INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team AlGorille

Algorithms for the Grid

Nancy - Grand Est

THEME NUM

Table of contents

1. Team

Head of project-team
Jens Gustedt [research director, INRIA, HdR]

Administrative assistant
Céline Simon [INRIA]

Permanent researcher
Emmanuel Jeannot [Chargé de Recherche, INRIA, HdR]
Stéphane Genaud [Chargé de Recherche, INRIA, on leave from Univ. Strasbourg, since 1/10/2007]

Faculty members
Sylvain Contassot-Vivier [Professor, UHP since 01/09/2007, HdR]
Martin Quinson [Maître de Conférences, UHP/ÉSIAL]
Frédéric Suter [Maître de Conférences, UHP]
Luiz Angelo Steffenel [Assistant, Nancy 2 until 31/08/2007]

Technical staff
Xavier Delaruelle [Ingénieur associé, INRIA until 31/10/2007]
Abdelmalek Cherier [Ingénieur associé, INRIA]

Ph. D. students
Louis-Claude Canon [Allocataire de recherche MENESR since 1/10/07]
Stefan Canzar [joint regional/CNRS BDI until 30/9/07, joint thesis with UdS, Saarbrücken, Germany]
Pierre-Nicolas Clauss [Allocataire de recherche MENESR since 01/10/06]
Soumeya Hernane [teaching assistant UST Oran, Algeria, since 01/10/07]
Constantinos Makassikis [Supélec, since 01/02/07]
Tchimou N'Takpé [joint regional/Côte d'Ivoire government grant since 01/10/05]

Internships
Louis-Claude Canon [M2R student between 01/03/2007 and 01/08/2007]
Eloi Du Bois [Undergraduate student between 01/07/2007 and 31/08/2007]
Olivier Dubuisson [CNAM Internship between 01/01/2007 and 31/12/2007]
Marc-Eduard Frincu [INRIA Internship, Univ Timisuara, Roumania, between 01/10/2007 and 31/12/2007]

External collaborator
Stéphane Vialle [professor, Supélec Metz Campus, HdR]

2. Overall Objectives

2.1. Overall Objectives

Keywords: *Grid computing*, *algorithms*, *data redistribution*, *data distribution*, *experiments*, *parallel and distributed computing*, *scheduling*.

The possible access to distributed computing resources over the Internet allows a new type of application that use the power of the machines and the network. The transparent and efficient access to these distributed resources that form *the Grid* is one of the major challenges of information technology. It needs the implementation of specific techniques and algorithms to make computers communicate with each other, let applications work together, allocate resources and improve the quality of service and the security of the transactions.

Challenge: We tackle several problems related to the first of the major challenges that INRIA has identified in its strategic plan:

"Design and master the future network infrastructures and communication services platforms."

Originality: Our approach emphasizes on *algorithmic aspects* of grid computing, in particular it addresses the problems of organizing the computation *efficiently*, be it on the side of a service provider, be it within the application program of a customer.

Research themes:

- Structuring of applications for scalability: modeling of size, locality and granularity of computation and data.
- Transparent resource management: sequential and parallel task scheduling; migration of computations; data exchange; distribution and redistribution of data.
- Experimental validation: reproducibility, extendibility and applicability of simulations, emulations and *in situ* experiments.

Methods: Our methodology is based upon three points (1.) modeling, (2.) design and (3.) engineering of algorithms. These three points interact strongly to form a feedback loop.

1. With models we obtain an abstraction of the physical, technical or social reality.
2. This abstraction allows us to design techniques for the resolution of specific problems.
3. These techniques are implemented to validate the models with experiments and by applying them to real world problems.

3. Scientific Foundations

3.1. Structuring of Applications for Scalability

Keywords: *message passing*, *models for parallel and distributed computing*, *performance evaluation*, *shared memory*.

Participants: Sylvain Contassot-Vivier, Jens Gustedt, Frédéric Suter, Stéphane Vialle.

Our approach is based on a "good" separation of the different problem levels that we encounter with Grid problems. Simultaneously, this has to ensure a good data locality (a computation will use data that are "close") and a good granularity (the computation is divided into non preemptive tasks of reasonable size). For problems for which there is no natural data parallelism or control parallelism such a division (into data and tasks) is mandatory when tackling the issues related to spatial and temporal distances as we encounter them in the Grid.

Several parallel models offering simplified frameworks that ease the design and the implementation of algorithms have been proposed. The best known of these provide a modeling that is called "*fined grained*", *i.e.,*at the instruction level. Their lack of realism with respect to the existing parallel architectures and their inability to predict the behavior of implementations, has triggered the development of new models that allow a switch to a *coarse grained* paradigm. In the framework of parallel and distributed (but homogeneous) computing, they started with the fundamental work of Valiant [44]. Their common characteristics are:

- to maximally exploit the data that is located on a particular node by a local computation,
- to collect all requests for other nodes during the computation, and
- to only transmit these requests if the computation can't progress anymore.

The coarse grained models aim at being realistic with regard to two different aspects: algorithms and architectures. In fact, the coarseness of these models uses the common characteristic of today's parallel settings: the size of the input is orders of magnitude larger than the number of processors that are available. In contrast to the PRAM (Parallel Random Access Machine) model, the coarse grained models are able to integrate the cost of communications between different processors. This allows them to give realistic predictions about the overall execution time of a parallel program. As examples, we refer to BSP (Bulk Synchronous Parallel model) [44], LogGP (Latency overhead gap Procs) [37], CGM (Coarse Grained Multicomputer) [39] and *PRO* (Parallel Resource Optimal Model) [5].

The assumptions on the architecture are very similar: p homogeneous processors with local memory distributed on a point-to-point interconnection network. They also have similar models for program execution that are based on *supersteps*; an alternation of computation and communication phases. For the algorithmics, this takes the distribution of the data on the different processors into account. But, all the mentioned models do not allow the design of algorithms for the Grid since they all assume homogeneity, for the processors as well as for the interconnection network.

Our approach is algorithmic. We try to provide a modeling of a computation on grids that allows an easy design of algorithms and realistic performing implementations. Even if there are problems for which the existing sequential algorithms may be easily parallelized, an extension to other more complex problems such as computing on large discrete structures (*e.g.,*web graphs or social networks) is desirable. Such an extension will only be possible if we accept a paradigm change. We have to explicitly decompose data and tasks.

We are convinced that this new paradigm should:

1. be guided by the idea of **supersteps** (BSP). This is to enforce a concentration of the computation to the local data,
2. ensure an economic use of all available resources.

On the other hand, we have to be careful that the model (and the design of algorithms) remains simple. The number of supersteps and the minimization thereof should by themselves not be a goal. It has to be constrained by other more "*natural*" parameters coming from the architecture and the problem instance.

A first solution that uses (item 1) to combine these objectives for homogeneous environments has been given in [5] with PRO.

In a complementary approach we have addressed (item 2) to develop a simple interface that gives a consistent view of the data services that are exported to an application, see [7].

Starting from this model, we try to design high level algorithms for grids. They will be based upon an abstract view of the architecture and as far as possible be independent of the intermediate levels. They aim at being robust with regard to the different hardware constraints and should be sufficiently expressive. The applications for which our approach will be feasible are those that fulfill certain constraints:

- they need a lot of computing power,
- they need a lot of data that is distributed upon several resources, or,
- they need a lot of temporary storage exceeding the capacity of a single machine.

To become useful on grids, coarse grained models (and the algorithms designed for them) must first of all overcome a principle constraint: the assumption of homogeneity of the processors and connections. The long term goal should be arbitrarily mixed architectures but it would not be realistic to assume to be able to achieve this in one step.

3.2. Transparent Resource Management

Keywords: *approximating algorithms*, *data redistribution*, *parallel and distributed computing*, *scheduling*.

Participants: Stéphane Genaud, Emmanuel Jeannot, Luiz Angelo Steffenel, Frédéric Suter.

We think of the future Grid as of a medium to access resources. This access has to be as transparent as possible to a user of such a Grid and the management of these resources has not to be imposed to him/her, but entirely done by a "system", so called middleware. This middleware has to be able to manage all resources in a satisfactory way. Currently, numerous algorithmic problems hinder such an efficient resource management and thus the transparent use of the Grid.

By their nature, distributed applications use different types of resources; the most important being these of computing power and network connections. The management and optimization of those resources is essential for networking and computing on Grids. This optimization may be necessary at the level of the computation of the application, of the organization of the underlying interconnection network or for the organization of the messages between the different parts of the application. Managing these resources relates to a set of policies to optimize their use and allows an application to be executed under favorable circumstances.

Our approach consists of the tuning of techniques and algorithms for a transparent management of resources, be they data, computations, networks, ...This approach has to be clearly distinguished from others which are more focused on applications and middlewares. We aim at proposing new algorithms (or improve the existing ones) *for* the resource management in middlewares. Our objective is to provide these algorithms in libraries so that they may be easily integrated. For instance we will propose algorithms to efficiently transfer data (data compression, distribution or redistribution of data) or schedule sequential or parallel tasks.

The problems that we are aiming at solving are quite complex. Therefore they often translate into combinatorial or graph theoretical problems where the identification of an optimal solution is known to be hard. But, the classical measures of complexity (polynomial versus NP-hard) are not very satisfactory for really large problems: even if a problem has a polynomial solution it is often infeasible in reality whereas on the other hand NP-hard problems may allow a quite efficient resolution with results close to optimality.

Consequently it is mandatory to study approximation techniques where the objective is not to impose global optimality constraints but to relax them in favor of a compromise. Thereby we hope to find *good* solutions at a *reasonable* price. But, these can only be useful if we know how to analyze and evaluate them.

