INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team AlGorille

Algorithms for the Grid

Nancy - Grand Est

Theme : Distributed and High Performance Computing

Table of contents

1. Team

Research Scientist

Jens Gustedt [Senior Researcher, INRIA, team leader, HdR]

Faculty Members

Sylvain Contassot-Vivier [Professor, UHP, HdR]
Stéphane Genaud [Associate Professor, Univ. Strasbourg, HdR]
Lucas Nussbaum [Associate Professor, U. Nancy 2]
Martin Quinson [Associate Professor, UHP/ÉSIAL]
Stéphane Vialle [Professor, SUPÉLEC Metz Campus, HdR]

Technical Staff

Pierre-Nicolas Clauss [Post-doc ANR project grant]
Fekari El Mehdi [Engineer, INRIA]
Philippe Robert [engineer, INRIA, from Oct. 11, 2009 to Oct. 9, 2010]
Sébastien Badia [engineer, INRIA, since Jul. 15, 2010]
Christophe Thiéry [engineer ANR project grant, since Oct. 1, 2010]

PhD Students

Soumeya Leila Hernane [teaching assistant UST Oran, Algeria, since Oct 1, 2007]
Thomas Jost [since October, 2009]
Constantinos Makassikis [SUPÉLEC & UHP, since Feb 1, 2007]
Vassil Iordanov [NATO (The Hague, Netherlands) and SUPÉLEC, since March 1, 2008]
Cristian Rosa [ANR project grant, since Nov 10, 2008]
Wilfried Kirschenmann [EDF R&D (Clamart, France) and SUPÉLEC, since January 1, 2009]

Administrative Assistants

Antoinette Courrier [CNRS]
Laurence Félicité [universities]
Isabelle Herlich [INRIA]

Others

Tomasz Buchert [Poznan University of Technology, internship from March to September 2010]
Marion Guthmuller [Ecole Supérieure d'Informatique et Applications de Lorraine, internship from June to August 2010]
David Marquez [University of Buenos Aires, Argentina, internship from July to November 2010]
Mauricio Chimento [University of Rosario, Argentina, internship from April to June 2010]

2. Overall Objectives

2.1. Introduction

The possible access to distributed computing resources over the Internet allows a new type of applications 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, extensibility 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.

2.2. Highlights

During the mid-term evaluation, the ANR project USS-SimGrid lead by our team was recognized as a program highlighted project ("projet phare").

3. Scientific Foundations

3.1. Structuring of Applications for Scalability

Participants: Pierre-Nicolas Clauss, Sylvain Contassot-Vivier, Jens Gustedt, Soumeya Leila Hernane, Vassil Iordanov, Thomas Jost, Wilfried Kirschenmann, 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 [57]. 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) [57], LogGP (Latency overhead gap Procs) [48], CGM (Coarse Grained Multicomputer) [51] 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. At the algorithmic level, 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 [41], [40] or have dealt with the implementation aspects [42]. 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 had been proposed in [43].

A natural extension of those works is the study of asynchronism in hierarchical and hybrid clusters, that is to say, clusters in which there are different levels of computational elements and those elements may be of different kinds. Typically, a cluster of workstations with at least one GPU in each node forms such a hierarchical and hybrid system. To the best of our knowledge, although GPGPU knows a great success since the last few years, it is not yet very much used in clusters. It is quite probable that this is mainly due to the rather important cost of data transfers between the GPU memory and its host memory which generates an additional communication overhead in parallel algorithms.

Still, there are some algorithms which may be less impacted than the others by that overhead, those that are asynchronous and iterative. This comes from the facts that they provide an implicit overlapping of communications by computations and that the iterations are no longer synchronized, which provides much more flexibility according to the parallel system.

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 had been given in [5] with PRO.

In a complementary approach we had addressed (3) to develop a simple interface that gives a consistent view of the data services that are exported to an application.

Starting from these models, 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

Participants: Pierre-Nicolas Clauss, Stéphane Genaud, Soumeya Leila Hernane, Constantinos Makassikis, Stéphane Vialle.

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 aim at solving are quite complex, involving hundreds or thousands of entities for which we search for optimized solutions. 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. Much of the evaluations must be experimental, thus we need good foundations tools for experiments in that area, see Sections 3.3, 5.3 and 5.2.

3.3. Experimentation Methodology

Participants: Sébastien Badia, Pierre-Nicolas Clauss, Fekari El Mehdi, Jens Gustedt, Lucas Nussbaum, Martin Quinson, Philippe Robert, Cristian Rosa, Christophe Thiéry.

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, Soumeya Leila Hernane, Vassil Iordanov, Thomas Jost, Wilfried Kirschenmann, Stéphane Vialle.

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 [47], [50] 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 afterward) 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

In application of our main algorithmic techniques, 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. 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.

From a parallel computing point of view, the main difficulty has been to efficiently redistribute data and computations at each step of the algorithm. Our parallel algorithm has been successfully implemented and experimented on multi-core PC clusters (up to 256 nodes and 512 cores) and on a Blue Gene/L and a Blue Gene/P supercomputers (using up to 8192 nodes and 32768 cores, this machine was ranked 13 in Top500 in first semester 2008). 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.

4.1.5. Heterogeneous Architecture Programming

Clusters of heterogeneous nodes, composed of CPUs and GPUs, require complex multi-grain parallel algorithms: coarse grain to distribute tasks on cluster nodes and fine grain to run computations on each GPU. Algorithms implementation is achieved on these architectures using a multi-paradigm parallel development environment, composed of MPI and CUDA libraries (compiling with both gcc and nVIDIA nvcc compilers).

We investigate the design of multi-grain parallel algorithm and multi-paradigm parallel development environment for GPU clusters, in order to achieve both speedup and size up on different kinds of algorithms and applications. Our main application targets are: financial computations, PDE solvers, and relaxation methods.

4.1.6. Energy

Nowadays, people are getting more and more aware of the energetic problem and are concerned with reducing their energy consumption. Computer science is not an exception and some effort has to be made in our domain in order to optimize the energetic efficiency of our systems and algorithms.

In that context, we investigate the potential benefit of using intensively parallel devices such as GPUs in addition to CPUs. Although such devices present quite high instantaneous energy consumptions, their energetic efficiency, that is to say their ratio of flops/Watt is often much better than the one of CPUs.

4.1.7. Load balancing

Although load balancing in parallel computing has been intensively studied, it is still an issue in the most recent parallel systems whose complexity and dynamic nature regularly increase. For the grid in particular, where the nodes or the links may be intermittent, the demand is stronger and stronger for non-centralized algorithms.

In a joint work with the University of Franche-Comté, we study the design and optimal tuning of a fully decentralized load balancing scheme (see 7.2.5).

Another aspect of load-balancing is also addressed by our team in the context of the Neurad project. Neurad is a multi-disciplinary project involving our team and some computer scientists and physicists from the University of Franche-Comté around the problem of treatment planning of cancerous tumors by external radiotherapy. In that work, we have already proposed an original approach in which a neural network is used inside a numerical algorithm to provide radiation dose deposits in any heterogeneous environments, see [58]. The interest of the Neurad software is to combine very small computation times with an accuracy close to the most accurate methods (Monte-Carlo). It has to be noted that the Monte-Carlo methods take several hours to deliver their results where Neurad requires only a few minutes on a single machine.

In fact, in Neurad most of the computational cost is hidden in the learning of the internal neural network. This is why we work on the design of a parallel learning algorithm based on domain decomposition. However, as learning the obtained sub-domains may take quite different times, a pertinent load-balancing is required in order to get approximately the same learning times for all the sub-domains. The work here is thus more focused on the decomposition strategy as well as the load estimator in the context of neural learning.

4.2. Providing Environments for Experiments

Participants: Sébastien Badia, Pierre-Nicolas Clauss, Sylvain Contassot-Vivier, Fekari El Mehdi, Jens Gustedt, Lucas Nussbaum, Martin Quinson, Philippe Robert, Cristian Rosa, Christophe Thiéry, Stéphane Vialle.

