Indirect cube: a power-efficient topology for compute clusters

Javier Navaridas
School of Computer Science
The University of Manchester
Oxford Road,
Manchester M13 9PL, UK.
javier.navaridas@cs.man.ac.uk

José Miguel-Alonso
Computer Architecture and Technology Dept.
The University of the Basque Country UPV/EHU
P. Manuel de Lardizabal 1,
20018 San Sebastián, Spain
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 significant additional costs, especially if a rack containing only a few switches has to be added.

In this paper we propose and evaluate the performance and power-consumption of an alternative arrangement that connects elements in a cube-like topology. Building an indirect cube topology requires only the use of the switches that are integrated within the compute elements and also simplifies deployment. 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 while keeping adequate performance levels and low power demands. An additional advantage of our proposal is that the same equipment can be reused to form a tree-based topology if a performance boost is needed.

Keywords—*Direct Topologies, High Performance Computing Clusters, Indirect Topologies, Interconnection Networks, Performance Evaluation, Power-Efficiency, Simulation.*

1. Introduction

The development of off-the-shelf, standardized high performance networking technologies (such as Gigabit Ethernet [1], Myrinet [2] or InfiniBand [3]) has made viable the utilization of clusters of computers as high performance computing systems. The widespread utilization of different scales of such systems has favoured the research and development of a bunch of hardware and software technologies which have made the construction of compute clusters more affordable. The community has witnessed a movement from handcrafted clusters to perfectly integrated, *ready-to-work* systems which include a large collection of software tools that allow centralized management. The development of such technologies, in turn, have favoured an even broader utilization of computing clusters which, in fact, have become the preferred way to build high performance computing systems. For instance, looking at the current Top500 list [4], we can see that only a few of the listed systems are built using ad-hoc supercomputing technologies (Cray's XT families [5], NEC's earth simulator [6], IBM's Blue Gene families [7] and IBM's ASC [8]), while most supercomputers are actually different instances of clusters.

Compute clusters are often arranged as multistage topologies, usually based on the fat-tree topology [9]. Other indirect topologies such as the full-Clos implemented in the TACC Ranger [10] are less commonly utilized. The main rationale behind the selection of tree-like indirect networks is that they offer some desirable properties that can be exploited by constructors and users: high bandwidth, low latency, good scalability, high path diversity, routing simplicity (deadlock-freedom), fault resilience, low disposition to congestion, etc.

As far as we know, cube-like topologies, inherited from massively parallel processing systems, have been rarely used to build superclusters, being the NAS Pleiades (11D hypercube) a significant exception [11]. There are two main reasons to justify this neglect of direct networks from the cluster community. On the one hand, 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 [12]. On the other hand, cube-like topologies do not scale as well as multistage topologies do.

The most common networking technologies currently used to interconnect state-of-the-art clusters are InfiniBand, 10Gigabit Ethernet and Gigabit Ethernet. The latter is a cost-effective alternative to build the network. The former two are high performance choices which have support for optical links and offer much higher bandwidths and lower latencies associated to data transmission. SCI-based approaches [13], whose specification had explicit support for the implementation of direct networks, did not gain enough market acceptance. Myrinet [2], which was one of the pioneering cluster-friendly networking technologies, has suffered a similar fate.

In this paper we explore the feasibility of using networks composed of standard switches arranged as 2D torus topologies, showing their scalability when compared to tree-like topologies both in terms of performance and power dissipation. 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. Sections 4 and 5 show and discuss the obtained results focusing respectively on performance and power-consumption. The paper is closed in Section 6 with the main conclusions of this work.

2. Motivation

The main proposal of this paper can be summarized as follows. Let us assume that a cluster *building block* is composed of a collection of compute nodes within an enclosure. Integrated within this enclosure there is also a network switch connected to all the internal nodes, usually through a back-plane, and have external links that allow connecting the enclosures forming a network. These external links are normally used to build a tree-like network, connecting them to an upper-stage switch. Instead of doing that, we propose connecting building blocks directly, 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-, Y+, Y-, etc. Therefore, for a 3D cube, each building block requires at least six external links.

