Task Mapping in Rectangular Twisted Tori

Cristóbal Camarero[†], Enrique Vallejo[†], Carmen Martinez[†], Miquel Moreto[⋆], Ramón Beivide[†]
[†]University of Cantabria
Electronics and Computers Department
Avda. Los Castros s/n, Santander, Spain
{camareroc, vallejе, martinezc, beivider}@unican.es
[⋆]Universitat Politècnica de Catalunya
Computer Architecture Department
Jordi Girona, 1-3, C6-204, 08034, Barcelona
mmoreto@ac.upc.edu

Keywords: network topologies, application mapping, rectangular twisted torus

Abstract

Twisted torus topologies have been proposed as an alternative to toroidal rectangular networks, improving distance parameters and providing network symmetry. However, twisting is apparently less amenable to task mapping algorithms of real life applications. In this paper we make an analytical study of different mapping and concentration techniques on 2D twisted tori that try to compensate for the twisted peripheral links. We introduce a performance model based on the network average distance and the detection of the set of links which receive the highest load. The model also considers the amount of local and global communications in the network. Our model shows that the twisted torus can improve latency and maximum throughput over rectangular torus, especially when global communications dominate over local ones and when some concentration is employed. Simulation results corroborate our synthetic model. For real applications from the NPB benchmark suite, the use of the twisted topologies with an appropriate mapping provides overall average application speedups of 2.9%, which increase to 4.9% when concentrated topologies ($c = 2$) are considered.

1. INTRODUCTION

The interconnection network of a High Performance Computing (HPC) system is critical to obtain its maximum performance. The topology of the network defines the interconnection pattern for the nodes in the system. While commodity networks such as Ethernet or Infiniband typically employ multi-tree topologies, many HPC systems employ other topologies which fit better to some computational problems. Hypercubes, meshes and tori have been common in the HPC arena, since they are closer to the actual data transfers that occur in many applications.

Twisted tori are variants of the torus topology in which a twist is applied to the peripheral links in one or more dimensions [5, 23, 25]. Different variants of 2D and 3D twisted tori have been studied in the past [8, 9, 24]. Rectangular tori and meshes are often built for practical reasons of packaging and modularity. In this paper we focus on the Rectangular Twisted Torus (RTT) which is a twisted version of the 2D Rectangular Torus (RT) topology. Its peripheral twist modifies the distance properties of the base topology, reducing the diameter, average distance, and more importantly, balancing the use of the network links in different dimensions. As a consequence, it can achieve a 50% increase in network throughput under uniform traffic.

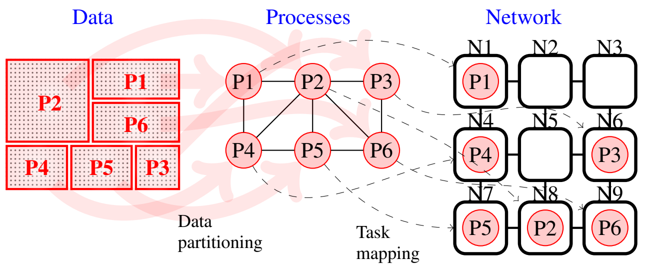

Figure 1. Data partitioning and task mapping.

Traffic from real applications behaves according to the nature of parallel algorithms and depends on the allocation of each logical task in the network and on their communication requirements. The assignment of work to system nodes is a two-step process. First, *data partitioning* divides the program dataset into multiple groups of data to be operated in parallel by each process. The second step is *task mapping*, which assigns each of the processes to an individual computation node. Both steps are illustrated in Figure 1.

Depending on the application, processes are arranged according to a certain *logical topology* (or *communication graph*), which reflects the communication pattern between them. The logical topology depends on the data partitioning employed, which is largely dependant on the data structures and the algorithm used by the application. Modifying the data partitioning mechanism to fit the underlying physical topology is generally considered very difficult since it implies modifying the algorithm. On the contrary, task mapping considers the *logical topology* of the application and the *physical topology* of the system to provide an efficient solution that preserves the communications locality as much as possible. Task mapping is a graph embedding problem, in which a guest graph (the logical topology) must be accommodated to a host graph (the physical topology) minimizing an objective cost function such as byte-hop [2], maximum dilation or average dilation [26]. Task mapping is an NP-complete problem [7, 16], but multiple heuristic mechanisms have been deployed to provide acceptable results [6].

Concentration is a technique typically employed in HPC to reduce system cost and increase scalability. A concentrated system connects multiple computation nodes to a single (higher-radix) switch. This has been routinely employed in fat trees, but also in direct topologies. One example is the Gordon Supercomputer [21] which employs a 3D concentrated torus using commodity Infiniband technology. The use of concentration adds a new dimension to the mapping problem, which also needs to consider which logical tasks are concentrated into the same network node.

This paper presents a comprehensive analysis of the RTT topology under realistic conditions that considers the mappings of HPC applications and the use of concentration. In this way, although twisting is apparently less amenable for task mapping, we can show how the topological advantages of RTTs translates in execution time reductions by choosing the adequate mapping technique. Specifically, the main contributions of this paper are the following:

1. We provide an exhaustive topological comparative review of two-dimensional RTs and RTTs, introducing a novel routing for RTTs.
2. We elaborate an analytical model which estimates the expected performance of both topologies with different mapping algorithms, according to the amount of local and global communications in the system and observing that RTTs should outperform RTs in many scenarios. We present the counterintuitive result that using "twists" in the concentration function of RTTs, despite not allocating neighbor tasks to the same network node, helps to improve performance.
3. We validate the theoretical model, considering both standard and concentrated versions of each topology, by simulating synthetic traffic with local and global communications.
4. Finally, we consider the performance in a real scenario by simulating several benchmarks from the NAS Parallel Benchmark (NPB 3.2) suite [4].

The rest of this work is organized as follows. Section 2. defines and compares RTs and RTTs and introduces their routing algorithms. Section 3. considers the problem of mapping applications, presenting a performance model. Section 4. details the experimental environment while Section 5. evaluates both topologies with different mapping and concentration functions using the NPB benchmarks. Finally, Section 6. concludes the paper.

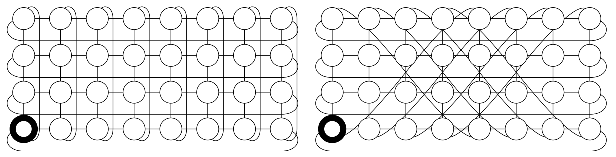
Figure 2. RT(4) and RTT(4)

2. RECTANGULAR AND TWISTED TORUS TOPOLOGIES AND ROUTING

In this section, we compare both topologies in terms of their distance properties. Networks are modeled by graphs, where vertices represent network switches and edges represent the communication links among them. Since the graphs considered here are built over rectangular meshes, we denote for convenience the set of nodes of the basic rectangular $m \times n$ mesh as:

$$\mathcal{R}_{m,n} = \{(x,y) \in \mathbb{Z}^2 \mid 0 \le x \le m-1, 0 \le y \le n-1\}.$$

Hence, RTs can be constructed over this set by adding orthogonal parallel wraparound links to the mesh in both dimensions, as can be seen in Figure 2. RTTs, also in Figure 2, can be built from the previous torus by replacing parallel vertical links by twisted ones, as given in the following definition. Since we focus on 2 : 1 aspect ratio, we will denote by RT(a) the torus over $R_{2a,a}$.

