Adaptive Approaches for Efficient Parallel Algorithms on Cluster-based Systems

Wahid Nasri, Luiz Angelo Steffenel, Denis Trystram

Adaptive Approaches for Efficient Parallel Algorithms on Cluster-based Systems

Wahid Nasri*
Département d'Informatique,
École Supérieure de Sciences et Techniques de Tunis - ESSTT, Tunis, Tunisia
E-mail: Wahid.Nasri@ensi.rnu.tn
*Corresponding author

Luiz Angelo Steffenel
CReSTIC - SYSCOM,
Université de Reims Champagne-Ardenne, Reims, France
E-mail: Luiz-Angelo.Steffenel@univ-reims.fr

Denis Trystram
Laboratoire ID-LIG,
Institut National Polytechnique de Grenoble - INPG, Grenoble, France
E-mail: Denis.Trystram@imag.fr

Abstract: Few years ago, there was a huge development of new parallel and distributed systems. Due to many reasons, such as the inherent heterogeneity, the diversity, and the continuous evolution of such computational supports, it is very hard to solve efficiently a target problem by using a single algorithm or to write portable programs that perform well on any architecture. Toward this goal, we propose a generic framework combining communication models and adaptive approaches to deal with the performance modeling problem associated to the design of efficient parallel algorithms on grid computing environments, and we apply this methodology on collective communication operations. Experiments performed on a grid platform prove that the framework provides significant performances while determining the best combination model-algorithm depending on the problem and architecture parameters.

Keywords: Cluster computing, Performance modeling, Adaptive approaches, Polymodels of communications

Reference to this paper should be made as follows: Nasri, W., Steffenel, L.A. and Trystram D. (2008) 'Adaptive Approaches for Efficient Parallel Algorithms on Cluster-based Systems', Int. J. Grid and Utility Computing, Vol. X, No. Y, pp. Z–ZZ.

Biographical notes: Wahid Nasri is an Assistant-Professor at the Higher School of Sciences and Techniques of Tunis. He obtained a PhD in Computer Science in 2002 at the Faculty of Sciences of Tunis, with the collaboration of the ID-IMAG laboratory. His research interests are in high performance computing, including performance modeling, the design of efficient parallel algorithms, scheduling and load balancing for cluster computing. Currently, he is looking for models and adaptive techniques for modeling performances and adapting parallel poly-algorithms on grid environments.
Luiz Angelo Steffenel is an Assistant-Professor at the Université de Reims Champagne-Ardenne, France. He obtained a Ph.D. in Computer Science in 2005 at the Institut National Polytechnique de Grenoble, France, and a MSc. in Communication Systems in 2002 from the École Polytechnique Fédérale de Lausanne, Switzerland. His research interests include parallel and distributed systems, grid computing, scheduling algorithms, fault tolerance and pervasive computing. Dr. Steffenel actively contributes with french Grid'5000 and european CoreGRID projects.
Denis Trystram is Professor at INP Grenoble since 1991. He obtained a first PhD in Applied Mathematics at INPG in 1984 and a second one in Computer Science in 1988 from the same university. He is currently Regional Editor for Europe for the Parallel Computing Journal, belongs to the board of IEEE Transactions on Parallel and Distributed Systems and participates regularly to the Program Committee of the major conferences of the field. His research activities concern all aspects of the study on the design of efficient parallel algorithms. Today, his major interest is approximation algorithms for scheduling, multi-objective analysis and Game theoretic approaches with a special emphasis on grid computing.

1 INTRODUCTION

1.1 Recent Challenges of Efficient Parallel Algorithms

Over recent years, high performance computing has undergone impressive change. New architectures, like clusters and grids, and applications have rapidly become the central focus of the field. They have gained this prominence by providing reliable, robust and cost-effective platforms for solving many complex computational problems, accessing and visualizing data, and providing information service. Nevertheless, the ability to use efficiently both small and large systems is an ongoing effort in the areas of networking, management, interconnects and application optimization.

Today, with the spreading of complex architectures, we face a great challenge to model and optimize communications for high performance computing applications. Indeed, unlike dedicated parallel systems, new architectures are inherently heterogeneous and many of them are characterized by a hierarchical organization. They consist of a great diversity of resources of different performances interconnected via heterogeneous networks providing communication links with different latencies and bandwidths. Moreover, these systems are often upgraded with new generation of processors and network technologies. Due to the increasing diversity of existing parallel systems consisting on collections of heterogeneous machines, it is very difficult - and mostly impossible - for a user to choose an adequate algorithm because the execution supports are continuously evolving. One version will be well-suited for a parallel configuration and not for another one. This problem of portability becomes crucial with present architectures and applications.

These different elements require to revise the classical parallel algorithms and to develop more powerful approaches to efficiently use this type of platforms.

1.2 Contributions of this Work

Our objective within this work is to propose a generic framework based on communication models and adaptive techniques for dealing with performance modeling towards the design of efficient parallel algorithms, and integrating scalability and portability issues. The idea is to use several models for different parts of a large heterogeneous parallel computing platform and to associate them to some powerful adaptive approaches. More precisely, the contribution of this paper is to propose a two-level adaptive methodology. Indeed, at the first level, after a preprocessing phase required to reduce the impact of heterogeneity of a given architecture, we proceed by determining the more appropriate model from a set of selected ones to better predicting performances, as described later in section 3.2. Next, we will have to determine the best algorithm among multiple algorithmic options for resolving a given problem. Figure 1 illustrates this combination of performance modeling and adaptive computing which will lead to performance improvements of implemented parallel algorithms.

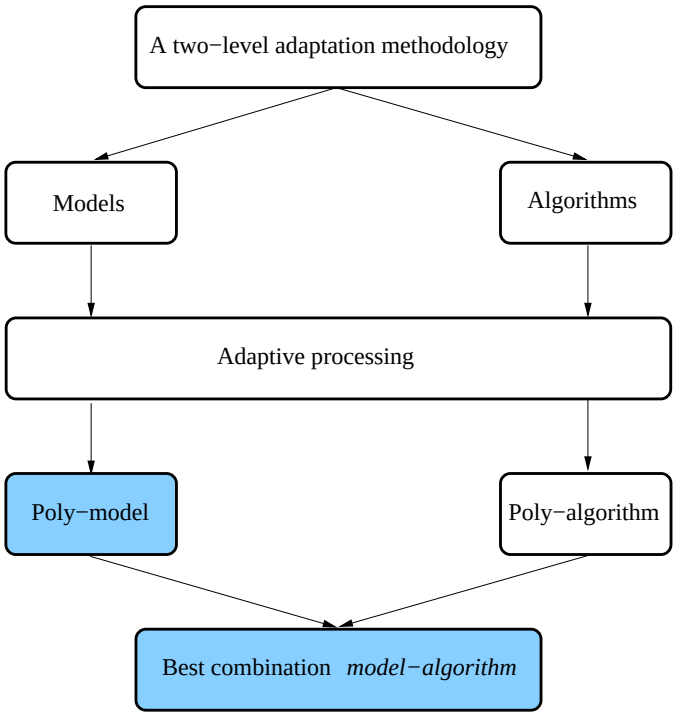


Figure 1: Adaptive computing and communications.

