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 [research director, INRIA, team leader, HdR]
Emmanuel Jeannot [INRIA, until Sep. 30, 2009, HdR]
Stéphane Genaud [INRIA, on leave from Univ. Strasbourg until Sep. 30, 2009]

Faculty Member

Sylvain Contassot-Vivier [professor, UHP, HdR]
Lucas Nussbaum [assistant professor, U. Nancy 2, since Sep. 1, 2009]
Martin Quinson [assistant professor, UHP/ÉSIAL]
Stéphane Vialle [professor, SUPÉLEC Metz Campus, HdR]
Stéphane Genaud [Univ. Strasbourg, since Oct. 1, 2009]

Technical Staff

Pierre-Nicolas Clauss [Post-doc, since Dec. 1, 2009]
Fekari El Mehdi [engineer, INRIA, since Dec. 1, 2009]
Philippe Robert [engineer, INRIA, since Oct. 10, 2009]

PhD Student

Louis-Claude Canon [MENESR grant, until Sep. 30, 2009]
Pierre-Nicolas Clauss [MENESR grant, until Oct. 31, 2009]
Soumeya Leila Hernane [teaching assistant UST Oran, Algeria, since Oct 1, 2007]
Thomas Jost [since October, 2009]
Constantinos Makassikis [SUPÉLEC, 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 Janaury 1, 2009]

Administrative Assistant

Roxane Auclair [CNRS and UHP]
Cécilia Claude [INRIA]

2. Overall Objectives

2.1. Overall Objectives

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.

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 [55]. Their common characteristics are:

- Maximally exploit the data that is located on a particular node by a local computation.
- Collect all requests for other nodes during the computation.
- Only transmit these requests if the computation can't progress anymore.

The coarse grained models aim at being realistic with regard to two different aspects: algorithms and architectures. In fact, the coarseness of these models uses the common characteristic of today's parallel settings: the size of the input is orders of magnitude larger than the number of processors that are available. In contrast to the PRAM (Parallel Random Access Machine) model, the coarse grained models are able to integrate the cost of communications between different processors. This allows them to give realistic predictions about the overall execution time of a parallel program. As examples, we refer to BSP (Bulk Synchronous Parallel model) [55], LogP (Latency overhead gap Procs) [48], CGM (Coarse Grained Multicomputer) [50] 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 [43], [42] or have dealt with the implementation aspects [44]. But to fully exploit the benefits of those algorithms, not only the computations need to be asynchronous but also the controls of those algorithms. To fulfill such needs, a decentralized convergence detection algorithm has been proposed in [45].

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, the asynchronous iterative ones. 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.

In that context, we study the adaptation of asynchronous iterative algorithms on a cluster of GPUs for solving EDP 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 EDP 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. Our parallel scheme uses the Multisplitting-Newton which is a more flexible kind of block decomposition. In particular, it allows for asynchronous iterations.

Finally, each sub-domain of the problem is treated on one node. The non-linear computations are performed on the CPU whereas the linear resolutions are done on the local GPU. The nodes communicate their local results between each others according to a dependency graph induced by the problem.

Concerning (2), the number of supersteps and the minimization thereof should by themselves not be a goal. It has to be constrained by other more "*natural*" parameters coming from the architecture and the problem instance. A first solution that uses (2) to combine these objectives for homogeneous environments has been given in [5] with PRO.

In a complementary approach we have addressed (3) to develop a simple interface that gives a consistent view of the data services that are exported to an application, see [7].

Starting from 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: Louis-Claude Canon, Stéphane Genaud, Emmanuel Jeannot, Tchimou N'takpé.

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

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

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

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

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

3.3. Experimentation Methodology

Participants: Fekari El Medhi, Jens Gustedt, Emmanuel Jeannot, Lucas Nussbaum, Martin Quinson, Philippe Robert, Cristian Rosa.

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

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

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

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

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

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

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

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

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

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

4. Application Domains

4.1. High Performance Computing

Participants: Pierre-Nicolas Clauss, Sylvain Contassot-Vivier, Jens Gustedt, Soumeya Leila Hernane, Vassil Iordanov, Thomas Jost, Wilfried Kirschenmann.

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], [49] 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 greater 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.

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.

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 [9]. 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 the learnings of 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. Evolution of Scheduling Policies

Participants: Louis-Claude Canon, Stéphane Genaud, Emmanuel Jeannot, Tchimou N'takpé.

4.2.1. Scheduling on the Grid

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

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

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

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

4.2.2. Fault-Tolerant MPI

Message Passing Interface (MPI), is the main standard for message passing and SPMD programming. The libraries implementing this standard are widely used for programming parallel scientific applications.

This standard was designed for small scale systems and shows some limitations for nowadays systems (distributed or very large-scale). One of the main problem is the lack of fault-tolerance. Indeed, in case of a node crash a standard MPI application fails. However, node failure is very common in distributed environments and happens frequently in today supercomputers.

It is therefore required to cope with such failures in the case of MPI programs. There exists several possibilities among which check-point and restart or redundancy.

The P2P-MPI middleware offers a transparent support for redundancy through the replication of computations. The level of replication i.e, the number of replicas per process, can be chosen at runtime depending on the volatility of the environment. To maintain the coherence of the system, extra message exchanges are required, hence adding an overhead increasing with the replication level. We have studied and modeled the overhead depending on the replication level, and which trade-off between execution time and failure probability is optimal for a given failure distribution, based on a study of real failure traces [53].