Definition 2..1. *A Rectangular Twisted Torus RTT(a) is defined over $\mathcal{R}_{2a,a}$. All the inner links in the rectangle form an orthogonal 2D mesh and the wraparound links are defined as:*

- $(x,0)$ *is adjacent to* $(x+a,a-1)$ $0 \le x \le a-1$.
- $(x,0)$ *is adjacent to* $(x-a,a-1)$ $a \le x \le 2a-1$.
- $(0,y)$ *is adjacent to* $(2a-1,y)$ $0 \le y \le a-1$.

Table 1 summarizes approximated distance-properties of both topologies, where the diameter is denoted k, and the average distance $\overline{k}$. We also include the average distance per dimension, so that $\overline{k} = \overline{k_x} + \overline{k_y}$. Note that RTTs have better distance properties than RTs for the same number of nodes. However, a topological difference with more impact on performance is symmetry. RTTs and RTs are node-symmetric topologies, i.e. any node can *observe* the same local environment. Nevertheless, RTs are not edge-symmetric graphs since horizontal links are not equivalent to the vertical ones. For example, note that in a RT(a) horizontal links form cycles of length $2a$ and the vertical links form cycles of length a. On the contrary, RTTs are completely symmetric, since the twist in the vertical dimension makes all links locally equivalent [10].

Finally, we present Algorithms 1 and 2 to compute routing records in RTs and RTTs. The *routing record* to reach destination node v_d from source node v_s can be defined as $r = (r_x, r_y)$ such that $v_d - v_s = r$ with $|r| = |r_x| + |r_y|$ of minimum weight.

Table 1. Topology distance properties [9].

Topology	k	$\overline{k}$	$\overline{k_v}$	$\overline{k_h}$
RT(a)	$\frac{3a}{2}$	$\frac{3a}{4}$	$\frac{a}{2}$	$\frac{a}{4}$
RTT(a)	a	$\frac{2a}{3}$	$\frac{a}{3}$	$\frac{a}{3}$

The original routing algorithm for the RTT introduced in [9] was significantly more complex than the routing in RT. By contrast, Algorithm 2 presents a novel routing record computation for RTTs. The algorithm uses some ideas from a more general algorithm in [15] particularizing it to RTT, which reduces the total number of computations presented in [9]. Note that both algorithms have the same complexity and the most costly operation is the remainder calculation (rem), which is straightforward in the cases of a being a power of 2.

```
Algorithm 1: Routing in RT($a$)
  Input: $x,y := v_d - v_s \in \mathcal{R}_{2a,a} - \mathcal{R}_{2a,a}$
  Output: $r$ routing record
  $x' := \text{rem}(x+a, 2a) - a;$
  $y' := \text{rem}(y + \lfloor a/2 \rfloor, a) - \lfloor a/2 \rfloor;$
  $r := (x', y');$
```

```
Algorithm 2: Routing in RTT($a$)
  Input: $x,y := v_d - v_s \in \mathcal{R}_{2a,a} - \mathcal{R}_{2a,a}$
  Output: $r$ routing record
  $p := \text{rem}(x+y+a, 2a);$
  $q := \text{rem}(y-x+a, 2a);$
  $x' := (p-q)/2;$
  $y' := (p+q-2a)/2;$
  $r := (x', y');$
```

3. TASK MAPPING IN RECTANGULAR AND TWISTED TORUS

As presented in Section 1., we will study different mapping and concentration functions to determine how they impact performance. Many scientific applications rely on structured grid communication patterns which employ both local (near-neighbour) and global communication, [3, 13]. Across this paper we restrict our study to 2D topologies for both the application communication pattern (meshes or tori) and the network topology (RT and RTT); the extension to 3D (or more) is left for future work. The mapping of meshes into both RT and RTT is simple, since the peripheral links do not have an impact on the adjacency of the mesh. Therefore, we focus on the mapping of 2D torus into RT and RTT. We will consider communication graphs with the same number of nodes as the physical topology, or a multiple value when using concentration.

The *mapping function* maps each process (or a set of concentrated processes) from the logical topology into one physical network node. We will consider two mapping functions,

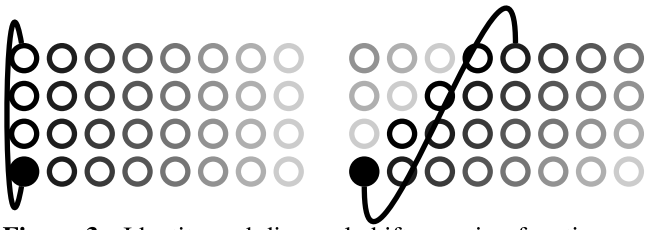
Figure 3. Identity and diagonal-shift mapping functions on RT(4).

depicted in Figure 3: the *identity* function *id* maps the processes grid directly into the internal mesh to preserve internal adjacency; by contrast, the *diagonal-shift* f^d introduces an internal incremental twist to compensate the twisted peripheral links in RTT. The mapping f^d is inspired by the mappings considered for double loop networks in [12]. These mapping functions are formally defined for logical and physical topologies of the same size, a rectangle $\mathcal{R}_{m,n}$, as follows:

$$id(x,y) = (x,y)$$
$$f^d(x,y) = (x+y \mod m, y)$$

In concentrated networks, a *concentrating function* determines which processes are placed on the same network node prior to the mapping function. We consider two rectangles $\mathcal{R}_1 = \mathcal{R}_{pm,qn}$ and $\mathcal{R}_2 = \mathcal{R}_{m,n}$ for $m,n,p,q \in \mathbb{N}$. A concentrating function of concentration $c = pq$ sends c processes from $\mathcal{R}_1$ to the same node in $\mathcal{R}_2$. We consider several concentration and mapping functions that seek to preserve the communication locality from the logical graph. Specifically, the *horizontal union* f_c^h, the *vertical union* f_c^v and the *twisted union* f_c^t are defined as:

$$f_c^h(x,y) = \left(\left\lfloor \frac{x}{c} \right\rfloor, y \right), \ q = 1$$
$$f_c^v(x,y) = \left(x, \left\lfloor \frac{y}{c} \right\rfloor \right), \ p = 1$$
$$f_c^t(x,y) = \left(x + \left\lfloor \frac{y}{n} \right\rfloor \frac{m}{2} \pmod m, y - \left\lfloor \frac{y}{n} \right\rfloor n \right)$$

Figure 4 represents the three concentration functions with $c = 2$ applied to a 8×8 mesh. Note how the twisted concentration function f^t does not concentrate neighbor nodes; rather, it is designed to compensate for the twist in the peripheral links of RTTs, by preserving adjacency in the logical topology when the identity mapping is employed. Different concentration functions can be combined for $c > 2$. We will denote by $f \circ g$ the composition such that $(f \circ g)(x,y) = f(g(x,y))$.