1.3 Related Works

There exist in the literature many works dealing with the use of adaptive techniques in order to ensure efficiency and portability. The target problems are various and the techniques are applied differently. For instance, works related to the parallel matrix multiplication are presented in Hong and Prasanna (2002); Nasri et al. (2007); Whaley et al. (2001) and those related to communication algorithms in Bhat et al. (1998); Hartmann et al. (2006); Pjesivac-Grbovic et al. (2004); Vadhiyar et al. (2004).

However, only few developed parallel adaptive algorithms are implemented as frameworks. We may particularly refer to works presented in Thomas et al. (2005); Cuenca et al. (2003). In Thomas et al. (2005), the authors have developed a general framework for adaptive algorithm selection using machine learning techniques to analyze data and to determine tests that will select among algorithmic options at run-time. Another methodology is described in Cuenca et al. (2003) where the authors presented the architecture of an automatically tuned linear algebra library on platforms where both the CPU load and the network traffic vary. During the installation process in a system, the linear algebra routines will be tuned automatically to the system conditions. At run-time, the parameters that define the system characteristics are adjusted to the actual load of the platform.

Our framework differs from the other works in a significant aspect. Indeed, existing adaptive approaches presented in the literature proceed by selecting one algorithm or eventually combining multiple algorithms from a library containing multiple algorithmic choices. Therefore, to the best of our knowledge, our framework provides the first general methodology for automatically associate the more appropriate model to the best algorithm among multiple model and algorithmic options. It determines the best combination model-algorithm and computes an efficient execution scheme that minimizes the overall execution time of a parallel application.

1.4 Organization of the Paper

The remainder of the paper is organized as follows. We begin, in section 2, by presenting the architectural model of the target parallel and distributed system and the performance evaluation models of a parallel algorithm. In section 3, we first define the concept of poly-model, present our adaptive framework for poly-models of communication, and detail its components. Section 4 is devoted to a case study where we apply our adaptive framework on collective communication operations. We present indeed numerical simulations and practical experiments performed on an heterogeneous hierarchical grid proving the interest of this work. Finally, we conclude the paper and discussing some perspectives to extend this work.

2 BACKGROUND AND MOTIVATION

2.1 Description of the Architectural Model

We assume in this work a generic model of a platform composed by heterogeneous hierarchical clusters. The platform studied enjoys heterogeneity along three orthogonal axes:

1. Computational Power: the processors that populate the clusters may differ in computational powers, even within the same cluster.

2. Interconnection Networks: the clusters comprising the platform are organized hierarchically and are interconnected via a hierarchy of networks of possibly different latencies, bandwidths and speeds.

3. Topology Heterogeneity: the clusters at each level of the hierarchy may differ in sizes.

2.2 Adaptive Approaches

It is well-known that no single algorithm can always achieve the best performance of a sequential or parallel application for different problem sizes and number of processors on a target parallel system. We can obtain good performances by mixing multiple algorithms for solving the same problem, where each algorithm can dominate the others in specific contexts. Thus, we should determine the more appropriate algorithm (which provides the best performance) in terms of a set of parameters (size of the problem, number of available processors, performances of the interconnection network, etc.), or to combine multiple ones for improving performances to fit well the characteristics of the target computational system.

The software mechanism responsible for determining the best available choices at run-time is known as a switching function. The optimal choice of algorithm can be determined at run-time, typically by using data obtained by monitoring tools, such as the NWS (Network Weather Service, Wolski et al. (1997)) which permits to measure many useful information, such as the hardware characteristics, the communication bandwidth, the system load, or any input-data that may influence the performance of the application. The result of this mechanism is called adaptive algorithm which encapsulate a number of algorithms for solving the same problem. This algorithm may use different techniques to adaptively determine the best algorithm. For instance, the algorithms presented in Thomas et al. (2005), Frigo and Johnson (1998) and Nasri et al. (2007) use respectively machine learning, cascading and poly-algorithmic techniques. In this work, we are interested in the adaptivity at the algorithmic level, which can be developed in the middleware (MPI collective communications, for example) or at the application level.

There exist other techniques for adapting software to execution contexts. In ATLAS library (Whaley et al. (2001)), all computations are decomposed into blocks whose size is automatically optimized specifically for the target processor. A more sophisticated technique of adaptive algorithms is cascading. Here, the computations are decomposed recursively up to a certain threshold is reached. For instance, in FFTW (Frigo and Johnson (1998)), a dynamic programming algorithm is run to determine the size for stopping the recursion. Such examples show that adaptive algorithms can frequently do as well as or even better than hand-tuned vendor code (McCracken et al. (2003)). Another available package related to adaptive algorithms is LAPACK for Clusters (Chen et al. (2003)). Other aspects on which algorithms may be adapted are data storage and communication patterns. Let us note that it is possible to include mechanisms for the adaptive processing into middleware or resource manager. Such approaches have been investigated for instance in McCracken et al. (2003).

2.3 Communication Models

There are many parallel communication models in the literature that analyze performances based on system parameters (see Alexandrov et al. (1995); Culler et al. (1996); Faik et al. (2005); Frank et al. (1997); Hockney (1994); Kielmann et al. (2001); Lastovetsky et al. (2006); Moritz and Frank (2001); Teresco et al. (2005)). More sophisticated models have been proposed for complex architectures (Cappello et al. (2005); Steffenel (2006)). All these models differ on the assumptions about the computational support parameters, such as latency, heterogeneity, network con-

tention, etc., and therefore are able to cover a great variety of architectures and modeling aspects. For instance, the selection between these models will depend on the data size to communicate, the accuracy of the models and their relative cost (parameters acquisition and complexity of models). We summarize in Table 1 a comparison between some of these models where we present the targeted platform of each model and the modeled parameters.