3.3. Experimental Validation

Keywords: *applicability*, *emulations*, *extendibility*, *in situ experiments*, *reproducibility*, *simulations*.

Participants: Abdelmalek Cherier, Xavier Delaruelle, Olivier Dubuisson, Emmanuel Jeannot, Martin Quinson.

An important issue for the research on complex systems such as grids is to validate the obtained results. This validation constitutes a scientific challenge by itself since we have to validate models, how well they fit to reality and the algorithms that we design inside these models. Whereas mathematical proofs establish soundness within such a context, the overall validation must be done by experiments. A successful experiment shows the validity of both the algorithm and the modeling at the same time. But, if the algorithm does not provide the expected result, this might be due to several factors: a faulty modeling, a weak design, or a bad implementation.

Experimental validation of grid systems is a particularly challenging issue. Such systems will be large, rapidly changing, shared and severely protected. Naive experiments on real platforms will usually not be reproducible, while the extensibility and applicability of simulations and emulations will be very difficult to achieve. These difficulties imply the study phases through modeling, algorithm design, implementation, tests and experiments. The test results will reveal precious for a subsequent modeling phase, complementing the process into a feedback loop.

In addition to this idea of validating the whole (modeling, design and implementation) in our research we are often restricted by a lack of knowledge: the systems that we want to describe might be too complex; some components or aspects might be unknown or the theoretical investigations might not yet be sufficiently advanced to allow for provable satisfactory solutions of problems. We think that an experimental validation is a valuable completion of theoretical results on protocol and algorithm behavior.

The focus of algorithmic research on the parallel systems which preceded grids follows to goals being solely upon performance. In addition to these, grids aim at enabling the resolution of problem instances larger than the ones previously tractable. The instability of the target platforms also implies that the algorithms must be robust and tolerant to faults and uncertainty of their environment.

These elements have strong implications on the way grid experiments should be done. To our opinion, such experiments should fulfill the following properties:

reproducibility: Experimental settings must be designed and described such that they are reproducible by others and must give the same result with the same input.

extensibility: A report on a scientific experiment concerning performance of an implementation on a particular system is only of marginal interest if it is simply descriptive and does not point beyond the particular setting in which it is performed. Therefore, the design of an experiment *must* allow for comparisons with other work, be it passed or future. A rigorous documentation and an exploitation of the full parameter range is necessary for the extensions to more and other processors, larger data sets, different architectures and alike. Several dimensions have to be taken into account: scalability, portability, prediction and realism.

applicability: Performance evaluation should not be a goal *in fine* but should result in concrete predictions of the behavior of programs in the real world. However, as the set of parameters and conditions is potentially infinite, a good experiment campaign must define realistic parameters for platforms, data sets, programs, applications, *etc.* and must allow for an easy calibration.

revisability: When an implementation does not perform as expected, it should be possible to identify the reasons, be they caused by the modeling, the algorithmic design, the particular implementation and/or the experimental environment. Methodologies that help to explain mispredictions and to indicate improvements have to be developed.

4. Application Domains

4.1. High Performance Computing

Participants: Pierre-Nicolas Clauss, Jens Gustedt, Frédéric Suter.

4.1.1. Models and Algorithms for Coarse Grained Computation

With this work we aim at extending the coarse grained modeling (and the resulting algorithms) to hierarchically composed machines such as clusters of clusters or clusters of multiprocessors.

To be usable in a Grid context this modeling has first of all to overcome a principal constraint of the existing models: the idea of an homogeneity of the processors and the interconnection network. Even if the long term goal is to target arbitrary architectures it would not be realistic to think to achieve this directly, but in different steps:

- Hierarchical but homogeneous architectures: these are composed of an homogeneous set of processors (or of the same computing power) interconnected with a non-uniform network or bus which is hierarchic (CC-Numa, clusters of SMPs).
- Hierarchical heterogeneous architectures: there is no established measurable notion of efficiency or speedup. Also most certainly not any arbitrary collection of processors will be useful for computation on the Grid. Our aim is to be able to give a set of concrete indications of how to construct an extensible Grid.

In parallel, we have to work upon the characterization of architecture-robust efficient algorithms, *i.e.,*algorithms that are independent, up to a certain degree, of low-level components or the underlying middleware.

The literature about fine grained parallel algorithms is quite exhaustive. It contains a lot of examples of algorithms that could be translated to our setting, and we will look for systematic descriptions of such a translation.

List ranking, tree contraction and graph coloring algorithms already have been designed following the coarse grained setting given by the model *PRO* [5].

To work in the direction of understanding what problems might be "*hard*" we tackled a problem that is known to be P-complete in the PRAM/NC framework, but for which not much had been known when only imposing the use of relatively few processors: the *lexicographic first maximal independent set* (LFMIS) problem [41].

We already are able to give a work optimal algorithm in case we have about $\log n$ processors and thus to prove that the NC classification is not necessarily appropriate for today's parallel environments which consist of few processors (up to some thousands) and large amount of data (up to some terabytes).

4.1.2. External Memory Computation

In the mid-nineties several authors [36], [38] developed a connection between two different types of computation models: BSP-like models of parallel computation and IO efficient external memory algorithms. Their main idea is to enforce data locality during the execution of a program by simulating a parallel computation of several processors on one single processor.

While such an approach is convincing on a theoretical level, its efficient and competitive implementation is quite challenging in practice. In particular, it needs software that induces as little computational overhead as possible by itself. Up to now, it seems that this has only been provided by software specialized in IO efficient implementations.

In fact, the stability of our library *parXXL* (formerly *SSCRAP*), see Section 5.1, also showed in its extension towards external memory computing [6]. *parXXL* has a consequent implementation of an abstraction between the *data* of a process execution and the memory of a processor. The programmer acts upon these on two different levels:

- with a sort of *handle* on some data array which is an abstract object that is common to all *parXXL* processes;
- with a map of its (local) part of that data into the address space of the *parXXL* processor, accessible as a conventional pointer.

Another add-on was the possibility to fix a maximal number of processors (*i.e.,*threads) that should be executed concurrently

4.1.3. Irregular Problems

Irregular data structures like sparse graphs and matrices are in wide use in scientific computing and discrete optimization. The importance and the variety of application domains are the main motivation for the study of efficient methods on such type of objects. The main approaches to obtain good results are parallel, distributed and out-of-core computation.

We follow several tracks to tackle irregular problems: automatic parallelization, design of coarse grained algorithms and the extension of these to external memory settings.

In particular we study the possible management of very large graphs, as they occur in reality. Here, the notion of "*networks*" appears twofold: on one side many of these graphs originate from networks that we use or encounter (Internet, Web, peer-to-peer, social networks) and on the other side the handling of these graphs has to take place in a distributed Grid environment. The principal techniques to handle these large graphs will be provided by the coarse grained models. With the PRO model [5] and the *parXXL* library we already provide tools to better design algorithms (and implement them afterwards) that are adapted to these irregular problems.

In addition we will be able to rely on certain structural properties of the relevant graphs (short diameter, small clustering coefficient, power laws). This will help to design data structures that will have good locality properties and algorithms that compute invariants of these graphs efficiently.

4.2. Evolution of Scheduling Policies and Network Protocols

Participants: Emmanuel Jeannot, Tchimou N'Takpé, Luiz Angelo Steffenel, Frédéric Suter.

4.2.1. Scheduling on the Grid

Recent developments in grid environment have focused on the need to efficiently schedule tasks onto distributed computational servers. The problem consists in deciding which compute resource should perform which task when, in a view to optimizing some quality metric.

Thus, environments based on the client-agent-server model such as NetSolve, Ninf or DIET are able to distribute client requests on a set of distributed servers. The performance of such environments greatly depends on the scheduling heuristic implemented. In these systems, a server executes each request as soon as it has been received: it never delays the start of the execution.

In order for a such a system to be efficient, the mapping function must choose a server that fulfills several criteria. First, the total execution time of the client application (*e.g.,*the makespan) has to be as short as possible. Second, each request of every client must be served as fast as possible. Finally, the resource utilization must be optimized. However, these objectives are often contradictory. Therefore it is required to design multi-criteria heuristics that guarantee a balance between these criteria.

An other characteristic of grid environments is their dynamicity and volatility. The availability of resources can change with time and they also can be shared with other users. Moreover, workloads submitted to a grid are subject to uncertainty in terms of duration, or of submission time. In order to cope with these different levels of unpredictability it is important to model this unpredictability and to design scheduling algorithms that use these models. In this case the metrics can be robustness (a schedule is said robust if it is able to absorb some uncertainty) or fault-tolerance (giving a schedule that is efficient in the case of failures).

4.2.2. Parallel Task Scheduling

The use of parallel computing for large and time-consuming scientific simulations has become mainstream. Two kinds of parallelism are typically exploited in scientific applications: *task parallelism* and *data parallelism*. In task parallelism the application is partitioned into a set of tasks organized in a Directed Acyclic Graph (DAG) in which nodes correspond to tasks and edges correspond to precedence and/or data communication constraints. In data parallelism an application exhibits parallelism typically at the level of loops. A way to expose and exploit increased parallelism, to in turn achieve higher scalability and performance, is to write parallel applications that use both task and data parallelism. This approach is termed *mixed parallelism* and allows several data-parallel tasks to be executed concurrently.

