Design Exploration of Multi-tier interconnects for Exascale systems

DOI:
10.1145/3337821.3337903

Document Version
Accepted author manuscript

Design Exploration of Multi-tier interconnects for Exascale systems

Javier Navaridas
University of Manchester
Oxford rd
Manchester, UK
javier.navaridas@manchester.ac.uk

Josh Lant
University of Manchester
Oxford rd
Manchester, UK
joshua.lant@manchester.ac.uk

Jose A. Pascual
Univ. of the Basque Country
Manuel Lardizabal Ibilbidea
Donostia, Spain
joseantonio.pascual@ehu.es

Mikel Luján
University of Manchester
Oxford rd
Manchester, UK
mikel.lujan@manchester.ac.uk

John Goodacre
University of Manchester
Oxford rd
Manchester, UK
john.goodacre@manchester.ac.uk

ABSTRACT

Interconnection networks are one of the main limiting factors when it comes to scale out computing systems. In this paper, we explore what role the hybridization of topologies has on the design of an state-of-the-art exascale-capable computing system. More precisely we compare several hybrid topologies and compare with common single-topology ones when dealing with large-scale application-like traffic. In addition we explore how different aspects of the hybrid topology can affect the overall performance of the system. In particular, we found that hybrid topologies can outperform state-of-the-art torus and fattree networks as long as the density of connections is high enough–one connection every two or four QFDBs seems to be the sweet spot–and the size of the subtori is limited to a few nodes per dimension.

CCS Concepts

•Computer systems organization → Interconnection architectures; •Networks → Network architectures;

Keywords

Interconnection networks, Performance evaluation, Simulation

1. INTRODUCTION

Our society has come to depend on information and computer systems as the platforms where we carry out most of our activities. This has both driven forward the development of a plethora of IT technologies and motivated the construction of increasingly larger computing facilities. Indeed, we are relentlessly approaching the ExaScale Milestone, in which a single computer system will be able to execute a mind-blowing 10^{18} instructions per second. This magnificent computing power is required both in business and scientific contexts. For instance, in the context of business-centric computing, companies require increasingly higher computing power to support operations such as mining data from service records, offering on-line services and supporting increasingly large amounts of data. If we look at the world's largest companies it is speculated that they have hundreds of thousands of servers to sustain their infrastructure. As examples of well-known companies, Google may have around one million servers scattered among 13 datacentres worldwide whereas Amazon may have roughly half a million servers in 7 datacentres around the world.

We can find an analogous trend within the scientific community, with systems of similar sizes and an always-increasing greed for more computing power. In this context, applications typically used number-crunching approaches – e.g. computer simulation of various natures (molecular dynamics, finite elements or weather modelling, to cite a few) which are carried out using increasingly finer-grain models to improve accuracy which in turn require of greater computing power. However, in recent years, the advent of data analytics technologies has opened new avenues for scientific research. The highest exponent of data analytics in science is the Large Hadron Collider at CERN, which generates data at a stunning rate of 50 Petabytes per year. In order to be able to analyse all the generated data a Grid-like system with over 150 computing centres all over the world is used (See Worldwide LHC Computing Grid website[1]). This data generation rate will be dwarfed by the Square Kilometer Array project which is expected to generate a mind-blowing amount of data exceeding the Exabyte per day once it is built by 2020 (See Square Kilometre Array website[2]). At any rate, the advent of data analytics within the scientific community has motivated the convergence of datacentre and HPC architectures.

In all these kinds of systems the interconnection network (IN, hereafter) – a specific-purpose network that allows compute nodes to interchange messages with high throughput

ICPP '19 Kyoto, Japan

DOI:

[1]Available at: http://wlcg.web.cern.ch/

[2]Available at https://www.skatelescope.org/

and low latency – is a key element. This is specially true for large-scale computing platforms because its performance has a definite impact on the overall execution time of applications, particularly for those that are fine-grained and communication intensive. Indeed, INs are widely acknowledged (e.g., [12, 1, 23, 26]) to be one of the limiting factors for scaling up computing systems, essentially because the communication and synchronisation penalties suffered by applications increase with the size of the system. Current trends show the number of nodes used in data centre networks or supercomputers can be hundreds of thousands [2, 15, 16, 7] and these numbers are expected to increase over the millions in the next decade as anticipated by [23].

In this paper, we argue that with the uncontrolable growth of networks in terms of endpoints, standard topologies such as the ones that are in use in most systems today will not be able to cope with such systems either because of a lack of scalability or practicality. Therefore, we will need to look for new network arrangements that allow to simplify the design and deployment of systems without sacrificing their performance. Our objective is to analyse the suitability of hybrid topologies for large-scale systems as well as to understand the trade-offs involved in the design of such networks. We then study possible alternatives for a large-scale system based around the ExaNeSt technologies and some topologies of interest. In particular, we want to explore different parameters affecting the hybridization of topologies so to find the best trade-off when nesting the rigidity of the Lower levels (backplane-connected torus) with the flexibility offered by our FPGA-based routers [8] in the higher levels. This flexibility would allow us to adopt different topologies, but for simplicity we will stick to the well known fattree topology. For completeness and, given that we have several spare uplinks, we will also explore the suitability for our workloads of interest of the generalized hypercube topology.

With these in mind, we investigate which is the sweet spot when it comes to separating the different levels. In other words, we want to know to what extent we can expand the Torus topology without severely affecting the performance. In addition, we look into how dense the connectivity to the upper levels of the network needs to be in order to sustain adequate performance, i.e., which proportion of our Quad-FPGA daughter boards (QFDBs) should have their uplink active. This is important, because it will have a definite impact on the amount of hardware required to implement the main interconnect and in turn on cost and power.

To study these aspects, we start our research with an analysis of the topologies where we show the number of network components required by each configuration and from these, we estimate the cost and power overheads imposed by the extra components required to form the higher level of the network. From this analysis we see that there is little difference between using a fattree or a generalised hypercube in the higher levels. We also look at the distribution of distances and see that the generalised hypercube provides shorter paths by a slight margin. From our simulation results we confirm that the difference between the two arrangements is minimal except for specific workloads. We also found that, for most workloads, a higher density of connections is generally better, but that slightly reducing the density to 1 connection every 2 or even 4 QFDBs does not affect performance significantly, except for the heaviest loads, so there is room for reducing the cost and power of the system by thinning the connections. Note this process is akin to the oversubscription mechanisms commonly implemented in large-scale computing platforms.

