I/O NODE PLACEMENT FOR PERFORMANCE AND RELIABILITY IN TORUS NETWORKS

Babatunde Azeez, Hogil Kim, Yuho Jin, Eun Jung Kim
Department of Computer Science
Texas A&M University
College Station, TX, USA
tuniks@tamu.edu, {hogil,yuho,ejkim}@cs.tamu.edu

ABSTRACT
When a cluster system interconnects processor and I/O nodes through a network, an optimal placement of I/O nodes is critical to improve the overall system performance by reducing its communication latency. In this paper, we propose an efficient and scalable I/O node placement scheme, called a relaxed quasi-perfect, for torus-based interconnection networks using Lee distance error-correcting code. It provides a more general placement than the previous quasi-perfect placement [1]. We also suggest a fault-tolerant scheme using our I/O placement model for a guaranteed performance. Simulation results show that our scheme provides 53% speed-up over the quasi-perfect. Also the fault tolerant scheme provides a graceful slowdown especially until the number of faulty I/O nodes becomes less than half of the initial I/O nodes.

KEY WORDS
Resource Allocation, Torus Interconnection Networks, Fault Tolerance

1 Introduction

In recent years, massively parallel systems with unprecedented number of processors are emerging in order to realize a peta-scale computing environment by forming interconnection networks. Torus topology has been widely used as the high performance interconnection networks in several parallel systems such as IBM Blue Gene [2], Cray3T [3], and MIT J-Machine [4]. A torus network doubles the bisection bandwidth of a mesh with the same radix and dimension, and does not have the edge asymmetry of the mesh where the central channels are significantly more utilized than the edge channels. Therefore, a torus is a natural choice of regular topologies to achieve the performance and load balance on diverse communication patterns.

In a heterogeneous node configuration of a network that integrates processor and I/O nodes, a careful placement of I/O nodes drastically improves I/O performance and reduces the effect on computation performance while achieving cost-effective and energy-efficient systems [5, 6]. As the efforts of intelligent embedding of the I/O nodes, several placement strategies have been studied as follows.

The perfect dominating set in 2D torus enables the resource nodes (I/O) to cover all other non-resource nodes (processor) with the minimum number of resource nodes [7]. Also Bae and Bose presented a perfect distance-*t* placement of resource nodes in torus network where every non-resource node is at exactly distance-*t* or less to a resource node using Lee distance error code [8]. They further showed that the perfect distance-*t* placement is not possible in all of the 2D/3D torus networks, and proposed a quasi-perfect distance-*t* placement as an alternative where a non-resource node can be located within at most distance-(*t+1*) to some resource nodes in [1, 9]. However, the quasi-perfect placement cannot be generalized for a torus that has a higher dimension than three, and is not optimal when the number of resource nodes obtained is less than the minimum bound of the perfect distance placement. As well as the performance oriented design, it is important for the system to be able to withstand a substantial number of faults with less alteration of the pre-built network configuration. Most studies have focused on providing fault tolerance to only inter-processor communications with little effort to I/O communications [10, 11]. In this paper, we investigate intertwined issues of both I/O node placement and reliability for torus interconnection networks as follows.

1. Present a relaxed quasi-perfect I/O node placement to find more flexible placement scheme than perfect and quasi-perfect placement.

2. Provide a reliable I/O communication method in the event of I/O node failures using the I/O placement strategies including perfect and relaxed quasi-perfect placement.

The rest of the paper is organized as follows. In Section 2, we provide a brief background on I/O placement in the torus network. We propose relaxed quasi-perfect placement in Section 3, and fault-tolerant I/O message redirection scheme in Section 4. We present simulation results in Section 5 and the conclusion is given in Section 6.

2 Background

A torus network can be categorized into as a k-ary n-cube (Q_k^n) or a mixed radix n-cube ($T_{k_0,k_1,\ldots,k_{n-1}}$). A k-ary n-cube has equal number of nodes, radix k, in every n dimension, while at least one or more of the radix of each di-

Table 1. Summary of Perfect Distance-t Placements in Torus

Dim.	1-distance	2-distance	3-distance	t-distance
1	T_3	T_5	T_7	T_{2t+1}
2	$T_{5,5}$	$T_{13,13}$	$T_{25,25}$	$T_{2t^2+2t+1,2t^2+2t+1}$
3	$T_{7,7,7}$	-	-	-

mension is not the same in a mixed radix torus. In a vector notation, a node in Q_k^n and $T_{k_0,k_1,...,k_{n-1}}$ is represented as $\hat{X} = (x_{n-1}, x_{n-2}, ..., x_0)$ such that $x_i \in \{0, 1, ..., k-1\}$ and $x_i \in \{0, 1, ..., k_i - 1\}$ respectively, where $0 \leq i \leq n-1$.