In the case of mixed-parallel applications, data parallelism adds a level of difficulty to the task-parallel scheduling problem. Indeed, the common assumption is that data-parallel tasks are moldable, *i.e.,*they can be executed on arbitrary numbers of processors, with more processors leading to smaller task execution times. This is typical of most mixed-parallel applications, and raises the question: how many processors should be allocated to each data-parallel task? There is thus an intriguing tension between running more concurrent data-parallel tasks with each fewer processors, or fewer concurrent data-parallel tasks with each more processors. Not surprisingly this scheduling problem is NP-complete. Consequently, several researchers have attempted to design scheduling heuristics for mixed-parallel applications. Most of them assume an execution on a homogeneous computing environment. While homogeneous platforms are relevant to many real-world scenarios, heterogeneous platforms are becoming increasingly common and powerful. Indeed many current computing platforms consist of multiple compute clusters aggregated within or across institutions. Mixed-parallel applications appear then ideally positioned to take advantage of such large-scale platforms. However, the clusters in these platforms are rarely identical. Because deployed by different institutions at different times, they typically consist of different numbers of different compute nodes (*e.g.,*there can be large slow clusters and small fast clusters).

We followed two approaches to schedule mixed-parallel applications on heterogeneous platforms. The first approach consists in adapting the aforementioned two-phase algorithms for mixed-parallel applications on homogeneous platforms and making them amenable to heterogeneous platforms [43]. The second approach consists in adapting list scheduling algorithms that were specifically designed for executing task-parallel applications on heterogeneous platforms and making them amenable to mixed-parallelism [33]. Both approaches

have merit and correspond to different categories of users. The heuristics of the former approach produce schedules that are a good balance between time and resource usage. Such schedules will satisfy the expectations of most users. Algorithms derived of the latter approach are more tunable and thus require a better knowledge of scheduling issues to the users.

Another aspect of current computing platforms that could not be neglected is that their resources are shared by many users and applications. Only a few scheduling heuristics address issues relevant to this characteristic such as ensuring a fair access to resources between application users, but not at the price of a tremendous increase of the average execution time or idle times on processors.

4.2.3. Data Redistribution and Collective Communication Between Clusters

During computations performed on clusters of machines it occurs that data has to be shifted from one cluster to an other. For instance, these two clusters may differ in the resources they offer (specific hardware, computing power, available software) and each cluster may be more adequate for a certain phase of the computation. Then the data have to be redistributed from the first cluster to the second one. Such a redistribution should use the capacities of the underlying network in an efficient way.

This problem of redistribution between clusters generalizes the redistribution problem inside a parallel machine, which already is highly non trivial.

Redistributing data between clusters has recently received considerable attention as it occurs in many application frameworks. Examples of such frameworks are distributed code-coupling, parallel task execution and persistence and redistribution for metacomputing.

The problem is easily modeled by a decomposition of a bipartite graph into matchings of a given size. However finding a minimal decomposition is NP-Hard and therefore it is required to look for heuristics or approximation algorithms.

This problem can be generalized to the case where during one communication phase every node of any cluster can potentially communicate to any other node of any other cluster (as in a total exchange or all-to-all). This problem occurs when executing an MPI program using several clusters. In this case, collective communications need to be optimized in order to take into account the topology and optimize the overall execution time.

4.2.4. Dynamic and Adaptive Compression of Network Streams

A commonly used technique to speed up transfer of large data over networks with restricted capacity during a distributed computation is data compression. But such an approach fails to be efficient if we switch to a high speed network, since here the time to compress and decompress the data dominates the transfer time. Then a programmer wanting to be efficient in both cases, would have to provide two different implementations of the network layer of his code, and a user of this program would have to determine which of the variants he/she has to run to be efficient in a particular case.

A solution of this problem is a adaptive service which offers the possibility to transfer data while compressing it. The compression level is dynamically changed according to the environment and the data. The adaptation process is required by the heterogeneous and dynamic nature of grids. For instance if the network is very fast, there may be no time to compress the data. But, if the visible bandwidth decreases (due to some congestion on the network), some time to compress the data may become available.

Then the problems to solve are to never degrade the performance, to offer a portable implementation, to deal with all kinds of networks and environments.

4.3. Providing Environments for Experiments

Participants: Sylvain Contassot-Vivier, Xavier Delaruelle, Olivier Dubuisson, Jens Gustedt, Emmanuel Jeannot, Martin Quinson, Stéphane Vialle.

4.3.1. Simulating Grid Platforms

We participate in the development of the SIMGrid tool. It enables the simulation of distributed applications in distributed computing environments for the specific purpose of developing and evaluating scheduling algorithms. Simulations not only allow repeatable results (what is hard to achieve on shared resources) but also make it possible to explore wide ranges of platform and application scenarios. SIMGrid implements realistic fluid network models that result in very fast yet precise simulations. SIMGrid also enables the simulation of distributed scheduling agents, which has become critical for current scheduling research in large-scale platforms. This tool is being used by several groups in the Grid Scheduling literature.

4.3.2. Emulating Heterogeneity

We have designed a tool called *Wrekavoc*. The goal of Wrekavoc is to define and control the heterogeneity of a given platform by degrading CPU, network or memory capabilities of each node composing this platform. Our current implementation of Wrekavoc has been tested on an homogeneous cluster. We have shown that configuring a set of nodes is very fast. Micro-benchmarks show that we are able to independently degrade CPU, bandwidth and latency to the desired values. Tests on algorithms of the literature (load balancing and matrix multiplication) confirm the previous results and show that Wrekavoc is a suitable tool for developing, studying and comparing algorithms in heterogeneous environments.

4.3.3. Grid'5000

The purpose of Grid'5000 is to serve as an experimental testbed for research in Grid Computing. In addition to theory, simulators and emulators, there is a strong need for large scale testbeds where real life experimental conditions hold. Grid'5000 aims at building a highly reconfigurable, controllable and monitorable experimental Grid platform gathering nine sites geographically distributed in France featuring a total of five thousands CPUs. We are in charge of one of these nine sites and we currently provide 574 cores to the community.

4.3.4. InterCell

Intercell aims at setting up a cluster (256 PCs) for interactive fine grain computation. It is granted by the Région Lorraine (CPER 2007), and managed at the Metz campus of Supélec.

The purpose is to allow easy fine grain parallel design, providing interactive tools for the visualization and the management of the execution (debug, step by step, *etc*). The parallelization effort is not visible to the user, since InterCell relies on the dedicated *parXXL* framework, see 5.1 below. Among the applications that will be tested is the interactive simulation of PDEs in physics, based on the Escapade project, see [21].

5. Software

5.1. parXXL

Participants: Pierre-Nicolas Clauss, Jens Gustedt, Stéphane Vialle.

parXXL is a library for large scale computation and communication that executes fine grained algorithms (computation and communication are of the same order of magnitude) on coarse grained architectures (clusters, grids, mainframes).

Historically *parXXL* is the result of a merge of two different projects, *ParCeL* (from Supélec) and *SSCRAP* (from INRIA), that stand for a consequent modeling and implementation of fine grained networks (*ParCeL*) and coarse grained algorithmics (*SSCRAP*) respectively.

This library takes the requirements of *PRO*, see Section 3.1, into account, *i.e.,*the design of algorithms in alternating computation and communication steps. It realizes an abstraction layer between the algorithm as it was designed and its realization on different architectures and different modes of communication. The current version of this library is available at http://parxxl.gforge.inria.fr/ and integrates:

- a layer for message passing with MPI,
- a layer for shared memory with POSIX threads, and,
- a layer for out-of-core management with file mapping (system call *mmap*).

All three different realizations of the communication layers are quite efficient. They let us execute programs that are otherwise unchanged within the three different contexts. Usually, they reach the performance of programs that are directly written for a given context. Generally they outperform programs that are executed in a different context than they were written for, such as MPI programs that are executed on a shared memory mainframe, or such as multi-threaded programs that are executed on a distributed shared memory machine.

5.2. AdOC

Participant: Emmanuel Jeannot.

The AdOC *(Adaptive Online Compression) library implements the* AdOC *algorithm for dynamic adaptive compression of network streams.*

AdOC is written in C and uses the standard library *zlib* for the compression part. It is realized as an additional layer above TCP and offers a service of adaptive compression for the transmission of program buffers or files. Compression is only used if it doesn't generate an additional cost, typically if the network is slow or the sending/receiving processors are not too charged. It integrates overlap techniques between compression and communication as well as mechanisms that avoid superfluous copy operations. The send and receive functions have exactly the same semantics as the system calls `read` and `write` so the integration of AdOC into existing libraries and application software is straightforward. Moreover, AdOC is thread-safe.

5.3. Wrekavoc

Participants: Olivier Dubuisson, Jens Gustedt, Emmanuel Jeannot.

Wrekavoc addresses the problem of controlling the heterogeneity of a cluster. Our objective is to have a configurable environment that allows for reproducible experiments on large sets of configurations using real applications with no emulation of the code. Given an homogeneous cluster Wrekavoc degrades the performance of nodes and network links independently in order to build a new heterogeneous cluster. Then, any application can be run on this new cluster without modifications.

Wrekavoc is implemented using the client-server model. A server, with administrator privilege, is deployed on each node one wants to configure. The client reads a configuration file and sends orders to each node in the configuration. The client can also order the nodes to recover the original state.

CPU Degradation. We have implemented several methods for degrading CPU performance. The first approach consists in managing the frequency of the CPU through the kernel CPU-Freq interface. We propose two other solutions in case CPU-Freq is not available. One is based on CPU burning. A program that runs under real-time scheduling policy burns a constant portion of the CPU, whatever the number of processes currently running. The other is based on user-level process scheduling called CPU-lim. A CPU limiter is a program that supervises processes of a given user. On Linux, using the `/proc` pseudo-filesystem, it suspends the processes when they have used more than the required fraction of the CPU.