This building block model is possible because 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 find several examples of blade enclosures that include slots for communications: IBM [14], HP [15], Dell [16], and SuperMicro [17]. The latter offers the so-called TwinBlade servers which will be the technology considered throughout this paper because of their compute density, the highest we found. TwinBlade servers allow integrating, in a 7U enclosure, 20 compute nodes plus a 36-port QDR InfiniBand switch. Six of these enclosures fit into a regular rack (42U). Therefore, in a single rack we can fit 120 nodes *and* the first stage of the interconnection network. Depictions of the blade enclosures considered through this paper and a rack containing six of them are shown in Fig. 1. We propose connecting these extra-dense blade enclosures directly among them, which simplifies deployment because external switches are not required (therefore, we do not need extra racks). Wiring is simpler too, compared to the trees, as we will see below. Briefly, the indirect cube topology proposed in this paper can be seen as a *hybridization* of direct and indirect networks.

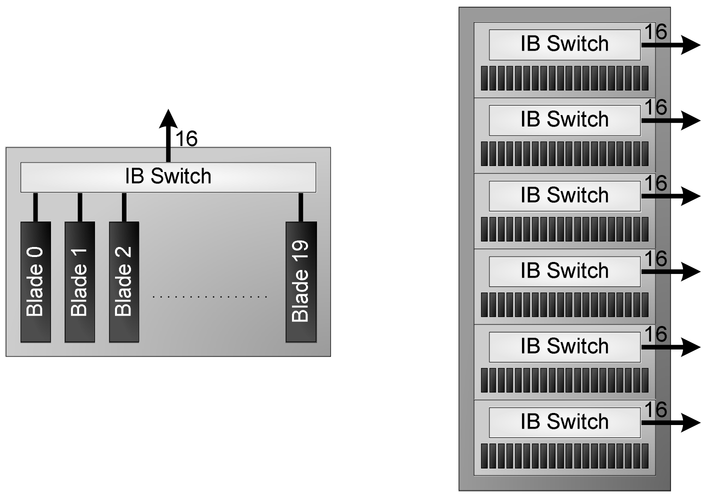

Fig. 1. A TwinBlade enclosure (left) and a rack storing 6 of them (right).

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 as well as its impact in terms of power consumption and heat dissipation. Additionally, we can take advantage of the vast research work carried out for direct 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 a large body of knowledge on how to exploit the characteristics of these topologies. There are also well-known *best practices* to use them. Furthermore, many 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 virtual topologies, as can be seen in [18] and [19].

Regarding 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 any 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, we show in Fig. 2 the floor plan of a large-scale Sun Constellation cluster, in which the racks containing the interconnection network are located in the middle of the figure. 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 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 tree-like network. In Fig. 2 we can also notice that the racks housing the core of the interconnection network are placed roughly in the centre of the floor plan in order to minimise link length. Still, those racks located in the borders of the room will require links of, at least, several tens of meters. In the case of an indirect cube, there are lots of wiring among nodes inside the same rack, and only a few wires are connected to closely located racks if the topology is folded [12]. All the connections, consequently, will require links of a few meters at most.

Fig. 3 reveals the differences in wiring among the tree and the indirect cube. In the tree, 96 links (6 enclosures with 16 output ports each) have to be connected from each rack to the switches in the network core which is located in the centre of Fig. 3a. Note that in the figure the network core is depicted as a single rack, although in a real configuration more than one could be needed, as we will see in the next section. Note also that the distance between the racks and the switches would increase as the system grows. In the indirect cube most of the connections will be between elements in the same rack (56 out of 96 – 28 links). Only 40 links will be connected to elements in external racks. If the cube is not folded (Fig. 3b), neighbouring nodes will be connected among them except for wrap-around links. Note that the length of the wrap-around links will increase with the system size. Finally, link length will be always one or two if a folded torus is constructed (Fig. 3c).

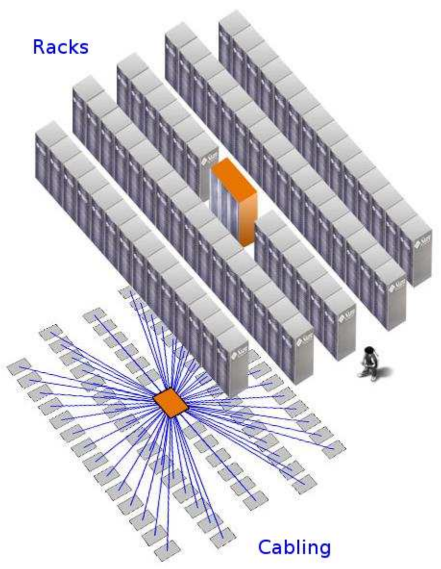

Fig. 2. Floor plan of a Sun Constellation system.

