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

Research Scientist

Jens Gustedt [research director, INRIA, team leader, HdR]
Emmanuel Jeannot [INRIA, HdR]
Stéphane Genaud [INRIA, on leave from Univ. Strasbourg]

Faculty Member

Sylvain Contassot-Vivier [professor, UHP, HdR]
Martin Quinson [assistant professor, UHP/ÉSIAL]
Frédéric Suter [assistant professor, UHP, until Sep. 30, 2008]
Stéphane Vialle [professor, SUPÉLEC Metz Campus, HdR]

Technical Staff

Malek Cherier [engineer, INRIA, until Sep 30, 2008]

PhD Student

Louis-Claude Canon [MENESR grant, since Oct 1, 2007]
Stefan Canzar [joint thesis with UdS, Saarbrücken, Germany]
Pierre-Nicolas Clauss [MENESR grant, since Oct 1, 2006]
Soumeya Leila Hernane [teaching assistant UST Oran, Algeria, since Oct 1, 2007]
Constantinos Makassikis [SUPÉLEC, since Feb 1, 2007]
Vassil Iordanov [CHI Systems (Philadelphia - USA) and SUPÉLEC, since March 1, 2008]
Cristian Rosa [ANR project grant, since Nov 10, 2008]
Tchimou N'takpé [joint regional/Côte d'Ivoire government grant since Oct 1, 2005]

Administrative Assistant

Roxane Auclair [INRIA and CNRS]

2. Overall Objectives

2.1. Overall Objectives

Keywords: *Grid computing*, *algorithms*, *data distribution*, *data redistribution*, *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.

– Experimentation methodology: 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 [55]. Their common characteristics are:

- Maximally exploit the data that is located on a particular node by a local computation.
- Collect all requests for other nodes during the computation.
- 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) [55], LogGP (Latency overhead gap Procs) [47], CGM (Coarse Grained Multicomputer) [49] 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 have the following properties:

1. It should use asynchronous algorithmic schemes when possible. Those algorithms are very well suited to grid contexts but are not applicable to all scientific problems.
2. Otherwise, be guided by the idea of **supersteps** (BSP). This is to enforce a concentration of the computation to the local data.
3. Ensure an economic use of all available resources.

At the same time, we have to be careful that the model (and the design of algorithms) remains simple.

Several studies have demonstrated the efficiency of (1) in large scale local or grid contexts [43], [42] or have dealt with the implementation aspects [44]. But to fully exploit the benefits of those algorithms, not only the computations need to be asynchronous but also the controls of those algorithms. To fulfill such needs, a decentralized convergence detection algorithm has been proposed in [17].

Concerning (2), 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 (2) to combine these objectives for homogeneous environments has been given in [5] with PRO.

In a complementary approach we have addressed (3) 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, 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. Experimentation Methodology

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

Participants: Sabina Akhtar, Malek Cherier, Emmanuel Jeannot, Martin Quinson, Cristian Rosa.

Experimental validation is an important issue for the research on complex systems such as grids. It constitutes a scientific challenge by itself since we have to validate simulation and emulation 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 is particularly challenging for grid systems. Such systems are 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 is 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.

Several kind of experiments are to be run in computer science. The most common in the grid computing scientific community are meant to compare the performance of several algorithms or implementations. It is the classical way to assess the improvement of some newly proposed work over the state of the art. But because of the complexity of grid systems, testing the effectiveness of a given algorithm (whether it is deadlock-free for instance) becomes also a compelling challenge, which must be addressed specifically.

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, Sylvain Contassot-Vivier, 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 SMP s).
- 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.

Asynchronous algorithms are very good candidates as they are robust to dynamical variations of the performances of the interconnection network used. Moreover, they are even tolerant to the loss of message related to the computations. However, as mentioned before they cannot be used in all cases. We will then focus on the feasibility to modify those schemes in order to widen their range of applicability while preserving a maximum of asynchronism.

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].

4.1.2. External Memory Computation

In the mid-nineties several authors [46], [48] 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*, see Section 5.1, permitted 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.

4.1.4. Large Scale Computing

Within the context of a SUPÉLEC and EDF-R&D collaboration, we have developed a distribution of a stochastic control algorithm based on Dynamic Programming, which has been successively applied to large problem on large scale architectures.

Since 1957, Dynamic Programming has been extensively used in the field of stochastic optimization. The success of this approach is due to the fact that its implementation by backward recursion is very easy. The main drawback of this method is due to the number of actions and the number of state control to test at each time step. In order to tackle this problem other methods are described in the literature, but either they require convexity of the underlying function to optimize or they are not suitable for large multi-step optimizations. Therefore, we have parallelized the stochastic control algorithm based on dynamic programming as it is used by EDF.

From a parallel computing point of view, the main difficulty has been to redistribute data and computations at each step of the algorithm efficiently. Our parallel algorithm has been successfully implemented and experimented on multi-core PC clusters (up to 256 nodes and 512 cores) and on Blue Gene/L and Blue Gene/P supercomputers (using up to 8192 nodes and 32768 cores). Furthermore, a strong collaboration with IBM allowed to implement many serial optimizations and help to decrease the execution times significantly, both on PC clusters and on Glue Gene architecture.

The Stochastic Dynamic Programming method is usually thought to be limited to problems with less than 3 or 4 state variables involved. But our parallel version currently allows to optimize an electricity asset management problem with 7-energy-stocks and 10-state-variables and still achieves both speedup and size-up. A larger industrial use-case with more than 10 stocks is under development. A future development on large clusters of GPUs is scheduled to be able to process quickly more than 20 stocks.

4.1.5. Heterogeneous Architecture Programming

Main Monte-Carlo simulations are independent computations and are thus well adapted to distributed computing on clusters and on hardware accelerators like GPUs. The aim of this research was to design and experiment a Monte-Carlo simulation on a cluster of GPUs, and to compare this solution to a more classical cluster of CPUs.

