IN PARTNERSHIP WITH:
CNRS

Université Paris-Sud (Paris 11)

Université des sciences et technologies de Lille (Lille 1)

Activity Report 2013

Project-Team GRAND-LARGE

Global parallel and distributed computing

IN COLLABORATION WITH: Laboratoire d'informatique fondamentale de Lille (LIFL), Laboratoire de recherche en informatique (LRI)

RESEARCH CENTER
Saclay - Île-de-France

THEME
Distributed and High Performance Computing

Table of contents

Project-Team GRAND-LARGE

Keywords: Fault Tolerance, Grid Computing, High Performance Computing, Parallel Solver, Peer-to-peer

Creation of the Project-Team: 2003 October 02, end of the Project-Team: 2013 December 31.

1. Members

Research Scientists
- Franck Cappello [Senior Researcher, until Mar 2013, HdR]
- Christine Eisenbeis [Inria, Senior Researcher]
- Grigori Fursin [Inria, Researcher]

Faculty Members
- Marc Baboulin [Team leader (interim), Univ. Paris XI, Professor, HdR]
- Cédric Bastoul [Univ. Paris-Sud 11, until Aug. 2013, then Univ. Strasbourg, Professor, HdR]
- Joel Falcou [Univ. Paris XI, Associate Professor]
- Frédéric Gruau [Univ. Paris XI, Associate Professor]
- Brigitte Rozoy [Univ. Paris XI, Professor, HdR]

Engineer
- Taj Muhammad Khan [Inria]

PhD Students
- Michael Kruse [Univ. Paris-Sud 11, until Sep 2013, then Inria]
- Lénaïc Bagnères [Inria]
- Pierre Esterie [Univ. Paris XI, until Sep 2013]
- Alessandro Ferreira Leite [cotutelle univ. Paris-Sud 11 and univ. of Brazilia]
- Tatiana Martsinkevich [Inria]
- Adrien Remy de Zotti [Univ. Paris XI]

Post-Doctoral Fellow
- Chen Chen [Inria, from Dec 2013]

Visiting Scientists
- Serge Petiton [Univ. Lille I]
- Masha Sosonkina [June and December 2013]

Administrative Assistant
- Katia Evrat [Inria]

2. Overall Objectives

2.1. Grand-Large General Objectives

Grand-Large was evaluated in september 2012 and was supposed to end in December 2012. Additional information can be found in the 2012 Grand-Large evaluation report. It was continued in 2013 with purpose to create a new team involving some of the Grand-Large members.

Grand-Large is a research project investigating the issues raised by High Performance Computing (HPC) on Large Scale Distributed Systems (LSDS), where users execute HPC applications on a shared infrastructure and where resources are subject to failure, possibly heterogeneous, geographically distributed and administratively independent. More specifically, we consider large scale distributed computing mainly, Desktop Grids, Grids, and large scale parallel computers. Our research focuses on the design, development, proof and experiments of programming environments, middleware and scientific algorithms and libraries for HPC applications. Fundamentally, we address the issues related to HPC on LSDS, gathering several methodological tools that raise themselves scientific issues: theoretical models and exploration tools (simulators, emulators and real size experimental systems).

Our approach ranges from concepts to experiments, the projects aims at:

1. models and fault-tolerant algorithms, self-stabilizing systems and wireless networks.
2. studying experimentally, and formally, the fundamental mechanisms of LSDS for high performance computing;
3. designing, implementing, validating and testing real software, libraries, middleware and platforms;
4. defining, evaluating and experimenting approaches for programming applications on these platforms.

Compared to other European and French projects, we gather skills in 1) large scale systems formal design and validation of algorithms and protocols for distributed systems and 2) programming, evaluation, analysis and definition of programming languages and environments for parallel architectures and distributed systems.

This project pursues short and long term researches aiming at having scientific and industrial impacts. Research topics include:

1. the design of middleware for LSDS (XtremWeb and PVC)
2. large scale data movements on LSDS (BitDew)
3. fault tolerant MPI for LSDS, fault tolerant protocol verification (MPICH-V)
4. algorithms, programming and evaluation of scientific applications LSDS;
5. tools and languages for large scale computing on LSDS (OpenWP, YML).
6. Exploration systems and platforms for LSDS (Grid'5000, XtremLab, DSL-Lab, SimBOINC, FAIL, V-DS)

These researches should have some applications in the domain of Desktop Grids, Grids and large scale parallel computers.

As a longer term objective, we put special efforts on the design, implementation and use of Exploration Tools for improving the methodology associated with the research in LSDS. For example we had the responsibility of the Grid eXplorer project founded by the French ministry of research and we were deeply involved in the Grid5000 project (as project Director) and in the ALADDIN initiative (project scientific director).

3. Research Program

3.1. Large Scale Distributed Systems (LSDS)

What makes a fundamental difference between recent Global Computing systems (Seti@home), Grid (EGEE, TeraGrid) and former works on distributed systems is the large scale of these systems. This characteristic becomes also true for large scale parallel computers gathering tens of thousands of CPU cores. The notion of Large Scale is linked to a set of features that has to be taken into account in these systems. An example is the system dynamicity caused by node volatility: in Internet Computing Platforms (also called Desktop Grids), a non predictable number of nodes may leave the system at any time. Some recent results also report a very low MTTI (Mean Time To Interrupt) in top level supercomputers gathering 100,000+ CPU cores. Another example of characteristics is the complete lack of control of nodes connectivity. In Desktop Grid, we cannot assume that external administrator is able to intervene in the network setting of the nodes, especially their

connection to Internet via NAT and Firewalls. This means that we have to deal with the in place infrastructure in terms of performance, heterogeneity, dynamicity and connectivity. These characteristics, associated with the requirement of scalability, establish a new research context in distributed systems. The Grand-Large project aims at investigating theoretically as well as experimentally the fundamental mechanisms of LSDS, especially for the high performance computing applications.

3.1.1. Computing on Large Scale Global Computing systems

Large scale parallel and distributed systems are mainly used in the context of Internet Computing. As a consequence, until Sept. 2007, Grand-Large has focused mainly on Desktop Grids. Desktop Grids are developed for computing (SETI@home, Folding@home, Decrypthon, etc.), file exchanges (Napster, Kazaa, eDonkey, Gnutella, etc.), networking experiments (PlanetLab, Porivo) and communications such as instant messaging and phone over IP (Jabber, Skype). In the High Performance Computing domain, LSDS have emerged while the community was considering clustering and hierarchical designs as good performance-cost tradeoffs. Nowadays, Internet Computing systems are still very popular (the BOINC platform is used to run over 40 Internet Computing projects and XtremWeb is used in production in three countries) and still raise important research issues.

Desktop Grid systems essentially extend the notion of computing beyond the frontier of administration domains. The very first paper discussing this type of systems [79] presented the Worm programs and several key ideas that are currently investigated in autonomous computing (self replication, migration, distributed coordination, etc.). LSDS inherit the principle of aggregating inexpensive, often already in place, resources, from past research in cycle stealing/resource sharing. Due to its high attractiveness, cycle stealing has been studied in many research projects like Condor [69] , Glunix [64] and Mosix [45], to cite a few. A first approach to cross administration domains was proposed by Web Computing projects such as Jet [73], Charlotte [46], Javeline [57], Bayanihan [77], SuperWeb [42], ParaWeb [52] and PopCorn [54]. These projects have emerged with Java, taking benefit of the virtual machine properties: high portability across heterogeneous hardware and OS, large diffusion of virtual machine in Web browsers and a strong security model associated with bytecode execution. Performance and functionality limitations are some of the fundamental motivations of the second generation of Global Computing systems like BOINC [44] and XtremWeb [60]. The second generation of Global Computing systems appeared in the form of generic middleware which allow scientists and programmers to design and set up their own distributed computing project. As a result, we have seen the emergence of large communities of volunteers and projects. Currently, Global Computing systems are among the largest distributed systems in the world. In the mean time, several studies succeeded to understand and enhance the performance of these systems, by characterizing the system resources in term of volatility and heterogeneity and by studying new scheduling heuristics to support new classes of applications: data-intensive, long running application with checkpoint, workflow, soft-real time etc... However, despite these recent progresses, one can note that Global Computing systems are not yet part of high performance solution, commonly used by scientists. Recent researches to fulfill the requirements of Desktop Grids for high demanding users aim at redesigning Desktop Grid middleware by essentially turning a set of volatile nodes into a virtual cluster and allowing the deployment of regular HPC utilities (batch schedulers, parallel communication libraries, checkpoint services, etc...) on top of this virtual cluster. The new generation would permit a better integration in the environment of the scientists such as computational Grids, and consequently, would broaden the usage of Desktop Grid.