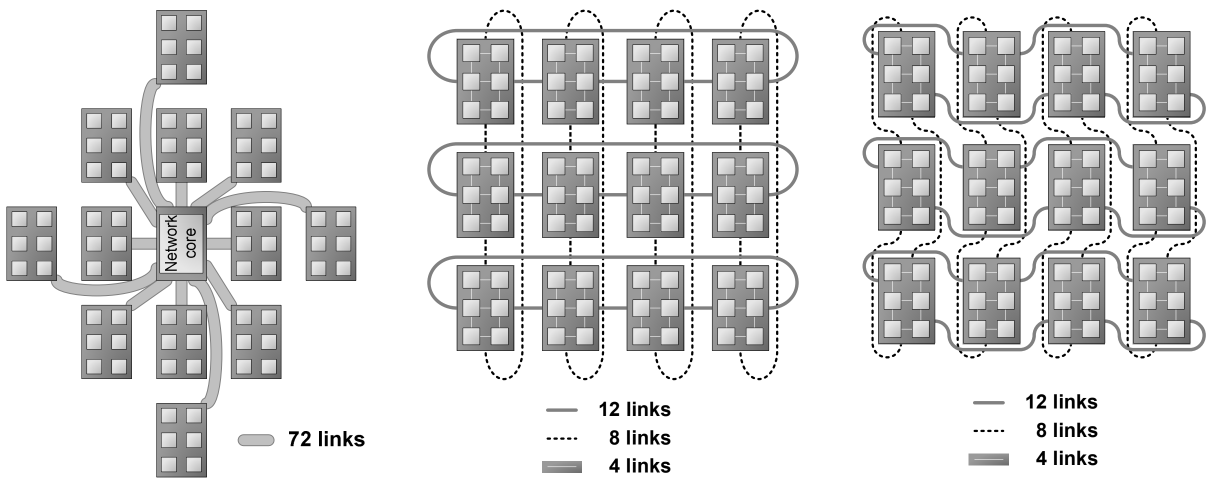

Fig. 3. Example of the wiring of the different arrangements for a 12-rack system.
a) Tree. b) Unfolded indirect cube. c) Folded indirect cube.

We want to remark that shorter links means faster transmission speeds and reduced latencies. However, as we assume the utilization of standardized technologies commonly based on optical links, we will not consider these advantages in our experimental set-up. We will consider, though, latencies derived from switching times.

We conclude this section with TABLE I, which shows a count of the different elements required to build a selection of the network configurations considered in this evaluation. The number of nodes, switches and links can be derived easily from the topologies. The number of extra racks is measured as the number of racks that would be required to store the remaining stages of the tree, considering off-the-shelf equipment: 36-port switches of height 1U and racks of 42U. Therefore this figure is computed as 1/42 of the difference between the number of switches in a tree and those in an indirect cube. Finally the cable length is measured as follows: intra-rack links are count as 1*u*. Inter-rack links are count as the Manhattan distance between connected racks plus 2*u* to go up or down to the wiring trays. For the sake of simplicity we compute the cable length to build the trees considering that all the second- and third-stage switches are located in the same rack, *i.e.* the distances will be 1*u*. Note that this is only true for the 4 rack system, and therefore in larger trees, we are showing an optimistic estimation of cable length.

3. Experimental Set-Up

Our in-house developed Interconnection Network Simulation and Evaluation Environment, in short INSEE [20], was the workbench used to evaluate the feasibility and scalability of the indirect cube network. As stated before we will model the systems bearing in mind the design of the SuperMicro TwinBlade servers [17]: 20 compute nodes plus a 36-port InfiniBand switch per enclosure and racks containing six enclosures; look again at Fig. 1.

TABLE I. Elements required to build the different topologies.

System Size	**4 compute racks**	**12 compute racks**	**24 compute racks**	**40 compute racks**	**60 compute racks**
#nodes	480	1,440	2,880	4,800	7,200
#switches – *tree*	49	175	383	679	1,067
#switches – *cube*	24	72	144	240	360
#extra racks – *tree*	1	3	6	11	17
#links – *tree*	784	2,800	6,128	10,864	17,072
#links – *cube*	192	576	1152	1,920	2,880
aggregated cable length – *tree*	1,559 *u*	5,869 *u*	13,806 *u*	26,225 *u*	43,957 *u*
aggregated cable length – *folded cube*	432 *u*	1,296 *u*	2,592 *u*	4,320 *u*	6,480 *u*
aggregated cable length – *unfolded cube*	544 *u*	1,632 *u*	3,264 *u*	5,440 *u*	8,160 *u*

