Indirect cube topology for small- and medium-scale clusters

Javier Navaridas
Dep. de Arquitectura y Tecnología de Computadores
Universidad del País Vasco - UPV/EHU
P. Manuel de Lardizabal 1, 20080 San Sebastián
javier.navaridas@ehu.es

José Miguel-Alonso
Dep. de Arquitectura y Tecnología de Computadores
Universidad del País Vasco - UPV/EHU
P. Manuel de Lardizabal 1, 20080 San Sebastián
j.miguel@ehu.es

Abstract
Interconnection networks arranged as *k*-ary *n*-trees or spines are widely used to build high-performance computing clusters. Current blade-based technology allows the integration of the first level of the network together with the compute elements. The remaining network stages require dedicated rack space. In most systems one or several racks house the upper network stages, separated from the compute elements. This incurs in additional costs, which are significant for small systems. In this paper we evaluate an alternative arrangement that connects elements in a cube-like topology. This organization requires only the use of the switches that are integrated within the compute elements. We explore a wide variety of system scales, ranging from 120 to 7680 compute nodes, in order to find out to which size the proposed topology can scale. Results of the experiments suggest that the proposed arrangement is not viable for large-scale systems, but show interesting advantages in small- and medium-scale clusters.

1. Introduction

The development of *off-the-shelf*, standardized high performance networking technologies (such as Myrinet [5], Gigabit Ethernet [6]or InfiniBand [10]) has made viable the utilization of clusters of computers as high performance computing systems. In fact, the widespread utilization of different scales of such systems, has favored the research and development of a bunch of hardware and software technologies which have made the construction of compute clusters more affordable, moving from handcrafted clusters to perfectly integrated, ready-to-work systems that include a large collection of software tools that support centralized management. The development of such technologies, in turn, have favored an even broader utilization of HPC clusters which, in fact, have become the preferred way to build high performance computing systems. For instance, looking at the current Top500 list [8], we can see that only a few listed systems are built using *ad-hoc* supercomputing technologies (Cray's XT families [2], NEC's earth simulator [12], IBM's BlueGene families [13] and IBM's ASC [14]), while most of the systems in the lists are different instances of clusters.

The most common networking technologies used to interconnect *state-of-the-art* clusters are InfiniBand and Gigabit Ethernet, being the former a high performance alternative and the later a cost-effective choice. At any rate, compute clusters tend to be interconnected using multistage, fat-tree based topologies [20]. This class of indirect networks offer high bandwidth, low latency communications and have some desirable properties that can be exploited both by constructors and users: good scalability, high path diversity, routing simplicity (deadlock-freedom), high resilience to failures, low disposition to congestion, etc. Other indirect topologies such as the full-Clos, implemented in the TACC Ranger [26] are less commonly utilized.

As far as we know, cube-like topologies, inherited from massively parallel processing systems, have been rarely used to build super-clusters. A significant exception is the 11D hypercube implemented in the NAS Pleiades [19]. SCI [23], a similar technology whose specification considered the construction of direct networks, did not gain enough market acceptance.

There are two main reasons to justify this neglect of direct networks from the cluster community. Firstly, these topologies are deadlock-prone and, therefore, their use would require adding deadlock avoidance mechanisms into the switching elements or, alternatively, a careful selection of the routing scheme. Secondly, cube-like topologies do not scale as well as multistage topologies do, as will be discussed later on.

The main proposal of this paper is as follows. Let us assume that a cluster *building block* is composed of a collection of compute nodes in an enclosure. This integrates a network switch that connects all those nodes and have some external links. These links are normally used to build a tree-like network, connecting them to an upper-stage switch. Instead of doing that, we propose to connect the building blocks directly among them, without intermediate switches, in the form of a 2D or 3D cube. We call this an indirect cube because it does not connect nodes directly; nodes are connected to switches that form a cube. Note that external links have to be arranged in blocks, using one or more links per dimension – direction: X+, X-, etc. In the experiments below, 2D tori with 4 parallel links per direction were used.