It is also apparent from our results that keeping the subtorus small would be necessary, since in general the configurations with the largest subtorus required of higher execution times. While this can be explained by a larger number of network hops to reach the destination, it is somewhat unexpected as a larger subtorus should also reduce the pressure in the higher level network as locality is increased. It seems, though, that this effect does not have a significant effect when compared with the impact of the increase in path length.

Apart from helping us to understand the trade-offs involved in the selection of the size of the subtorus and the connection density, our experiments also help us find some interesting case studies, where some of the workloads are obtaining some unexpected results. We proceeded to analyse these unexpected results thoroughly.

The rest of the paper is organized as follows. Section 2 discusses the research on scalable topologies that has been carried out by the community and introduces the most typical topologies that we can find in high performance computing sites and datacentres. Then, Section 3 provides a high level description of the architecture of the system we are developing and justifies the decisions taken in the design of the interconnection network. Section 4 discusses our experimental environment including our simulation framework, the workloads we use in our study, the topologies that we consider and the connection rules employed to define different levels of density. From there we move to look at the results in Section 5, starting with a revision of the topological characteristics including an estimation of the power and cost increase and a measure of the average and maximum distances for each configuration. From there we move to look at our simulation results where we look at the workloads considering the effect that they have in the network and discuss the obtained results in detail. Finally we conclude the paper with some concluding remarks and a discussion of our next research objectives.

2. RELATED WORK

Given the large interest that network architectures arouse, there is a large body or research on different approaches to organize the interconnection network of computing systems, and great emphasis is put on the scalability of the designs.

For example many existing computing systems use the well-known torus topology [6, 14, 19, 7, 2] which has been historically used to interconnect massively parallel processors and has a massive body of research around it. Nodes in a torus are arranged in a d-dimensional grid with wrap-around links, so system deployment is trivial and there is no need for extra cabinets for the interconnect. However, the torus topology does not scale very well as distance related properties increase with the number of connected nodes. Note that the underlying topology of our system is based on small subtorus for exactly this reason.

Another common family of networks revolve around **Tree-like** topologies such as the k-ary n-tree topology [30] (fat-tree), the k:k'-ary n-tree topology [28] (thintree) or the generalized tree topology (gtree). These are multi-stage networks that can provide very high throughput and even non-blocking behaviour and thus, became the architecture of choice as soon as the technology enabled the use of large-

radix switches.

One of the latest network organizations that is getting a great interest from the community is the **Dragonfly** topology. This is a large-radix, low-diameter, recursive topology that uses a group of high-radix routers (a network group) as a virtual router to increase the effective radix of the network. Large numbers of these network groups are interconnected in an all-to-all fashion using a given connection rule [22, 13]. Many of these connection rules do exist and they have a great impact on the performance of the IN. While Dragonflies can sustain very high throughput for average cases, they are very sensitive to the communication patterns of the applications and there exist many pathological scenarios that will reduce greatly the performance of the interconnect, primarily with unbalanced loads.

Jellyfish is a random network designed to provide high connectivity in datacenters and was demonstrated to be able to outperform tree-like topologies in some scenarios. One of the main characteristics of Jellyfish is that it is incrementally expandable – i.e., does not require any specific size, as opposed to most common topologies which are restricted to relative few configurations [32]. However, one of the main drawbacks of Jellyfish is its lack of structure which brings many challenges from a practical perspective in terms of routing, physical layout, and wiring.

Another topology that has shown to have great potential for large scale systems is the **Generalised Hypercube** (GHC) topology [5, 36]. While the study of these topologies has typically been performed from a more formal perspective, there have been many proposals for the deployment of Generalised Hypercube-based systems. One such proposal is **Bcube**, a server-centric network architecture designed for shipping-container modular datacenters [17]. Interestingly, the Stellar construction which was proposed to generate dual-port server-centric networks was firstly exercised with a Generalised Hypercube topology [10].

Still in the realms of server-centric datacentre networks, we can highlight **DPillar**, which is somewhat inspired by the classic butterfly topology [25] but with servers connected between each stage of the topology and wrapped around. DPillar provides several nice properties such as scalability, network performance, and cost efficiency, which make it suitable for building large scale future datacenters. However, the routing function that was proposed originally was shown to be non optimal [11].

The reader should note that all of these network arrangements are based on a single architecture and there is little attention within the community to hybrid networks. While, arguably, the Dragonfly could be considered a hybrid topology in that it is generated recursively – and this could also apply to many other recursive topologies, such as DCell [16], Ficonn [24], or the recursive diagonal torus [35] – there is little effort on investigating hybrid systems in which different topologies are used at different levels of the interconnection network. One such example is **HCN/BCN**, a recursively-defined family of networks, where the BCN construct is built using (copies of) HCNs by including an additional layer of interconnecting links [18]. However, this is a highly theoretical approach and no clear pathway to a practical deployment can be anticipated from the authors discussion.

The research work that we perform here aims to fill this gap by looking at the hybridization of common topologies in the context of a real prototype system and driven by the physical constraints of the devices and boards developed within ExaNeSt. Our Mezzanine boards enforce the use of a torus topology in the lower level of our communication infrastructure and, in this paper, we will explore some promising candidates for being implemented in the higher levels: a fattree and a generalized hypercube – both discussed above.

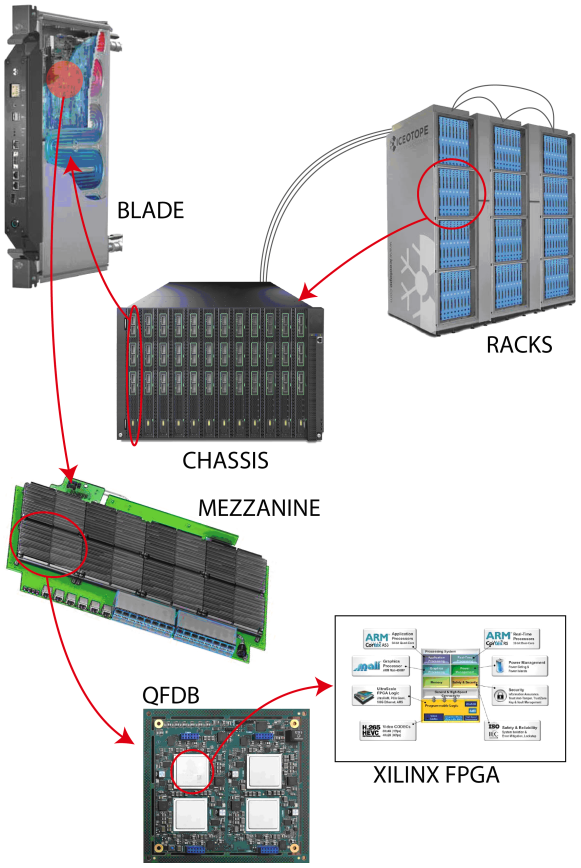


Figure 1: Overview of ExaNeSt architecture.