4.2.1. Simulating Grid Platforms

We are major contributors to the SIMGrid tool, see 5.3, a collaboration with the Univ. of Hawaii, Manoa, and INRIA Grenoble-Rhône-Alpes, France. It enables the simulation of distributed applications in large-scale settings for the specific purpose of evaluating and assessing algorithms. Simulations not only allow repeatable results (what is near to impossible when experimenting the applications on real experimental facilities) 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. This is one of the main simulation tools used in the Grid Computing community.

4.2.2. Emulating Heterogeneity

We have designed a tool called *Wrekavoc*. The goal of Wrekavoc is to emulate a heterogeneous computing environment consisting of nodes of different compute and memory capacity and varying network bandwidth and latency. On such an emulated environment we want to execute a real, unmodified application.

Wrekavoc is a software for homogeneous Linux clusters that achieves this emulation by controlling the heterogeneity of a given platform by degrading CPU, network or memory of each node composing this platform.

4.2.3. 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.

4.2.4. Aladdin-G5K

The Aladdin-G5K initiative 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 1216 cores to the community.

4.2.5. 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 is tested is the interactive simulation of PDEs in physics, based on the Escapade project, see [4].

4.2.6. Experimental platform of GPU clusters

We participate in the scientific exploitation of two experimental 16-node clusters of GPUs that are installed at the SUPÉLEC Metz site. This platform allows to experiment scientific programming on GPU ("GPGPU"), and to track computing and energetic performances, with specific monitoring hardware. Development environments available on these GPU clusters are mainly the gcc suite and its OpenMP library, OpenMPI and the CUDA environment of nVIDIA's nvcc compiler.

5. Software

5.1. parXXL

Participants: 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 respectively for a consequent modeling and implementation of fine grained networks (*ParCeL*) and coarse grained algorithmic (*SSCRAP*).

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 has been registered at the *APP* and 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, Lucas Nussbaum, Tomasz Buchert.

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 understand the impact of platform parameters (latency, bandwidth, CPU speed, memory) 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.

Overlay networking Thanks to the notion of gateways it is possible to connect islets of nodes and to construct an overlay network matching a user-defined topology. Such network emulates TCP/IP based network with congestion emulation or routing.

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

Configuring and Controlling Nodes and Links. The configuration of a homogeneouscluster 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.

The current version of Wrekavoc has been registered at the *APP* and is available at http://wrekavoc.gforge.inria.fr/.

5.3. SimGrid

Participants: Pierre-Nicolas Clauss, Fekari El Mehdi, Martin Quinson, Lucas Nussbaum, Cristian Rosa, Christophe Thiéry.

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

The SIMGrid framework 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 main strong point of this is its carefully assessed **model validity**. To this end, the simulation kernel relies on 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 good level of accuracy [59].

The SIMGrid framework is currently extremely **fast**. Independent authors demonstrated its superior scalability over its main concurrence [45], [52]. In addition to the efficiency of the simulation models, this **scalability** is ensured by a layered architecture, with a simulation kernel computing the time taken by *actions* which need to consume *resources* to complete. Another layer of abstraction introduces the notion of processes and network routing between hosts. On top of this come the user interfaces aiming at providing the syntactic sugar easing the tool usage.

Several such user interfaces exist, ensuring the **versatility** of the SIMGrid framework by adapting to the user goal: *MSG* helps the study of distributed heuristics. This is the historical interface of SIMGrid, and remains the most used one. *SMPI* is a new interface which allows the simulation of MPI programs designed for multi-processor systems on a single computer [2]. *SimDag* eases the study of scheduling heuristics for DAGs of (parallel) tasks, which helps the work on parallel task scheduling. *GRAS* (Grid Reality And Simulation) eases the development of Grid services and infrastructures [8] through a specific interface implemented twice: once on top of the simulator for the comfort of development, and once using regular sockets for live deployments.

SIMGrid can be freely downloaded SimGrid and its user base is rapidly growing. Over the last decade, it grounded the experimental section of more than eighty scientific publications, only twenty of them being co-authored by members of the development team.

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, among 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.

5.5. ORWL and P99

Participant: Jens Gustedt.

ORWL is a reference implementation of the Ordered Read-Write Lock tools as described in [11].

It aims to work as standalone library in a shared memory and distributed setting. The implementation is uniquely based on C99 and POSIX. ORWL is has already been registered at the *APP*. Final tests and benchmarks have still to be performed before it will be made publicly available.

The macro definitions and tools for programming in C99 that have been implemented for ORWL have been separated out into a toolbox called P99. This toolbox allows *e.g* the simplified use of variable length argument list for macros and functions, default arguments of functions, compile time code unrolling, scope bound resource management, transparent allocation and initialization. It has been registered at the *APP* and is available at http://p99.gforge.inria.fr/.

6. New Results

6.1. Structuring of Applications for Scalability

Participants: Pierre-Nicolas Clauss, Sylvain Contassot-Vivier, Vassil Iordanov, Thomas Jost, Jens Gustedt, Soumeya Leila Hernane, Constantinos Makassikis, Stéphane Vialle.

6.1.1. Large Scale and Interactive Fine Grained Simulations

Our library *parXXL* allows the validation of a wide range of fine grained applications and problems. This year we were able to test the interactive simulation of PDEs in physics, see [4], on a large scale. Also, biologically inspired neural networks have been investigated using *parXXL* and the InterCell software suite. The InterCell suite and these applicative results have been presented in [22].

6.1.2. Distribution of a N-dimensional problem

In the previous years we have distributed a Stochastic Control Algorithm with EDF R&D. Our parallel algorithm has been designed to support dynamic N-dimensional problems on large scale architectures. It successfully run some electricity asset management problem with 7-energy-stocks and 10-state-variables, and achieved both speedup and size-up on PC clusters (up to 256 nodes and 512 cores) and on a Blue Gene/P (up to 8192 nodes and 32768 cores). In 2009, EDF already used this distributed and optimized algorithm and implementation in three applications. In 2010 we published a book chapter that introduces all algorithmic issues of this research [33].

However, we designed a parallel algorithm embedded in a stochastic control applicative algorithm. So, we design an applied research project aiming to develop a portable library to distribute and manage some dynamic N-dimensional arrays on large scale architectures, independently of the final application. Collaboration with EDF would be greatly helpful, but the project can be achieved by SUPÉLEC and INRIA.

6.1.3. Large Scale Models and Algorithms for Random Structures

A realistic generation of 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.

This year, we have run intensive simulations of these models that confirm the theoretical results and that showed the ability of that approach to model the properties of graphs from application domains. A manuscript reporting on these experimental results has been submitted to a journal, see [37].

6.1.4. Structuring algorithms for co-processing units

In 2009 and 2010, we have designed and experimented several algorithms and applications, in the fields of option pricing for financial computations, generic relaxation methods, and PDE solving applied to a 3D transport model simulating chemical species in shallow waters. We aim at designing a large range of algorithms for GPU cluster architectures, to develop a real knowledge about mixed coarse and fine grained parallel algorithms, and to accumulate practical experience about heterogeneous cluster programming.

Our PDE solver on GPU cluster has been designed in the context of a larger project on the study of asynchronism (see 3.1 and 6.1.5). We needed an efficient sparse linear solver. So, we have designed and developed such a solver on a cluster of GPU (up to 16 GPUs). As the GPU memory is still limited and iterative algorithms are less memory consuming than direct ones, our approach was to compare several iterative algorithms on a GPU. The results have lead to several deductions:

- there does not exist one method which fulfills both performances and generality
- speedups of GPUs according to CPUs are quite variable but most often around 10
- the GPU versions are less accurate than their CPU counterparts

In 2010 we have optimized our synchronous and asynchronous algorithms and implementations of our PDE solver, both on CPU and GPU cluster. The asynchronous parallel algorithm runs faster iterations, but requires more iterations and more complex convergence detection, see Section 6.1.5. It appears not always faster than the synchronous algorithm, depending on the problem size and the cluster features and size. We measured both computing and energy performances of our PDE solver in order to track the best solution, function of the problem size, the cluster size and the features of the cluster nodes. We are tracking the most efficient solution for each configuration. It can be based on a CPU or a GPU computing kernel, and on a synchronous or asynchronous parallel algorithm. Moreover, the fastest solution is not always the less energy consuming. See Section 6.2.2. Our recent results are introduced in [18] and [31]. We aim to design and automatic selection of the right kernel and the right algorithm, and to implement an auto-adaptive application, avoiding to the user to have to choose the kernel and algorithm to run.

