# Parallel applications mapping over mesh topologies and its effects on scheduling performance

Jose A. Pascual, Jose Miguel-Alonso and Jose A. Lozano

Intelligent Systems Group

Computer Science and Engineering School

The University of the Basque Country

San Sebastian, Spain 20018

{joseantonio.pascual, j.miguel, ja.lozano}@ehu.es

## Resumen

The optimal mapping of tasks of a parallel program onto nodes of a parallel computing system has a remarkable impact on application performance. In this paper we propose a new criterion to solve the mapping problem in 2D and 3D meshes which takes into account the communication matrix of the application and a cost matrix that depends on the topology of the parallel system. Using simulation, we test the performance of optimization-based mappings, and compare it with consecutive and random trivial mappings using the NPB (NAS Parallel Benchmarks) applications. We also compare each application runtime on each topology to determine the best choice in order to improve the scheduling process.

## 1. Introduction

Message-passing parallel applications are composed of a collection of tasks that interchange information and synchronize among them using different communication patterns. These applications are often designed and developed with a parallel communications architecture in mind, and arrange the interchanges of data blocks using some scheme that tries to use efficiently the underlying network. The way tasks are arranged to perform communications is called the virtual topology.

These applications, once programmed, can be executed in a wide variety of parallel architectures, with different interconnection networks. These architectures vary from clusters with simple LAN-based networks to supercomputers with custom made networks organized as meshes, tori, trees, etc.

Given this variety of topologies, it is not uncommon to find a mismatch between the network architecture (machine-dependent) and the virtual topology (application-dependent). This may result in an inefficient utilization of the network, that materializes in large delays and bandwidth bottlenecks that, in turn, results in a general loss of application performance. The way in which tasks are mapped onto network nodes has a remarkable impact on the overall system performance [2] [13].

Current supercomputers generally share their resources between different users and applications. This means that the system can be running simultaneously several parallel jobs. A system scheduler is in charge of allocating jobs to resources, applying different policies to select the collection of nodes (partition) in which a given job (application) will run. In this dynamic environment, it is difficult to find a match between the physical topology of the selected partition and the applications virtual topology. Actually, many schedulers totally ignore the machines topology, considering a flat network. Some others search for contiguous partitions, a policy that can result in performance improvements due to the efficient exploitation

of communication locality [11] [14]. In any case, a mapping strategy is required to allocate tasks onto the assigned nodes. Ideally, this mapping should allow the tasks to make an efficient usage of the assigned resources.

The most used network topology [1] in current supercomputers is the 3D torus. This topology provides good characteristics like symmetry, low node degree and low diameter. However, the share of the resources between multiple applications results in the partition of the torus in 2D or 3D sub-meshes as depicted in Figure 1. Most of the research made in the partition of 3D topologies [6] focused on the search for 3D sub-topologies assuming that all applications will run faster in them. However, this assumption does not take into account the way that applications tasks communicate each other (communication pattern) that is, in fact, an important factor on applications performance.

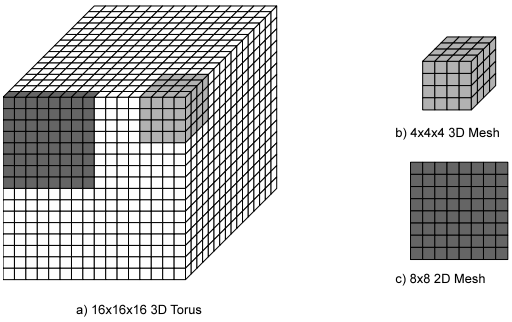


Figura 1: Examples of the partition of a 3D torus in 2D and 3D sub-meshes. a) 3D Torus (16x16x16) b) 3D sub-mesh (4x4x4) c) 2D sub-mesh(8x8)

In this paper, we adapt a previously developed mapping strategy [13] for 2D topologies to work with 3D topologies. The procedure is expressed as an optimization problem, in which we define some target functions to minimize. A first, classic strategy tries to minimize the average distance traversed by messages. A second approach, tries to reduce contention for the use of network resources and to avoid the creation of hot-spots, taking into account the communication patterns of the application tasks as well as the topology of the interconnection network; we call this the Traffic Distribution (TD) strategy. Once we have a mapping problem expressed as a function to minimize, a given mapping is better than another one if the value of the target function is smaller. However, one thing is to minimize a function and another thing is to obtain a mapping that, when used for a real application on a real machine, results in short execution times. Consequently, we have to validate the mappings, proving that a reduction in the target function actually makes the application run faster.