3 DESCRIPTION OF THE FRAMEWORK

3.1 Concept of Poly-Model

Generally, in order to predict and model the overhead due to communications in a parallel application, we use a single communication model, like, for instance, Hockney (Hockney (1994)), LogP (Culler et al. (1996)), pLogP (Kielmann et al. (2001)), etc. Due to the wide variety of existing parallel machines, which requires determining multiple parameters to get more precise results, a sophisticated model is necessary. On the other hand, using such a model on "simple" architectures is not useful due to the cost of determining their parameters. For that reasons, we introduce the concept of Poly-model which determines adaptively the more appropriate model for a target platform according to the problem and architecture parameters. In fact, the poly-model that we propose is equivalent to a combination of multiple models, to be used on different clusters. It chooses an adequate model in terms of several information, including the size of data to communicate, the type of interconnection network (with contention or not), the number of nodes, etc. Let us remark that the generated poly-model which uses adaptive techniques will:

1. Better model the communications in terms of the characteristics of the hardware resources of the target parallel system.
2. Reduces the cost of modeling parallel applications on complex architectures.
3. Provides precise prediction results.

We finish this description by mentioning that the complexity of the overall process of the scheme adaptation does not affect the global cost of the application to be modeled.

3.2 Overview of the Methodology

In this section, we describe our framework for adaptively modeling communications in an execution environment characterized by its heterogeneity and its hierarchical organization. An overview of the methodology is sketched in Figure 2. The processing is separated into two successive phases. During the first one, we aim to partition the target execution platform to form subnets of similar characteristics by automatically discover the network topology. Then, when executing the second phase, we have to determine for each subnet (i.e. cluster) the more appropriate communication model to better predicting performances. We will finally obtain a Poly-model of communication. This processing can be described by Algorithm 1.

Algorithm 1 General scheme of the methodology

```
Call NWS
Call a clustering algorithm
Do for each obtained logical cluster
    Determine the model (according to data size and network
performances)
    Obtain the parameters of the model (call logp_mpi routine of
MagPIe library - Kielmann et al. (2001))
End do
Obtain the parameters of the model used inter-clusters (according to
techniques in Steffenel (2006))
```

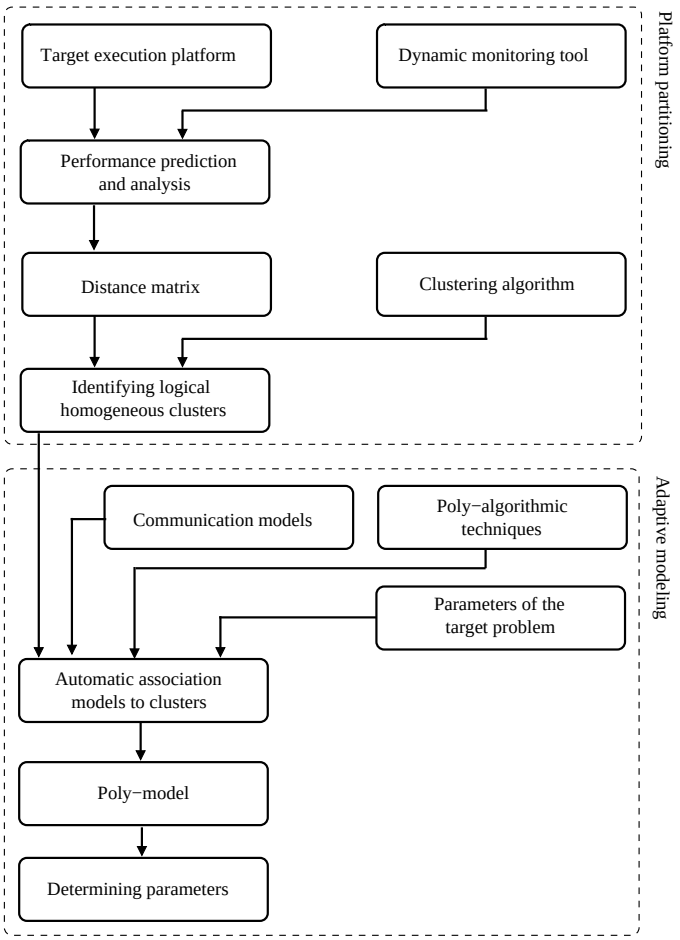


Figure 2: Adaptive framework.

In the sequel, we give more details on the major components of the framework.

3.3 Platform Partitioning

Since the target parallel system may be heterogeneous at many levels (computing powers, interconnection network

Table 1: Communication models for performance prediction

	Heterogeneous environment	Hierarchical	Latency	Gap	Sender overhead	Receiver overhead	Contention modeling
Hockney	No	No	α	$1/\beta$			No
LogP	No	No	L	g	o	o	No
LogGP	No	No	L	$g+G$	o	o	No
LoPC	No	No	L	g	o	o	Yes
LoGPC	No	No	L	$g+G$	o	o	Yes
pLogP	Yes	Yes	L	g	o_s	o_r	No
Model of Cappello et al. (2005)	Yes	Yes	λ	Included	π	π	No
Model of Steffenel (2006)	Yes	Yes	L	g	o_s	o_r	No
Model of Lastovetsky et al. (2006)	Yes	No	T_{net}	Included	L_i	L_j	No

performances, etc.), it is very difficult to manage such platform towards a high performance computing. One way to answer this problem and to minimize the inherent heterogeneity is to subdivide the network in homogeneous subnets (or logical clusters), as described below. At the end of this phase, we will get a set of logical clusters of homogeneous nodes and accurate interconnection network, which will be used to adaptively modeling communications inside each cluster during the second phase of the framework.

3.3.1 Network Performance Measurements

The framework starts by collecting available information from the target execution environment to be used in the step of clustering (see next section). There exist many tools for network monitoring, such as NWS. These tools permit to determine many useful parameters of the target parallel system like the current network status, the communication latency, the speeds of the processors, the CPU load, the available memory, etc. For instance, the communication latency and throughput permit to identify groups of machines with similar communication parameters.

3.3.2 Clustering

One reason to construct logical clusters is that machines may behave differently, and the easiest way to optimize communications is to group machines with similar performances (Steffenel and Mounié (2004)). In order to classify nodes in logical clusters we can use a clustering algorithm. Several strategies can be used to group machines, from simple metrics such as presented by Lowekamp and Beguelin (1996) or using more elaborate techniques (Dubois et al. (2007), for example). In this paper we consider a distance matrix constructed with the help of NWS, which further is evaluated according to the latencies between links in order to group nodes for which their incident edges respect a latency bound (by default 20%) inside that subnet.