In this paper we explore the feasibility of using networks composed of standard switches arranged as 2D torus topologies, showing how their performance does scale, when compared to a tree-like topology. The remaining of this paper is structured as follows. Section 2 discusses the reasons that motivate the utilization of cube-like topologies to interconnect clusters. Section 3 gives detailed explanation of the simulation-based evaluation work carried out: systems and workloads. Section 4 shows and discusses the obtained results. The paper is closed in Section 5 with the main conclusions of this work.

2. Motivation

We can find two main motivations for arranging the *off-the-shelf* switches commonly used in clusters and superclusters as an indirect cube. The most compelling one is reducing the cost of the interconnection network; additionally, we can take advantage of the vast research work carried out for this kind of topologies in order to build a system with good performance levels.

Massively Parallel Processing systems have been commonly built around cubes (or meshes) and, therefore, there is plenty of previous work on how to exploit the characteristics of these topologies. There are also well-known *best practices* to use them. Furthermore, lots of parallel scientific applications are implemented bearing in mind that they are going to be executed over a network adopting these topologies. In fact, the implementation of a broad range of math operations is straightforward using cube-like

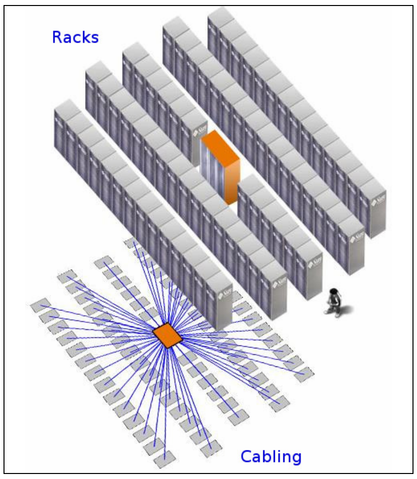

Fig. 1. Floor plan of a Sun Constellation system.

virtual topologies, as can be seen in [3] and [4]. Briefly, the kind of interconnection networks proposed in this paper can be considered as a *hybridization* of direct and indirect networks.

Regarding the costs, it is remarkable that the number of network components (switching elements and links) of a cube-like topology increases in $O(N)$, while in a tree-like topology it increases in $O(N \log N)$, being N the number of nodes to be interconnected. Basically, the number of network components in an indirect cube is the same as the number of network components in the first stage of a comparable tree. In other words, the savings obtained are all the extra links, switches and racks required to form and house the remaining stages of the network. To illustrate this fact, reader can see in Fig. 1 the floor plan of a large-scale Sun Constellation cluster, in which the racks containing the interconnection network are shown in red. Implementing an indirect cube would allow removing these interconnection racks, with the subsequent savings in terms of space, power consumption and heat dissipation.

Note that current blade server technologies allow the integration of network switches inside the blade enclosure. Looking at the web pages of the top server manufacturing companies, we can found several examples of blade enclosures that include slots for communications: Dell [7], HP [9], IBM [11] and SuperMicro [25]. If first-stage switches are integrated with the compute nodes, our building blocks could be extra-dense enclosures that can be connected directly.

Note that the cost overheads of purchasing extra racks to form a multistage topology would become more noticeable for small systems. If we assume that we always fill a rack with servers, we would need to purchase an extra rack just to house the switches required to build the network. An extreme scenario would be purchasing a rack to store a single switch.

Building the proposed network with the above-mentioned blade enclosures would be extremely easy, because it would not require any external switch (therefore, no extra racks). Also, cabling would be much simpler, compared to the tree. Looking again at Fig. 1 we can see that the racks storing the interconnection are placed roughly in the center of the floor plan, in order to keep the links as short as possible. Still, those racks located in the borders of the room will require links of, at least, a few tens of meters. In the case of an indirect cube, the racks can be connected to their physical neighbors, which will require links of a few meters at most. Note also that shorter links means faster transmission speeds and reduced latencies – although, as we assume the utilization of standardized technologies commonly based on optical links, we will not consider these advantages. We do consider, though, the latencies derived from switching times.

