Article

HyperZ: Hyper Z-Tree Topology

Mo Adda, IEEE Member

University of Portsmouth, School of Computing, Buckingham Building, Lion Terrace, Portsmouth PO2 3HE, United Kingdom E-mail: mo.adda@port.ac.uk

Abstract: Multi-level direct networks fueled by the evolving technology of active optical cables and increasing pin-bandwidth achieve reduced diameters and cost, with high-radix switches. These networks, like Dragonfly, are becoming the preferred aspirants for extreme high-performance parallel machines such as exascale computing. In this paper introduces a Hyper Z-Tree topology, which deploys the Z-Tree, a variant of fat-tree, as a computing node of a Generalized Hypercube (GHC) configuration. The resulting configuration provides higher bisection bandwidth, lower latency for some applications and higher throughput. Furthermore, the levels of the fat tree offer several path diversities across the GHC dimensions, hence conceding a more fault tolerant architecture. To profit from these path diversities, Two adaptive routing algorithms are proposed in this paper, which are extensions to the routing algorithm suggested in HyperX topology. These two algorithms exhibit better latencies and throughput than the HyperX.

Keywords: Adaptive routing; deterministic routing; connectivity; Hypercube; fat-tree; deadlock; high switch-radix

1. Introduction

The next generation of supercomputers will face more challenges to tackle far more complex and larger programming problems and open new horizons in the scientific, engineering and business disciplines [1], [2], [3]. However, an increase in the computational speed of processors can lead to extreme power constrains [4]. Interconnecting a larger number of chips, while keeping the overall cost under budget and delivering the required performance, is a trade-off. A need to interconnect those high-speed processing elements in the best manner possible is a challenging task [5]. While the technology for higher speed processors keeps advancing, the search for resilient interconnection networks of high performance computing lags behind and shows trifling contributions [6].

Fat tree topologies [7], with their simplistic properties and ability to scale, are deemed to be the most popular solution for computing clusters, data centers and HPC [8], [9], [10]. They can be seen as universal networks that emulate any other topology deploying the same amount of resources with slight reduction in the power consumption. The high path diversity in fat trees, with its fault tolerant property, plays a critical role in the overall performance of the communication networks. One interesting approach is hybrid networks which involve the incorporation of fat tree design as building blocks within large scale parallel systems. This is the inspiration of the research presented in this paper. There are several hybrid interconnections that use different topologies at each level as in BlueGene IBM [11], [12], [13], Dragonfly [14], [15], X-Mesh [16] and Torus [17]. Furthermore, newer technologies on optical networks and photonic switches are emerging. These technologies have shown potential to attain higher bandwidth and effective power consumption. The papers [18], [19] show the possibility to interconnect fat tree topologies by photonic routers. Although issues such as crosstalk that limit the scalability of these systems can be an obstacle. Current research is advancing in this direction to provide effective solutions. Our proposed architecture takes advantage of photonic switches of any radix. Fat trees can also be considered a candidate for wireless networks in HPC environments. Initial work has been done in this direction [20] to execute parallel tasks on wireless nodes with up to 8 transceivers acting as nodes' degree. The major constraints of on-chip radio devices are the limitation of the transmission rate and high-power consumption which

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 June 2023 doi:10.20944/preprints202306.1262.v1

make them less favorable for modern parallel platforms. However, with the new 802.11ac and beyond, the speed of the WiFi is reaching the 7Gbps and over.

This paper considers hosting the Z-tree [21] as nodes for a generalized hypercube network (GHC) [22]. The choice of a GHC is procured based on the scalability of the architecture and its small radius. Its high degree per node is compensated for by the provision of the side links (global links) offered by the switch of the Z-Tree. This approach has already been adopted by HyperX topology [23], [24]. However, HyperX offers a single switch of high radix per node. This makes, as this paper proves later on, HyperZ a generalized version of HyperX. The proposed approach outperforms the HyperX and allows a larger design space to select optimal topologies that answer the demands of parallel applications.

This paper contributes by developing a new multi-level direct network based on a Z-tree as a computational node for a GHC interconnect. The topology is viewed as an extension to HyperX. It accommodates far more processors with smaller switch radix compared to the HyperX of the same size. This design relying on the number of levels of the Z-Tree offers high bisection bandwidth, lower latency, and multiple concurrent communications between nodes of the GHC. To take advantage of this structure, two adaptive routing algorithms which are more efficient and fault tolerant than HyperX are suggested. The algorithm exploits the path diversities constructed from the Hyper Z-Tree.

2. Related Work

Symmetric Tori connected Torus Network (STTN) [26] is a hierarchical interconnection network of 2D Torus hosting iterative nodes formed from other tori. STTN architecture is a good solution for moderate and small high performance systems due to its scalable nature as the node degree is always fixed. However, for large scale networks, the diameter is inevitably large which leads to higher latencies, with large bisection bandwidth. Torus embedded hypercube interconnect proposed by Kini [27] is a network architecture that can offer scalability in parallel computation. However, for this solution to work, the size of hypercube is required to stay constant at all times to avoid node degree changes. But for large scale systems this will not be the case, and the diameter has to increase to fit the size of the network. The hierarchical network (HTN) consists of 3D-tori modules connected together by a 2D torus network. The Pruned HTN version [28] removes links along the dimensions to reduce complexity, and it achieves better results than the original HTN in terms of power consumption, latency and bandwidth. The dimension order routing is adopted to route messages. However, the dimension routing algorithm is not adaptive and it does not exploit all the paths within the network, hence putting a limit on the performance for large-scale parallel systems. Hyper nodes torus is an architecture that extends torus interconnection by replacing each node in the torus with a ring [29]. A distributed environment is laid down as each node can be either a memory or a processor. An optimal version of the XY-routing was proposed to take into consideration the routing within the ring. The results presented show the hyper node torus achieves lower latency than the hypercube and torus for 16, and 64 nodes. There is no evidence that the proposed system will work for large-scale networks, as the size of the ring has to be kept to a minimum, which is not always the case with higher number of processors.