4.3. Providing Environments for Experiments

Participants: Sylvain Contassot-Vivier, Fekari El Medhi, Jens Gustedt, Emmanuel Jeannot, Lucas Nussbaum, Martin Quinson, Philippe Robert, Cristian Rosa, Stéphane Vialle.

4.3.1. Simulating Grid Platforms

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

4.3.2. Emulating Heterogeneity

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

4.3.3. Use of Formal Methods to Assess Distributed Algorithms

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

4.3.4. Aladdin-G5K

The Aladdin-G5K 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.

4.3.5. InterCell

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

The purpose is to allow easy fine grain parallel design, providing interactive tools for the visualization and the management of the execution (debug, step by step, *etc*). The parallelization effort is not visible to the user, since InterCell relies on the dedicated *parXXL* framework, see 5.1 below. Among the applications that will be tested is the interactive simulation of PDEs in physics, based on the Escapade project, see [4].

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

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

Historically *parXXL* is the result of a merge of two different projects, *ParCeL* (from SUPÉLEC) and *SSCRAP* (from INRIA), that stand 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 is available at http://parxxl.gforge.inria.fr/ and integrates:

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

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

5.2. Wrekavoc

Participants: Jens Gustedt, Emmanuel Jeannot.

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

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.

5.3. SimGrid

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

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

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

The framework relies on a simulation kernel using a blend of analytical models and coarse-grain discrete event simulation. It proves several orders of magnitude faster than usual packet-level simulators used in the networking community (such as ns2 or GTNetS) while providing an acceptable level of accuracy [51].

SIMGrid provides several user interfaces depending on the user goal.

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

SimDag eases the study of scheduling heuristics for DAGs of (parallel) tasks, which helps the work on parallel task scheduling.

SMPI allows the emulation of MPI program on top of a simulated environment. This work is still in progress.

GRAS (Grid Reality And Simulation) eases the development of Grid services and infrastructures [8].

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

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

SIMGrid is freely downloadable SimGrid and its user base is rapidly growing. It grounded the experimental section of more than fifty scientific publications (two third of them from users not being part of the core 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.

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

The integration of the formerly separated libraries *ParCeL* and *SSCRAP* into *parXXL* allows the validation of the whole on a wide range of fine grained applications and problems. Among the applications that we started testing this year is the interactive simulation of PDEs in physics, based on the InterCell project, see [4]. There the idea is to express PDEs as local equations in the discrete variable space and to map them in terms of update functions on a cellular automaton. With the help of *parXXL* this fine-grained automaton can then be mapped on the coarse grained target machine, and the automaton cells can communicate synchronously or asynchronously. Finally any fine-grained automaton required to solve the expressed PDE can be generated quickly and evaluated efficiently. Our hope is to be able to find solutions for certain types of physically motivated problems, for which currently no performing solvers exist.

A first applicative result has been presented in [32], and several experiments have been exhibited at SuperComputing-2009 on the INRIA booth. At the end of 2009, several complex physical problems and also biologically inspired neural networks are under investigation using *parXXL* and the InterCell software suite.

6.1.2. Distribution of a Stochastic Control Algorithm

The current version of our Stochastic Control Algorithm, see Section 4.1.4, allows to successfully optimize an electricity asset management problem with 7-energy-stocks and 10-state-variables, and to achieve 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, rank 13 in Top500 in first semester 2008).

In 2009, EDF used this distributed and optimized algorithm and implementation in three applications: (1) an electricity asset management tool at *EDF R&D* (optimizing the simultaneous control of N electricity production units to minimize the cost of the electricity production), (2) a valorisation tool for a thermic power station in the English subsidiary company *EDF Energy* (tracking the best production control of this power station function of the energy market), and (3) in another valorisation tool of a thermic power station in Holland (with more constraints on the production control). In parallel, we have attempted to adapt our parallel algorithm for GPU clusters. We succeeded to design and implement a coarse and fine grained solution on GPU clusters, but we did not achieve good performances, and pointed out the limit of the GPU approach for this problem. Some new large scale experiments are planed in 2010 at EDF, and we aim to experiment the fault tolerance mechanisms designed in the PhD thesis of Constantinos Makassikis on some of these applications.

In 2009, we also attempted to port this parallel algorithm on a GPU cluster while EDF has exploited the multicore CPU cluster version in 3 applications. New applications are under development at EDF, and large scale experiments are planed for 2010.

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. The description of the model is found in [38]; two internships, **[stale citation GUSTEDT-TR-2009-to-be-finished]**, show promising experimental results.

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

Mutual exclusion is one of the classical problems of distributed computing. Several solutions have been devised in the literature, but most of them remain relatively far from practitioner needs. This concern first of all distributed platforms, on which are particularly difficult to control and to certify. We proposed an extension to the classical Naimi-Trehel algorithm to allow partial locks on a given data. Such ranged locks offer a semantic close to the POSIX file locking, where each thread locks the sub-part of the file it is working on, in the hope that this work can prove useful to high performance computing practitioner [30].

