INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team AlGorille

Algorithms for the Grid

Lorraine

THEME NUM

Table of contents

1. Team

Team Leader
Jens Gustedt [research director, INRIA, HdR]

Administrative Assistant
Josiane Reffort [UHP]

Staff Members
Emmanuel Jeannot [[Temporary assignment as Chargé de Recherche, INRIA, between 01/01/06 and 31/08/06. Chargé de Recherche, INRIA, since 01/09/05]
Martin Quinson [Maître de Conférences, UHP/ESIAL]
Frédéric Suter [Maître de Conférences, UHP]

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

Engineer
Xavier Delaruelle [Ingénieur associé, INRIA]
Abdelmalek Cherier [Ingénieur associé, INRIA, since 01/10/06]

Teaching Assistant
Luiz Angelo Steffenel [Nancy 2]
Pierre-François Dutot [UHP/ESIAL, until 31/08/06]
Flavien Vernier [UHP, until 31/08/06]

Ph. D. Students
Tchimou N'Takpé [joint regional/Ivorian goverment grant since 01/10/05]
Pierre-Nicolas Clauss [Allocataire de recherche MENESR since 01/10/06]

Student Intern
Pierre-Nicolas Clauss [M2R student between 01/02/2006 and 01/08/2006]
Ahmed Harbaoui [M2R student between 01/02/2006 and 01/08/2006]
Christophe Thiery [Internship between 12/06/2006 and 18/08/2006]

2. Overall Objectives

2.1. Overall Objectives

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

The possible access to distributed computing resources over the Internet allows a new type of 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.
- Experimental validation: reproducibility, extendability and applicability of simulations, emulations and *in situ* experiments.

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

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

3. Scientific Foundations

3.1. Structuring of Applications for Scalability

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

Participants: Jens Gustedt, Frédéric Suter, Stéphane Vialle.

Our approach is based on a "good" separation of the different problem levels that we encounter with Grid problems. Simultaneously this has to ensure a good data locality (a computation will use data that are "close") and a good granularity (the computation is divided into non preemptive tasks of reasonable size). For problems for which there is no natural data parallelism or control parallelism such a division (into data and tasks) is indispensable 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 [50]. Their common characteristics are:

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

The coarse grained models aim at being realistic with regard to two different aspects: algorithms and architectures. In fact, the coarseness of these models uses the common characteristic of today's parallel settings: the size of the input is orders of magnitude larger than the number of processors that are available. In contrast to the PRAM (Parallel Random Access Machine) model, the coarse grained models are able to integrate the cost of communications between different processors. This allows them to give realistic predictions about the overall execution time of a parallel program. As examples we refer to BSP (Bulk Synchronous Parallel model) [50], LogGP (Latency overhead gap Procs) [46], CGM (Coarse Grained Multicomputer) [48] and *PRO* (Parallel Resource Optimal Model) [6].

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

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

We are convinced that this new paradigm should:

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

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

A first solution that uses (1) to combine these objectives for homogeneous environments has been given in [6] with PRO.

In a complementary approach we have addressed (2) to develop a simple interface that gives a consistent view of the data services that are exported to an application, see [8].

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

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

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

3.2. Transparent Resource Management

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

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

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 allow 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 exiting ones) *for* the resource management in middlewares. Our objective is to provide these algorithms in libraries so that they may be easily integrated. For instance we will propose algorithms to efficiently transfer data (data compression, distribution or redistribution of data) or schedule sequential or parallel tasks.

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

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

3.3. Experimental Validation

Keywords: *applicability*, *emulations*, *experiments in situ*, *extendability*, *reproductibility*, *simulations*.

Participants: Malek Cherier, Xavier Delaruelle, Emmanuel Jeannot, Martin Quinson.

An important issue for the research on complex systems such as grids is to validate the obtained results. This validation constitutes a scientific challenge by itself since we have to validate models, their adequation 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 or performance, this might be due to several factors: a faulty modeling, a weak design, or a bad implementation.

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

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

We think that an experimental validation is a valuable completion of theoretical results on protocol and algorithm behavior. The focus of algorithmic research being upon performance, the main experiments we are concerned with are performance evaluation. 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* target possible comparisons with passed and (in particular) future work, 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: Jens Gustedt, Frédéric Suter, Pierre-Nicolas Clauss.

4.1.1. Models and Algorithms for Coarse Grained Computation

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

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

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

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

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

List ranking, tree contraction and graph coloring algorithms already have been designed following the coarse grained setting given by the model *PRO* [6].

To work in the direction of understanding of what problems might be "*hard*" we tackled a problem that is known to be P-complete in the PRAM/NC framework, but for which not much had been known when only imposing the use of relatively few processors: the *lexicographic first maximal independent set* (LFMIS) problem [9].