In order to carry out this validation we use simulation, due to its inherent flexibility for testing different system configurations. The simulation workbench is INSEE [15] which, given a trace of an application, a target network, and a mapping, can provide estimations of execution times. As applications we use (traces of) a subset of the well-known NAS Parallel Benchmarks (NPB) [10]. For the interconnection networks, we use two different topologies: 2D and 3D meshes. For simplicity, we consider only configurations in which the number of tasks equals the number of network nodes. Regarding mappings, we have tested some trivial ones (consecutive, random) and those provided by the optimization procedure, considering both the classic and the TD strategies.

Finally, we perform a direct comparison of the simulation results for each topology. The results will show that some applications run faster in 2D topologies due to its communication pattern that exploits the underlying topology. This fact is very important for the scheduling process and the inclusion of partition strategies onto it. These strategies will have to take into account the applications communication patterns in order to search for the best partition topology (2D or 3D). The inclusion of this partition strategy will result in a reduction of the external fragmentation [8] caused by the search for contiguous partitions.

The rest of the paper is organized as follows. In Section II we carry out a brief review of the mapping framework. Section III is devoted to

the extension of the mapping criterion to 3D network topologies. In Section IV we detail the experimental set-up used to generate and validate the results. Section V presents the results of the experimental work carried out. Finally, the paper ends in Section VI with some conclusions and lines of future work.

## 2. Optimization-based mapping framework

In this section we explain briefly our proposal of an optimization-based framework to find mappings vectors [13]. First we formalize the definition of the mapping problem, and then we describe the use of a GRASP (Greedy Randomized and Adaptive Search Procedure) algorithm to generate the mappings. The framework will be applied to dynamic environments where several applications share a machine, and the partition temporally assigned to a particular job can have an arbitrary topology given by the decision of the scheduler. In this case, in addition to obtain good solutions, we have time restrictions, in order to maintain the scheduling overheads low.

### 2.1. The mapping problem

The mapping problem can be formally defined as follows: given a set of tasks belonging to a parallel job $T = \{t_1, ..., t_n\}$ and a set of processing nodes of a parallel computing system $P = \{p_1, ..., p_n\}$ find a mapping function $\pi$ that assigns a task $t$ to a node $p$

$$\begin{array}{rccc} \pi: & T & \longrightarrow & P \\ & t & \longrightarrow & \pi(t) = p \end{array}$$

trying to optimize a given objective function. Note that we use (network) node and processor interchangeably.

The mapping problem can be expressed easily as a QAP (Quadratic Assignment Problem) [12] in the matrix form: given $T$ and $P$ two equal size sets representing parallel application tasks and processors respectively, matrix $W = [w_{i,j}]_{i,j \in T}$ representing the number of bytes interchanged between each pair of tasks, and matrix $C = [c_{i,j}]_{i,j \in P}$ representing some cost characteristic involving pairs of network nodes, find the bijection $\pi : T \longrightarrow P$ that minimizes:

$$\sum_{i,j \in P} w_{i,j} \cdot c_{\pi(i),\pi(j)} \quad (1)$$

The formulation shown in Equation 1, that states the mapping problem as an instance of the QAP, allows us to leverage the strategies developed for the QAP to find solutions for the mapping problem. We need two data structures: the communication matrix $W$ representing the amount of information interchanged by the tasks, and the cost matrix $C$ which models the criterion selected to distribute the traffic across the network. Note that the first depends solely on the characteristics of the application, while the other depends on the characteristics of the network: topology and routing function. In order to fill $W$ for a particular application, we use traces containing all the interchanges of messages, counting the messages interchanged between each task pair multiplied by their lengths (in bytes). The procedure to fill $C$ depends on the criterion selected to create mappings represented as a cost function.

### 2.2. Mapping algorithm

In our experiments we used the GRASP solver, because we found in the literature that this is the algorithm that provides the best results known when solving the generic QAP. We have adapted it slightly in order to deal with the data structures of the mapping problem. GRASP is a constructive and a multi-step algorithm that iterates over two steps: in a first stage an initial solution is created by means of a adaptive greedy randomized algorithm following by a local search to improve that solution. A detailed explanation of the algorithm can be found in [13].