3. SYSTEM ARCHITECTURE

In this Section we describe the main characteristics of our novel architecture, ExaNeSt—see [21, 4] for a more detailed description. The architecture has recently been showcased by means of a small, 2-cabinet prototype held at Foundation for Research and Technology - Hellas (FORTH). Our system will require millions of low-power-consumption ARM+FPGA MPSoCs to reach Exascale and includes a unified, low-latency interconnect and a fully distributed storage subsystem with data spread across the nodes using local Non-Volatile Memory (NVM) storage [34], using BeeGFS[3], an state-of-the-art parallel file-system that has been ported to our architecture. The ultimate objective being to provide near-data processing, which should be essential for emerging data-centric applications.

As shown in Fig. 1, our building block is a small form-factor QFDB containing 4 Zynq Ultrascale+ MPSoCs[4]. Each of these QFDB is configured to use up to 10×communication

[3]Available at: https://www.beegfs.com

[4]See https://www.xilinx.com/support/documentation/white_papers/wp482-zu-pwr-perf.pdf

ports attached to 10Gbps transceivers. In addition we employ the FPGA fabric in the MPSoC to implement a proprietary network protocol developed by our group [3]. The system scales as follows: Up to 16 QFDBs can be connected within a *Blade* which uses a backplane that delivers high-bandwidth connectivity, whilst reducing the costs and power consumption of external cables and transceivers. Due to pragmatic reasons during the design phase of our architecture, the boards are arranged in the blade as a static 3D mesh-like structure ($4 \times 2 \times 2$) where each board uses 6 links for internal connection and exposes 4 links to the outside of the blade for uplinking. This mesh can be extended seamlessly to a torus topology using the external connectivity of the blades. One of the uplinks is dedicated for 10G Ethernet communications with the outside world, so we can use, at most, 3 links per QFDB to connect to the higher levels of our hybrid network [8]. With these constraints in mind, we need to stick to a subtorus as the low-level topology of our system and will look at well-known topologies for the upper levels: fattrees and generalised hypercubes, see below.

4. EXPERIMENTAL SET-UP

In this section we describe our experimental work. First we present the simulation environment and the application models that will be used to analyze the networks. Then we describe the networks we are considering in our study, the parameters of interest and the connection rules we use to explore the effect that the size of the subtorus and the density of connections have on the execution time.

4.1 Simulation Environment and Application Model

The evaluation has been carried out using our in-house developed simulator INRFlow [29], which models the behaviour of large-scale parallel systems, including the network topology (link arrangement), the workload generation and the scheduling policies (selection, allocation and mapping) and measures several static (application-independent) and dynamic (with applications) properties. During execution, the links of the network have capacities and each flow is specified with a bandwidth reflecting the data that must be routed while also respecting the causal relationships between network flows – i.e, some flows must finish before others are allowed to be injected. As a result, it provides realistic, flow-level simulation of general real-world workloads, as well as a good estimation of the completion times of a collection of application-inspired workloads.

Since one of our main objectives is to ensure the developed interconnect is, indeed, appropriate for large-scale systems as we approach exascale we will analyze our architecture with systems composed by over half a million Zynq FPGAs. Indeed, we need to ensure scalability and also to look for the best design decisions for the higher levels of our network. To obtain more realistic results, our experimental work considers a large collection of workloads, which are modelling real-world applications.

Collective operations: Collective operations are central to many parallel applications and, thus, are very useful to provide an estimation of the performance of applications. In this study, we consider two different collective operations. The first one, **Reduce**, models a non-optimized N-to-1 collective in which all nodes send messages to the root task. This generates a lot of congestion in the network around the root task, as all tasks are sending traffic towards the same point. While an optimized, logarithmic implementation would be preferred in a real system, we use the non optimized one to evaluate the behaviour of the networks under a pathological scenario with a hot spot. In addition, we also consider an optimised **AllReduce** collective operation based on a logarithmic implementation where only $log_2 tasks$ steps are needed [33].

MapReduce: MapReduce [9] is a common application in the Datacenter domain commonly used to analyze large amounts of data (big data analytics). In MapReduce, a root task partitions and distributes the original data amongst all servers. Once computing nodes receive data from the root, they perform the mapping of the data and shuffle it to the other servers in an all-to-all fashion and then send their results back to the root.

Sweep3D: The Sweep3D is a common function in the HPC domain for the deterministic particle transport problem [27]. It performs a wavefront communication in which a grid of tasks is traversed in diagonal starting from one of the corners and advancing in all 3 dimensions towards the opposite corner of the 3D grid.

Flood: This workload is similar to the sweep workload above, but sending data from a source node in all directions with several wavefronts being propagated at the same time, which exerts a much heavier pressure into the network.

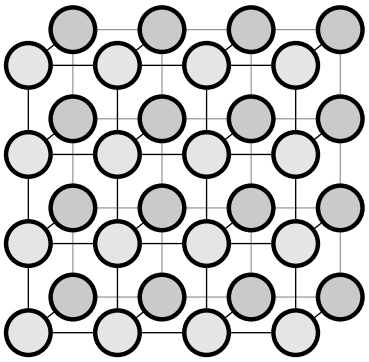
Near Neighbors: This workload has a communication pattern found in many HPC codes such as LAMMPS [31] (a consolidated molecular dynamics simulator) or RegCM (climate modelling) which are some of ExaNeSt's target applications. The tasks are arranged in a virtual grid and communicate only with their neighbors as defined by a stencil pattern.

n-Bodies: n-Bodies is another common application model, broadly used in different physics domains to model the interaction between bodies (particles, planets, etc). In our model, tasks are arranged in a virtual ring in which each task starts a chain of messages that travel clockwise across half of the ring.