3.4 Adaptive Modeling

Because each network in a grid environment presents different performance behaviors, we follow an adaptive strategy that first organizes the platform in separated homogeneous hierarchical clusters, finding the best approach for each cluster and then composing the inter-cluster communication schedule. Indeed, we modeled and implemented several models from the literature, which perform differently according to the network environment. By selecting the best adapted communication to each different cluster in our grid, we contribute to a poly-model modeling of communications in a grid environment. Any necessary characteristics are measured during the first phase corresponding to the network partitioning. We recall that the model selection is made in terms of information which is interesting to the problem, such as the size of data to communicate, the type of interconnection network, the number of nodes, the communication algorithms, etc.

It is also important to take into account the cost involved on the acquisition of model parameters. For instance, both LogP and LogGP models rely on a reduced number of measurements (they extrapolate the cost per byte from a few measurements), while pLogP requires several measurements to cover a large range of message sizes. Hence, while the later model is most expensive, it can be more precise when the communication cost does not varies linearly with the message size (a typical case with MPI, whose transmission policies depend on the message size).

For instance, Figures 3 and 4 present the performance predictions from LogP, LogGP and pLogP on two different networks, namely Gigabit Ethernet and Myrinet. From these figures, it is clear that the most accurate predictions are provided by pLogP performance model, as already observed by Pjesivac-Grbovic et al. (2005). Nevertheless, the other models are good candidates for specific networks/message sizes. LogP is quite interesting when modeling the performance in a Gigabit Ethernet network with small messages. LogGP model is also a good candidate in the case of large messages, with both Gigabit Ethernet and Myrinet networks. Because LogP and LogGP parameters can be obtained with a relatively reduced cost (when comparing with pLogP), they represent an interesting alternative when the application messages sizes are known in advance.

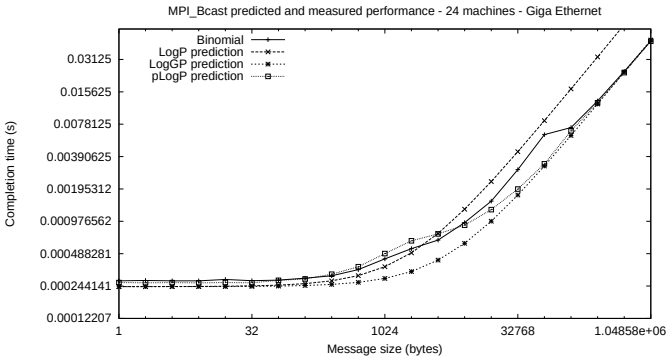


Figure 3: Performance predictions on a Gigabit Ethernet network.

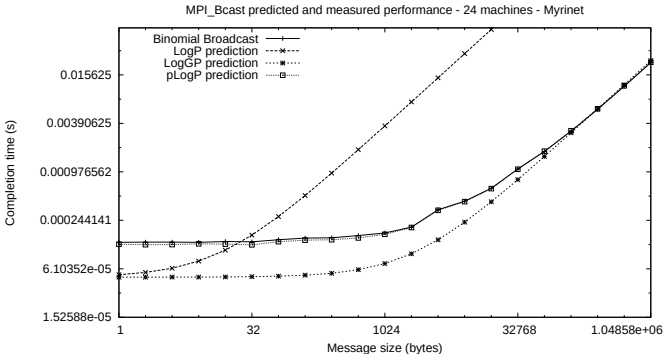


Figure 4: Performance predictions on a Myrinet network.

4 CASE STUDY: COLLECTIVE COMMUNICATIONS

In order to ensure a high-performance computing on actual execution supports, we should not neglect the impact of communications during the initial and final data redistribution phases of parallel applications. Similarly, problems that require intermediate data exchange among processes must take into account also the communication overhead that may eventually impact performances of algorithms.

Therefore, in this section we focus on communications in heterogeneous networks. We examine the case of collective communications, an important class of communication operations that are essential for data redistribution. Because collective communications are strongly influenced by the network heterogeneity, we rely on adaptive performance models to select the fastest communication algorithms in a given environment. Additionally, we inspect different performance modeling alternatives in order to find the model that best fits the effective communication behavior (Pjesivac-Grbovic et al. (2005)).

Collective communication operations encompass a wide range of possible algorithms. The optimal implementation of such operations for a given parallel system depends on many factors, including for example, physical topology of the system, number of involved processes, message sizes, and the location of the root node (Steffenel and Mounié (2006); Pjesivac-Grbovic et al. (2005)). Figure 5 shows how to implement a collective communication operation using our framework. Indeed, the adaptive algorithm selection is based on two types of models: the analytical and experimental ones. Experimental techniques use information derived from previous operation executions to optimize processing for future problem instances, a service similar to an optimization cache. The analytical models will be useful to validate actual versus predicted performances.

Through the analysis of the inter-cluster communication performances and the intra-cluster performance prediction we are able to define a communication schedule that minimizes the overall execution time. Once again, we implement different schedule policies, which are selected according to their estimated completion time. The framework allows, indeed, to implement scheduling heuristics that act on different communication levels, be it at inter-cluster level (mostly appropriate to collective operations like broadcast and reduce) or at node-to-node level (for operations such as the *all-to-all*).

Therefore, we apply our adaptive framework on collective communication operations by determining the best combination model-algorithm depending on the problem and architecture parameters. We refer here by algorithm a possible method to resolve the operation, as for example Pipeline, Binomial, Binary and Linear for Broadcast. At this level, our framework automatically associates the more appropriate model to the best algorithm among multiple model and algorithmic options.

4.1 Description of the Execution Platform

We have considered a Grid platform belonging to the project Grid'5000 to achieve our experiments. The architecture is initially composed of four clusters, distributed over sites in France: C1 (20 machines ORSAY), C2 (19 machines GRENOBLE), C3 (19 machines SOPHIA-ANTIPOLIS) and C4 (20 machines TOULOUSE) with two levels of hierarchy. Figure 6(a) shows the initial organization of the clusters.

4.2 Network Partitioning

The first phase of the framework leads to a new organization of the machines (see Figure 6(b)). The cluster C2 will be partitioned into three sub-clusters, C21 (11 machines), C22 (1 machine) and C23 (7 machines), according to the latencies presented in Table 2 of the links between machines. Once the clustering phase is done, we have six logical clusters with homogeneous resources.