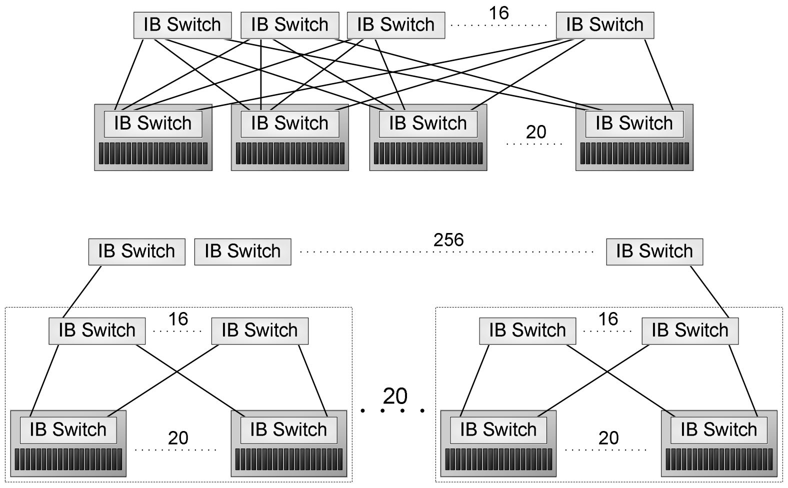

Fig. 4. Examples of the trees used in our experiments: 2-stage tree (up) and 3-stage tree (down). Most links and switches are hidden for the sake of clarity.

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

To keep things simple, we have modelled systems using external switches with a fixed number of ports. We have selected 36-port switches, similar to those included within the enclosures. With these switches we can build a thin-tree topology using 20 ports to connect with the lower stage of the tree and 16 ports to connect to the upper one. Note that, as suggested by previous research [21], 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. 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. 4.

In the case of the indirect cube, as explained before, only the first stage of switches (those inside the TwinBlade enclosures) is required to interconnect all the compute nodes, *i.e.* it will not be necessary 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. 5.

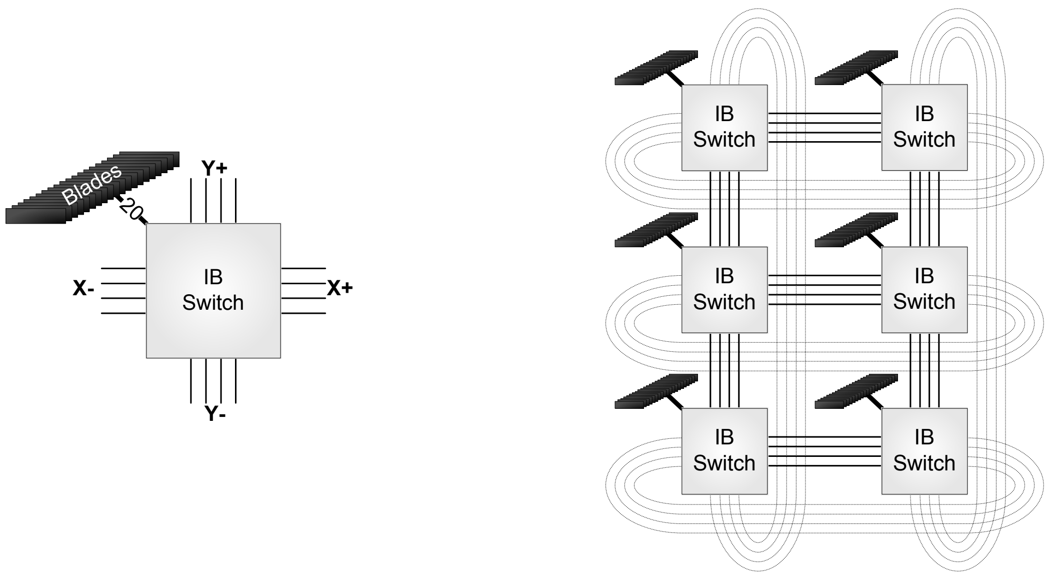

Fig. 5. Port arrangement of the switch included in a TwinBlade enclosure to form an indirect cube (left). The indirect cube constructed with the six enclosures within a single rack (right).

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 with a queue 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 available credit. Credits are transmitted out-of-band and, therefore, do not interfere with regular application traffic. In the case of the tree, adaptivity means that packets can adapt in the upward route, but downward routes are 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. We want to remark that using adaptive routing schemes allows taking the best of each topology, which in turn allows a fair comparison between them.