The high performance potential of LSDS platforms has also raised a significant interest in the industry. Performance demanding users are also interested by these platforms, considering their cost-performance ratio which is even lower than the one of clusters. Thus, several Desktop Grid platforms are daily used in production in large companies in the domains of pharmacology, petroleum, aerospace, etc.

Desktop Grids share with Grid a common objective: to extend the size and accessibility of a computing infrastructure beyond the limit of a single administration domain. In [61], the authors present the similarities and differences between Grid and Global Computing systems. Two important distinguishing parameters are the user community (professional or not) and the resource ownership (who own the resources and who is using them). From the system architecture perspective, we consider two main differences: the system scale

and the lack of control of the participating resources. These two aspects have many consequences, at least on the architecture of system components, the deployment methods, programming models, security (trust) and more generally on the theoretical properties achievable by the system.

Beside Desktop Grids and Grids, large scale parallel computers with tens of thousands (and even hundreds of thousands) of CPU cores are emerging with scalability issues similar to the one of Internet Computing systems: fault tolerance at large scale, large scale data movements, tools and languages. Grand-Large is gradually considering the application of selected research results, in the domain of large scale parallel computers, in particular for the fault tolerance and language topics.

3.1.2. Building a Large Scale Distributed System

This set of studies considers the XtremWeb project as the basis for research, development and experimentation. This LSDS middleware is already operational. This set gathers 4 studies aiming at improving the mechanisms and enlarging the functionalities of LSDS dedicated to computing. The first study considers the architecture of the resource discovery engine which, in principle, is close to an indexing system. The second study concerns the storage and movements of data between the participants of a LSDS. In the third study, we address the issue of scheduling in LSDS in the context of multiple users and applications. Finally the last study seeks to improve the performance and reduce the resource cost of the MPICH-V fault tolerant MPI for desktop grids.

3.1.2.1. The resource discovery engine

A multi-users/multi-applications LSDS for computing would be in principle very close to a P2P file sharing system such as Napster [78], Gnutella [78] and Kazaa [68], except that the shared resource is the CPUs instead of files. The scale and lack of control are common features of the two kinds of systems. Thus, it is likely that solutions sharing fundamental mechanisms will be adopted, such as lower level communication protocols, resource publishing, resource discovery and distributed coordination. As an example, recent P2P projects have proposed distributed indexing systems like CAN [75], CHORD [80], PASTRY [76] and TAPESTRY [84] that could be used for resource discovery in a LSDS dedicated to computing.

The resource discovery engine is composed of a publishing system and a discovery engine, which allow a client of the system to discover the participating nodes offering some desired services. Currently, there is as much resource discovery architectures as LSDS and P2P systems. The architecture of a resource discovery engine is derived from some expected features such as speed of research, speed of reconfiguration, volatility tolerance, anonymity, limited use of the network, matching between the topologies of the underlying network and the virtual overlay network.

This study focuses on the first objective: to build a highly reliable and stable overlay network supporting the higher level services. The overlay network must be robust enough to survive unexpected behaviors (like malicious behaviors) or failures of the underlying network. Unfortunately it is well known that under specific assumptions, a system cannot solve even simples tasks with malicious participants. So, we focus the study on designing overlay algorithms for transient failures. A transient failure accepts any kind of behavior from the system, for a limited time. When failures stop, the system will eventually provide its normal service again.

A traditional way to cope with transient failures are self-stabilizing systems [59]. Existing self-stabilizing algorithms use an underlying network that is not compatible with LSDS. They assume that processors know their list of neighbors, which does not fit the P2P requirements. Our work proposes a new model for designing self-stabilizing algorithms without making this assumption, then we design, prove and evaluate overlay networks self-stabilizing algorithms in this model.

3.1.2.2. Fault Tolerant MPI

MPICH-V is a research effort with theoretical studies, experimental evaluations and pragmatic implementations aiming to provide a MPI implementation based on MPICH [71], featuring multiple fault tolerant protocols.

There is a long history of research in fault tolerance for distributed systems. We can distinguish the automatic/transparent approach from the manual/user controlled approach. The first approach relies either on coordinated checkpointing (global snapshot) or uncoordinated checkpointing associated with message logging. A well known algorithm for the first approach has been proposed by Chandy and Lamport [56]. This algorithm requires restarting all processes even if only one process crashes. So it is believed not to scale well. Several strategies have been proposed for message logging: optimistic [82], pessimistic [43], causal [83]. Several optimizations have been studied for the three strategies. The general context of our study is high performance computing on large platforms. One of the most used programming environments for such platforms is MPI.

Within the MPICH-V project, we have developed and published several original fault tolerant protocols for MPI: MPICH-V1 [49], MPICH-V2 [50], MPICH-Vcausal, MPICH-Vcl [51], MPICH-Pcl. The two first protocols rely on uncoordinated checkpointing associated with either remote pessimistic message logging or sender based pessimistic message logging. We have demonstrated that MPICH-V2 outperforms MPICH-V1. MPICH-Vcl implements a coordinated checkpoint strategy (Chandy-Lamport) removing the need of message logging. MPICH-V2 and Vcl are concurrent protocols for large clusters. We have compared them considering a new parameter for evaluating the merits of fault tolerant protocols: the impact of the fault frequency on the performance. We have demonstrated that the stress of the checkpoint server is the fundamental source of performance differences between the two techniques. MPICH-Vcausal implements a causal message logging protocols, removing the need for waiting acknowledgement in contrary to MPICH-V2. MPICH-Pcl is a blocking implementation of the Vcl protocol. Under the considered experimental conditions, message logging becomes more relevant than coordinated checkpoint when the fault frequency reaches 1 fault every 4 hours, for a cluster of 100 nodes sharing a single checkpoint server, considering a data set of 1 GB on each node and a 100 Mb/s network.

Multiple important events arose from this research topic. A new open source implementation of the MPI-2 standard was born during the evolution of the MPICH-V project, namely OpenMPI. OpenMPI is the result of the alliance of many MPI projects in the USA, and we are working to port our fault tolerance algorithms both into OpenMPI and MPICH.

Grids becoming more popular and accessible than ever, parallel applications developers now consider them as possible targets for computing demanding applications. MPI being the de-facto standard for the programming of parallel applications, many projects of MPI for the Grid appeared these last years. We contribute to this new way of using MPI through a European Project in which we intend to grid-enable OpenMPI and provide new fault-tolerance approaches fitted for the grid.

When introducing Fault-Tolerance in MPI libraries, one of the most neglected component is the runtime environment. Indeed, the traditional approach consists in restarting the whole application and runtime environment in case of failure. A more efficient approach could be to implement a fault-tolerant runtime environment, capable of coping with failures at its level, thus avoiding the restart of this part of the application. The benefits would be a quicker restart time, and a better control of the application. However, in order to build a fault-tolerant runtime environment for MPI, new topologies, more connected, and more stable, must be integrated in the runtime environment.

For traditional parallel machines of large scale (like large scale clusters), we also continue our investigation of the various fault tolerance protocols, by designing, implementing and evaluating new protocols in the MPICH-V project.

3.2. Volatility and Reliability Processing

In a global computing application, users voluntarily lend the machines, during the period they don't use them. When they want to reuse the machines, it is essential to give them back immediately. We assume that there is no time for saving the state of the computation (for example because the user is shooting down is machine). Because the computer may not be available again, it is necessary to organize checkpoints. When the owner takes control of his machine, one must be able to continue the computation on another computer from a checkpoint as near as possible from the interrupted state.

The problems raised by this way of managing computations are numerous and difficult. They can be put into two categories: synchronization and repartition problems.

- Synchronization problems (example). Assume that the machine that is supposed to continue the computation is fixed and has a recent checkpoint. It would be easy to consider that this local checkpoint is a component of a global checkpoint and to simply rerun the computation. But on one hand the scalability and on the other hand the frequency of disconnections make the use of a global checkpoint totally unrealistic. Then the checkpoints have to be local and the problem of synchronizing the recovery machine with the application is raised.
- Repartition problems (example). As it is also unrealistic to wait for the computer to be available again before rerunning the interrupted application, one has to design a virtual machine organization, where a single virtual machine is implemented as several real ones. With too few real machines for a virtual one, one can produce starvation; with too many, the efficiency is not optimal. The good solution is certainly in a dynamic organization.

These types of problems are not new ([62]). They have been studied deeply and many algorithmic solutions and implementations are available. What is new here and makes these old solutions not usable is scalability. Any solution involving centralization is impossible to use in practice. Previous works validated on former networks can not be reused.