This research started in the context of a collaboration between researchers from SUPÉLEC, CERMICS and CERTIS laboratories of ENPC and the MathFi INRIA project. We have designed a general path-dependent exotic European pricing for clusters of CPUs and for clusters of GPUs. Several technical and mathematic problems appeared and finally we achieved the following results:

- We have designed a highly parallel RNG, adapted to large parallel and distributed architectures, and we obtained similar results with a cluster of GPUs using simple precision and with a cluster of CPUs using double precision.
- We have designed a mixed coarse and fine grain parallelization of Monte-Carlo simulations, using MPI and OpenMP on CPU cluster, or MPI and CUDA on GPU cluster. This strategy is efficient and scales up to 256×2 CPU cores and up to 16×112 GPU cores.
- Our experimentations has shown the execution time and energy consumption of Monte-Carlo simulations can be both efficiently reduced when using GPU clusters in place of pure CPU clusters.

Next steps of this research will consist in exploiting asynchronous computing and communication capabilities of GPU-CPU nodes.

In parallel, in collaboration between EDF and SUPÉLEC we investigate the design of a unified framework based on generic programming to achieve a development environment adapted both to multi-core CPUs and GPUs, for neutronic computations. We plan to include our algorithmic solutions about Monte-Carlo simulations on GPUs in this development environment.

4.2. Evolution of Scheduling Policies and Network Protocols

Participants: Emmanuel Jeannot, Tchimou N'takpé, 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

If two kinds of parallelism (*task* and *data* parallelisms) are typically exploited in scientific applications, 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 interdependent data-parallel tasks to be executed concurrently.

In this context 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 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. This scheduling problem being NP-complete, several scheduling heuristics were designed, most of them targeting homogeneous computing environment.

While homogeneous platforms are relevant to many real-world scenarios, heterogeneous platforms, such as multi-clusters are becoming increasingly common and powerful. Mixed-parallel applications appear then ideally positioned to take advantage of such large-scale platforms. 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: Sabina Akhtar, Sylvain Contassot-Vivier, Jens Gustedt, Emmanuel Jeannot, Martin Quinson, Cristian Rosa, 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 is one of the main simulation tools used in the Grid Computing community.

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. Use of Formal Methods to Assess Distributed Algorithms

In joint research with Stephan Merz of the Mosel team of INRIA Nancy and LORIA, we are interested in the verification (essentially via model checking) of distributed and peer-to-peer algorithms. Whereas model checking is now routinely used for concurrent and embedded systems, existing algorithms and tools can rarely be effectively applied for the verification of asynchronous distributed algorithms and systems. Our goal is to adapt these methods to our context.

4.3.4. Aladdin-G5K

The Aladdin-G5K intiative is an action funded by INRIA that ensures the sustainability of the Grid'5000 platform.

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.

4.3.5. InterCell

Intercell aims at setting up a cluster (256 PCs) for interactive fine grain computation. It is granted by the Lorraine Region (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 [51].

4.3.6. Experimental cluster of GPUs

The GPELEC cluster is a 16 node cluster of GPUs and designed for computer science experimentation. It has been granted and bought by SUPÉLEC. Each node is a PC hosting a dual-core CPU and a GPU card: a nVIDIA GeForce 8800 GT, with 512MiB of RAM (on the GPU card). The 16 nodes are interconnected across a devoted Gigabit Ethernet switch. An Infiniband network is also available on half of the GPELEC cluster (on 8 nodes). Some Wattmeters have been installed on the GPELEC cluster (nodes and switches) in order to measure and analyse the energetic consumption, function of the computations run. Development environment available on GPELEC are mainly the gcc suite and its OpenMP library, OpenMPI and the CUDA environment of nVIDIA (nvcc compiler).

The objective of GPELEC platform was to quickly provide an experimental GPU cluster to researchers of SUPÉLEC and AlGorille in order to experiment scientific programming on GPU ("GPGPU"), and to track computing and energetic performances. In 2008 GPELEC has allowed to experiment the compatibility of MPI and CUDA frameworks, and to develop some fast Monte-Carlo simulations for an option pricing problem. Others developments and experimentations are planned in 2009 in collaboration with EDF researchers, and with our colleagues from CERMICS and MathFi INRIA team.

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,
- 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. Wrekavoc

Participants: 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. Therefore, Wrekavoc helps to validate parallel and distributed applications and algorithms. Moreover, on the modeling side, it helps to under- stand the impact of platform parameters (latency, bandwidth, CPU speed, mem- ory) on application performance.

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.

5.3. SimGrid

Participants: Malek Cherier, Martin Quinson.

The SIMGrid *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 [52].

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 one.

SimDag eases the study of scheduling heuristics for DAGs of (parallel) tasks, which 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/.

5.4. 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, Martin Quinson, 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 [51]. 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, and to optimize the electricity asset management. A stochastic control algorithm which is successively applied to a 1-dimensional problem: a gas storage valuation, and extended to a N-dimensional problem: an electricity asset management, has been distributed and evaluated on large scale problems and machines in 2008. The initial 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. Then, some direct collaborations between SUPÉLEC and EDF R&D allowed to achieve the distributed N-dimensional version of the stochastic control algorithm.

Finally, this algorithm has been successfully implemented and experimented on PC clusters (up to 256 nodes and 512 cores) and on Blue Gene architecture: on a Blue Gene/L (up to 4096 nodes and 8192 cores) and on a Blue Gene/P (up to 8192 nodes and 32768 cores, rank 13 in Top500 in first semester 2008). The designed distribution allows to run gas storage valuation models which require considerable amounts of computational power and memory space, and to successfully optimize an electricity asset management problem with 7-energy-stocks and 10-state-variables while achieving both speedup and size-up.

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 different *pricing models* have been successfully plugged into our generic and distributed skeleton of stochastic control algorithm and experimented.