## 3. Mapping criteria on mesh topologies

The mapping problem has been stated often as a location problem [4], searching a map-

ping vector that minimizes the average distance traversed by application messages. This is what we call the *classic* (optimization-based) strategy. The result of using this strategy is a mapping that locates in neighbouring nodes those tasks that interchange large data volumes. Apparently, this is a good policy, because it exploits communication locality, reducing the utilization of network resources. However, experiments show that a reduction in average distance does not always result in a reduction of application execution time, because it may create contention hot-spots inside the network [2] [3]. For this reason, we propose a different cost matrix to be used instead of the distance matrix. With this matrix the criterion of minimizing average distance is relaxed, in order to favour communication paths that distribute traffic through the network, avoiding bottlenecks. We have called this the *Traffic Distribution* (TD) criterion.

The definition of both criteria, expressed as cost matrices, depends on topological characteristics of the target network in which applications will run. In this work we consider just one network topology: meshes with two and three dimensions. In these networks each node has an identifier $i$ in the range $0 \ldots N-1$. We will consider routing functions that make messages advance using shortest-path routes between source and a destination. The simplest form of routing is Dimension Order Routing (DOR) [5], in which messages traverse first all the necessary hops in each dimension, but we do not make any assumption regarding the routing algorithm, except that it must use minimal paths.

### 3.1. Classic Distance Criterion on Meshes

In a 3D mesh with $N = n \times n \times n$ the number of nodes, given two nodes with identifiers $i$ and $j$, the distance between them is given by Equation 2. Due to the limited space we will only show the equations for 3D topologies. To obtain the equations for 2D topologies, the $z$ term must be ignored.

$$\begin{aligned} dist(i,j) = {} & dist_x(i,j) \\ & + dist_y(i,j) \\ & + dist_z(i,j) \end{aligned} \tag{2}$$

where the first term of the sum is the number of hops through the X axis a message must traverse when going from $i$ towards $j$, the second term is the number of hops through the Y axis and the third term is the number of hops through the Z axis. Equations 3, 4 and 5 show terms $dist_x$, $dist_y$ and $dist_z$.

$$\begin{aligned} dist_x(i,j) = {} & |j \bmod n \times n \bmod n/n \\ & - i \bmod n \times n \bmod n/n| \end{aligned} \tag{3}$$

$$\begin{aligned} dist_y(i,j) = {} & |j \bmod n \times n \bmod n \bmod n \\ & - i \bmod n \times n \bmod n \bmod n| \end{aligned} \tag{4}$$

$$\begin{aligned} dist_z(i,j) = {} & |j/n \times n \bmod n \\ & - i/n \times n \bmod n| \end{aligned} \tag{5}$$

Therefore, the classic criterion to tackle the mapping problem identifies $C$ as the distance matrix, whose components can be filled using Equation 2 for each pair of source-destination nodes.

$$C = [c_{i,j}]_{i,j=1\ldots N} \text{ where } c_{i,j} = dist(i,j) \tag{6}$$

### 3.2. Traffic Distribution Criterion on Meshes

The main reason to use the classic criterion is clear: minimizing the distance should result in lower latencies (because latency depends on the number of hops) and in lower utilization of network resources, because the number of involved links will be minimized. However, the distance criterion hides a fundamental characteristic of interconnection networks: resources (routers, links) have to be shared between many simultaneous communications. It may happen that a given mapping that is optimal in terms of distance, results in multiple

communications contending for the use of a single link (or a small collection of links). The contention generated for the shared resources results in increased latencies.