4.3 Model Selection

We have considered the Broadcast operation. With Broadcast, a single process, called *root,* sends the same message of size m to all other $(P-1)$ processes. Classical implementations of the Broadcast operation rely on fixed shapes such as Linear (Flat Tree) for small number of nodes, and Binomial Trees for $P > 3$.

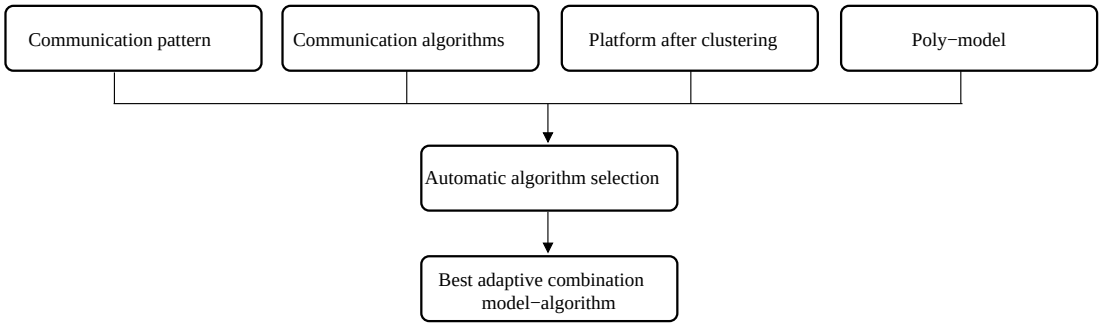
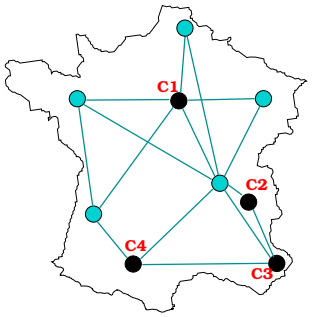


Figure 5: Execution of a collective communication operation using the framework.

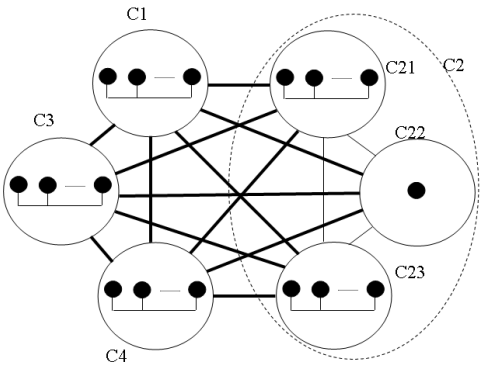
Table 2: Latency (microsec) intra and inter-clusters.

	C1	C21	C22	C23	C3	C4
	20 x Orsay	11 x Grenoble	1 x Grenoble	7 x Grenoble	20 x Toulouse	19 x Sophia
C1	48.39	6577.49	6592.51	6586.49	5211.94	8602.73
C21	6577.49	35.52	59.96	59.96	5387.48	2736.56
C22	6592.51	59.96	0*	79.51	5405.78	2745.98
C23	6586.49	59.96	79.51	60.08	5393.98	2740.26
C3	5211.94	5387.48	5405.78	5393.98	26.94	3630.51
C4	8602.73	2736.56	2745.98	2740.26	3630.51	35.04

* this "logical cluster" has only one machine.



(a)



(b)

Figure 6: (a) Grid'5000 sites and (b) Platform after partitioning

In our work we developed the communication models for some current techniques and their "flavors", which are presented in Table 3 in terms of pLogP parameters (which can be easily adapted to other models such as LogP and LogGP). Hence, we choose to compare in this paper four of the most known techniques, namely the Linear, the Binary, the Binomial and the Pipeline Broadcasts. Indeed, we proceed the MPI_Bcast optimization in two hierarchical levels, as described below.

4.3.1 First hierarchical level

First, we deal with local-area communications, where a cluster "coordinator" will be charged to broadcast locally the message to all processes composing the cluster. Later, we integrate these clusters through the generation of efficient inter-cluster communication schedules. Therefore, at the first hierarchical level, we proceed by selecting the most appropriate performance model and broadcast algorithm for each cluster according to two steps:

Poly-adapt-1. For a given problem (message size and number of processes) and for each implementation algorithm, we select the most accurate performance model such that its predictions correspond to our base of experiments. As discussed in Section 3, the selection of the most accurate performance model can also consider other factors such as parameters measurement cost and network intrusiveness;

Poly-adapt-2. From the performance models selected in the previous step, we select the most efficient implementation algorithm (i.e., the algorithm that terminates earlier).

Table 3: Communication models for the *Broadcast* operation

Strategy	**Communication Model**
Linear (Flat Tree)	$L + (P-1) \times g(m)$
Pipeline (Segmented Chain)	$(P-1) \times (g(s)+L) + (g(s) \times (k-1))$
Binary Tree	$\leq \lceil log_2 P \rceil \times (2 \times g(m) + L)$
Binomial Tree	$\lceil log_2 P \rceil \times L + \lfloor log_2 P \rfloor \times g(m)$

As a result, we select the algorithm that minimizes the completion time of the operation using the most accurate performance model for each problem instance (as we can observe in Figure 7). Table 4 summarizes these choices when considering two different message sizes, 8kB and 512kB.

4.3.2 Second hierarchical level

Once Poly-adapt-2 selected the best algorithm for each cluster, we must determine an efficient inter-cluster communication scheduling. Using inter-cluster communication parameters, we can construct an optimized broadcast tree between clusters using scheduling heuristics, an approach that provides better performances on grid environments than traditional grid-unaware algorithms found on most MPI distributions (Steffenel and Mounié (2006)). Different heuristics can be compared at this step, as different optimization parameters or objectives can lead to different inter-cluster communication strategies. Traditional scheduling heuristics such as *First Edge First* - FEF or *Early Completion Edge First* - ECEF heuristic, proposed by Bhat et al. (1998) can be used, as well as heuristics that consider special parameters such as the communication time inside each cluster (the ECEF-LAt heuristic Steffenel and Mounié (2006)).