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

4.1.2. External Memory Computation

In the mid-nineties several authors [45], [47] 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.

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

In fact, the stability of our library *parXXL* (formerly *SSCRAP*), see Section 5.1, also showed in its extension towards external memory computing [7]. *parXXL* nas 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

In [2], we develop a pipeline algorithm aware of the use of external memory to store the handled data. The originality of our approach is to overlap computation, communication, and IO through an original strategy using several memory blocks accessed in a cyclic manner. The resulting pipeline algorithm achieves a saturation of the disk resource which is the bottleneck in algorithms relying on external memory.

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 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 [6] and the *parXXL* library we already provide tools to better design algorithms (and implement them afterwards) that are adapted to these irregular problems.

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

4.2. Evolution of Scheduling Policies and Network Protocols

Participants: Emmanuel Jeannot, Frédéric Suter, Pierre-François Dutot, Tchimou N'Takpé, Luiz Angelo Steffenel.

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.

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. Performances of such environments greatly depend on the scheduling heuristic implemented. In these systems, a server executes each request as soon as it has been received: it never delays the start of the execution.

In order for a such a system to be efficient, the mapping function must choose a server that fulfills several criteria. First, the total execution time of the client application, e.g. the makespan, has to be as short as possible. Second, each request of every clients 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.

4.2.2. Parallel Task Scheduling

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

A well-known challenge for the efficient execution of task-parallel applications is scheduling. The problem consists in deciding which compute resource should perform which task when, in a view to optimizing some metric such as overall execution time. In the case of mixed-parallel applications, data parallelism adds a level of difficulty to the task-parallel scheduling problem. Indeed, the common assumption is that data-parallel tasks are moldable, i.e., they can be executed on arbitrary numbers of processors, with more processors leading to faster task execution times. This is typical of most mixed-parallel applications, and raises the question: how many processors should be allocated to each data-parallel task? There is thus an intriguing tension between running more concurrent data-parallel tasks with each fewer processors, or fewer concurrent data-parallel tasks with each more processors. Not surprisingly this scheduling problem is NP-complete. Consequently, several researchers have attempted to design scheduling heuristics for mixed-parallel applications. The most successful approaches proceed in two phases: one phase to determine how many processors should be allocated to each data-parallel task, one phase to schedule these tasks on the platform using standard list scheduling algorithms.

A limitation of these two-phase scheduling algorithms is that they assume a homogeneous computing environment. While homogeneous platforms are relevant to many real-world scenarios, heterogeneous platforms are becoming increasing common and powerful. Indeed, in the face of increasing computation and memory demands of scientific application, many current computing platforms consist of multiple compute clusters aggregated within or across institutions. Mixed parallel applications appear then ideally positioned to take advantage of such large-scale platforms. However, the clusters in these platforms are rarely identical. Because deployed by different institutions at different times, they typically consist of different numbers of different compute nodes (e.g., there can be large slow clusters and small fast clusters).

Two approaches can be followed to schedule mixed-parallel applications on heterogeneous platforms. The first approach consists in adapting the aforementioned two-phase algorithms for mixed-parallel applications on homogeneous platforms and making them amenable to heterogeneous platforms. The second approach consists in adapting list scheduling algorithms that were specifically designed for executing task-parallel applications

on heterogeneous platforms and making them amenable to mixed parallelism [4]. Both approaches have merit and an interesting question is: is one approach significantly better than the other, and if so, which one?

4.2.3. Data Redistribution Between Clusters

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

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

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

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

4.2.4. Dynamic and Adaptive Compression of Network Streams

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

A solution of this problem is a adaptive service which offers the possibility to transfer data while compressing it. The compression level is dynamically changed according to the environment and the data. The adaptation process is required by the heterogeneous and dynamic nature of grids. For instance if the network is very fast, time to compress the data may not be available. But, if the visible bandwidth decreases (due to some congestion on the network), some time to compress the data may become available.

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

4.3. Providing Environments for Experiments

Participants: Xavier Delaruelle, Jens Gustedt, Emmanuel Jeannot, Martin Quinson.

4.3.1. Simulating Grid Platforms

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

4.3.2. Emulating Heterogeneity

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

4.3.3. Grid'5000

We participate to the development of the Grid'5000 platform. Its purpose 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, controlable 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 about one hundred processors to the community.

5. Software

5.1. parXXL

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

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

The integration, testing and benchmarking of *parXXL* started as a joint effort with Amelia De Vivo, university of Basilicata, Ponteza, Italy, until her sudden and unexpexted death in June 2006.

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

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

All three different realizations of the communication layers are quite efficient. They let us execute programs that are otherwise unchanged within the three different contexts such that they reach or maybe outperform programs that are directly written for them.