Note that in an actual implementation, the four parallel links of an indirect cube could be aggregated and used as a single, 4-time faster, link; we have not adopted this approach. A bubble scheme [22] is added in this topology in order to guarantee deadlock freedom. Given that it 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 [23] (therefore, penalizing new injections from nodes). For the sake of completeness we will measure the performance of the indirect cube topology when this mechanism is activated, and also when it is deactivated.

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 generated and used a collection of synthetic workloads that resemble the way in which scientific applications communicate. Most of these workloads are explained and justified in [24] and [25]. We will consider a computation to communication ratio of 9:1. We proceed with a brief description of the workloads used in this paper.

- In **All-to-All**, all the tasks send a message to all the other tasks, starting from their next one. After sending all the messages, each task 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 task interchanges messages only with a small subset of tasks, instead of with all of them. The message size is 10 Kbytes.
- In **Near Neighbour**, the tasks are arranged in a virtual 2D torus. Each of them sends a message to each of its neighbours following a dimension order (X+, X-, Y+ and finally Y-) and then waits for the reception of its neighbours’ messages. The message size is 10 Kbytes.
- In **n-Bodies**, tasks are arranged in a virtual ring. Each task starts a chain of messages that travel clockwise across half of the ring. When the chain finishes, the task at the other side of the ring sends a message to the source. The message size is 10 Kbytes. Note that this workload is only defined for an odd number of tasks. 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 [26].
- In **Random**, multiple waves of messages are generated in such way that all the messages of a wave must be received before start sending the messages if 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 our experimental set-up only allowed to scale them up to 16 racks (1920 nodes).

All-to-All, Butterfly, Near Neighbour and all the Random cases can be considered *communication-intensive* workloads, as they will stress the network with scenarios with high levels of contention and congestion. In contrast, Binary Tree and n-Bodies have high levels of causality which will limit the formation of persistent contention for the utilization of network resources.

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 this placement will favour the communication pattern of the Near Neighbour workload, as we will see later, but 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. Performance Results

The execution time for the different systems and workloads are plotted in Fig. 6. 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 1. A lower value represents a faster dispatching of the messages. The tree topology 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 the congestion control mechanism is activated.

The execution times needed by the indirect cube are noticeably longer than those of the tree; this was to be expected, considering the greater amount of network resources (links, switches) used in the tree. The only exceptions are Binary Tree and Near Neighbour workloads. The former has such a low-demanding communication pattern that most networks could 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 behaviour.