In the case of this run, the adaptation mechanism selected the *Early Completion Edge First* - ECEF heuristic, proposed by Bhat et al. (1998). This heuristic proceeds by selecting a pair *sender-receiver* where the sender is available and the choice of the sender-receiver pair depends on the earliest possible moment when this transmission may effectively be finished. Therefore, we use a *Ready Time* (RT_i) parameter, evaluated conjointly with the transmission time between the processes, such that the pair i, j minimizes the expression:

$$t = RT_i + g_{i,j}(m) + L_{i,j}$$

4.4 Results Analysis

In this section we detail the execution steps of the Broadcast operation on the Grid'5000 platform when using our adaptation approach. As stated in the previous sections, our policy of performance evaluation consists in a multi-layered analysis of the different algorithms and elements of the network. Indeed, the first step - ***Poly-adapt-1*** - consists on choosing, for each network component (in this case, each cluster that composes the grid platform), the most accurate performance model for a fixed algorithm. This helps choosing a performance model that is able to predict the communication cost inside each cluster, which will be used in the second step of adaptation.

The second adaptation level (***Poly-adapt-2***) compares different algorithms for a given operation. With the help of the performance model chosen in the first step, we are able to predict the performance of the different algorithms and select the most adapted to our needs (in this case, the algorithm that performs faster inside a cluster). Once selected the best algorithm for each cluster, the second hierarchical adaptation level defines a communication schedule that minimizes the overall communication time.

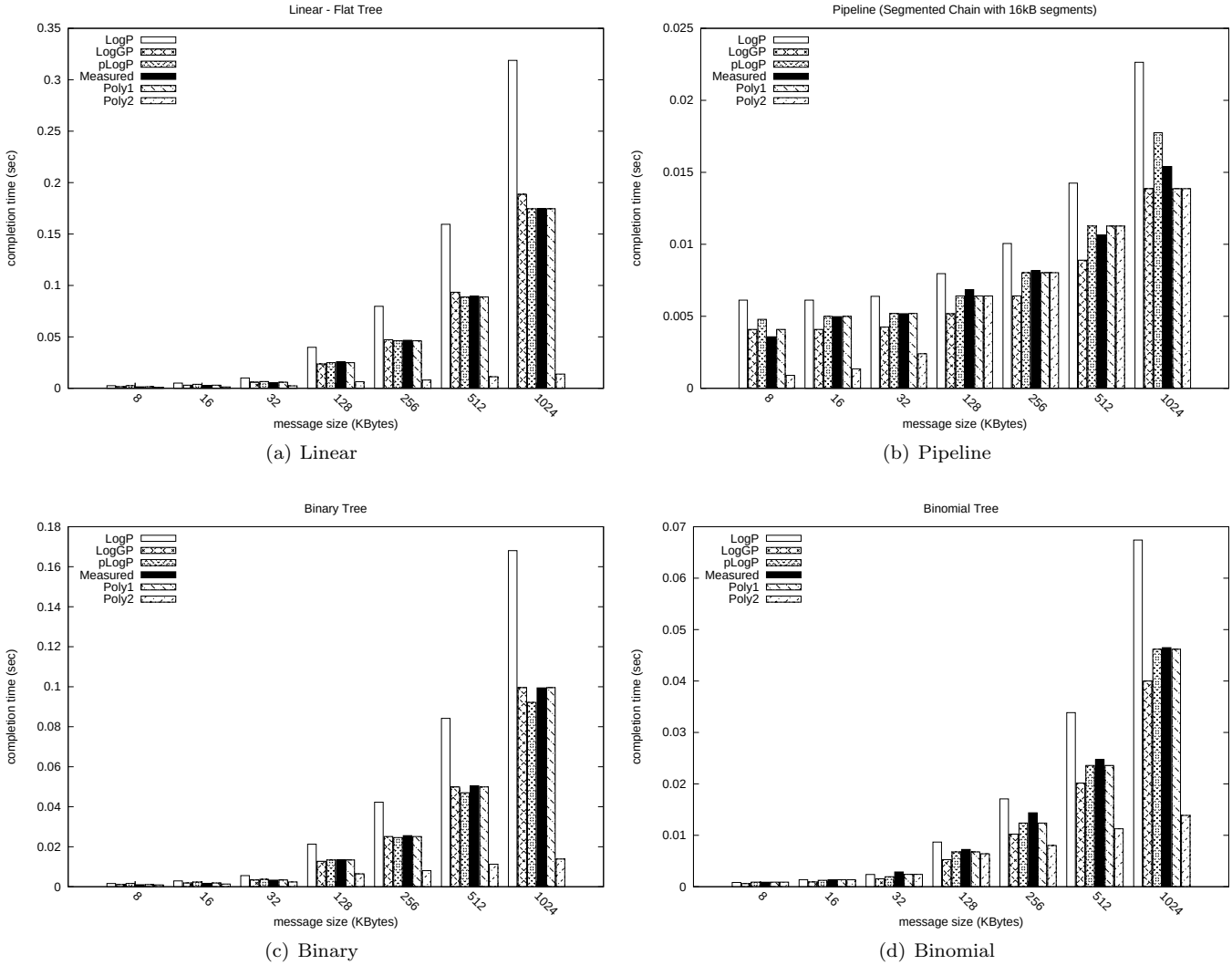
To better understand the multi-levels adaptation mechanism, we present in Figure 7 an extract of the adaptation steps representing the ORSAY cluster. Please note that the same mechanism is applied simultaneously on all clusters, as the result of these steps will be used on the second hierarchical adaptation level. Indeed, Figure 7 shows both Poly-adapt-1 and Poly-adapt-2 steps applied on different algorithms and message sizes. Let's take for example Figure 7(a), which represents the Flat Tree broadcast algorithm. Predictions from different performance models (LogP, LogGP, pLogP) are compared with a measured sample; Poly-adapt-1 retains the model that better fits the sample measures. As the same procedure is applied on other algorithms, Poly-adapt-2 is able to select the fastest algorithm (that's why Poly2 in Figure 7(a) is so different from Poly1, as it corresponds to the performance of the Pipeline algorithm).

When Poly-adapt-1 and Poly-adapt-2 were applied to the ensemble of the clusters, we can proceed with the second hierarchical adaption level. Here, we compare different scheduling heuristics aiming the optimization of inter-cluster communications. From the network connectivity data and predictions from Poly-adapt-2 we are able to determine the heuristic that allows the broadcast to finish earlier, the ECEF heuristic in the case of our example. To illustrate the performance improvement we can obtain with this multi-level adaptation approach, we depict in Figure 8 the measured completion time of a broadcast among 88 machines over four clusters from Grid'5000. In this figure, we compare the performance of the standard algorithm used on MPI (a binomial tree) against different grid-aware broadcast schedules used on the second hierarchical adaptation level.