In parallel, in the framework of the PhD thesis of Wilfried Kirschenmann, co-supervised by Stéphane Vialle (SUPELEC & AlGorille team) and Laurent Plagne (EDF SINETICS team), we have designed and implemented a unified framework based on generic programming to achieve a development environment adapted both to multi-core CPUs, multi-core CPUS with SSE units, and GPUs, for linear algebra applied to neutronic computations, see [27] and [23]. Our framework is composed of two layers: (1) MTPS is a low-level layer hiding the real parallel architecture used, and (2) Legolas++ is a high-level layer allowing to the application developer to rapidly implement linear algebra operations. The Legolas++ layer aims to decrease the development time, while the MTPS layer aims to automatically generates very optimized code for the target architecture and to decrease the execution time. Experimental performances of the MTPS layer appeared very good, the same source code achieved performances close to 100% of the theoretical ones, on any supported target architecture. Our strategy is to generate optimized data storage and data access code for each target architecture, not just different computing codes. A new version of Legolas++ is under development and will be achieved in 2011. It is optimized to use the MTPS layer.

At least, we have continued to design option pricers on clusters of GPUs, with Lokman Abbas-Turki (PhD student at University of Marne-la-Valée) and some colleagues from financial computing. In the past we developed some European option pricers, distributing independent Monte-Carlo computations on the nodes of a GPU cluster. In 2010 we succeeded to develop an American Option pricer on our GPU clusters, distributing strongly coupled Monte-Carlo computations. The Monte-Carlo trajectories depend on each others, and lead to many data transfers between CPUs and GPUs, and to many communications between cluster nodes. First results are encouraging, we achieve speedup and size up. Our algorithm and implementation will be optimized

in 2011. Again, we investigate bot computing and energy performances of our developments, in order to compare interests of CPU clusters and GPU clusters considering execution speed and the exploitation cost of our solution.

6.1.5. Asynchronism

In the previous paragraph is mentioned a project including the study of sparse linear solvers on GPU. That project deals with the study of asynchronism in hierarchical and hybrid clusters mentioned in 3.1.

In that context, we study the adaptation of asynchronous iterative algorithms on a cluster of GPUs for solving PDE problems. In our solver, the space is discretized by finite differences and all the derivatives are approximated by Euler equations. The inner computations of our PDE solver consist in solving linear equations (generally sparse). Thus, a linear solver is included in our solver. As that part is the most time consuming one, it is essential to get a version as fast as possible to decrease the overall computation time. This is why we have decided to implement it on GPU, as discussed in the previous paragraph. Our parallel scheme uses the Multisplitting-Newton which is a more flexible kind of block decomposition. In particular, it allows for asynchronous iterations.

Our first experiments, conducted on an advection-diffusion problem, have shown very interesting results in terms of performances [54]. Moreover, another aspect which is worth being studied is the full use of all the computational power present on each node, in particular the multiple cores, in conjunction with the GPU. This is a work in progress.

6.2. Transparent Resource Management

Participants: Pierre-Nicolas Clauss, Stéphane Genaud, Soumeya Leila Hernane, Constantinos Makassikis, Stéphane Vialle.

6.2.1. New Control and Data Structures for Efficiently Overlapping Computations, Communications and I/O

With the thesis of Pierre-Nicolas Clauss we introduced the framework of *ordered read-write locks*, ORWL, see [11], [17]. These 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 asynchronism 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.

Whereas the first experiments for ORWL had been done with our library *parXXL*, we now provided a standalone distributed implementation of the API that is uniquely based on C and POSIX socket communications. Our goal is to simplify the usage of ORWL and to allow portability to a large variety of platforms. This implementation runs on different flavors of Linux and BSD, on different processor types Intel and ARM, and different compilers, gcc, clang, opencc and icc. An experimental evaluation of the performance is on its way.

Data Handover, DHO, is a general purpose API that combines locking and mapping of data in a single interface. The access strategies are similar to ORWL, but locks and maps can also be hold only partially for a consecutive range of the data object. It is designed to ease the access to data for client code, by ensuring data consistency and efficiency at the same time.

In the thesis of Soumeya Hernane, we use the Grid Reality And Simulation (GRAS) environment of SimGrid, see 5.3, as a support of an implementation of DHO. GRAS has the advantage of allowing the execution in either the simulator or on a real platform. A first series of tests and benchmarks of that implementation demonstrates the ability of DHO to provide a robust and scalable framework, [38].

6.2.2. Energy performance measurement and optimization

Several experiments have been done on the GPU clusters of SUPÉLEC with different kinds of problems ranging from an embarrassingly parallel one to a strongly coupled one, via an intermediate level. Our first results tend to confirm our first intuition that the GPUs are a good alternative to CPUs for problems which can be formulated in a SIMD or massively multi-threading way. However, when considering not embarrassingly parallel applications the supremacy of a GPU cluster tends to decrease when the number of nodes increases (these results have been introduced to the European COST-IC0804 about *energy efficiency in large scale distributed systems* [19]). So, the energy saving can clearly be one of the decision criteria for using GPUs instead of CPUs, depending on the parallel algorithm and the number of computing nodes. The other criteria will generally be the ease of development and maintenance which have a direct impact of the economical cost of the software.

A straight sequel of that work has been to propose a model linking together the computing and energy performances [60]. That model allows the user to estimate the minimal speedup that a GPU version must get over its CPU counterpart in order to become more energy efficient. Thanks to this model, it also becomes possible to design a *Execution Control System* (ECS) that would dynamically choose the best conjunction of software version and hardware to run a given scientific application.

We intend to continue our investigations in that domain at different levels. Among them, we can cite further studies over our model as well as the development of the control system mentioned below. As we have developed an American option pricer on a CPU cluster and on a GPU cluster (see section 6.1.4), we plan to evaluate our performance model on this second distributed application. It is not obvious the GPU version is always the most interesting, depending on the problem size and the number of used nodes. Also, as mentioned in the previous section, we plan to study the energy aspect of hybrid processing, which corresponds to the full exploitation of the computational power present on every node of a parallel system. Typically, this includes the use of all the cores in a node in conjunction with any co-processing device. Finally, we would like to determine the best trade-off between energy saving and design/implementation costs.

6.2.3. Load balancing

A load-balancing algorithm based on asynchronous diffusion with bounded delays has been designed to work on dynamical networks [14]. It is by nature iterative and we have provided a proof of its convergence in the context of load conservation. Also, we have given some constraints on the load migration ratios on the nodes in order to ensure the convergence. This work has been extended, especially with a detailed study of the imbalance of the system during the execution of a parallel algorithm simulated in the SimGrid platform.

The perspectives of that work are double. The first one concerns the internal functioning of our algorithm. There is an intrinsic parameter which tunes the load migration ratios and we would like to determine the optimal value of that ratio. The other aspect is on the application side in a real parallel environment. Indeed, with Stéphane Genaud, we intend to apply this algorithm to a parallel version of the AdaBoost learning algorithm. We will compare our load-balancing scheme to other existing ones in different programming environments among which the P2P-MPI framework.
Concerning the Neurad project, our parallel learning proceeds by decomposing the data-set to be learned. However, using a simple regular decomposition is not sufficient as the obtained sub-domains may have very different learning times. Thus, we have designed a domain decomposition of the data set yielding sub-sets of similar learning times [32]. One of the main issue in this work has been the determination of the best estimator of the learning time of a sub-domain. As the learning time of a data set is directly linked to the complexity of the signal, several estimators taking into account that complexity have been tested, among which the entropy. Although our current results are satisfying, we are convinced that the quality of the decomposition can still be improved by several ways. The first one could be the design of a better learning time estimator. The second one is the strategy, *i.e.* the global scheme and its inner choice. Until now, we have opted for an URB (Unbalanced Recursive Bisection) approach, but we are currently working on the characterization of the best choice of dimension to divide at each decomposition step.

