Energy Efficient Torus Networks with On/Off Links

Francisco J. Andújar[a], Salvador Coll[b], Marina Alonso[c], Juan-Miguel Martínez[c], Pedro López[c], José L. Sánchez[d], Francisco J. Alfaro[d], Raúl Martínez[e]

[a]*Departamento de Informática, Universidad de Valladolid, Valladolid, Spain*
[b]*Instituto de Instrumentación para Imagen Molecular (I3M), Universitat Politècnica de València, Valencia, Spain*
[c]*Department of Computer Engineering, Universitat Politècnica de València, Valencia, Spain*
[d]*Computing System Department, University of Castilla-La Mancha, Albacete, Spain*
[e]*Amazon Lab126, Cupertino, California, United States*

Email addresses: fandujarm@infor.uva.es (Francisco J. Andújar), scoll@upv.es (Salvador Coll), malonso@upv.es (Marina Alonso), jmmr@upv.es (Juan-Miguel Martínez), plopez@upv.es (Pedro López), jose.sgarcia@uclm.es (José L. Sánchez), fco.alfaro@uclm.es (Francisco J. Alfaro), raumar@amazon.com (Raúl Martínez)

Abstract

Future exascale computing systems will require energy and performance efficient interconnection networks to respond to the high data movement demands of new applications, such as those coming from big-data and artificial intelligence areas. The network structure plays a major role in the overall interconnect performance, for this reason torus is a common topology used in the current largest supercomputers. There are several proposals to improve energy efficiency of interconnection networks. However, few works combine both energy and performance, and sometimes they are treated as opposed issues. In this paper, we try to determine which torus network configuration offers the best performance/energy ratio when high-radix switches are used to build the interconnect system. The performance/energy evaluation has been performed by trace-driven simulation under realistic scenarios, where several mixes of scientific applications share a supercomputer system and are scheduled to be executed with the available resources at each moment.

Keywords: Interconnection networks; energy and performance evaluation; energy consumption; power consumption; n-dimensional torus; port aggregation

1. Introduction

One of the most popular topologies in the largest current supercomputers is the k-ary n-cube, also known as n-dimensional torus [1]. The torus topology has a fixed switch radix that facilitates the network fabric implementation, it is scalable and has a linear cost of expansion. Moreover, the torus provides multiple paths for every pair of source/destination nodes in such a way that fault tolerance and load balancing become feasible. For these reasons, several commercial interconnection systems allow to implement a torus network.

For example, 3D tori are supported by EXTOLL switches [2] and the interconnection network of the Cray CS Series supercomputers [3], while the interconnection network of the IBM Blue Gene/Q [4] implements a 5D torus. In November 2018, there are two supercomputer machines using the torus topology in the top ten Top500 list [5] and seven in the top ten Graph500 list [6].

For a given size, the performance of a torus interconnection network directly depends on its number of dimensions. The higher the number of dimensions, the lower the distances among nodes and therefore, the higher the network performance. The path diversity is also increased [7, 1], improving the efficiency of adaptive routing algorithms.

However, as shown in the previous examples, commercial torus networks usually have a low number of dimensions, since wiring becomes more complex as the number of dimensions increases. Indeed, Dally [8] and Agarwal [9] showed that under fixed bisection and chip packaging, lower radix networks offer lower packet latency. Scott and Goodman [10] introduced link pipelining in the network model, favoring a slightly higher dimensionality for large networks. In addition, multiple scientific applications use 3D mathematical models, whose communication patterns naturally fit in a 3-dimensional torus. The combination of ease of wiring with the use of 3D models makes low dimensional tori very appropriate topologies for designing an interconnection network.

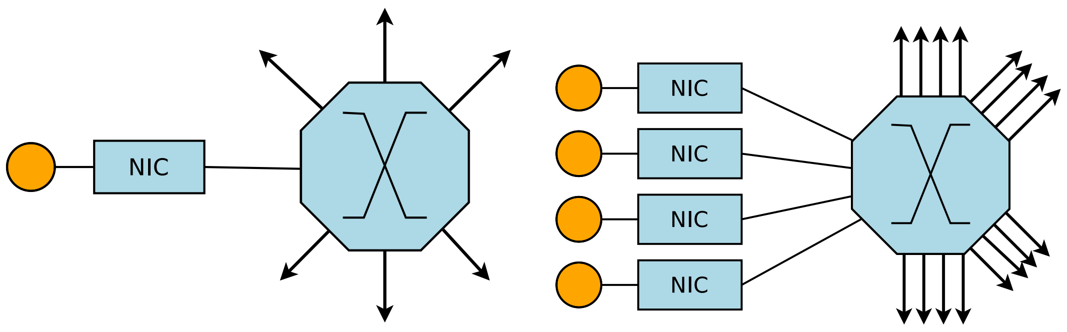
Currently, switches with a high number of ports (i.e. high-degree switches) are commercially available [11, 12]. Network designers can take advantage of these high-degree switches to build low dimensional tori using the link aggregation concept (also denoted as link trunking). This concept consists of connecting every pair of adjacent switches by means of two or more physical channels. Two design approaches are possible. On one hand, we can combine several physical channels to work as a single wider physical link, therefore increasing the channel bandwidth. For example, the 4X QDR links on Mellanox products are composed of four QDR lanes [13] and therefore, four flits of the same packet are transmitted in parallel by the 4X link. On the other hand, we can use the channels as independent links, i.e. the ports of a trunk link transmit different packets, in order to increase path-diversity and routing flexibility. For example, the Cray Gemini [14] uses 10 links for building a 3D torus: 4 links for the X and Z dimensions, and only 2 links for the Y dimension. In addition, Gemini has two nodes connected to each router, reducing the number of routers in the network. Note that combining both design approaches is also possible. In fact, the internal router of Cray Gemini has 48 ports [14]: 8 ports are used to communicate the router with the 2 Gemini NICs, while the remaining 40 ports comprise the 10 Gemini links, using 4 ports per link.

The use of trunk links increases switch to switch bandwidth. We can exploit this increased bandwidth by attaching several NICs to every switch (as stated above, the Cray Gemini attaches two NICs). An interesting observation is that, for a fixed system size (in number of NICs -i.e., computing nodes-), the number of network switches in a torus is reduced by the number of NICs attached to each switch. Of course, switches with a higher number of ports are required.

Network performance is not the only parameter to take into account when designing the network. A trade-off between performance and cost is usually required. Regarding cost, we must take into account not only the cost of network design and deployment but also exploitation costs. Leaving aside the possible network failures, this cost greatly depends on the network power consumption.

From this point of view, for a given network size N, two design options are possible. The first one is using a high dimensional network without link aggregation and one NIC per router. The extreme case would be a hypercube[1] with $\log_2 N$ dimensions. The second option is using a lower dimensional network with link aggregation, several NICs per router and a lower number of routers. The first option requires a higher number of smaller switches, while the second requires a lower number of bigger switches. To

[1]Remember that an hypercube is an n-dimensional torus with 2 nodes in each dimension. Therefore, the number of dimensions of the hypercube is $\log_2 N$.



(a) 7-port switch for 3D tori (b) 20-port switch for 2D tori

Figure 1: Torus node configuration for building a 64-node network.

do a fair comparison, the bisection bandwidth of both design choices should be the same, in order to keep constant the theoretical network bandwidth. Notice, though, that there may exist several intermediate design options that use different levels of link aggregation.

As an example, let's consider a 64-node network. Using the first approach, we could build a $4 \times 4 \times 4$ 3D torus with 64 switches with one computing node per switch. Each switch would have 7 links (one port per each network direction plus one port to attach the computing node). Figure 1a shows this kind of switch. The network would have $64 \times 6 = 384$ links, an average distance of $n \times \frac{k}{4} = 3 \times \frac{4}{4} = 3$ and a bisection bandwidth of $2 \times k^{n-1} = 2 \times 4^2 = 32$ bidirectional links.