Through our adaptation approach, we strongly improve the performance of broadcast communications in this grid platform, going two to four times faster than the standard MPI implementation. Additionnally, the dynamic adapta-

Table 4: Summary of the adaptive execution scheme for m=8kB and m=512kB.

Cluster		C1	C2*	C3	C4	inter-clusters
m=8k	Selected model	LogP	LogP	LogP	LogP	Hierarchical (Steffenel and Mounié (2006))
	Selected algorithm	binomial	binomial	binomial	binomial	ECEF
m=512k	Selected model	LogGP	pLogP	pLogP	pLogP	Hierarchical (Steffenel and Mounié (2006))
	Selected algorithm	pipeline	pipeline	pipeline	pipeline	ECEF



(a) Linear (b) Pipeline

(c) Binary (d) Binomial

Figure 7: Completion time of Broadcast in the cluster ORSAY: (a) Linear, (b) Pipeline, (c) Binary and (d) Binomial

tion mechanism identifies which communication schedule effectively perform well on a given platform. We observe indeed that the mechanism prevents the use of the Flat-tree heuristic (used on MagPIe - Kielmann et al. (2001)), as it does not performs well for messages large than 64Kb in our grid platform.

Although in this example we explore only two communication levels (inter and intra-cluster), additional adaptation levels could be considered if need. Indeed, it becomes interesting today to extend the optimization to a communication granularity at the level of applications running on SMP or multicore processors.

5 CONCLUSIONS AND FUTURE WORKS

We have presented in this paper a new adaptive framework for dealing with performance modeling and the design of efficient parallel algorithms on grid computing platforms. The developed methodology, mixing communication models and powerful adaptive approaches in a self adapting fashion, proceeds in two-level adaptation to automatically associate the more appropriate model to the best algorithm among several models and algorithmic options for each part of a large heterogeneous parallel computing platform being used. The best identified combination model-

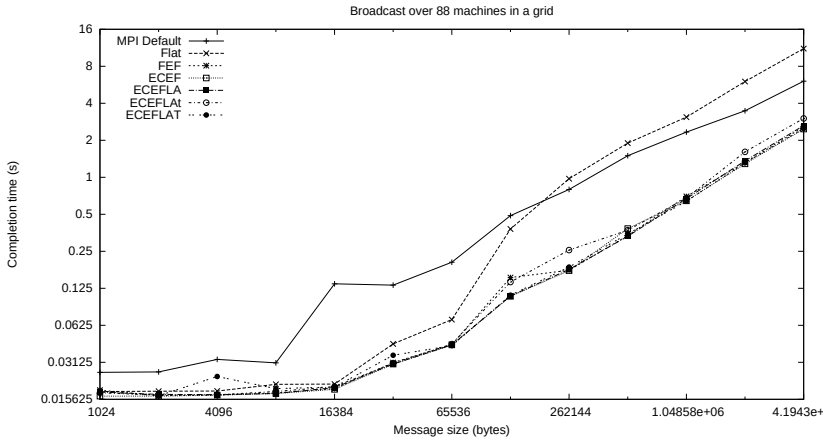


Figure 8: Completion time of the overall execution of a Broadcast on the global hierarchy platform.

algorithm and the determined execution scheme permit to minimize the overall execution time of a target problem. To illustrate the interest of this approach, we demonstrate how an important collective communication pattern, namely the broadcast operation, can be improved to better adapt to a grid environment, reaching significant performance improvements through a multilevel adaptive approach. Although a simple example, the case of the broadcast operation demonstrates the advantages of associating the accuracy of different performance models and the flexibility of a multi-algorithm adaptive technique.

Currently we are working on the extension of our adaptive methodology, applying the same principles on a well-known numerical problem, namely computing the product of two (large) dense square matrices. The parallelization of the matrix multiplication problem was widely studied in the literature. Various optimized versions of this problem have been implemented in libraries on all existing (homogeneous or heterogeneous) parallel systems. We may particularly refer to works presented in Beaumont et al. (2001); Desprez and Suter (2004); Hunold et al. (2004); Ohtaki et al. (2004), where various methods have been applied, such as standard, fast, mixed, etc. Nevertheless, only few parallel adaptive implementations have been developed (Hong and Prasanna (2002); Nasri et al. (2007); Thomas et al. (2005)). To the best of our knowledge, no original work has been devoted to implement adaptive algorithms for matrix multiplication on heterogeneous hierarchical clusters where both computing resources and interconnection links are heterogeneous. It is our aim, therefore, to show the importance of adaptive algorithms in order to efficiently perform such operations in arbitrary environments.

As future prospects, we intend to perform experiments on other numerical problems whose performances depend on both the distributions of the processes and their related communications. We also plan to integrate other existing adaptive approaches to our framework to benefit well from the powerful of these techniques.

ACKNOWLEDGEMENTS

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see *https://www.grid5000.fr*).

REFERENCES

Alexandrov, A., Ionescu, M., Schauser, K. and Scheiman, C. (1995) 'LogGP: Incorporating long messages into the LogP model - one step closer towards a realistic model for parallel computation', *Proc. of the 7th Annual Symposium on Parallel Algorithms and Architecture (SPAA'95)*.

Beaumont, O., Boudet, V., Rastello, F. and Robert, Y. (2001) 'Matrix multiplication on heterogeneous platforms', *IEEE Trans. Parallel Distributed Systems*, Vol. 12 No. 10, pp. 1033–1051.

Bhat, P., Prasanna, V.K. and Raghavendra, C. (1998) 'Adaptive communication algorithms for distributed heterogeneous systems', *Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC 1998)*.

Cappello, F., Fraigniaud, P., Mans, B. and Rosenberg, A. (2005) 'An algorithmic model for heterogeneous hyperclusters: Rationale and experience', *International Journal of Foundations of Computer Science*, Vol. 16 No. 2, pp. 195–215.

Chen, Z., Dongarra, J., Luszczek, P. and Roche, K. (2003) 'Self-adapting software for numerical linear algebra and LAPACK for clusters', *Parallel Computing*, Vol. 29 No. 11-12, pp. 1723–1743.