6.2.4. Fault Tolerance

6.2.4.1. Application-level fault tolerance

Concerning the fault tolerance, we have worked with Marc Sauget, from the University of Franche-Comté, on a parallel and robust algorithm for neural network learning in the context of the Neurad project [44]. A short description of that project is given in Section 4.1.7.

As that learning algorithm is to be used in local clusters we have opted for a simple approach based on the client-server model. So, there is a server which distributes all the learning tasks (the data-set sub-domains to be learned) to the clients which perform the learning. However, although the parallel context is very classical, the potentially very long learning times of the sub-domains (the data-sets may be huge) imply the insertion of a fault-tolerance mechanism to avoid the loss of any learning.

So, we have developed a detection mechanism of the clients faults together with a restarting process. We have also studied some variants of task redistribution as a fault may only be at the link level and may not imply the loss of the learning in progress on the corresponding client. Our final choice was to redistribute the task as soon as a fault is detected. Then, if that fault is later canceled by the client, this means that there are two clients performing the same learning. However, we do not stop any of the clients but let them run until one of them sends back its result. Only then, the other client is stopped.

That strategy has shown to be rather efficient and robust in our different experiments performed with real data on a local cluster where faults were generated. Although those results are rather satisfying, we would like to investigate yet more reactive mechanisms as well as the insertion of robustness at the server level.

6.2.4.2. Programming model and frameworks for fault tolerant applications

In the framework of the PhD thesis of Constantinos Makassikis, supervised by Stéphane Vialle, we have designed a new fault tolerance model for distributed applications which is based on a collaboration of fault-tolerant development frameworks and application-semantic knowledge supplied by users. Two development frameworks have been designed according to two different parallel programming paradigms: one for *Master-Workers* applications [39] and another one for some kind of *SPMD* applications including inter-nodes communications [25]. Users' task is limited as he merely needs to supply some computing routines (function of the application), and add some extra code to use parallel programming skeletons and to tune checkpointing frequency.

Our experiments have exhibited limited overheads when no failure happens and acceptable overheads in the worst case failures. These overheads appears less than the one obtained with all fault tolerant middlewares we have experimented, while development time overhead is very limited using our frameworks. Moreover, detailed experiments up to 256 nodes of our cluster have shown it is possible to finely tune the checkpointing policies of the frameworks in order to implement different fault tolerance strategies according, for example, to cluster reliability.

6.3. Experimentation Methodology

Participants: Tomasz Buchert, Pierre-Nicolas Clauss, Fekari El Mehdi, Jens Gustedt, Martin Quinson, Cristian Rosa, Lucas Nussbaum.

6.3.1. Overall Improvements of the SimGrid Framework

This year was the second year (out of three) of the ANR project centered on SIMGrid, of which we are principal investigator (see 7.2.4). Several improvement therefore occurred in the framework, with numerous contributions from the ANR participants. This served as a flagship for the whole SIMGrid project and hosted several of our research efforts, detailed in the subsequent sections (up to 6.3.6).

In addition, the software quality efforts were pursued through the INRIA ADT project (see 7.2.1) in order to maximize the impact of our research on our current user community. First, we improved further our automated regression testing infrastructure, by increasing the test coverage and through nightly builds on the INRIA pipol infrastructure. Then, we began a reorganization of the user documentation. Also, performance tuning deserved a lot of our attention this year too. Finally, two new bindings were added to make the framework usable to user preferring the Ruby or lua programming languages over the C (Java bindings were already available).

Finally, several operations were conducted to increase our user community, such as tutorials and flier distributions during conferences or the edition of a SimGrid newsletter aiming at explaining the newest tool evolutions to the users. The most preeminent actions were the organization of the first SimGrid User Days in April in Corsica. This event, hosting about 20 members of the development team alongside to 20 (current or potential) users for one week, were the occasion to present, and get feedback on, some new features as well as to ensure that the planned developments match the user community expectations. We were also present at the SuperComputing conference to try to meet potential users.

6.3.2. Synthesizing Generic Experimental Environments for Simulation

Simulation allows the fast and reproducible exploration of numerous experimental scenarios, including *what if* scenarios testing conditions not available at experimenter hand. But this almost unbound freedom in the experimental setup can also reveal disturbing. We we analyzed the requirements expressed by different research communities. As the existing tools of the literature are too specific, we then propose a more generic experimental environment synthesizer called Simulacrum. This tool allows its users to select a model of a currently deployed computing grid or generate a random environment, extract a subset of it that fulfills his/her requirements and import the result into the SIMGrid framework. This work, conducted in collaboration with F. Suter from the Computing Center at IN2P3 as well as with L. Bobelin from the Mescal team at INRIA Rhônes-Alpes, lead to a publication [28].

6.3.3. Model-Checking of Distributed Applications

For a few years we have cooperated with Stephan Merz from the VeriDis project team on adding model checking capabilities to the SIMGrid framework. The expected benefit of such an integration is that programmers can complement simulation runs by exhaustive state space exploration in order to detect errors such as race conditions that would be hard to reproduce by testing. Indeed, a simulation platform provides a controlled execution environment that mediates interactions between processes, and between processes and the environment, and thus provides the basic functionality for implementing a model checker. The principal challenge is the state explosion problem, as a naive approach to systematic generation of all possible process interleavings would be infeasible beyond the most trivial programs. Moreover, it is impractical to store the set of global system states that have already been visited: the programs under analysis are arbitrary C programs with full access to the heap, making even a hashed representation of system states very difficult and costly to implement.

In 2010, we made major advances on both the theoretical and the practical side of designing and implementing a model checker for SimGrid. Given that processes interact through explicit message passing, which is ultimately implemented in the SIMIX layer of SimGrid, the model checker need only control the possible interleavings of the operations provided at this layer (`Send`, `Recv`, `Test`, and `WaitAny`). Redundant interleavings can be ruled out by relying on Dynamic Partial-Order Reduction (DPOR) [53], which requires determining which interleaving orders may potentially lead to different results. Because we have only four primitive operations to consider, we could formally specify them in TLA$^+$ and prove independence results based on this model. This compares very favorably to a similar analysis carried out by Pervez et al. [56] at the MPI level, requiring more than 100 pages of TLA$^+$ specifications alone, for comparable reductions. This result has been published at a workshop [29].

In fact, the same techniques are also useful for avoiding redundant computations when performing large numbers of simulation runs. A stateless model checker relying on DPOR has now been implemented within the SimGrid platform, and a submission to a major conference is in preparation.

6.3.4. SMPI

The final goal of SMPI is to simulate a C/C++/FORTRAN MPI program designed for a multi-processor system on a single computer without any source code modification. This address one of the main limitation of SIMGrid, which requires the application to be written using one of the specific interfaces atop the simulator. New efforts have been put since July 2009 in this project, hereby continuing the work initiated by Henri Casanova and Mark Stilwell at University of Hawai'i at Manoa.

Previous work included a prototype implementation of various MPI primitives such as `send`, `recv`, `isend`, `irecv` and `wait`. Since the project's revival, many of the collective operations (such as `bcast`, `alltoall`, `reduce`) have been implemented. The standard network model used in SIMGrid has also been reworked to reach a higher precision in communication timings. Indeed, MPI programs are traditionally run on high performance computers such as clusters, and this requires to capture fine network details to correctly model the program behavior. Starting from the existing, validated network model of SimGrid, we have derived for SMPI a specific model a piece-wise linear model which closely fits real measurements. In particular, it enables to correctly models small messages and messages above the eager/rendezvous protocol limit. This work has been accepted for publication at the IPDPS conference in may 2011, and is already published as a research report [35].

6.3.5. Application Workload and Trace Replay

Simulations in SIMGrid are usually written as sets of agents exchanging messages. In some settings, an event-oriented approach may however lead to simpler solutions. Instead of specifying a large function containing the full logic of the agent, one simply specify how it should react to each possible incoming message or event. This could be particularly interesting for example when replaying post-mortem traces captured on real applications. In 2010, we worked on two aspects of this problem: the trace capture and the trace replay.