Recently, a lot of interest has been devoted to multi-level direct networks as promising scalable topology for high performance computing such as PETCS [30], Dragonfly [14], [15] and Cray [30]. These networks built from high-radix switches give the impression of an all-to-all interconnection. PERCS high performance interconnect aims to improve the bisection bandwidth for high performance supercomputers challenge. The hub chip comes with a large number of local and remote links fully equipped with routing components, so it requires no external switches or routers. Similarly, to Dragonfly and Cray, PERCS implements 2-level direct-connect. Unlike PERCS, Cray and the Dragonfly networks offers the possibility to emulate any topologies within the groups. The full connectivity at level 1 and 2 of the supernodes in PERCS brands it as non-scalable for large size networks. PERCS also uses 3 hopes routing for inter and intra supernodes communications for normal traffic conditions. For hot-spot traffic, the routing algorithm deviates the path to a random

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 June 2023 doi:10.20944/preprints202306.1262.v1

supernode and then routes to the destination making a path of 5 hopes. The dragonfly is a hierarchical topology that reduces the network diameter and the number of long cables. It includes multiple groups that are connected by all-to-all links. Each groups can have different internal topology, and connects fully to other groups via global long optical channels, when it is possible.

HyperZ, presented in this paper, and HyperX belong to these classes of networks but using hypercube (GHC) rather than all-to-all interconnections to maximise the bisection bandwidth, and minimise the diameter and the latency, at the expense of more cables. The cost of the external cabling can be justified by the usage of optical connections. HyperZ, like others, is driven by the technology evolution in active optical cables and the increase in pin-bandwidth that is high radix switches with large number of narrow ports. HyperZ uses Z-tree as a computing node. The levels or height of the Z-tree add extra global cables but create parallel layers that segregate traffic and achieve higher bandwidth and lower latencies. The HyperX has only 1 level, and therefore a single layer.

3. Hyper Z construction

The HyperZ is a generalized hypercube interconnection whose processing nodes are fat trees. The fat trees have the specifications of the Z-tree [21]. A regular Z-node has equal number of routing nodes (switches) of equal radix at all levels of the tree. Unlike HyperX which can be viewed as flattened butterfly or hypercube with nodes containing single switches with several processing elements, our proposed architecture is more general and includes also HyperX configuration in its space when the height of the Z-node is 1. A group of Z-nodes in the k^{th}-dimension is refere to as a super node, S-node, where k is $1 \le k \le d$ with d representing the number of dimensions in the GHC topology.

The definition of the HyperZ of dimension d and height h can be written as

$$\boldsymbol{H_{d,h}\,(s_1, \ldots, s_d;\, Q_1, \ldots, Q_d)\,(Z_h)} \tag{1}$$

The specification of $\boldsymbol{(Z_h)}$ is defined in [21], as $\boldsymbol{Z_h\ (z_1, \ldots, z_h;\ r_1, \ldots, r_h;\ g_1, \ldots g_h)}$, where h is the number of levels (height), z_i of zones, r_i switches, and g_i of internal links in the Z-tree at level i. The vector $\boldsymbol{Q_k = (q_{1k}, \ldots\ q_{ik}, \ldots, q_{hk})}$ is the set of external or side connections in the k^{th}-dimension of GHC, $1 \le k \le d$, where q_{ik} is the number of links between two z-nodes along the dimension k, at level i of the Z-Tree, $1 \le i \le h$. The vector $\boldsymbol{(s_1, \ldots, s_d)}$ symbolizes the number of S-Nodes in each dimension, 1 to d. For convenience, when all the links at each level of the Z-nodes are equal, the parameter, $\boldsymbol{Q_{k,}}$ is replaced by $\boldsymbol{q_k}$. The HyperX topology has only one level, a height of $h=1$, and a single switch per Z-node. As shown in Fig. 1, at level 2, the HyperZ can be seen as 2-parallel layers of HyperX. This configuration will increase the bandwidth and accommodate far more processing nodes while reducing the connectivity – ports numbers - per routing nodes.

