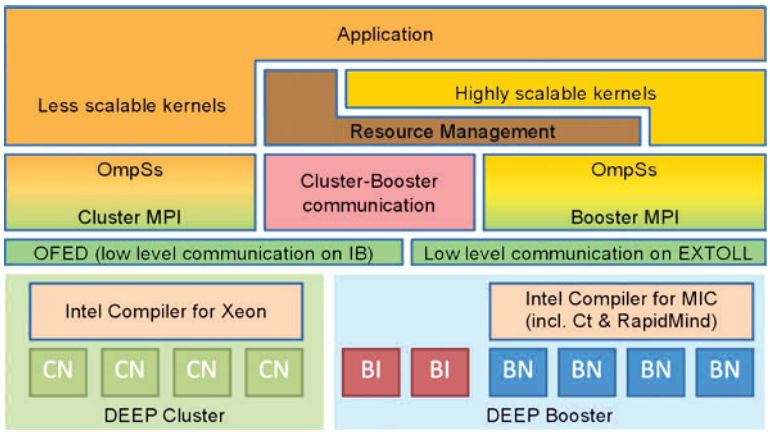

Figure 2: High-level view of the Cluster-Booster software-stack.

A Cluster-Booster Software-stack as sketched in figure 2 will support the user in expressing the different levels of parallelism of the problem. The application's kernels of limited scalability remain on the Cluster part and make use of a standard software-stack comprising of standard compilers, MPI, etc. Highly scalable kernels will be offloaded to the Booster with the help of a resource-management layer. For that, a Cluster-Booster communication layer is required in order to bridge the gap between the flexible Cluster interconnect and the highly scalable Booster network.

In order to implement the highly scalable kernels on the Booster an additional software stack is required there. Since the Booster might be seen as a Cluster of accelerators it is no surprise to find here a software-stack similar to the one on the Cluster. Besides native compilers supporting the processors of the Booster nodes a communication library is required. For simplicity MPI is chosen here, too. Of course in contrast to the Cluster-MPI the Booster-MPI has to support the Booster interconnect.

The application itself is formed by both types of kernels running on the Cluster and Booster parts of the overall system glued together by the resource management layer.

6 Conclusions and Outlook

We presented a novel HPC-architecture extending the concept of classical Clusters and newer developments of accelerated Clusters, i.e. Cluster systems with nodes equipped with GPGPUs besides the commodity processors. Our concept includes a Cluster that is accompanied by a so called Booster comprising of nodes of accelerators and a high-speed interconnect with torus-topology. We argue that our concept provides the additional

flexibility and scalability that is required to pave Cluster-Architectures the way into the future of Exascale.

We plan to implement a prototype of this architecture using Intel's MIC (Many Integrated Cores) architecture [10] for the Booster nodes and the EXTOLL interconnect [11] for the Booster fabric.

References

[1] http://www.top500.org

[2] Gordon E. Moore, "Cramming more components onto integrated circuits.", Electronics. 19, Nr. 3, 1965, pp. 114-117.

[3] http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/exascale_final_report_100208.pdf

[4] http://www.theregister.co.uk/2010/11/22/ibm_blue_gene_q_super

[5] http://www.green500.org

[6] H. Baier et al., "QPACE: power-efficient parallel architecture based on IBM PowerXCell 8i", Computer Science - R&D 25 (2010), pp. 149-154. doi:10.1007/s00450-010-0122-4.

[7] Gene Amdahl (1967), "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities" (PDF), AFIPS Conference Proceedings (30), pp. 483-485.

[8] John L. Gustafson, "Re-evaluating Amdahl's Law", Communications of the ACM 31(5), 1988, pp. 532-533.

[9] Charles Clos, "A Study of Non-blocking Switching Networks", The Bell System Technical Journal, 1953, vol. 32, no. 2, pp. 406-424

[10] http://www.intel.com/pressroom/archive/releases/20100531comp.htm

[11] Mondrian Nssle et al., "A resource optimized remote-memory-access architecture for low-latency communication", The 38th International Conference on Parallel Processing (ICPP-2009), September 22-25, Vienna, Austria.