We propose in this paper a new criterion for 3D meshes, expressed as an alternative cost matrix, that allows us to find mappings that, while not optimal in terms of distance, avoid network bottlenecks. In particular, we pay attention to the number of hops per dimension that messages have to travel. A 6-hop route in which all the hops are in the $X$ axis is worse than another 6-hop route with 2 hops per dimension for 3D meshes, because the latter imposes less pressure in the $X$ axis, spreading the utilization of network resources evenly among the three axes. Formalizing this, we favour those routes in which the number of hops per dimension is equalized – or, in other words, we penalize those routes in which the number of hops per dimension is not equal. The level of penalization depends on the level of asymmetry, as represented in Equation 7.

$$\begin{aligned} td(i,j) = & \; dist(i,j) \\ & + |dist_x(i,j) - dist_y(i,j)| \\ & + |dist_x(i,j) - dist_z(i,j)| \\ & + |dist_y(i,j) - dist_z(i,j)| \end{aligned} \quad (7)$$

Hence, when using the TD criterion for the resolution of the mapping problem using our framework, the cost matrix is defined as:

$$C = [c_{i,j}]_{i,j=1\ldots N} \text{ where } c_{i,j} = td(i,j) \quad (8)$$

## 4. Experimental Set-up

In this section we detail the collection of experiments that we carried out to evaluate the effectiveness of the two approaches to the mapping problem described in the previous section. We will use networks of size 64 with 2D (8x8) and 3D (4x4x4) mesh topology, in order to test the impact of network topology on that effectiveness.

### 4.1. Experimental Workbench

We use traces extracted from the NPB suite [10]. For each application we need to generate the traffic matrix $W$. We have done so for the class A size 64 instances of the benchmarks. These applications were run in a real parallel machine, in which traces of all the messages interchanged by tasks were captured. These traces contain the necessary information to fill $W$, and are also valid to be used within the INSEE simulation environment.

The QAP Solver implements the GRASP algorithm. It accepts as parameters matrix $W$ (the communication matrix of the application trace to evaluate, expressed in bytes), matrix $C$ (the cost matrix modeling a mapping criterion) and the number of iterations to be performed. The output is a mapping vector that associates each application task with one node of the network partition. The creation of this vector obeys the criterion represented in the cost matrix.

The simulation has been carried out using INSEE [15] [9]. This tool simulates the execution of a message-passing application on a multicomputer connected via an interconnection network. INSEE performs a detailed simulation of the interchange of the messages through the network, considering network characteristics (topology, routing algorithm) and application behaviour (causality among messages). The input includes the application's trace file and the mapping vector. The output is a prediction of the time that would be required to process all the application messages, in the right order, including causal relationships and resource contention. However, it only measures the communication costs, assuming infinite-speed CPUs.

### 4.2. Design of the Experiments

To perform the validation, we created a set of 50 different mappings for each NPB application, using the classic and the TD expressions of the cost matrix. Each of the 50 solutions were selected after 50 GRASP iterations.

In addition to the classic and the TD mapping strategies, we also evaluated the beha-

viour of the consecutive and random trivial mappings. Given a network partition, the consecutive mapping criterion allocates the application tasks onto the partition nodes in order of identifiers, starting with the assignment of the first task to the node with the lowest identifier, and ending with the assignment of the last task to the node with the highest identifier. In the case of the random criterion, given a network partition, application tasks are assigned to partition nodes randomly. To evaluate this strategy, a set of 50 different random assignments were performed.

## 5. Analysis of results

This section analizes the results of the simulations carried out with the previously generated mappings. First we analyze the mapping procedure itself for both 2D and 3D mesh topologies showing which mapping criteria performs the best results for each NPB application. In addition to this, we are also interested in determine the best topology for the execution of each application. For doing that, we compare each application runtime (measured in simulation cycles) on each topology (2D and 3D mesh).

### 5.1. Results for the mapping strategies in 2D and 3D meshes

The results of the experiments are depicted in the Figure 2 for 2D meshes and in Figure 3 for 3D meshes. Due to the limited space, we do not show the best, the average and the standard deviation of the execution times reported by the simulator for each application-topology-mapping combination. However, the statistical significance of the differences between the three non-consecutive mapping strategies (note that the consecutive has not random component) has been assessed running Kruskal-Wallis [7] paired tests on the results obtained for each of the traces. In the analysis of results we say that a criterion is better than other if exist significant statistical differences between both. The tests have been performed between the strategy that obtained the best result and each of the others, at level of significance 0.05.

We start analyzing the results obtained for the 2D mesh topology. It is clear that we can divide the applications in three groups. The first group is composed by BT, LU and SP. A property of these applications is that the communications pattern (virtual topology) matches the physical one. Therefore the consecutive mapping is clearly the best performer. In a second group we include the FT and IS applications. In this case, the best criteria are the random and the consecutive in both cases. However, the TD criterion performs almost the same results. The third group is composed by CG and SP. In this case is when the TD strategy shows its potential improving remarkably the results of the other three criteria.