Cuenca, J., Giménez, D.,González, J.,Dongarra, J. and Roche, K. (2003) 'Automatic optimisation of parallel linear algebra routines in systems with variable load', *11th Euromicro Workshop on Parallel, Distributed and Network-Based Processing (PDP 2003)*, Genova, Italy.

Culler, D., Karp, R., Patterson, D., Sahay, A., Schauser, K.E., Santos, E., Subramonian, R. and von Eicken, T. (1996) 'LogP - a practical model of parallel computing', *Communication of the ACM*, Vol. 39 No. 11, pp. 78–85.

Desprez, F. and Suter, F. (2004) 'Impact of Mixed-Parallelism on Parallel Implementations of Strassen and Winograd Matrix Multiplication Algorithms', *Concurrency and Computation: practice and experience*, Vol. 16, No. 8, pp. 771–797.

Dubois, L.E., Legrand, A., Quinson, M. and Vivien, F. (2007) 'A first step towards automatically building network representations', *Euro-Par 2007*, Rennes, France, pp. 160–169.

Faik, J., Teresco, J.D., Devine, K.D., Flaherty, J.E. and Gervasio, L.G. (2005) 'A model for resource-aware load balancing on heterogeneous clusters', *Tech. Rep. CS-05-01*, Williams College Department of Computer Science.

Frank, M.I., Agarwal, M. and Vernon, M.K. (1997) 'LoPC: Modeling contention in parallel algorithms', *Proc. of 6th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, Las Vegas, Nevada.

Frigo, M. and Johnson, S.G. (1998) 'FFTW: an adaptive software architecture for the fast fourier transform', *Proceedings of ICASSP*, Vol. 3.

Hartmann, O., Kuhnemann, M., Rauber, T. and Runger, G. (2006) 'Adaptive selection of communication methods to optimize collective MPI operations', *Proc. of the 12th Workshop on Compilers for Parallel Computers (CPC'06)*, La Coruna, Spain.

Hockney, R. (1994), 'The communication challenge for MPP: Intel Paragon and Meiko CS-2', *Parallel Computing*, Vol. 20, pp. 389–398.

Hong, B. and Prasanna, V.K. (2002) 'Adaptive matrix multiplication in heterogeneous environments', *Proc. of ICPADS*.

Hunold, S., Rauber, T. and Runger, G. (2004) 'Multilevel hierarchical matrix multiplication on clusters', *ICS '04: Proceedings of the 18th Annual International Conference on Supercomputing*, ACM Press, New York, NY, USA.

Kielmann, T., Bal, H., Gorlatch, S., Verstoep, K. and Hofman, R. (2001) 'Network performance-aware collective communication for clustered wide area systems', *Parallel Computing*, Vol. 27, No. 11, pp. 1431–1456.

Lastovetsky, A.L., Mkwawa, I.-H. and O'Flynn, M. (2006) 'An accurate communication model of a heterogeneous cluster based on a switch-enabled ethernet network', *12th International Conference on Parallel and Distributed Systems (ICPADS 2006)*.

Lowekamp B. and Beguelin, A. (1996) 'ECO: Efficient collective operations for communication on heterogeneous networks', *Proc. of the 10th International Parallel Processing Symposium*.

McCracken, M.O., Snavely, A. and Malony, A. (2003) 'Performance modeling for dynamic algorithm selection', *Intl. Conference on Computational Science*, LNCS vol. 2660, Springer.

Moritz, C.A. and Frank, M.I. (2001) 'LoGPC: Modeling network contention in message-passing programs', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, No. 4, pp. 404–415.

Nasri, W., Trystram, D. and Achour, S. (2007) 'Adaptive algorithms for the parallelization of the dense matrix multiplication on clusters', *Internat. J. of Computational Science and Engineering, Special Issue on best selected papers of PDSEC'04*, to appear.

Ohtaki, Y., Takahashi, D., Boku, T. and Sato, M. (2004) 'Parallel implementation of strassen's matrix multiplication algorithm for heterogeneous clusters', *IPDPS*.

Pjesivac-Grbovic, J., Angskun, T., Bosilca, G., Fagg, G.E., Gabriel, E. and Dongarra, J.J. (2005) 'Performance analysis of MPI collective operations' *Proc. of the Wokshop on Performance Modeling, Evaluation and Optimisation for Parallel and Distributed Systems (PMEO), in IPDPS 2005*.

Pjesivac-Grbovic, J., Bosilca, G., Fagg, G.E., Angskun, T. and Dongarra, J.J. (2007) 'Decision trees and MPI collective algorithm selection problem', *Proceedings of Euro-Par 2007*, LNCS Vol. 4641, pp. 107–117, Rennes, France.

Steffenel, L.A. and Mounié, G. (2004) 'Identifying logical homogeneous clusters for efficient wide-area communication'. *Proc. of the Euro PVM/MPI 2004*, LNCS Vol. 3241, pp. 319–326, Budapest, Hungary.

Steffenel, L.A. and Mounié, G. (2006) 'Scheduling heuristics for efficient broadcast operations on grid environments'.,*Proc. of the Performance Modeling, Evaluation and Optimization of Parallel and Distributed Systems Workshop - PMEO'06 (associated to IPDPS'06)*, IEEE Computer Society, Rhodes Island, Greece.

Steffenel, L.A. (2006) 'Modeling network contention effects on alltoall operations', *Proc. of the IEEE Conference on Cluster Computing (CLUSTER 2006)*, Barcelona, Spain.

Teresco, J.D., Faik, J. and Flaherty, J.E. (2005) 'Resource-aware scientific computation on a heterogeneous cluster', *Computing in Science and Engineering*, Vol. 7, No. 2, pp. 40–50.

Thomas, N., Tanase, G., Tkachyshyn, O., Perdue, J., Amato, N.M. and Rauchwerger, L. (2005) 'A framework for adaptive algorithm selection in STAPL', *Proc. ACM SIGPLAN Symp. Prin. Prac. Par. Prog. (PPOPP)*.

Vadhiyar, S., Fagg, G. and Dongarra, J. (2004) 'Towards an accurate model for collective communications', *International Journal of High Performance Computing Applications*, Vol. 8, No. 1, pp. 159–167.

Whaley, R.C., Petitet, A. and Dongarra, J.J. (2001) 'Automated empirical optimization of software and the ATLAS project', *Parallel Computing*, Vol. 27, No. 1-2, pp. 3–35.

Wolski, R., Spring, N. and Peterson, C. (1997) 'Implementing a performance forecasting system for metacomputing: The network weather service', *Proc. of the Supercomputing*.