3.2.1. Reliability Processing

We voluntarily presented in a separate section the volatility problem because of its specificity both with respect to type of failures and to frequency of failures. But in a general manner, as any distributed system, a global computing system has to resist to a large set of failures, from crash failures to Byzantine failures, that are related to incorrect software or even malicious actions (unfortunately, this hypothesis has to be considered as shown by DECRYPTHON project or the use of erroneous clients in SETI@HOME project), with in between, transient failures such as loss of message duplication. On the other hand, failures related accidental or malicious memory corruptions have to be considered because they are directly related to the very nature of the Internet. Traditionally, two approaches (masking and non-masking) have been used to deal with reliability problems. A masking solution hides the failures to the user, while a non-masking one may let the user notice that failures occur. Here again, there exists a large literature on the subject (cf. [70], [81], [59] for surveys). Masking techniques, generally based on consensus, are not scalable because they systematically use generalized broadcasting. The self-stabilizing approach (a non-masking solution) is well adapted (specifically its time adaptive version, cf. [67], [66], [47], [48], [63]) for three main reasons:

1. Low overhead when stabilized. Once the system is stabilized, the overhead for maintaining correction is low because it only involves communications between neighbours.
2. Good adaptivity to the reliability level. Except when considering a system that is continuously under attacks, self-stabilization provides very satisfying solutions. The fact that during the stabilization phase, the correctness of the system is not necessarily satisfied is not a problem for many kinds of applications.
3. Lack of global administration of the system. A peer to peer system does not admit a centralized administrator that would be recognized by all components. A human intervention is thus not feasible and the system has to recover by itself from the failures of one or several components, that is precisely the feature of self-stabilizing systems.

We propose:

1. To study the reliability problems arising from a global computing system, and to design self-stabilizing solutions, with a special care for the overhead.
2. For problem that can be solved despite continuously unreliable environment (such as information retrieval in a network), to propose solutions that minimize the overhead in space and time resulting from the failures when they involve few components of the system.
3. For most critical modules, to study the possibility to use consensus based methods.
4. To build an adequate model for dealing with the trade-off between reliability and cost.

3.3. Parallel Programming on Peer-to-Peer Platforms (P5)

Several scientific applications, traditionally computed on classical parallel supercomputers, may now be adapted for geographically distributed heterogeneous resources. Large scale P2P systems are alternative computing facilities to solve grand challenge applications.

Peer-to-Peer computing paradigm for large scale scientific and engineering applications is emerging as a new potential solution for end-user scientists and engineers. We have to experiment and to evaluate such programming to be able to propose the larger possible virtualization of the underlying complexity for the end-user.

3.3.1. Large Scale Computational Sciences and Engineering

Parallel and distributed scientific application developments and resource managements in these environments are a new and complex undertaking. In scientific computation, the validity of calculations, the numerical stability, the choices of methods and software are depending of properties of each peer and its software and hardware environments; which are known only at run time and are non-deterministic. The research to obtain acceptable frameworks, methodologies, languages and tools to allow end-users to solve accurately their applications in this context is capital for the future of this programming paradigm.

GRID scientific and engineering computing exists already since more than a decade. Since the last few years, the scale of the problem sizes and the global complexity of the applications increase rapidly. The scientific simulation approach is now general in many scientific domains, in addition to theoretical and experimental aspects, often link to more classic methods. Several applications would be computed on world-spread networks of heterogeneous computers using some web-based Application Server Provider (ASP) dedicated to targeted scientific domains. New very strategic domains, such as Nanotechnologies, Climatology or Life Sciences, are in the forefront of these applications. The development in this very important domain and the leadership in many scientific domains will depend in a close future to the ability to experiment very large scale simulation on adequate systems [65]. The P2P scientific programming is a potential solution, which is based on existing computers and networks. The present scientific applications on such systems are only concerning problems which are mainly data independents: i.e. each peer does not communicate with the others.

P2P programming has to develop parallel programming paradigms which allow more complex dependencies between computing resources. This challenge is an important goal to be able to solve large scientific applications. The results would also be extrapolated toward future petascale heterogeneous hierarchically designed supercomputers.

3.3.2. Experimentations and Evaluations

We have followed two tracks. First, we did experiments on large P2P platforms in order to obtain a realistic evaluation of the performance we can expect. Second, we have set some hypothesis on peers, networks, and scheduling in order to have theoretical evaluations of the potential performance. Then, we have chosen a classical linear algebra method well-adapted to large granularity parallelism and asynchronous scheduling: the block Gauss-Jordan method to invert dense very large matrices. We have also chosen the calculation of one matrix polynomial, which generates computation schemes similar to many linear algebra iterative methods, well-adapted for very large sparse matrices. Thus, we were able to theoretically evaluate the potential throughput with respect to several parameters such as the matrix size and the multicast network speed.

Since the beginning of the evaluations, we experimented with those parallel methods on a few dozen peer XtremWeb P2P Platforms. We continue these experiments on larger platforms in order to compare these results to the theoretical ones. Then, we would be able to extrapolate and obtain potential performance for some scientific applications.

Recently, we also experimented several Krylov based method, such as the Lanczos and GMRES methods on several grids, such as a French-Japanese grid using hundred of PC in France and 4 clusters at the University of Tsukuba. We also experimented on GRID5000 the same methods. We currently use several middleware such as Xtremweb, OmniRPC and Condor. We also begin some experimentations on the Tsubame supercomputer

in collaboration with the TITech (Tokyo Institute of Technologies) in order to compare our grid approaches and the High performance one on an hybrid supercomputer.

Experimentations and evaluation for several linear algebra methods for large matrices on P2P systems will always be developed all along the Grand Large project, to be able to confront the different results to the reality of the existing platforms.

As a challenge, we would like, in several months, to efficiently invert a dense matrix of size one million using a several thousand peer platform. We are already inverting very large dense matrices on Grid5000 but more efficient scheduler and a larger number of processors are required to this challenge.

Beyond the experimentations and the evaluations, we propose the basis of a methodology to efficiently program such platforms, which allow us to define languages, tools and interface for the end-user.

3.3.3. Languages, Tools and Interface

The underlying complexity of the Large Scale P2P programming has to be mainly virtualized for the end-user. We have to propose an interface between the end-user and the middleware which may extract the end-user expertise or propose an on-the-shelf general solution. Targeted applications concern very large scientific problems which have to be developed using component technologies and up-to-dated software technologies.

We introduced the YML framework and language which allows to describe dependencies between components. We introduced different classes of components, depending of the level of abstraction, which are associated with divers parts of the framework. A component catalogue is managed by an administrator and/or the end-users. Another catalogue is managed with respect to the experimental platform and the middleware criteria. A front-end part is completely independent of any middleware or testbed, and a back-end part is developed for each targeted middleware/platform couple. A YML scheduler is adapted for each of the targeted systems.

The YML framework and language propose a solution to develop scientific applications to P2P and GRID platform. An end-user can directly develop programs using this framework. Nevertheless, many end-users would prefer avoid programming at the component and dependency graph level. Then, an interface has to be proposed soon, using the YML framework. This interface may be dedicated to a special scientific domain to be able to focus on the end-user vocabulary and P2P programming knowledge. We plan to develop such version based on the YML framework and language. The first targeted scientific domain will be very large linear algebra for dense or sparse matrices.

3.4. Methodology for Large Scale Distributed Systems

Research in the context of LSDS involves understanding large scale phenomena from the theoretical point of view up to the experimental one under real life conditions.

One key aspects of the impact of large scale on LSDS is the emergence of phenomena which are not coordinated, intended or expected. These phenomena are the results of the combination of static and dynamic features of each component of LSDS: nodes (hardware, OS, workload, volatility), network (topology, congestion, fault), applications (algorithm, parameters, errors), users (behavior, number, friendly/aggressive).

Validating current and next generation of distributed systems targeting large-scale infrastructures is a complex task. Several methodologies are possible. However, experimental evaluations on real testbeds are unavoidable in the life-cycle of a distributed middleware prototype. In particular, performing such real experiments in a rigorous way requires to benchmark developed prototypes at larger and larger scales. Fulfilling this requirement is mandatory in order to fully observe and understand the behaviors of distributed systems. Such evaluations are indeed mandatory to validate (or not!) proposed models of these distributed systems, as well as to elaborate new models. Therefore, to enable an experimentally-driven approach for the design of next generation of large scale distributed systems, developing appropriate evaluation tools is an open challenge.

Fundamental aspects of LSDS as well as the development of middleware platforms are already existing in Grand-Large. Grand-Large aims at gathering several complementary techniques to study the impact of large scale in LSDS: observation tools, simulation, emulation and experimentation on real platforms.

3.4.1. Observation tools

Observation tools are mandatory to understand and extract the main influencing characteristics of a distributed system, especially at large scale. Observation tools produce data helping the design of many key mechanisms in a distributed system: fault tolerance, scheduling, etc. We pursue the objective of developing and deploying a large scale observation tool (XtremLab) capturing the behavior of thousands of nodes participating to popular Desktop Grid projects. The collected data will be stored, analyzed and used as reference in a simulator (SIMBOINC).