The next subsections will study the relative performance obtained with each topology (RT and RTT) using the different mapping and concentration functions presented here. We will first introduce a model of a generic application that considers both the amount of local and global messages sent. Considering this model, next we determine expressions for the expected performance with a given topology and mapping in

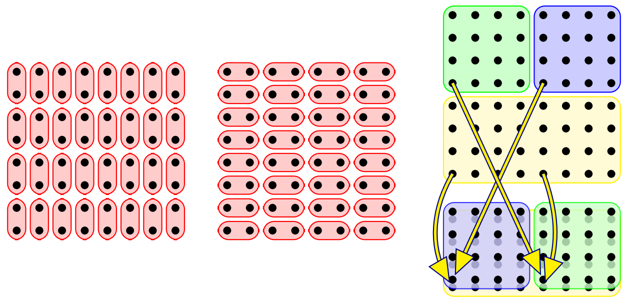

Figure 4. Concentration functions $f^v_{c=2}$, $f^h_{c=2}$ and $f^t_{c=2}$ on a 8×8 mesh.

terms of base latency and maximum accepted throughput. Finally we calculate these performance values for logical torus mapped into RT and into RTT with different combinations of mapping and concentration functions.

3.1. Modelling a Generic Application

We will consider a simple model for the communications of an application. Our model considers a variable rate of local and collective (global) communications in the application graph. We denote with α the proportion of local (l) messages, and $(1-\alpha)$ the proportion of messages corresponding to collective communications (global, g), assuming the same message size.

Local traffic communicates each process with one of its (up to) four direct neighbours in the application graph, which will be a 2D mesh or torus. Depending on the mapping and concentration functions, these neighbor nodes could be mapped far away in the physical topology. Collectives communicate a given process (or each of them) with a set (or all) of the other processes in the system. While the behavior of local communications is dependant on the mapping algorithm, we will assume that global communications can be averaged as uniform traffic, whose performance only depends on the physical topology. Our model does not consider the less frequent point-to-point messages sent to remote (not neighbour) nodes.

3.2. Performance modelling

The performance of a network depends on which metric is most restrictive during the execution of an application. Specifically, we will consider both maximum accepted throughput and average latency. In general, average (base) latency, or average latency on zero load, depends on the average distance of the topology while maximum (base) latency depends on its diameter. Actual latencies in the network will depend on the base latency and the network contention which is not considered in our simple model.

Under uniform traffic, the maximum throughput in an asymmetric torus depends on the maximum average distance *per dimension* [9], since longer dimensions will typically saturate earlier. However, when certain mapping and concentration functions are considered, the links in a given dimension may not receive all the same load. One such case are the peripheral links of the RTT when the *id* mapping is employed. In such cases, it is the subset of links that receives the highest load which determines the maximum throughput.

In this section we study the theoretical performance of RT and RTT when the network performance is limited by either throughput or communication latency, considering different mapping and concentration functions and a variable rate of local and global traffic.

3.2.1. Latency Estimation

We denote by τ the average distance travelled by the packets in the network. Assuming a constant link latency in the network, τ is an indicator of the base latency in the network. τ differs from the topological average distance $\bar{k}$, since τ depends on the communication pattern and the mapping and concentration functions. Note that τ can be divided into two terms, considering the contribution of local and global traffic:

$$\tau = \tau^l + \tau^g = \alpha \cdot d + (1-\alpha) \cdot \bar{k},$$

where $\tau^l = \alpha \cdot d$ represents the contribution from local messages and depends on the *average dilation* of the mapping function, d. Dilation, that is network distance of adjacent processes in the logical topology, has been employed as an objective function in mapping algorithms. However, in our model it does not directly determine the performance, since the global communications are not affected by the mapping algorithm. Interestingly, τ^g only depends on the physical topology, with its overall value being determined by the average distance, $\bar{k}$, in Table 1 from Section 2..

3.2.2. Throughput Estimation

Let th be the number of phits sent per cycle by each of the N nodes in the network. Let E be the set of edges (links) in the graph, and $|E|$ its cardinal. If all the links in the network were used in a balanced way, the maximum throughput of the network (in phits/cycle) would be calculated by:

$$N \cdot th \cdot \tau \leq 2|E| \Rightarrow th \leq \frac{2|E|}{N \cdot \tau}$$

For example, when all nodes communicate with their four direct neighbours in the network ($\tau = 1$), up to $th = 4$ phits/(node·cycle) can be accepted, since $|E| = 2N$ in both RT and RTT.

By contrast, when the load on different links of the network differs, the set of links that receives the highest load will saturate first and become the bottleneck that limits the maximum throughput accepted by the network. Let $E_1, E_2, \ldots E_s$ be all the possible different sets of links in the network and τ_j be the average distance that packets traverse across links in E_j. Then, the maximum throughput in the network will be given by:

$$max_{th} = \frac{2}{N} \min \left\{ \frac{|E_j|}{\tau_j} \right\}, E_j \subseteq E$$

This calculation is valid as long as all nodes are limited by the subsets selected, which happens in all cases considered in this paper. For example, it was shown in [9] that under uniform traffic $\tau_{max} = \max(\tau_h, \tau_v)$ serves to calculate the maximum throughput in RTs and RTTs, where $\tau = \tau_h + \tau_v$ represents the division of the average distance on the horizontal and vertical dimensions. This is true because all the links in a dimension (horizontal or vertical) are used in the same proportion under uniform traffic. However, when a mapping algorithm and local traffic are taken into account, the internal and peripheral links within a given dimension can receive different loads. In such case, the maximum throughput is determined by the subset of links receiving the higher load. Therefore, in order to estimate the maximum throughput, we need to determine the subset of links that will first saturate. Specifically, we will denote E_{hi} and E_{vi} the sets of horizontal and vertical internal links, and E_{hp} and E_{vp} the peripheral ones. In the RT and RTT $|E_{hi}| = 2a^2 - a$, $|E_{vi}| = 2a^2 - 2a$, $|E_{hp}| = a$ and $|E_{vp}| = 2a$.

As before, τ_j can be divided in its local and global components: $\tau_j = \tau_j^l + \tau_j^g = \alpha \cdot d_j + (1-\alpha)\bar{k}_j^g$, where d_j represents the average number of hops of local packets in E_j. In our model the global communications are approximated by uniform traffic, so $\bar{k}_j$ is independent of the mapping, and can be derived from the values given in Table 1 and the specific $|E_j|$. For example, if $E_j = E_{hj}$, then $\overline{k_j} = \overline{k_x} \cdot |E_{hj}|/|E| = \overline{k_x} \cdot \frac{2a^2-a}{2a^2} = \overline{k_x} \cdot \left(1 - \frac{1}{2a}\right)$.

The next subsections will calculate the expected latency and maximum throughput of applications mapped into RT and RTT. We will first consider standard topologies and next the case of concentration $c = 2$.

3.3. Mapping 2D Logical Tori into Standard RT and RTT

In this section we apply the previous model to estimate the latency and maximum throughput when the logical topology is a 2D ($2a \times a$) torus with the same number of tasks as nodes in the network. When the physical topology is a RT, the *id* mapping is the only one that makes sense, since f^d would otherwise break the locality. In the RTT, we will consider both *id* and f^d.