In general, the total number of nodes (processing, $i=0$ or routing, $i \ge 1$, nodes) in the Z-node, R_i, and HyperZ, N_i, can be expressed as

$$\begin{aligned} R_i &= r_i \prod_{j=i+1}^{h+1} z_j \\ N_i &= R_i \prod_{k=1}^{d} s_k \end{aligned} \tag{2}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 June 2023 doi:10.20944/preprints202306.1262.v1

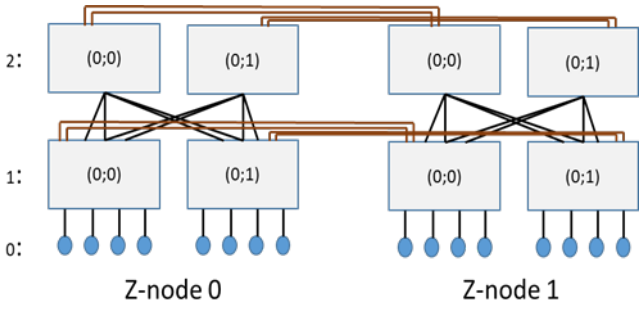


Figure 1. HyperZ of dimension, d=1, and height, h=3, $H_{1,3}(2;2)(Z_2)$ with Z_2 (4,2;1,2;1,2). There are (q_1=2) external links per level per switch.

3.1. Hyper Z-Fat Tree Connectivity Rules

Each node at level i, $0 \le i \le h$, within the Z-node, can be represented by a flat address, X_i, where $X_i \in \{0, 1, \ldots, R_i\text{-}1\}$. On the other hand, each node in the HyperZ can be extended by the tuple $(a_d, a_{d-1}, \ldots, a_1; X_i)$ where $a_k \in \{0, 1, \ldots, s_k\text{-}1\}$ for $1 \le k \le d$. The hierarchical address $(a_d, a_{d-1}, \ldots, a_1)$ is also an abstract address of a Z-node in the HyperZ. For instance, in Fig. 1, at level 2 there are 2 routing nodes, R_2=2, in the Z-node 0 and Z-node 1. Thus, the tuple (a_1, X_2) has the flat address $X_2 \in \{0, 1\}$, and the hierarchical address $a_1 \in \{0, 1\}$ since $s_1 = 2$, giving the tuples (0; 0), (0; 1) for the Z-node 0 and the tuples (1; 0), (0; 1) for the Z-node 1 at level 2.

The levels offer the traffic several paths from each Z-node to the others across the network. Thus, the communications take place concurrently though the layers. In Fig. 2, there are 2 layers corresponding to 2 levels of the Z-node. Each layer has several paths emerging from the links of each routing node, switch.

The Z-tree bidirectional connectivity can be established by first instantiating all the nodes (processing node i=0 and routing nodes i>0), assigning them unique flat address X_i and then linking them internally according to the rule of the Z-tree connections [21]. The inter-connections of the Z-nodes in the GHC network along each dimension can be established as stated below.

A node A of tuple $(a_d, a_{d-1}, \ldots, a_1; X_i)$ connects by bidirectional links to a node B of tuple $(b_d, b_{d-1}, \ldots, b_1; Y_i)$, if and only if:

1. They have the same flat address within their Z-nodes $X_i = Y_i$,
2. The vector connectivity degree $Q_k = (q_{1k}, q_{2k}, \ldots, q_{ik} \ldots, q_{hk})$ in the k^{th}-dimension has at least one element $q_{ik} \ne 0$,
3. They differ in exactly one value in the k^{th} dimension, $a_k \ne b_k$ for all $1 \le k \le d$.

In Fig. 2 illustrates an example on how Z-nodes are connected in a HyperZ topology define as $\boldsymbol{H_{3,h}(4,3,4;1,1,1)(Z_h)}$. In this example, a Z-node is an abstract computational node, $\boldsymbol{(Z_h)}$, of any height and internal structure. In the dimension 1, 2 and 3, there are 4, 3 and 4 S-nodes, s_1 =4, s_2=3 and s_3=4, respectively. All the connections between the Z-nodes in all the dimensions at all levels are equal to 1, q_k= 1. Dimension 0 is the Z-node itself, shown as a filled triangle. In dimension 1, there are four S-nodes, labeled 0, 1, 2, and 3, each containing 1 single Z-node. In dimension 2, there are three S-nodes, labelled 0, 1, and 2, each containing four Z-nodes making a total of twelve Z-nodes. Finally, in dimension 3, there are four S-nodes, labelled 0, 1, 2, and 3, each accommodating twelve Z-nodes. The Z-node (0, 0, 0) connects to Z-nodes (0, 0, 1), (0, 0, 2) and (0, 0, 3) for all levels i as they differ in dimension 1. It also connects to Z-nodes (0, 1, 0) and (0, 2, 0), which differs in dimension 2, and finally connects to Z-nodes (1, 0, 0), (2, 0, 0) and (3, 0, 0) by differing in dimension 3.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 June 2023 doi:10.20944/preprints202306.1262.v1

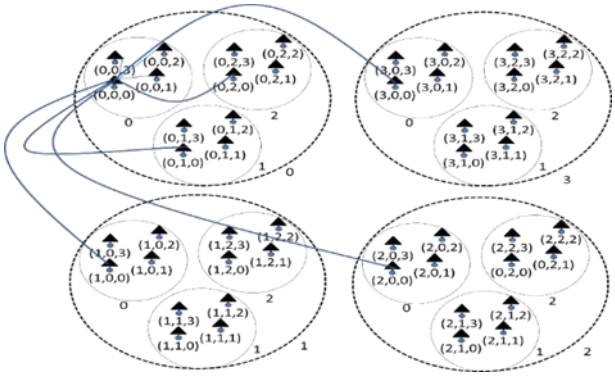


Figure 2. Hyper Z-Tree, $H_{3,h}$ (3, 4;*1, 1, 1*) (Z_h) , the degree of connectivity vector is 1 in all levels of Z-tree. Only the connections from Znode (0, 0, 0) to all others are shown.