3.4.2. Tool for scalability evaluations

Several Grid and P2P systems simulators have been developed by other teams: SimGrid [55], GridSim [53], Briks [41]. All these simulators considers relatively small scale Grids. They have not been designed to scale and simulate 10 K to 100 K nodes. Other simulators have been designed for large multi-agents systems such as Swarm [72] but many of them considers synchronous systems where the system evolution is guided by phases. In the P2P field, ad hoc many simulators have been developed, mainly for routing in DHT. Emulation is another tool for experimenting systems and networks with a higher degree of realism. Compared to simulation, emulation can be used to study systems or networks 1 or 2 orders of magnitude smaller in terms of number of components. However, emulation runs the actual OS/middleware/applications on actual platform. Compared to real testbed, emulation considers conducting the experiments on a fully controlled platform where all static and dynamic parameters can be controlled and managed precisely. Another advantage of emulation over real testbed is the capacity to reproduce experimental conditions. Several implementations/configurations of the system components can be compared fairly by evaluating them under the similar static and dynamic conditions. Grand-Large is leading one of the largest Emulator project in Europe called Grid explorer (French funding). This project has built and used a 1K CPUs cluster as hardware platform and gathers 24 experiments of 80 researchers belonging to 13 different laboratories. Experiments concerned developing the emulator itself and use of the emulator to explore LSDS issues. In term of emulation tool, the main outcome of Grid explorer is the V-DS system, using virtualization techniques to fold a virtual distributed system 50 times larger than the actual execution platform. V-DS aims at discovering, understanding and managing implicit uncoordinated large scale phenomena. Grid Explorer is still in use within the Grid'5000 platform and serves the community of 400 users 7 days a week and 24h a day.

3.4.3. Real life testbeds: extreme realism

The study of actual performance and connectivity mechanisms of Desktop Grids needs some particular testbed where actual middleware and applications can be run under real scale and real life conditions. Grand-Large is developing DSL-Lab, an experimental platform distributed on 50 sites (actual home of the participants) and using the actual DSL network as the connection between the nodes. Running experiments over DSL-Lab put the piece of software to study under extremely realistic conditions in terms of connectivity (NAT, Firewalls), performance (node and network), performance symmetry (DSL Network is not symmetric), etc.

To investigate real distributed system at large scale (Grids, Desktop Grids, P2P systems), under real life conditions, only a real platform (featuring several thousands of nodes), running the actual distributed system can provide enough details to clearly understand the performance and technical limits of a piece of software. Grand-Large members are strongly involved (as Project Director) in the French Grid5000 project which intents to deploy an experimental Grid testbed for computer scientists. This testbed features about 4000 CPUs gathering the resources of about 9 clusters geographically distributed over France. The clusters will be connected by a high speed network (Renater 10G). Grand-Large is the leading team in Grid5000, chairing the steering committee. As the Principal Investigator of the project, Grand-Large has taken some strong design decisions that nowadays give a real added value of Grid5000 compared to all other existing Grids: reconfiguration and isolation. From these two features, Grid5000 provides the capability to reproduce experimental conditions and thus experimental results, which is the cornerstone of any scientific instrument.

3.5. High Performance Scientific Computing

This research is in the area of high performance scientific computing, and in particular in parallel matrix algorithms. This is a subject of crucial importance for numerical simulations as well as other scientific and industrial applications, in which linear algebra problems arise frequently. The modern numerical simulations coupled with ever growing and more powerful computational platforms have been a major driving force behind a progress in numerous areas as different as fundamental science, technical/technological applications, life sciences.

The main focus of this research is on the design of efficient, portable linear algebra algorithms, such that solving a large set of linear equations or a least squares problem. The characteristics of the matrices commonly encountered in this situations can vary significantly, as are the computational platforms used for the calculations. Nonetheless two common trends are easily discernible. First, the problems to solve are larger and larger, since the numerical simulations are using higher resolution. Second, the architecture of today's supercomputers is getting very complex, and so the developed algorithms need to be adapted to these new achitectures.

3.5.1. Communication avoiding algorithms for numerical linear algebra

Since 2007, we work on a novel approach to dense and sparse linear algebra algorithms, which aims at minimizing the communication, in terms of both its volume and a number of transferred messages. This research is motivated by technological trends showing an increasing communication cost. Its main goal is to reformulate and redesign linear algebra algorithms so that they are optimal in an amount of the communication they perform, while retaining the numerical stability. The work here involves both theoretical investigation and practical coding on diverse computational platforms. We refer to the new algorithms as *communication avoiding algorithms* [58] [10]. In our team we focus on communication avoiding algorithms for dense direct methods as well as sparse iterative methods.

The theoretical investigation focuses on identifying lower bounds on communication for different operations in linear algebra, where communication refers to data movement between processors in the parallel case, and to data movement between different levels of memory hierarchy in the sequential case. The lower bounds are used to study the existing algorithms, understand their communication bottlenecks, and design new algorithms that attain them.

This research focuses on the design of linear algebra algorithms that minimize the cost of communication. Communication costs include both latency and bandwidth, whether between processors on a parallel computer or between memory hierarchy levels on a sequential machine. The stability of the new algorithms represents an important part of this work.

3.5.2. Preconditioning techniques

Solving a sparse linear system of equations is the most time consuming operation at the heart of many scientific applications, and therefore it has received a lot of attention over the years. While direct methods are robust, they are often prohibitive because of their time and memory requirements. Iterative methods are widely used because of their limited memory requirements, but they need an efficient preconditioner to accelerate their convergence. In this direction of research we focus on preconditioning techniques for solving large sparse systems.

One of the main challenges that we address is the scalability of existing methods as incomplete LU factorizations or Schwarz-based approaches, for which the number of iterations increases significantly with the problem size or with the number of processors. This is often due to the presence of several low frequency modes that hinder the convergence of the iterative method. To address this problem, we study direction preserving solvers in the context of multilevel filtering LU decompositions. A judicious choice for the directions to be preserved through filtering allows us to alleviate the effect of low frequency modes on the convergence. While preconditioners and their scalability are studied by many other groups, our approach of direction preserving and filtering is studied in only very few other groups in the world (as Lawrence Livermore National Laboratory, Frankfurt University, Pennsylvania State University).

3.5.3. Fast linear algebra solvers based on randomization

Linear algebra calculations can be enhanced by statistical techniques in the case of a square linear system $Ax = b$ where A is a general or symmetric indefinite matrix [3]& [1]. Thanks to a random transformation of A, it is possible to avoid pivoting and then to reduce the amount of communication. Numerical experiments show that this randomization can be performed at a very affordable computational price while providing us with a satisfying accuracy when compared to partial pivoting. This random transformation called Partial Random Butterfly Transformation (PRBT) is optimized in terms of data storage and flops count. A PRBT solver for LU factorization (and for LDL^T factorization on multicore) has been developed. This solver takes advantage of the latest generation of hybrid multicore/GPU machines and gives better Gflop/s performance than existing factorization routines [19].

3.5.4. Sensitivity analysis of linear algebra problems

We derive closed formulas for the condition number of a linear function of the total least squares solution [4]. Given an over determined linear systems $Ax = b$, we show that this condition number can be computed using the singular values and the right singular vectors of $[A, b]$ and A. We also provide an upper bound that requires the computation of the largest and the smallest singular value of $[A, b]$ and the smallest singular value of A. In numerical experiments, we compare these values with condition estimates from the literature.

4. Software and Platforms

4.1. APMC-CA

Participants: Sylvain Peyronnet [correspondant], Joel Falcou, Pierre Esterie, Khaled Hamidouche, Alexandre Borghi.

The APMC model checker implements the state-of-the-art approximate probabilistic model checking methods. Last year we develop a version of the tool dedicated to the CELL architecture. Clearly, it was very pedagogic, but the conclusion is that the CELL is not adapted to sampling based verification methods.

This year we develop, thanks to the BSP++ framework, a version compatible with SPM/multicores machines, clusters and hybrid architectures. This version outperforms all previous ones, thus showing the interest of both these new architectures and of the BSP++ framework.

4.2. YML

Participants: Serge Petiton [correspondant], Nahid Emad, Maxime Hugues.

Scientific end-users face difficulties to program P2P large scale applications using low level languages and middleware. We provide a high level language and a set of tools designed to develop and execute large coarse grain applications on peer-to-peer systems. Thus, we introduced, developed and experimented the YML for parallel programming on P2P architectures. This work was done in collaboration with the PRiSM laboratory (team of Nahid Emad).

The main contribution of YML is its high level language for scientific end-users to develop parallel programs for P2P platforms. This language integrates two different aspects. The first aspect is a component description language. The second aspect allows to link components together. A coordination language called YvetteML can express graphs of components which represent applications for peer-to-peer systems.