Using the second approach, we could build a 4×4 2D torus with 16 switches, with 4 trunk links and 4 computing nodes per switch. Each switch has 20 links (4 ports per each network direction plus 4 ports to attach 4 computing nodes). Figure 1b shows this second kind of switch. This network has $16 \times 16 = 256$ links, an average distance of $2 \times \frac{4}{4} = 2$ and a bisection bandwidth of $4(\#\text{ of trunk links}) \times 2 \times k^{n-1} = 4 \times 2 \times 4^1 = 32$ bidirectional links.

Although both networks have the same bisection bandwidth, they will show different behaviors. The network with aggregated links (the 2D torus in the previous example) should have lower message latencies under low traffic loads because it has lower average distance than the network with more dimensions. However, the switch allocator performance decreases when the number of ports increases [15]. After reaching certain traffic load, the network with more dimensions (the 3D torus in the previous example) should obtain better performance since its switches allow to achieve greater throughput as they are smaller. On the other hand, although the 3D network will achieve better performance under heavy traffic loads, the 2D network still has a lower power consumption because it has less network links (see Sections 2 and 3.2 for details).

The questions we try to answer in this paper are: which design option for the network is overall more energy efficient? A high-dimensional network with single links or a low-dimensional network with aggregated links? Is the performance loss for high loads of the network with aggregated links acceptable if it consumes lower energy? Since the energy consumed for running an application depends on both the system power consumption and the execution time, these are not trivial questions. Although a lower dimensional network with aggregated links has lower power consumption, the execution time could be longer. This might increase the energy consumption of the system.

Another issue that must be taken into account is the fact that interconnection networks can provide dynamic mechanisms to save energy [16, 17]. In particular, the mechanism proposed by Alonso [17] turns off links when low network utilization is detected. This mechanism is very well suited for networks with trunk links, since turning off individual links from trunk links maintains the topology and does not require changes in the routing algorithm, provided that there is at least one operating link. Similar techniques are used in real products for saving energy. For example, in Mellanox switches, a link-level power saving feature reduces the width of the trunk link when a fabric is underutilized [13].

In this paper, we analyze the performance of different k-ary n-cube configurations to determine which configuration is more energy-efficient. To do so, we analyze both performance and energy consumption, also applying dynamic power saving techniques. Performance of the different configurations is determined by using application execution traces. The rest of the document is organized as follows. In Section 2 we describe the system power consumption model. Section 3 presents the system and evaluation model. After that, we analyze and discuss network performance and energy evaluation results in Section 4. Section 5 shows the related work. Finally, in Section 6 we outline the conclusions and future work.

2. A simple power consumption model

As mentioned in Section 1, the goal of this paper is to analyze the impact of different configurations of the interconnection network topology on the energy consumption of a high performance computing (HPC) platform (cluster or supercomputer).

The study presented in this paper has been developed by simulation, using a tool that models the nodes and the network that interconnects them. To obtain energy results, we have included in the simulator a simple power consumption model where the contribution of the main components of the interconnection network is considered. However, notice that we are interested in carrying out a comparative study, and therefore it is not totally relevant to use the absolute power consumption of each system component, but to determine the fraction of the total power consumed by those components.

Using the simulation tool, we will obtain the total energy consumed by an application running on the HPC platform from its execution time and the calculated power consumption during the simulation. As it is well known, the power consumption of many building blocks of a computing system is the sum of a static or fixed component, and a dynamic component, that varies according to their utilization. For this reason, during the simulation, data related to dynamic power is collected, in order to calculate the total power consumption at the end of the simulation.

Due to the relevance of the links in the performance and energy efficiency of the network, in order to determine the total power consumption of a switch, our model considers the power consumption of the links and the power consumption of the remaining switch logic. According to the state of the art, we consider the following general hypotheses:

- The switch power consumption increases linearly with the number of ports [18].

- We assume two states for the switch ports: *wake-up* (or *turned on*) and *sleep* (or *turned off*), similar to the ones used in the Low Power Idle (LPI) power-saving mechanism proposed in the IEEE **E**nergy-**E**fficient **E**thernet standard (IEEE 802.3az, and from now on, the EEE standard) [19]. LPI freezes transceiver state when it enters in sleep mode and restores it when the port is waked up, drastically reducing the power consumption and allowing to turn on/off the links in a few microseconds [20]. Therefore:

    - Since the transceiver is working regardless of whether the port is transmitting data or not, the port power consumption is 100% when it is turned on.
    - When the port is in sleep mode, it consumes a small part of the total power consumption.
    - During the transitions from one state to another, the port power consumption is 100%.

- We assume the power saving strategy proposed by Alonso et al [17] for the aggregated torus topology. Briefly, this strategy always maintains one active port per aggregated link, turning on/off the remaining ports depending on the aggregated link utilization. Note that topology is not modified, preserving connectivity, and the same routing algorithm is used.

At the end of this section, we provide an equation that allows us to obtain the total energy consumed by the execution of a given application for each network configuration on the HPC platform. Previously, a set of definitions is introduced, related to the system components and their contribution to the total power consumption of such system.

2.1. Definitions

We introduce the parameters we use to quantify both the main components of the system and their contribution to power consumption and total energy.

In order to compare networks with different number of routers/ports, we normalize the power consumption with respect to a "reference" network. Let's consider as an example the networks built from the switches shown in Figure 1 (page 5). Considering the 3D torus network as the reference network, its maximum power consumption will be 1. According to our initial hypotheses, switch (and network) power consumption linearly increases with the number of ports. The ratio of ports between the 2D torus and the 3D torus is $\frac{16\times 20}{64\times 7} = 0.714$. Therefore, the maximum power consumption of the 2D torus network will be 0.714, although it could be even lower if power saving strategies were applied (e.g. turning off unused or underutilized links). For this reason, we have included several terms related to the reference network, that are denoted with the prefix REF.

- N_{ports}: Number of ports per switch.
- N_{sw}: Number of switches in the network.
- REF_{ports}: Number of ports per switch in the reference network.
- REF_{sw}: Number of switches in the reference network.
- U^{p}_{port}: Fraction of time that a port p is turned on.
- ω^{p}_{Sport}: Fraction of the power consumption that a port p consumes while in *sleep* mode.
- Run_{task}: Fraction of *Runtime* that a process/task attached to a node is running; i.e. the time that the process/task is not idle because it is performing a network communication.
- ω_{Snodes}: Fraction of the power consumption that a node always consumes, independently of the node load.
- ω^{s}_{ports}: Contribution of the ports in a switch s to the total power consumption of that switch.
- ω_{net}: Contribution of the network to the total power consumption of the system.

2.2. Power consumption model

Let W^{s}_{ports} be the fraction of the power consumption that all the ports consume in a switch s with respect to the maximum power consumption of those ports. When a port p is in sleep mode, only consumes ω^{p}_{Sport} of the total power consumption. Then, a port p always consumes ω^{p}_{Sport}, plus $(1-\omega^{p}_{Sport})$ when it is turned on. Therefore:

$$W^{s}_{ports} = \frac{1}{N_{ports}} \sum_{i=1}^{N_{ports}} \left(\omega^{i}_{Sport} + (1-\omega^{i}_{Sport}) \cdot U^{i}_{port}\right)$$

As ports in a switch have the same characteristics, $\omega^{1}_{Sport} = \omega^{2}_{Sport} = \cdots = \omega^{N_{ports}}_{Sport} = \omega_{Sports}$. Therefore:

$$W^{s}_{ports} = \omega_{Sports} + (1-\omega_{Sports})\frac{1}{N_{ports}} \sum_{i=1}^{N_{ports}} U^{i}_{port}$$

If the average value U_{ports} for all U^i_{port} is considered:

$$U_{ports} = \frac{1}{N_{ports}} \sum_{i=1}^{N_{ports}} U^i_{port}$$

we obtain:

$$W^s_{ports} = \omega_{Sports} + (1 - \omega_{Sports}) \cdot U_{ports}$$

Once we have the port power consumption, we can calculate the proportion of the switch power consumption with respect to its maximum power consumption. In order to simplify the model, we consider that the remaining logic of the switch always consumes the maximum power regardless of its utilization, i.e. the switch logic always consumes $(1 - \omega^s_{ports})$. Therefore:

$$W^s_{sw} = (1 - \omega^s_{ports}) + \omega^s_{ports} \cdot W^s_{ports}$$

If we consider all switches in the network, we can obtain the network power consumption:

$$W_{net} = \frac{1}{N_{sw}} \sum_{i=1}^{N_{sw}} W^i_{sw} = \frac{1}{N_{sw}} \sum_{i=1}^{N_{sw}} ((1 - \omega^i_{ports}) + \omega^i_{ports} \cdot W^i_{ports})$$

Without loss of generality, and reasoning at network level in the same way as switch level, we consider that the contribution of switch ports to the total switch power consumption is the same for all switches $(\omega^1_{ports} = \omega^2_{ports} = \ldots \omega^{N_{sw}}_{ports} = \omega_{ports})$. Considering W_{ports} as the average value of W^i_{ports}:

$$W_{net} = (1 - \omega_{ports}) + \omega_{ports} \cdot W_{ports}$$

However, this way of obtaining the network power consumption is only valid to compare network topologies using the same type of switch; i.e. the number of ports per switch must be the same in each topology. Since we want to compare networks with different number of switches or ports per switch, we need to normalize the power consumption with respect to a reference network.