Network Limitation. Limiting latency and bandwidth is done using *tc* (traffic controller) based on *Iproute2* a program that allows advanced IP routing. With this tool it is possible to control both incoming and outgoing traffic. Furthermore, the latest versions (above 2.6.8.1) allow to control the latency of the network interface.

Memory Limitation. Wrekavoc is able to limit the amount of memory available by the processes thanks to the use of the mlock and munlock syscalls.

Configuring and Controlling Nodes and Links. The configuration of a homogeneous cluster is made through the notion of islet. An islet is a set of nodes that share similar limitations. Two islets are linked together by a virtual network which can also be limited. An islet is defined as a union of IP addresses (or machine names) intervals.

Each islet configuration is stored into a configuration file. At the end of this file is described the network connection (bandwidth and latency) between each islet.

5.4. SimGrid

Participants: Abdelmalek Cherier, Martin Quinson.

The SIMG*rid framework aims at being a scientific instrument to the evaluation of algorithmic solutions for large-scale distributed experiments.*

The SIMGrid tool is the result of a collaboration with Henri Casanova (Univ. of Hawaii, Manoa) and Arnaud Legrand (MESCAL team, INRIA Grenoble-Rhône-Alpes, France). Simulation is a common answer to the grid specific challenges such as scale and heterogeneity. SIMGrid is one of the major simulators in the Grid community.

The framework relies on a simulation kernel using a blend of analytical models and coarse-grain discrete event simulation. It proves several orders of magnitude faster than usual packet-level simulators used in the networking community (such as ns2 or GTNetS) while providing an acceptable level of accuracy [40].

SIMGrid provides several user interfaces depending on the user goal.

MSG helps the study of distributed heuristics. This is the historical interface of SIMGrid, and remains the most used interface.

SimDag eases the study of scheduling heuristics for DAGs of (parallel) tasks. It was developed by Christophe Thiery during an internship in the EPI during the summer 2006. This functionality helps the work on parallel task scheduling (*cf.* 6.2.3).

GRAS (Grid Reality And Simulation) eases the development of Grid services and infrastructures [10].

GRAS provides a C ANSI interface to build distributed services and infrastructures for the Grid. Two implementations of this API are provided: the first one (called Grid R&D Kit) lets the developers experiment, test and debug their work within the SimGrid simulator. The other implementation (called Grid Runtime Environment) allows the resulting programs to run efficiently on real systems.

The simulator thus greatly eases the research and development of Grid services (such as for example monitoring infrastructure or distributed storage systems). In addition, the Grid Runtime Environment is ported to Linux, Windows, Solaris, Mac OS X, AIX and IRIX operating systems, and to 11 hardware architectures. Services built on top of this achieve better communication performance than heterogeneous implementations of the MPI protocol.

The SIMGrid framework is available from http://gforge.inria.fr/projects/simgrid/.

5.5. LaPIe

Participant: Luiz Angelo Steffenel.

LAPIE *is an automatically tuned collective communication library designed for large-scale heterogeneous systems.*

The popularity of heterogeneous parallel processing environments like clusters and computer grids has emphasized the impact of network heterogeneity on the performance of parallel applications. Collective communication operations are especially concerned by this problem, as heterogeneity interferes directly on the communication performance.

LAPIE provides a set of MPI collective communication operations especially designed to perform automatic adaptation according to the network characteristics. Indeed, LAPIE combines both topology discovery and performance prediction to choose the best communication scheduling that minimizes the overall communication time for a given operation.

LAPIE is distributed as a programming library that overloads (through profiling) existing MPI calls. This allows LAPIE to be easily integrated into existing applications without modifying their code - we just need to recompile them. In addition, LAPIE was designed to facilitate the addition of new scheduling techniques and collective communication operations. Therefore, LAPIE provides an excellent testbed to develop and evaluate new communication scheduling techniques and/or architecture-specific optimizations.

LAPIE currently supports *MPI_Bcast*, *MPI_Scatter* and *MPI_Alltoall* operations, whose efficiency was evaluated in several papers we published. New operations such as *MPI_Reduce*, *MPI_Gather* and *MPI_Allgather* should be added very soon.

5.6. P2P-MPI

Participant: Stéphane Genaud.

P2P-MPI *is an integrated middleware and communication library designed for large-scale applications deployment.*

Many obstacles hinder the deployment of parallel applications on grids. One major obstacle is to find, amongst an heterogeneous, ever-changing and unstable set of resources, some reliable and adapted resources to execute a job request. P2P-MPI alleviates this task by proposing a peer-to-peer based platform in which available resources are dynamically discovered upon job requests, and by providing a fault-tolerant message-passing library for Java programs.

Communication library. P2P-MPI provides an MPI-like implementation in Java, following the MPJ specification. Java has been chosen for its "run everywhere" feature, which has shown to be useful in grid environments.

Fault-tolerance. The communication library implements fault-tolerance through replication of processes. A number of copies of each process may be asked to run simultaneously at runtime. So, contrarily to an MPI application that crashes as soon as any of its processes crash, a P2P-MPI using replication will be able to continue as long as at least one copy of each process is running.

Resource discovery. Contrarily to most MPI implementations that rely on a static description of resources, P2P-MPI has adopted a peer-to-peer architecture to adapt to volatility of resources. A resource joins the P2P-MPI grid and becomes available to others when a simple user (no root privilege needed) starts a P2P-MPI peer. Thus, at each job request, the middleware handles a discovery of available resources, possibly guided by simple strategies indicated by the user, to satisfy the job needs.

6. New Results

6.1. Structuring of Applications for Scalability

Participants: Stefan Canzar, Pierre-Nicolas Clauss, Jens Gustedt, Constantinos Makassikis, Frédéric Suter, Stéphane Vialle.

6.1.1. Large Scale Experiments

The integration of the formerly separated libraries *ParCeL* and *SSCRAP* into *parXXL* allows to validate the whole on a wide range of fine grained applications and problems. Among the applications that we started testing this year is the interactive simulation of PDEs in physics, based on the Escapade project, see [21]. There the idea is to express PDEs as local equations in the discretized variable space and to map them in terms of update functions on a cellular automaton. With the help of *parXXL* this fine-grained automaton can then be mapped on the coarse grained target machine and evaluated efficiently. Our hope is to be able to allow to find solutions for certain types of physically motivated problems, for which currently no performing solvers exist.

6.1.2. Distribution of a Stochastic Control Algorithm

For investment purpose EDF has to valuate gas storage assets used to secure its gas provisioning. *Real option models* are used to give a price to this kind of model and lead us to solve stochastic control problems where the uncertainty is due to the price variations. The gas price is modeled by different Markovian stochastic processes. In 2007 we have designed, implemented and evaluated the distribution of this stochastic control algorithm, see [23]. The framework for this research was the thesis of Constantinos Makassikis, in collaboration with Supélec. This thesis took part in the ANR-CIGC "GCPMF" about distribution of financial computations, involving both Supélec and EDF R&D.

From a parallel algorithmic point of view, the challenge was to efficiently distribute an iterative algorithm for which amount of computations and the input data range evolved at each step. To avoid duplication of data it has been mandatory to redistribute computations and data at each such step: a distribution map and a routing scheme are computed on each processor. The sequential stochastic computing routines have been developed by EDF R&D, and three different *pricing models* (of gas and gas storage) have been successfully plugged into our generic and distributed skeleton of stochastic control algorithm and experimented.

The distributed application has been implemented with MPI-C++ and has achieved very good performances on three different platforms: a low cost cluster of 32 desktop PC, a cluster of 71 bi-processor PC, and an IBM Blue Gene/L supercomputer up to 8192 processors. All three different pricing models that we implemented with our distributed stochastic control skeleton have achieved significant speedup: for example, up to 407 on 512 processors and 680 on 1024 processors of the Blue Gene supercomputer, running the "*normal inverse Gaussian*" model. Moreover, it has been possible to run the so-called "*2 factor Gaussian*" model on realistic data using at least 16 PC or 32 nodes of Blue Gene supercomputer, which allows for an easy comparison of sophisticated pricing models at EDF.

In contrast to the Blue Gene, on the PC clusters we observed many failures that forced an abortion of the execution. We are now investigating fault tolerance for financial computations, and we will use our distributed stochastic control application to test the fault tolerance mechanisms that we will design.

6.1.3. Models and Algorithms for Coarse Grained Computation

For testing and benchmarking the generation of large random input data with known probability distributions is crucial. In [12], we show how to uniformly distribute data at random in two related settings: *coarse grained parallelism* and *external memory*. In contrast to previously known work for parallel setups, our method is able to fulfill the three criteria of uniformity, work-optimality and balance among the processors simultaneously. To guarantee the uniformity we investigate the matrix of communication requests between the processors. We show that its distribution is a generalization of the multivariate hypergeometric distribution and we give algorithms to sample it efficiently in the two settings.

If instead of shuffling existing data we constrain ourselves to produce permutations of integers $0, ..., n$ for some large number n, solutions that are much more efficient become possible. First, we are able to show that the communication mentioned above can be improved by using adapted compression techniques. In particular the information theoretic lower bound of $\Theta(n \log p)$ (instead of $\Theta(n \log n)$) for the overall communication can be matched up to a constant factor. Second, if instead of communicating data (integers in that case) we generate them in place, we achieve a scheme that allows for a trade-off between the number of random bits (*i.e.,*the quality of the target distribution) and the total running time of the algorithm. This approach is presented in [31].

6.1.4. Overlapping Computations and Communications with I/O

In [2], we noticed that the performance of our pipeline algorithm was impacted by asynchronous communications that introduced gaps between I/O operations. To address this issue we studied how to adapt this kind of algorithms, that is wavefront algorithms, to shared memory platforms.