5.2. AdOC

Participant: Emmanuel Jeannot.

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

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

5.3. Wrekavoc

Participant: 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 set of configurations using real applications with no emulation of the code. Given an homogeneous cluster Wrekavoc degrades the performance of nodes and network links independently in order to build a new heterogeneous cluster. Then, any application can be run on this new cluster without modifications.

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

CPU Degradation. We have implemented several methods for degrading CPU performance. The first approach consists in managing the frequency of the CPU through the kernel CPU-Freq interface. We propose two other solutions in case CPU-Freq is not available. One is based on CPU burning. A program that runs under real-time scheduling policy burns a constant portion of the CPU, whatever the number of processes currently running. The other is based on user-level process scheduling called CPU-lim. A CPU limiter is a program that supervises processes of a given user. 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 tools it is possible to control both incoming and outgoing traffic. Furthermore, the latest versions (above 2.6.8.1) allows to control the latency of the network interface.

Memory Limitation. Wrekavoc is able to limit the largest malloc a user can make. This is possible through the security module PAM. However, we have not been able to limit the whole memory usable by all the processes yet.

Configuring and Controlling Nodes and Links. The configuration of a homogeneous cluster is made through the notion of islet. An islet is a set of nodes that share similar limitation. 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.

We have also Wrekavoc on the Grid-explorer cluster with 216 nodes. Each node has two 2 GHz AMD Opteron 246 with 2 GB of RAM. It runs under Linux Debian 3.1 with kernel 2.6.8. Results show that we are able to independently degrades each resources and that the obtained degradation is very close to the desired one.

5.4. GRAS

Participant: Martin Quinson.

The GRAS *(Grid Reality And Simulation) framework eases the development of Grid services and infrastructures.*

GRAS provides a C ANSI interface to build distributed services and infrastructure 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 9 hardware architectures. Services built on top of this achieve better communication performance than heterogeneous implementations of the MPI protocol.

This tool is now integrated into the SIMGRID framework, and available from http://gforge.inria.fr/projects/simgrid/.

5.5. LaPIe

Participant: Luiz Angelo Steffenel.

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

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

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

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

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

LAPIE is available at http://libresource.inria.fr/projects/LaPIe.

6. New Results

6.1. Structuring of Applications for Scalability

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

6.1.1. Large Scale Experiments

The merge of the two libraries *ParCeL* and *SSCRAP* into the new software suite *parXXL* has been accomplished this year. It consists of different toolboxes, `par::cpp` (interfaces for the C++ language), `par::sys` (interfacing POSIX systems), `par::mem` (tools for managing memory), `par::step` (manage supersteps), `par::cell` (management of cellular networks) and `par::cellnet` (defining default network types).

The integration of the formerly seperated libraries allows to validate the whole on a wide range of fine grained applications and problems. A report on the design and the first benchmarking of the integrated code can be found in [27].

Now that the communication layer of *parXXL* can handle large numbers of POSIX threads (shared memory) or distributed processes (MPI), we were able to run large scale experiments on mainframes and clusters. These have proved the scalability of our approach as a whole, including engineering, modeling and algorithmic aspects: the algorithms that are implemented and tested show a speedup that is very close to the best possible theoretical values, and these speedups are reproducible on a large variety of platforms, see [25].

6.1.2. Models and Algorithms for Coarse Grained Computation

We continued the design of algorithms in the coarse grained setting as given by the model *PRO* [6]. In particular we aimed for the design of an algorithm that takes advantage of the structure commonly encountered with massive graphs, namely the fact that they usually have a bounded arboricity. There we gave algorithms for computing probability vectors that can be used for the clustering of communities, see [26].

For testing and benchmarking the generation of large random input data with known probability distributions is crucial. In [20], we show how to uniformly distribute data at random in two related settings: *coarse grained parallelism* and *external memory*. In contrast to previously known work for parallel setups, our method is able to fulfill the three criteria of uniformity, work-optimality and balance among the processors simultaneously. To guarantee the uniformity we investigate the matrix of communication requests between the processors. We show that its distribution is a generalization of the multivariate hypergeometric distribution and we give algorithms to sample it efficiently in the two settings.

6.1.3. Overlapping Computations and Communications with I/O

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

Using the *parXXL* library we were able to propose a architecture-independant out-of-core implementation of a well known hydrodynamics kernel, see [40]. In addition to this implementation we proposed a optimized data layout that allow to reduce the I/O impact on each iteration of the algorithm by an order of magnitude at the cost of an initial rewriting of the data. This work is currently under submission, see [41].

6.2. Transparent Ressource Management

Participants: Pierre-François Dutot, Emmanuel Jeannot, Tchimou N'Takpé, Luiz Angelo Steffenel, Frédéric Suter.