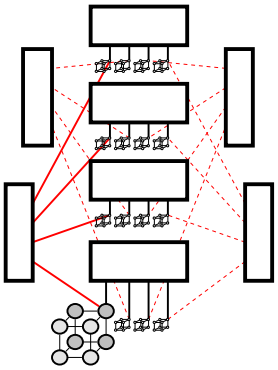
Unstructured applications: Unstructured workloads often arise when considering system management traffic, system runtimes based on work-stealing, or graph analytics applications (a key application area within datacenters). We model four such applications. First, **UnstructuredApp** is based on fixed-length messages to model an unstructured application in which data has been partitioned evenly across the tasks. The second one, **UnstructuredMgnt**, follows the traffic size distribution produced by the management software in common large-scale systems as described in [20]. The third one, **UnstructuredHR**, models an application in which a subset of the tasks is more likely to be targeted as destinations (*Hot tasks*). Finally, **Bisection** models an application in which the tasks perform pair-wise communications swapping pairs randomly every round.

4.2 Topologies

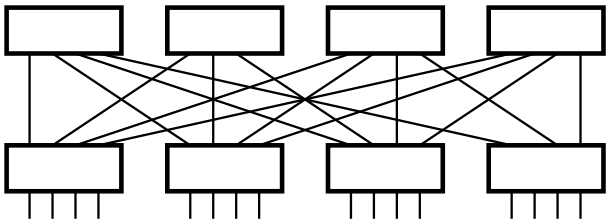
To perform the study, we have implemented in our simulator new topologies that hybridize a torus and nests it into a tree (NestTree) as well as a torus nested into a generalized hypercube (NestGHC). These topologies accept a large number of parameters to define the size of the torus, the number of connections within a torus to the upper network and the parameters of the upper network (levels/dimensions of the network + ports per switch per level). For simplicity, we will



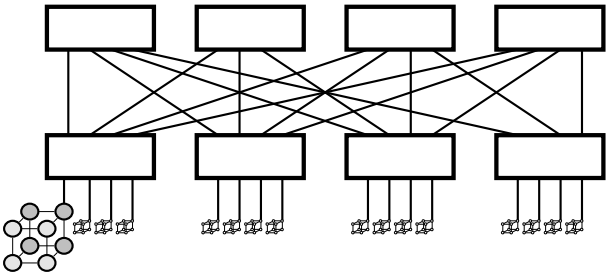
(a) Torus 3D ($4 \times 4 \times 2$). Wrap-around link not shown for the sake of simplicity



(b) A torus nested in a generalised hypercube, NestGHC, topology (t=2, u=8), 4-ary 2-GHC.



(c) A 4-ary 2-tree



(d) A torus nested in a fattree, NestTree, topology (t=2, u=8), 4-ary 2-tree

Figure 2: Examples of the topologies under study

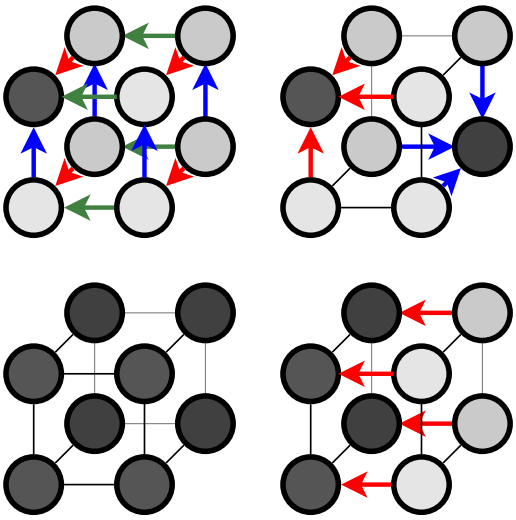


Figure 3: The 4 different densities considered in our study (top: $u = 8$ and $u = 4$, bottom: $u = 1$ and $u = 2$). Darker nodes represent uplinked nodes. The arrows show which paths are used to get to the uplinked nodes from the not connected nodes. Arrows of different colors have been used for the sake of clarity.

denote them as NestTree(t, u) and NestGHC(t, u), where t is the number of nodes per dimension in the subtorus and $\frac{1}{u}$ is the density of uplinks per subtorus, or in other words, there will be one uplink every u QFDBs. Note that it is anticipated that these two parameters, t and u, have a definite impact on the characteristics of the system, both in terms of performance and extra HW necessary to build the higher level of the interconnect. Understanding the trade-offs of these two parameters, is therefore, essential to design an effective interconnect.

In both cases, we implemented routing functions in which communications within a subtorus always stay within the subtorus, to reduce the pressure on the higher level. Communications between nodes in two different subtori are more complex as they require routing from the source to the closest uplinked node in its subtorus (could be the source itself) then route minimally within the upper network to get to the closest uplinked node to the destination (again, could be the destination itself) and, if needed route to the destination within its subtorus. Routing within a subtorus is performed using dimensional order routing. Routing within a tree uses minimal UP*/DOWN* routing. Finally routing in a generalized hypercube uses e-cube routing which traverses the generalized hypercube dimensions in order. We assume all links can transmit at 10Gbps similar to the transceivers available in our QFDBs.

In order to assess the suitability of hybridizing topologies we will compare our proposals, NestTree and NestGHC, with two standard topologies: (i) a 3D torus as this is the one we would implement if we deter ourselves from using the extra uplinks in our QFDBs to build a higher level network and (ii) a fattree topology as a representative of standard high performance topologies. The fattree is expected to operate very proficiently with most workloads due to its non-blocking nature. We have depicted some examples of these four network

architectures in Figure 2. Note that no over-subscription is applied to the fattrees under consideration (both on their own or nested). Also, we restrict our study to fattrees with three stages.