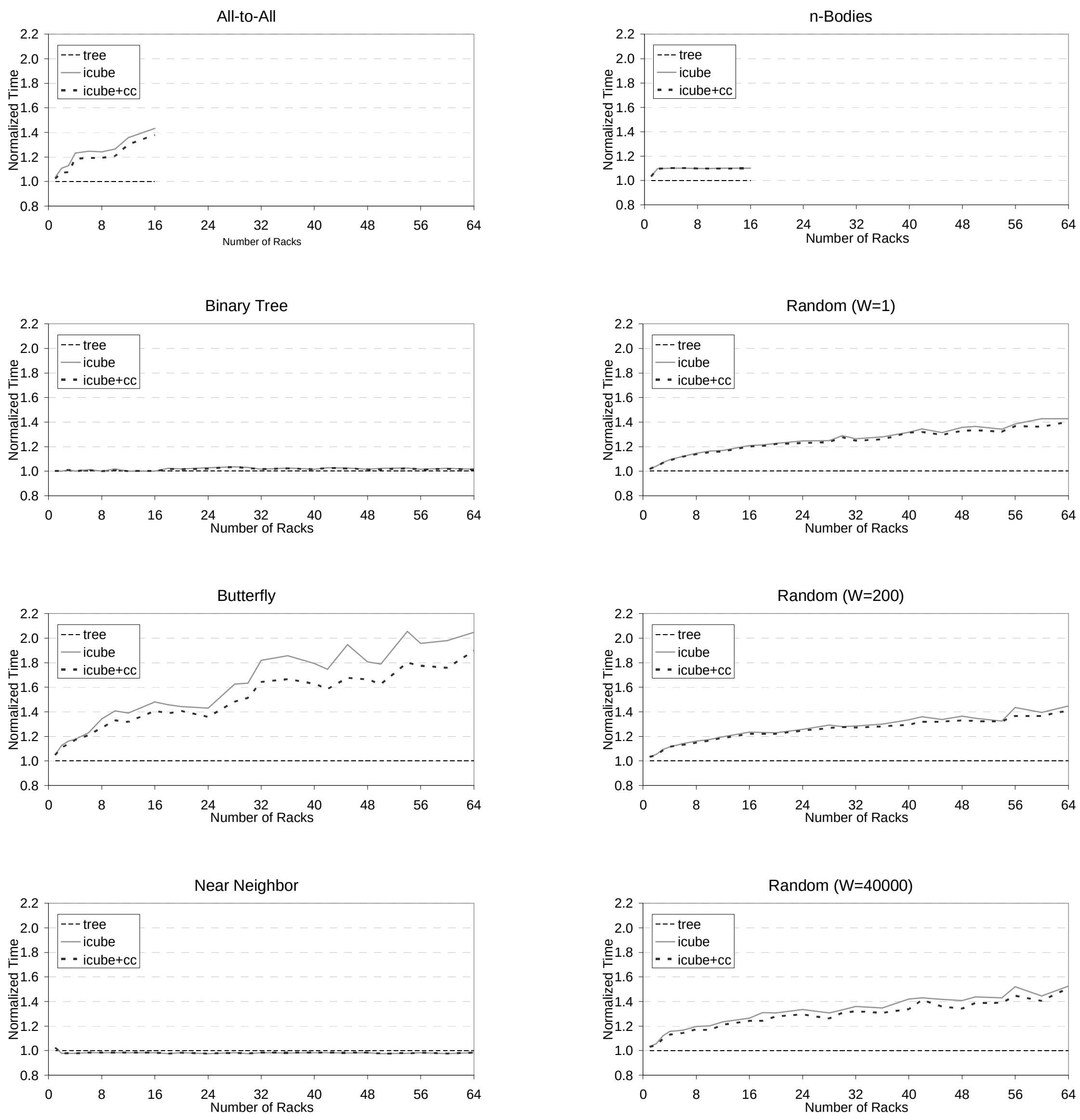

Fig. 6. Execution time employed by each topology and workload. Values normalized to the tree.

It is also noticeable that in those cases where the tree outperforms the indirect cube, the benefits of the tree increase with the network size. This is explained by the lower scalability of cube-like topologies when compared with the trees: distance scales $O(\sqrt{N})$ in the indirect cube versus $O(\log N)$ in the tree, being N the number of nodes of the topology.

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. 1.1). 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 found to be more affected by the task-to-node placement than by the topology selection.

The reader should note that, for the All-to-All, Butterfly and Random workloads, the activation of the congestion control mechanism is noticeably effective to boost execution time. In the remaining cases, 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 running workloads whose communication patterns resemble a mesh-like topology. Loosely-coupled workloads which do not tend to flood the network are also a good target application for this topology. However, our results show that for workloads of other nature the network may become an important performance bottleneck which will be increasingly poor as the system grows.

To close the section, we emphasise 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, budget, power consumption and heat dissipation. Furthermore the number of links and their length is also reduced, which should lead to a faster and cheaper system deployment.

5. Power Consumption Results

Results shown in the previous section have focused on the achievable performance. This section will be devoted to carry out a power consumption-oriented evaluation. We will model the total consumption of the network using the following methodology. The consumption, C, of a given configuration will depend on the number of switches, S, the time needed to dispatch the workload, T, and the average port-utilization during the (simulated) execution time, L. Furthermore, it will take into account two constants referred to the switches, the passive consumption, p_c, and the active consumption, a_c. To sum up, S depends on the topology, T and L are measured by simulation and, p_c and a_c depend on the switching technology. We have considered values of a_c which are ten times higher than p_c. Note that the selected values of these two constants are utterly pessimistic and favour the tree because it has more switches that are more often in *inactive* states. The formulation of C is as follows.

$$C = S \cdot \left(p_c + \left(a_c \cdot L\right)\right) \cdot T$$

With this function, we have modelled the power-consumption of the different configurations. Note that this model only has into account the power requirements of the network itself, while other related energy requirements (such as, for example, those required for heat dissipation) are not included. The obtained results are plotted in Fig. 7. Again, results have been normalized to those of the trees. A lower value represents a lower power-consumption.

The results show that, in terms of power-consumption, the indirect-cube topology is a better topological alternative than the tree because, for most workloads, it provides noticeably better figures than the tree. With those workloads in which the performance was good (Near Neighbour and Binary Tree), the indirect cubes are able to deliver the workloads using roughly one half of the power required by the tree. With n-Bodies, the indirect cubes offer better figures than the tree but with a much smaller difference, around 15% power savings.