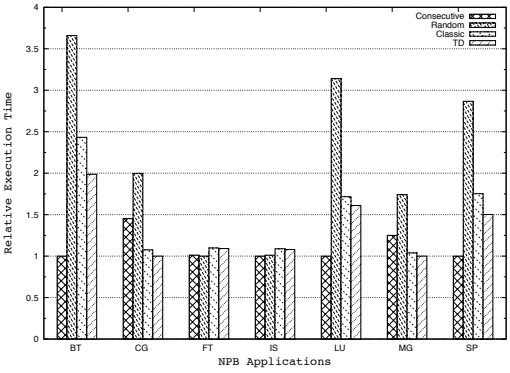


Figura 2: Execution times (simulation cycles) for the NPB applications (class A, size 64) running on a 8 x 8 mesh for each mapping criterion. Values are normalized, so that 1 represents the best result.

Let us pay attention now to results for 3D mesh topology. In this case the scenario is quite different from the previously explained. For all application-mapping combination the best performer is the TD criterion. Only in two cases, FT and MG, the random and consecutive criteria respectively, improve the results of the TD. In the case of the MG application the consecutive criterion is the best choice. The reason for this behaviour is the perfect match between its virtual topology and the underlying physical one. In the case of FT, as ocu-

rred for the 2D topology, the random criterion performs the best, due to its all-to-all communications pattern.

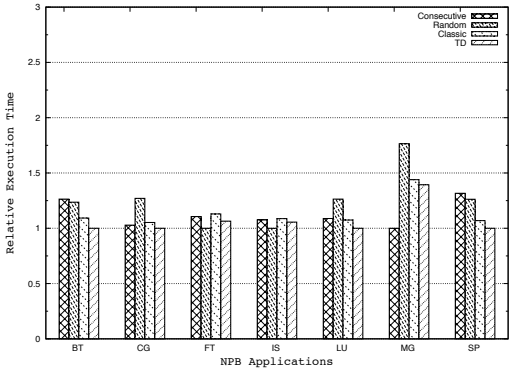


Figura 3: Execution times (simulation cycles) for the NPB applications (class A, size 64) running on a 4 x 4 x4 mesh for each mapping criterion. Values are normalized, so that 1 represents the best result.

In summary, the TD procedure provides good mappings of applications onto mesh topologies. The trivial mappings are able to produce better results in those cases in which the virtual network matches the actual one (consecutive mapping) and when the communications pattern performs mainly all-to-all communications (random mapping). We have to remark that the TD criterion performs better than the Classic in all cases.

### 5.2. Comparison of the NPB applications performance in 2D and 3D mesh topologies

As we stated before, we are interested in compare the performance of the same application over the two topologies analyzed in this paper. In the literature, generally, it is assumed that the execution of an application on 3D topologies will perform better than in 2D topologies. However, the results presented in this paper show an opposite behaviour in some cases. Figure 4 compares the best execution times (extracted from the previously explained results) for each application-topology.

The results show that in four cases, the 3D topology that provides better topologic characteristics, improves the performance. However, BT, LU and SP applications perform better over a 2D topology. The reason for this behaviour is their communications pattern and the way in which they have been implemented.

The search for the correct topology for each application can have a remarkable impact in the scheduling process (reducing the external fragmentation) in addition to the improvement in the application runtime.

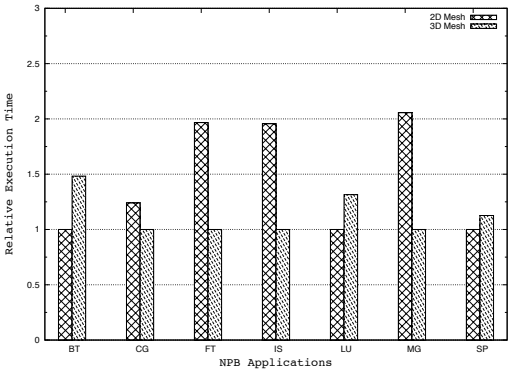


Figura 4: Comparison of the execution time (simulation cycles) of the NPB applications in 2D and 3D meshes. Values are normalized, so that 1 represents the best result.