As our initial hypothesis is that the power consumption increases linearly with the number of ports, once we have chosen the reference network, we can calculate the relative network power consumption:

$$W_{net} = ((1 - \omega_{ports}) + \omega_{ports} \cdot W_{ports}) \cdot \frac{N_{ports}}{REF_{ports}} \cdot \frac{N_{sw}}{REF_{sw}}$$

In order to obtain the total energy of the HPC platform, we need to consider the power consumption of the computing portion, mainly due to the compute nodes. Taking into account the definitions above, and again considering the average behavior of the nodes, the fraction of the power consumption that all the nodes consume in the system, with respect to the maximum power consumption of those nodes, can be expressed as:

$$W_{nodes} = \omega_{Snodes} + (1 - \omega_{Snodes}) \cdot Run_{task}$$

where we assume that there is a part of the total power consumption that is always consumed (ω_{Snodes}), while the remaining power consumption depends on the node load, that is, the fraction of time that the processes/tasks attached to the node are not idle due to the network communications (Run_{task})

Then, considering the nodes and network power consumption, we can calculate the HPC platform power consumption:

$$W_{cluster} = \omega_{net} \cdot W_{net} + (1 - \omega_{net}) \cdot W_{nodes}$$

$W_{cluster}$ provides the fraction of the maximum power (both network and nodes) consumed during the application execution and normalized with respect to the reference network. Finally, the network and system energy consumed by an application is, respectively:

$$\begin{aligned} E_{net} &= W_{net} \cdot Runtime \\ E_{cluster} &= W_{cluster} \cdot Runtime \end{aligned}$$

2.3. Parameter characterization

Once the power model has been defined, we must select the values for the parameters that determine the fraction of power consumed by the various network building blocks, as defined in our model. Table 1 summarizes all the selected values.

According to the EEE standard, the power consumption of an idle link[2] is estimated to be 10% of the link power consumption [19, 20]. Therefore, we set ω_{Sport} to 0.1.

The weight of the link power consumption with respect to the switch power consumption is 65% and 63% for the Dell PowerConnect 5324 (24-port

[2]Note that, in our terminology, an idle link is equivalent to an *off link*, while an active link is equivalent to an *on link*.

Table 1: Power model parametrization

Parameter	ω_{Sport}	ω_{ports}	ω_{net}	ω_{Snodes}
Value	0.1	0.65	0.15	Variable

switch) and the Dell PowerConnect 6248 (48-port switch) [21], respectively; 64% for an IBM Infiniband 8-Port 12X switch [22] and 68% for the EXTOLL Tourmalet switch [23]. According to that, we consider that 0.65 is a realistic estimation for ω_{ports}.

Finally, we set ω_{net} to 0.15, since the network power consumption is 10%~20% [24, 25] of the full system. We have not fixed ω_{Snodes}, since we want to study the impact of node power consumption on the final results. A realistic estimation of this parameter is ≈ 0.5 since even energy-efficient servers still consume half of their power while idle [26].

3. System model

After presenting the power model, we describe the system model used as evaluation testbed. Section 3.1 outlines the switch architecture model while Section 3.2 shows the topologies selected for our experiments. Finally, Section 3.3 briefly explains the network load model.

3.1. Switch model

We consider that all the switches in the network use the same technology, independently of their number of ports. The modeled architecture is realistic and representative of current state of the art HPC platforms, since the design parameters have been chosen based on several commercial networks [2, 4, 11, 27, 28].

The main specifications of the switch architecture are the following: *IQ* (*Input Queued*) switches [29], *virtual cut-through* switching [30], credit-based flow control and the three-stage allocation algorithm implemented in IBM Blue Gene L [31], with the only difference that our algorithm employs round-robin arbiters in all the allocator stages.

Data are transmitted in flits of 16 bytes, grouped in 8-flit packets (or 128-byte packets). The switch logic frequency is 625 MHz (i.e. switch clock resolution is 1.6 ns). Since the switch crossbar can deliver one flit per cycle, each switch port offers a peak bandwidth of 10 Gbytes/s.

Table 2: Case studies

Nodes	Torus	Dim.	Num. ports	Allocator latency	Port aggr.	Num. switches	Network ports	Port ratio	Name (No PS)	Name (PS)
64	3D	4x4x4	7	3	1	64	448	–	3D-1X	–
	2D	4x4	20	5	4	16	320	0.714	2D-4X	2D-4X-PS
	1D	4	48	6	16	4	192	0.428	1D-16X	1D-16X-PS
256	4D	4x4x4x4	9	4	1	256	2304	–	4D-1X	–
	3D	4x4x4	28	5	4	64	1792	0.777	3D-4X	3D-4X-PS

The latency per hop is approximately 50 ns, slightly varying as a function of the number of ports, as explained below. Since we assume that the switches are implemented with the same technology, we can keep fixed the latency of the switching units. The latency of input buffering and the port serializer does not depend on the number of ports. On the other hand, considering a table-based routing algorithm, its latency mainly depends on the memory technology and the maximum network size supported by the architecture. The only switch unit with a variable latency is the switch allocator. Algorithms like iSLIP [15] or our allocator based on the IBM Blue Gene L allocator [31] are composed of several stages of multiple round-robin arbiters. Increasing the number of ports linearly increases the number of round-robin arbiters (and therefore the hardware complexity) but the number of stages is kept fixed and the increment of the round-robin arbiter latency is only logarithmic [32]. This slightly increases the allocator latency. Table 2 shows the allocator latency, measured in cycles, for each router size.

Regarding the buffer organization, each input port has an input buffer of 1024 flits, or 16 Kbytes, statically split between four virtual channels (VCs). The virtual channels are employed to avoid deadlocks and to provide adaptiveness. In this case, the switch implements a fully-adaptive routing algorithm [1]. Two of the four VCs are fully-adaptive while the remaining ones are used as escape paths, implementing the DOR algorithm using two VCs to avoid deadlocks.

Finally, we have implemented the trunk links using the second approach described in Section 1, i.e. each trunk link comprises several independent ports transmitting independent packets. We have also implemented the power saving strategy described in [17][3]. Note that we have not employed a

[3]Although the network implements adaptive routing, note that an off port can not be

power saving strategy in the reference network (the torus network without port-aggregation). Beside being the reference topology, the main reason is to avoid a great performance penalty caused, not by the topology, but by the power saving strategy. In particular, although other strategies like turning-off the ports after a certain unused time could be applied, the performance penalty is significant, while the Alonso's power saving technique has very little impact on performance since the network is always fully-connected.

Regarding the power saving strategy parameters, although our router is not related to any specific technology, we have used the time values specified in EEE standard [20] to configure the delays for turning on (4.16 μs) and turning off (2.88 μs) a link. These delays are used in all the networks when power-saving is active.

3.2. Case studies

We have selected systems with 64 and 256 compute nodes. Table 2 shows the topologies evaluated for the same amount of compute nodes, showing the most relevant parameters. Note that all the networks evaluated for each system size have the same bisection bandwidth.