Meanwhile, error detection and correction codes are used to encode a *codeword* for detecting or correcting t errors by adding redundant bits to the information bits. For example, Lee distance (D_{Lee}) is applicable when the information digits are non-binary [12] and defined between two length-n codes $\hat{U} = (u_{n-1}, u_{n-2}, ..., u_0)$ and $\hat{V} = (v_{n-1}, v_{n-2}, ..., v_0)$ as following:

$$D_{Lee}(\hat{U}, \hat{V}) = \sum_{i=0}^{n-1} \min((u_i - v_i) \bmod k, (v_i - u_i) \bmod k)$$

, where $u_i, v_i \in \{0, 1, ..., k-1\}$. Then, the linear code generated by Lee distance metric has a property that the minimum distance among the codewords for detecting and correcting t errors is $2t + 1$. An important application of Lee code is that a set of codewords of length-n can be generated over a linear n-dimensional vector space such that the minimum distance among codewords is $2t+1$, and the distance between a non-codeword and a codeword is t. With this concept, several I/O node placements were developed in a torus network using Lee distance code.

- **perfect distance-t:** Each processor node is exactly at a distance-t or less to one I/O node. None of I/O nodes are adjacent to each other [8].
- **quasi-perfect (QP) distance-t:** Each processor node is either at exactly distance-t or less to one I/O node, or at maximally distance-$(t + 1)$ to one or some I/O nodes [1, 9].
- **j-adjacency:** Each processor node has j adjacent I/O nodes. It is also said to be perfect if each processor node is adjacent to j I/O nodes and no I/O nodes are adjacent to each other [8].

Other studies showed that these placements are only possible in some tori and thus cannot be generalized for all types of n-dimensional tori [8, 9]. Table 1 gives a summary of known torus configurations of perfect I/O node placement for different distances and dimensions. Note that these are a tiling block size and only multiple blocks of one tiling block can be used to build a larger torus network.

3 Relaxed Quasi-Perfect Placement

In this section, we study two related topics for I/O placement. First, we suggest a Relaxed Quasi-Perfect (RQP) distance-t placement as an alternative if there exists no configuration of perfect distance-t placement with a given k-ary n-cube (Q_k^n) torus or mixed radix torus ($T_{k_0,k_1,...,k_{n-1}}$). Second, we consider how to efficiently place I/O nodes using the RQP placement when only the number of available I/O nodes is given without any limitation on the distance t. The definition on the RQP placement is as follows.

Definition 1. (RQP distance-t placement) *Relaxed quasi-perfect (RQP) distance-t is an I/O palcement strategy with the maximum numbers of processor nodes at a distance-t to one or more I/O nodes while the remaining processor nodes are at maximally distance-$(t + 1)$ to some other I/O nodes.*

In other words let a, b be two I/O nodes, and S_a and S_b be the set of processor nodes at distance-t or less from each I/O node but not more than distance-$(t+1)$ to a and b, then $S_a \cap S_b \neq \emptyset$, with some processor nodes at distance-t or less to both a and b. This property relaxes a limitation of one-to-one mapping between processor and I/O node at distance-t in the existing perfect or quasi-perfect distance-t algorithms. Also, the property enables the distance-t placement for any size of torus network with the small number of distance-$(t + 1)$ placement.

3.1 RQP distance-t Placement in k-ary n-cube

We show how the RQP placement can be formed from the perfect distance-t placement. Consider a k_1-ary n-cube torus network $Q_{k_1}^n$ having no perfect placement, which means that k_1 is not divisible by p and another $Q_{k_2}^n$ torus with $k_2 = \lceil \frac{k_1}{p} \rceil \times p$ such that k_2 is divisible by p. Here, p is the volume of a packing sphere with radius t and is the number of processor nodes within a distance t or less from an I/O node. For example, p is defined as $p = 2t^2 + 2t + 1$ for 2D torus in [8] and $p = \frac{(2t)(2t+1)(t+1)}{3} + 2t + 1$ for 3D torus in [9]. It is also shown that if a given radix k is divisible by p, then there exists a perfect distance-t placement in $k \times k$ torus networks [8].