6.2.1. Scheduling under Uncertainty

When scheduling a set of task modeled by a DAG, due to runtime variation, the actual makespan can be different than the makespan computed by the scheduling algorithm. A schedule is said *robust* when this variation is not too large, *i.e.* when the schedule is not too sensible to runtime variation.

We have addressed the problem of matching and scheduling of DAG-structured application to both minimize the makespan and maximize the robustness in a heterogeneous computing system. Due to the conflict of the two objectives, it is usually impossible to achieve both goals at the same time. We have given two definitions of robustness of a schedule based on tardiness and miss rate. Slack was proved to be an effective metric to be used to adjust the robustness. We have employed ϵ-constraint method to solve the bi-objective optimization problem where minimizing the makespan and maximizing the slack are the two objectives. We have defined the overall performance of a schedule considering both makespan and robustness such that user have the flexibility to put emphasis on either objective. Experiment results have validated the performance of the proposed algorithm.

6.2.2. Parallel Task Scheduling

When conducting our initial experimental comparison of M-HEFT [4] and the HCPA heuristic developed during the master thesis of Tchimou N'Takpé we identified several limitations in both algorithms. This year we proposed improvements to address these limitations. The allocation phase of HCPA has been improved by proposing a new stopping criterion that allows smaller but still efficient allocations. We also introduced

a packing strategy in order to fill gaps that may appear in its placement phase as a task may be delayed unnecessarily just because its computed processor allocation is (perhaps only slightly) larger than the number of processors available at the time when the task is ready for execution.

We also address a glaring drawbacks of M-HEFT, which is that it tends to use very large processor allocations for application tasks. This is simply due to the fact that a task's processor allocation is chosen "blindly" so that the task's completion time is minimized. To remedy this problem with M-HEFT we propose three simple ways to bound a task's processor allocation.

Part of these research have been published in [31], [30], and a comparison between improved versions of our heuristics is under submission.

Finally we are currently on the implementation of a guaranteed heuristic in collaboration with Henri Casanova, at University of Hawai'i, Manoa and Pierre-Fran cois Dutot. An optimal allocation is computed by a linear program and a list scheduling algorithm is then used to place these task's allocation.

6.2.3. Data Redistribution

Various redistribution algorithm of the literature have been implemented and tested this year. Indeed, many algorithms have been proposed to redistribute data on a same cluster. However, no fair comparison exists between these algorithms. Moreover, some of them can easily be extended to solve the KPBS problem that consists in redistributing data between clusters over a backbone.

We have carried out experiments on the grid explorer cluster and on grid 5000 between the Orsay site and the Rennes site as well as on a singe cluster.

Surprisingly, on the machines we tested, we have found that avoiding contention is not always useful. Indeed, in most of the cases, the brute-force method is the fastest way to redistribute data from a block-cyclic distribution to another block-cyclic distribution. This result is mainly due to the fact that contention does not degrade the performance of the networks we have used. However, in the case where the pattern is irregular OGGP is the best scheduling algorithm. We also showed that preemption is useful only if its cost is taken into account by the algorithm.

In conclusion, if performance is the only issue, the brute-force method is often the best one. However, if other issues have to be considered (QOS, memory constraints, predictability and stability), scheduling algorithms such as OGGP are a very good options.

6.2.4. Performance Prediction

Being able of accurately estimating the runtime of a program and communication time of data transfer is critical for efficiently scheduling application on distributed environments such as grids.

We have introduced a template based modeling mechanism that is able to accurately predict the runtime of the service based on previous execution. It improves the standard runtime estimation of GridSolve as it is more accurate and takes into account the specificity of the service and the machine it runs on. Second, we have developed an estimator of the communication cost between the client and the server. Since communication cost is often very large such an estimator enables to discard fast remote server if the gain in terms of computation time is overshadowed by the communication time.

We have also worked on modeling the dense LU factorization in order to predict the runtime on a parallel machine. With this model we are able to predict a block-size close to the optimal for a given size of the matrix and a given number of processors

6.2.5. Total Exchange Performance Prediction

One of the most important collective communication patterns for scientific applications is the total exchange (also called All-to-All, in which each process holds n different data items of size m that should be distributed among the n processes, including itself. However, this communication pattern tend to saturate network resources, causing unexpected transmission delays - the network contention.

Having accurate predictions is extremely important on the development of application performance prediction frameworks such as PEMPIs [49] and GridSolve. Because it is not always possible to use contention-aware All-to-All implementations (as in the case of popular MPI libraries), it is important to design performance models that take into account the effects of network contention.