Concerning the trace capture, we initiated the Simterpose project, which aims at providing emulation capabilities on top of SimGrid, by intercepting the actions of the application and providing them to the simulator. During the internship of Marion Guthmuller in 2010, different methods to intercept the actions of applications were evaluated (`ptrace`, `LD_PRELOAD`, `DynInst`, `Valgrind`), and a trace extraction tool using `ptrace` was developed (thus similar to the classical tools `strace` or `gdb`). A preliminary publication is in preparation while we aim at improving this prototype and integrate it properly in the SIMGrid framework to actually allow the emulation of arbitrary applications through the simulator.

Concerning the trace replay, we worked on a replay mechanism in SIMGrid specialized for the study of MPI applications through post-mortem analysis. This work was conducted in collaboration with F. Suter from the Computing Center at IN2P3 as well as with F. Desprez and G. Markomanolis from the Graal team at INRIA Rhônes-Alpes. Its main originality is to rely on time-independent execution traces. This allows us to completely decouple the acquisition process from the actual replay of the traces in a simulation context. Then we are able to acquire traces for large application instances without being limited to an execution on a single compute cluster. Finally our replay framework is built directly on top of the SIMGrid simulation kernel. This work was recently submitted to the CCGrid conference, and is also available as research report: [36].

6.3.6. SimGrid Scalability Improvements

In addition to the software tuning and improvement described in 6.3.1, we tackled the main SIMGrid scalability limitations at an algorithmic level.

One of the main remaining limitation were the memory consumption due to its current network representation. Indeed, SimGrid used to use complete routing table, with the whole set of links used to go from any host to any other host. This large $O(N^2)$ routing table currently prevented SimGrid to simulate a platform with more than a few thousands of nodes, regardless of the simulated application. De Munck et al. have proposed [49] to recompute dynamically this routing using shortest-path algorithms. However, this approach suffers both from performance and scalability issues.

This year, we proposed a new platform description formalism to handle very large platforms in collaboration with A. Legrand and L. Bobelin from the Mescal team at INRIA Rhônes-Alpes, and with F. Suter from the Computing Center at IN2P3. This formalism takes advantage from the hierarchical structure of the platforms and from the regularity of some parts of it. This enables to build a small memory footprint description of the platform that would replace the current comprehensive routing table and would be much faster and scalable than the previous generic approaches proposed in [49]. This improvement will enable to keep using the same realistic flow-based models as earlier but with platforms that are many orders of magnitude larger than what is currently possible. The implementation of this new approach constituted the main work of David Marquez from University of Buenos Aires, Argentina, during his internship in our team. A publication summarizing these improvements is currently under preparation.

These improvements completely solved the memory pressure due to the internal network representation, but the memory remains the main limitation for example when simulating high performance applications through the new SMPI interface. In that case, it may reveal necessary to distribute the simulation in order to leverage the memory of several computing facilities. For that, we completely redesigned the SIMIX layer of the simulator in order to further decouple the execution of each user process from the others and from the simulation kernel execution. This decoupling were recently finished and we are now working on executing the simulation both in parallel (to leverage several computing cores) and in distributed (to leverage the memory of several computers).

6.3.7. Formal Verification of Distributed Algorithms' Specifications

In joint research with Stephan Merz and Sabina Akhtar of the Mosel team of INRIA Nancy and LORIA, we extended the PlusCal language [55] to allow the description and verification of distributed algorithms, whereas the original language is geared towards shared-memory concurrent programming. In 2010, the compiler from this language to the TLA$^+$ tool suite were made operational, and were presented in [13]. We are now exploring how to allow the TLC model-checker to apply dynamic partial order reduction technique to fight the combinatorial explosion. This could be made possible by adding specific information in the TLA$^+$ files generated from our extension of PlusCal.

6.3.8. Wrekavoc

During an Internship with Tomasz Buchert, the implementation of CPU performance evaluation in Wrekavoc was reconsidered to handle the case of the emulation of multi-core systems using multi-core nodes. Three different methods for the emulation of multi-core CPU performance were designed (Fracas, CPU-Hogs, CPU-Gov). This work resulted in a first publication in a workshop on the Fracas method [15]. A more complete publication is also in preparation, and was already published as a research report [34]. Tomasz Buchert submitted this work as a master thesis at his home university [46].

6.3.9. Grid'5000 and ADT Aladdin-G5K

Grid'5000 is an experimental platform for research on distributed systems, composed of 9 sites in France. In 2010, the "*grelon*" cluster was replaced by the newly bought "*graphene*" cluster, composed or 144 nodes (with 4 cores and 16 GB of RAM each). In addition to "*graphene*", the "*griffon*" cluster (92 nodes, 8 cores each) is still operational.

Lucas Nussbaum collaborated with the INRIA CACAO team on the technical aspects of the RSA-768 experiment that led to a new record in integer factorization. This work led to a publication [24].

6.3.10. DSL-Lab

The DSL-Lab platform was built during the ANR JC DSL-Lab. It is composed of 40 nodes dedicated to experiments on the broadband Internet. In 2010, a final publication summarizing the results of the project was published [21].

6.3.11. Experimental cluster of GPUs

The experimental platform of SUPÉLEC for "GPGPU", see Section 4.2.6, has been improved again in 2010.

First, the 16 NVIDIA GPUs GTX285 have been replaced by new NVIDIA GPUs GTX480 ("Fermi" architecture), and the 16 old GT8800 have been replaced by the 16 GTX285. So, the first cluster is now composed of 16 PCs, each one hosting a dual-core CPU and a GPU card: a nVIDIA GeForce GT285, with 1GB of RAM (on the GPU card). The 16 nodes are interconnected across a devoted Gigabit Ethernet switch. The second cluster has 16 more recent nodes, composed of an Intel Nehalem CPU with 4 hyper-threaded cores at 2.67GHz, and a nVIDIA GTX480 ("Fermi") GPU card with 1.5GB of memory. This cluster has a Gigabit Ethernet interconnection network too. These 2 clusters can been accessed and used like one 32-nodes heterogeneous cluster of hybrid nodes. This platform has allowed us to experiment different algorithms on an heterogeneous cluster of GPUs.

Second, the energy consumption of each node of the cluster hosting the GTX285 GPUs is monitored by a Raritan DPXS20A-16 device that continuously measures the electric power consumption (in Watts). But the cluster hosting the GTX480 GPUs consumes more energy, and exceeds the maximum energy supported by a Raritan DPXS20A-16 device. So, we have improved its energy monitoring system, and it is now monitored by two different Raritan devices.

Third, we have also improved our software (Perl and shell script) that sample the electrical power (Watt) measured by the Raritan devices and compute the energy (Joule or Watt Hour) consumed by the computation on each node and on the complete cluster (including the interconnection switch). This energy consumption monitoring system (hardware and software) has been intensively used to measure performances of our PDE solver (see sections 6.1.4 and 6.1.5), and our American option pricer (see section 6.1.4).

This platform has been intensively used to get experimental performance measures published in [31], [19], [18] and in a book chapter to appear in 2011 [60].

7. Other Grants and Activities

7.1. Regional initiatives

7.1.1. Lorraine Regional Council

Martin Quinson received a grant from the Lorraine Region for two years (2010–2011) 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. The results of this action are described in Sections 6.3.3 and 6.3.7.

Martin Quinson and Lucas Nussbaum are leading a project of the regional CPER contract on experimental grids. In 2011, this project benefited of a funding of 400k euros to maintain and improve the local Grid'5000 infrastructure and animate both the research on experimental grids and the research community using these facilities. The publication [24] is resulting from this effort.

7.2. National Initiatives

7.2.1. INRIA ADTs

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 development of Grid'5000 by providing funding for engineers and training and some parts of the hardware.

SimGrid for human beings is another INRIA Technological Development Action, aiming at providing engineering manpower to the SIMGrid project to improve the documentation, provide stock implementations of classical algorithms in order to ease its usage by the users. The results of this actions are described in Section 6.3.1.

7.2.2. INRIA AEs

Héméra is an INRIA Large Wingspan project, started in 2010, that aims at demonstrating ambitious up-scaling techniques for large scale distributed computing by carrying out several dimensioning experiments on the Grid'5000 infrastructure, at animating the scientific community around Grid'5000 and at enlarging the Grid'5000 community by helping newcomers to make use of Grid'5000.

Within that project, Martin Quinson, Lucas Nussbaum and Stéphane Genaud lead three working groups, respectively on *simulating large-scale facilities*, on *conducting large and complex experimentations on real platforms*, and on *designing scientific applications for scalability*.