Using the *parXXL* library we were able to propose an architecture-independent out-of-core implementation of a well known hydrodynamics kernel, see [35]. In addition to this implementation we proposed an optimized data layout that allows to reduce the I/O impact on each iteration of the algorithm by an order of magnitude at the cost of an initial rewriting of the data. This work will be published in the proceedings of the 16th Euromicro International Conference on Parallel, Distributed and network-based Processing (EuroPDP 2008) [16].

6.1.5. Combinatorial Optimization in Bioinformatics

In recent years, more and more biological problems were formulated in terms of combinatorial optimization models, and both heuristics and approximation algorithms, as well as exact algorithms were proposed to solve these problems. The aim of the thesis of Stefan Canzar, a collaboration with the MPI Saarbrücken that predates the INRIA project creation of AlGorille, is to pick up some well established combinatorial problems and work on a state-of-the-art implementation, or to formalize problems from biology in terms of combinatorial optimization models and propose methods to solve these problems efficiently within the corresponding model. In either case we want to show that modern methods of optimization are capable of solving large instances such as those from biology.

The works [15] and [30] tackle the multiple sequence alignment problem (MSA), one of the most important problems in computational biology. They provide Lagrangian relaxation of an integer linear programming formulation for the problem and report on an implementation that outperforms all known exact algorithms for the MSA.

6.2. Transparent Resource Management

Participants: Eloi Du Bois, Louis-Claude Canon, Emmanuel Jeannot, Tchimou N'Takpé, Luiz Angelo Steffenel, Frédéric Suter.

6.2.1. Reliable Scheduling

In some environments, resources can be volatile. They can come and go in an unpredictable way. Scheduling an application on such resources is therefore, very challenging because failure of the resources can cause a global failure of the whole application.

We have tackled the problem of scheduling task graphs onto a heterogeneous set of machines, where each processor has a probability of failure governed by an exponential law. The goal was to design algorithms that optimize both makespan and reliability. First, we provide an optimal scheduling algorithm for independent unitary tasks where the objective is to maximize the reliability subject to makespan minimization. For the bi-criteria case, we provided an algorithm that approximates the Pareto-curve. Next, for independent non-unitary tasks, we have showed that the product failure rate times unitary instruction execution time is crucial to distinguish processors in this context. Based on these results we were able to let the user choose a trade-off between reliability maximization and makespan minimization. For general task graphs we provide a method for converting scheduling heuristics on heterogeneous cluster into heuristics that take reliability into account. Here again, we have shown how we can help the user to select a trade-off between makespan and reliability. All these results are published in [18].

6.2.2. Robust Scheduling

A schedule is said robust if it is able to absorb some degree of uncertainty in tasks duration while maintaining a stable solution. This intuitive notion of robustness has led to many different interpretations and metrics. However, no comparison of these different metrics has ever been performed. We have performed an experimental study of these different metrics and shown how they are correlated to each other in the case of task scheduling, with dependencies between tasks.

Computing the makespan distribution when task and communication duration are given by probabilistic distribution is a #-P complete problem. We have studied different ways to approximate this problem based on previous results of the literature on the PERT network. For comparing these different methods we have computed the makespan distribution using Monte-Carlo simulation.

We have determined that the makespan standard deviation is a good and easy to compute metric for this problem. Based on that, we have designed a robust scheduling strategy based on our understanding of the problem. More precisely we have proposed a parameterized approach that helps the user to choose between a fast schedule (makespan optimization) or a robust one. We have compared this heuristic with a genetic algorithm that is able to find a very good set of solutions [14].

6.2.3. Parallel Task Scheduling

This year we proposed improvements of the M-HEFT [33] and the HCPA heuristic [43] to address some of their limitations. The allocation phase of HCPA has been improved by proposing a new stopping criterion that allows smaller but still efficient allocations. We also introduced a packing strategy in order to fill gaps that may appear in its placement phase as a task may be delayed unnecessarily just because its computed processor allocation is (perhaps only slightly) larger than the number of processors available at the time when the task is ready for execution. We also address a glaring drawback of M-HEFT, which is that it tends to use very large processor allocations for application tasks. This is simply due to the fact that a task's processor allocation is chosen "blindly" so that the task's completion time is minimized. To remedy this problem with M-HEFT we propose three simple ways to bound a task's processor allocation. We proposed a comparison of the heuristics derived from two orthogonal approaches in [9]. Approach #1 adapts two-phase scheduling algorithms for mixed-parallel applications on homogeneous platforms to make them amenable to heterogeneous platforms while Approach #2 adapts list scheduling algorithms for task-parallel applications on heterogeneous platforms to make them amenable to mixed-parallelism.The main result of this comparison is that no algorithm emerges as a clear winner. However, it seems that Approach #1 leads to the most likely desired trade-off between makespan and efficiency. One advantage of Approach #2 is that the algorithms are tunable with simple parameters to achieve trade-offs between performance and efficiency. Our conclusion is that Approach #1 is appropriate for the vast majority of the users, but that sophisticated users could opt for Approach #2.

After noticing that heuristics of Approach #2 often take selfish processor allocation decisions for each task of the scheduling list. For instance these heuristics do not consider that potentially concurrent tasks may be impacted by the decision taken for the first task of the list. To address this issue we proposed an original heuristic, called Δ-Critical Task Scheduling, in which we decompose the scheduling list into groups of tasks having almost the same priority and thus being almost as critical [29]. For a task in such a group, the maximal number of processors that can be allocated to it is then bounded to ensure that all the tasks of the group may have a chance to be executed concurrently.

We also conducted a comparison of the previously cited heuristics with a guaranteed algorithm. This work made in collaboration with Henri Casanova, at University of Hawai'i, Manoa and Pierre-François Dutot, at University of Grenoble 2, computes an optimal allocation by linear programming and relies on a list scheduling algorithm to place these allocated tasks.

Finally we started to study the scheduling of multiple Parallel Task Graphs onto a shared heterogeneous platform. We made a first step towards this objective by focusing on the allocation procedure and handling the concurrent access to resources by imposing a resource constraint on the schedule. This led us to propose two procedures that determine allocations while respecting that constraint expressed as a ratio of the available processing power of the target platform [24]. The next step is to design a placement procedure taking several allocated graphs as input, each allocation respecting a resource constraint. Some challenging issues arise in the design of this *multiple*-application scheduler such as ensuring fairness between users, defining priority functions to favor applications scheduled under tighter constraints, or allowing the dynamic submission of applications and thus adapting the resource constraint to the new load conditions.

6.2.4. Adaptive On-line Compression of Medical Images

Transferring medical images through a network is a very important task for two reasons. First, medical images can be acquired or stored on a different place than where they are used or analyzed. Second, such images can be very large (up to several Gbytes for 3D multi-resolution ones). The need to rapidly transfer these images is therefore crucial on a daily basis for any practitioner who works in this field. AdOC [8] (Adaptive Online Compression) is a tool we have developed and that enables on-the-fly lossless compression of data and files

for speeding-up their transfer on standard network (*e.g.,*TCP). It uses two ideas: (1) overlapping compression and communication; (2) adaptation of the compression effort depending on the current speed of the end CPUs and the network. However, lossless compression as used by AdOC (*e.g.,*Lempel-Ziv based algorithms) is not very efficient in the case of images and especially medical ones. We, therefore, have proposed to use lossy compression for transferring these images. Nevertheless, using lossy compression in the case of medical images raises several issues. First, lossy compression when aggressively used (*i.e.,*at a high compression rate) degrades the quality of the image and therefore hinders the ability of the practitioner to make correct diagnostics. Second, lossy compression algorithms, such as the one based on wavelet transform are very costly and time consuming. Hence, using such compression on-the-fly is very challenging.

We have tackled the two above issues as follows. Concerning the quality of the image we have adopted a conservative strategy where the quantization steps are computed such that the image is never visually degraded. With our approach, the Peak Signal-to-Noise Ratio (PSNR) is always above 35 db and sometimes above 40 db. Concerning the second problem we have used a very fast wavelet transform called WAILI: it enables to compress a single image in some milliseconds.

We have tested this implementation on several images and several kinds of networks. Preliminary results show that the proposed approach is very promising for wide-area network such as Internet both in terms of transfer speed (it can be multiplied by 3 on transatlantic connection compared to FTP transfer) and quality of the images (they do not suffer visible degradation). On very fast network (such as local area network or Gbit network), this approach does not provide any improvement, but this is not a very important case as the transfer time is already very small.

In conclusion, this work is a proof of concept that lossy adaptive on-line compression can be useful for medical images in the case of slow network without preventing practitioner of making a correct diagnostic.

6.2.5. Grid-aware Total Exchange

Total Exchange is one of the most important collective communication patterns for scientific applications. We have observed that the traditional MPI_Alltoall implementation is not suited for grid environments, as it is both inefficient and hard to model. Hence, we have proposed an algorithm called $\mathcal{L}G$ for the total exchange redistribution problem between two clusters. In our approach we perform communications in two different phases, aiming to minimize the number of communication steps through the wide-area network. Therefore, we are able to reduce the number of messages exchanged through the backbone to only $2 \times \max(n_1, n_2)$ against $2 \times n_1 \times n_2$ messages with the traditional strategy (where n_1 and n_2 are the numbers of nodes of each clusters). Experimental results show that we reach over than 50% of performance improvement comparing to the traditional strategies [22].