Moreover, we designed a framework to take advantage of the YML language. It is based on two component catalogues and an YML engine. The first one concerns end-user's components and the second one is related to middleware criteria. This separation enhances portability of applications and permits real time optimizations. Currently we provide support for the XtremWeb Peer-to-Peer middleware and the OmniRPC grid system. The support for Condor is currently under development and a beta-release will be delivered soon (in this release, we plan to propagate semantic data from the end-users to the middleware). The next development of YML concerns the implementation of a multi-backend scheduler. Therefore, YML will be able to schedule at runtime computing tasks to any global computing platform using any of the targeted middleware.

We experimented YML with basic linear algebra methods on a XtremWeb P2P platform deployed between France and Japan. Recently, we have implemented complex iterative restarted Krylov methods, such as Lanczos-Bisection, GMRES and MERAM methods, using YML with the OmniRPC back-end. The experiments are performed either on the Grid5000 testbed of on a Network of Workstations deployed between Lille, Versailles and Tsukuba in Japan. Demos was proposed on these testbeds from conferences in USA. We recently finished evaluations of the overhead generated using YML, without smart schedulers and with extrapolations due to the lack of smart scheduling strategies inside targeted middleware.

In the context of the FP3C project funded by ANR-JST, we have recently extended YML to support a directive distributed parallel language, XcalableMP http://www.xcalablemp.org/. This extension is based on the support of the XcalableMP language inside YML components. This allows to develop parallel programs with two programming paradigm and thus two parallelism levels. This work is a part of the project that targets post-Petascale supercomputer that would be composed of heterogeneous and massively parallel hardware.

The software is available at http://yml.prism.uvsq.fr/

4.3. The Scientific Programming InterNet (SPIN)

Participant: Serge Petiton [correspondant].

SPIN (Scientific Programming on the InterNet), is a scalable, integrated and interactive set of tools for scientific computations on distributed and heterogeneous environments. These tools create a collaborative environment allowing the access to remote resources.

The goal of SPIN is to provide the following advantages: Platform independence, Flexible parameterization, Incremental capacity growth, Portability and interoperability, and Web integration. The need to develop a tool such as SPIN was recognized by the GRID community of the researchers in scientific domains, such as linear algebra. Since the P2P arrives as a new programming paradigm, the end-users need to have such tools. It becomes a real need for the scientific community to make possible the development of scientific applications assembling basic components hiding the architecture and the middleware. Another use of SPIN consists in allowing to build an application from predefined components ("building blocks") existing in the system or developed by the developer. The SPIN users community can collaborate in order to make more and more predefined components available to be shared via the Internet in order to develop new more specialized components or new applications combining existing and new components thanks to the SPIN user interface.

SPIN was launched at ASCI CNRS lab in 1998 and is now developed in collaboration with the University of Versailles, PRiSM lab. SPIN is currently under adaptation to incorporate YML, cf. above. Nevertheless, we study another solution based on the Linear Algebra KErnel (LAKE), developed by the Nahid Emad team at the University of Versailles, which would be an alternative to SPIN as a component oriented integration with YML.

4.4. V-DS

Participant: Franck Cappello [correspondant].

This project started officially in September 2004, under the name V-Grid. V-DS stands for Virtualization environment for large-scale Distributed Systems. It is a virtualization software for large scale distributed system emulation. This software allows folding a distributed systems 100 or 1000 times larger than the experimental testbed. V-DS virtualizes distributed systems nodes on PC clusters, providing every virtual node its proper and confined operating system and execution environment. Thus compared to large scale distributed system simulators or emulators (like MicroGrid), V-DS virtualizes and schedules a full software environment for every distributed system node. V-DS research concerns emulation realism and performance.

A first work concerns the definition and implementation of metrics and methodologies to compare the merits of distributed system virtualization tools. Since there is no previous work in this domain, it is important to define what and how to measure in order to qualify a virtualization system relatively to realism and performance. We defined a set of metrics and methodologies in order to evaluate and compared virtualization tools for sequential system. For example a key parameter for the realism is the event timing: in the emulated environment, events should occur with a time consistent with a real environment. An example of key parameter for the performance is the linearity. The performance degradation for every virtual machine should evolve linearly with the increase of the number of virtual machines. We conducted a large set of experiments, comparing several virtualization tools including Vserver, VMware, User Mode Linux, Xen, etc. The result demonstrates that none of them provides both enough isolation and performance. As a consequence, we are currently studying approaches to cope with these limits.

We have made a virtual platform on the GDX cluster with the Vserver virtualization tool. On this platform, we have launched more than 20K virtual machines (VM) with a folding of 100 (100 VM on each physical machine). However, some recent experiments have shown that a too high folding factor may cause a too long execution time because of some problems like swapping. Currently, we are conducting experiments on another platform based on the virtualization tool named Xen which has been strongly improved since 2 years. We expect to get better result with Xen than with Vserver. Recently, we have been using the V-DS version based on Xen to evaluate at large scales three P2P middleware [74].

This software is available at http://v-ds.lri.fr/

4.5. PVC: Private Virtual Cluster

Participant: Franck Cappello [correspondant].

Current complexity of Grid technologies, the lack of security of Peer-to-Peer systems and the rigidity of VPN technologies make sharing resources belonging to different institutions still technically difficult.

We propose a new approach called "Instant Grid" (IG), which combines various Grid, P2P and VPN approaches, allowing simple deployment of applications over different administration domains. Three main requirements should be fulfilled to make Instant Grids realistic: simple networking configuration (Firewall and NAT), no degradation of resource security, no need to re-implement existing distributed applications.

Private Virtual Cluster, is a low-level middle-ware that meets Instant Grid requirements. PVC turns dynamically a set of resources belonging to different administration domains into a virtual cluster where existing cluster runtime environments and applications can be run. The major objective of PVC is to establish direct connections between distributed peers. To connect firewall protected nodes in the current implementation, we have integrated three techniques: UPnP, TCP/UDP Hole Punching and a novel technique Traversing-TCP.

One of the major application of PVC is the third generation desktop Grid middleware. Unlike BOINC and XtremWeb (which belong to the second generation of desktop Grid middleware), PVC allows the users to build their Desktop Grid environment and run their favorite batch scheduler, distributed file system, resource monitoring and parallel programming library and runtime software. PVC ensures the connectivity layer and provide a virtual IP network where the user can install and run existing cluster software.

By offering only the connectivity layer, PVC allows to deploy P2P systems with specific applications, like file sharing, distributed computing, distributed storage and archive, video broadcasting, etc.

4.6. OpenWP

Participant: Franck Cappello [correspondant].

Distributed applications can be programmed on the Grid using workflow languages, object oriented approaches (Proactive, IBIS, etc), RPC programming environments (Grid-RPC, DIET), component based environments (generally based on Corba) and parallel programming libraries like MPI.

For high performance computing applications, most of the existing codes are programmed in C, Fortran and Java. These codes have 100,000 to millions of lines. Programmers are not inclined to rewrite then in a "non standard" programming language, like UPC, CoArray Fortran or Global Array. Thus environments like MPI and OpenMPI remain popular even if they require hybrid approaches for programming hierarchical computing infrastructures like cluster of multi-processors equipped with multi-core processors.

Programming applications on the Grid add a novel level in the hierarchy by clustering the cluster of multi-processors. The programmer will face strong difficulties in adapting or programming a new application for these runtime infrastructures featuring a deep hierarchy. Directive based parallel and distributed computing is appealing to reduce the programming difficulty by allowing incremental parallelization and distribution. The programmer add directives on a sequential or parallel code and may check for every inserted directive its correction and performance improvement.

We believe that directive based parallel and distributed computing may play a significant role in the next years for programming High performance parallel computers and Grids. We have started the development of OpenWP. OpenWP is a directive based programming environment and runtime allowing expressing workflows to be executed on Grids. OpenWP is compliant with OpenMP and can be used in conjunction with OpenMP or hybrid parallel programs using MPI + OpenMP.

The OpenWP environment consists in a source to source compiler and a runtime. The OpenWP parser, interprets the user directives and extracts functional blocks from the code. These blocks are inserted in a library distributed on all computing nodes. In the original program, the functional blocks are replaced by RPC calls and calls to synchronization. During the execution, the main program launches non blocking RPC calls to functions on remote nodes and synchronize the execution of remote functions based on the synchronization directives inserted by the programmer in the main code. Compared to OpenMP, OpenWP does not consider a shared memory programming approach. Instead, the source to source compiler insert data movements calls in the main code. Since the data set can be large in Grid application, the OpenWP runtime organize the storage of data sets in a distributed way. Moreover, the parameters and results of RPC calls are passed by reference, using a DHT. Thus, during the execution, parameter and result references are stored in the DHT along with the current position of the datasets. When a remote function is called, the DHT is consulted to obtain the position of the parameter data sets in the system. When a remote function terminates its execution, it stores the result data sets and store a reference to the data set in the DHT.