The distributed applications has been implemented in C++ with MPI message passing library and with OpenMP or Intel TBB multithreading libraries. The distributed and parallel codes have reached very good performances on PC clusters and on Blue Gene architecture (with very good scaling). The 1-dimensional problem of gas storage valuation has reached a speedup close to 407 on 512 processors and 680 on 1024 processors of the Blue Gene supercomputer, running the "*normal inverse Gaussian*" pricing 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. The N-dimensional problem of electricity asset management runs ans scales up to 512 cores on PC cluster installed at SUPÉLEC and up to 32768 cores on the Blue Gene/P supercomputer of EDF. Then a strong collaboration with IBM engineers allowed to implement complex serial optimizations and to continue to improve performances. The new version, cumulating the work of EDF, SUEPELEC and IBM is under test and evaluation. It currently allows to process a benchmark of electricity asset management with 7-energy-stocks and 10-state-variables in less than 1h on a 512 cores PC cluster.

However, 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. Large Scale Models and Algorithms for Random Structures

For testing and benchmarking the generation of large random input data with known probability distributions is crucial. In [14], 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 [24].

In a new approach, we use random integer permutations for a new approach to parallel random number generation. We use a small amount of real random bits that are supposed to be available on each processor to seed the generation process on each processor. Other than commonly used approaches we don't use these random bits to just chose a different entry point into the same state space of the random generator. Instead, we use it to construct substantially different state spaces that depend on the random bits that are invested. By that we are able to guarantee that the generation on different processors (or just different runs) are completely independent. We report on experiments that prove the quality of the random process as well as its competitiveness in terms of computing resources. A report that details is work is currently in preparation.

In another stream of research we investigated the generation of random graphs. Such a generation of realistic graphs is crucial as an input for testing large scale algorithms, theoretical graph algorithms as well as network algorithms, e.g our platform generator in Section 6.2.

Commonly used techniques for the random generation of graphs have two disadvantages, namely their lack of bias with respect to history of the evolution of the graph, and their incapability to produce families of graphs with non-vanishing prescribed clustering coefficient. In this work we propose a model for the genesis of graphs that tackles these two issues. When translated into random generation procedures it generalizes well-known procedures such as those of Erdős & Rény and Barabási & Albert. When just seen as composition schemes

for graphs they generalize the perfect elimination schemes of chordal graphs. The model iteratively adds so-called *contexts* that introduce an explicit dependency to the previous evolution of the graph. Thereby they reflect a historical bias during this evolution that goes beyond the simple degree constraint of preference edge attachment. Fixing certain simple statical quantities during the genesis leads to families of random graphs with a clustering coefficient that can be bounded away from zero. A first report on the model is found in [38]; an internship, [39], showed promising experimental results.

6.1.4. New Control and Data Structures for Efficiently Overlapping Computations, Communications and 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. To have an efficient control of such algorithms, efficient control structures are needed that regulate the concurrent access to resources.

Using the *parXXL* library we were able to propose an architecture-independent out-of-core implementation of a well known hydrodynamics kernel, see [45]. 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 is published in [22].

Mutual exclusion is one of the classical problems of distributed computing. Several solutions have been devised in the literature, but most of them remain relatively far from practitioner needs. This concern first of all distributed platforms, on which are particularly difficult to control and to certify. We proposed an extension to the classical Naimi-Trehel algorithm to allow partial locks on a given data. Such ranged locks offer a semantic close to the POSIX file locking, where each thread lock the subpart of the file it is working on, in the hope that this work can prove useful to high performance computing practitioner [31].

Further, we introduced the framework of *ordered read-write locks*, ORWL, that are characterized by two main features: a strict FIFO policy for access and the attribution of access to *lock-handles* instead of processes or threads. These two properties allow applications to have a controlled pro-active access to resources and thereby to achieve a high degree of asynchronicity between different tasks of the same application. For the case of iterative computations with many parallel tasks which access their resources in a cyclic pattern we provide a generic technique to implement them by means of ORWL. It was shown that the possible execution patterns for such a system correspond to a combinatorial lattice structure and that this lattice is finite iff the configuration contains a potential deadlock. In addition, we provide efficient algorithms: one that allows for a deadlock-free initialization of such a system and another one for the detection of deadlocks in an already initialized system. For first results see [35].

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 thesis of Stefan Canzar [11], a collaboration with the MPI Saarbrücken that predates the INRIA project creation of AlGorille, has been defended in Nov. 2008. It is devoted to two NP-complete combinatorial optimization problems arising in bioinformatics, the well-studied multiple sequence alignment problem and the new formulated interval constrained coloring problem. It shows that advanced mathematical programming techniques are capable of solving large scale real-world instances from Biology to optimality. Furthermore, it reveals alternative methods that provide approximate solutions.

6.2. Transparent Resource Management

Participants: Louis-Claude Canon, Stéphane Genaud, Emmanuel Jeannot, Tchimou N'takpé, 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 studied the problem of scheduling independent tasks on a set of related processors which have a probability of failure governed by an exponential law. We are interested in the bi-objective analysis, namely simultaneous optimization of the makespan and the reliability. We show that this problem cannot be approximated by a single schedule. A similar problem has already been studied leading to a $\langle \overline{2}, 1 \rangle$-approximation algorithm (*i.e.,*for any fixed value of the makespan, the obtained solution is optimal on the reliability and no more than twice the given makespan). We have provided an algorithm which has a much lower complexity. This algorithm was finally used to derive a $(2+\epsilon, 1)$-approximation of the Pareto set of the problem, for any $\epsilon > 0$, see [28].

We have also worked in the case where jobs can fail. More precisely, we have studied the problem of random brokering on platforms directly inspired from existing grids such as EGEE. In such environments incoming jobs are randomly dispatched to computational elements with a probability proportional to the accumulated speed of this element. We have studied three strategies to improve the reliability (namely local resubmission, global resubmission and duplication). For each heuristic we are able to model the saturation ratio (when the incoming load exceeds the maximum throughput of the environment), the average waiting time of the jobs and the probability of success of each job. Our experiments show that the proposed models are very realistic. For each of the above metric the models usually predict a very precise value. Our experiments show that, on the average, the global resubmission is the best strategy. The global resubmission outperforms, in almost every case, the local resubmission. The comparison also shows that in very few cases the duplication is the best strategy (very large platform with few submitted jobs) and, even in this case the average gain against the global resubmission is not very high and this gain degrades with the number of submitted jobs discarding its use in steady state.