The labels used to identify the topologies use the following nomenclature: **n**D-**p**X, where **n** is the number of dimensions and **p** is port aggregation.

Note that if a power saving mechanism is used in the network, the corresponding label ends with **-PS**. And finally, remember that we have chosen the torus topology with the highest number of dimensions (and no aggregated links) as the reference system for both system sizes (i.e., 3D torus for 64 nodes and 4D torus for 256 nodes).

3.3. Network load

The assessment of energy consumption of an HPC platform relies on the correct estimation of power consumption and execution time for a given load. Synthetic traffic patterns, typically used in performance evaluation, are not appropriate for this purpose since the difference among message latency on different network configurations cannot be translated into differences in execution time. As a result, for modeling the network load, we have used an open access trace-driven traffic model, called *VEF trace model* [33, 34]. The MPI traffic injected by parallel applications is captured in a trace file which is

chosen by the routing function.

later used to generate the traffic in the network simulator. VEF traces model both MPI point-to-point and MPI collective communication primitives, using the collective communication algorithms implemented in OpenMPI [35].

Specifically, we have performed an evaluation using the VEF traces generated by parallel applications run in the *GALGO* supercomputer [36]. For our tests, we have selected the following applications, trying to consider different realistic scenarios[4]:

- *Namd* is a parallel application for simulating large biomolecular systems [37]. The application logically maps tasks in a 3D grid and tasks communicate mainly with their neighbor in the grid. For this reason, its traffic pattern shows a great spatial locality. Our traces correspond to the STMV benchmark.

- *Gromacs* [38] is a scientific application to perform molecular dynamics. Similarly to the previous application, it shows a great spatial locality. We generated the trace using the input "d.poly-ch2" available in the Gromacs benchmark[5].

- *HPCC **MPI** **R**andom **A**ccess* [39] (or MPIRA). Most of the communications are performed by MPI point-to-point primitives. Messages are evenly distributed among all the tasks, producing a very close to uniform traffic pattern.

- *HPCC Linpack* [39] is used to solve a dense system of linear equations. This application follows a special communication model in which every task typically communicates with the same tasks, following a ping-pong pattern.

- *Graph500 benchmark* using the *replicated-csr* implementation, a scale factor of 20 and an edge factor of 16 [6]. All the communications are generated by MPI collective primitives that generate a great exchange of data among tasks. This application generates the highest network load of all the tested applications.

[4]All the VEF traces described in this work and the software needed to run the VEF traces are available free at the VEF website [33].

[5]`http://www.gromacs.org/About_Gromacs/Benchmarks`

We have used the aforementioned traces in two ways. First, we evaluate system performance running only one application at a time. Next, we launch a set comprising several applications that are scheduled in the system.

Single trace evaluation

In first place, we have evaluated each system executing a single application each time. We have used 512-task traces for simulating each application. In order to represent a more realistic environment, we simulate multicore nodes whenever possible. That is, for testing the 64-node networks, we have simulated 8-core nodes, but for testing the 256-node networks, we have simulated only two cores per node. Unfortunately, the trace generation is limited by the size of the GALGO supercomputer, thus we can only simulate nodes with a small number of cores.

Trace scheduler evaluation

We have developed an oblivious trace scheduler to overcome the limitation of the number of tasks per trace and to evaluate the interconnection networks under a more realistic environment. Given a set of traces, the operation of the trace scheduler is as follows. In first place, the scheduler checks the number of available cores. Then, it checks the number of tasks of each trace to determine which traces are selectable; that is, which traces can be executed with the available resources. Finally, the scheduler randomly choses a selectable trace and maps it to the first free nodes. Note that the nodes are identified with a number from 0 to $N - 1$. The scheduler checks the nodes following an ascending order and mapping the tasks in the first free nodes found. This process is repeated until all the traces are mapped or there are no free nodes to map another trace. In the latter case, the scheduler will be executed again when a trace finishes and frees the resources it was using.

For the trace scheduler evaluation, we consider that all the nodes have 8 cores; i.e. the 64-node network has 512 cores while the 256-node network has 2048 cores. We have evaluated three different sets of traces, combining traces from the five applications shown in Section 3.3 with three different task sizes: 128, 256 and 512 tasks. Every trace set has the same number of applications (15 traces: 5 applications multiplied by 3 task sizes). Since Graph500 application generates the highest network load, with the purpose of stressing the network performance and its power saving mechanism, each set includes extra instances of Graph500 traces. In particular, the three sets of traces considered are the following:

- **Set A**: Namd, Gromacs, Linpack, MPI Random Access and Graph500.
- **Set B**: Gromacs, Linpack, MPI Random Access and Graph500 ($\times 2$).
- **Set C**: Linpack, MPI Random Access and Graph500 ($\times 3$).

Finally, note that since the scheduler randomly selects traces for execution, we have averaged results from 30 different executions of each trace set and topology.

4. Evaluation

In this section, we show the results of the performance and energy consumption evaluation. Section 4.1 shows the results for each application individually executed, while Section 4.2 shows the results obtained using the trace sets and the scheduler.

4.1. Single trace evaluation

64-node network

Firstly, we describe the results obtained for 64-node networks. Figure 2 shows the $Runtime$, Run_{task} and E_{net} obtained for each application, while Figure 3 shows $E_{cluster}$ as a function of ω_{Snodes}. Note that $Runtime$ and E_{net} are normalized with respect to the reference network, while Run_{task} shows the average fraction of time that the tasks are running (i.e. tasks are not idle waiting for network communications) obtained in each simulation. Finally, remember that ω_{Snodes} indicates the fraction of power consumed by compute nodes while idle. That is, if $\omega_{Snodes} = 0$, we have the ideal case of totally energy-proportional nodes, while when $\omega_{Snodes} = 1$ we have the worst case: nodes always consume the same power regardless of their utilization.

Namd, Gromacs, Linpack and MPIRA applications obtain similar results. Since all the network topologies achieve the same performance, the network energy consumption directly depends on port ratio (shown in Table 2). The 2D torus consumes 71.5% of the reference network energy (or 50% with the power-saving strategies), while the 1D torus only consumes 43% (or 32% with the power-saving). The aggregated-link torus networks only have a little performance penalty in the MPIRA application (less than 1% in the 2D torus and 3% in the 1D torus). Moreover, notice that using the power-saving strategy has no significant impact on performance in these applications.

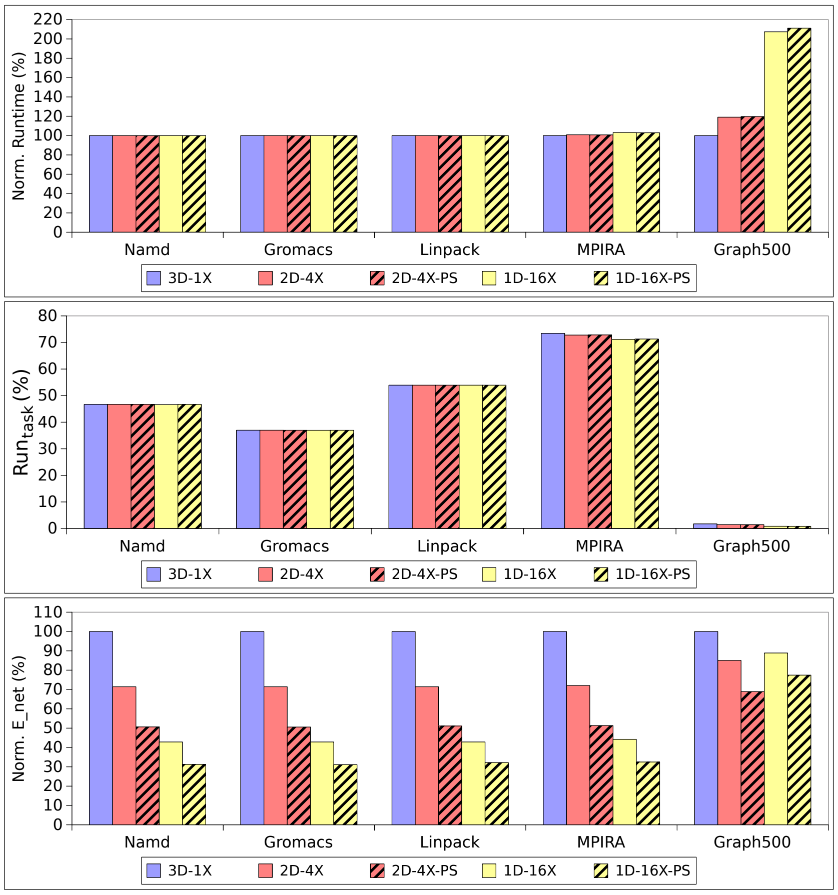
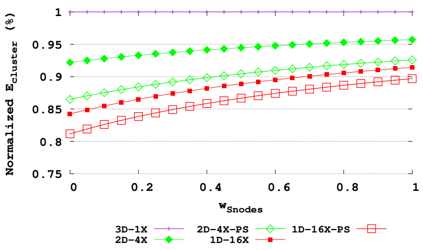