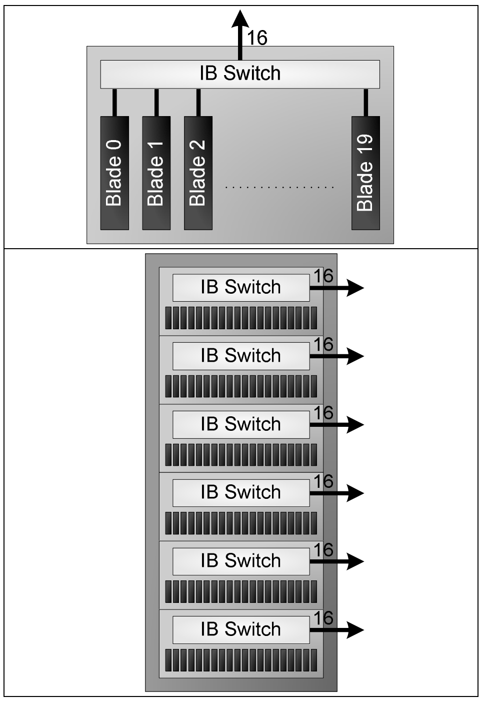

Fig. 2. Schematic depiction of a TwinBlade enclosure (up) and a rack storing 6 of them (down).

3. Experimental Set-Up

Our in-house developed interconnection network simulation and evaluation environment, in short INSEE [22], was the workbench used to evaluate the feasibility and scalability of the indirect cube network. Bearing in mind current technologies to build high performance clusters, we have modeled the systems basing on the TwinBlade servers offered by SuperMicro [25] because of their compute density, the highest we found. This technology allows integrating 20 compute nodes in 7U enclosures. Each enclosure can also house a 36-port QDR Infiniband switch, with 20 ports devoted to connect the compute nodes via a backplane, plus 16 external output ports. A regular rack (42U) can include six TwinBlade enclosures – that already include the first stage of the interconnection. A depiction of a blade enclosures and a rack are shown in Fig. 2.

Our study includes systems composed of small to medium-scale clusters, assuming always the utilization of complete racks: from a single rack (120 compute nodes) to 64 racks (7680 compute nodes). We focus on how the performance of the indirect cube scales in comparison with that of the kind of tree topologies implemented in *state-of-the-art* clusters.

To keep things simple, we have modeled systems using switches with a fixed number of ports. We have selected 36 ports, to emulate the switches included in the TwinBlade enclosures by Supermicro. With these switches we built a thin-tree topology using the 20 internal ports to connect with lower stage of the tree and 16 ports to connect to the upper one. Note that the performance of such topology will be very close to the performance of a full-fledged fat-tree because the *trimming* is not very aggressive [18]. With the number of racks included in our set-up this topology will have in most cases three stages. The only exceptions are the systems composed by 1, 2 and 3 racks, which will fit in a 2-stage tree. Examples of the 2-stage and the 3-stage trees are depicted in Fig. 3