3.3.1. *id* Mapping of 2D Tori into RT

In this case the communication graph coincides with RT(a), so with the mapping function *id* the locality is preserved and the dilation is $d = 1$. Global traffic follows a uniform distribution with $\bar{k} = \frac{3a}{4}$, so base latency can be calculated from:

$$\tau = \tau^l + \tau^g = \alpha \cdot d + (1-\alpha) \cdot \bar{k} = \alpha + (1-\alpha) \cdot \frac{3a}{4}$$

To determine the maximum throughput we consider the sets of horizontal and vertical links, E_h and E_v. The average distance of local traffic is $(0.5, 0.5)$. As a consequence, $\tau_h^l = \tau_v^l = 0.5\alpha$. Table 1 provides the average distances of global traffic in each dimension, so:

$$\tau_h^g = (1-\alpha)\overline{k_x} = \frac{a}{2}(1-\alpha)$$

$$\tau_v^g = (1-\alpha)\overline{k_y} = \frac{a}{4}(1-\alpha)$$

Horizontal links E_h are the ones which first saturate with $\tau_h = \frac{1}{2}\alpha + \frac{a}{2}(1-\alpha)$, and the maximum throughput is:

$$max_{th} = \frac{2}{N}\frac{|E_h|}{\tau_h} = \frac{2}{2a^2}\frac{2a^2}{\frac{1}{2}\alpha + \frac{a}{2}(1-\alpha)} = \frac{4}{\alpha + a(1-\alpha)}$$

This expression shows that under local traffic ($\alpha = 1$) up to 4 phits/(node·cycle) can be accepted, since communication occurs with the four direct neighbors on independent links. Under uniform traffic, the maximum load will be $\frac{4}{a}$.

3.3.2. *id* Mapping of 2D Tori into RTT

In this case the locality of the application is broken. The internal and horizontal peripheral links preserve locality. By contrast, peripheral vertical communications which would follow the path $(0,1)$ in the logical graph are transformed into routes $(0, -(a-1))$ in the physical network (and reciprocally for peripheral hops $(0,-1)$), with maximum dilation $(a-1)$. This happens in a fraction $1/a$ of the vertical local messages in the network, so the average dilation is $d = \frac{1}{2a} \cdot (a-1) + \frac{2a-1}{2a} \cdot 1 = \frac{3}{2} - \frac{1}{a}$. With the value of $\bar{k}$ from Table 1 we obtain:

$$\tau = \tau^l + \tau^g = \alpha \cdot d + (1-\alpha) \cdot \bar{k} = \left(\frac{3}{2} - \frac{1}{a}\right)\alpha + (1-\alpha)\frac{2a}{3}$$

We will calculate which dimension determines maximum throughput. The local load on horizontal links is the same as in the case of *id* mapping on RT, so using the value $\overline{k_x} = \frac{a}{3}$ we obtain:

$$max_{th} \leq \frac{2}{N}\frac{|E_h|}{\tau_h} = \frac{4}{\alpha + \frac{2a}{3}(1-\alpha)}$$

On vertical links, all the local traffic is sent on the internal links E_{vi}. Similarly to the dilation calculation, the average distance of local traffic on E_{vi} will be $\tau_{vi}^l = (1-\frac{1}{a})\frac{1}{2}\alpha + \frac{1}{a}\frac{(a-1)}{2}\alpha = (1-\frac{1}{a})\alpha$.

Global traffic uses every vertical link equally, which implies that $\tau^g_{vi} = \overline{k_y}\frac{|E_{vi}|}{|E_v|}(1-\alpha) = \frac{a}{3}(1-\frac{1}{a})(1-\alpha)$. With these values we can determine that vertical links impose a lower limit on maximum throughput than horizontal links:

$$max_{th} = \frac{2}{N}\frac{|E_{vi}|}{\tau_{vi}} = \frac{2}{N}\frac{|E_{vi}|}{(1-\frac{1}{a})\alpha + \frac{a}{3}(1-\frac{1}{a})(1-\alpha)} = $$
$$= \frac{4}{2\alpha + \frac{2a}{3}(1-\alpha)}.$$

3.3.3. f^d Mapping of 2D Tori into RTT

Using the mapping f^d with the RTT, horizontal locality is preserved and vertical locality suffers dilation 2, with local traffic on vertical links requiring routes $\pm(1,1)$. Average dilation is 3/2. The distances for local traffic are:

$$\tau^l_h = \alpha \qquad \tau^l_v = \frac{\alpha}{2} \qquad \tau^l = \frac{3\alpha}{2}$$

Using the values from Table 1 we get: $\tau = \tau^l + \tau^g = \frac{3\alpha}{2} + (1-\alpha)\cdot\frac{2a}{3}$ To calculate maximum throughput we observe that distances on the horizontal dimension are longer: $\tau_h = \tau^l_h + \tau^g_h = \alpha + \frac{a}{3}(1-\alpha)$. Then we obtain:

$$max_{th} = \frac{2}{N}\frac{|E_h|}{\tau_h} = \frac{4}{2\alpha + \frac{2a}{3}(1-\alpha)}$$

Figure 5 shows the throughput and latency results when mapping a 2D logical torus into RT and RTT. The identity mapping *id* provides the best results in RTT. With this mapping, the RTT achieves better throughput and latency when global communications dominate with up to 50% throughput improvements. However, the *id* mapping in RTT has a maximum dilation of $a-1$. The diagonal-shift mapping f^d minimizes the maximum dilation on the RTT(4) and it obtains the same throughput as *id* but worse average latency. Both curves intersect in $\frac{a}{a+3}$, which tends to 1 for larger networks. As a consecuence, the RTT will obtain better throughput than RT except for traffic with high locality (α).

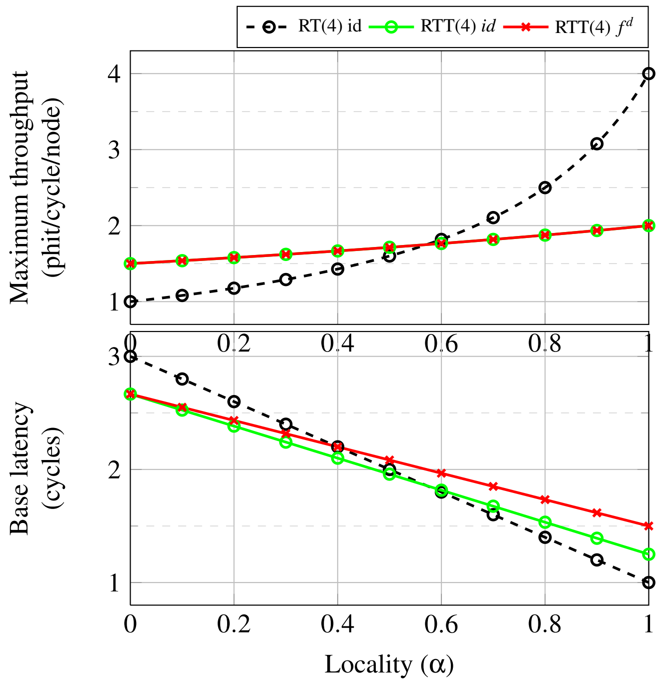

Figure 5. Maximum throughput and latency for logical torus mapped on RT(4) and RTT(4)

3.4. Mapping 2D Logical Tori into Concentrated RT and RTT

We will consider now the case of an application with more processes than routers in the physical topology. We will restrict our calculations to an application whose local communication graph is a square torus $2a \times 2a$ and concentration $c = 2$ has to be employed. Cases with larger concentration can be calculated similarly. Two concentration functions can be employed to reduce the vertical dimension, f^y_2 and f^t_2. Then, any mapping can be applied, leaving 4 possibilities: f^y_2, $f^d \circ f^y_2$, f^t_2, $f^d \circ f^t_2$, but we omit the latter because both the concentration f^t_2 and mapping f^d are designed to cope with twisted peripheral links. In the RT, f^y_2 is the only sensible combination, since it exploits the maximum locality. We will study their performance next.