We are evaluating OpenWP from an industrial application (Amibe), used by the European aerospace company EADS. Amibe is the mesher module of jCAE [1]. Amibe generates a mesh from a CAD geometry in three steps. It first creates edges between every patch of the CAD (mesh in one dimension), then generates a surface mesh for every unfolded patch (mesh in two dimensions) and finally adds the third dimension to the mesh by projecting the 2D mesh into the original CAD surfaces. The first and third operation cannot be distributed. However the second step can easily be distributed following a master/worker approach, transferring the mesh1d results to every computing node and launching the distributed execution of the patches.

4.7. OpenScop

Participant: Cédric Bastoul.

OpenScop is an open specification which defines a file format and a set of data structures to represent a *static control part* (SCoP for short), i.e., a program part that can be represented in the *polyhedral model*, an algebraic representation of programs used for automatic parallelization and optimization (used, e.g., in GNU GCC, LLVM, IBM XL or Reservoir Labs R-Stream compilers). The goal of OpenScop is to provide a common interface to various polyhedral compilation tools in order to simplify their interaction.

OpenScop provides a single format for tools that may have different purposes (e.g., as different as code generation and data dependence analysis). We could observe that most available polyhedral compilation tools during the last decade were manipulating the same kind of data (polyhedra, affine functions...) and were actually sharing a part of their input (e.g., iteration domains and context concepts are nearly everywhere). We

[1] project page: http://jcae.sourceforge.net

could also observe that those tools may rely on different internal representations, mostly based on one of the major polyhedral libraries (e.g., Polylib, PPL or isl), and this representation may change over time (e.g., when switching to a more convenient polyhedral library). OpenScop aims at providing a stable, unified format that offers a durable guarantee that a tool can use an output or provide an input to another tool without breaking a compilation chain because of some internal changes in one element of this chain. The other promise of OpenScop is the ability to assemble or replace the basic blocks of a polyhedral compilation framework at no, or at least low engineering cost. The OpenScop Library (licensed under the 3-clause BSD license) has been developped as an example, yet powerful, implementation of the OpenScop specification.

4.8. Clay

Participant: Cédric Bastoul.

Clay is a free software and library devoted to semi-automatic optimization using the polyhedral model. It can input a high-level program or its polyhedral representation and transform it according to a transformation script. Classic loop transformations primitives are provided. Clay is able to check for the legality of the complete sequence of transformation and to suggest corrections to the user if the original semantics is not preserved (experimental at this document redaction time). Main authors include Joël Poudroux and Cédric Bastoul.

4.9. Fast linear system solvers in public domain libraries

Participant: Marc Baboulin [correspondant].

Hybrid multicore+GPU architectures are becoming commonly used systems in high performance computing simulations. In this research, we develop linear algebra solvers where we split the computation over multicore and graphics processors, and use particular techniques to reduce the amount of pivoting and communication between the hybrid components. This results in efficient algorithms that take advantage of each computational unit [16]. Our research in randomized algorithms yields to several contributions to propose public domain libraries PLASMA and MAGMA in the area of fast linear system solvers for general and symmetric indefinite systems. These solvers minimize communication by removing the overhead due to pivoting in LU and $LDLT$ factorization. Different approaches to reduce communication are compared in [2].

See also the web page http://icl.cs.utk.edu/magma/.

4.10. cTuning: Repository and Tools for Collective Characterization and Optimization of Computing Systems

Participant: Grigori Fursin [correspondant].

Designing, porting and optimizing applications for rapidly evolving computing systems is often complex, ad-hoc, repetitive, costly and error prone process due to an enormous number of available design and optimization choices combined with the complex interactions between all components. We attempt to solve this fundamental problem based on collective participation of users combined with empirical tuning and machine learning.

We developed cTuning framework that allows to continuously collect various knowledge about application characterization and optimization in the public repository at cTuning.org. With continuously increasing and systematized knowledge about behavior of computer systems, users should be able to obtain scientifically motivated advices about anomalies in the behavior of their applications and possible solutions to effectively balance performance and power consumption or other important characteristics.

Currently, we use cTuning repository to analyze and learn profitable optimizations for various programs, datasets and architectures using machine learning enabled compiler (MILEPOST GCC). Using collected knowledge, we can quickly suggest better optimizations for a previously unseen programs based on their semantic or dynamic features [8].

We believe that such approach will be vital for developing efficient Exascale computing systems. We are currently developing the new extensible cTuning2 framework for automatic performance and power tuning of HPC applications.

For more information, see the web page http://cTuning.org.

5. New Results

5.1. Automated Code Generation for Lattice Quantum Chromodynamics

Participants: Denis Barthou, Konstantin Petrov, Olivier Brand-Foissac, Olivier Pène, Gilbert Grosdidier, Michael Kruse, Romain Dolbeau, Christine Eisenbeis, Claude Tadonki.

This ongoing work is about a Domain Specific Language which aims to simplify Monte-Carlo simulations and measurements in the domain of Lattice Quantum Chromodynamics. The tool-chain, called Qiral, is used to produce high-performance OpenMP C code from LaTeX sources. We discuss conceptual issues and details of implementation and optimization. The comparison of the performance of the generated code to the well-established simulation software is also made.[33][20][37]

5.2. A Fine-grained Approach for Power Consumption Analysis and Prediction

Participants: Alessandro Ferreira Leite, Claude Tadonki, Christine Eisenbeis, Alba Cristina de Melo.

Power consumption has became a critical concern in modern computing systems for various reasons including financial savings and environmental protection. With battery powered devices, we need to care about the available amount of energy since it is limited. For the case of supercomputers, as they imply a large aggregation of heavy CPU activities, we are exposed to a risk of overheating. As the design of current and future hardware is becoming more and more complex, energy prediction or estimation is as elusive as that of time performance. However, having a good prediction of power consumption is still an important request to the computer science community. Indeed, power consumption might become a common performance and cost metric in the near future. A good methodology for energy prediction could have a great impact on power-aware programming, compilation, or runtime monitoring. In this paper, we try to understand from measurements where and how power is consumed at the level of a computing node. We focus on a set of basic programming instructions, more precisely those related to CPU and memory. We propose an analytical prediction model based on the hypothesis that each basic instruction has an average energy cost that can be estimated on a given architecture through a series of micro-benchmarks. The considered energy cost per operation includes all of the overhead due to context of the loop where it is executed. Using these precalculated values, we derive an linear extrapolation model to predict the energy of a given algorithm expressed by means of atomic instructions. We then use three selected applications to check the accuracy of our prediction method by comparing our estimations with the corresponding measurements obtained using a multimeter. We show a 9.48% energy prediction on sorting.[35]

5.3. Switcheable scheduling

Participants: Lénaïc Bagnères, Cédric Bastoul, Taj Khan.

Parallel applications used to be executed alone until their termination on partitions of supercomputers. The recent shift to multicore architectures for desktop and embedded systems is raising the problem of the coexistence of several parallel programs. Operating systems already take into account the *affinity* mechanism to ensure a thread will run only onto a subset of available processors (e.g., to reuse data remaining in the cache since its previous execution). But this is not enough, as demonstrated by the large performance gaps between executions of a given parallel program on desktop computers running several processes. To support many parallel applications, advances must be made on the system side (scheduling policies, runtimes, memory management...). However, automatic optimization and parallelization can play a significant role by generating programs with dynamic-auto-tuning capabilities to adapt themselves to the complete execution context, including the system load.

Our approach is to design at compile-time programs that can adapt at run-time to the execution context. The originality of our solution is to rely on *switcheable scheduling*, a selected set of program restructuring which allows to swap between program versions at some meeting points without backtracking. A first step selects pertinent versions according to their performance behavior on some execution contexts. The second step builds the auto-adaptive program with the various versions. Then at runtime the program selects the best version by a low overhead sampling and profiling of the versions, ensuring every computation is useful.

This work has been started at Paris-Sud University by Cédric Bastoul before he joined the Inria CAMUS project team during this year. The first results have been presented in 2013 at the HiPEAC System Week and at the Rencontres Françaises de Compilation.

5.4. Solving Navier-Stokes equations on heterogeneous parallel architectures

Participants: Marc Baboulin, Jack Dongarra, Joël Falcou, Yann Fraigneau, Olivier Lemaître, Yushan Wang.