6.2.2. Robust Scheduling

A schedule is said robust if it is able to absorb some degrees of uncertainty in tasks duration while maintaining a stable solution. We are using probablisitic models to compute the robustness of a schedule. Our main matric is the standard deviation of this distribution (for the robustness) and the average (for the duration/makespan). We have worked on optimizing both criteria.

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, see [34], [19].

We have studied the robustness of 20 static makespan-centric DAG scheduling heuristics from the literature, using as a metric for robustness the standard deviation of the makespan over a large number of measurements.

Our results are three-fold. First, we have shown that it is better to respect the static order of the tasks on the processors than to change this order dynamically. Second, we have shown that robustness and makespan are somehow correlated: schedules that perform well statically tend to be the most robust. Third, we have shown that, for the cases we have studied, heuristics such as HEFT, HBMCT, GDL, PCT, are among the best for both makespan and robustness, see [20].

6.2.3. Parallel Task Scheduling

This year we conducted a comparison of a pragmatic two-step scheduling heuristic with an original guaranteed algorithm called MCGAS on almost homogeneous multi-cluster platforms[13]. 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. This guarantee provided by MCGAS is tunable via two parameters. We have determined the values of these parameters that lead to the tightest performance guarantee both analytically and in practice. While having a performance guarantee is always desirable, it does not mean that the algorithm leads to good average performance in practice. Our key finding, however, was that MCGAS outperforms the non-guaranteed algorithm on average over a large range of application configurations.

We also studied the scheduling of multiple Parallel Task Graphs onto a shared heterogeneous platform. We followed an approach imposing a resource constraint to each application to build its schedule, see [40]. Depending on how this constraint is determined we obtain different tradeoffs between the overall completion time and the fairness of the produced schedules.

In collaboration with Sascha Hunold and Thomas Rauber at University of Bayreuth we have revisited the problem of scheduling dynamically generated Parallel Task Graphs and presented a novel algorithm, called DMHEFT, for scheduling such applications onto a heterogeneous collection of clusters, see [26]. The scheduling decisions are based on the predicted runtime of a modable task as well as the estimation of the redistribution costs between data-dependent tasks. The algorithm also takes care of unfavorable placements of tasks by considering the postponing of ready tasks even if idle processors are available. We have also addressed a potential drawback of two-step scheduling algorithms, see [25]. The two steps, allocation and mapping, are generally fully decoupled. This separation can induce unnecessary or costly data redistributions that have an impact on the overall performance. This is particularly true for data intensive applications. We thus proposed an original approach in which the allocations determined in the first step can be adapted during the second step in order to minimize the impact of these data redistributions. Two redistribution aware mapping strategies were developed and studied with regard to their impact on the schedule length over a broad range of experimental scenarios.

Finally Tchimou N'takpé wrote his PhD dissertation that summarizes his different contributions on the scheduling of Parallel Task Graphs on homogeneous, heterogeneous and shared platforms. His defense is scheduled in early January 2009.

6.2.4. Analysis of the AllToAllV in cluster of cluster environements

The context of this work is the total exchange of data between processes running in different clusters linked by a wide-area link (backbone). We can experiment such an environment using several sites of the Grid5000 testbed. Starting from the work done on the `AllToAll` operation [54], we have extended the study to the `AllToAllV` operation. `AllToAll` in the MPI terminology means a total exchange of pieces of data of the same size between participant processes. `AllToAllV` implies that each piece of data may be of a different size. Unlike `AllToAll`, it is difficult to choose a good routing algorithm (e.g binomial tree, Bruck algorithm) depending on the data size for `AllToAllV` since the processes do not know the amount of data sent by the others. MPI implementations (OpenMPI, MPICH-2, GridMPI) have today a very simple implementation for `AllToAllV`, in which all processes send and receive asynchronously their data to all other processes and wait for the completion of all communications. Yet simple, this strategy performs better than any optimized routing scheme used in the other collective operations. We have tried more sophitiscated approaches based on message aggregation, congestion regulation and load-balance. In this kind of strategy, we select forwarders processes at each cluster, whose task is to aggregate single messages into a larger one sent over the backbone. The number of forwarders allows to control how many TCP streams simultaneously compete for the backbone. We can balance the size of the aggregated messages with an extra step at the begining where process first exchange the information about how much data they send. However, our proposal have only equaled the original `AllToAllV` implementation so far, bringing no significant improvement. One of the key finding of this work is that message aggregation does not improve the overall throughtput of the streams over the wide area link. Work is currently ongoing to model the behavior of the various routing strategies in the PLogP model.

6.3. Experimentation Methodology

Participants: Sabina Akhtar, Malek Cherier, Emmanuel Jeannot, Jens Gustedt, Martin Quinson, Cristian Rosa, Frédéric Suter.

6.3.1. Improvement of the SimGrid tool

This year, we pursued our work on stabilizing SIMGrid in order to make the tool as usable as possible. For that, we set up a automatic build system allowing to enforce the software quality. In order to increase the user community of SIMGrid, we also added C++ bindings in addition to the already existing Java ones.

The main goal of this year's work was to improve the tool scalability. We intend to make it usable not only by the grid computing community, but also by the peer-to-peer scientists. A new platform representation [37] (specified with XML) allows to represent the Grid'5000 platform in 80Kb instead of previously 550Mb. We spotted and fixed some performance bottlenecks resulting in a speedup of two order of magnitude for some scenarios. We are also principal investigator of an ANR project accepted this year aiming at allowing P2P simulation using SIMGrid (see 7.2.4).