Further, with the thesis [10] we introduced the framework of *ordered read-write locks*, ORWL, that are characterized by two main features: a strict FIFO policy for access and the attribution of access to *lock-handles* instead of processes or threads. These two properties allow applications to have a controlled pro-active access to resources and thereby to achieve a high degree of 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. The model and theoretical properties of it are published in [15]. An experimental validation of our approach is given in [22].

6.1.5. structuring algorithms for co-processing units

In 2009 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 to design 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 6.1.6). 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

All those results are rather encouraging for the use of GPUs in linear problems. Those results have been presented in [33]. The following of that work will be to develop a direct method. The main interest of direct methods lies in the better performances compared to iterative ones. This work will be the subject of the thesis of Thomas Jost, which will be co-supervised by Bruno Lévy from the Alice INRIA team and Sylvain Contassot-Vivier from the AlGorille team.

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 investigate the design of a unified framework based on generic programming to achieve a development environment adapted both to multi-core CPUs and GPUs, for linear algebra applied to neutronic computations.

6.1.6. Asynchronism

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

Our first experiments, conducted on an advection-diffusion problem, have shown very interesting results in terms of performances [26] as well as in terms of energetic efficiency (in submission). 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 work in progress.

6.1.7. Heterogeneous Architecture Programming

In 2009 we have designed and experimented several algorithms and applications, in the fields of option pricing for financial computations, PDE solving applied to a 3D transport model simulating chemical species in shallow waters, and generic relaxation methods. In parallel, in collaboration between EDF and SUPÉLEC we investigate the design of a unified framework based on generic programming to achieve a development environment adapted both to multi-core CPUs and GPUs, for linear algebra applied to neutronic computations.

Moreover we have measured computing and energetic performances of our algorithms, in order to compare interests of CPU clusters and GPU clusters, function of the algorithms and application domains.

6.2. Transparent Resource Management

Participants: Louis-Claude Canon, Stéphane Genaud, Emmanuel Jeannot, Tchimou N'takpé.

6.2.1. Reliable Scheduling

We have worked on the case where jobs can fail. More precisely, we have studied the problem of random brokering for platforms that are directly inspired by existing grids such as EGEE. In such environments incoming jobs are randomly dispatched to computational elements with a probability that is proportional to the accumulated speed of this element. Resubmission is a general strategy employed to cope with failures in grids. We have studied resubmission analytically and experimentally for the case of random brokering (jobs are dispatched to a computing elements with a probability proportional to its computing power). We have compared two cases where jobs are resubmitted to the broker or to the computing element. Results show that resubmitting to the broker is a better strategy. Our approach is different from most existing race-based ones as it is a bottom-up: we start from a simple model of a grid and derive its characteristics, see [20].

Also, we have studied the problem of scheduling tasks (with and without precedence constraints) on a set of related processors which have a probability of failure governed by an exponential law. We have provided a general method for converting scheduling heuristics on heterogeneous cluster into heuristics that take reliability into account when precedence there exists constraints. As shown by early experimental results, these heuristics are able to construct a good approximation of the Pareto-Front. Moreover we have studied the case where duplication is allowed. In this case, we have shown that for certain classes of problems, evaluating the reliability of a schedule is #-P Complete.

6.2.2. Robust Scheduling

A schedule is said to be robust if it is able to absorb some degrees of uncertainty in tasks duration while maintaining a stable solution. This intuitive notion of robustness has led to a lot of different metrics but almost no heuristics. We have performed an experimental study of these different metrics and show how they are correlated to each other. Additionally, we have proposed different strategies for minimizing the makespan while maximizing the robustness: from an evolutionary meta-heuristic (best solutions but longer computation time) to more simple heuristics making approximations (medium quality solutions but fast computation time). We have compared these different approaches experimentally and show that we are able to find different approximations of the Pareto front for this bi-criteria problem [14].

Computing the makespan distribution when task and communication duration are given by probabilistic distribution is a #-P complete problem. We have studied different ways to approximate this problem based on previous results of the literature on the PERT network. For comparing these different methods we have computed the makespan distribution using Monte-Carlo simulation, see [39], [21].

6.2.3. Detecting Collusion in Desktop Grid

By exploiting idle time on volunteer machines, desktop grids provide a way to execute large sets of tasks with negligible maintenance and low cost. Although desktop grids are attractive for cost-conscious projects, relying on external resources may compromise the correctness of application execution due to the unreliability of nodes.

While most of the existing work considers that failures in desktop grid are uncorrelated. We have explored efficient and accurate techniques for representing, detecting, and characterizing the presence of correlated or collusive behavior in desktop grid systems.

Precisely, we consider the most challenging model for threats: organized groups of cheaters that may collude to produce incorrect results. We have proposed two on-line algorithms to detect collusion and to characterize the behavior of participants. Using several real-life traces, we have shown that our approach is accurate and efficient in identifying collusion and in estimating group behavior. **[stale citation canon-jeannot-weismann-ipdps] [Action JG: Réponse 7 décembre.]**

6.2.4. A general methodology for computing the Pareto-front

Optimization problems, such as scheduling, can often be tackled with respect to several objectives. In such cases, there can be several incomparable Pareto-optimal solutions. Computing or approximating such solutions is a major challenge in algorithm design. We have proposed a generalization of the greedy methodology to the multi-objective case. This methodology, called meta-greedy, allows to design a multi-criteria algorithm when a mono-criteria greedy one is known. We have applied our proposed solution to two problems of the literature (a scheduling and a knapsack problem) and we have shown that we can generate, in a single execution a set of non dominated solutions that are close to the set of Pareto-optimal solutions.