## 6. Conclusions and future work

Current supercomputer centres share their resources between different users and applications. A system scheduler plays a crucial role, selecting the jobs to be executed (extracting them from a queue) and assigning some resources to it (nodes grouped as partitions). Often, the allocation/mapping procedures are carried out using simple mechanisms that ignore the communication patterns of the application and the topology of the network.

In this work we have focused on the mapping problem, describing a new criterion for 2D and 3D mesh topologies to find good task-to-node assignments that takes into account

the communication characteristics of the application, as well as the topology of the network. The choice of the mesh topology for this work is its importance in scheduling environments. The search for partitions in big size 3D torus or meshes will result in 2D or 3D submeshes where allocate the applications.

This paper also states the importance of searching for the correct topology for a concrete application that will result in the improvement of applications runtimes and in the reduction of the fragmentation in the scheduling process.

We plan to extend this work in other directions. In particular, we plan to investigate the way "smart", topology-aware partitioning schemas could be incorporated into schedulers. A combination of a good partitioning with an optimization-based mapping strategy could greatly enhance the productivity of large-scale parallel computers. In a previous work we have investigated the great potential of topology-aware partitioning [14], that can be exploited only if applications experiment a significant speed-up when tasks are located contiguously. If we incorporate better mappings, this requirement could be relaxed while still improving systems performance.

## Referencias


[1] . Top 500 list. `http://www.top500.org`.

[2] T. Agarwal, A. Sharma, A. Laxmikant, and L. V. Kalé. Topology-aware task mapping for reducing communication contention on large parallel machines. In *IEEE International Parallel and Distributed Processing Symposium*, page 122, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[3] A. Bhatele and L. V. Kalé. An evaluation of the effect of interconnect topologies on message latencies in large supercomputers. In *Proceedings of Workshop on Large-Scale Parallel Processing (IPDPS)*, May 2009.

[4] S. H. Bokhari. On the mapping problem. *IEEE Transaction on Computers*, 30(3):207–214, 1981.

[5] W. Dally and B. Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[6] M. Kang, C. Yu, H. Y. Youn, B. Lee, and M. Kim. Isomorphic strategy for processor allocation in k-ary n-cube systems. *IEEE Transactions on Computers*, 52:645–657, 2003.

[7] W. Kruskal and W. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47:583–621, 1952.

[8] V. Lo, K. Windisch, W. Liu, and B. Nitzberg. Noncontiguous processor allocation algorithms for mesh-connected multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, 8:712–726, 1997.

[9] J. Miguel-Alonso, J. Navaridas, and F. J. Ridruejo. Interconnection network simulation using traces of MPI applications. *International Journal of Parallel Programming*, 37(2):153–174, 2009.

[10] NASA Advanced Supercomputer (NAS) division. NAS parallel benchmarks. `http://www.nas.nasa.gov/Resources/Software/npb.html`, 2002.

[11] J. Navaridas, J. A. Pascual, and J. Miguel-Alonso. Effects of job and task placement on the performance of parallel scientific applications. In *Proceedings 17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 55–61, Washington, DC, USA, February 2009. IEEE Computer Society.

[12] P. M. Pardalos, F. Rendl, and H. Wolkowicz. The quadratic assignment problem: A survey and recent developments. In *Proceedings of the DIMACS Workshop on Quadratic Assignment Problems*, volume 16, pages 1–42, AMS, Providence,

USA, 1994. American Mathematical Society.

[13] J. A. Pascual, J. Miguel-Alonso, and J. A. Lozano. Optimization-based application framework for parallel applications. Technical report, The University of the Basque Country, 2010.

[14] J. A. Pascual, J. Navaridas, and J. Miguel-Alonso. Effects of topology-aware allocation policies on scheduling performance. In *Job Scheduling Strategies for Parallel Processing (IPDPS)*, volume 5798, pages 138–156. Springer-Verlag, Berlin, Germany, 2009.

[15] F. J. Ridruejo and J. Miguel-Alonso. INSEE: An interconnection network simulation and evaluation environment. In *Proceedings of the 11th international Euro-Par conference on Parallel Processing*, pages 1014–1023, Berlin, Germany, 2005. Springer-Verlag.