6.2.6. Total Exchange Performance Prediction

Based on the above algorithm, we have addressed the problem of modeling the performance of *Total Exchange* communication operations in grid environments. Because traditional performance models are unable to predict the real completion time of an All-to-All operation, we try to cope with this problem by identifying the factors that can interfere in both local and distant transmissions. Our proposed algorithms aim at minimizing the number of communication steps through the wide-area network. We have seen that such reduction has a direct impact on the performance modeling of the MPI_Alltoall operation, as we minimize the factors that interfere with wide-area communications. Hence, we are able to define an accurate performance modeling of a total exchange between two clusters [27].

6.3. Experimental Validation

Participants: Abdelmalek Cherier, Xavier Delaruelle, Olivier Dubuisson, Emmanuel Jeannot, Jens Gustedt, Martin Quinson, Frédéric Suter.

6.3.1. Synthetic but Realistic Platform Configurations for Driving Simulations

Simulation allows repeatable results, makes it possible to explore various platform scenarios at will, is not as labor-intensive or as costly as running experiments on a real platform, and often makes it possible to run enormous numbers of experiments quickly. But every researcher resorting to simulation is then faced with the question: "which platform configurations should I simulate?" One approach is to generate random platform configurations using simple uniform probability distributions, while another consists in building statistical models of real-world clusters and to use these models to generate cluster configurations based on a large set of real-world cluster configurations. Although these models should lead to more representative platform configurations, many researchers opt for using real-world platform configurations directly. The drawback is that typically only a few such configurations are constructed. Instead, we proposed the use of a compendium of platform configurations that correspond to subsets of the Grid'5000 platform [32].

This work raised a problem with the XML format used by SIMGrid to represent synthetic platforms as every host and every possible network route have to be described. In case of an homogeneous cluster, most of this information is redundant. Furthermore each computer is only represented by its computing power while many scheduling algorithms aim at performing matchmaking for other resources (such as memory or disk space) before building the schedule. These issues led us to reconsider the representation format used by SIMGrid to simulate synthetic large scale platforms.

6.3.2. Improvement of the SimGrid tool

This year, we pursued our efforts to increase the user community of SIMGrid. We improved the overall stability of the tool, and in particular the usability of the Windows port. We also developed Java bindings of the tool because we felt that imposing the C language to the users was hindering the adoption of the tool by the community.

We also improved the XML representation to factorize routing information as explained in previous point. This allows SIMGrid to deal with larger platform instances, and ease the model instantiation from user point of view. Several code refactoring done in the tool also allowed to gain a factor of 4 in performance.

SIMGrid is freely downloadable [34] and its user base is rapidly growing, resulting in the publication of about ten publications (half of them from users not being part of the core team).

6.3.3. Grid Platform Discovery

Due to the changing characteristics of the Grids, distributed applications targeting these platforms must be network-aware and react to the condition changes. To make this possible, applications must have a synthetic view of the network condition they experiment. Several platform monitoring tools exist, but they provide irrelevant or incomplete information to network-aware applications. Most of these tools intend to help the network administrator to detect abnormalities in their system. They thus concentrate on very low level metrics such as the amount of data emited by a given host where network-aware applications need to access the available bandwidth between host pairs. Some tools were designed specifically to provide such higher-level information (the most predominant being NWS – [45]), but they are limited to quantitative information about the bandwidth, latency and processor availability.

We pursued our effort in the design of network mapping from the application point of view. We are mainly interested in predicting the effect of resource sharing between concurrent data stream. This information is for example crucial to schedule individual messages of group communications or to compute the optimal localization of backup servers and storage areas.

Using the testing framework designed previous year [42], we showed that the existing algorithms from the literature lead to wrong prediction when the network contention increases. We proposed several new heuristics leading to better results, as shown in [20], [19].

6.3.4. Wrekavoc

This year has seen an important stabilization and re-engineering effort for Wrekavoc. The library is now very modular, cleanly separating the different components of the emulation (CPU, memory, network). The group of E. Aubanel at Univ. of Newbrunswick, Canada, builds on Wrekavoc for an emulator for the alpha architecture.

We have added the possibility to set an IP alias for each emulated node in order to simplify configuration, calibration and experiments. We also have enabled islet gateways in order to aggregate the bandwidth from several islets. We have implemented the RIP routing strategy to enable communications between two islets that are not directly connected.

To validate Wrekavoc, we have compared the results obtained on a real heterogeneous cluster composed by twelve nodes (from Pentium II to Pentium IV) to emulations obtained with Wrekavoc, executed on a cluster composed of homogeneous nodes. We compared different software packages like Povray, sorting programs implemented with *parXXL*, some matrix multiplication algorithms and other parallel softwares. The results show that Wrekavoc is very realistic both quantitatively and qualitatively.

6.3.5. Grid'5000

Grid'5000 aims at building an experimental Grid platform featuring a total of five thousands CPUs over nine sites in France. We have built one of these sites by installing two clusters. The first machine is a 47-nodes HP cluster. Each compute node of the HP cluster has two 2 GHz AMD Opteron 246 with 2 GB of RAM and runs under Linux Debian. The second machine is a 120-nodes HP cluster. Each compute node has two 1.6 GHz Dual-core Intel Xeon 5110 with 2 GB of RAM and two Gigabit Ethernet interfaces. The two clusters are connected to the grid through a 10 Gigabit Ethernet network, provided by Renater. We were the first site to provide this 10 Gigabit uplink. We manage the day to day usage of the cluster and regularly update it to fit as close as possible the Grid'5000 recommendations.

We support the local and national Grid'5000 users by helping them using the platform. We provide them trainings and we try to find with them the best way for their experiments to use the grid. We take a significant part in the organization of the "Grid'5000 spring school 2006" and we created our own education-day called "Journée Grid'5000 au Loria".

We are taking a significant part in the national development of the Grid'5000 platform. We help consolidating the production infrastructure, by developing tools like the account management software called cpu-g5k (https://gforge.inria.fr/projects/grid5000/) or creating Linux-based grid environment for users. We also participate in most of the existing working groups of the project, like the one for the next Kadeploy version (https://gforge.inria.fr/projects/kadeploy/). Moreover we take in charge some of the collaborative services like the wiki website and the bug reporting tool. We also have work on homogenizing the Grid'5000 infrastructure by preparing and deploying the new OAR version 2.0 [17].

7. Other Grants and Activities

7.1. National Initiatives

7.1.1. CNRS initiatives, GDR-ARP and specific initiatives

We participate at numerous national initiatives. In the GDR-ASR (architecture, systems, and networks) we take part in TAROT[1], Grappes[2], and RGE[3]. We also participate to the animation of the GDR-ASR as a whole, and Luiz Angelo Steffenel organized the RGE meeting that took place on February 8th 2007 in Nancy.

The finances of RGE, lead by Stéphane Vialle at Supélec, are maintained by AlGorille.

[1] *Techniques algorithmiques, réseaux et d'optimisation pour les télécommunications*
[2] *Architecture, systèmes, outils et applications pour réseaux de stations de travail hautes performances*
[3] *Réseau Grand Est*

7.1.2. ACI initiatives of the French Research Ministry

We are partners in one ACI initiative:

- In the 2004 initiative ACI AGIR we participate in the definition and design of a set of services for medical image processing on the grid. More precisely we are in charge of transfer with compression task and the evaluation of grid middleware.

7.1.3. ARA Initiatives of the French Research Ministry

We are partners in one project of the ARA Masses de données (thematic call to project from the French Research Ministry):

- ALPAGE.

  The new algorithmic challenges associated with large-scale platforms have been approached from two different directions. On the one hand, the parallel algorithms community has largely concentrated on the problems associated with heterogeneity and large amounts of data. Algorithms have been based on a centralized single-node, responsible for calculating the optimal solution; this approach induces significant computing times on the organizing node, and requires centralizing all the information about the platform. Therefore, these solutions clearly suffer from scalability and fault tolerance problems.

  On the other hand, the distributed systems community has focused on scalability and fault-tolerance issues. The success of file sharing applications demonstrates the capacity of the resulting algorithms to manage huge volumes of data and users on large unstable platforms. Algorithms developed within this context are completely distributed and based on peer-to-peer communications. They are well adapted to very irregular applications, for which the communication pattern is unpredictable. But in the case of more regular applications, they lead to a significant waste of resources.

  The goal of the ALPAGE project is to establish a link between these directions, by gathering researchers (ID, LIP, LORIA, LaBRI, LIX, LRI) from the distributed systems and parallel algorithms communities. More precisely, the objective is to develop efficient and robust algorithms for some elementary applications, such as broadcast and multicast, distribution of tasks that may or may not share files, resource discovery. These fundamental applications correspond well to the spectrum of the applications that can be considered on large scale, distributed platforms.

7.2. European Initiatives

7.2.1. NoE CoreGrid

We take part in the NoE "*CoreGrid*" lead by Thierry Priol from INRIA Rennes. More precisely we are part of the work package 6 on scheduling. Emmanuel Jeannot is the leader for CNRS of task 6.5: evaluation and benchmarking.

7.2.2. Bilateral Collaborations

We maintain several European collaborations with other research teams. The two most fruitful are with the team of Jan Arne Telle from Bergen University, Norway and with Vandy Berten and Joël Goossens of the Université Libre de Bruxelles on scheduling problems under stochastic models. Recently we setup a new collaboration with Jon Weissman (University of Minnesota, Twin Cities) on scheduling with uncertainty models.

7.3. International Initiatives

7.3.1. NSF-INRIA Grant

Our collaboration with Jack Dongarra of the University of Tennessee, Knoxville and the GRAAL project of INRIA, has been formalized in an INRIA-NSF project which handles the aspects of the integration of our scheduling algorithms into NetSolve.