3.2. Topology Features

This section describes the extent of the topology and its design space to accommodate large number of processors.

Definition 3.1

A regular HyperZ is a topology with $z_i = z$ for all $1 \le i \le h$ and $s_k = s$ and $q_k=q$ for all $1 \le k \le d$.

This means that the regular HyperZ is constituted from Regular Z-nodes. Furthermore, if the number of zones in the Z-nodes is equal to the number of S-nodes at any levels and dimensions, a fully regular HyperZ is constructed.

Definition 3.2

A fully regular HyperZ is a topology with $z_i = s_k = s$ and $q_k=q$ for all $1 \le i \le h$ and $1 \le k \le d$.

For a fully regular HyperZ to accommodate N_0 processors, there should be at least R_0 processors per regular Z-node, with each switch radix limited to, $2z + dq(z-1) \le L_{max}$, where L_{max} is a given technology dependent switch radix, q the number of side links between Z-nodes and z is the number of zones within the Z-node, yielding $2z$ ports per switch. Thus, the number of S-nodes ($s=z$) required in all dimensions is bounded by the radix available

$$z \le \frac{dk + L_{min}}{dk + 2} \tag{3}$$

A lower bound to the minimum number of processors accommodated by the HyperZ can therefore be expressed as $z^h s^d \ge R_{0min}$, refer to Eq. 2, since z=s for a fully regular hyperZ, one can write

$$z \ge (R_{0min})^{1/(h+d)} \tag{4}$$

The equations (3) and (4) establish both an upper and lower bound for each possible network configuration. It is apparent from Fig. 3 that topologies with lower height such as HyperX, to maintain fixed number of ports per routing node, they required higher dimensions, and therefore higher diameter leading to higher latencies. HyperZ with higher height comparable to $h=3$ and *8* S-nodes would interconnect a minimum number of processors ranging from 131072, with yet smaller number of ports per routing node in the order of 32, with GHC of 3 dimensions, see Fig.3. This is an important result to minimise complexities of switches and thus reduce power consumption.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 June 2023 doi:10.20944/preprints202306.1262.v1

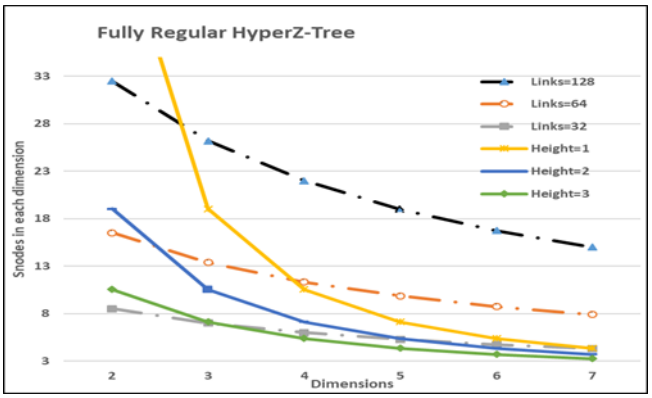


Figure 3. HyperZ design space for R_{0min} =131072 processors.

4.. Adaptive Routing in HyperZ

The routing in the HyperZ is considered within the Z-node travelling through the levels and outside the Z-node travelling through the dimensions. In the up direction, the messages travel to one of the common levels before making their way down to the destination processor if it destination is within the same Z-node. If the destination of the message is another Z-node, the message crosses the dimensions of the GHC connections, by following the offset dimensions though the aligned routing nodes belonging to the shortest path, until the destination, Z-node, is reached. Finally, the message will make it way down the tree to the destination processor.

Definition 4.1

Given two Z-nodes $(a_d, \ldots, a_2, a_1; q)(Z_h)$ and $(b_d, \ldots, b_2, ; q)(Z_h)$ where $a_k, b_k \in \{0, 1, \ldots, s_k\text{-}1\}$ for all $1 \le k \le d$, the ordered set of shortest paths for a dimension d between the two nodes can be writte as $\boldsymbol{P}^{(d)} = \{x_k \mid x_k = 1 \text{ if } a_k = b_k \vee x_k = 0 \text{ if } a_k \ne b_k, \forall 1 \le k \le d\}$.

For instance, refer to Fig. 2, the set of short paths between the Z-nodes $(0, 0, 0)$ and $(3, 0, 1)$ is $\boldsymbol{P}^{(3)} = \{1, 0, 1\}$ meaning the routes have to go through dimension 3 and then dimension 1, or dimension 1 followed by dimension 3. There is no route through dimension 2. This is a well-known concept of GHC topology.

Definition 4.2

A set of offset dimensions is a subset of $\boldsymbol{P}^{(d)}$ where all the elements are equal to 1 referring to routable or offset dimensions, $\boldsymbol{O} \subseteq \boldsymbol{P}^{(d)} = \{x_k \ne 0, \forall 1 \le k \le d\}$.

The cardinality, $|\boldsymbol{O}|$, of the set, $\boldsymbol{O}$, determines the number of hops to reach the destination and the factorial of the cardinality, $|\boldsymbol{O}|!$ defines the number of possible shortest paths between two Z-nodes. Again in Fig. 3, the ordered set of short paths between the Z-nodes $(0, 0, 0)$ and $(3, 0, 1)$ is $\boldsymbol{P}^{(3)} = \{1, 0, 1\}$, and the set, $\boldsymbol{O} = \{1, 1\}$, $|\boldsymbol{O}|! = 2! = 2$, indicates that there are 2 shortest paths of 2 hops each.