7.2.3. 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] and in the Embedded Pole. 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.

7.2.4. ANR Initiatives

Martin Quinson is the principal investigator 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 leading team, we are involved in most parts of this project, which allows the improvement of our tool even further and set it as the reference in its domain (see Sections 6.3.1 through 6.3.6).

7.2.5. Bilateral Collaborations

With Jacques M. Bahi from the University of Franche-Comté, we work on the design and implementation of a decentralized load-balancing algorithm which works with dynamical networks. In such a context, we consider that the nodes are always available but the links between them may be intermittent. According to the load-balancing task, this is a rather difficult context of use. Our algorithm is based on asynchronous diffusion.

Lucas Nussbaum and Martin Quinson are participating to a research effort lead by F. Suter from the Computing Center of IN2P3. This project is jointly funded by CNRS' Institut des Grilles and INRIA's ADT Aladdin in a program that aims at bridging the production grids and distributed systems research communities. The overall goal of the project is to explore ways to enable the scientific study and the evaluation of improvements in the context of the gLite grid middleware. This project was granted 5000 euros for 2010-2011.

7.3. European Initiatives

7.3.1. Energy efficiency in large scale distributed systems

We participate to the COST (European Cooperation in the field of Scientific and Technical Research) Action IC0804
Energy efficiency in large scale distributed systems, started this year. We plan several contributions to this project. First, we want to integrate into the SIMGrid tool the energy models developed in the work package 2 of this project to allow the use of our simulator during the evaluation of algorithms aimed by the work package 3. Second, in the work package 2 we want to design energy models compatibles with our experiments on the CPU+GPU clusters of SUPÉLEC, and in the work package 3 we aim to use these models and our various experiments to design parallel algorithms for CPU+GPU clusters, that will be optimized both in speed and in energy consumption (tracking a right compromise).

[1] *Réseau Grand Est*

7.3.2. Bilateral Collaborations

Martin Quinson is leading a research effort aiming at adapting the SIMGrid framework to ease its use by the CERN team in charge of managing the data associated to the LHC instrument through the DQ2 middleware. In particular, we are working on modeling the storage elements used in DQ2, which are currently not simulated in SIMGrid. This work is done in collaboration with the team PH-ADP-DDM lead by V. Garone at CERN and with F. Suter from the Computing Center of IN2P3, and benefited of a funding of 5000 euros for 2010-2011 from the CNRS (through the grid institute – IdG) and the INRIA (through the Aladdin ADT).

7.4. International Initiatives

7.4.1. Bilateral Collaborations

We collaborate with Henri Casanova of University of Hawai'i at Manoa on the SimGrid framework, as detailed in 5.3. The result obtained this year on the simulation of MPI applications are detailed in Section 6.3.4.

Finally, we also collaborate with David Elizondo from the University of Leicester in Great Britain on the problem of linear separability determination. Our current work deals with the design and implementation of a fast algorithm to determine whether or not two or more sets of points in $\mathbb{R}^n$ are linearly separable. That work is almost finished and should be submitted for publication during the next year.

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.

Sylvain Contassot-Vivier is co-animator of the Embedded Pole in the ASR GDR of CNRS.

Martin Quinson and Lucas Nussbaum are members of the steering committee for ADT Aladdin-G5K, and serve as head and vice-head for the Nancy Grid'5000 site from the national perspective. Also, they are missioned by the direction of the Nancy INRIA research center to animate the regional researches on experimental grids. In addition, Lucas Nussbaum is also a member of the technical committee. Finally, Martin Quinson, Lucas Nussbaum and Stéphane Genaud are leading three working groups (respectively on *simulating large-scale facilities*, on *conducting large and complex experimentations on real platforms*, and on *designing scientific applications for scalability*) of the INRIA AE project Héméra on experimental facilities for distributed computing.

Jens Gustedt is co-responsible of the Computer Science Department of the doctoral school IAEM Lorraine, and as such member of the board of that school. He also is member of the executive committee of the project assembly of the INRIA Center Nancy – Grand Est and of the recruitment commission for doctoral students and postdocs of the same center.

8.1.2. Scientific Expertise

In 2010, Jens Gustedt was a member of the thesis committee of Maria Pentcheva, Nancy University.

In 2010, Stéphane Genaud was referee and member of the thesis committee of Fabrice Dupros, University Bordeaux 1, France.

In 2010, Sylvain Contassot-Vivier was a member of the thesis committees of Ahmed Ahmed, University of Strasbourg, Thomas Girod, University Henri Poincaré - Nancy 1, and referee of the thesis of Mirna Eskandar, University of Franche-Comté.

In 2010, Stéphane Vialle was president of the thesis committee of Sekou Diakite, University of Franche-Comté.

8.1.3. Teaching Activities

Martin Quinson is teaching the following modules at ÉSIAL (University Henri Poincaré - Nancy 1): "*C and Shell*", "*Techniques and Tools To Program*" (1A) and "*Networks and Systems*" (2A). He also participates to the modules "*Object Oriented Programming*" (1A). He is also responsible of the first year curricula of ÉSIAL.

Lucas Nussbaum is teaching the following modules at IUT Nancy-Charlemagne (University Nancy 2): "*Object-oriented programming*" (1A), "*Networking*" (AS), "*Installation and configuration of Linux*" (Licence pro ASRALL), "*Administration of applications and services*" (Licence pro ASRALL) and Systems (1A).

Sylvain Contassot-Vivier is teaching "*Algorithmic and complexity*" (M1), "*Introduction to parallelism*" (M1), "*Communicating systems*" (M2), "*Dynamical systems*" (M2), "*Networks*" (M2) and "*Parallel algorithms*" (M2) at the University Henri Poincaré - Nancy 1. He is also responsible of the part of the professional master in "*Networks, Services and Mobility*" delocalized at Casablanca, Morocco.

Stéphane Vialle is teaching "*Information Systems*" (M1) at SUPÉLEC at Paris, "*Parallel and Distributed Computing, and Computing Grids*" (M2) at SUPÉLEC at Metz, "*Parallel Programming*" (M2) at SUPÉLEC at Paris, "*Distributed Applications*" (M2-ILC) and "*Distributed Systems and Grids*" (M2-RISE) at the University of Strasbourg, and "*Concurrent Applications: concepts and tools*" at CNAM (National French Institution for Adult Training Courses) in Lorraine. He taught "*GPU programming and GPU cluster programming*" to engineers and researchers in SUPÉLEC and LORIA (one day course and lab). He is also responsible of the organization 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. This year, DMTCS published several special issues and proceedings volumes of noticeable events in the domain.

In 2010, Jens Gustedt has served as program committee member of the 19^{th} Euromicro International Conference on Parallel, Distributed and network-based Processing PDP 2011, and of the 2^{nd} Workshop on Complex Networks CompleNet 2010.

Lucas Nussbaum was member of the organization and program committee for the Grid'5000 Spring School 2010, and delivered a keynote talk on "Working efficiently on Grid'5000".

Martin Quinson has served as program committee member of the following events: The Grid'5000 Spring School 2010, the Third IEEE/ACM International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SimuTools'10), the 11th IEEE/ACM International Symposium on Cluster, Clouds and Grid Computing (CCGrid'10) and the 5th International Workshop on Modeling, Simulation, and Optimization of Peer-to-peer environments (MSOP2P 2011, associated to Euromicro PDP 2011).

Stéphane Genaud was member of the program committee of the IEEE/ACM International Conference on Grid Computing 2010 and the 22nd IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2010).

Stéphane Vialle was one of the Program Chairs of the 3rd International Workshop on Parallel and Distributed Computing in Finance (PDCoF'2010), and of the 4th International Workshop on Parallel and Distributed Computing in Finance (PDCoF'2011). Stéphane Vialle was also member of the program committee of the international conference on Architecture of Computing Systems (ARCS 2011), and was member of the twentieth RenPar conference committee ("Rencontres francophones du Parallélisme") (Renpar 2011).