3.4.1. f^y_2 Concentration of 2D Tori into RT

In this case locality is preserved similarly to the *id* mapping in RT, with two vertical neighbour processes mapped into the same network node. From every 8 local communications from each node, 2 are internal to the node, 2 imply a vertical hop and 4 imply a horizontal jump.

$$\tau^l_h = \frac{4}{8}\alpha = \frac{\alpha}{2} \qquad \tau^l_v = \frac{2}{8}\alpha = \frac{\alpha}{4} \qquad \tau^l = \frac{3\alpha}{4}$$

Average dilation is $d = 3/4$, lower than 1 thanks to neighbor nodes being concentrated together. With respect to global traffic, the values in Table 1 remain approximately valid when using concentration, so we can get the estimation of average latency:

$$\tau = \tau^l + \tau^g = \frac{3\alpha}{4} + \frac{3a}{4}(1-\alpha)$$

Regarding throughput, it is straightforward that horizontal links are saturated first since both local and global average distances are larger in X than in Y. We obtain the same result as in RT without concentration:

$$\tau_h = \tau^l_h + \tau^g_h = \frac{\alpha}{2} + \overline{k_x}(1-\alpha) = \frac{\alpha}{2} + \frac{a}{2}(1-\alpha)$$

$$max_{th} = \frac{2}{N}\frac{|E_h|}{\tau_h} = \frac{4}{\alpha + a(1-\alpha)}$$

3.4.2. f_2^v Concentration of 2D Tori into RTT

Again, this case is similar to the *id* mapping in RTT without concentration, with locality preserved in the internal mesh but not in the vertical peripheral links. Now, of each 8 local communications of each node of the first and last rows ($\frac{2}{a}$ from total of rows) we have that 2 are internal to the network node, 4 are $(\pm1,0)$, 1 is $(0,\pm1)$ and 1 $(0,\pm(a-1))$. Therefore, vertical peripheral links are not used by local communications. The nodes in the internal rows $(1-\frac{2}{a})$ send 4 messages to $(\pm1,0)$ and 2 to $(0,\pm1)$ from each 8 messages. Then, average local distances are:

$$\tau_h^l = \frac{4}{8}\alpha = \frac{1}{2}\alpha$$
$$\tau_{vi}^l = \frac{a}{8}\frac{2}{a}\alpha + \frac{2}{8}\left(1-\frac{2}{a}\right)\alpha = \frac{a-1}{2a}\alpha$$
$$\tau^l = \tau_h^l + \tau_v i^l = \left(1-\frac{1}{2a}\right)\alpha$$

With the global values we can get the average distance in the network:

$$\tau = \tau^l + \tau^g = \left(1-\frac{1}{2a}\right)\alpha + (1-\alpha)\frac{2a}{3}$$

Local distances are larger in X than in Y, and global distances are balanced in the RTT, so the throughput will be determined by distances in horizontal links. With the value of $\overline{k_x}$ from Table 1 we get

$$max_{th} = \frac{2}{N}\frac{|E_h|}{\tau_h} = \frac{4}{\alpha + \frac{2a}{3}(1-\alpha)}$$

3.4.3. $f^d \circ f_2^v$ Mapping of 2D Tori into RTT

In this case horizontal locality is preserved but vertical locality is modified: two vertical neighbours are mapped into each node, so half of the vertical local messages are internal to the node. The remaining vertical communications suffer dilation 2, similar to the case of the f^d mapping in RTT without concentration. From each 8 local messages of each network node, 6 use horizontal links and 2 use vertical links, so average distances are:

$$\tau_h^l = \frac{6}{8}\alpha = \frac{3}{4}\alpha \qquad \tau_v^l = \frac{2}{8}\alpha = \frac{1}{4}\alpha$$

Then, $\tau^l = \alpha$ and base latency will be determined by

$$\tau = \alpha + \frac{2a}{3}(1-\alpha)$$

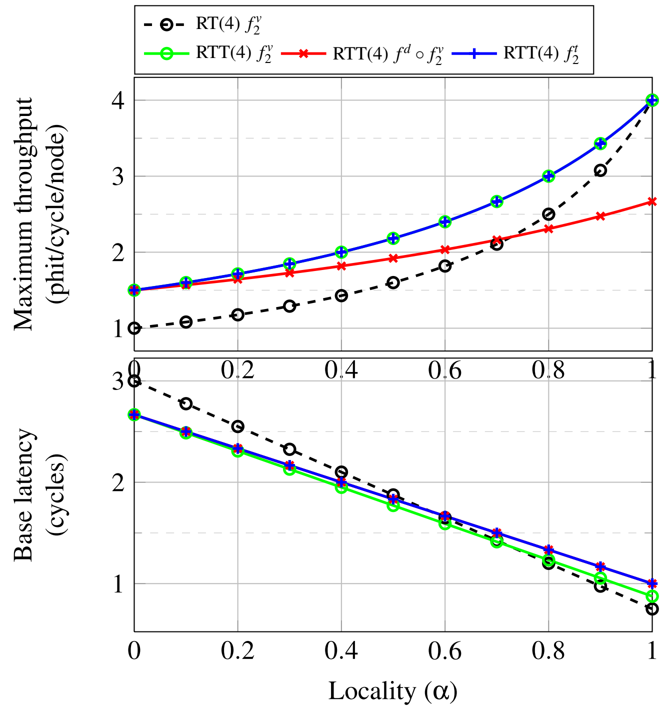

Figure 6. Maximum throughput and Latency for logical torus mapped on RT(4) and RTT(4) with concentration c=2.

Thus, the network throughput is limited by horizontal traffic and it is done in:

$$max_{th} = \frac{2}{N}\frac{|E_h|}{\tau_h} = \frac{2}{\frac{3}{4}\alpha + \frac{a}{3}(1-\alpha)} = \frac{4}{\frac{3}{2}\alpha + \frac{2a}{3}(1-\alpha)}$$

3.4.4. f_2^t Mapping of 2D Tori into RTT

In this case, the f_2^t mapping preserves the neighborhood in the original task graph, but no neighbours are collocated in the same node. The dilation of the network is $d=1$ in both dimensions, so $\tau_h^l = \tau_v^l = \frac{\alpha}{2}, \tau = \alpha$. The average distance will be determined by

$$\tau = \alpha + (1-\alpha)\frac{2a}{3}$$

Traffic is balanced, so we can consider any dimension as the throughput limiter. Global traffic is $\tau_h^g = \tau_v^g = \frac{a}{3}(1-\alpha)$. Then, the maximum network throughput will be:

$$max_{th} = \frac{2}{N}\frac{|E_h|}{\tau_h} = \frac{2}{\frac{\alpha}{2} + \frac{a}{3}(1-\alpha)} = \frac{4}{\alpha + \frac{2a}{3}(1-\alpha)}$$

Throughput and average latency results with $c=2$ are presented in Figure 6. In the RTT both the twisted f_2^t or vertical f_2^v concentrations (with *id* mapping) obtain the best results. The latency is better in the latter case, since the vertical concentration puts neighbor processes together, reducing the amount of local communications in the network. However, the f_2^t concentration obtains maximum dilation 1, while in f_2^v it is $a-1$. Interestingly, both concentrations on the RTT obtain better throughput than the RT for any locality value, and the average base latency is similar in all cases (slightly better for uniform traffic and slightly worse for local traffic).