Pseudo Algorithm for RQP distance-t Placement in $Q_{k_1}^n$

1. Construct a perfect distance-t placement for $Q_{k_2}^n$.
2. Form $Q_{k_1}^n$ having RQP distance-t placement by eliminating each $i - radix$ for all $i \geq k_1$ along each dimension of $Q_{k_2}^n$.
3. Assign each processor node to an I/O node at distance-t. If there is no I/O node at distance-t, a processor node can be assigned to an I/O node at distance-$(t+1)$.

To get all I/O node locations for perfect distance-t placement, a parity check matrix H is used such that every

I/O node location $\hat{X}$ satisfies $(\hat{X} \cdot H^T \bmod p) \equiv 0$, where $H = [2t^2, t]$ for 2D torus [8] and the obtained locations are equal to Lee distance t-error correcting codes. A necessary condition for RQP is that there must exist a perfect distance-t placement for a given t in n-dimensional torus with a k_2, which is larger than given k_1. A RQP can then be constructed for a $Q^n_{k_1}$ by building a perfect distance-t placement in $Q^n_{k_2}$ larger than $Q^n_{k_1}$ in terms of the size of radix k by eliminating redundant radix positions in each dimension of $Q^n_{k_2}$.

Theorem 1. *The number of I/O nodes required for RQP is equal to the minimum bound M for the perfect distance-t.*

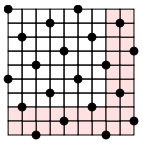
Proof. As defined in [8], the minimum bound M refers to the minimum number of I/O nodes required for constructing the perfect distance-t placement and can be obtained by $M \geq \frac{N}{p}$, where $N = k^n$ and p is the volume of a packing sphere with radius t. According to the necessary condition for RQP, we know that RQP is constructed using the same t and n with the perfect distance-t placement. If k is divisible by p, the perfect and RQP placements are the same. Otherwise, RQP assigns some I/O nodes out of M I/O nodes to some processor nodes at distance-$(t+1)$, while no perfect distance-t placement exists with M. This is possible by relaxing the constraint of one-to-one mapping between a processor node and an I/O node, which is the main idea of the RQP placement. Since the M calculated using the given t, n and k for the perfect and RQP placement is same, the minimum bound remains same between the perfect and RQP placement. □

The number of I/O nodes obtained through the quasi-perfect placement is lower than that in the perfect distance placement. Theorem 1, hence, implies that the RQP placement is better than the quasi-perfect placement in terms of the average distance between processor and I/O nodes. Figures 1 and 2 [1] show the construction of RQP distance-1 for Q^2_8 and Q^3_4. Figure 1-(a) is a perfect distance-1 placement for Q^2_{10} before constructing RQP distance-1 Q^2_8. I/O nodes are represented by black circles and all cross points represent processor nodes. From the perfect distance-1 placement, all radices ≥ 8 are eliminated resulting in RQP distance-1 for Q^2_8 in Figure 1-(b). A square node is a processor node that is at distance-2 from an I/O node. Similar procedure is applied to get RQP distance-1 in Q^3_4 using perfect distance-1 in Q^3_7 and the result is shown in Figure 2. For this construction, two parity check matrices $H = [1, 3]$ and $H = [1, 2, 3]$ are used to construct perfect distance-1 placement for Q^2_5 and Q^7_7, relatively.

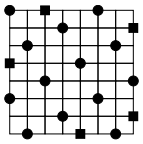
3.2 Distance-t Placement in Mixed Radix Torus

There exists no perfect distance-t in mixed radix torus $T_{k_1,k_2,\cdots,k_n}$ when one of radix $k_1, k_2, \cdots, k_n$ is not divisible by p. We can construct a RQP if no perfect distance-t exists by forming a perfect distance-t in torus $T_{q1q_2\cdots q_n}$,

[1] We do not draw wrap-around links in each dimension.



(a) Row/Column Elimination in Perfect distance-1 for Q^2_{10}



(b) RQP distance-1 in Q^2_8

Figure 1. RQP distance-1 Construction for Q^2_8

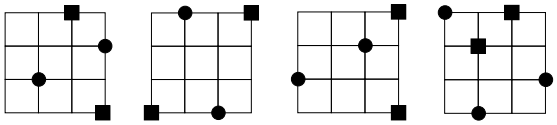


Figure 2. RQP distance-1 in Q^3_4

where $q_i = \left(\lceil \frac{k_i}{k} \rceil \times k\right)$ for $1 \leq i \leq n$. Their RQP distance-t placement follows the same procedure with the construction of RQP placement for k-ary n-cube network along every dimension in $T_{q_1,q_2,\cdots,q_n}$. All other properties remain the same as k-ary n-cube.