8.1.5. Refereeing

In 2010, members of the team served as referees for the following journals and conferences:

Journals: Discrete Applied Mathematics, Engineering Applications of Artificial Intelligence, Journal of Grid Computing, Journal of Parallel and Distributed Computing, Neural Networks

Conferences: ARCS 2011, EuroPar 2010, Grid 2010, IPDPS 2011, PDCoF 2010, PDP 2011, PDCS 2010,

Projects: Jens Gustedt also participated to the evaluation process for PhD thesis CIFRE, and has been expert for a funding program of the government of Wallonia Region, Belgium.

8.1.6. Invitations and participations to scientific events

Stéphane Vialle was invited speaker to the *Mons GPU Day* seminar, in Mons, Belgium, in November 2010.

Martin Quinson was invited to give a three hours tutorial at the 8th ACM/IEEE International Conference on High Performance Computing & Simulation (HPCS'10) on *Experimenting HPC Systems with Simulation* in Caen, France in June.

He was also invited to give a regular presentation on *Performance Assessment of Distributed Scientific Applications* at the workshop "Challenges & Pitfalls of Performance Assurance", associated to CECMG'10 in Darmstadt, Germany in March.

9. Bibliography

Major publications by the team in recent years

[1] 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/.

[2] P.-N. CLAUSS, M. STILLWELL, S. GENAUD, F. SUTER, H. CASANOVA, M. QUINSON. *Single Node On-Line Simulation of MPI Applications with SMPI*, in "25th IEEE International Parallel and Distributed Processing Symposium (IPDPS)", 2011, Accepted for publication.

[3] 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, Springer, August 2007, vol. 4641, p. 160–169, http://hal.inria.fr/inria-00130734/en/.

[4] 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/.

[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", 2006, vol. 13, 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, Springer, February 2003, vol. 2668, p. 269-278.

[7] J. GUSTEDT, E. JEANNOT, M. QUINSON. *Experimental Validation in Large-Scale Systems: a Survey of Methodologies*, in "Parallel Processing Letters", 2009, vol. 19, n^{o} 3, p. 399-418, RR-6859, http://hal.inria.fr/inria-00364180/en/.

[8] 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.

Publications of the year

Articles in International Peer-Reviewed Journal

[9] L.-C. CANON, O. DUBUISSON, J. GUSTEDT, E. JEANNOT. *Defining and Controlling the Heterogeneity of a Cluster: the Wrekavoc Tool*, in "Journal of Systems and Software", 2010, vol. 83, n^{o} 5, p. 786-802 [*DOI :* 10.1016/J.JSS.2009.11.734], http://hal.inria.fr/inria-00438616/en.

[10] L.-C. CANON, E. JEANNOT. *Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments*, in "IEEE Transactions on Parallel and Distributed Systems", April 2010, vol. 21, n^{o} 4 [*DOI :* 10.1109/TPDS.2009.84>], http://www.computer.org/portal/web/csdl/doi/10.1109/TPDS.2009.84, http://hal.inria.fr/inria-00430920/en.

[11] P.-N. CLAUSS, J. GUSTEDT. *Iterative Computations with Ordered Read-Write Locks*, in "Journal of Parallel and Distributed Computing", 2010, vol. 70, n^{o} 5, p. 496-504 [*DOI :* 10.1016/J.JPDC.2009.09.002], http://hal.inria.fr/inria-00330024/en.

[12] L. GHEMTIO, E. JEANNOT, B. MAIGRET. *Efficiency of a hierarchical protocol for highthroughput structure-based virtual screening on Grid5000 cluster grid*, in "Open Access Bioinformatics", May 2010, vol. 2, p. 41-53 [*DOI :* 10.2147/OAB.S7272], http://hal.inria.fr/hal-00547970/en.

International Peer-Reviewed Conference/Proceedings

[13] S. AKHTAR, S. MERZ, M. QUINSON. *A High-Level Language for Modeling Algorithms and their Properties*, in "13th Brazilian Symposium on Formal Methods (SBMF'10)", Brazil Natal, November 2010, http://hal.inria.fr/inria-00537779/en.

[14] J. M. BAHI, S. CONTASSOT-VIVIER, A. GIERSCH. *Load balancing in dynamic networks by bounded delays asynchronous diffusion*, in "VECPAR'10", United States Berkeley, 2010, Paper 31, An extended version is to appear in LNCS, http://hal.inria.fr/hal-00547300/en.

[15] T. BUCHERT, L. NUSSBAUM, J. GUSTEDT. *Accurate emulation of CPU performance*, in "8th International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms - HeteroPar'2010", Italy Ischia, August 2010, http://hal.inria.fr/inria-00490108/en.

[16] L.-C. CANON, E. JEANNOT, J. WEISSMAN. *A Dynamic Approach for Characterizing Collusion in Desktop Grids*, in "24th IEEE International Parallel and Distributed Processing Symposium - IPDPS 2010", United States Atlanta, IEEE, 2010, p. 1-12 [*DOI :* 10.1109/IPDPS.2010.5470422], http://hal.inria.fr/inria-00441256/en.

[17] P.-N. CLAUSS, J. GUSTEDT. *Experimenting Iterative Computations with Ordered Read-Write Locks*, in "18th Euromicro International Conference on Parallel, Distributed and network-based Processing", Italy Pisa, M. DANELUTTO, T. GROSS, J. BOURGEOIS (editors), IEEE, 2010, p. 155-162 [*DOI :* 10.1109/PDP.2010.11], http://hal.inria.fr/inria-00436417/en.

[18] S. CONTASSOT-VIVIER, T. JOST, S. VIALLE. *Impact of asynchronism on GPU accelerated parallel iterative computations*, in "PARA 2010 : State of the Art in Scientific and Parallel Computing", Iceland Reykjavik, June 2010, 4 pages, http://hal.inria.fr/hal-00491976/en.

[19] S. CONTASSOT-VIVIER, S. VIALLE, T. JOST. *Optimizing computing and energy performances on GPU clusters: experimentation on a PDE solver*, in "1st Year workshop of the COST Action IC0804", Germany Passau, J.-M. PIERSON, H. HLAVACS (editors), IRIT, 2010, p. 50-54, http://hal.inria.fr/hal-00517374/en.

[20] J. DAVIES, H. ZHANG, L. NUSSBAUM, D. GERMAN. *Perspectives on Bugs in the Debian Bug Tracking System*, in "7th IEEE Working Conference on Mining Software Repositories (MSR'2010): Mining Challenge", South Africa Cape Town, May 2010, http://hal.inria.fr/inria-00502883/en.

[21] G. FEDAK, J.-P. GELAS, T. HÉRAULT, V. INIESTA, D. KONDO, L. LEFÈVRE, P. MALECOT, L. NUSSBAUM, A. REZMERITA, O. RICHARD. *DSL-Lab: a Low-power Lightweight Platform to Experiment on Domestic Broadband Internet*, in "9th International Symposium on Parallel and Distributed Computing (ISPDC'2010)", Turkey Istanbul, July 2010, http://hal.inria.fr/inria-00502888/en.

[22] J. GUSTEDT, S. VIALLE, H. FREZZA-BUET, D. BOUMBA SITOU, N. FRESSENGEAS. *InterCell: a Software Suite for Rapid Prototyping and Parallel Execution of Fine Grained Applications*, in "PARA 2010 : State of the Art in Scientific and Parallel Computing", Iceland Reykjavick, June 2010, 4 pages, http://hal.inria.fr/hal-00491969/en.

[23] W. KIRSCHENMANN, L. PLAGNE, S. VIALLE. *Multi-Target Vectorization With MTPS C++ Generic Library*, in "PARA 2010 : State of the Art in Scientific and Parallel Computing", Iceland Reykjavik, June 2010, 4 pages, http://hal.inria.fr/hal-00491980/en.

[24] T. KLEINJUNG, L. NUSSBAUM, E. THOMÉ. *Using a grid platform for solving large sparse linear systems over GF(2)*, in "11th ACM/IEEE International Conference on Grid Computing (Grid 2010)", Belgium Brussels, October 2010, http://hal.inria.fr/inria-00502899/en.

[25] C. MAKASSIKIS, V. GALTIER, S. VIALLE. *A Skeletal-Based Approach for the Development of Fault-Tolerant SPMD Applications*, in "The 11th International Conference on Parallel and Distributed Computing, Applications and Technologies", China Wuhan, December 2010 [*DOI :* 10.1109/PDCAT.2010.89], http://hal.inria.fr/inria-00548953/en.