4. EXPERIMENTAL ENVIRONMENT

The previous section presented an analytical study showing that the RTT can be competitive against the RT. However, it did not consider the impact of other factors such as maximum dilation, remote communications or the network load. In section 5. different RT and RTT configurations using synthetic traffic and real applications from the NPB suite [4] will be evaluated. Hence, in the present section we introduce the configuration of our experiments.

4.1. Workloads

First, we use independent traffic sources under random traffic. In this case, a ratio $1-\alpha$ of packets are distributed evenly along the whole network, while a ratio α of packets is sent to neighbor nodes. The inter-injection interval at each node is random following a Poisson distribution chosen as to modulate the provided load in terms of phits/cycle/node. Some parallel applications exhibit traffic patterns in which nodes communicate with their nearest neighbors in a torus topology. This can be either due to the inherent symmetry of the application or because of mapping big data matrices on the network nodes. For that reason, we include in this study nearest-neighbor (NN) communication patterns.

The NPB suite is typically employed in large parallel systems and is representative of general HPC applications running in those systems. We use the Extrae MPI tracing tool [14] to obtain traces from NPB applications using problem sizes A, running on 32, 64 and 128 nodes of a supercomputer based on IBM JS21 blades. We make use of the task mapping algorithms presented in the previous section to assign processes to computation nodes. We measure the execution time (in cycles) of each application, which is the parallel section between calls to MPI_Init and MPI_Finalize. Execution time will be normalized to the base case of a RT with identity mapping.

4.2. Simulation Configuration

We simulate the benchmarks with different sizes and concentration levels employing the FSIN simulator [20] with the parameters presented in Table 2. For this study, the router employed is similar to the one implemented in the IBM BlueGene/L: virtual cut-through switching strategy, [17], and bubble flow control deadlock avoidance, [1], with an static virtual channel plus two fully adaptive virtual ones. BlueGene family of supercomputers implements a congestion control mechanism that prioritizes in-transit traffic against new injections, which is also implemented in our router. In our experiments, packets have a fixed length of 64 phits of 4 bytes each. The FSIN tool simulates traces of applications, preserving causal dependencies between messages and modeling computation time. We extended FSIN to support MPI collective communications: each collective primitive is implemented as a series of unicast messages (e.g., broadcast) or a series of pairs of messages sent between different pairs of nodes (e.g. all-to-all), which are common implementations in networks without multicast support.

Table 2. Simulation parameters

Processor Frequency	2 GHz	Virtual Channels	3
Phit size	4 bytes	Routing Mechanisms	adaptive
Packet size	64 phits	Arbitration mechanims	random
Link speed	1 Gbps	Deadlock avoidance	bubble

5. PERFORMANCE EVALUATION

This section is organized into two parts, each one devoted to experiments driven by traffic of different nature.

5.1. Synthetic Traffic

In this subsection we corroborate the analytical model presented in Section 3. with simulation results using synthetic traffic. We model a synthetic traffic in which each process communicates with one of his neighbors with probability α. With probability $1-\alpha$ the packet is sent to a random process, not necessarily one of the four neighbors. The application is mapped to the physical network with different concentrations (1 or 2 processes per node) and different mapping functions.

When measuring the maximum throughput of the network, we make use of 4 injectors, similar to the BlueGene/Q chips [11], and packets with a length of 4 phits. Multiple injectors are required to saturate the network with local traffic, since α close to 1 can provide throughput up to 4 phits/(node$\cdot$ cycle). When measuring minimum latency, we inject a load of 0.01 packets per node per cycle, each of length 1 phit. In this way we eliminate the delays due to network congestion and packet consumption time, allowing us to measure the minimum average latency to transit the network.

The results of a 16×8 toroidal application over a 16×8 network are shown in Figure 7. Both throughput and latency results are really close to the ones predicted by the analytical model shown in Figure 5.

The same results for a network with concentration ($c = 2$) are shown in Figure 8. In this case, a 16×16 toroidal application is mapped onto a 16×8 network. For both throughput and minimum latency, results are very similar to the ones predicted in Figure 6 with the analytical model.

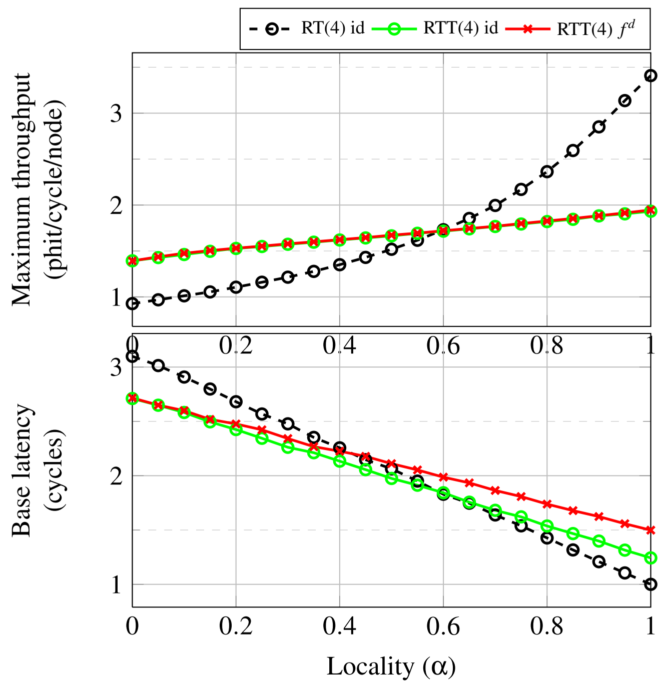

Figure 7. Simulation results for latency and maximum throughput for logical torus mapped on RT(4) and RTT(4).

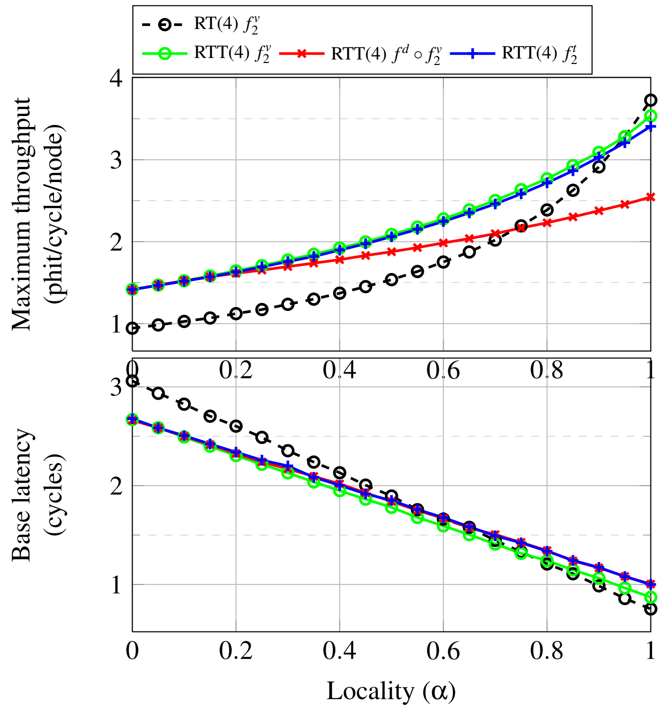

Figure 8. Simulation of base latency and maximum throughput for logical torus mapped on RT(4) and RTT(4) with concentration c=2.