With communication-intensive workloads the indirect cubes require less power than the tree up to a given scale (roughly 20 racks with all-to-all, 30 racks with butterfly and random W=40000, and 40 racks with the other two random workloads). Once these system sizes are reached, the indirect cube is not able to compete with the tree anymore. Note that this limitation has a simple explanation: the power consumed per time unit of the indirect cube is much lower than those of the tree; however the execution time is large enough to make this figure worthless.

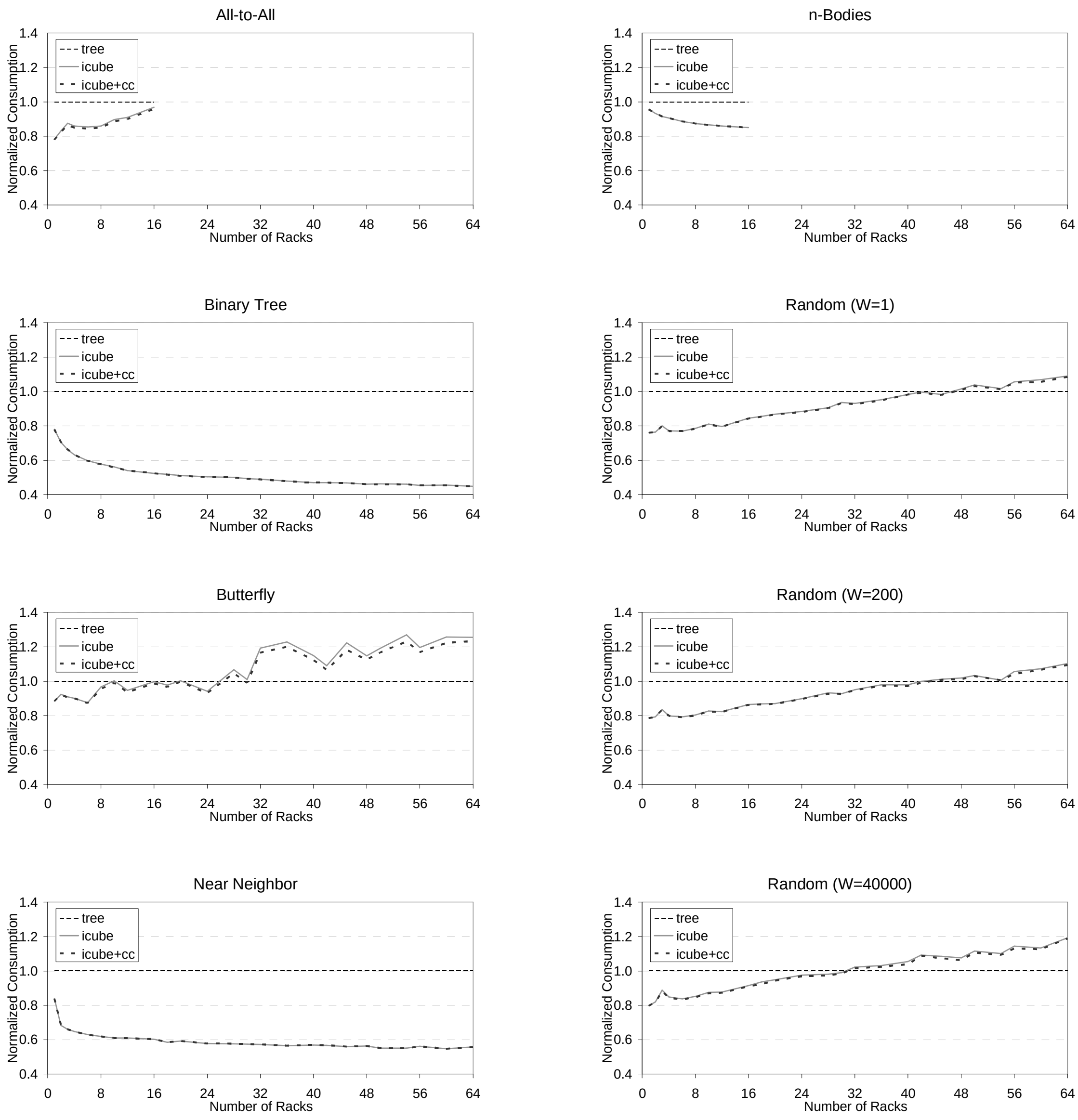

Fig. 7. Power consumed by each topology and workload. Values normalized to the tree.