6.2.5. Energetic 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*). 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.

We intend to continue our investigations in that domain at different levels. Among them, we can cite the hybrid processing context 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. Also, we would like to determine the best trade-off between energy saving and design/implementation costs.

6.2.6. Load balancing

A load-balancing algorithm based on asynchronous diffusion with bounded delays has been designed to work on dynamical networks. 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. Although this work is still in progress, our first results are very satisfying as the efficiency of our algorithm has been experimentally highlighted in the SimGrid framework.

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 learn. However, using a simple regular decomposition is not sufficient as the obtained sub-domains may have very different learning times. Thus, our work in progress concerns the determination of the best estimator of the learning time of a sub-domain in order to obtain similar learning times of all the sub-domains.

Also, some investigations are done according to the decomposition strategy i.e. the global scheme and its inner choice. Until now, we have opted for an URB (Unbalanced Recursive Bisection) approach. We are currently working on the characterization of the best choice of dimension to divide at each decomposition step.

6.2.7. Fault Tolerance

6.2.7.1. 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. A short description of that project is given in Section 6.2.6.

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 learnings. 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 [46]. 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.

6.2.7.2. Programming model and frameworks for fault tolerant applications

In the framework of the PhD thesis of Constantinos Makassikis, supervised by Stephane 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 and another one for *SPMD* applications including inter-nodes communications. 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 first 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.

6.2.7.3. System-level fault tolerance

The approach of fault tolerance we offer in the P2P-MPI framework is based on replication of computations. We have studied both theoretical and experimental aspects of this approach. Our protocol, which is an adaptation of the active replication principle, incurs an overhead as compared to an execution without replication because extra messages must be sent to replica. Being longer, the execution is more failure-prone

on one hand, while on the other hand, replication insures a greater level of robustness. Hence, there is a trade-off that we have studied to determines the optimal replication degree [16]. Our findings are based on the failure distribution model published by [53], which is based on real traces. Second, we have carried out a number of experiments to assess the overhead of replication. We have shown the cost of replication on varied programs from the Java Grande Forum benchmark and with two NAS benchmarks [25].

6.3. Experimentation Methodology

Participants: Pierre-Nicolas Clauss, Fekari El Medhi, Emmanuel Jeannot, Jens Gustedt, Martin Quinson, Cristian Rosa.

6.3.1. A survey comparing different experimental methodologies

We have worked on classifying and comparing experimental methodologies and tools, see [17]. In order to test or validate a solution, one need to execute a real (or a model of an) application on a real (or a model of an) environment. This leads to four classes of methodologies: *in-situ* where we execute a real application (program, set of services, communications, etc.) on real a environment (set of machines, OS, middleware, etc.), *emulation* where we execute a real application on a model of the environment, *benchmarking* where we execute a model of an application on a real environment and lastly *simulation* where we execute a model of an application on a model of the environment. Such classification is very general and applicable to other domains than large-scale systems. However, we have focused on large-scale systems and compared a set of relevant tools for each of these four classes.

6.3.2. Improvement of the SimGrid tool

This year was the first year of the ANR project centered on SIMGrid, of which we are principal investigator (see 7.2.3). Therefore, we worked on setting up the bases in preparation of the numerous contributions expected from the ANR participants in the next few years. This work on the internals aimed at simplifying the code base, and increasing its modularity.

We also continued our work from the previous years on the tool scalability to make it usable not only by the grid computing community, but also by the peer-to-peer scientists. These various software engineering improvements lead to performance speedup of up to 50%.

We initiated a collaboration with the Reso team of INRIA Rhônes-Alpes to study the possibility to add energy models to the simulator, so that it could be used to assess not only the time an algorithm needs to complete, but also the consumed energy during that process. This work did not lead to any publication so far.

Finally, we started to integrate model-checking facilities to the tools so that it could be used not only for performance assessment (mainly comparing the makespan), but also for correctness assessment (checking for example whether the algorithm contains deadlocks). See Section 6.3.4 for more details.

6.3.3. A Platform Description Archive for Reproducible Simulation Experiments

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

This year, we continued our work on the Platform Description Archive (PDA) which is an effort to make platform descriptions and tools available to users of simulation toolkits. In particular, we developed a tool called simulacrum to generate the platform descriptions needed by simulation users. A publication about this tool is pending approval to the CCGrid conference.

6.3.4. Formal Verification of Distributed Algorithms

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

In a first approach, Sabina Akhtar developed an extension of the PlusCal language [52] defined by Leslie Lamport. The extension is intended for describing and verifying models of distributed algorithms, whereas the original language is geared towards shared-memory concurrent programming. The compiler from this language to the TLA$^+$ tool suite is now operational, and were successfully used for several simple examples. We plan to build upon this tool by studying more complex examples and improving the tool performance by leveraging the partial state ordering specific to distributed systems in order to reduce the combinatorial problem posed by the current exhaustive exploration of the state space.