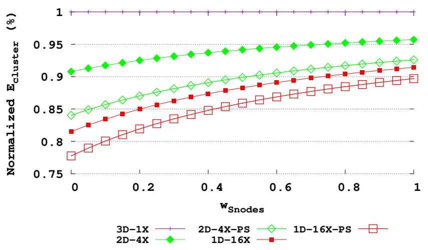
Figure 2: Runtime, Run_{task} and E_{net} obtained for 64-node networks

However, there are some differences when we analyze $E_{cluster}$. In Namd, Gromacs and Linpack, with no significant performance penalty, $E_{cluster}$ depends on the port ratio, with the 1D torus network consuming the lowest energy followed by the 2D torus. But the total saved energy also depends on the node utilization. As it can be seen, the applications with lower Run_{task} can save more energy, depending on how energy-proportional the nodes are. This is very reasonable: if Run_{task} decreases, the node energy consumption also decreases, and, as a consequence, E_{net} represents a more significant fraction of $E_{cluster}$.

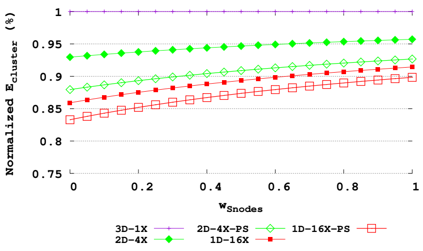
Finally, the Graph500 application provides captivating results. As shown in the figures, the network is the bottleneck of the system since the nodes



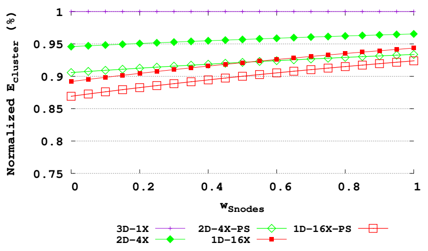
(a) Namd



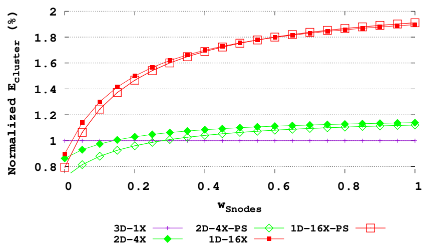
(b) Gromacs



(c) Linpack



(d) MPI Random Access



(e) Graph500

Figure 3: $E_{cluster}$ for 64-node networks

are idle most of the running time. The execution time on the 1D torus is unacceptable (around 208∼218%, depending on the case). Although E_{net} is reduced due its reduced port ratio, $E_{cluster}$ also greatly increases.

The 2D torus, considering the network subsystem alone, achieves significant energy savings (14% – 31%), but incurs in performance penalty (19%), with full system energy efficiency highly relying on compute nodes efficiency. This topology saves energy when the compute nodes are energy-proportional. Note that ω_{Snodes} must be lower than $0.2 \sim 0.4$, depending on the case, to save energy. For higher values of ω_{Snodes} the energy consumption of the full system increases due to the extended runtime when applying power saving techniques to the network.

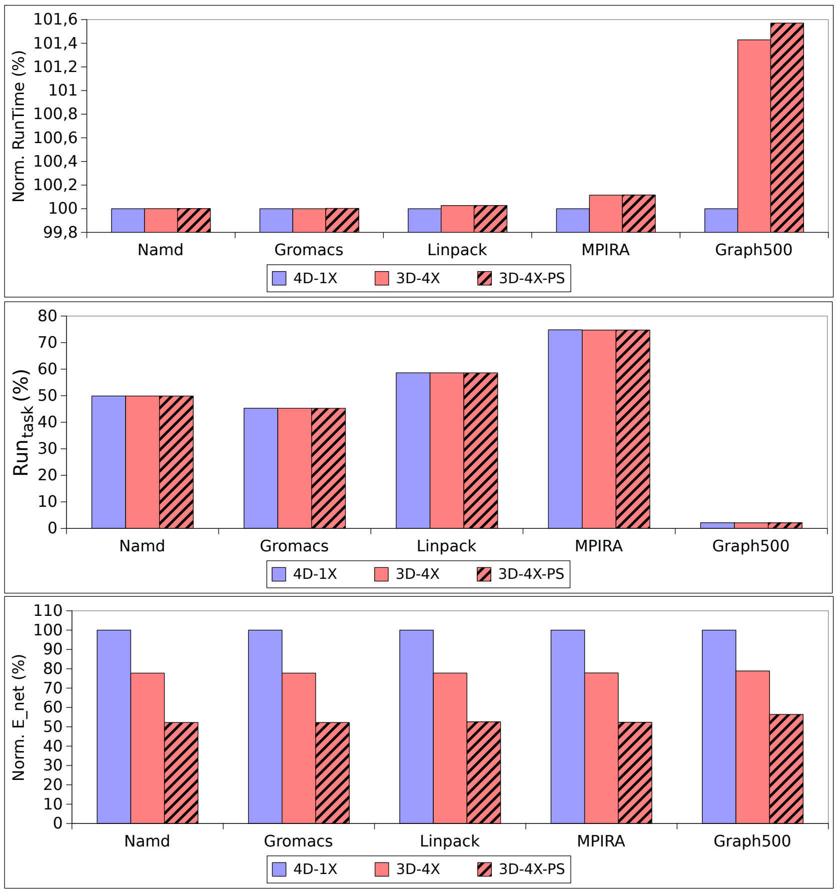
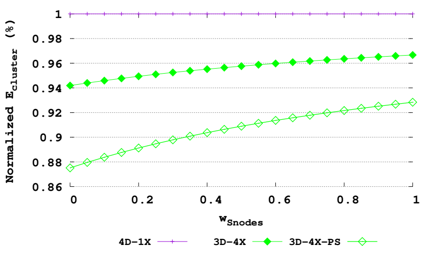
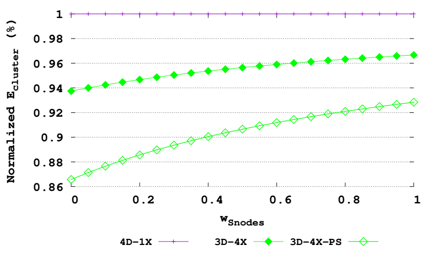
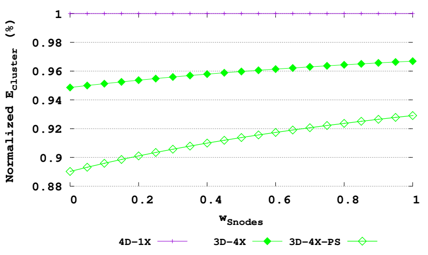


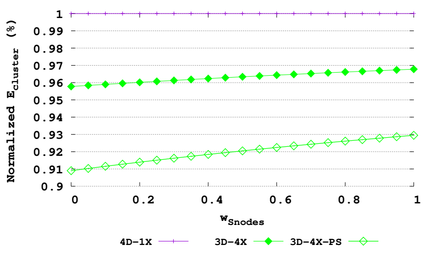
Figure 4: Runtime, Run_{task} and E_{net} obtained for 256-node networks

In summary, the 1D torus is the worst design option. Although this network can save energy under low network loads, the performance penalty and the energy consumption increase is unacceptable under high loads. Regarding the 2D torus, it obtains significant energy savings under low loads. Under high loads (illustrated with Graph500), energy efficiency of a system with 2D torus network depends on the energy-proportionality of compute nodes since no power can be saved on a network that is working close to its full capacity. However, the energy savings on the low load scenarios could compensate the energy increase on the high load scenario.