4.1. CLIOD and ALIOD Algorithms

In general, adaptive routing performs better than deterministic and oblivious routing both for benign and adversary traffics. Several adaptive routings have been proposed for Generalized Hyper Cube networks [22].

These routing algorithms can be classified as minimal and non-minimal routing. The minimal routing uses adaptively the shortest path between any source and destination pairs. Non-minimal avoids the traffic by taken longer routes. Valiant routing [32] initially routes to a randomly chosen destination and then performs minimal routing to the destination. Other non-minimal routing use information about the network at the source to decide if it is wise to route minimally or no minimally to adapt to the benign and adversary traffics. Universal Globally-Adaptive Load-Balanced algorithm [33] choses between minimal and Valiant routing on a packet-by-packet basis based on the queue lengths. One can argue the decision to route minimally or not at the source loses the ability

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 June 2023 doi:10.20944/preprints202306.1262.v1

to adapt to the traffic ahead that is dynamically changing, and in some other instance, it introduces dimensional asymmetry. This section proposes two minimal routing algorithms that use local information of the queues to randomly adapt to the traffic. Virtual channels are also deployed to prevent deadlock.

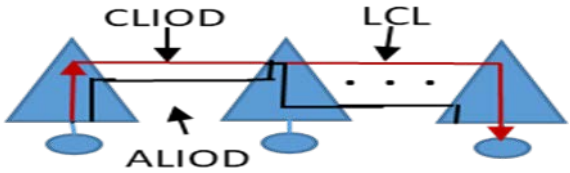


Figure 4. ALIOD and CLIOD routing paths in the Hyper Z, with the LCL least common level.

Any Level Idle Offset Dimension (ALIOD) routing algorithm, as shown in Fig. 4, alternates between up and side directions until the least common level in the Z-node is reached, then alternate between down and side directions until the destination is reached while changing layers to adapt to the traffic. When no idle offset dimensions are found, the ordered dimension routing is chosen in the side directions. The alternation is adopted to avoid heavily loaded links (large queues). On the other hand, the Common Level Idle offset dimensions (CLIOD) reaches the least common level (LCL) first in the source Z-node adaptively, and then searches for idle offset dimensions or use ordered dimension routing and then routes down within the destination Z-node. This paper shows that ALIOD performs better than CLIOD in all traffic patterns. CLIOD is an extension to the routing algorithm described for HyperX. ALIOD is described below.

Table 1. ALIOD Algorithm.

ALIOD algorithm
Get Dimension Set $P^{(d)}$
If O = ϕ (empty)
Route up to the LCL then down to Destination
Else If an idle offset dimension, $x_k \neq 0$, exists
Route through it
Set it to non-routable ($x_k=0$ in $P^{(d)}$ and O - x_k)
Else
If LCL is not reached
Route across a link with the shortest queue Up or Side - ODim
Elseif level is not 1
Route across a link with the shortest queue Down or Side - ODim
Else
Use ODim
Endif
Endif
Endif

The algorithm is deadlock free. When an idle offset dimension is found, the messages cross it and move one hop closer to the destination, without queuing delay. When no idle dimension is found, the message will use dimension-order routing, ODim, which is known to be deadlock free. It dynamically exploits a higher path diversity to avoid saturated links.

5. Performance Evaluation

This section describes the simulation models and parameters used to evaluate our proposed Z-Fat tree with the routing suggested in this paper, and provides the explanation of the obtained results.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 June 2023 doi:10.20944/preprints202306.1262.v1

5.1. Simulation Model

The simulation environment is based on a tool developed in C++ at the University of Portsmouth in the School of Computing.

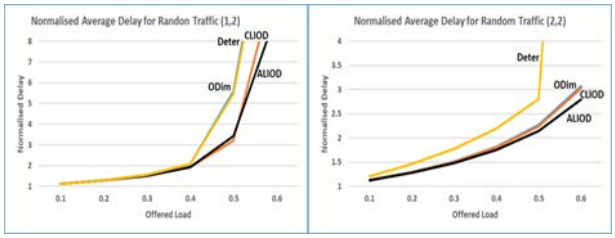
The topology consists 1024 processors transmitting at an effective link rate of 10Gbits/s. All the links in the network have 1-bit propagation delay (.1ηsec). The simulator generates 100000 messages with an average processing time derived from an exponential distribution. The processing time, uniformly distributed between 640-3200 ηsec, is the time taken by the processor to perform some computations before a communication takes place. The communication used a single Virtual Cut Through (VCT) buffer with virtual channels to avoid deadlock in each routing node. The average processing time normalizes the message transmission time to produce different offered loads per processing node. The payload of the messages is chosen to be 3200bits on average. The normalized delay (average delay/transmission time of a complete message) is recorded against the offered load, that is a fraction of the total capacity of the system. The adaptive routing dimension order (ODim), common level idle offset dimension (CLIOD), any level idle offset dimension (ALIOD) and the deterministic routing based on DRR are compared in this simulation.

Standard traffic patterns were used. They are commonly found in computational intensive scientific applications, namely the transpose, the bit-reversal and the complement patterns.