SIMGrid is freely downloadable, see [21], and its user base is rapidly growing. It grounded the experimental section of almost fifty scientific publications (two third of them from users not being part of the core team).

6.3.2. A Platform Description Archive for Reproducible Simulation Experiments

Simulation is a common approach in parallel and distributed computing to explore various experimental scenarios in a reproducible way and in a reasonable amount of time. This argument of reproducibility is often limited to the authors of a simulator as they rarely give access to the platform and applications they use in their articles.

This year we focused on aspects related to the platform configurations used in simulation. We proposed the Platform Description Archive (PDA) which is an effort to make platform descriptions and tools available to users of simulation toolkits. Our main contributions are the following: (i) a tool to generate the platform descriptions needed for individual publications; (ii) a public archive to store the experimental settings of published works to help improving the community methodology.

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. This must contain not only quantitative information, such as the one provided by NWS [56], but also qualitative ones about the interconnecting graph topology.

This year, we continued our work on the ALNeM framework, building upon [4], but this has not yet lead to any new publication.

6.3.4. Formal Verification of Distributed Algorithms

As stated before, distributed algorithms can get challenging to assess and debug. Whereas formal verification in general and model checking in particular is now routinely used for concurrent and embedded systems, existing algorithms and tools can rarely be effectively applied for the verification of asynchronous distributed algorithms and systems. In joint research with Stephan Merz of the Mosel team of INRIA Nancy and LORIA, we have started to explore two approaches to address this problem.

In a first approach, Sabina Akhtar in her Master thesis [41] investigated an extension of the +CAL language [53] defined by Leslie Lamport. The extension is intended for describing and verifying models of distributed algorithms, whereas the original language is geared towards shared-memory concurrent programming. In collaboration with Martin Quinson, Sabina Akhtar continues this research as a PhD student in the MOSEL team.

In a complementary approach, we are interested in adding model checking support to the SIMGrid framework and specifically to the GRAS API for simulating and implementing Grid algorithms. During his INRIA summer internship, Cristian Rosa explored this possibility and implemented a first prototype. During his PhD thesis supervised by Martin Quinson and Stephan Merz, Cristian Rosa will study dedicated model checking techniques and algorithms for this platform. In contrast to similar projects that aim at verification of essentially unrestricted C programs, we believe that significant optimizations are possible due to the clearly specified programming model of SimGrid and GRAS.

6.3.5. Wrekavoc

The main work has been done on debugging and testing the features implemented last year. The memoire [50] of last year's engineering internship has been defended this year and a publication, [36], has been prepared.

6.3.6. Aladdin-G5K

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 GiB 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 GiB 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 have worked on extending the platform by purchasing a new cluster that will arrive in early 2009 (92 Xeon compute node, 16 GiB 8 cores, with infiniband interconnect 20 Gbits)

7. Other Grants and Activities

7.1. Regional initiatives

7.1.1. Lorraine Regional Council

We received a grant from the Lorraine Region for two years to fund our exploratory work on the possibility to use formal methods such as model-checking to ensure some properties (such as the lack of deadlocks in any case) of large-scale distributed algorithms (see 6.3.4).

7.2. National Initiatives

7.2.1. INRIA ADT

Aladdin (A LArge-scale Distributed Deployable INfrastructure) is an INRIA Technological Development Action. It is a management structure for Grid'5000. We participate in this initiative. The goal of this action is to ensure the developement of Grid'5000 by providing funding for engineers and training and some parts of the hardware.

7.2.2. CNRS initiatives, GDR-ASR and specific initiatives

We participate at numerous national initiatives. In the GDR-ASR (architecture, systems, and networks) we take part in RGE action [1]. The finances of RGE, led by Stéphane Vialle at SUPÉLEC, are provided by the GDR ASR of CNRS and maintained by AlGorille. The RGE action organizes three meetings per year, and usually gathers 40-45 people per meeting.

We also participate to the animation of the GDR-ASR as a whole.

7.2.3. 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), called ALPAGE.

It aims at constituting a bridge for large-scale platforms's algorithmic challenges between the parallel algorithms community, traditionally more focused on heterogeneity and large amounts of data, and distributed systems community, traditionally more focused on scalability and fault-tolerance issues.

Our contribution to this project is to work on grid platform discovery (see 6.3.3) in order to ease the problem instantiation for the other researchers of the group.

[1] *Réseau Grand Est*

7.2.4. ANR Initiatives

We are leader of one project of the ARPEGE call from the ANR (french funding agency), called USS-SimGrid (Ultra Scalable Simulation with SimGrid). It aims at improving the scalability of the SIMGrid simulator to allow its use in Peer-to-Peer research in addition of Grid Computing research. The challenges to tackle include models being more scalable at the eventual price of slightly reduced accuracy, automatic instantiation of these models, tools to conduct experiments campaigns, as well as a partial parallelization of the simulator tool.

As project leader, we are involved in most parts of this project, which should allow to improve our tool even further and set it as the reference in its domain.

7.3. European Initiatives

7.3.1. COST Action IC0805

We are the leader of the COST (European Cooperation in the field of Scientific and Technical Research) Action IC0805. The goal of the Action is to establish a European research network focused on high performance heterogeneous computing in order to address the whole range of challenges posed by these new platforms including models, algorithms, programming tools and applications.

7.3.2. NoE CoreGrid

We participate in the NoE "*CoreGrid*" led by Thierry Priol from INRIA Rennes, more precisely of the work package 6 on scheduling. Emmanuel Jeannot is the leader for CNRS of task 6.5: evaluation and benchmarking.

7.3.3. Bilateral Collaborations

We maintain several European collaborations with other research teams. We work with Vandy Berten and Joël Goossens of the Université Libre de Bruxelles on scheduling problems under stochastic models. We work with the team led by Thomas Rauber (University of Bayreuth, Germany) on parallel task graph scheduling.

7.4. International Initiatives

7.4.1. 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 Jon Weissman (University of Minnesota, Twin Cities) on scheduling with uncertainty.