In a complementary approach, we are interested in adding model checking support to the SIMGrid framework and specifically to the GRAS API for simulating and implementing Grid algorithms. The main difficulties to achieve that goal are to save and restore the state of every user processes to allow the model-checker restoring the state of a process to explore another possible execution branch. This year, we implemented a prototype allowing to save the state of the whole simulation ([35]). This approach is easier to implement, but less effective since the state snapshots are larger, making the state explosion issue (inherent to model-checking) even worse. In order to implement another approach where the state of each process would be saved independently (planed for future work), we had to first do the software re-engineering and simplifications mentioned in 6.3.2.

6.3.5. SMPI

One of the main limitation of SIMGrid is that the application has to be written using a specific interface. One possible solution would be to add a new interface to the simulator matching exactly an existing interface. This year, we continued the work on this topic, 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. We implemented several collective operations such as bcast and reduce. We also setup a testing infrastructure to ensure that these new functionality remain usable in the future. One of the main difficulty we faced was the lack of specification of the MPI primitives themselves, as discussed in [54]. We plan to investigate this semantic specifications in the future as it is of major interest for SMPI itself, but also have strong implications on formal methods applied to parallel and distributed applications.

The work on SMPI is still in progress and did not lead to any publication so far. We plan to instigate some formal collaboration on this between us and the team of Prof. Casanova in the near future.

6.3.6. Wrekavoc

We have extensively tested Wrekavoc to show its accuracy and scalability. We have shown that it is able to correctly emulate a network made of hundreds of nodes where congestion happen [23] **[stale citation canon-dubuisson-gustedt-jeannot-JSS]**.

6.3.7. Aladdin-G5K

Grid'5000 aims at building an experimental Grid platform featuring a total of five thousands CPU cores over nine sites in France. In 2009, a third cluster was installed in Nancy. It is named "*griffon*" and is composed of 92 nodes (with 16 cores and 16 GB of RAM each) and an Infiniband-20G network. To control the power consumption, the first cluster that was installed in Nancy in 2005, "*grillon*", was shut down. The "*Grelon*" cluster (120 quad-core nodes, 2 GB of RAM each) is also operational.

The Grid'5000 Spring School was organized in Nancy from April 7th to April 10th.

6.3.8. Experimental cluster of GPUs

The experimental platform of SUPÉLEC for "GPGPU", see Section 4.3.6, has been greatly improved in 2009. Researchers of SUPÉLEC and AlGorille can now access two 16-nodes GPU clusters, supervised with remote energy monitoring devices. The first cluster was already available in 2008 and is composed of 16 PCs, each one hosting a dual-core CPU and a GPU card: a nVIDIA GeForce 8800 GT, with 512MiB of RAM (on the GPU card). The 16 nodes are interconnected across a devoted Gigabit Ethernet switch. 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 GTX285 GPU card with 1GB of memory. This cluster has a Gigabit Ethernet interconnection network too. Energy consumption of each node of each cluster is now monitored by a Raritan DPXS20A-16 device that continuously measures the electric power consumption (in Watts). Then a Perl script samples these values and computes the energy (Joules or WattHours) consumed by the computation on each node and on the complete cluster (including the interconnection switch). Each cluster is associated to a Raritan DPXS20A-16 device, which can monitor up to 20 nodes.

Also, several parallel applications, with different kind of algorithms, have been developed and evaluated on this platform in 2009. Results have been introduced in scientific conferences and to the European COST-IC0804 about *energy efficiency in large scale distributed systems*.

7. Other Grants and Activities

7.1. Regional initiatives

7.1.1. Lorraine Regional Council

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

7.2. National Initiatives

7.2.1. INRIA 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.

7.2.2. CNRS initiatives, GDR-ASR and specific initiatives

We participate at numerous national initiatives. In the GDR-ASR (architecture, systems, and networks) we take part in RGE action [1] 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.

7.2.3. ANR Initiatives

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

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

[1] *Réseau Grand Est*

7.3. European Initiatives

7.3.1. ComplexHPC

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

7.3.2. 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).

7.3.3. Bilateral Collaborations

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

7.4. International Initiatives

7.4.1. Bilateral Collaborations

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

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

Finally, we also collaborate with David Elizondo (University of Leicester, Great Britain) on linear separability by neural networks.

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.

Emmanuel Jeannot and Martin Quinson are members of the steering committee and the direction committee of the ADT Aladdin-G5K. Emmanuel Jeannot served as head of the Nancy site until September and was then replaced by Martin Quinson is serving as vice-head for the Grid'5000 project. Stéphane Genaud is also member of this Action.

8.1.2. Scientific Expertise

In 2009, Jens Gustedt was a referee and member of the thesis committee of Sascha Hunold, University of Bayreuth, Germany.

In 2009, Stéphane Genaud was member of the thesis committee of Heitem Abbes, University Paris 13, France and University de La Manouba, Tunis, Tunisia

In 2009, Emmanuel Jeannot was member of the thesis committee of Adrien Bellanger (INPL, Nancy)

In 2009, Martin Quinson was a referee and member of the thesis committee of Brice Videau, University of Grenoble, France.

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

In 2009, Sylvain Contassot-Vivier was a member of the thesis committees of Aurélien Vasseur, University of Franche-Comté, and Joris Rehm, University Henri Poincaré - Nancy 1.