Studying the effects of the network contention in the context of MPI programming environments, we introduced a new approach to model the performance of the All-to-All collective operation. Contrarily to existing models which rely on complex interference analysis, our strategy consists in identifying, based on a sample execution, a contention signature that characterizes a given network environment. Using such method we were able to accurately predicted the performance of the All-to-All operation on different network architectures (Fast Ethernet, Gigabit Ethernet and Myrinet, for example), as illustrated in our paper [35].

6.2.6. Grid-aware Total Exchange

As presented above, Total Exchange algorithms (also called All-to-All) are widely studied in the context of (partially) homogeneous clusters subjected to network contention. Only a few works try to optimize the execution of such communication patterns on grid environments, and up to now the results are far from being widely spread. Indeed, heterogeneity of the communication environment turns the optimization of the All-to-All operation into a NP-hard problem.

Based on preliminary experiments conducted by [14], we were able to implement on LAPIE some scheduling heuristics that are efficient for small messages (or better saying, for strongly heterogeneous environments. Nevertheless, these heuristics fail with large messages as they are unable to improve the utilisation of the wide-area bandwidth. For instance, we are currently observing the impact of different implementation algorithms from popular MPI distribution such as MPICH and OpenMPI on the communication schedule, and trying to figure out the causes of low-bandwidth utilisation sometimes observed with these algorithms. The next step will consist on developing specific heuristics to circumvent these restriction. They should be tested in both simulated and real environments, using respectively GRAS/MSG (or SMPI, if available) and LAPIE.

6.3. Experimental Validation

Participants: Malek Cherier, Xavier Delaruelle, Ahmed Harbaoui, Emmanuel Jeannot, Martin Quinson, Christophe Thiery.

6.3.1. Improvement of the SimGrid tool

The goal of the SIMGRID tool suite is to allow the study and development of distributed application on modern platforms. It is the result of a collaboration with Henri Casanova (Univ. of Hawaii, Manoa) and Arnaud Legrand (MESCAL team, INRIA Rhône-Alpes, France). Simulation is a common answer to the grid specific challenges such as scale and heterogeneity. SIMGRID is one of the major simulator in the Grid community.

This year, Malek Cherrier was hired as an engineer on an ODL contract (Opération de Développement Logiciel – software development operation) granted by INRIA, allowing us to assess the technical issues posed by the development of the tool. We completed the port SIMGRID to the Windows platforms, increasing the possible user base. We also worked on an automated testing infrastructure to ensure the software quality of the product. This step was necessary to stabilize the existing code base prior to the add of new functionalities, planed for the future.

In the same time, Christophe Thiery enhanced the simulator to add a new interface called SimDag. It allows to easily express applications modeled by DAG of tasks. Such an interface was present in SIMGRID version 2, but it were not ported yet into the SIMGRID 3 framework. This functionality helps the work on parallel task scheduling achieved within the team (*cf.* 6.2.2) and above.

SIMGRID is freely downloadable [44] and its user base is rapidly growing, resulting in the publication of about ten publications (half of them from users not being part of the core team).

6.3.2. Grid Platform Discovery

Due to the changing characteristics of the Grids, distributed applications targeting these platforms must be network-aware and react to the condition changes. To make this possible, applications must have a synthetic view of the network condition they experiment. Several platform monitoring tools exist, but they provide irrelevant or incomplete information to network-aware applications. Most of these tools intend to help the network administrator to detect abnormalities in their system. They thus concentrate on very low level metrics such as the amount of data emited by a given host where network-aware application need to access the available bandwidth between host pairs. Some tools were designed specifically to provide such higher-level information (the most predominant being NWS – [51]), but they are limited to quantitative information about the bandwidth, latency and processor availability.

We designed a tool to discover the network topology from an application point of view. We are mainly interested in predicating the effect of resource sharing between concurrent data stream. This information is for example crucial to schedule individual messages of group communications or to compute the optimal localization of backup servers and storage areas.

Testing and comparing the different possible heuristics to address this problem is difficult since it comes done to assessing how similar the discovered graph is from the real platform topology. We designed a testing framework on simulator, allowing to do so by comparing the performance of classical applications both on the discovered platform and on the real one. This comparison metric thus captures how the discovered platform matches the real one *from the application point of view*.

We compared several heuristics presented theoretically in the literature, and plan to improve them in a near future [43].

6.3.3. Grid'5000

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

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

Each Grid'5000 site aims at providing at least five hundreds CPUs. We have designed our needs for a second cluster and conduct its purchase. This new machine is a 120-nodes HP cluster. Each compute node has two 1.6 GHz Dual-core Intel Xeon 5110 with 2 GB of RAM and two gigabit Ethernet interfaces. We plan to receive and install this cluster at the end of the year.