5.2. Results of the simulation

Fig. 5 (a)(b) and (c) illustrate comparisons of normalized delays for a 2-dimensional HyperZ with levels 1, 2 and 3 respectively. Notice that level, *h=1,* represents the HyperX topology. The ALIOD routing performs better that all the others and manage to use the levels of the Z-node to avoid traffic built in the queues of the routing nodes. It outperforms HyperX based on the CLIOD. For higher load, with higher level, ALIOD achieves a delay of less than 3 times the packet length. For *h=1*, the deterministic (Deter) dimension-order routing and (ODim) are identical. For level *h>1*, ODim improves slightly by using the adaptive up direction of the Z-node, based on the tree routing.

Similar patterns can be observed for transport traffic in Fig. 6 (a)(b)(c). However, CLIOD and ODim saturate at 40% of offered load. Level 2 shows better performance. The deterministic is less tolerant to transport traffics and saturates at an offered load less than 10%, for higher levels.

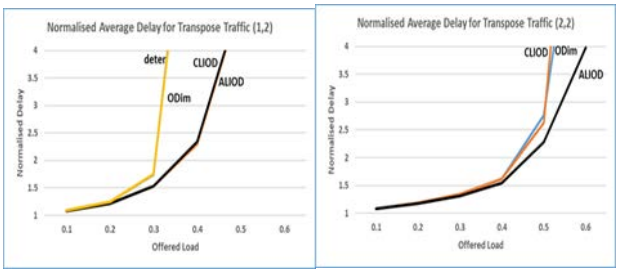


(a) (b)

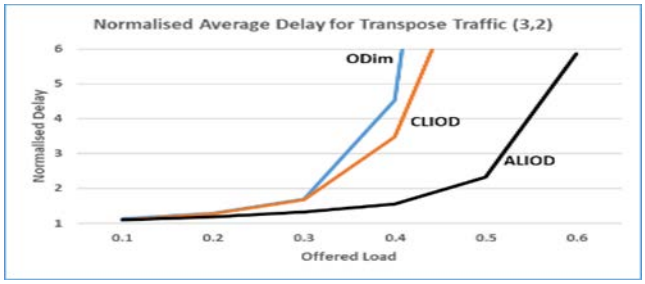
(c)

Figure 5. Normalized delay for 1024 processors with random pattern.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 June 2023 doi:10.20944/preprints202306.1262.v1



(a) (b)

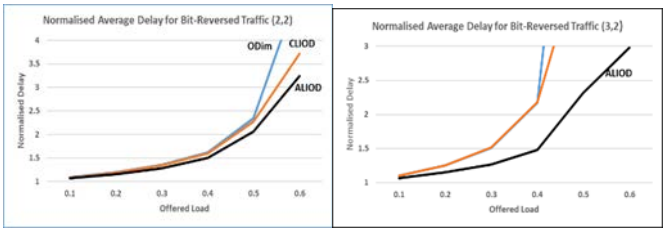


(c)

Figure 6. Normalized delay for 1024 processors with Transpose Traffic.

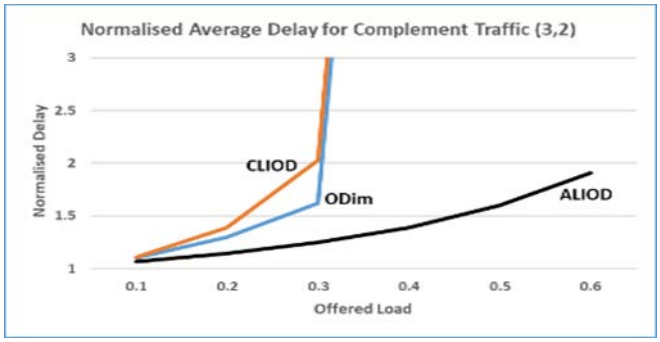
For bit Reverse traffic shown on Fig 7. (a)(b) at level 2 and 3 respectively for 2-dimensional (OHG), ALIOD achieves a normalized delay between 2.5 to 3 times the packet length at an offered load of 60%.

As illustrated in Fig. 8, with complement traffic, both CLIOD and ODim saturate at an offered load of 30%. ALIOD makes use of the up, and side links to divert the traffic. The deterministic is less tolerant to complement traffics and saturates at an offered load less than 10%, which is not shown in the graph. Results obtained in [23] have shown that the HyperX with the DAL which is similar to CLIOD and ALIOD, when the level of the computational node, Z-node, is 1, performs well compare to Valiant [32], ordered dimension, referred to in this paper by ODim, and the fat tree based on folded Closed–fat tree.



(a) (b)

Figure 7. Normalized delay for 1024 processors with bit-reverse Traffic.



Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 June 2023 doi:10.20944/preprints202306.1262.v1

Figure 8. Normalized delay for 1024 processors with Complement Traffic.

5.3. Scalability

It is possible to build a network with any size, while maintaining the regularity and the symmetry of hyper Z-Tree. One way to do it is to create an optimal Z-node with its components - links and switches - set to some maximum radix that is available in the market. The Z-node can then be replicated across the dimensions to meet the extended network requirements.