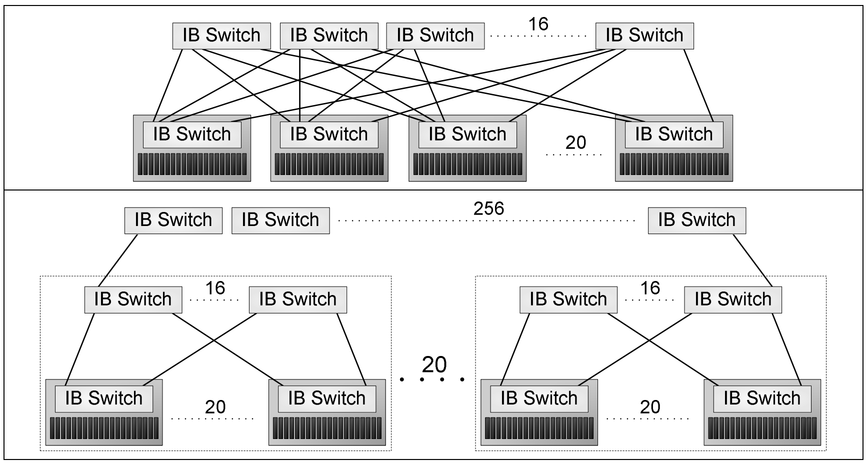

Fig. 3. Examples of the trees used in our experiments: 2 stages (up) and 3-stages (down). Most elements are hidden for the sake of clarity.

In the case of the indirect cube, as explained before, only the first stage of switches (those inside the TwinBlade enclosures) is needed to interconnect all the compute nodes. It will be unnecessary to add extra switches or racks to house them. Taking into account the characteristics of the switches, we will build 2D tori using 4 parallel links in each dimension, as depicted in Fig. 4.

Networking components are modeled as follows. The switching strategy is virtual cut-through with packets composed of 8 phits (physical units) each. Each input port is split into two virtual channels each of them having associated a queue able to store up to 4 packets. A shortest path credit-based adaptive routing scheme has been used in both the tree and the indirect cube topologies. In this scheme each input port sends the number of free slots in its queue to the neighboring output ports. Packets are routed through the feasible output port with more credit. Note that credits are transmitted *out-of-band* and, therefore, do not interfere with regular traffic of the applications.

In the case of the tree, adaptivity means that packets can adapt in the upward route, but the downward route is always static (*destination* mod *k*). On the other hand, the indirect cube adaptivity allows changing the direction and also the parallel link in the same dimension. Note that in an actual implementation, the four parallel links could be aggregated and used as a single connection working at 4 times the speed of a regular link. A

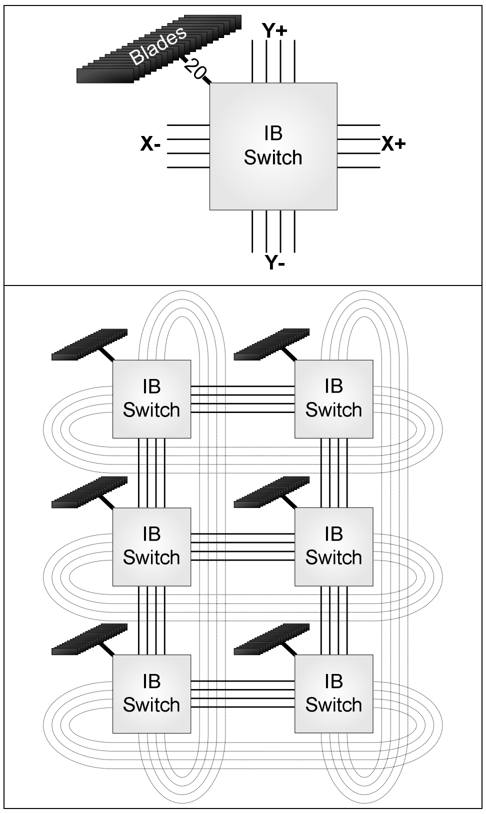

Fig. 4. Port arrangement of the switch included in a TwinBlade enclosure to form an indirect cube (up). The indirect cube constructed with the six enclosures within a single rack (down).

bubble scheme [21] is added in order to guarantee deadlock freedom. Given that the topology is composed of rings, when the number of racks increases, the network becomes congestion-prone. For this reason, we will measure the performance of the topology after including a simple congestion control mechanism that gives priority to the traffic already inside the network [15] (therefore, penalizing new injections from nodes). This mechanism is similar to the one used in the BlueGene/L family of supercomputers [1]. For the sake of completeness we will measure the performance of the indirect cube topology with this mechanism activated and deactivated. We want to remark that using adaptive routing schemes allows taking the best of each topology, which in turn allows fairly comparing them.