3.3 I/O Placement with a Given Number of I/O Nodes

In the previous section, we described RQP distance-t placement with a given distance value t for a particular size of torus network. However, there might be some cases where there is constraint on the number of available I/O nodes. In particular, we solve this problem of I/O node placement in torus network when the number of available I/O nodes is different from the required minimum bound of the perfect distance-t placement using the suggested RQP distance-t placement.

Consider a torus configuration T (Q^n_k or $T_{k_0,k_1,\ldots,k_{n-1}}$) with a given number of I/O nodes, W. Also, let t be a positive integer such that M_t, the minimum bound for the perfect distance-t, is the smallest integer greater than or equal to W, and M_{t+1} is the largest integer less than or equal to W assuming $t < k/2$ for some radix-k in T. If M_t is closer to W than M_{t+1}, ($|M_t - W| < |W - M_{t+1}|$), then we first construct a perfect or RQP distance-t placement for T and remove some I/O nodes to get W I/O node placement. Otherwise, if M_{t+1} is closer to W than M_t ($|M_t - W| < |W - M_{t+1}|$), we construct a perfect or RQP distance-$(t+1)$ placement for T with addition of some I/O node locations to obtain W I/O nodes placement. However, if $|M_t - W| = |W - M_{t+1}|$, then either addition or deletion procedure can be used. The procedures for adding or deleting I/O nodes are as follows.

Case I: Deletion Procedure ($|M_t - W| < |W - M_{t+1}|$)

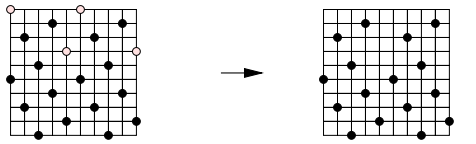
1. Construct a perfect distance-t if each radix k of the given torus T is divisible by p. Otherwise, construct a RQP distance-t placement.

2. In the I/O placement built in (1), delete an I/O node with the smallest radix address. For the RQP placement, choose a pair of I/O nodes that are located at distance $2t+1$ from each other and delete one of them.

3. Find and mark I/O nodes as *undeletable* that locate at distance $t+1$ from the processor nodes that locate at distance t from currently deleted I/O node.
4. For unmarked I/O nodes, repeat (2) and (3) until the remaining number of I/O nodes is equal to W.
5. If the number of remaining I/O nodes is still greater than W, then repeat (2) and (3) with marked I/O nodes until it is equal to W.

Case II: Addition Procedure($|M_t - W| > |W - M_{t+1}|$)

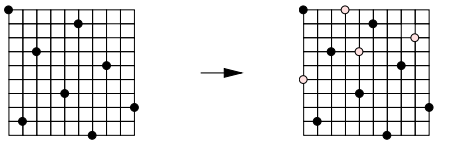
1. Construct a perfect distance-$(t+1)$ if each radix k of torus T is divisible by p. Otherwise, construct a RQP distance-$(t+1)$ I/O node placement.
2. For a perfect distance-$(t+1)$ placement, mark all processor nodes at distance-$(t+1)$ to an I/O node.
   2.1 Among all marked processor nodes, make one with the smallest radix an I/O node if it is at distance t or less to $2n-1$ other marked processor nodes, where n is the dimension of the torus.
   2.2 Unmark other marked processor nodes at distance t to the added I/O node.
   2.3 Repeat (2.1) and (2.2) until the number of I/O nodes is equal to W. Otherwise, repeat the process with distance-t.
3. For RQP distance-$(t+1)$, find and mark processor nodes at distance $t+2$ to an I/O node.
   3.1 Make any marked processor node an I/O node location if none of other marked processor nodes are at distance $t+2$ to it.
   3.2 Unmark any marked processor node with at most distance $t+1$ to the I/O node. Continue procedure (3.1) and (3.2) until no processor node can be unmarked.
   3.3 If the number of I/O nodes is still less than W, then repeat (2.1) to (2.3).