4.3 Connection rules

An important aspect of our hybrid topologies is the density of connections from the QFDBs to the higher levels of the network. In this paper, we will consider 4 different connection densities, depicted in Figure 3.

u = 1: The simplest case where all QFDBs have uplinks. There is no need for intra-torus communication when destinations are outside the subtorus.

u = 2: Only half of the nodes have a uplink connection. In this case, all non-connected nodes will have an uplinked neighbour at a single hop in the X dimension and there will be no contention for the use of subtorus resources.

u = 4: Only one of every 4 nodes has a connection. In this case, in order to be able to minimise the number of hops in the sub-torus, we connect the nodes in two opposite vertices of a $2 \times 2 \times 2$ subgrid, so that all the nodes are still one hop away from a connected node.

u = 8: The lowest density we consider with only one out of every 8 nodes having a connection and all the nodes will communicate through the root of a $2 \times 2 \times 2$ subgrid.

5. DISCUSSION OF RESULTS

In this section, we analyze the performance of large-scale ExaNeSt systems composed of 131,072 QFDBs (or around 50 cabinets). The number of endpoints is limited by the available memory in the servers where the experiments were conducted.

5.1 Topological characteristics

We open the section with a simple topological analysis of the topologies under consideration. Table 1 shows the average distance and diameter of the networks for uniform traffic which serves as a preliminary raw estimate of the achievable performance. Table 2 shows the number of extra switches required to conform the upper network topology as well as rough back-of-the-envelope estimations of the cost and power overheads that each of the topologies will entice, compared with a simple network using only the hardwired torus topology. It is worth noticing that neither the cost or the power overhead are really significant compared to the overall system, but still it will be preferable to reduce cost and power as much as reasonably possible.

If we compare the NestTree and the NestGHC, we can see that the two families of topologies have very similar characteristics, both in terms of extra hardware required and distance of the communications. As discussed above, it is clear that the amount of extra hardware required to build the topology greatly depends on the number of uplinks per subtorus, with very significant differences. We can also see that the average distance is greatly affected by the number of uplinks. However, notice that increasing the number of uplinks has a massive effect in terms of the number of extra switches required, so we need to find a trade-off between these two parameters. For this reason, we move now to evaluate how these characteristics affect the overall execution time of the workloads explained above.

5.2 Simulation work

Given the large variety of workloads we are employing to evaluate the systems and that performing a case by case analysis would be too cumbersome but provide little insight, we will split the discussion into 2 main types: (i) workloads generating a *heavy* utilization of the network, with long periods of congestion generated by a large proportion of the endpoints injecting traffic at once, and (ii) workloads with *light* network utilization, where inter-message causality limits the number of endpoints that inject traffic concurrently.

Let us start with the results of the heavy workloads as they are the most prominent, see Figure 4. With these workloads we can see that the simple torus topology fails to deliver appropriate performance (execution time is up to one order of magnitude slower than the fattree or the fastests of the hybrid topologies). This shows that going towards a multi-tier interconnect with a hybrid approach, instead of simply scaling the system by expanding the lower-tier torus, was a sensible design decision.

We can also observe that, provided that the uplink density is high enough, the hybrid approach is capable of outperforming the single fattree topology in most cases, albeit with a smaller margin. In addition, we can see that reducing density can have a severe effect in the performance, especially if the subtorus is too large (with worst-case scenarios where execution time is well over an order of magnitude slower). Indeed, we can see that increasing the size of the subtorus, generally increases the overall execution time.

If we compare the topology of the upper tier in our hybrid approach, we can see there is normally little difference between the performance of a fattree and the generalized hypercube. There are only two exceptions: The first one is bisection, where the fattree can deliver the workload much faster than the generalized hypercube. In contrast, UnstructuredHR executes quicker in the generalized hypercube than in the fattree, but to a lesser extent.

The results with lighter workloads, shown in Figure 5, are also of interest. For instance, we can see that in Sweep3D and Flood, some of the trends discussed above are inverted. The best performing topology is the torus because the topology matches better the grid-like nature of these two workloads and, therefore, it can outperform the other topologies. This is also apparent with our hybrid networks where having longer dimensions in the subtorus helps improving performance (although not to the same extent as they lose performance in the heavy workloads). This diverges with the results obtained with the Near Neighbors workload which, even when it has the same spatial pattern as Sweep3D and Flood, the torus topology still performed worse than the fattree and the best hybrid topologies because of the huge pressure placed into the network with all nodes sending at the same time.

If we move to MapReduce, it is another instance where the torus beats the other topologies, although by a slim margin. This is somewhat unexpected as the traffic pattern is not especially well suited for this topology. Even when this is the case for this workload, we can see that again, increasing the size of the subtorus in our hybrid networks is still counterproductive.

Finally, the Reduce collective is a special case where there is no noticeable difference between the different networks under evaluation. The reason for this is that the workload imposes congestion only in the root of the collective, which basically serializes packet delivery. In other words, the con-

(t,u)	Average distance		Diameter	
	NestGHC	NestTree	NestGHC	NestTree
$(2,8)$	8.75	8.88	12	12
$(2,4)$	7.31	7.44	8	8
$(2,2)$	6.84	6.97	8	8
$(2,1)$	5.87	5.98	6	6
$(4,8)$	8.69	8.87	12	12
$(4,4)$	7.31	7.44	8	8
$(4,2)$	6.84	6.97	8	8
$(4,1)$	5.87	5.98	6	6
$(8,8)$	8.72	8.87	12	12
$(8,4)$	7.32	7.44	11	11
$(8,2)$	6.85	6.97	11	11
$(8,1)$	5.88	5.99	11	11

Table 1: Average and Maximum distance for the topologies under evaluation.

(t,u)	Switches		Cost Increase (est.)		Power Increase (est.)	
	NestGHC	NestTree	NestGHC	NestTree	NestGHC	NestTree
$(2,8)$	2048	2048	1.17%	1.17%	0.39%	0.39%
$(2,4)$	3072	3072	1.76%	1.76%	0.59%	0.59%
$(2,2)$	5120	5120	2.93%	2.93%	0.98%	0.98%
$(2,1)$	8192	9216	4.69%	5.27%	1.56%	1.76%
$(4,8)$	2048	2048	1.17%	1.17%	0.39%	0.39%
$(4,4)$	3072	3072	1.76%	1.76%	0.59%	0.59%
$(4,2)$	5120	5120	2.93%	2.93%	0.98%	0.98%
$(4,1)$	8192	9216	4.69%	5.27%	1.56%	1.76%
$(8,8)$	2048	2048	1.17%	1.17%	0.39%	0.39%
$(8,4)$	3072	3072	1.76%	1.76%	0.59%	0.59%
$(8,2)$	5120	5120	2.93%	2.93%	0.98%	0.98%
$(8,1)$	8192	9216	4.69%	5.27%	1.56%	1.76%

Table 2: Number of switches and estimation of cost and power overhead for the topologies under evaluation.

sumption port at the root becomes the bottleneck, so the performance of the network does not affect the total execution time.