Given the wide variety of network sizes to evaluate, the use of network traffic taken from actual application traces is not feasible in our environment. For this reason we have used a collection of synthetically generated workloads that resemble the way in which scientific applications communicate. Most of these workloads are explained and justified in [16] and [17]. We proceed with a brief description of the workloads used in this paper.

In *All-to-All*, all the nodes send a message to all the other nodes, starting from their next one. After sending all the messages, each node waits for the reception of all the messages addressed to it. The message size is 1 Kbyte.

Binary Tree is an optimized (logarithmic) implementation of a reduce operation in which the reduction is made in several steps, each of them halving the number of messages. The message size is 10 Kbytes.

Butterfly is an optimized (logarithmic) implementation of an all-reduce operation. Each node interchanges messages only with a subset of the nodes, instead of with all of them. The message size is 10 Kbytes.

In *Near Neighbor*, the nodes are arranged in a virtual 2D torus. Each node sends a message to each of its neighbors following a dimension order (X+, X-, Y+ and finally Y-) and then waits for the reception of its neighbors' messages. The message size is 10 Kbytes.

In *n-Bodies*, the nodes are arranged in a virtual ring. Each node starts a chain of messages that travel clockwise across half of the ring. When the chain finishes, the node at the other side of the ring sends a message to the source node. The message size is 10 Kbytes. This workload is only defined for an odd number of nodes. For this reason one of the nodes of the network will not take part in the execution of this workload. More details about this workload can be found at [23].

In *Random*, multiple waves of messages are generated in such a way that all the messages of a wave must be received before start sending the messages in the next wave. The source and destination of the messages are selected randomly following a uniform distribution. The wave length, W, is a parameter and affects the causality of the workload. In this evaluation 40000 messages of 10 Kbytes each are generated using three different values for W: 1, 200 and 40000.

All workloads were generated to evaluate a wide variety of systems composed of 1 to 64 racks (120 to 7680 compute nodes). The only exceptions are All-to-All and n-Bodies because their formulation is quadratic and could only be scaled to systems composed by up to 16 racks.

All-to-All, Butterfly, Near Neighbor and all the Random cases can be considered *heavy* workloads, as they will stress the network with high contention and congestion scenarios. In contrast, Binary Tree and n-Bodies can be considered *light* because of their high levels of causality, which limit the formation of persistent contention for the utilization of network resources. At any rate, the workloads only include data interchanges (communication), and therefore the obtained results should be understood as a *worst-case* study because when adding computing intervals, the execution speed differences among topologies will be diluted.

Regarding the task-to-node placement policy, in the trees it will be *consecutive*, meaning that the nodes will be occupied from left to right. In the case of the indirect cubes, the 20 nodes connected to a switch will be conveniently placed as a 5×4 sub-mesh to keep the resultant indirect cube as close to a square as possible in all configurations. Note that, as we will see later, this placement will favor the communication pattern of the Near Neighbor workload but, in exchange, it will be detrimental to the performance for other workloads. In the case of the Random workloads, as the underlying pattern is uniform, the placement will barely affect the overall performance.

4. Analysis of Results

The execution time for the different systems and workloads are plotted in Fig. 5. For the sake of clarity the results of each configuration have been normalized to the time required by the tree topology. Therefore, the reported value for the tree will be always 1. A lower value represents a faster dispatching of the messages. The tree is represented by the line denoted as tree, while the indirect cube is represented by the lines denoted as icube and icube+cc, being the later the case in which congestion control is activated.

As we could expect given the difference in the amount of available network resources, in most cases the execution time required by the indirect cube is noticeably longer than that required by the tree. The only exceptions are the Binary tree and the Near Neighbor workloads. The former has such a low-demanding communication pattern that most networks can handle it easily. Regarding the latter, the placement selection for the indirect cube allowed for a perfect match between the virtual topology of the workload and the actual network topology, which resulted in a near-optimal behavior.