We take a significant part in the national development of the Grid'5000 platform. We help consolidating the production infrastructure, by developing tools like the account management software called cpu-g5k (https://gforge.inria.fr/projects/grid5000/) or creating Linux-based grid environment for users. We also participate at most of the existing workgroups of the project, like the one for the next Kadeploy version (https://gforge.inria.fr/projects/kadeploy/https://gforge.inria.fr/projects/kadeploy/). Moreover we take in charge some of the collaborative services like the wiki website and the bug reporting tool.

7. Other Grants and Activities

7.1. National Initiatives

7.1.1. CNRS initiatives, GDR-ARP and specific initiatives

We participate at numerous national initiatives. In the GDR-ASR (architecture, systems, and networks) we take part in TAROT[1], Grappes[2], and RGE[3]. We also participate to the animation of the GDR-ASR as a whole.

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

7.1.2. ACI initiatives of the French Research Ministry

We are partners in several projects of different ACI initiatives:

- Grid Explorer. We participate with a joint proposition together with Stéphane Vialle from Supélec, Metz Campus, which concerns the integration tests of *SSCRAP* and Parcel-6 as described in Section 6.3. We also work on designing a set of emulation tools for transforming an homogeneous platform into an heterogeneous one, see Section 5.3.
- In the 2004 initiative ACI AGIR we participate in the definition and design of a set of services for medical image processing on the grid. More precisely we are in charge of transfer with compression task and the evaluation of grid middleware.

7.1.3. ARA Initiatives of the French Research Ministry

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

- ALPAGE.

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

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

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

7.1.4. INRIA New Investigation Grant

The goal of the ARC OTaPHe is to federate conceptual and experimental researches around parallel task scheduling conducted by the AlGorille, GRAAL, MESCAL and MOAIS INRIA teams. Our approach consists in defining models taking computational grid heterogeneity into account. These models however have to remain simple. From those models guaranteed heuristics will be designed and implemented into the DIET and OAR middlewares in order to validate them.

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

7.2. European Initiatives

7.2.1. NoE CoreGrid

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

7.2.2. Bilateral Collaborations

We maintain several european collaborations with other research teams. The two most fruitful are with the team of Jan Arne Telle from Bergen University, Norway, and with Vandy Berten and Joël Goossens of the Université Libre de Bruxelles on scheduling problems under stochastic models.

7.3. International Initiatives

7.3.1. NSF-INRIA Grant

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

7.3.2. Bilateral Collaborations

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

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

7.4. Visits

From january to july 2006, Emmanuel Jeannot spent 6 month at the University of Tennessee under the NSF-INRIA Grant described above.

8. Dissemination

8.1. Dissemination

8.1.1. Leadership within the Scientific Community

Jens Gustedt is elected member of INRIA scientific board.

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

8.1.2. Scientific Expertise

In 2006, Jens Gustedt was a member of the thesis committee of Timothée Bossart, University Paris 6 and has served as an external expert for the evaluation of scientific projects in regional initiatives for information science and technology in a neighboring European country.

8.1.3. Teaching Activities

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

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

Luiz Angelo Steffenel is teaching the following modules at IUT Nancy Charlemagne (Nancy 2 University): *Introduction à l'Algorithmique* (DUT 1A), *Bases de la Programmation - Java* (DUT 1A), *Systèmes de Gestion de Bases de Données* (DUT 2A), *Architecture 1* (DUT 1A Bis), *Administration de Systèmes de Gestion de Bases de Données* (DUT AS). He also participated to the following modules at the UFR Mathématique et Informatique (Nancy 2 University): *Programmation C Avancée* (MIAGE 3A), *Certificat C2I* (1A).

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 an 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. In 2006, in addition to its journal activities DMTCS has published two conference proceedings.

In 2006, Jens Gustedt has served as program committee member of the 2006 International Conference on Parallel Processing ICPP '06 and for the special volume of *Parallel Computing* on large scale grids.

Emmanuel Jeannot, Jens Gustedt and Stéphane Vialle have been members of the program committee of RenPar 2006 (17ème rencontre du paralléllisme).

8.1.5. Refereeing

In 2006, members of the team served as referees for the following journals and conferences:

Journals: Advances in Engineering Software and Computer and Structures Journal Discrete Applied Mathematics, IEICE transactions, IEEE Transactions on Parallel and Distributed Systems, International Journal of High Performance Computing, Journal of Parallel Distributed Computing, Parallel Computing, Journal of Scientific Programming

Conferences: CARI 2006, e-Science 2006 EuroPar 2006, HCW 2007, ICON 2006, ICPP 2006, IPDPS 2006, RenPar 2006, SIMS 2006

9. Bibliography

Major publications by the team in recent years

[1] Y. CANIOU, E. JEANNOT. *New Dynamic Heuristics in the Client-Agent-Server Model*, in "IEEE Heterogeneous Computing Workshop - HCW'03, Nice, France", April 2003.