8.1.3. Teaching Activities

Martin Quinson is teaching the following modules at ÉSIAL (University Henri Poincaré - Nancy 1): "*C and Shell*", "*Technics and tools for programming*" (1A) and "*Networks and Systems*" (2A). He also participates to the following modules: "*Object oriented programming*" (1A), "*Parallel and Distributed Algorithms*" (3A) and "*Grid and advanced distributed algorithms*" at University Henri Poincaré - Nancy 1. He is also responsible of the specialization "*Distributed Systems and Applications*" of ÉSIAL.

Sylvain Contassot-Vivier is teaching "*Parallel algorithms*" (M1), "*Algorithmic and programming*" (M2), "*Parallelism and Grids*" (M2), "*Communicating systems*" (M2) and "*Networks*" (M2) at the University Henri Poincaré - Nancy 1. He is also responsible of the professional master in "*Networks, Services and Mobility*" at the same university.

Stéphane Vialle is teaching "*Information Systems*" (M1) at SUPÉLEC at Paris, "*Parallel and Distributed Computing, and Computing Grids*" (M2) at SUPÉLEC at Metz, "*Distributed Applications*" (M2) at the Louis Pasteur University of Strasbourg, and "*Concurrent Applications: concepts and tools*" at CNAM (National French Institution for Adult Training Courses) in Lorraine. He taught "*GPU cluster programming*" in the winter CNRS school about high performance computing on hardware accelerators (december 7-11, 2009, Banyuls sur Mer). He is also responsible of the organization of computer science teaching at CNAM in Lorraine.

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) and "*Administration of applications and services*" (Licence pro ASRALL).

8.1.4. Editorial Activities

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

In 2009, Jens Gustedt has served as program committee member of the 18th Euromicro International Conference on Parallel, Distributed and network-based Processing PDP 2010 and of the Fifteenth International Conference on Parallel and Distributed Systems (ICPADS'09).

Martin Quinson was member of the program committee of the Second International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SimuTools'09).

Emmanuel Jeannot was program topic chair of Euro-Par 2009 Euro-Par 2009 and Challenges of Large Applications in Distributed Environments (CLADE) 2009 CLADE 2009. He also was member of the program committee of the High Performance Distributed Computing conference HPDC 2009, Heteropar Heteropar, the International Conference on Parallel Computing ParCo, the 11th IEEE International Conference on High Performance Computing and Communications HPCC, dixième conférence de la Société Française de Recherche Opérationnelle et d'Aide à la Décision Roadef 2009, the second International Conference on

Computer Science and its Applications CSA 2009, the Eighth international conference on parallel processing and applied mathematics PPAM 2009, and of RenPar 2009 Renpar 2009.

Stéphane Genaud was member of the program committee of the second International Symposium on Grid and Distributed Computing GDC 2009.

Stéphane Vialle was one of the two Program Chairs of the 2nd International Workshop on Parallel and Distributed Computing in Finance (PDCoF'2009). Stéphane Vialle was also member of the program committee of the international conference on Architecture of Computing Systems (ARCS 2010), and of the nineteenth RenPar conference ("Rencontres francophones du Parallélisme") (Renpar 2009).

Sylvain Contassot-Vivier has been invited in the editorial board of the seventh international conference on Engineering Computational Technology (ECT2010) as well as in the organizing committee of the 16th Intl. workshop on cellular automata (AUTOMATA'2010).

Lucas Nussbaum was member of the organization and program committee for the Grid'5000 Spring School 2009.

8.1.5. Refereeing

In 2009, members of the team served as referees for the following journals and conferences:

Journals: Future Generation Computer Systems, IEEE Transactions on Neural Networks, Journal of Grid Computing, Journal of SuperComputing, Journal of Parallel and Distributed Computing, Parallel Computing, Software: Practice and Experience, Technique et Science Informatiques

Conferences: ARCS 2010, EuroPar 2009, HPDC 2009, ICANN 2009, ICNAAM 2009, ICPADS 2009, ParCO 2009, PDCoF 2009, PDP 2010, RenPar 2009,

Projects: we also participated to the evaluation process for PhD thesis CIFRE, and INRIA associated teams.

8.1.6. Invitations and participations to scientific events

Stéphane Genaud has been invited to present the article [25] at the High-Performance Grid Computing Workshop (HPGC 2009) at Rome, Italy.

Martin Quinson has given an invited talk on SIMGrid [29] at the 9th IEEE International Conference on Peer-to-Peer systems (P2P'09 September 2009 at Seattle, USA).

Jens Gustedt has given an invited talk at the Computer Science Colloquium of the University of Bayreuth, Germany.

Jens Gustedt and Stéphane Vialle presented the InterCell software suite, see Sec. 4.3.5, at the INRIA booth of SuperComputing '09.

Lucas Nussbaum was invited by Canonical, the commercial sponsor of Ubuntu, to attend and participate in the "*Ubuntu Developer Summit*" for "*Ubuntu Lucid (10.04)*".

Stéphane Vialle was invited to participate to a "*panel discussion on GPU Computing*" at the EuroGPU2009 mini-symposium in ParCo-2009 conference.