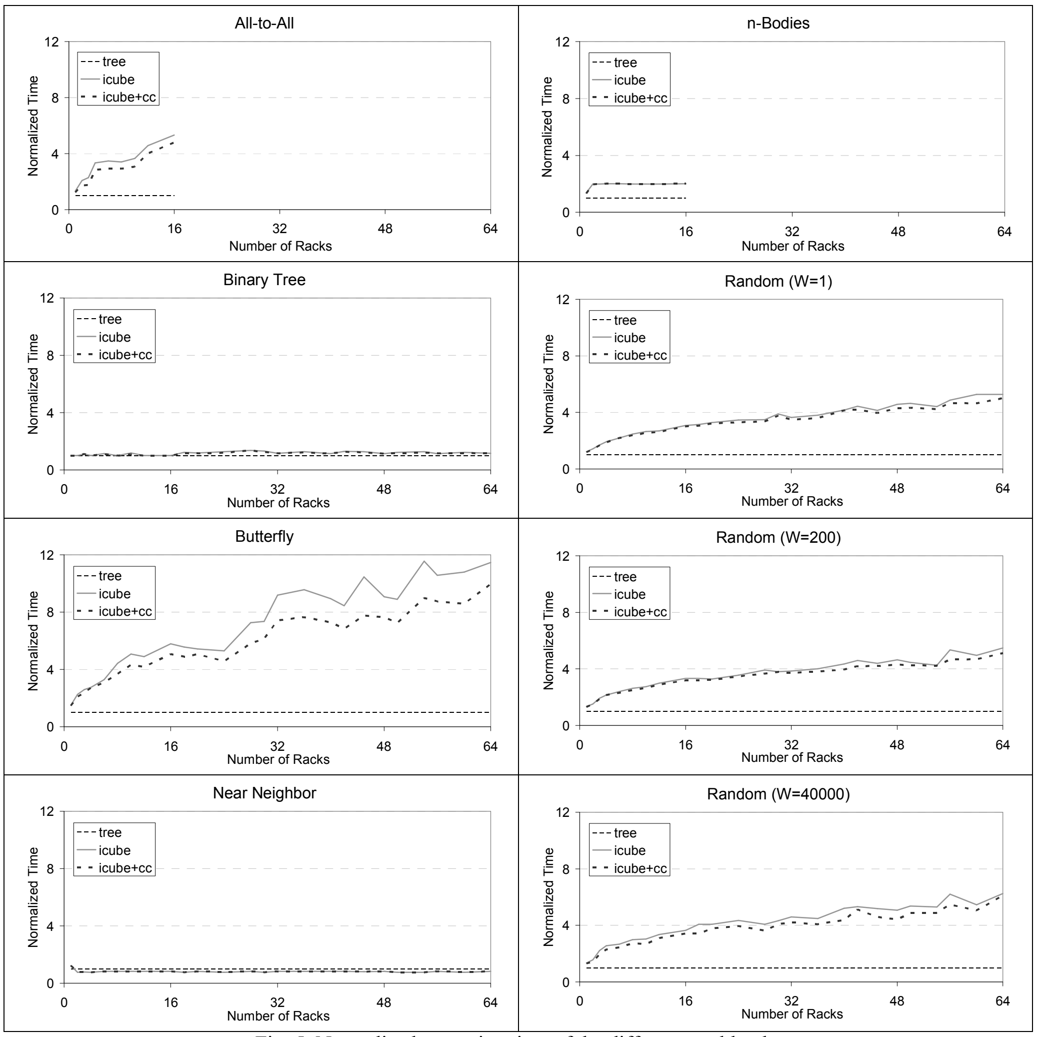

Fig. 5. Normalized execution time of the different workloads.