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

[3] E. CARON, F. SUTER. *Parallel Extension of a Dynamic Performance Forecasting Tool*, in "Accepted for publication in Parallel and Distributed Computing Practice (PDCP)", Special issue on selected papers of ISPDC'02, 2004.

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

[5] J. COHEN, E. JEANNOT, N. PADOY. *Messages Scheduling for Data Redistribution between Clusters*, in "Algorithms, models and tools for parallel computing on heterogeneous network - HeteroPar'03, workshop of SIAM PPAM 2003, Czestochowa, Poland", September 2003.

[6] A. GEBREMEDHIN, I. GUÉRIN LASSOUS, J. GUSTEDT, J. A. TELLE. *PRO: A Model for the Design and Analysis of Efficient and Scalable Parallel Algorithms*, 2005, http://hal.inria.fr/inria-00000899/en/.

[7] 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.

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

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

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

[11] E. JEANNOT, B. KNUTTSON, M. BJORKMAN. *Adaptive Online Data Compression*, in "Eleventh IEEE International Symposium on High Performance Distributed Computing - HPDC 11, Edinburgh, Scotland", IEEE, July 2002.

[12] E. JEANNOT, F. WAGNER. *Two fast and efficient message scheduling algorithms for data redistribution through a backbone*, in "18th International Parallel and Distributed Processing Symposium - IPDPS'04, Santa Fe, New Mexico", IEEE, Apr 2004.

[13] L. A. STEFFENEL, G. MOUNIÉ. *Scheduling Heuristics for Efficient Broadcast Operations on Grid Environments*, in "Proceedings of the Performance Modeling, Evaluation and Optimization of Parallel and Distributed Systems Workshop - PMEO'06 (associated to IPDPS'06), Rhodes Island, Greece", IEEE Computer Society, Apr 2006, http://hal.archives-ouvertes.fr/hal-00022008.

[14] L. A. STEFFENEL. *LaPIe: communications collectives adaptées aux grilles de calcul*, Ph. D. Thesis, INPG, France, 2005, http://tel.ccsd.cnrs.fr/tel-00011603.

[15] L. A. STEFFENEL. *Modeling Network Contention Effects on AlltoAll Operations*, in "Proceedings of the IEEE Conference on Cluster Computing (CLUSTER 2006), Barcelona, Spain", IEEE Computer Society, Sep 2006, http://hal.archives-ouvertes.fr/hal-00089242.

Year Publications

Articles in refereed journals and book chapters

[16] V. BERTEN, J. GOOSSENS, E. JEANNOT. *On the Distribution of Sequential Jobs in Random Brokering For Heterogeneous Computational Grids*, in "IEEE Transactions on Parallel and Distributed Systems", vol. 17, n^{o} 2, 2006, p. 113–124.

[17] R. BOLZE, F. CAPPELLO, E. CARON, M. DAYDÉ, F. DESPREZ, E. JEANNOT, Y. JÉGOU, S. LANTERI, J. LEDUC, N. MELAB, G. MORNET, R. NAMYST, P. PRIMET, B. QUETIER, O. RICHARD, E.-G. TALBI, I. TOUCHE. *Grid'5000: A Large Scale And Highly Reconfigurable Experimental Grid Testbed*, in "International Journal of High Performance Computing Applications", vol. 20, n^{o} 4, November 2006, p. 481–494.

[18] 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.

[19] 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.

[20] J. GUSTEDT. *Efficient Sampling of Random Permutations*, in "J. on Discrete Algorithms", accepted, 2006, http://hal.inria.fr/inria-00000900/en/.

[21] E. JEANNOT, F. WAGNER. *Scheduling Messages for Data Redistribution: an Experimental Study*, in "International Journal of High Performance Computing Applications", vol. 20, n^{o} 4, November 2006, p. 443–454.

Publications in Conferences and Workshops

[22] V. BERTEN, J. GOOSSENS, E. JEANNOT. *A Probabilistic Approach for Fault Tolerant Multiprocessor Real-time Scheduling*, in "14th Workshop on Parallel and Distributed Real-Time Systems, Island of Rhodes, Greece", April 2006.

[23] L.-C. CANON, E. JEANNOT. *Wrekavoc a Tool for Hemulating Heterogeneity*, in "15th IEEE Heterogeneous Computing Workshop (HCW'06), Island of Rhodes, Greece", April 2006.

[24] H. CASANOVA, K. FUJIWARA, A. LEGRAND, M. QUINSON. *The SIMGRID Project: Simulation and Deployment of Distributed Applications*, in "The 15th IEEE International Symposium on High Performance Distributed Computing (HPDC'06)", 2006, http://hal.inria.fr/inria-00108428.