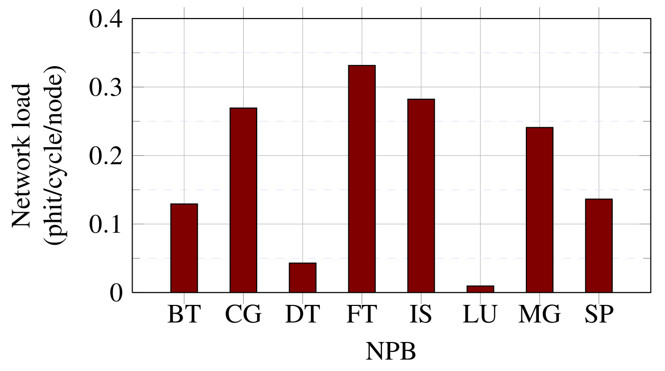

Figure 9. Network load for 64 tasks mapped onto a RT(4) with $c = 2$

5.2. Real Applications Traffic

5.2.1. Communications Characterization

The NPB applications and their communication patterns have been largely studied in the scientific literature [18, 19, 22]. From these sources and an analysis of the applications, we observe that CG and LU employ 2D meshes as their base logical topology, while BT and SP employ square 2D torus. By contrast, MG is a 3D torus, IS is a cycle, and the remaining applications are unstructured (DT and FT). Besides, some applications require a square number of processes. In all cases we evaluate the performance with all the mapping algorithms studied, considering all the logical tasks as an array of consecutive nodes.

The usage of the network limits the maximum performance differences between configurations. The traffic load of each application can be easily measured by simulation. Figure 9 shows the load measured when running on different networks with 64 tasks and concentration $c = 2$. Results for 32 or 128 tasks are similar. We observe that CG, FT, IS and MG are the applications with the highest network load. Therefore, the theoretical throughput results obtained by our model should be applicable to them.

Besides the network load, the performance can be limited by latency when there are multiple dependency chains among messages. Thus, a low average network load does not imply that the network is irrelevant: it can be either inactive (this occurs in the EP benchmark, omitted for this reason) or limited by latency. This is the case, for example, of DT: although having a low throughput (below 5%), the performance increase obtained by the RTT is above this 5%. Specifically, DT is a data traffic benchmark with large amounts of messages sent between nodes according to a certain pattern (we employed black hole), what introduces a large amount of dependencies in the traffic traces.

In general BT, LU, and SP use mainly near-neighbor communications (α close to 1), while DT, EP, FT and IS use more

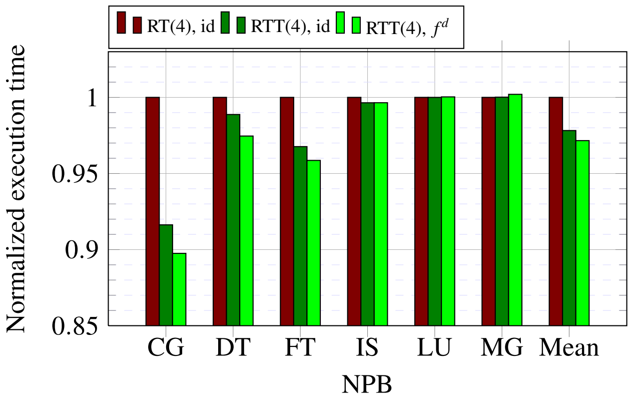

Figure 10. Execution time for 32 processors mapped onto a RT(4) or RTT(4) with $c = 1$

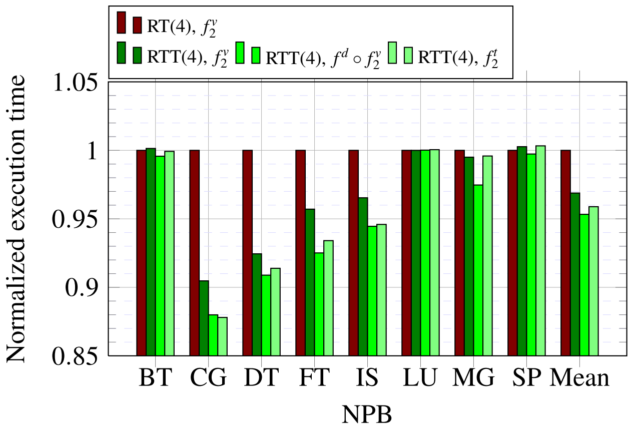

Figure 11. Execution time for 64 processes mapped onto a RT(4) or RTT(4) with $c = 2$

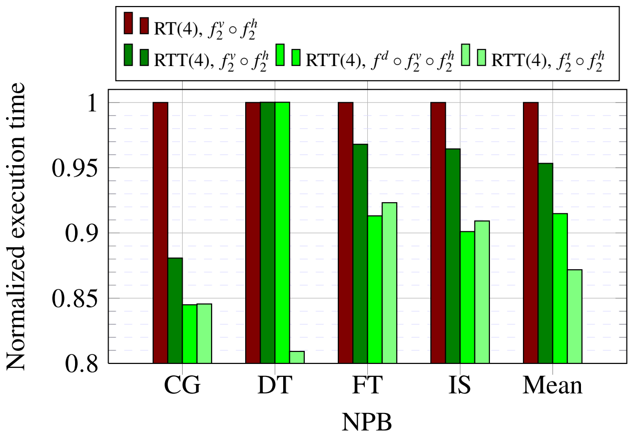

Figure 12. Execution time for 128 processes mapped onto a RT(4) or RTT(4) with concentration $c = 4$

global communications (α closer to 0). The communication in CG occurs between certain pairs of nodes, not necessarily neighbors (remote messages). Finally, the consecutive node labelling and task mapping on a 2D network does not preserve adjacency of the 3D torus of MG.

5.2.2. Performance Evaluation

First, we consider the non-concentrated scenario. Figure 10 shows the performance obtained when running NAS benchmarks of 32 processes on RT(4) and RTT(4). We are restricted to those benchmarks that allow a number of processes which is not a square number. We observe that the RTT always performs equal or better than the RT counterpart except just a slight loss in one case; CG and FT are the applications in which the performance improvement is higher, saving up to 10% of the execution time. Not surprisingly, these applications contain a large amount of global communications. LU and MG are the ones with the worse performance, without any improvement or even a slight loss of less than 0.1%. On average, the use of the RTT improves execution time in 2.2%, and f^d behaves slightly better than *id* for the RTT.

Next, several concentration techniques are evaluated. Figure 11 shows the execution time of NAS benchmarks running with 64 processes, mapped onto a RT(4) or RTT(4) with two compute nodes per network router. On average the RTT outperforms the RT. Interestingly, the f_2^t concentration function, which does not concentrate neighbor tasks, provides one of the best results thanks to the arrangement of tasks in relation to peripheral links, similar to $f^d \circ f_2^v$. Note that BT and SP, which employ 2D logical torus, do not vary significantly from the base case when using an RTT. On average, the applications running on the RTT save between 3.0% and 4.6% of the overall execution time.