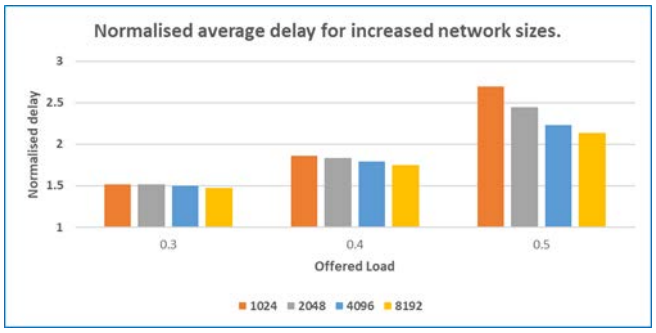


Figure 9. Normalized delay for different network sizes with 1024, 2048, 4096 and 8192 processors, level h =2 and dimension d=2, with random pattern, at offered loads 30%, 40% and 50%.

An optimal Z-node, $Z_2(8,4,8;\ 1,4,8;\ 1,2,2)$, connecting *256* processors is replicated into several $H_{2,2}(s_1, s_2)(Z_2)$ through the dimensions *1* and *2* giving *1024, 2048, 4096* and *8192* processors with (s_1, s_2) = (2, 2), (4, 2), (4, 4) and (8, 4) respectively. For instance, $1024 = s_1 * s_2 * 256 = 2*2*256$. Each network requires switches of equal radix 18, 20, 22, and 26 ports. The radix of the switch is chosen as 32, which is the maximum value, and unused ports are switched off. Larger size networks can be obtained with higher radix switch, or with higher dimensions. The graph in Fig.9 shows clearly that as the network size increases by replicating the Z_2 node across the first and second dimensions, the normalized average delay is bounded between 1.5 and 3 times the message size.

5.4. Fault Tolerance

In larger scale networks of millions of connections, the dominant failures come from the links' failures. The massive path diversity of the HyperZ defined by equation (4) creates many opportunities for fault tolerance. In [22] we have addressed fault tolerance within the Z-node. We proposed a recoiling algorithm to determine the paths of unfaulty links, and hence selected the uplinks that will avoid meeting faulty downlinks. ALIOD is more tolerant to faults than DAL routing. For ALIOD routing, there are h parallel links from one dimension to another, defined by the vector $Q_k = (q_{1k}, q_{2k}, \ldots, q_{hk})$ giving a maximum of $l_{max} = \sum_{i=1}^{h} q_{ik}$ links in an offset dimension k. For a message to be blocked due to a link failure, there should be only one remaining offset dimension with all the parallel links, l_{max}, failing. If the number of failed links is less than l_{max}, additional rerouting in the up and then down directions within the Z-node is required to locate the level of non-faulty links. This approach is already adopted within the Z-node, [22]. If all the l_{max} links fail, one has to reroute through the side links. This scheme is adopted by DAL.

6. CONCLUSIONS

This paper introduces a multi-level direct network called HyperZ which is an extension to HyperX. The main feature of this architecture is the inclusion of the Z-Tree as a computing node in a GHC interconnection. This topology is driven by the high-radix switches and low diameter. The organization based on a Z-Tree node offers tree-levels that make parallel layers and hence increases the bisection bandwidth, reduce the latency and provide a resilient fault tolerant network. As shown in the resuts, the computing nodes with multi-levels performs better than a single level. This is achieved by the ALIOD routing algorithm which makes better use of the link diversities and the parallel layers. The topology scales with the size of the network by adding more Z-nodes in the first

and second dimension; while maintaining the radius constant and for a low (GHC) dimension, the latency can be kept bounded to less than 3 times the message size.

REFERENCES

[1] C. Bekas, "Breaking the power wall in Exascale Computing, what is Exascale computing", Scientific colloquium, of Steinbach Centre for Computing of Karlsruhe Institute of Technology, [Online] Available: http://www.scc.kit.edu/downloads/oko/SCC_Kolloquium_20131114_exascale_computing.pdf 2013.

[2] D. A. Reed, J. Dongarra, "Exascale Computing and Big Data", Communications of the ACM, Vol. 58, No. 7, pp. 56-68, July 2015.

[3] P. M. Reed, D. Hadka, "Evolving Many-Objective Water Management to Exploit Exascale Computing", AGU Publications, 0 Oct 2014

[4] J. Torrelas, "Architectures for Extreme-Scale computing", Computer, IEEE Computer Society, pp. 28-35, Nov 2009.

[5] J. A. Ang, et all., "Abstract Machine Models and Proxy Architectures for Exascale Computing", Co-HPC '14 Proceedings of the 1st International Workshop on Hardware-Software Co-Design for High Performance Computing, pp. 25-32, 2014.

[6] F. Cappelo, et all., "Toward Exascale Resilience", The International Journal of High Performance Computing Applications, Volume 23, No. 4, pp. 374–388, 2009.

[7] C. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing", *IEEE Transactions on Computers*, Vol. 34, No. 10, pp. 892-901, 1985.

[8] B. Arimilli, et all., "The PERCS high-performance interconnect," in *Proc. 18th Annu. Symp. High-Perform. Interconnects*, pp. 75–82, 2010.

[9] Cray Inc., "Cray XK Series Supercomputers | Cray", *Cray.com*, 2013.[Online].Available:http://www.cray.com/Products/Computing/XK7.aspx. [Accessed: 02- Apr- 2013].

[10] IBM Bluewaters. "Blue Waters", *NCSA University of Illinois*. [Online] Available: https://bluewaters.ncsa.illinois.edu/ [Accessed: 2 Apr 2013].