[26] L. NUSSBAUM, S. ZACCHIROLI. *The Ultimate Debian Database: Consolidating Bazaar Metadata for Quality Assurance and Data Mining*, in "7th IEEE Working Conference on Mining Software Repositories (MSR'2010)", South Africa Cape Town, J. WHITEHEAD, T. ZIMMERMANN (editors), May 2010, http://hal.inria.fr/inria-00502886/en.

[27] L. PLAGNE, W. KIRSCHENMANN, S. VIALLE. *Toward a Multi-Target Linear Algebra Library for GPUs and Multicore CPUs*, in "SIAM Conference on Parallel Processing for Scientific Computing (PP10)", United States Seattle, Washington, February 2010, MS55, http://hal.inria.fr/hal-00525899/en.

[28] M. QUINSON, L. BOBELIN, F. SUTER. *Synthesizing Generic Experimental Environments for Simulation*, in "Fifth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing", Japan Fukuoka, November 2010, http://hal.inria.fr/inria-00502839/en.

[29] C. ROSA, S. MERZ, M. QUINSON. *A Simple Model of Communication APIs Application to Dynamic Partial-order Reduction*, in "10th International Workshop on Automated Verification of Critical Systems - AVOCS 2010", Germany Düsseldorf, September 2010, http://hal.inria.fr/inria-00532889/en.

National Peer-Reviewed Conference/Proceedings

[30] S. AKHTAR, S. MERZ, M. QUINSON. *Extending PlusCal: A Language for Describing Concurrent and Distributed Algorithms*, in "Actes des deuxièmes journées nationales du Groupement De Recherche CNRS du Génie de la Programmation et du Logiciel", France Pau, E. CARIOU, L. DUCHIEN, Y. LEDRU (editors), March 2010, http://hal.inria.fr/inria-00544137/en.

Scientific Books (or Scientific Book chapters)

[31] T. JOST, S. CONTASSOT-VIVIER, S. VIALLE. *An efficient multi-algorithms sparse linear solver for GPUs*, in "Parallel Computing: From Multicores and GPU's to Petascale (Volume 19)", B. CHAPMAN, F. DESPREZ, G. JOUBERT, A. LICHNEWSKY, F. PETERS, T. PRIOL (editors), Advances in Parallel Computing, IOS Press, 2010, vol. 19, p. 546-553, Extended version of EuroGPU symposium article, in the International Conference on Parallel Computing (Parco) 2009 [*DOI :* 10.3233/978-1-60750-530-3-546], http://hal.inria.fr/hal-00485963/en.

[32] M. SAUGET, R. LAURENT, J. HENRIET, M. SALOMON, R. GSCHWIND, S. CONTASSOT-VIVIER, L. MAKOVICKA, C. SOUSSEN. *Efficient Domain Decomposition for a Neural Network Learning Algorithm, used for the Dose Evaluation in External Radiotherapy*, in "Artificial Neural Networks - ICANN 2010 20th International Conference Proceedings", K. DIAMANTARAS, W. DUCH, L. ILIADIS (editors), Lecture Notes in Computer Science, Volume 6352, Springer-Heidelberg, 2010, p. 261-266, ISBN-978-3-642-15818-6 [*DOI :* 10.1007/978-3-642-15819-3_34], http://hal.inria.fr/hal-00520154/en.

[33] X. WARIN, S. VIALLE. *Design and Experimentation of a Large Scale Distributed Stochastic Control Algorithm Applied to Energy Management Problems*, in "Stochastic Control", C. MYERS (editor), Sciyo, August 2010, p. 103-124, Chapter 7, http://hal.inria.fr/hal-00537341/en.

Research Reports

[34] T. BUCHERT, L. NUSSBAUM, J. GUSTEDT. *Methods for Emulation of Multi-Core CPU Performance*, INRIA, November 2010, n° RR-7450, http://hal.inria.fr/inria-00535534/en.

[35] P.-N. CLAUSS, M. STILLWELL, S. GENAUD, F. SUTER, H. CASANOVA, M. QUINSON. *Single Node On-Line Simulation of MPI Applications with SMPI*, INRIA, October 2010, n° RR-7426, http://hal.inria.fr/inria-00527150/en.

[36] F. DESPREZ, G. MARKOMANOLIS, M. QUINSON, F. SUTER. *Assessing the Performance of MPI Applications Through Time-Independent Trace Replay*, INRIA, December 2010, n° RR-7489, http://hal.inria.fr/inria-00546992/en.

[37] J. GUSTEDT, P. SCHIMIT, H. RAGHAVAN. *Exploring the random genesis of co-occurrence graphs*, INRIA, January 2010, n° RR-7186, accepted for publication in Physica A, http://hal.inria.fr/inria-00450684/en.

[38] S. HERNANE, J. GUSTEDT, M. BENYETTOU. *Modeling and Experimental Validation of the Data Handover API*, INRIA, December 2010, n° RR-7493, http://hal.inria.fr/inria-00547598/en.

[39] C. MAKASSIKIS, V. GALTIER, S. VIALLE. *A Javaspace-based Framework for Efficient Fault-Tolerant Master-Worker Distributed Applications*, INRIA, December 2010, n° RR-7496, http://hal.inria.fr/inria-00548951/en.

References in notes

[40] J. M. 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.

[41] J. M. 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.

[42] J. M. 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.

[43] J. M. 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/.

[44] J. M. 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/.

[45] M. BARISITS, W. BOYD. *MartinWilSim Grid Simulator*, Vienna UT and Georgia Tech, CERN, Switzerland, 2009, http://www.slideshare.net/wbinventor/slides-1884876.

[46] T. BUCHERT. *Methods for Emulation of Multi-Core CPU Performance*, Poznań University of Technology, Poznań, Poland, December 2010.

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

[48] 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.

[49] S. DE MUNCK, K. VANMECHELEN, J. BROECKHOVE. *Improving the Scalability of SimGrid Using Dynamic Routing*, in "ICCS", 2009, p. 406-415.

[50] 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.

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

[52] W. DEPOORTER, N. MOOR, K. VANMECHELEN, J. BROECKHOVE. *Scalability of Grid Simulators: An Evaluation*, in "Proc. of the 14th Intl. Euro-Par Conf. on Parallel Processing", 2008, http://dx.doi.org/10.1007/978-3-540-85451-7_58.

[53] C. FLANAGAN, P. GODEFROID. *Dynamic partial-order reduction for model checking software*, in "SIGPLAN Not.", 2005, vol. 40, n^{o} 1, p. 110–121, http://doi.acm.org/10.1145/1047659.1040315.

[54] T. JOST, S. CONTASSOT-VIVIER, S. VIALLE. *An efficient multi-algorithms sparse linear solver for GPUs*, in "ParCo2009", France Lyon, Frédéric Desprez, 2009, http://hal.inria.fr/inria-00430520/en/.

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

[56] S. PERVEZ, G. GOPALAKRISHNAN, R. M. KIRBY, R. PALMER, R. THAKUR. *Practical Model Checking Method for Verifying Correctness of MPI Programs*, in "EuroPVM/MPI", Springer, 2007, p. 344-353.

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

[58] A. VASSEUR, L. MAKOVICKA, É. MARTIN, M. SAUGET, S. CONTASSOT-VIVIER, J. M. BAHI. *Dose calculations using artificial neural networks: a feasibility study for photon beams*, in "Nucl. Instr. and Meth. in Phys. Res. B", 2008, vol. 266, n^{o} 7, p. 1085-1093.

[59] P. VELHO, A. LEGRAND. *Accuracy Study and Improvement of Network Simulation in the SimGrid Framework*, in "Proceedings of the 2nd International Conference on Simulation Tools and Techniques (SIMUTools'09)", Roma, Italia, March 2009.

[60] S. VIALLE, S. CONTASSOT-VIVIER, T. JOST. *9*, in "Optimizing computing and energy performances in heterogeneous clusters of CPUs and GPUs", Chapman and Hall/CRC Press, Taylor and Francis Group LLC, 2011, in submission.