(a) Namd


(b) Gromacs


(c) Linpack


(d) MPI Random Access

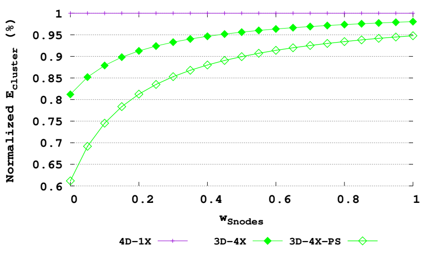

(e) Graph500

Figure 5: $E_{cluster}$ for 256-node networks

256-node network

In this section we present the results obtained for 256-node network configurations. Figure 4 shows the *Runtime*, Run_{task} and E_{net} obtained for each application, while Figure 5 shows $E_{cluster}$.

The results are similar to the ones shown for 64 nodes. All the networks achieve the same performance under Namd, Gromacs, Linpack and MPIRA applications, with energy consumption depending on the port ratio and the usage of the power saving mechanism. For the Graph500 application, the 3D torus only has 1.5% performance penalty, reducing the energy consumption at network and system level regardless of the value of ω_{Snodes}.

To sum up, the aggregated-link 3D torus using the power saving mechanism achieves the best results with energy savings at network level of up to 50% that translate into 5% to 40% at system level. And, as in the 64-node case, using the power saving mechanism has no significant impact on performance.

4.2. Trace scheduler evaluation

Finally, we show the results obtained by running three application sets using the trace scheduler. Figure 6 shows the *Runtime*, Run_{task} and E_{net} obtained for each trace set, while Figure 7 shows results for the overall system energy $E_{cluster}$.

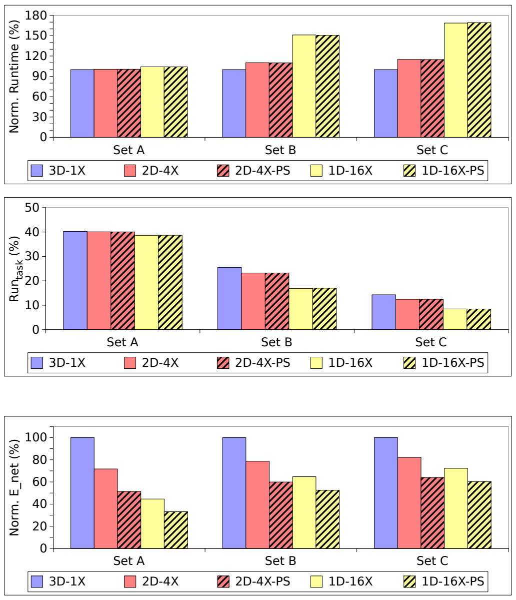
64-node network

As shown in Figure 6a, the results are consistent with the ones obtained in the single trace evaluation. For the application Set A, which injects the lowest network load, the energy consumption mainly depends on the network port ratio of the network under test with respect to the reference network and the usage of the power saving mechanism. Using a 2D torus has no significant impact on performance, even when activating the power saving mechanism; while the 1D torus increases the execution time around 4.5%.

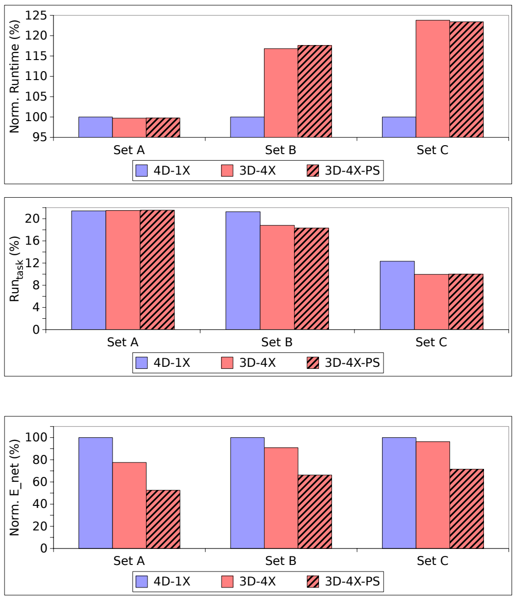
When the network load increases, the performance penalty of the 1D torus is huge, with execution times of 150% for Set B and 170% for Set C, approximately. This great increase in execution time makes the 1D torus to consume much more energy than the reference network (3D torus). Only when the compute nodes are very energy-efficient ($0 \leq \omega_{Snodes} < 0.15$) these networks (1D-16X and 1D-16X-PS) can save energy, but the performance penalty is still unacceptable.

The 2D torus also achieves lower performance than the reference network. The performance penalty is 10% for Set B and 15% for Set C, although the 2D tori (2D-4X and 2D-4X-PS) consume less energy ($60 \sim 64\%$ with power saving). As in the single trace evaluation, the 2D torus can save energy with respect to the reference network, even with high loads, depending on the value of ω_{Snodes}, saving energy when $\omega_{Snodes} < 0.7$ and $\omega_{Snodes} < 0.45$ for Sets B and C, respectively.

Summarizing, as observed for a single trace evaluation, the 3D torus (the reference network) is the best in terms of performance. The 1D torus is the worst design option due to its performance penalty. Again, the 2D torus achieves significant energy savings under low loads, but its energy efficiency



(a) 64 nodes



(b) 256 nodes

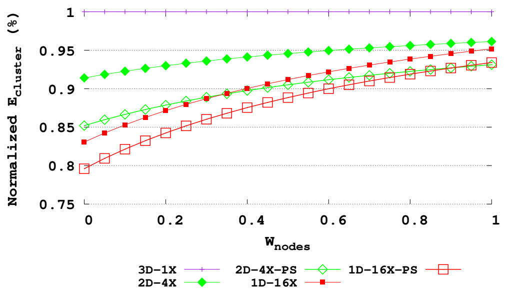
Figure 6: Runtime, Run_{task} and E_{net} using application sets and the trace scheduler

under high loads depends on the energy-proportionality of computing nodes (ω_{Snodes}). However, the values required for ω_{Snodes} to save energy are more moderate than the ones required for the networks evaluated only using the Graph500 benchmark. Finally, as expected, the energy saving achieved when applications inject low to moderate network load can compensate the energy increase generated by network-intensive applications as in Set B and Set C.

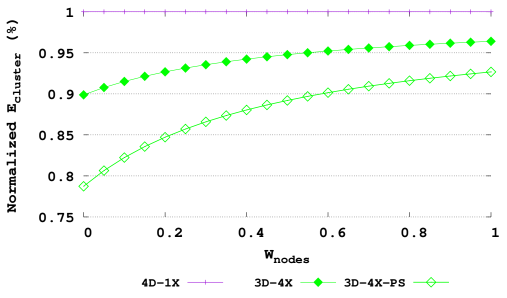
256-node network

The results for 256-node networks are similar to the ones obtained for 64 nodes. For Set A, there are no network with a significant performance penalty, and therefore, the energy consumption depends on the network port ratio with respect to the reference network and the activation of the power saving mechanism. When application sets are run on 3D torus significant performance penalties are observed for Sets B and C, although a significant amount of energy is saved at network level (in excess of 30%) It still requires energy-proportional nodes to save energy at system level, although requires more moderate values of ω_{Snodes} ($\omega_{Snodes} < 0.45$).