Finally, Figure 12 presents the results of non-square applications with 128 tasks (8×16) running on a 4×8 network[1]. We employ $f_2^v \circ f_2^h$ and $f_2^t \circ f_2^h$ combinations for concentration 4. Again, the RTT outperforms RT. The execution time savings of the RTT range from 8.7% (FT) to 19.1% (CG) using the best combination. On average, the best performance is obtained with the $id \circ f_2^t \circ f_2^h$ combination for the RTT, with an speedup of 12.8%.

The results are coherent with our analytical model. DT, FT and IS employ a large amount of collective communications, translating into a larger performance on RTT, especially on concentrated networks. CG sends remote messages and is also benefited by the distance reduction in the RTT. By contrast, BT, LU and SP employ local communications, and thus the performance on RTT is similar to RT. Regarding concentrated networks, we observe that in RTT the use of either the diagonal-shift mapping f^d or the twisted concentration f_2^t improves performance.

[1] LU and MG, are omitted because of technical difficulties. Results should be similar to the ones presented in Figure 10.

6. CONCLUSIONS

This paper makes a first exploration to mapping functions for rectangular torus topologies with peripheral twists. As we have proved, in non concentrated topologies the performance gain depends on the local traffic amount. On the other hand, with concentration RT always improves performance if the mapping technique is correctly chosen.. Particularly, we have given a theoretical study that shows that simple mapping algorithms obtain the maximum performance, with speedups ranging from -10% to 50% depending on the locality of the communications and the application topology. When concentrated tori are employed, proper concentration and mapping functions prevent this performance loss by compensating the effect of the twisted peripheral links.

Those numbers reflect the performance of applications bounded by the network. However, real applications alternate computation and different communication patterns on different phases. When simulating real applications (from the NPB benchmarks) the topological advantages of the RTT translate to average performance gains of 2.2-13.2% depending on the specific configuration.

Future work involves the extension of this study to a higher number of dimensions, which can achieve even higher speedups according to previous works [9], and the validation of results with larger applications. We are also interested in extending the twisted peripheral links model to partitioned architectures, such as the multi-toroidal topolgy in the BlueGene/L and /P [26].

ACKNOWLEDGMENTS

This work has been supported by the Spanish Ministry of Science under contracts TIN2010-21291-C02-02, TIN-2007-60625, AP2010-4900 and CONSOLIDER Project CSD2007-00050, and by the European HiPEAC Network of Excellence. M. Moretó is supported by a MEC/Fulbright Fellowship.

REFERENCES

[1] N. R. Adiga et al. Blue gene/l torus interconnection network. *IBM J. Res. Dev.*, 49(2):265–276, Mar. 2005.

[2] T. Agarwal, A. Sharma, and L. V. Kalé. Topology-aware task mapping for reducing communication contention on large parallel machines. In *IPDPS*, 2006.

[3] K. Asanović et al. The landscape of parallel computing research: A view from Berkeley. Technical report, UCB/EECS-2006-183, 2006.

[4] D. Bailey, T. Harris, W. Saphir, R. Van Der Wijngaart, A. Woo, and M. Yarrow. The NAS parallel benchmarks 2.0. Technical report, NAS-95-020, NASA Ames Research Center, 1995.

[5] G. Barnes, R. Brown, M. Kato, D. Kuck, D. Slotnick, and R. Stokes. The Illiac IV computer. *IEEE Trans. Comput.*, C-17(8):746–757, aug. 1968.

[6] A. Bhatele. Topology aware task mapping. In D. Padua, editor, *Encyclopedia of Parallel Computing*, pages 2057–2062. Springer US, 2011.

[7] S. H. Bokhari. On the mapping problem. *IEEE Trans. Comput.*, 30(3):207–214, Mar. 1981.

[8] J. M. Cámara, M. Moretó, E. Vallejo, R. Beivide, J. Miguel-Alonso, C. Martínez, and J. Navaridas. Mixed-radix twisted torus interconnection networks. In *IPDPS*, pages 1–10, 2007.

[9] J. M. Cámara, M. Moreto, E. Vallejo, R. Beivide, J. Miguel-Alonso, C. Martínez, and J. Navaridas. Twisted torus topologies for enhanced interconnection networks. *IEEE Trans. Parallel Distrib. Syst.*, 21:1765–1778, 2010.

[10] C. Camarero, C. Martínez, and R. Beivide. L-networks: A topological model for regular two-dimensional interconnection networks. *IEEE Trans. Comput.*, 99(PrePrints), 2012.

[11] D. Chen et al. The IBM Blue Gene/Q interconnection fabric. *Micro, IEEE*, 32(1):32 –43, jan.-feb. 2012.

[12] Y. Chen and H. Shen. Embedding meshes and tori on double-loop networks of the same size. *IEEE Trans. Comput.*, 60(8):1157–1168, Aug. 2011.

[13] P. Colella. Defining software requirements for scientific computing. slide of 2004 presentation included in David Patterson's 2005 talk, 2004.

[14] Extrae MPI profiling tool. `http://www.bsc.es/ssl/apps/performanceTools/`.

[15] M. Flahive and B. Bose. The topology of gaussian and eisenstein-jacobi interconnection networks. *IEEE Trans. Parallel Distrib. Syst.*, 21(8):1132 –1142, 2010.

[16] H. Kasahara and S. Narita. Practical multiprocessor scheduling algorithms for efficient parallel processing. *IEEE Trans. Comput.*, 33(11):1023–1029, Nov. 1984.

[17] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[18] J. Kim and D. Lilja. Characterization of communication patterns in message-passing parallel scientific application programs. *Network-Based Parallel Computing Communication, Architecture, and Applications*, pages 202–216, 1998.

[19] I. Lee. Characterizing communication patterns of NAS-MPI benchmark programs. In *Southeastcon*, pages 158–163, 2009.

[20] J. Navaridas, J. Miguel-Alonso, J. A. Pascual, and F. J. Ridruejo. Simulating and evaluating interconnection networks with INSEE. *Simulation Modelling Practice and Theory*, 19(1):494–515, 2011.

[21] M. L. Norman and A. Snavely. Accelerating data-intensive science with Gordon and Dash. In *TeraGrid*, 2010.

[22] R. Riesen. Communication patterns. In *IPDPS*, 2006.

[23] C. H. Sequin. Doubly twisted torus networks for VLSI processor arrays. In *ISCA*, pages 471–480, 1981.

[24] E. Vallejo, M. Moretó, C. Martínez, and R. Beivide. Peripheral twists for torus topologies with arbitrary aspect ratio. In *Actas XXII Jornadas de Paralelismo*, pages 421–426, 2011.

[25] Y. Yang, A. Funahashi, A. Jouraku, H. Nishi, H. Amano, and T. Sueyoshi. Recursive diagonal torus: An interconnection network for massively parallel computers. *IEEE Trans. Parallel Distrib. Syst.*, 12:701–715, 2001.

[26] H. Yu, I.-H. Chung, and J. E. Moreira. Topology mapping for Blue Gene/L supercomputer. In *SC*, 2006.