(a) Perfect distance-1 (b) 16 I/O nodes

Figure 3. Deletion Procedure

Figures 3 and 4 illustrate the deletion and addition procedures using Q_{10}^2 with W=16 and W=12, respectively. With t=1 and t=2, the minimum bounds are 20 and 8 I/O nodes. The deletion procedure is used since 16 is closer to 20. In Figure 3 (a), a perfect distance-1 is first constructed



(a) RQP Perfect distance-2 (b) 12 I/O nodes

Figure 4. Addition Procedure

with 20 I/O nodes. Starting with I/O node (0, 0), all I/O nodes denoted as gray circles are deleted to form the original placement as shown in Figure 3 (b). Likewise, the addition procedure is used when W=12 by first constructing RQP distance-2 with 8 I/O nodes. Some processor nodes with distance-3 are made as I/O nodes until no processor nodes are at distance-3. Afterward, the procedure is applied to processor nodes with distance-2 to form the final placement.

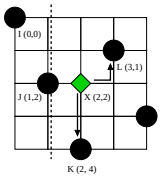
4 Fault Tolerance to I/O Communication

The development of a fault tolerance capability with respect to I/O nodes is important to minimize additional I/O latencies that may be incurred as a result of failures. In this section, we show that our I/O placement scheme can be easily extended to an alternative against I/O node faults. We assume that a fault occurs only in I/O nodes while the router in the node is still active. Furthermore, we define *default* I/O nodes as the I/O nodes acquired by one of perfect or RQP placement and *alternative* as the next shortest distance I/O node except the default I/O nodes.

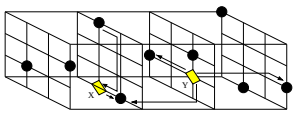
Perfect distance-t and RQP provide redundancy that is a key element in design of the fault tolerant system as depicted in Theorem 2.

Theorem 2. *In a perfect distance-t placement, a processor node at distance $0 < r \leq t$ from its default I/O node has n alternative I/O nodes at distance $d - r$ except a default I/O node, where n is the number of dimension, d is the minimum Lee distance among I/O nodes, and r is the distance between an affected processor node and its default I/O node.*

Due to space limitation, we give the proof for the theorem 2 in [13]. The RQP distance-t also exhibits the same redundancy properties as proved in [13]. The correctness of the Theorem 2 can be demonstrated in Figure 6. Figure 6 (a) shows five I/O nodes required for the perfect distance-1 placement. Taking I/O node J(1,2) as a reference point, each of three remaining I/O nodes I, K, and L has exactly Lee distance 3, the minimum distance for perfect distance-1 to node J. Figure 6 (b) shows the redundancy in RQP distance-1. Processor node X is at distance-1 to two other I/O nodes and a processor node Y is at distance-2 to other three I/O nodes. Among those n alternative I/O nodes, a processor node with a faulty default I/O node can



(a) Perfect distance-1 in Q_5^2



(b) RQP distance-1 in Q_4^3

Figure 5. Redundancy in Perfect and RQP

Table 2. Simulation Parameters

Torus Network Size	8×8, 4×8, $4 \times 4 \times 4$
Physical Link Bandwidth	2.5 Gbps
Physical Links	4 (2D) or 6 (3D)
VCs / Physical Link	16
Buffer Size	256 flit buffers
Flit Size	256 bits
Processor Message Size	32 flits
I/O Message Size	128 flits
Switching Mechanism	Packet Switching
Routing	Deterministic (E-Cube)

select a new default I/O node considering a load balancing for channels and I/O nodes. We can achieve this by dimension-order selection like deterministic routing algorithm. Whenever a processor node has more than one alternate I/O nodes, it selects an I/O node with the same radix position in the lowest dimension (X). If there is no alternate one, it checks in the next higher dimension (Y) and so on. When no I/O nodes fall on the same radix, then the next radix positions adjacent to the current radix position will be searched until an I/O node is found. A detailed algorithm can be found in [13].

5 Performance Results

In order to analyze the effect of RQP placement of I/O nodes in a torus network, we adapt a cycle accurate simulator developed in [14] for our purpose. The simulator generates both processor and I/O messages at a specific inter-arrival time that follows an exponential distribution. The main parameters used are as shown in Table 2.

5.1 RQP vs. QP Placements

In order to measure the effectiveness of our relaxed quasi-perfect (RQP) compared to quasi-perfect placement (QP), we simulated a 8×8 torus (Q_8^2) for RQP and QP distance-1 placements. Each I/O placement strategy was simulated at constant 10% and 20% I/O ratio while varying the offered input load as shown in Figure 6.