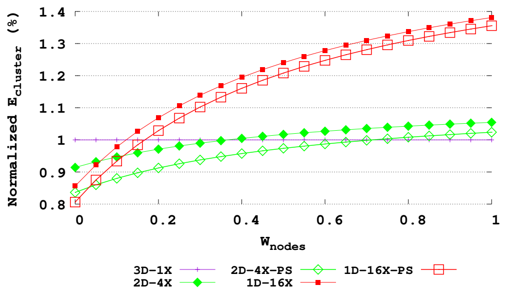
Summarizing, 3D torus networks with aggregated links and a power saving mechanism based on on/off links are a good configuration for low and



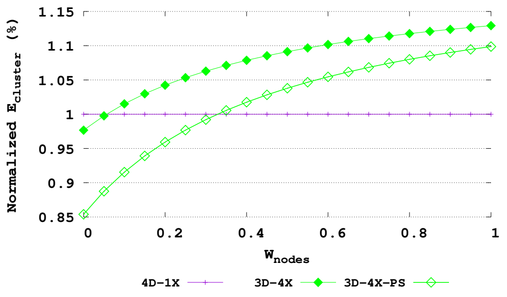
(a) Set A: 64 nodes



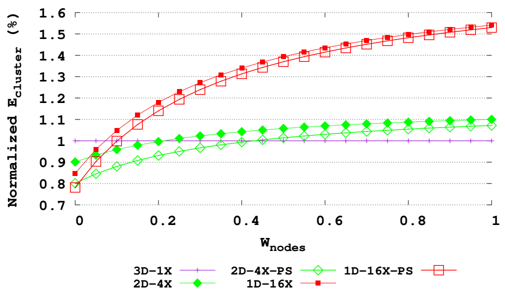
(b) Set A: 256 nodes



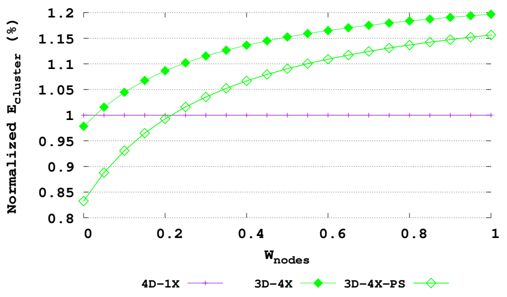
(c) Set B: 64 nodes



(d) Set B: 256 nodes



(e) Set C: 64 nodes



(f) Set C: 256 nodes

Figure 7: $E_{cluster}$ using application sets and the trace scheduler

moderate network loads. When network resources are going to be fully utilized at any given time there is no opportunity to save energy by switching off unused resources. In that case, the topology with best performance (4D torus in our experiments) also provides the best results in terms of energy consumption.

5. Related work

Although power consumption in HPC interconnection networks has not received much attention, several works have proposed alternatives to improve energy efficiency of interconnection networks.

Using a dynamic power management (DPM) mechanism is proposed in [16] for mesh topologies. Depending on network utilization, interconnection links are turned on or off. Traffic is redirected using alternative routes when specific links are turned off making use of a deadlock-free, fully-adaptive routing algorithm that guarantees packet arrival to destination. Significant power savings with moderate impact on performance indicate that more efficient dynamic switchable link designs would be critical.

Dynamic Adjusting Link Width (DALW) [40] dynamically sets the available network bandwidth as a function of the network traffic. Unlike DPM that completely switch links off when they are not fully utilized, DALW is based on reducing link bandwidth by narrowing link width. As the network topology is not modified, the same routing algorithm can be used simplifying the router design. Significant power consumption reduction can be achieved but message latency for low loads increases.

Work done in [41], for torus networks, takes advantage of the availability of high-degree switches to connect them through several links (i.e., trunk links) and apply power consumption reduction techniques switching off links for low loads, as long as network connectivity is guaranteed, (i.e. every pair of switches should be connected through at least one active link). A set of utilization thresholds is used to control on/off links. In [17], the authors propose a method to reduce interconnect power consumption by merging two techniques for network topologies based on aggregated links: firstly, dynamically switching on and off network links as a function of traffic; secondly, dynamically reducing the link bandwidth, with low traffic. As in the case of DALW, an advantage with respect to DPM is that the topology of the network is not modified so the same routing algorithm can be used.

Alonso et al. [42] propose a mechanism to reduce power consumption in fat-tree networks while guaranteeing network connectivity. Simulation results show that to obtain a significant network power consumption reduction with minimum performance degradation is possible . In [43] the authors improve the mechanism by defining a dynamic behavior that considers the switch status to modulate the sensitivity to traffic variations. The aggressiveness of the power reduction strategy can also be set. This solution significantly outperforms their previous proposals without additional cost.

Gunaratne et al. [44] investigate adaptive link rate to reduce the energy consumption of an Ethernet link by adaptively varying the link data rate depending on utilization. Output buffer queue length thresholds and utilization monitoring are used to set data rate. An evaluation using an analytic model and simulation using synthetic traffic patterns shows that an Ethernet link can operate at a low data rate most of the time yielding to significant energy savings with small impact on packet delay.

Totoni et al. [45] propose a hardware support for turning off interconnection links in software when they are not used during parallel applications execution and their management using adaptive runtime systems. Their proposal is evaluated by simulation for torus and dragonfly topologies.

PerfBound and DynamicFastwake mechanisms were presented in [46] to minimize interconnect link energy consumption. A performance overhead bound is introduced to dynamically manage on/off based networks. The techniques use local information already available at network switches and interfaces and require no change to the application and no communication with nodes and switches.

6. Conclusions

This paper explores the performance/energy dilemma in current high-performance computing systems with focus on torus network topologies. We have provided evidence that an appropriate selection of the interconnect configuration design parameters is a key issue to make an efficient use of a large scale computing platform. Our experiments, conducted on a selection of real application traces, represent typical high-demanding computing scenarios. We have evaluated network performance/efficiency balance at network level and complete system level. Various configurations of torus networks, for the same system size and bisection bandwidth, have been tested. We explored the impact of aggregating links and implementing an dynamic on/off link power saving mechanism. As observed in our results, using a power saving mechanism always reduces energy consumption at network level. When the full system is considered, two main factors determine what network topology is more energy efficient at system level: the regular network load and the energy efficiency of the compute nodes.

Under low to moderate network load scenarios, the best option is to implement an aggregated-link torus. Tori with aggregated links have not significant impact on performance and lead to the lowest power consumption, getting the best results on energy consumption. In addition, using power-saving mechanisms based on on/off links, provides significant power saving with minimum performance penalties.

However, the results change under high loads. In this case, the best performance is obtained by the torus with the highest number of dimensions, which was used as the reference network in all the experiments. When the load is very high, neither topological organization nor link on/off mechanism aiming at saving power provides benefits. The most performing topology, with no power saving mechanism, has been shown to be also the most energy-efficient: it uses all network resources, all the time at full power and the run time is typically lower than the more energy-friendly configurations we tested.

The aggregated-link torus topologies require very energy-efficient compute nodes (low ω_{Snodes}) to save energy. These topologies offer the best trade-off between performance and energy consumption. As indicated, only in the scenario with the highest network load, the most energy efficient network is the reference topology, but in the remaining scenarios, the aggregated-link torus becomes the best design option.

Finally, notice that, in all the evaluated aggregated-link torus networks, applying power saving strategies significantly reduces energy consumption without incurring in significant performance penalties. That is, independently of the torus network selected for our cluster, using a power-saving mechanism is a good idea.

For future work, we plan to evaluate larger systems, also including more topologies in the performance analysis, like fat-trees or dragonflies, since most of the available studies in the literature only compare the different topologies from a performance point of view.

Acknowledgment

This work has been supported by the Spanish MINECO and European Commission (FEDER funds) under project TIN2015-66972-C5-1-R and project TIN2015-66972-C5-2-R. Francisco J. Andújar is also funded by the Spanish MINECO under a Juan de la Cierva grant FJCI-2015-26080.

References

[1] J. Duato, S. Yalamanchili, L. Ni, Interconnection networks. An engineering approach, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[2] H. Fröning, M. Nüssle, H. Litz, C. Leber, U. Brüning, On achieving high message rates, in: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, 2013, pp. 498–505.