The Navier-Stokes equations describe a large class of fluid flows but are difficult to solve analytically because of their nonlinearity. We implemented a parallel solver for the 3-D Navier-Stokes equations of incompressible unsteady flows with constant coefficients, discretized by the finite difference method. We applied the prediction-projection method which transforms the Navier-Stokes equations into three Helmholtz equations and one Poisson equation. For each Helmholtz system, we applied the Alternating Direction Implicit (ADI) method resulting in three tridiagonal systems. The Poisson equation is solved using partial diagonalization which transforms the Laplacian operator into a tridiagonal one. Our implementation is based on MPI where the computations are performed on each subdomain and information is exchanged on the interfaces, and where the tridiagonal system solutions are accelerated using vectorization techniques. We provided performance results on a current multicore system.[31]

5.5. Optimizing NUMA effects in dense linear algebra software

Participants: Marc Baboulin, Adrien Rémy, Brigitte Rozoy, Masha Sosonkina.

We studied the impact of non-uniform memory accesses (NUMA) on the solution of dense general linear systems using an LU factorization algorithm. In particular we illustrated how an appropriate placement of the threads and memory on a NUMA architecture can improve the performance of the panel factorization and consequently accelerate the global LU factorization. We applied these placement strategies and presented performance results for a hybrid multicore/GPU LU algorithm as it is implemented in the public domain library MAGMA.

6. Partnerships and Cooperations

6.1. Regional Initiatives

- **CALIFHA project (DIM Digiteo 2011)**: CALculations of Incompressible Fluid flows on Heterogeneous Architectures. Funding for a PhD student. Collaboration with LIMSI/CNRS. Participants: Marc Baboulin (Principal Investigator), Joel Falcou, Yann Fraigneau (LIMSI), Laura Grigori, Olivier Le Maître (LIMSI), Laurent Martin Witkowski (LIMSI)

6.2. National Initiatives

6.2.1. ANR

- **ANR SPADES** Coordinated by LIP-ENS Lyon. (Sylvain Peyronnet, Franck Cappello, Ala Rezmerita)

- **ANR Cosinus project PetaQCD - Towards PetaFlops for Lattice Quantum ChromoDynamics** (2009-2012) Collaboration with Lal (Orsay), Irisa Rennes (Caps/Alf), IRFU (CEA Saclay), LPT (Orsay), Caps Entreprise (Rennes), Kerlabs (Rennes), LPSC (Grenoble). About the design of architecture, software tools and algorithms for Lattice Quantum Chromodynamics. (Cédric Bastoul, Christine Eisenbeis, Michael Kruse)

6.3. European Initiatives

6.3.1. Collaborations in European Programs, except FP7

Program: ITEA

Project acronym: MANY

Project title: Many-core Programming and Resource Management for High-Performance Embedded Systems

Duration: 09/2011 - 08/2014

Coordinator: XDIN

Other partners: France: Thales Communications and Security, CAPS Entreprise, Telecom SudParis; Spain: UAB; Sweden: XDIN; Korea: ETRI, TestMidas, SevenCore; Netherlands: Vector Fabrics, ST-Ericsson, TU Eindhoven; Belgium: UMONS.

Abstract: Adapting Industry for the for the disruptive landing of many-core processors in Embedded Systems in order to provide scalable, reusable and very fast sofware development.

6.4. International Initiatives

6.4.1. Inria International Labs

- Franck Cappello, Co-Director of the **Inria - Illinois Joint Laboratory** on PetaScale Computing, since 2009

6.4.2. Participation In other International Programs

Stic AmSud: BioCloud-EEAmSud **Participants:** Christine Eisenbeis, Alessandro Ferreira Leite, Claude Tadonki.

BioCloud-EEAmSud is a cooperation project integrated by Brazil, Chile and France following the 2012 STIC-AmSud call. Partners in Brazil are Universidade de Brasilia, Universidade Federal Fluminense, and EMBRAPA-Genetic Resources and Biotechnology (CENARGEN), through the support of the Coordination of Improvement of Senior Staff of the Ministry of Education in Brazil (CAPES). In Chile, the main partner is Universidad de Santiago de Chile, through the support of the National Commission for Scientific and Technological Research of Chile (CONICYT). In France, the institutions involved are Mines ParisTech (CRI) and Inria-Saclay, through the support of the Ministry of Foreign and European Affairs (MAEE). The international project coordinator is Pr. Maria Emília Machado Telles Walter (UnB). Alessandro Ferreira Leite' thesis work is a joint University of Brazilia - université Paris-Sud 11 thesis and is partially supported by BioCloud-EEAmSud. Maria Emilia Machado Telles Walter and Alba Cristian de Melo visited Grand-Large in 2013, as well as Taina Rajol.

6.5. International Research Visitors

6.5.1. Internships

German Schinca

Subject: Minimizing communication in scientific computing

Date: from Sep 2012 until Mar 2013
Institution: University of Buenos Aires (Argentina)

7. Dissemination

7.1. Scientific Animation

Christine Eisenbeis

- IJPP (International Journal on Parallel Programming) editorial board.

Marc Baboulin

- Member of Steering Committee of ACM High Performance Computing Symposium (HPC 2013), San Diego, April 7-10, 2013.

7.2. Teaching - Supervision - Juries

7.2.1. Teaching

Licence : Cédric Bastoul, Réseaux niveau licence, IUT d'Orsay (60h), Système niveau licence, IUT d'Orsay (40h).

Master : Christine Eisenbeis, coordinatrice du module "Optimisations et compilation" du M2 recherche NSI ("Nouveaux systèmes informatiques") de l'université Paris-Sud 11. 3 heures de cours.

Master: M. Baboulin and J. Falcou teach in "Calcul Haute Performance" of M2 recherche NSI of University Paris Sud 11.

Polytech 5th year: M. Baboulin and J. Falcou teach the "Parallel Computing" class.

7.2.2. Supervision

PhD : Amal Khabou, Dense matrix computations: communication cost and numerical stability, University Paris Sud 11, 11 February 2013, PhD Supervisor: L. Grigori

PhD in progress: Ian Masliah, Automatic code generation in high-performance computing numerical libraries, University Paris Sud 11, Supervisors: M. Baboulin and J. Falcou

PhD in progress: Adrien Rémy, Solving dense linear systems on accelerated multicore architectures, University Paris Sud 11, Supervisors: M. Baboulin and B. Rozoy

PhD in progress: Yushan Wang, Numerical simulations of incompressible fluid flows on heterogeneous parallel architectures, University Paris Sud 11, Supervisors: M. Baboulin and O. Le Maître

PhD in progress: Lénaïc Bagnères, université Paris-Sud 11, supervisors: Cédric Bastoul and Christine Eisenbeis

PhD in progress: Alessandro Leite, université Paris-Sud 11, supervisors: Alba de Melo (university of Brazilia), Claude Tadonki (CRI, école des Mines de Paris), Christine Eisenbeis

PhD in progress: Michael Kruse, Polytopic memory layout optimization, université Paris-Sud 11, supervisor: Christine Eisenbeis

7.2.3. Committees

- Marc Baboulin, President of the PhD committee of Marc Letournel: "Approches duales dans la résolution de problèmes stochastiques", September 27, 2013.
- Christine Eisenbeis, jury de HdR de Claude Tadonki, "High Performance Computing as a Combination of Machines and Methods and Programming", jeudi 16 mai 2013, université Paris-Sud.

7.3. Popularization

Christine Eisenbeis est membre du conseil scientifique des programmes du centre d'Alembert, Centre Interdisciplinaire d'Étude de l'Évolution des Idées, des Sciences et des Techniques (CIEEIST), de l'université Paris-Sud.

8. Bibliography

Major publications by the team in recent years

[1] M. BABOULIN, D. BECKER, J. DONGARRA. *A Parallel Tiled Solver for Dense Symmetric Indefinite Systems on Multicore Architectures*, in "Proceedings of IEEE International Parallel & Distributed Processing Symposium (IPDPS 2012)", 2012, pp. 14-24

[2] M. BABOULIN, S. DONFACK, J. DONGARRA, L. GRIGORI, A. RÉMY, S. TOMOV. *A class of communication-avoiding algorithms for solving general dense linear systems on CPU/GPU parallel machines*, in "International Conference on Computational Science (ICCS 2012)", Procedia Computer Science, Elsevier, 2012, vol. 9, pp. 17–26

[3] M. BABOULIN, J. DONGARRA, J. HERRMANN, S. TOMOV. *Accelerating linear system solutions using randomization techniques*, in "ACM Trans. Math. Softw.", 2012, vol. 39, n^{o} 2

[4] M. BABOULIN, S. GRATTON. *A contribution to the conditioning of the total least squares problem*, in "SIAM J. Matrix Anal. and Appl.", 2011, vol. 32, n^{o} 3, pp. 685–699

[5] R. BOLZE, F. CAPPELLO, E. CARON, M. J. DAYDÉ, F. DESPREZ, E. JEANNOT, Y. JÉGOU, S. LANTERI, J. LEDUC, N. MELAB, G. MORNET, R. NAMYST, P. PRIMET, B. QUÉTIER, O. RICHARD, E.-G. TALBI, T. IRENA. *Grid'5000: a large scale and highly reconfigurable experimental Grid testbed*, in "International Journal of High Performance Computing Applications", November 2006, vol. 20, n^{o} 4, pp. 481-494