To conclude this section, we want to remark that the indirect cube has been shown to offer reasonable performance levels with non-demanding workloads while noticeably reducing the power-consumption of the interconnection network. With very demanding workloads, the low performance makes power requirements to increase due to a run time increase. Still this topology can handle demanding workloads in an efficient way if the system scale is not excessively large.

6. Conclusions

In this paper we have proposed arranging indirect networks, such as those used in current clusters and superclusters, forming cube topologies somewhat similar to those traditionally used in massively parallel processing systems. These indirect cubes reduce power consumption and heat dissipation when compared with the tree-like topologies traditionally used in compute clusters, because they reduce the number of interconnection network elements. The rationale behind this proposal was discussed focusing basically in reducing complexity of deployment and cost, even if performance is somewhat affected.

The indirect cube was compared to a tree-like topology, the *de facto* standard when building 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. On the other hand, the performance of indirect cubes with less intense workloads (for example, those with long chains of dependencies) is comparable to that of the tree. It is also worth to highlight that an underlying cube-like topology can be effectively exploited by applications, by means of arrangements of processes using virtual meshes and tori, which can exploit locality in communications. In our set-up, the indirect cube topology experienced an improvement in performance of roughly 10% when dealing with mesh-like workloads.

Regarding power consumption, results show that the indirect cube can handle most workloads with lower power consumption than the tree. The only exceptions to this rule are some communication-intensive workloads composed of thousands of communicating tasks because of the severe slow-down they may experience.

The overall conclusion of this paper is that clusters arranged as an indirect cube can be competitive up to a given scale. However due to their limited scalability they may be counterproductive when dealing with communication-intensive workloads once the system reaches a given size, between two and five thousand nodes in our set-up depending on the workload. Still, as the networking technology remains the same, it is possible to scale up from a small system arranged as an indirect cube to a larger system interconnected by a tree-like topology, just by adding more racks. Note that the switching elements and the wires can be reused seamlessly.

Previous studies in cost-effective topologies for large-scale clusters have analyzed thinned tree topologies [21]. As future work, we plan to compare these thin trees against indirect cubes in order to explore if there is any limit of the slimming factor for which the indirect cube outperforms the trees. In other words, we want to find out whether it is more reasonable to build indirect cubes rather than excessively thinned trees.

Moreover, we plan to conduct more research regarding deadlock-free routing schemes for indirect cubes in order to circumvent the need for deadlock avoidance mechanisms that are not implemented in current InfiniBand and Ethernet switches. Approaches relying on dimension order routing [12] could be a first step towards deadlock-free routing in this topology. More sophisticated schemes may consider link aggregation and/or splitting the cube in separate virtual networks, such as sub-meshes. Another interesting approach would be providing application-specific routing schemes as those proposed in [27] for tree-like topologies.

Acknowledgements

This work has been supported by the Ministry of Education and Science (Spain), grant TIN2010-14931, and by the Basque Government, grant IT-242-07. Dr. Navaridas held a post-doctoral fellowship from the University of the Basque Country UPV/EHU when this research was performed and is currently supported by a Newton International Fellowship awarded by the Royal Society.

References

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

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

[3] InfiniBand Trade Association. "InfiniBand Architecture Specification". Vol. 1, R. 1.0.a.

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

[5] 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.

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

[7] 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

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

[9] 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.

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

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

[12] WJ Dally and B Towles. “Principles and Practices of Interconnection Networks”. Morgan Kaufmann Series in Computer Architecture and Design, 2004. ISBN: 0-12-200751-4

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

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

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

[16] 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

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

[18] 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

[19] 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.

[20] J Navaridas, et al. “Simulating and Evaluating Interconnection Networks with INSEE”. Elsevier’s Simulation Modelling Practice and Theory 19(1), Jan. 2011, pp. 494-515. DOI: 10.1016/j.simpat.2010.08.008.

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

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

[23] 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.

[24] 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.

[25] 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.

[26] CL Seitz. "The cosmic cube". Communications of the ACM 28 (1) (Jan. 1985), pp. 22-33. DOI: 10.1145/2465.2467

[27] G Rodriguez, et al. "Exploring pattern-aware routing in generalized fat tree networks". Proceedings of the 23rd International Conference on Supercomputing, Yorktown Heights, NY, USA, June 2009, pp. 276–285, doi:10.1145/1542275.1542316