Overall, we can see that hybrid networks employing different layers with different topologies can outperform the other topologies in isolation. A single torus topology is not adequate for most workloads due to the large distance packets need to traverse and the high congestion that this can generate. The only cases where the torus outperforms the rest of the topologies is when the workload does not put too much pressure into the network, especially if it matches the topology, such as in Sweep3D and Flood.

While the fattree usually outperforms the torus, our torus + fattree and torus + GHC alternatives tend to perform better if the density of uplink connections is high enough. In the cases where uplink density is reduced, the congestion created around the uplink node severely affects performance, particularly for workloads imposing a huge pressure into the network. In our analysis, it seems like the inflection point is around a density of one uplink connection every 2 or 4 nodes. Going over that cripples the network significantly. Considering the cost and power savings that can arise it seems that a density of $\frac{1}{4}$ could be the sweet spot for this topology, although it might not be enough if very heavy workloads are expected to be run in the system. With regards what is the best topology to use in the higher tiers, no significant difference was found between the fattree and the generalized hypercube, except for specific workloads.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have investigated the hybridization of topologies starting from a static hardware-induced torus-like building block. Given that a torus substructure was imposed by pragmatical reasons related to the engineering of the boards, we decided to explore what was the best alternative for scaling up our platform rather than naively relying on a simple, relatively low-performance torus topology. To avoid the performance drop that we could expect from a torus topology, we decided to hybridise it by nesting it into a superior topology such as a fattree or a generalised hypercube.

With this in mind, we moved to understand what are the trade-offs and build upon the best practices to construct such a network. Particularly we wanted to know to what extent the underlying subtorus can be expanded without severely affecting the performance. Not only that, we also need to realize how connection density affects the overall performance and look for trade-offs that allow the network to sustain adequate performance while, at the same time, using as few resources as necessary. This is important, because it will have a definite impact on the amount of hardware required to implement the main interconnect and in turn on cost and power. Indeed, we looked a the cost and power of different system configurations and found out that the cost of deploying any of the hybrid topology discussed here is relatively small, but since our budget is limited, we still need to balance the amount of resources that are put into

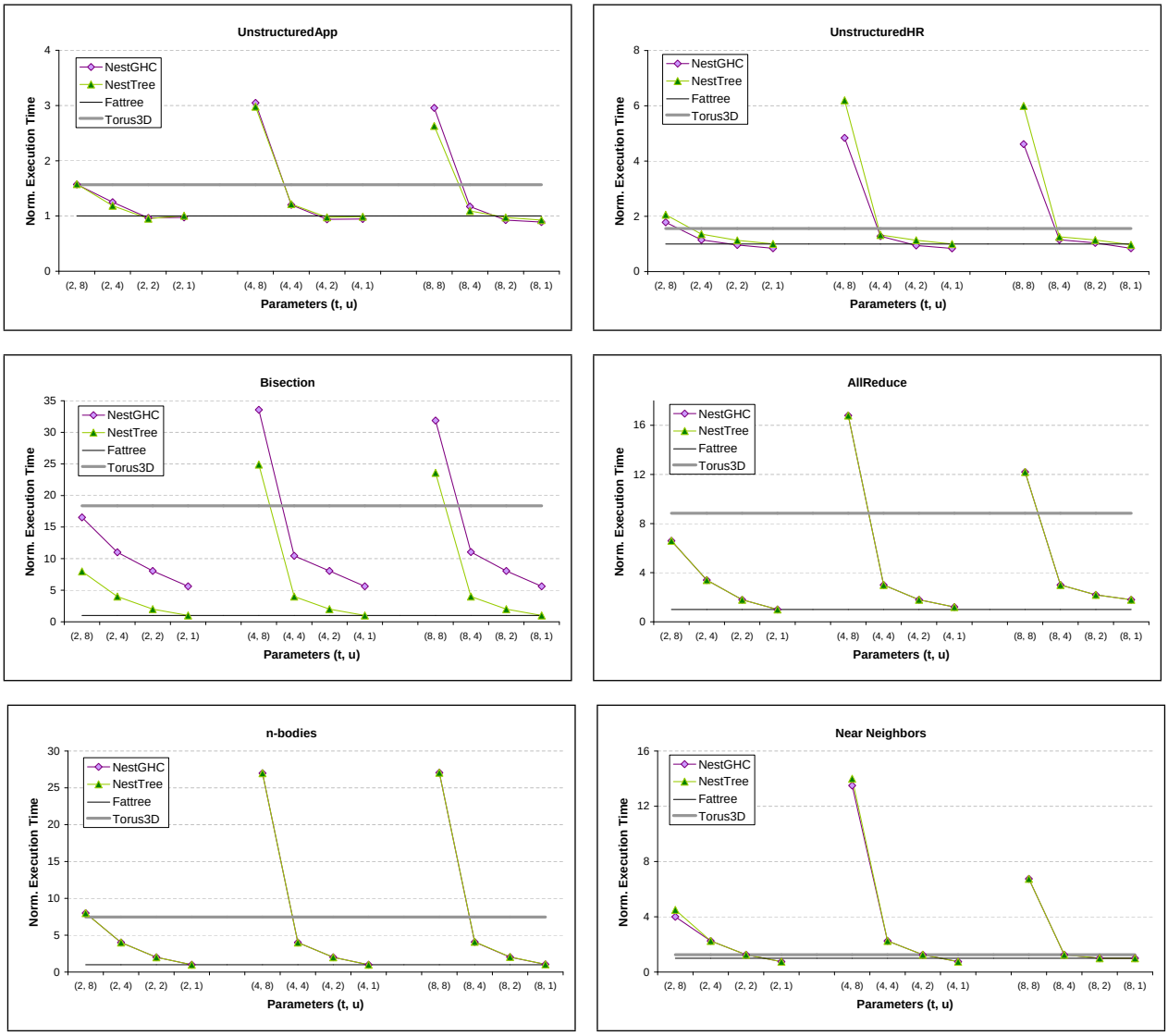


Figure 4: Normalized simulation time for the heavy workloads. We compare several instances, varying the nodes per subtorus, t, and the uplinks per subtorus, u.

the construction of the network and into the procurement of computing devices.