8. Dissemination

8.1. Dissemination

8.1.1. Leadership within the Scientific Community

Stéphane Vialle is the leader of the RGE action (*Réseau Grand Est*) in the ASR GDR of CNRS.

Emmanuel Jeannot is member of the steering committee and the direction comittee of the ADT Aladdin-G5K and head of the Nancy site. Martin Quinson is serving as vice-head for the Grid'5000 project. Stéphane Genaud is also member of this Action.

8.1.2. Scientific Expertise

In 2008, Jens Gustedt was a referee and member of the thesis committee of Jean-Baptiste Ernst-Desmullier, University of Franche-Comté, France, and as well as for the thesis of Sascha Hunold, University of Bayreuth, Germany.

In 2008, Emmanuel Jeannot was referee and member of the thesis committee of Johnatan Eliabeth Pecero Sanchez, INP Grenoble; Laurent Bobelin (Université Aix-Marseille II) and Bernard Miegemolle (INSA Toulouse). Emmanuel Jeannot was also member of the thesis committee of Hinde Bouziane (University of Rennes 1) and Jean-François Pineau (ENS-Lyon).

In 2008, Martin Quinson also participated in the committee of Ernst-Desmullier, see above, and referee and member of the thesis committee of Cyril Briquet, University of Liège, Belgium.

Emmanuel Jeannot is member of the COST-GTAI of INRIA that selects and supervises INRIA Research Cooperative Actions (ARC).

In 2008, Sylvain Contassot-Vivier was a member of the thesis committee of Jérémy Fix (University Henri Poincaré - Nancy 1).

In 2008, Stéphane Vialle was a referee and member of the thesis committee of Olivier Hoenen, University Louis Pasteur of Strasbourg, France, and as well as for the thesis of Alexandre Chariot, ENPC (*Ecole des Ponts ParisTech*) & University of Paris-Est, France.

8.1.3. Teaching Activities

Frédéric Suter is teaching "*Algorithmique et programmation*" (L1) at University Henri Poincaré - Nancy 1.

Martin Quinson is teaching the following modules at ÉSIAL (University Henri Poincaré - Nancy 1): "*C et Shell*" (1A), "*Réseaux et systèmes*", "*Réseaux et systèmes avancés*" (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 University Henri Poincaré - Nancy 1. He is also responsible of the specialization "*Système et Applications Distribuées*" of ÉSIAL.

Sylvain Contassot-Vivier is teaching "*Algorithmic and programmation*" (M1), "*Parallel computation*" (M1), "*Grids*" (M2), "*Statistical learning*" (M1) and "*Networks*" (M2) at the University Henri Poincaré - Nancy 1. He is also co-responsible of the professional master in "*Networks, Services and Mobility*" at the same university.

Stéphane Vialle is teaching "*Information Systems*" (M1) at SUPÉLEC at Paris, "*Parallel and Distributed Computing, and Computing Grids*" (M2) at SUPÉLEC at Metz, "*Distributed Applications*" (M2) at the Louis Pasteur University of Strasbourg, and "*Concurrent Applications: concepts and tools*" at CNAM (National French Institution for Adult Training Courses) in Lorraine. He is also responsible of the organisation of computer science teaching at CNAM in Lorraine.

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 2008, Jens Gustedt has served as program committee member of the 16th Euromicro International Conference on Parallel, Distributed and network-based Processing PDP 2009 and of the International Conference on High Performance Computing (HiPC 2008).

Martin Quinson was member of the program committee of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SimuTools'08).

Emmanuel Jeannot was program vice chair of The 9th IEEE/ACM International Conference on Grid Computing GRID 2008. Emmanuel Jeannot was member of the program committee of the High Performance Distributed Computing conference HPDC 2008 and the 8th IEEE International Symposium on Cluster Computing and the Grid CCGRID 2008.

Emmanuel Jeannot and Frédéric Suter were members of the program committee of the seventeenth Heterogeneity in Computing Workshop (HCW 2008).

Stéphane Genaud was member of the program committee of the 9th IEEE/ACM International Conference on Grid Computing Grid 2008, and of the first International Symposium on Grid and Distributed Computing GDC 2008.

Stéphane Vialle was one of the two Program Chairs of the 2nd International Workshop on Parallel and Distributed Computing in Finance (PDCoF'2009). Stéphane Vialle was also member of the International, Conference on Distributed Frameworks & Applications 2008 and of the program committee of the international conference on Architecture of Computing Systems (ARCS 2009).

8.1.5. Refereeing

In 2008, members of the team served as referees for the following journals and conferences:

Journals: Annals of Operations Research, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Neural Networks, International Journal of High Performance Computing, Journal of Grid Computing Journal of Parallel Distributed Computing, Parallel Computing, Software: Practice and Experience.

Conferences: Algotel 2008, ARCS 2009, DFmA 2008, EuroPar 2008, Grid 2008, HCW 2008, HPDC 2008, HiPC 2008, ICIC 2008, IPDPS 2009, PDCoF 2009, PDP 2009, SimuTools 2008, VecPar 2008, WSES 2008, SuperComputing 2008.

8.1.6. Invitation

Emmanuel Jeannot has been invited to present the article *Experimental Validation of Grid Algorithms: a Comparison of Methodologies* [27] at the High-Performance Grid Computing Workshop (HPGC 2008) at Miami, FL, USA.

Martin Quinson presented a tutorial called *Simulation for Large-Scale Distributed Computing Research* at the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'08 May 2008 at Lyon, France) and at the 9th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'08 December 2008 at Dunedin, New-Zealand).

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://www.inria.fr/rrrt/rr-5383.html.

[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, 2006, http://hal.inria.fr/inria-00108389.

Year Publications

Doctoral Dissertations and Habilitation Theses

[11] S. CANZAR. *Lagrangian Relaxaton. Solving NP-hard Problems via Combinatorial Optimization*, Ph. D. Thesis, Nancy Univ., France, and Univ. d. Saarlandes, Germany, 2008.