[25] M. ESSAÏDI, J. GUSTEDT. *An experimental validation of the PRO model for parallel and distributed computation*, in "14th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP 2006), 15/02/2006, Montbeliard-Sochaux, France", B. D. MARTINO (editor). , in: IEEE CS, 14th Euromicro International Conference on Parallel, Distributed and Network based Processing, IEEE, 2006, p. 449-456, http://hal.inria.fr/inria-00000612/en/.

[26] G. GOEL, J. GUSTEDT. *Bounded Arboricity to Determine the Local Structure of Sparse Graphs*, in "32 International Workshop on Graph-Theoretic Concepts in Computer Science - WG 2006, 21/06/2006, Bergen, Norway", in: Lecture Notes in Computer Science, Graph-Theoretic Concepts in Computer Science 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006 Revised Papers, vol. 4271, Springer, 2006, p. 159-167, http://hal.inria.fr/inria-00103755/en/.

[27] J. GUSTEDT, S. VIALLE, A. DE VIVO. *parXXL: A Fine Grained Development Environment on Coarse Grained Architectures*, in "Workshop on State-of-the-Art in Scientific and Parallel Computing - PARA'06, Umeå, Sweden", June 2006, http://hal.inria.fr/inria-00103772/en/.

[28] E. JEANNOT, F. VERNIER. *A Practical Approach of Diffusion Load Balancing Algorithm*, in "12th International Euro-Par Conference, Dresden, Germany", LNCS, vol. 4128, Springer Verlag, August 2006, p. 211–221.

[29] E. JEANNOT, F. WAGNER. *Modeling, Predicting and Optimizing Redistribution between Clusters on Low Latency Networks*, in "The First International Symposium on Frontiers in Networking with Applications (FINA 2006), Vienna, Austria", IEEE, April 2006.

[30] T. N'TAKPÉ. *Algorithmes d'ordonnancement de graphes de tâches parallèles sur plates-formes hétérogènes*, in "Dix-septièmes Rencontres Francophones du Parallélisme (RenPar'17), Perpignan", Oct 2006.

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

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

[33] Z. SHI, E. JEANNOT, J. J. DONGARRA. *Robust Task Scheduling in Non-Deterministic Heterogeneous Systems*, in "Proceedings of IEEE International Conference on Cluster Computing, Barcelona, Spain", IEEE, October 2006.

[34] L. A. STEFFENEL, G. MOUNIÉ. *Scheduling Heuristics for Efficient Broadcast Operations on Grid Environments*, in "Proceedings of the Performance Modeling, Evaluation and Optimization of Parallel and Distributed Systems Workshop - PMEO'06 (associated to IPDPS'06), Rhodes Island, Greece", IEEE Computer Society, 4 2006, http://hal.archives-ouvertes.fr/hal-00022008.

[35] L. A. STEFFENEL. *Modeling Network Contention Effects on AlltoAll Operations*, in "Proceedings of the IEEE Conference on Cluster Computing (CLUSTER 2006), Barcelona, Spain", IEEE Computer Society, 9 2006, http://hal.archives-ouvertes.fr/hal-00089242.

Internal Reports

[36] E. JEANNOT, F. VERNIER. *A Practical Approach of Diffusion Load Balancing Algorithms*, Research Report, n^{o} 5875, INRIA, March 2006, http://hal.inria.fr/inria-00071394.

[37] L. A. STEFFENEL. *A Framework for Adaptive Collective Communications on Heterogeneous Hierarchical Networks*, Research Report, n^{o} 6036, INRIA, 11 2006, https://hal.inria.fr/inria-00116897.

[38] L. A. STEFFENEL. *Fast and Scalable Total Order Broadcast for Wide-area Networks*, Research Report, n^{o} 6037, INRIA, 11 2006, https://hal.inria.fr/inria-00116895.

[39] L. A. STEFFENEL. *Modelling Network Contention Effects on All-to-All Operations*, Research Report, n^{o} 6038, INRIA, 11 2006, https://hal.inria.fr/inria-00116891.

Miscellaneous

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

[41] P.-N. CLAUSS, J. GUSTEDT, F. SUTER. *Optimized Data Layout for Architecture Independant Implementation of an Out-of-core Wavefront Algorithm*, 2006.

[42] J. J. DONGARRA, E. JEANNOT, J. LANGOU. *Modeling the LU Factorization for SMP Clusters*, Presented at 4th International Workshop on Parallel Matrix Algorithms and Applications (PMAA 06), September 2006.

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

References in notes

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

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] E. T. MIDORIKAWA, H. M. OLIVEIRA, J. M. LAINE. *PEMPIs: A New Metodology for Modeling and Prediction of MPI Programs Performance*, in "Proceedings of the SBAC-PAD 2004", IEEE Computer Society/Brasilian Computer Society, 2004, p. 254–261.

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

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