7.3.2. Bilateral Collaborations

We collaborate with Henri Casanova of University of Hawai'i at Manoa on parallel task scheduling heuristics for heterogeneous environments as well as on the simulation of grid platforms within the SimGrid project.

We collaborate with Prof. Rich Wolski of University of California at Santa Barbara on grid platforms monitoring and characteristics discovering within the NWS project.

7.4. Visits

From November, 23 to December, 8 2007, Fréderic Suter spent two weeks at the University of Hawai'i at Manoa under the INRIA "Explorateur" Program. The purpose of this visit is to define the frame of a future extension of the existing bilateral collaboration and to prepare a application file to the "INRIA Associate Team" program call for 2008.

8. Dissemination

8.1. Dissemination

8.1.1. Leadership within the Scientific Community

Emmanuel Jeannot is member of the steering committee of the Grid'5000 project and head of the Nancy site. Martin Quinson is serving as vice-head for the Grid'5000 project.

Aladdin (A LArge-scale Distributed Deployable INfrastructure) is an INRIA Technological Development Action. It is a management structure for Grid'5000. Emmanuel Jeannot is member of its direction committee. Stéphane Genaud, Emmanuel Jeannot and Martin Quinson co-steer two working groups on two scientific challenges namely "*scalable application for large scale systems (algorithm, programming, execution models)*" and "*Modeling of large scale systems and validation of their simulators*".

8.1.2. Scientific Expertise

In 2007, Jens Gustedt was a member of the thesis committee of Pascal Pons, University Paris 7 and has served as an external expert for the evaluation of scientific projects in regional initiatives for information science and technology in a neighboring European country.

In 2007, Emmanuel Jeannot was a member of the thesis committee of Mohand Mezmaz, Université des Sciences et Technologies de Lille.

8.1.3. Teaching Activities

Frédéric Suter is teaching *Algorithmique et programmation* (L1), *Réseaux et Internet* (M2Pro-IMOI) and *Grilles informatiques et algorithmique distribuée avancée* at Henri Poincaré University.

Martin Quinson is teaching the following modules at ÉSIAL (University Henri Poincaré): *C et Shell* (1A), *Réseaux et système* (2A) and *Programmation d'applications réparties* (3A). He also participates to the following modules: *Informatique de base* (1A), *Algorithmique Parallèle et Distribuée* (3A) and *Grilles informatiques et algorithmique distribuée avancée* at Henri Poincaré University. He is also responsible of the specialization *Système et Applications Distribuées* of ÉSIAL.

Luiz Angelo Steffenel taught the following modules at IUT Nancy Charlemagne (Nancy 2 University): *Introduction à l'Algorithmique* (DUT 1A), *Bases de la Programmation - Java* (DUT 1A), *Systèmes de Gestion de Bases de Données* (DUT 2A), *Architecture 1* (DUT 1A Bis), *Administration de Systèmes de Gestion de Bases de Données* (DUT AS). He also participated to the following modules at the UFR Mathématique et Informatique (Nancy 2 University): *Programmation C Avancée* (MIAGE 3A), *Certificat C2I* (1A). From September 2007 Luiz Angelo Steffenel moves to University of Reims Champagne-Ardenne, where he is now Assistant Professor.

Sylvain Contassot-Vivier is teaching *Parallel computation* (M1), *Grids* (M2), *Statistical learning* (M2) and *Network security* (M2) at the Henri Poincaré University - Nancy 1.

8.1.4. Editorial Activities

Since October 2001, Jens Gustedt is Editor-in-Chief of the journal *Discrete Mathematics and Theoretical Computer Science* (DMTCS). DMTCS is a journal that is published electronically by an independent association under French law. Based on a contract with INRIA, its main web-server is located at the LORIA. DMTCS has acquired a good visibility within the concerned domains of Computer Science and Mathematics, which is confirmed by a relatively high impact factor.

In 2007, Jens Gustedt has served as program committee member of the 16th Euromicro International Conference on Parallel, Distributed and network-based Processing PDP 2008.

Emmanuel Jeannot was member of the program committee of the High Performance Distributed Computing conference HPDC 2007, Europar 2007 and the Sixth International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks Heteropar 2007.

Emmanuel Jeannot and Frédéric Suter were members of the program committee of the sixteenth Heterogeneity in Computing Workshop (HCW 2007).

8.1.5. Refereeing

In 2007, members of the team served as referees for the following journals and conferences:

Journals: Discrete Applied Mathematics, IEICE transactions, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Neural Networks, International Journal of High Performance Computing, Journal of Parallel and Distributed Computing, Scientific Programming, Theoretical Computer Science,

Conferences: e-Science 2007, EuroPar 2007, Heteropar 2007, HCW 2007, PDP 2008.

8.1.6. Invitation

Emmanuel Jeannot has given an invited tutorial entitled “Research issues in grid computing : an algorithmician point of view” at the Grid@Mons 2007 conference on May 4th 2007.

Emmanuel Jeannot has given an invited talk entitled “*Modeling the LU Factorization on SMP clusters*” at the Meeting on Parallel Routines Optimizations and Applications in Murcia, Spain, 12-13 June 2007.

9. Bibliography

Major publications by the team in recent years

[1] Y. CANIOU, E. JEANNOT. *Multi-Criteria Scheduling Heuristics for GridRPC Systems*, in "International Journal of High Performance Computing Applications", vol. 20, n^{o} 1, spring 2006, p. 61–76.

[2] E. CARON, F. DESPREZ, F. SUTER. *Out-of-Core and Pipeline Techniques for Wavefront Algorithms*, in "Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS’05), Denver, CO", April 2005.

[3] J. COHEN, E. JEANNOT, N. PADOY, F. WAGNER. *Message Scheduling for Parallel Data Redistribution between Clusters*, in "IEEE Transactions on Parallel and Distributed Systems", vol. 17, n^{o} 10, October 2006, p. 1163–1175.

[4] L. EYRAUD-DUBOIS, A. LEGRAND, M. QUINSON, F. VIVIEN. *A First Step Towards Automatically Building Network Representations*, in "Proceedings of the 13th International EuroPar Conference, Rennes, France", Lecture Notes in Computer Science, vol. 4641, Springer, August 2007, p. 160–169, http://hal.inria.fr/inria-00130734/en/.

[5] A. H. GEBREMEDHIN, J. GUSTEDT, M. ESSAÏDI, I. GUÉRIN LASSOUS, J. A. TELLE. *PRO: A Model for the Design and Analysis of Efficient and Scalable Parallel Algorithms*, in "Nordic Journal of Computing", vol. 13, 2006, p. 215-239, http://hal.inria.fr/inria-00000899/en/.

[6] J. GUSTEDT. *Towards Realistic Implementations of External Memory Algorithms using a Coarse Grained Paradigm*, in "International Conference on Computer Science and its Applications - ICCSA'2003, Montréal, Canada", Lecture Notes in Computer Science, vol. 2668, Springer, February 2003, p. 269-278.

[7] J. GUSTEDT. *Data Handover: Reconciling Message Passing and Shared Memory*, Technical report, n^{o} RR-5383, INRIA, Nov 2004, http://hal.inria.fr/inria-00070620.

[8] E. JEANNOT. *Improving Middleware Performance with AdOC: an Adaptive Online Compression Library for Data Transfer*, in "International Parallel and Distributed Processing Symposium 2005 (IPDPS'05), Denver, Colorado, USA", April 2005.

[9] T. N'TAKPÉ, F. SUTER, H. CASANOVA. *A Comparison of Scheduling Approaches for Mixed-Parallel Applications on Heterogeneous Platforms*, in "6th International Symposium on Parallel and Distributed Computing, Hagenberg, Austria", July 2007, http://hal.inria.fr/inria-00151812/en/.

[10] M. QUINSON. *GRAS: a Research and Development Framework for Grid and P2P Infrastructures*, in "The 18th IASTED International Conference on Parallel and Distributed Computing and Systems", IASTED (editor), 2006, http://hal.inria.fr/inria-00108389.

Year Publications

Doctoral dissertations and Habilitation theses

[11] E. JEANNOT. *Algorithmes et protocoles pour la gestion des données et des calculs dans les environnements distribués et hétérogènes*, Habilitation à Diriger des Recherches, Université Henri Poincaré - Nancy I, October 2007, http://tel.archives-ouvertes.fr/tel-00185887/en/.

Articles in refereed journals and book chapters

[12] J. GUSTEDT. *Efficient Sampling of Random Permutations*, in "Journal of Discrete Algorithms", to be published in J. on Discrete Algorithms, doi:10.1016/j.jda.2006.11.002, 2007, http://hal.inria.fr/inria-00000900/en/.

[13] E. JEANNOT, K. SEYMOUR, A. YARKHAN, J. J. DONGARRA. *Improved Runtime and Transfer Time Prediction Mechanisms in a Network Enabled Servers Middleware*, in "Parallel Processing Letters", vol. 17, 2007, p. 47 - 59, http://hal.inria.fr/inria-00177526.

Publications in Conferences and Workshops

[14] L.-C. CANON, E. JEANNOT. *A Comparison of Robustness Metrics for Scheduling DAGs on Heterogeneous Systems*, in "Sixth International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks - HeteroPar'07, Austin États-Unis d'Amérique", In conjunction with [Cluster 2007]

The 2007 IEEE International Conference on Cluster Computing. The workshop proceedings will be published through the IEEE Computer Society Press as part of the Cluster 2007 proceedings., 2007, 10 pages, http://hal.inria.fr/inria-00170581/en/.