[3] Cray Inc, Cray CS Series Specifications., `https://www.cray.com/sites/default/files/resources/CrayCS400-ACBrochure.pdf` (2017).

[4] D. Chen, et al., The IBM Blue Gene/Q interconnection network and message unit, in: 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2011, pp. 1–10.

[5] TOP500 homepage, `https://www.top500.org`, (Accessed December 20, 2021).

[6] Graph500 homepage, `https://graph500.org`, (Accessed December 20, 2021).

[7] W. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2003.

[8] W. J. Dally, Performance analysis of k-ary n-cube interconnection networks, IEEE Trans. Computers 39 (6) (1990) 775–785.

[9] A. Agarwal, Limits on interconnection network performance, IEEE Transactions on Parallel and Distributed Systems 2 (4) (1991) 398–412.

[10] S. L. Scott, J. R. Goodman, The impact of pipelined channels on k-ary n-cube networks, IEEE Transactions on Parallel and Distributed Systems 5 (1) (1994) 2–16.

[11] S. Derradji, T. Palfer-Sollier, J. P. Panziera, A. Poudes, F. W. Atos, The BXI Interconnect Architecture, in: 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects, 2015, pp. 18–25.

[12] Mellanox infiniband switch systems, `http://www.mellanox.com/page/switch_systems_overview` (2017).

[13] Mellanox White Paper, Power saving features in Mellanox products, Tech. rep., Mellanox Technologies (2013).

[14] R. Alverson, D. Roweth, L. Kaplan, The Gemini system interconnect, in: IEEE 18th Annual Symposium on High Performance Interconnects (HOTI), 2010, pp. 83–87.

[15] N. McKeown, The iSLIP scheduling algorithm for input-queued switches, IEEE/ACM Transactions on Networking 7 (2) (1999) 188–201.

[16] V. Soteriou, L.-S. Peh, Dynamic power management for power optimization of interconnection networks using on/off links, in: 11th Symposium on High Performance Interconnects, 2003, pp. 15–20.

[17] M. Alonso, S. Coll, J.-M. Martínez, V. Santonja, P. López, J. Duato, Power saving in regular interconnection networks, Parallel Computing 36 (12) (2010) 696 – 712.

[18] F. Guo, O. Ormond, M. Collier, X. Wang, Power measurement of NetFPGA based router, in: 2012 IEEE Online Conference on Green Communications (GreenCom), 2012, pp. 116–119.

[19] K. Christensen, et al., IEEE 802.3az: the road to energy efficient ethernet, IEEE Communications Magazine 48 (11) (2010) 50–56.

[20] P. Reviriego, J. A. Hernandez, D. Larrabeiti, J. A. Maestro, Performance evaluation of Energy Efficient Ethernet, IEEE Communications Letters 13 (9) (2009) 697–699.

[21] M. Koibuchi, et al., An on/off link activation method for low-power ethernet in PC clusters, in: 2009 IEEE International Symposium on Parallel Distributed Processing, 2009, pp. 1–11.

[22] J. Li, W. Huang, C. Lefurgy, L. Zhang, W. E. Denzel, R. R. Treumann, K. Wang, Power shifting in thrifty interconnection network, in: 2011 IEEE 17th International Symposium on High Performance Computer Architecture, 2011, pp. 156–167.

[23] F. Zahn, P. Yebenes, S. Lammel, P. J. Garcia, H. Fröning, Analyzing the energy (dis-)proportionality of scalable interconnection networks, in: 2nd IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era, 2016, pp. 25–32.

[24] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, H. Liu, Energy proportional datacenter networks, in: Proceedings of the 37th Annual International Symposium on Computer Architecture, ISCA '10, ACM, New York, NY, USA, 2010, pp. 338–347.

[25] A. Greenberg, J. Hamilton, D. A. Maltz, P. Patel, The cost of a cloud: Research problems in data center networks, SIGCOMM Comput. Commun. Rev. 39 (1) (2008) 68–73.

[26] L. A. Barroso, U. Hölzle, The case for energy-proportional computing, Computer 40 (12) (2007) 33–37.

[27] M. Yokokawa, F. Shoji, A. Uno, M. Kurokawa, T. Watanabe, The K-Computer: Japanese next-generation supercomputer development project, in: Proceedings of the International Symposium on Low Power Electronics and Design, 2011, pp. 371–372.

[28] B. Alverson, E. Froese, L. Kaplan, D. Roweth, Cray XC series network, Cray Inc., White Paper WP-Aries01-1112.

[29] M. Karol, M. Hluchyj, Queuing in high-performance packet-switching, IEEE Journal on Selected Areas 1 (1998) 1587–1597.

[30] T. Anderson, S. Owicki, J. Saxe, C. Thacker, High-speed switch scheduling for local-area networks, ACM Transactions on Computer Systems 11 (1993) 319–352.

[31] N. Adiga, et al., Blue Gene/L torus interconnection network, IBM Journal of Research and Development 49 (2) (2005) 265–276.

[32] S. Q. Zheng, M. Yang, Algorithm-hardware codesign of fast parallel round-robin arbiters, IEEE Transactions on Parallel and Distributed Systems 18 (1) (2007) 84–95.

[33] VEF traces homepage, `http://www.i3a.uclm.es/VEFtraces/`, (Accessed December 20, 2021).

[34] F. J. Andújar, J. A. Villar, J. L. Sánchez, F. J. Alfaro, J. Escudero-Sahuquillo, VEF Traces: A Framework for Modelling MPI Traffic in Interconnection Network Simulators, in: The 1st IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era, Chicago, IL, USA, 2015, pp. 841–848.

[35] E. Gabriel, et al., Open MPI: Goals, concept, and design of a next generation MPI implementation, in: Proceedings of the 11th European PVM/MPI Users' Group Meeting, 2004, pp. 97–104.

[36] GALGO - Supercomputer Center of Albacete Research Institute of Informatics homepage, `https://www.i3a.uclm.es/i3a_t/galgo-supercomputing/`, (Accessed December 20, 2021).

[37] J. C. Phillips, et al., Scalable molecular dynamics with NAMD, Journal of Computational Chemistry 26 (16) (2005) 1781–1802.

[38] S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel, B. Hess, E. Lindahl, Gromacs 4.5: a high-throughput and highly parallel open source molecular simulation toolkit, Bioinformatics 29 (7) (2013) 845–854.

[39] HPC challenge benchmark homepage, `http://icl.cs.utk.edu/hpcc/index.html`, (Accessed December 20, 2021).

[40] M. Alonso, J.-M. Martínez, V. Santonja, P. López, Reducing Power Consumption in Interconnection Networks by Dynamically Adjusting Link Width, in: Lecture Notes in Computer Science, Vol. 3149, Springer-Verlag, 2004, pp. 882–890.

[41] M. Alonso, J. M. Martinez, V. Santonja, P. Lopez, J. Duato, Power saving in regular interconnection networks built with high-degree switches, in: 19th IEEE International Parallel and Distributed Processing Symposium, 2005, pp. 5b–5b.

[42] M. Alonso, S. Coll, J. M. Martinez, V. Santonja, P. Lopez, J. Duato, Dynamic power saving in fat-tree interconnection networks using on/off links, in: Proceedings 20th IEEE International Parallel Distributed Processing Symposium, 2006, pp. 8 pp.–.

[43] M. Alonso, S. Coll, J. Martínez, V. Santonja, P. López, Power consumption management in fat-tree interconnection networks, Parallel Computing 48 (C) (2015) 59–80.

[44] C. Gunaratne, K. Christensen, B. Nordman, S. Suen, Reducing the energy consumption of ethernet with adaptive link rate (alr), IEEE Trans. Comput. 57 (4) (2008) 448–461.

[45] E. Totoni, N. Jain, L. V. Kale, Toward runtime power management of exascale networks by on/off control of links, in: 2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum, 2013, pp. 915–922.

[46] K. P. Saravanan, P. Carpenter, Perfbound: Conserving energy with bounded overheads in on/off-based hpc interconnects, IEEE Transactions on Computers (2018) 1–1.