It is also noticeable that in those cases where the tree outperforms the indirect cube, the larger the system is the worse are the results of the cube. This is explained by the lower scalability of cube-like topologies when compared by trees: $O(\sqrt[D]{N})$ versus $O(\log N)$ in terms of distance, being N the number of nodes of the topology and D the number of the dimensions of the cube, 2 in our set-up. The only exception to this rule is the *n*-Bodies workload, in which the difference between topologies seems to be independent of the system scale; a constant (approx. 2). This is because the high causality of the workload, composed by large chains of dependencies, does not allow fully exploiting the network resources of a tree. In fact, the results for this workload are affected more by the placement than by the topology itself.

The reader should note that, for those workloads for which the indirect cube can not compete with the tree (All-to-All, Butterfly and the Random workloads), the activation of the congestion control mechanism is noticeably effective to boost the execution time of the workload. In the remaining cases, the congestion control does not affect the execution speed at all, neither positively or negatively.

We can conclude that the indirect cube topology can be a good choice for small- and medium-scale systems, especially for those that would require extra racks to store only a few switches. As shown by our results, scaling up the system may lead to the network becoming an important performance bottleneck, especially for those workloads that tend to stress the interconnection network.

To close the section, authors want to insist in that, while in most cases the indirect cube performs worse than the tree, it offers in exchange clear savings in terms of network complexity. These savings manifest in the form of fewer switches and racks, with the subsequent diminution in the requirements of space, power consumption and heat dissipation. Furthermore the number of links and their length is also reduced, which should lead to faster system deployment and simpler system management.

5. Conclusions

In this paper we have proposed arranging indirect networks, such as those used in current clusters and superclusters, in the form of cube-like topologies somewhat similar to those traditionally used in supercomputing systems. A thorough motivation was discussed focusing basically in reducing cost and complexity of deployment, even if performance is somewhat affected.

The indirect cube was compared to a tree-like topology, the *de facto* standard in the composition of superclusters. Results of our simulation-based experimental work show that the indirect cube network is not as good as the tree when handling communication-intensive workloads. However, adding a simple, local congestion control mechanism was enough to boost performance with these workloads. On the other hand, the performance of indirect cubes with less intense workloads (for example, those with long chains of dependencies) is comparable of that of the tree.

It is remarkable that an underlying cube-like topology can be effectively exploited by applications, by means of arrangements of processes using virtual meshes and tori, which exploit locality in communications. In our set-up, the indirect cube topology experienced an improvement of roughly 20% in comparison to the tree when dealing with this kind of workloads.

The overall conclusion of this paper is that, while clusters arranged as an indirect cube are competitive in small- and medium-scale configurations, they should not be used in large-scale systems because of their limited scalability.

Still, it would be possible to scale up from a small system arranged as an indirect cube to a large system interconnected by a tree-like topology, just by adding more racks (compute and networking). This is a viable option because the networking technology is the same as that used in regular clusters.

Anyway, more research needs to be performed in order to provide routing schemes that support efficient deadlock-free static routing for indirect networks arranged forming this topology.

Acknowledgements

This work has been supported by the Ministry of Education and Science (Spain), grant TIN2007-68023-C02-02, and by grant IT-242-07 from the Basque Government. Dr. Navaridas is supported by a post-doctoral grant of the University of the Basque Country UPV/EHU.

References

[1] NR Adiga et al. "Blue Gene/L torus interconnection network." IBM Journal of Research and Development 49 (2/3), 2005.

[2] R Alam et al. "Cray XT4: An Early Evaluation for Petascale Scientific Simulation". Procs. of the ACM/IEEE Conf. on Supercomputing, Reno, NE, USA, 10-16 Nov., 2007.

[3] Y Aoyama, J Nakano. "RS/6000 SP: Practical MPI Programming". IBM Red Books SG24-5380-00. Available at: http://www.redbooks.ibm.com/redbooks/pdfs/sg245380.pdf

[4] E Barszcz, et al. "Solution of Regular, Sparse Triangular Linear Systems on Vector and Distributed-Memory Multiprocessors". Technical Report NAS RNR-93-007, NASA Ames Research Center, April 1993.