9. Bibliography

Major publications by the team in recent years

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

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

[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, vol. 4641, Springer, August 2007, 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", vol. 13, 2006, p. 215-239, http://hal.inria.fr/inria-00000899/en/.

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

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

[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.

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

Year Publications

Doctoral Dissertations and Habilitation Theses

[10] P.-N. CLAUSS. *Algorithmes à front d'onde et accès transparent aux données*, Université Henri Poincaré - Nancy I, 11 2009, http://tel.archives-ouvertes.fr/tel-00432366/en/, Ph. D. Thesis.

[11] T. N'TAKPÉ. *Ordonnancement de tâches parallèles sur plates-formes hétérogènes partagées*, Université Henri Poincaré - Nancy I, 01 2009, http://tel.archives-ouvertes.fr/tel-00385329/en/, Ph. D. Thesis.

Articles in International Peer-Reviewed Journal

[12] J. BAHI, S. CONTASSOT-VIVIER. *Corrections to "basins of attraction in fully asynchronous discrete-time discrete-state dynamic networks"*, in "IEEE Transactions on Neural Networks", vol. 20, n^{o} 8, 2009, p. 1372-1374, http://hal.inria.fr/inria-00430275/en/.

[13] J. BAHI, S. CONTASSOT-VIVIER, M. SAUGET. *An incremental learning algorithm for functional approximation*, in "Advances in Engineering Software", vol. 40, n^{o} 8, 2009, p. 725-730, http://hal.inria.fr/inria-00430288/en/.

[14] 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", 2009, http://hal.inria.fr/inria-00430920/en/.

[15] P.-N. CLAUSS, J. GUSTEDT. *Iterative Computations with Ordered Read-Write Locks*, in "Journal of Parallel and Distributed Computing", 2009, http://hal.inria.fr/inria-00330024/en/, RR-6685.

[16] S. GENAUD, E. JEANNOT, C. RATTANAPOKA. *Fault management in P2P-MPI*, in "International Journal of Parallel Programming", vol. 37, n^{o} 5, 2009, p. 433-461, http://hal.inria.fr/inria-00425516/en/.

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

Articles in National Peer-Reviewed Journal

[18] T. N'TAKPÉ. *Heuristiques d'ordonnancement en deux étapes de graphes de tâches parallèles*, in "Technique et Science Informatiques (TSI)", vol. 28, n^{o} 1, 2009, p. 75-99, http://hal.inria.fr/inria-00125269/en/.

International Peer-Reviewed Conference/Proceedings

[19] L. ABBAS-TURKI, S. VIALLE, B. LAPEYRE, P. MERCIER. *High Dimensional Pricing of Exotic European Contracts on a GPU Cluster, and Comparison to a CPU Cluster*, in "Second International Workshop on Parallel and Distributed Computing in Finance - PDCoF 2009, Italie Rome", 2009, p. Proceedings on CD-ROM (8 pages), http://hal-supelec.archives-ouvertes.fr/hal-00390289/en/.

[20] V. BERTEN, E. JEANNOT. *Modeling Resubmission in Unreliable Grids: the Bottom-Up Approach*, in "Seventh International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks - heteroPar'09, Pays-Bas Delft", 2009, http://hal.inria.fr/inria-00430918/en/BE.

[21] L.-C. CANON, E. JEANNOT. *Precise Evaluation of the Efficiency and the Robustness of Stochastic DAG Schedules*, in "10ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision - ROADEF 2009, France Nancy", 2009, p. 13-24, http://hal.archives-ouvertes.fr/hal-00431193/en/.

[22] 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, Italie Pisa", M. DANELUTTO, T. GROSS, J. BOURGEOIS (editors), 2010, http://hal.inria.fr/inria-00436417/en/, RR-7123.

[23] O. DUBUISSON, J. GUSTEDT, E. JEANNOT. *Validating Wrekavoc: A tool for heterogeneity emulation*, in "Heterogenity in Computing Workshop (HCW 09) in International Parallel and Distributed Processing Symposium, Italie Rome", D. A. BADER (editor), IEEE Computer Society, 2009, p. 1-12, http://hal.inria.fr/inria-00435609/en/.

[24] V. GALTIER, S. GENAUD, S. VIALLE. *Implementation of the AdaBoost Algorithm for Large Scale Distributed Environments: Comparing JavaSpace and MPJ*, in "Fifteenth International Conference on Parallel and Distributed Systems (ICPADS'09), Chine Shenzhen", IEEE CS, 2009, http://hal.inria.fr/inria-00425518/en/.

[25] S. GENAUD, C. RATTANAPOKA. *Evaluation of Replication and Fault Detection in P2P-MPI*, in "6th High Performance Grid Computing International Workshop in conjunction with International Parallel and Distributed Processing Symposium - IPDPS 2009, Italie Rome", IEEE CS, 2009, p. 1-8, http://hal.inria.fr/inria-00425519/en/.

[26] 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/.

[27] W. KIRSCHENMANN, L. PLAGNE, S. PLOIX, A. PONÇOT, S. VIALLE. *Massively Parallel Solving of 3D Simplified PN Equations on Graphics Processing Units*, in "2009 International Conference on Mathematics Computational Methods and Reactor Physics - M&C 2009, États-Unis d'Amérique Saratoga Springs, New York", 2009, p. Proceedings on CD-ROM (12 pages), http://hal-supelec.archives-ouvertes.fr/hal-00390291/en/.