At constant 10% I/O ratio, the difference in the average I/O network latency for both schemes is small at low input load. However, as the input load increases, there is a clear gap between RQP and QP with RQP maintaining lower average latency until it reaches the saturation point. RQP saturates the network at 60% input load, while QP does at 45% input load. At constant 20% I/O ratio, the difference is clearer with RQP maintaining lower average latency while QP quickly enters saturation at 25% input load compared with 40% input load in RQP.

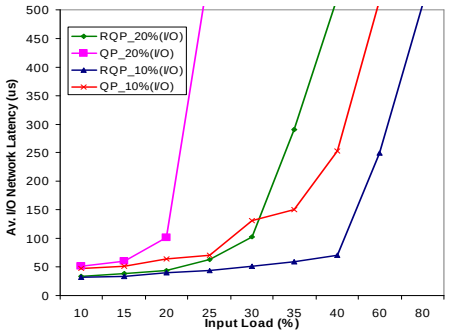
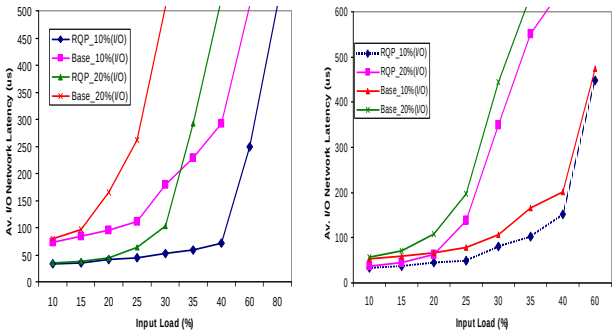


Figure 6. Latency Comparison for RQP vs. QP in Q_8^2

5.2 RQP vs. Base Placements

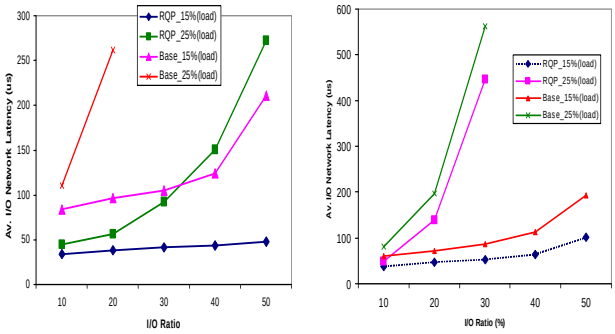
We analyze the performance benefit of careful placement of I/O nodes within torus networks using RQP especially for those networks that have no perfect distance-t placement. Figure 7 shows the effect of RQP placement on the average network latency compared with the Base placement in 8×8 torus (Q_8^2) and $4 \times 4 \times 4$ torus (Q_4^3) at constant I/O ratio 10% and 20%. Here, the *Base* placement is used for comparison such that I/O nodes are concentrated at the base radix or plane of torus network. For example, nodes of $(k-1, i), 0 \leq i \leq k-1$ are used for I/O node in 2-dimension torus network Q_k^2. Clearly, RQP outperforms the base placement with substantially lower average I/O network latency even at low input load. The base placement becomes saturated faster than the RQP placement for both 10% and 20% I/O ratio workloads. Note that Q_4^3 has a higher saturation point than Q_8^2, since the bisection bandwidth is increased two times.

We also observe the effect of I/O ratio on the performance of RQP, at constant input load for Q_8^2 and Q_4^3. Figure 8 (a) shows the results obtained for RQP distance-1 in Q_8^2 at 15% and 25% input load. At 15% input load, increasing the I/O ratio has little effect on the average network latency of RQP but the effect is significant for a base placement especially at high I/O ratio of above 40%. At 25% input load, there is a linear increase in the latency for both schemes while RQP still maintains lower latency than base scheme. A similar result is observed with Q_4^3 as shown in Figure 8 (b). However, at higher input load (25%), the effect of increasing I/O ratio is more significant than in the two dimensional torus.