[15] S. CANZAR, E. ALTHAUS. *A Lagrangian Relaxation Approach for the Multiple Sequence Alignment Problem*, in "Combinatorial Optimization and Applications, First International Conference, COCOA 2007, Xi'an Chine", vol. 4616, 2007, p. 267-278, http://hal.archives-ouvertes.fr/hal-00177224/en/.

[16] P.-N. CLAUSS, J. GUSTEDT, F. SUTER. *Out-of-Core Wavefront Computations with Reduced Synchronization*, in "16th Euromicro International Conference on Parallel, Distributed and network-based Processing, Toulouse France", J. BOURGEOIS, F. SPIES, D. E. BAZ (editors), To appear, IEEE, 2008, 8 pages, http://hal.inria.fr/inria-00176084/en/.

[17] P. D'ANFRAY, E. JEANNOT. *GRID'5000 une plate-forme d'expérimentation pour les systèmes distribués à large échelle*, in "Journées Réseaux - JRES 2007, Strasbourg France", 2007, 6 pages, http://hal.inria.fr/inria-00181446/en/.

[18] J. J. DONGARRA, E. JEANNOT, E. SAULE, Z. SHI. *Bi-objective Scheduling Algorithms for Optimizing Makespan and Reliability on Heterogeneous Systems*, in "Symposium on Parallelism in Algorithms and Architectures (SPAA'07), San-Diego, CA États-Unis d'Amérique", IEEE, 2007, 9 pages.

[19] L. EYRAUD-DUBOIS, A. LEGRAND, M. QUINSON, F. VIVIEN. *A First Step Towards Automatically Building Network Representations*, in "13th International European Conference on Parallel and Distributed Computing - Euro-Par 2007, Rennes France", Lecture Notes in Computer Science, vol. 4641, Springer Berlin / Heidelberg, August 2007, p. 160–169, http://hal.inria.fr/inria-00130734/en/.

[20] L. EYRAUD-DUBOIS, M. QUINSON. *Assessing the Quality of Automatically Built Network Representations*, in "CCGrid (Workshop on Programming Models for Grid Computing), Rio de Janeiro, Brazil.", May 2007, p. 795-800, http://hal.inria.fr/inria-00188408/en/.

[21] H. FREZZA-BUET, J. GUSTEDT, N. FRESENGEAS, S. VIALLE. *An Interactive Problem Modeller and PDE Solver Distributed on Large Scale Architectures*, in "Third International Workshop on Distributed Frameworks for Multimedia Applications - DFMA '07, Paris France", http://lifc.univ-fcomte.fr/dfma07/ CPER Région Lorraine MIS - InterCell, IEEE, June 2007, http://hal.inria.fr/inria-00139660/en/.

[22] E. JEANNOT, L. A. STEFFENEL. *Fast and Efficient Total Exchange on Two Clusters*, in "13th International European Conference on Parallel and Distributed Computing - Euro-Par 2007, Rennes France", Lecture Notes in Computer Science, vol. 4641, Springer Berlin / Heidelberg, August 2007, p. 848 - 857, http://hal.inria.fr/inria-00177533/en/.

[23] C. MAKASSIKIS, X. WARIN, S. VIALLE. *Distribution of a Stochastic Control Algorithm Applied to Gas Storage Valuation*, in "The 7th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT-2007), Cairo, Egypt", December 2007, 6.

[24] T. N'TAKPÉ, F. SUTER. *Self-Constrained Resource Allocation Procedures for Parallel Task Graph Scheduling on Shared Computing Grids*, in "19th IASTED International Conference on Parallel and Distributed Computing and Systems - PDCS 2007, Cambridge, MA", November 2007, http://hal.inria.fr/inria-00179735/en/.

[25] T. N'TAKPÉ, F. SUTER, H. CASANOVA. *A Comparison of Scheduling Approaches for Mixed-Parallel Applications on Heterogeneous Platforms*, in "6th International Symposium on Parallel and Distributed Computing, Hagenberg, Austria", July 2007, http://hal.inria.fr/inria-00151812/en/.

[26] W. NASRI, L. A. STEFFENEL, D. TRYSTRAM. *Adaptive Performance Modeling on Hierarchical Grid Computing Environments*, in "7th International Symposium on Cluster Computing and the Grid - CCGrid 2007", 8 pages, IEEE Computer Society, May 2007, http://hal.archives-ouvertes.fr/hal-00136202/en/.

[27] L. A. STEFFENEL, E. JEANNOT. *Total Exchange Performance Prediction on Grid Environments: modeling and algorithmic issues*, in "Towards Next Generation Grids – Proceedings of the CoreGRID Symposium 2007, Rennes France", T. PRIOL, M. VANNESCHI (editors), Springer US, 2007, p. 131-140, http://hal.inria.fr/inria-00177535/en/.

[28] L. A. STEFFENEL, M. MARTINASSO, D. TRYSTRAM. *Assessing Contention Effects on MPI_Alltoall Communications*, in "Proceedings of the 2nd International Conference on Grid and Pervasive Computing - GPC 2007", Lecture Notes in Computer Sciences (LNCS), Springer Verlag, May 2007, 12 pages, http://hal.archives-ouvertes.fr/hal-00136204/en/.

[29] F. SUTER. *Scheduling Delta-Critical Tasks in Mixed-Parallel Applications on a National Grid*, in "8th IEEE/ACM International Conference on Grid Computing (Grid 2007), Austin, TX", IEEE, September 2007, p. 2-9, http://hal.inria.fr/inria-00165868/en/.

Internal Reports

[30] S. CANZAR, E. ALTHAUS. *Solving the extended pairwise alignment problem efficiently*, Technical report, n^{o} MPI-I-2007-1-002, Max-Planck-Institut für Informatik, May 2007, http://hal.archives-ouvertes.fr/hal-00177235/en/.

[31] J. GUSTEDT. *Sublinear Communication for Integer Permutations*, Technical report, n^{o} RR-6403, INRIA, December 2007, http://hal.inria.fr/inria-00201503/en/.

[32] F. SUTER, H. CASANOVA. *Extracting Synthetic Multi-Cluster Platform Configurations from Grid'5000 for Driving Simulation Experiments*, technical report, n^{o} RT-0341, INRIA, 2007, http://hal.inria.fr/inria-00166181/en/.

References in notes

[33] H. CASANOVA, F. DESPREZ, F. SUTER. *From Heterogeneous Task Scheduling to Heterogeneous Mixed Parallel Scheduling*, in "Proceedings of the 10th International Euro-Par Conference (Euro-Par'04), Pisa, Italy", M. DANELUTTO, D. LAFORENZA, M. VANNESCHI (editors), Lecture Notes in Computer Science, vol. 3149, Springer, August/September 2004, p. 230–237.

[34] H. CASANOVA, A. LEGRAND, L. MARCHAL. *Scheduling Distributed Applications: the SimGrid Simulation Framework*, in "Proceedings of the third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)", may 2003.

[35] P.-N. CLAUSS. *Algorithme à front d'onde et pipeline out-of-core sur architecture à mémoire partagée*, Technical report, Université Henri Poincaré - Nancy I, June 2006.

[36] T. H. CORMEN, M. T. GOODRICH. *A Bridging Model for Parallel Computation, Communication, and I/O*, in "ACM Computing Surveys", vol. 28A, n^{o} 4, 1996.

[37] D. CULLER, R. KARP, D. PATTERSON, A. SAHAY, K. SCHAUSER, E. SANTOS, R. SUBRAMONIAN, T. VON EICKEN. *LogP: Towards a Realistic Model of Parallel Computation*, in "Proceeding of 4-th ACM SIGPLAN Symp. on Principles and Practises of Parallel Programming", 1993, p. 1-12.

[38] F. DEHNE, W. DITTRICH, D. HUTCHINSON. *Efficient external memory algorithms by simulating coarsegrained parallel algorithms*, in "ACM Symposium on Parallel Algorithms and Architectures", 1997, p. 106-115.

[39] F. DEHNE, A. FABRI, A. RAU-CHAPLIN. *Scalable parallel computational geometry for coarse grained multicomputers*, in "International Journal on Computational Geometry", vol. 6, n^{o} 3, 1996, p. 379-400.

[40] K. FUJIWARA, H. CASANOVA. *Speed and Accuracy of Network Simulation in the SimGrid Framework*, in "First International Workshop on Network Simulation Tools (NSTools), Nantes, France", October 2007.

[41] J. GUSTEDT, J. A. TELLE. *A work-optimal coarse-grained PRAM algorithm for Lexicographically First Maximal Independent Set*, in "Italian Conference on Theoretical Computer Science - ICTCS'03, Bertinoro, Italy", C. BLUNDO, C. LANEVE (editors), Lecture notes in Computer Science, vol. 2841, Springer, EATCS, October 2003, p. 125-136.

[42] A. HARBAOUI. *Étude comparative des algorithmes de découverte de la topologie de la grille*, Technical report, LORIA, 2006.

[43] T. N'TAKPÉ, F. SUTER. *Critical Path and Area Based Scheduling of Parallel Task Graphs on Heterogeneous Platforms*, in "Proceedings of the Twelfth International Conference on Parallel and Distributed Systems (ICPADS), Minneapolis, MN", July 2006.

[44] L. G. VALIANT. *A bridging model for parallel computation*, in "Communications of the ACM", vol. 33, n^{o} 8, 1990, p. 103-111.

[45] R. WOLSKI, N. SPRING, J. HAYES. *The NWS: A Distributed Resource Performance Forecasting Service for Metacomputing*, in "Future Generation Computing Systems, Metacomputing Issue", vol. 15, n^{o} 5–6, 1999, p. 757–768.