[5] NJ Boden et al. "Myrinet: A Gigabit-per-second Local Area Network," IEEE Micro, 15 (1), February 1995, pp. 29-36.

[6] DG Cunningham, WG Lane. "Gigabit Ethernet Networking". Macmillan Publishing Co. Inc., Indianapolis, IN, USA. 1999.

[7] Dell. "PowerEdge Blade Servers". Available at: http://www.dell.com/us/en/enterprise/servers/blade/ct.aspx?refid=blade&s=biz&~ck=anav&cs=555

[8] JJ Dongarra et al. "Top500 Supercomputer sites". Available at: http://www.top500.org/

[9] Hewlett-Packard. "Blade Servers and Blade Systems". Available at: http://h18004.www1.hp.com/products/blades/bladesystem/index.html

[10] InfiniBand Trade Association. "InfiniBand Architecture Specification". Vol. 1, r1.0.a.

[11] International Business Machines. "IBM BladeCenter Hardware". Available at: http://www-03.ibm.com/systems/bladecenter/hardware/

[12] Japan Agency for Marine-Earth Science and Technology. "EARTH SIMULATOR". Available at: http://www.jamstec.go.jp/es/en/es1/system/index.html

[13] G Lakner, GP Sosa. "Evolution of the IBM System Blue Gene Solution". IBM Red Books REDP-4247-00. Available at: http://www.redbooks.ibm.com/redpapers/pdfs/redp4247.pdf

[14] Lawrence Livermore National Laboratory. "ASC Purple". Available at https://asc.llnl.gov/computing_resources/purple/

[15] J Miguel-Alonso et al. "Improving the Performance of Large Interconnection Networks using Congestion-Control Mechanisms". Intl. Journal on Performance Evaluation, 65 (2008), pp. 203–211.

[16] J Navaridas, J Miguel-Alonso. “Realistic Evaluation of Interconnection Networks Using Synthetic Traffic”. 8th Intl Symposium on Parallel and Distributed Computing. June 30 to July 4 2009. Lisbon, Portugal.

[17] J Navaridas et al. "On synthesizing workloads emulating MPI applications". The 9th IEEE Intl. Workshop on Parallel and Distributed Scientific and Engineering Computing, 2008, Miami, FL, USA.

[18] J Navaridas et al. “Reducing Complexity in Tree-like Computer Interconnection Networks”. Parallel Computing 36 (2-3), 2010, pp. 71-85.

[19] NASA Advanced Supercomputing (NAS) Division. "NAS Computing Resources - Pleiades Supercomputer". Available at: http://www.nas.nasa.gov/Resources/Systems/pleiades.html

[20] F Petrini and M Vanneschi. “k-ary n-trees: High Performance Networks for Massively Parallel Architectures”. Procs. of the 11th Intl Parallel Processing Symposium, Geneva, Switzerland, 1-5 April, 1997, pp. 87-93.

[21] V Puente et al. "The adaptive bubble router". Journal of Parallel and Distributed Computing 61 (9), 2001, pp. 1180-1208.

[22] FJ Ridruejo, J Miguel-Alonso. "INSEE: an Interconnection Network Simulation and Evaluation Environment". Lecture Notes in Computer Science, Volume 3648 / 2005 (Proc. Euro-Par 2005), pp. 1014-1023.

[23] “Scalable Coherent Interface”, ANSI/IEEE Standard 1596-1992, IEEE Service Center, Piscataway, New Jersey, 1993.

[24] CL Seitz. "The cosmic cube". Comms. of the ACM, 28 (1), 1985, pp. 22-33.

[25] SuperMicro Computer, Inc. "Products | SuperBlade". Available at: http://www.supermicro.com/products/Superblade/TwinBlade/

[26] Texas Advanced Computing Center. "HPC Systems - Sun Constellation Linux Cluster: Ranger". Available at: http://www.tacc.utexas.edu/resources/hpc/#ranger