[12] T. N'TAKPÉ. *Ordonnancement de tâches parallèles sur plates-formes hétérogènes partagées*, to appear, Ph. D. Thesis, Nancy Univ., 2009.

Articles in International Peer-Reviewed Journal

[13] P.-F. DUTOT, T. N'TAKPÉ, F. SUTER, H. CASANOVA. *Scheduling Parallel Task Graphs on (Almost) Homogeneous Multi-cluster Platforms*, in "IEEE Transactions on Parallel and Distributed Systems", 2008, http://hal.inria.fr/inria-00347273/en/.

[14] J. GUSTEDT. *Efficient Sampling of Random Permutations*, in "Journal of Discrete Algorithms", vol. 6, 2008, p. 125-139, http://hal.inria.fr/inria-00000900/en/.

Articles in National Peer-Reviewed Journal

[15] T. N'TAKPÉ. *Heuristiques d'ordonnancement en deux étapes de graphes de tâches parallèles*, in "Technique et Science Informatiques (TSI)", vol. 28, 2009, p. 75-99, http://hal.inria.fr/inria-00125269/en/.

Invited Conferences

[16] S. VIALLE, X. WARIN, C. MAKASSIKIS, P. MERCIER. *Stochastic control optimization & simulation applied to energy management: From 1-D to N-D problem distributions, on clusters, supercomputers and Grids*, in "Grid@Mons conference, Belgique Mons", 2008, http://hal-supelec.archives-ouvertes.fr/hal-00291821/en/.

International Peer-Reviewed Conference/Proceedings

[17] J. BAHI, S. CONTASSOT-VIVIER, R. COUTURIER. *An efficient and robust decentralized algorithm for detecting the global convergence in asynchronous iterative algorithms*, in "8th International Meeting on High Performance Computing for Computational Science, VECPAR'08, France Toulouse", 2008, p. 251–264, http://hal.inria.fr/inria-00336515/en/.

[18] J. BAHI, S. CONTASSOT-VIVIER, M. SAUGET, A. VASSEUR. *A Parallel Incremental Learning Algorithm for Neural Networks with Fault Tolerance*, in "8th International Meeting on High Performance Computing for Computational Science, VECPAR'08, France Toulouse", 2008, p. 502–515, http://hal.inria.fr/inria-00336521/en/.

[19] L.-C. CANON, E. JEANNOT. *Scheduling Strategies for the Bicriteria Optimization of the Robustness and Makespan*, in "11th International Workshop on Nature Inspired Distributed Computing, États-Unis d'Amérique Miami", IEEE, 2008, p. 1-8, http://hal.inria.fr/inria-00333900/en/.

[20] L.-C. CANON, E. JEANNOT, R. SAKELARIOU, W. ZHENG. *Comparative Evaluation of the Robustness of DAG Scheduling Heuristics*, in "Integration Research in Grid Computing, CoreGRID integration workshop Grid Computing, Grèce Hersonissos, Crete", S. GORLATCH, P. FRAGOPOULO, T. PRIOL (editors), Crete University Press / Springer US, 2008, p. 63–74, http://hal.inria.fr/inria-00333904/en/.

[21] H. CASANOVA, A. LEGRAND, M. QUINSON. *SimGrid: a Generic Framework for Large-Scale Distributed Experiments*, in "10th IEEE International Conference on Computer Modeling and Simulation - EUROSIM / UKSIM 2008, Royaume-Uni Cambrige", IEEE, 2008, http://hal.inria.fr/inria-00260697/en/.

[22] 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, France Toulouse", J. BOURGEOIS, F. SPIES, D. E. BAZ (editors), IEEE, 2008, p. 293-300, http://hal.inria.fr/inria-00176084/en/.

[23] S. GENAUD, C. RATTANAPOKA. *Large-Scale Experiment of Co-allocation Strategies for Peer-to-Peer SuperComputing in P2P-MPI*, in "Fifth High-Performance Grid Computing Workshop in conjunction with 22nd IEEE International Parallel and Distributed Processing Symposium - IPDPS 2008, États-Unis d'Amérique Miami", IEEE Computer Society, 2008, http://hal.inria.fr/inria-00214137/en/.

[24] J. GUSTEDT. *Engineering Parallel In-Place Random Generation of Integer Permutations*, in "International Workshop on Experimental Algorithms, WEA 2008 Experimental Algorithms, 7th International Workshop, WEA 2008 LNCS, États-Unis d'Amérique Provincetown, MA", C. C. MCGEOCH (editor), LNCS, vol. 5038, Springer Verlag, 2008, p. 129-141, http://hal.inria.fr/inria-00312131/en/.

[25] S. HUNOLD, T. RAUBER, F. SUTER. *Redistribution Aware Two-Step Scheduling for Mixed-Parallel Applications*, in "IEEE International Conference on Cluster Computing - Cluster 2008, Japon Tsukuba", IEEE, 2008, p. 50 - 58, http://hal.inria.fr/inria-00329784/en/.

[26] S. HUNOLD, T. RAUBER, F. SUTER. *Scheduling Dynamic Workflows onto Clusters of Clusters using Postponing*, in "3rd International Workshop on Workflow Systems in e-Science (WSES 08), France Lyon", IEEE, 2008, p. 669-674, http://hal.inria.fr/inria-00329779/en/.

[27] E. JEANNOT. *Experimental Validation of Grid Algorithms: a Comparison of Methodologies*, in "Fifth High-Performance Grid Computing Workshop (HPGC 2008), in conjunction with IPDPS 2008, États-Unis d'Amérique Miami", IEEE, 2008, p. 1-8, http://hal.inria.fr/inria-00333898/en/.