[28] W. KIRSCHENMANN, L. PLAGNE, S. VIALLE. *Multi-target C++ implementation of parallel skeletons*, in "POOSC'09, Italie Genova", 2009, p. 1-10, http://hal-supelec.archives-ouvertes.fr/hal-00437542/en/.

[29] M. QUINSON. *SimGrid: a Generic Framework for Large-Scale Distributed Experiments*, in "9th International conference on Peer-to-peer computing - IEEE P2P 2009, États-Unis d'Amérique Seattle", IEEE, 2009, http://hal.inria.fr/inria-00435802/en/.

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

[31] P. VEZOLLE, S. VIALLE, X. WARIN. *Large Scale Experiment and Optimization of a Distributed Stochastic Control Algorithm. Application to Energy Management Problems*, in "International workshop on Large-Scale Parallel Processing 2009 part of IPDPS, Italie Rome", IEEE, 2009, p. Proceedings on CD-ROM (8 pages), http://hal-supelec.archives-ouvertes.fr/hal-00390290/en/.

Workshops without Proceedings

[32] D. BOUMBA SITOU, S. OULD SAAD HAMADY, N. FRESSENGEAS, H. FREZZA-BUET, S. VIALLE, J. GUSTEDT, P. MERCIER. *Cellular based simulation of semiconductors thin films*, in "Innovations in Thin Film Processing and Characterization - ITFPC 09, France Nancy", 2009, http://hal.archives-ouvertes.fr/hal-00433062/en/.

[33] S. CONTASSOT-VIVIER, S. VIALLE, T. JOST. *Algorithmes itératifs asynchrones et grappe de GPUs*, in "Journée jeunes chercheurs sur les Multiprocesseurs et Multicoeurs, France Paris", 2009, http://hal.inria.fr/inria-00430524/en/.

[34] T. JOST, S. CONTASSOT-VIVIER, S. VIALLE. *Solveur linéaire sur GPU*, in "Journée jeunes chercheurs sur les Multiprocesseurs et Multicoeurs, France Paris", 2009, http://hal.inria.fr/inria-00430529/en/.

[35] C. ROSA, M. QUINSON, S. MERZ. *Model-checking Distributed Applications with GRAS*, in "Exploiting Concurrency Efficiently and Correctly - EC 2 workshop associated to CAV 2009, France Grenoble", 2009, http://hal.inria.fr/inria-00378374/en/.

Scientific Books (or Scientific Book chapters)

[36] S. GENAUD, C. RATTANAPOKA. *A Peer-to-Peer Framework for Message Passing Parallel Programs*, in "Parallel Programming, Models and Applications in Grid and P2P Systems", F. XHAFA (editor), vol. 17, IOS Press, 2009, p. 118–147, http://hal.inria.fr/inria-00425484/en/.

[37] C. GERMAIN-RENAUD, V. BRETON, P. CLARYSSE, B. DELHAY, Y. GAUDEAU, T. GLATARD, E. JEANNOT, Y. LEGRÉ, J. MONTAGNAT, J.-M. MOUREAUX, A. OSORIO, X. PENNEC, J. SCHAERER, R. TEXIER. *Grid analysis of radiological data*, in "Handbook of Research on Computational Grid Technologies for Life Sciences, Biomedicine and Healthcare", M. CANNATARO (editor), Medical Information Science Reference, 2009, ., http://hal.archives-ouvertes.fr/hal-00382594/en/.

[38] J. GUSTEDT. *Generalized Attachment Models for the Genesis of Graphs with High Clustering Coefficient*, in "Complex Networks - Results of the 2009 International Workshop on Complex Networks (CompleNet 2009)", S. FORTUNATO, G. MANGIONI, R. MENEZES, V. NICOSIA (editors), vol. 207, Springer Berlin / Heidelberg, 2009, p. 99-113, http://hal.inria.fr/inria-00312059/en/, RR-6622.

Research Reports

[39] L.-C. CANON, E. JEANNOT. *Precise Evaluation of the Efficiency and the Robustness of Stochastic DAG Schedules*, 2009, http://hal.inria.fr/inria-00372651/en/, RR-6895, Rapport de recherche.

[40] C. ROSA, M. QUINSON, S. MERZ. *Model-checking Distributed Applications with GRAS*, 2009, http://hal.inria.fr/inria-00422159/en/, RR-7052, Rapport de recherche.

Other Publications

[41] L. ABBAS-TURKI, S. VIALLE. *European Option Princing on a GPU Cluster*, 2009, http://hal-supelec.archives-ouvertes.fr/hal-00364242/en/.

References in notes

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

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

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

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

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

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

[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] 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.

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

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

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

[53] D. NURMI, J. BREVIK, R. WOLSKI. *Modeling Machine Availability in Enterprise and Wide-Area Distributed Computing Environments*, in "Euro-Par 2005, Parallel Processing, 11th International Euro-Par Conference", Lecture Notes in Computer Science, vol. 3648, Springer, September 2005, p. 432-441.

[54] T. SAIF, M. PARASHAR. *Understanding the Behavior and Performance of Non-blocking Communications in MPI*, in "Euro-Par", 2004, p. 173-182.

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