To study these aspects, we start with an analysis of the topologies where we show the number of network components required by each configuration and from them estimate the cost and power overheads imposed by the extra components required to form the higher level of the network. From this analysis we see that there is little difference between using a fattree or a generalised hypercube in the higher levels. We also look at the distribution of distances and see that the generalised hypercube provides shorter paths by a slight margin. From our simulation results we confirm that the difference between the two arrangements is minimal except for specific workloads. We also found that, for most workloads, a higher density of connections is generally better, but that slightly reducing the density to 1 connection each 2 or even 4 QFDBs does not affect performance significantly, so there is room for reducing the cost and power of the system by spreading the connections among the network.

Our results also suggest that keeping the subtorus small seems like a good idea, since in general the configurations with the largest subtorus required a much higher execution time. While this can be explained by a larger number of network hops to reach the destination, it is somewhat unexpected as a larger subtorus should also reduce the pressure in the higher level network as communication locality should be increased. It seems, though, that this effect is not as significant as the impact of the increase in path length.

As future work we will continue with the development of the different subsystems of our architecture. In particular, we are developing several low-level aspects of our NIC and routers to incorporate essential features such as mechanisms for fault tolerance, low-level bandwidth scheduling to give priority to critical flows, or a holistic method for congestion control. Furthermore we plan to look into other aspects of the scalability of the interconnect and how these mechanisms

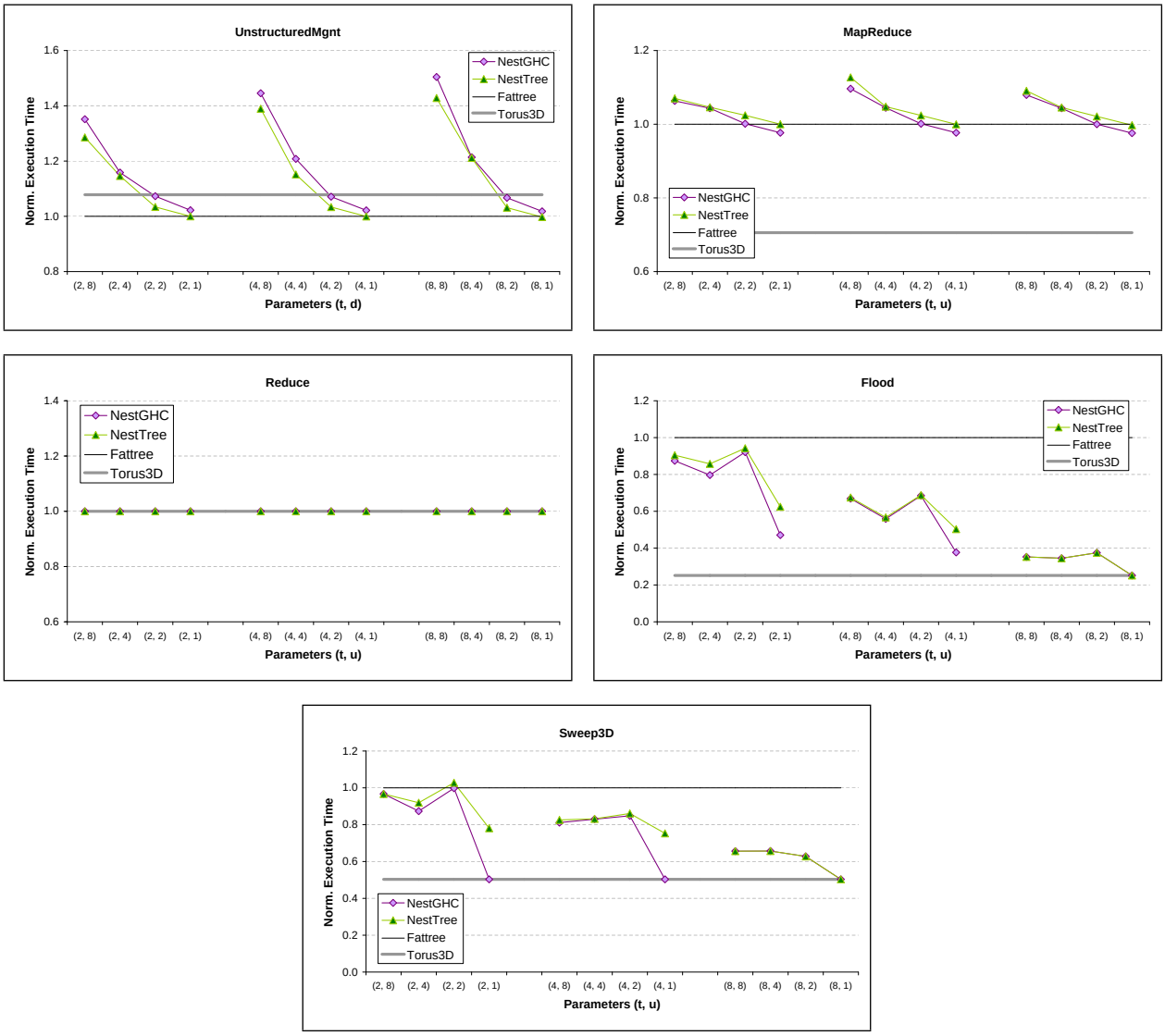


Figure 5: Normalized simulation time for the light workloads. We compare several instances, varying the nodes per subtorus, t, and the uplinks per subtorus, u.

affect the scalability of our approach. With relation to the nesting of topologies, we would like to extend our analysis so that it can incorporate some other aspects of the design such as energy consumption or fault tolerance. For this we will carry out a revamp of our simulation tools so to be able to perform energy estimation at the scale we are interested in. At a longer-term, once our prototype has incorporated enough nodes (at least a few hundreds of them, currently only a few tens have been deployed), we would like to perform a similar investigation as this one, but over the real prototype. While the results would not be as interesting as the ones presented here, given is small-scale, we expect that some insights can be discovered from empirical experimentation.

7. ACKNOWLEDGMENTS

This research work has been supported by ExaNeSt and EuroEXA which are funded by the European UnionâĂŹs Horizon 2020 programme under Grant Agreements No. 671553 and 754337.