[28] E. JEANNOT, E. SAULE, D. TRYSTRAM. *Bi-Objective Approximation Scheme for Makespan and Reliability Optimization on Uniform Parallel Machines*, in "The 14th International Euro-Par Conference on Parallel and Distributed Computing (Euro-Par 2008), Espagne Las Palmas de Gran Canaria", E. LUQUE, T. MARGALEF, D. BENÍTEZ (editors), Lecture Notes in Computer Science, vol. 5168, Springer Berlin / Heidelberg, 2008, p. 877-886, http://hal.inria.fr/inria-00333906/en/.

[29] C. MAKASSIKIS. *Distribution Large Echelle d'un Algorithme Financier de Contrôle Stochastique*, in "RenPar'18, Suisse Fribourg", 2008-02, p. Proceeding on CD-ROM (8 pages), http://hal-supelec.archives-ouvertes.fr/hal-00264911/en/.

[30] C. MAKASSIKIS, S. VIALLE, X. WARIN. *Large Scale Distribution of Stochastic Control Algorithms for Financial Applications*, in "PDCoF08, États-Unis d'Amérique Miami", 2008-04, 8 pages, http://hal-supelec.archives-ouvertes.fr/hal-00290440/en/.

[31] M. QUINSON, F. VERNIER. *Byte-Range Asynchronous Locking in Distributed Settings*, in "17th Euromicro International Conference on Parallel, Distributed and network-based Processing, Allemagne Weimar", 2009, http://hal.inria.fr/inria-00338189/en/.

[32] S. VIALLE, X. WARIN, P. MERCIER. *A N-dimensional Stochastic Control Algorithm for Electricity Asset Management on PC cluster and Blue Gene Supercomputer*, in "PARA 2008, Norvège Trondheim", 2008-05, 4 pages, http://hal-supelec.archives-ouvertes.fr/hal-00291814/en/.

Scientific Books (or Scientific Book chapters)

[33] C. GERMAIN-RENAUD, V. BRETON, P. CLARYSSE, B. DELHAY, Y. GAUDEAU, T. GLATARD, E. JEANNOT, Y. LEGRÉ, J. MONTAGNAT, J. MARIE MOUREAUX, A. OSORIO, X. PENNEC, J. SCHAERER, R. TEXIER. *Grid Analysis of Radiological Data*, in "Handbook of Research on Computational Grid Technologies for Life Sciences, Biomedicine and Healthcare", M. CANNATARO (editor), Information Science Reference, 2008, http://hal.inria.fr/inria-00334032/en/.

Research Reports

[34] L.-C. CANON, E. JEANNOT. *Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments*, RR-6476, Rapport de recherche, INRIA, 2008, http://hal.inria.fr/inria-00261376/en/.

[35] P.-N. CLAUSS, J. GUSTEDT. *Iterative Computations with Ordered Read-Write Locks*, RR-6685, Rapport de recherche, INRIA, 2008, http://hal.inria.fr/inria-00330024/en/.

[36] O. DUBUISSON, J. GUSTEDT, E. JEANNOT. *Multi-Site Emulation using Wrekavoc: Validating Distributed Algorithms and Applications*, RR-6696, Rapport de recherche, INRIA, 2008, http://hal.inria.fr/inria-00331627/en/.

[37] M.-E. FRINCU, M. QUINSON, F. SUTER. *Handling Very Large Platforms with the New SimGrid Platform Description Formalism*, RT-0348, Rapport Technique, INRIA, 2008, http://hal.inria.fr/inria-00256883/en/.

[38] J. GUSTEDT. *Generalized Attachment Models for the Genesis of Graphs with High Clustering Coefficient*, RR-6622, Rapport de recherche, INRIA, 2008, http://hal.inria.fr/inria-00312059/en/.

[39] J. GUSTEDT, P. SCHIMIT. *Numerical results for generalized attachment models for the genesis of graphs*, RT-0361, Rapport Technique, INRIA, 2008, http://hal.inria.fr/inria-00349461/en/.

[40] T. N'TAKPÉ, F. SUTER. *Concurrent Scheduling of Parallel Task Graphs on Multi-Clusters Using Constrained Resource Allocations*, RR-6774, Rapport de recherche, 2008, http://hal.inria.fr/inria-00347203/en/.

References in notes

[41] S. AKHTAR. *Formal Verification of Distributed Algorithms in +CAL 2.0*, Master Thesis, LORIA, 2008.

[42] J. BAHI, S. CONTASSOT-VIVIER, R. COUTURIER. *Asynchronism for Iterative Algorithms in a Global Computing Environment*, in "The 16th Annual International Symposium on High Performance Computing Systems and Applications (HPCS'2002), Moncton, Canada", June 2002, p. 90–97.

[43] J. BAHI, S. CONTASSOT-VIVIER, R. COUTURIER. *Coupling Dynamic Load Balancing with Asynchronism in Iterative Algorithms on the Computational Grid*, in "17th IEEE and ACM int. conf. on International Parallel and Distributed Processing Symposium, IPDPS 2003, Nice, France", IEEE computer society press, April 2003, 40a, 9 pages.

[44] J. BAHI, S. CONTASSOT-VIVIER, R. COUTURIER. *Performance comparison of parallel programming environments for implementing AIAC algorithms*, in "18th IEEE and ACM Int. Conf. on Parallel and Distributed Processing Symposium, IPDPS 2004, Santa Fe, USA", IEEE computer society press, April 2004, 247b, 8 pages.

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

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] O. DUBUISSON. *Système d'émulation multi-sites pour la validation d'algorithmes distribués*, mémoire d'ingénieur CNAM, Conservatoire National des Arts et Metiers, October 2008.

[51] N. FRESSENGEAS, H. FREZZA-BUET, J. GUSTEDT, 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, France Paris", IEEE, 2007, http://hal.inria.fr/inria-00139660/en/.

[52] 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.

[53] L. LAMPORT. *Checking a Multithreaded Algorithm with +CAL.*, in "20th Intl. Symp. Distributed Computing (DISC 2006)", Lecture Notes in Computer Science, vol. 4167, 2006, p. 151-163.

[54] 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/.

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

[56] 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.