(a) 8×8 torus (Q_8^2) (b) $4 \times 4 \times 4$ torus (Q_4^3)

Figure 7. RQP vs. Base with Varying Input Load



(a) 8×8 torus (Q_8^2) (b) $4 \times 4 \times 4$ torus (Q_4^3)

Figure 8. RQP vs. Base with Varying I/O Ratio

5.3 I/O Placement and Fault Tolerance

We analyze the effect of our fault tolerance scheme based on perfect or RQP I/O placement in the presence of faulty I/O nodes. A Q_4^3 torus network is simulated with RQP distance-1 placement while I/O nodes are randomly made faulty. We compare the slowdown obtained with 1, 2, 3, and 4 faulty I/O nodes relative to when there are no faulty I/O nodes. Relatively small performance degradation is observed with up to 3 faulty I/O nodes in Figure 9. In general, we can conclude the fault tolerant redirection scheme provides a small slowdown, when the number of available I/O nodes is greater than half of the required minimum lower bound for a distance-t placement.

6 Conclusion

In this paper, we proposed relaxed quasi-perfect (RQP) distance-t placement that loosens the constraint of one-to-one mapping between processor and I/O nodes when there exists no perfect distance-t placement for a particular torus network. We also implemented an algorithm for the fault tolerance using the I/O placement schemes to provide minimal slowdown in the event of I/O node failures. Based on our simulation results, RQP outperforms both QP and the simple base I/O placement strategies. The design of redirection also proved its benefits on the performance. For future work, we plan to explore reliability issues in presence of link failure.

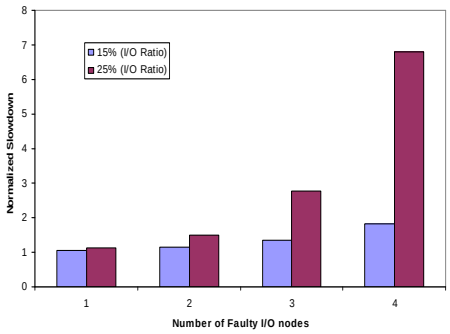


Figure 9. Effect of Faulty I/O Nodes

References

[1] B. Almohammad and B. Bose, "Resource Placement in 2D Tori," in *Proceedings of IPPS/SPDP*, pp. 431–438, 1998.

[2] T. B. Team, "An Overview of the BlueGene/L Supercomputer," in *Proceedings of SC*, pp. 1–22, 2002.

[3] S. Scott and G. Thorson, "The Cray T3E Network: Adaptive Routing in High Performance 3D torus," *HOT Interconnects IV*, 1996.

[4] M. D. Noakes, D. A. Wallach, and W. J. Dally, "The J-Machine Multicomputer: An Architectural Evaluation," in *Proceedings of ISCA*, 1993.

[5] J. Ghosh, K. Goveas, and J. Draper, "Performance Evaluation of Parallel I/O Subsystem for Hypercube Multicomputers," *Journal of Parallel and Distributed Computing*, vol. 17, pp. 90–106, 1993.

[6] R. Jain, J. Werth, and J. Browne, "Input/Output in Parallel and Distributed Computer Systems," *Kluwer Academic Publishers, Massachusetts*, 1996.

[7] M. Livingston and Q. Stout, "Perfect Dominating Sets," *Congressus Numerantum 79*, pp. 187–203, 1990.

[8] M. M. Bae and B. Bose, "Resource Placement in Torus-Based Networks," in *Proceedings of IPPS*, pp. 327–331, 1996.

[9] B. F. AlBdaiwi and B. Bose, "On Resource Placements in 3D tori," *J. Parallel Distrib. Comput.*, vol. 63, no. 9, pp. 838–845, 2003.

[10] M. M. Bae and B. Bose, "Spare Processor Allocation for Fault Tolerance in Torus-Based multicomputers," in *Proceedings of FTCS*, pp. 282–291, 1996.

[11] J. Bruck, R. Cypher, and D. Soroker, "Tolerating Faults in Hypercubes Using Subcube Partitioning," *IEEE Trans. Computers*, vol. 41, no. 5, pp. 599–605, 1992.

[12] C. Lee, "Some Properties of Nonbinary Error-correcting Codes," *IEEE Transactions on Information Theory*, vol. 4, no. 2, pp. 77–82, 1958.

[13] B. T. Azeez, "Reliable Low Latency I/O in Torus Based Interconnection Networks," Master's thesis, Texas A&M University, College Station, Texas, 2005.

[14] E. J. Kim, K. H. Yum, C. R. Das, M. S. Yousif, and J. Duato, "Performance Enhancement Techniques for Infi niBandTM Architecture," in *Proceedings of HPCA*, pp. 253–, 2003.