8. REFERENCES

[1] D. Abts and B. Felderman. A guided tour through data-center networking. *Queue*, 10(5):10:10–10:23, May 2012.

[2] Y. Ajima et al. The Tofu interconnect. *Micro, IEEE*, 32(1):21 –31, 2012.

[3] R. Ammendola et al. Low latency network and distributed storage for next generation HPC systems: the ExaNeSt project. *Journal of Physics: Conference Series*, 898:082045, oct 2017.

[4] R. Ammendola et al. The Next Generation of Exascale-Class Systems: The ExaNeSt Project. volume 00, pages 510–515, Los Alamitos, CA, USA, 2017.

[5] L. N. Bhuyan and D. P. Agrawal. Generalized

hypercube and hyperbus structures for a computer network. *IEEE Trans. Comput.*, 33(4):323–333, Apr. 1984.

[6] R. Brightwell, K. Pedretti, and K. D. Underwood. Initial performance evaluation of the cray seastar interconnect. In *13th Symposium on High Performance Interconnects (HOTI'05)*, pages 51–57, Aug 2005.

[7] D. Chen et al. The IBM Blue Gene/Q interconnection network and message unit. In *2011 Intl. Conf. for High Performance Computing, Networking, Storage and Analysis*, pages 1–10, New York, NY, USA, 2011.

[8] C. Concatto et al. A cam-free exascalable hpc router for low-energy communications. In *Architecture of Computing Systems – ARCS 2018*, pages 99–111, Cham, 2018. Springer International Publishing.

[9] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[10] A. Erickson et al. The stellar transformation: From interconnection networks to datacenter networks. *Computer Networks*, 113:29 – 45, 2017.

[11] A. Erikson et al. An optimal single-path routing algorithm in the datacenter network dpillar. *I E E E Transactions on Parallel and Distributed Systems*, 28(3):689 – 703, 3 2017.

[12] J. Escudero-Sahuquillo and P. J. Garcia. High-performance interconnection networks in the exascale and big-data era. *The Journal of Supercomputing*, 72(12):4415–4417, Dec 2016.

[13] G. Faanes et al. Cray Cascade: A scalable hpc system based on a dragonfly network. In *Procs. of the Intl. Conf. on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 103:1–103:9, Los Alamitos, CA, USA, 2012.

[14] A. Gara et al. Overview of the Blue Gene/L system architecture. *IBM Journal of Research and Development*, 49(2.3):195 –212, march 2005.

[15] A. Greenberg et al. VL2: A scalable and flexible data center network. *SIGCOMM Comput. Commun. Rev.*, 39(4):51–62, Aug. 2009.

[16] C. Guo et al. DCell: A scalable and fault-tolerant network structure for data centers. In *ACM Conference on Data Communication*, SIGCOMM '08, pages 75–86, New York, NY, USA, 2008.

[17] C. Guo et al. BCube: A high performance, server-centric network architecture for modular data centers. In *ACM Conference on Data Communication*, SIGCOMM'09, pages 63–74, 2009.

[18] D. Guo et al. Expandable and cost-effective network structures for data centers using dual-port servers. *IEEE Transactions on Computers*, 62(7):1303–1317, July 2013.

[19] IBM Blue Gene team. Overview of the IBM Blue Gene/P project. *IBM Journal of Research and Development*, 52(1/2):199–220, 2008.

[20] S. Kandula et al. The nature of data center traffic: Measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, IMC '09, pages 202–208, New York, NY, USA, 2009.

[21] M. Katevenis et al. The ExaNeSt project: Interconnects, storage, and packaging for exascale systems. *2016 Euromicro Conf. on Digital System Design*, 00:60–67, 2016.

[22] J. Kim et al. Technology-driven, highly-scalable dragonfly topology. In *Procs. of the 35th Annual Intl. Symposium on Computer Architecture*, ISCA '08, pages 77–88, Washington, DC, USA, 2008.

[23] P. M. Kogge, P. L. Fratta, and M. Vance. Facing the exascale energy wall. In *2010 International Workshop on Innovative Architecture for Future Generation High Performance*, pages 51–58, Jan 2010.

[24] D. Li et al. Scalable and cost-effective interconnection of data-center servers using dual server ports. *IEEE/ACM Transactions on Networking*, 19(1):102–114, February 2011.

[25] Y. Liao et al. DPillar: Dual-port server interconnection network for large scale data centers. *Comp. Networks*, 56(8):2132–2147, May 2012.

[26] O. Lysne et al. Interconnection networks: Architectural challenges for utility computing data centers. *Computer*, 41(9):62–69, Sept 2008.

[27] O. M. Lubeck et al. Implementation and performance modeling of deterministic particle transport (sweep3d) on the IBM Cell/B.E. *Scientific Programming*, 17:199–208, 01 2009.

[28] J. Navaridas et al. Reducing complexity in tree-like computer interconnection networks. *Parallel Computing*, 36:71–85, 02 2010.

[29] J. Navaridas et al. INRFlow: an interconnection networks research flow-level simulation framework. *Journal of parallel and distributed computing.*, January 2019.

[30] F. Petrini and M. Vanneschi. K -ary n -trees: High performance networks for massively parallel architectures. In *Proceedings of the 11th International Symposium on Parallel Processing*, IPPS '97, pages 87–, Washington, DC, USA, 1997.

[31] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1 – 19, 1995.

[32] A. Singla et al. Jellyfish: Networking data centers randomly. In *Procs. of the 9th USENIX Conf. on Networked Systems Design and Implementation*, NSDI'12, pages 17–17, 2012.

[33] R. Thakur and W. D. Gropp. Improving the performance of collective operations in MPICH. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 257–267, 2003.

[34] Q. Xu et al. Performance analysis of NVMe SSDs and their implication on real world databases. In *Procs. of the 8th ACM Intl. Systems and Storage Conf.*, pages 1–11, New York, NY, USA, 2015.

[35] Y. Yang et al. Recursive diagonal torus: An interconnection network for massively parallel computers. *Parallel and Distributed Systems, IEEE Transactions on*, 12:701 – 715, 08 2001.

[36] S. D. Young and S. Yalamanchili. Adaptive routing in generalized hypercube architectures. In *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing*, pages 564–571, Dec 1991.