[11] R. A. Haring, et all., "The IBM Blue Gene/Q Computer Chip", IEEE Micro, Vol. 32, No. 2, pp. 48-60, Mar/Apr 2012

[12] IBM Blue Gene team, "Design of the IBM Blue Gene/Q Compute Chip," IBM Journal of Research and Development, Issue 1/2, Jan-Mar 2013.

[13] D. Chen, et all. "Looking under the hood of the IBM Blue Gene/Q network", *2012 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2012.

[14] N. Jain, et all. " Maximizing Throughput on a Dragonfly Network", SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 16-21 Nov 2014.

[15] M. Garcia, et all., , "On-the-fly adaptive routing in high-radix hierarchical networks," in *41st International Conference on Parallel Processing (ICPP)*, pp. 279–288, 2012

[16] X. ZHU, "Xmesh: A Mesh-Like Topology for Network on Chip", *Journal of Software*, vol. 18, no. 9, p. 2194, 2007.

[17] M. Rahman, Y. Inoguchi, F. Faisal and M. Kundu, "Symmetric and Folded Tori Connected Torus Network", *Journal of Networks*, vol. 6, no. 1, 2011.

[18] M. Nikdast, J. Xu, X. Wu, Z. Wang, X. Wang and Z. Wang, "Fat-tree-based optical interconnection networks under crosstalk noise constraint", *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 23, no. 1, pp. 156-169, 2015.

[19] H. Gu, S. Wang, Y. Yang and J. Xu, "Design of Butterfly-Fat-Tree Optical Network-on-Chip", *Optical Engineering*, 2010

[20] Amir Mansoor Kamali Sarvestani, PHD THESIS "Evaluating Techniques for Wireless Interconnected 3D Processor Arrays", York: University of York, 2013.

[21] M. Adda, A. Peratikou, "Routing and Fault Tolerance in Z-Fat Tree", IEEE Transaction on Parallel and Distributed Systems, Vol. 28, No. 8, 2017.

[22] D. Agrawal, "Generalized Hypercube and Hyperbus Structures for a computer", IEEE Transactions on Computers Vol. 33, No.3, pp. 323-33, 1984.

[23] J. H. Ahn, N. L. Binkert, R. Schreider "HyperX: Topology, Routing, and Packaging of Efficient Large-Scale Networks", Proceedings of the ACM/IEEE Conference on High Performance Computing, SC 2009, Nov 14-20, 2009.

[24] S. Azizi, F. Safaei, N. Hashemi, "On the Topology Properties of HyperX", the Journal of Supercomputing, Vol. 66, No. 1, pp. 572-593, 2013.

[25] Mu, Y. & Li, K. (2011). "Extended Folded Cube: A Improved Hierarchical Interconnection Network". *Fourth International Symposium On Parallel Architectures*, pp. 77-81.

[26] Rahman, M., Inoguchi, Y., Faisal, F. A. & Kundu, M. K. (2011). « Symmetric and Folded Tori Connected Torus Network". *Journal of Networks*, 6 (1).

[27] N. Kini, M. Kumar and H. Mruthyunjaya, "Torus Embedded Hypercube Interconnection Network: A Comparative Study", *International Journal of Computer Applications*, vol. 1, no. 4, pp. 32-35, 2010.

[28] Rahman, M. H., Jiang, X., Masud, M. & Horiguchi, S. (2009), "Network performance of pruned hierarchical torus network", *Sixth IFIP International Conference On Network and Parallel Computing*, Oct 2009.Gold Coast: pp. 9--15.

[29] Khosravi, A., Khors, I, S. & Akbari, M. K. (2011). "Hyper node torus: A new interconnection network for high speed packet processors". *International Symposium On Computer Networks and Distributed Systems (CNDS*, pp. 106--110.

[30] Arimilli, B., Arimilli, R., Chung, V., Clark, S., Denzel, W., Drerup, B., Hoefler, T., Joyner, J., Lewis, J., Li, J. & Others (2010). "The PERCS high-performance interconnect". *18Th IEEE Symposium On High Performance Interconnects.*, pp. 75--82.

[31] Faanes G, et Al (2012), "Cray cascade: a scalable HPC system based on dragonfly network", Proceedings of the International Conference on Computing, Networking, Storage & Analysis.

[32] L. G. Valiant (1982), "A scheme for fast parallel communication ", SIAM Journal on Computing, 11(2): 350-361.

[33] J. Kim, W. J. Dally, S. Scott & D. Abts (2008), "Technology-Driven, Highly-Scalable Dragonfly Topology", International Symposium on Computer Architecture.

[34] A. Peratikou, "*An optimised and generalised node for fat tree classes"* PhD thesis, School of Computing, University of Portsmouth, Portsmouth, 2014.

[35] OPNET Technologies "Opnet modeller accelerating network R&D"– Network Simulator | Riverbed", *Riverbed*, 2016. [Online].

Mo Adda received his PhD degree in parallel and distributed systems from Surrey University, United Kingdom. He is currently a principal lecturer and MSc course leader for cyber security and digital forensics at Portsmouth University and Cambridge (EG). He has also worked for many years, as a senior consultant in the industry for simulation and modeling, where he developed many discreet event simulation tools for shipments and business process modelling. His current research includes parallel architectures, big data analytics, embedded systems, intelligent agents, network management, forensic information technology, network security and IoT forensics and cyber security. He is a member of the IEEE.