[6] A. BOUTEILLER, T. HÉRAULT, G. KRAWEZIK, P. LEMARINIER, F. CAPPELLO. *MPICH-V Project: a Multiprotocol Automatic Fault Tolerant MPI*, in "International Journal of High Performance Computing Applications", 2005, vol. 20, n^{o} 3, pp. 319–333

[7] F. CAPPELLO, S. DJILALI, G. FEDAK, T. HÉRAULT, F. MAGNIETTE, V. NÉRI, O. LODYGENSKY. *Computing on Large Scale Distributed Systems: XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid*, in "FGCS Future Generation Computer Science", 2004

[8] G. FURSIN, Y. KASHNIKOV, A. MEMON, Z. CHAMSKI, O. TEMAM, M. NAMOLARU, E. YOM-TOV, B. MENDELSON, A. ZAKS, E. COURTOIS, F. BODIN, P. BARNARD, E. ASHTON, E. BONILLA, J. THOMSON, C. WILLIAMS, M. O'BOYLE. *Milepost GCC: Machine Learning Enabled Self-tuning Compiler*, in "International Journal of Parallel Programming", 2011, vol. 39, pp. 296-327, 10.1007/s10766-010-0161-2, http://dx.doi.org/10.1007/s10766-010-0161-2

[9] L. GRIGORI, J. DEMMEL, X. S. LI. *Parallel Symbolic Factorization for Sparse LU Factorization with Static Pivoting*, in "SIAM Journal on Scientific Computing", 2007, vol. 29, n^{o} 3, pp. 1289-1314

[10] L. GRIGORI, J. DEMMEL, H. XIANG. *Communication Avoiding Gaussian Elimination*, in "Proceedings of the ACM/IEEE SC08 Conference", 2008

[11] L. GRIGORI, J. DEMMEL, H. XIANG. *CALU: a communication optimal LU factorization algorithm*, in "SIAM Journal on Matrix Analysis and Applications", 2011, vol. 32, pp. 1317-1350

[12] L. GRIGORI, F. NATAF. , *Generalized Filtering Decomposition*, May 2011, Session 7, http://hal.inria.fr/inria-00581744/en

[13] L. GRIGORI, F. NATAF. , *Generalized Filtering Decomposition*, Inria, March 2011, n^{o} RR-7569, 8 p. , http://hal.inria.fr/inria-00576894/en

[14] T. HÉRAULT, R. LASSAIGNE, S. PEYRONNET. *APMC 3.0: Approximate Verification of Discrete and Continuous Time Markov Chains*, in "Proceedings of the 3rd International Conference on the Quantitative Evaluation of SysTems (QEST'06)", California, USA, September 2006

[15] Q. NIU, L. GRIGORI, P. KUMAR, F. NATAF. *Modified tangential frequency filtering decomposition and its Fourier analysis*, in "Numerische Mathematik", 2010, vol. 116, n^{o} 1, pp. 123-148

[16] S. TOMOV, J. DONGARRA, M. BABOULIN. *Towards dense linear algebra for hybrid GPU accelerated manycore systems*, in "Parallel Computing", 2010, vol. 36, n^{o} 5&6, pp. 232–240

[17] B. WEI, G. FEDAK, F. CAPPELLO. *Scheduling Independent Tasks Sharing Large Data Distributed with BitTorrent*, in "IEEE/ACM Grid'2005 workshop Seattle, USA", 2005

Publications of the year

Articles in International Peer-Reviewed Journals

[18] G. ANTONIU, J. BIGOT, C. BLANCHET, L. BOUGÉ, F. BRIANT, F. CAPPELLO, A. COSTAN, F. DESPREZ, G. FEDAK, S. GAULT, K. KEAHEY, B. NICOLAE, C. PÉREZ, A. SIMONET, F. SUTER, B. TANG, R. TERREUX. *Towards Scalable Data Management for Map-Reduce-based Data-Intensive Applications on Cloud and Hybrid Infrastructures*, in "International Journal of Cloud Computing (IJCC)", 2013, vol. 2, n^{o} 2/3 [*DOI :* 10.1504/IJCC.2013.055265], http://hal.inria.fr/hal-00767029

[19] M. BABOULIN, J. DONGARRA, J. HERRMANN, S. TOMOV. *Accelerating linear system solutions using randomization technique*, in "ACM Transactions on Mathematical Software", February 2013, vol. 39, n^{o} 2 [*DOI :* 10.1145/2427023.2427025], http://hal.inria.fr/hal-00908496

[20] D. BARTHOU, O. BRAND-FOISSAC, O. PENE, G. GROSDIDIER, R. DOLBEAU, C. EISENBEIS, M. KRUSE, K. PETROV, C. TADONKI. *Automated Code Generation for Lattice Quantum Chromodynamics and beyond*, in "Journal of Physics: Conference Series", December 2013, LPT-Orsay-13-142, http://hal.inria.fr/hal-00926513

[21] G. BOSILCA, A. BOUTEILLER, É. BRUNET, F. CAPPELLO, J. DONGARRA, A. GUERMOUCHE, T. HÉRAULT, Y. ROBERT, F. VIVIEN, D. ZAIDOUNI. *Unified Model for Assessing Checkpointing Protocols at Extreme-Scale*, in "Journal of Concurrency and Computation: Practice and Experience", November 2013 [*DOI :* 10.1002/CPE.3173], http://hal.inria.fr/hal-00908447

[22] P. HAVE, R. MASSON, F. NATAF, M. SZYDLARSKI, H. XIANG, T. ZHAO. *Algebraic Domain Decomposition Methods for Highly Heterogeneous Problems*, in "SIAM Journal on Scientific Computing", 2013, vol. 35, n^{o} 3, pp. C284-C302, http://hal.inria.fr/hal-00611997

[23] B. NICOLAE, F. CAPPELLO. *BlobCR: Virtual Disk Based Checkpoint-Restart for HPC Applications on IaaS Clouds*, in "Journal of Parallel and Distributed Computing", February 2013, vol. 73, n^{o} 5, pp. 698-711 [*DOI :* 10.1016/J.JPDC.2013.01.013], http://hal.inria.fr/hal-00857964

International Conferences with Proceedings

[24] A. BOUTEILLER, F. CAPPELLO, J. DONGARRA, A. GUERMOUCHE, T. HÉRAULT, Y. ROBERT. *Multi-criteria checkpointing strategies: response-time versus resource utilization*, in "Euro-Par 2013", Aachen, Germany, S. VERLAG (editor), LNCS, 2013, vol. 8097, pp. 420-431 [*DOI :* 10.1007/978-3-642-40047-6_43], http://hal.inria.fr/hal-00926606

[25] S. DI, D. KONDO, F. CAPPELLO. *Characterizing Cloud Applications on a Google Data Center*, in "42nd International Conference on Parallel Processing (ICPP'13)", 2013, pp. 468-473 [*DOI :* 10.1109/ICPP.2013.56], http://hal.inria.fr/hal-00936827

[26] S. DI, Y. ROBERT, F. VIVIEN, D. KONDO, C.-L. WANG, F. CAPPELLO. *Optimization of Cloud Task Processing with Checkpoint-Restart Mechanism*, in "SC13 - Supercomputing - 2013", Denver, United States, ACM, November 2013 [*DOI :* 10.1145/2503210.2503217], http://hal.inria.fr/hal-00847635

[27] M. E. M. DIOURI, O. GLÜCK, L. LEFÈVRE, F. CAPPELLO. *ECOFIT: A Framework to Estimate Energy Consumption of Fault Tolerance protocols during HPC executions*, in "13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)", Delft, Netherlands, May 2013, http://hal.inria.fr/hal-00806500

[28] M. E. M. DIOURI, O. GLÜCK, L. LEFÈVRE, F. CAPPELLO. *Towards an Energy Estimator for Fault Tolerance Protocols*, in "18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)", Shenzhen, China, February 2013, pp. 313–314 [*DOI :* 10.1145/2442516.2442561], http://hal.inria.fr/hal-00806499

[29] A. W. MEMON, G. FURSIN. *Crowdtuning: systematizing auto-tuning using predictive modeling and crowdsourcing*, in "PARCO mini-symposium on "Application Autotuning for HPC (Architectures)"", Munich, Germany, September 2013, http://hal.inria.fr/hal-00944513

[30] B. NICOLAE, F. CAPPELLO. *AI-Ckpt: Leveraging Memory Access Patterns for Adaptive Asynchronous Incremental Checkpointing*, in "HPDC '13: 22th International ACM Symposium on High-Performance Parallel and Distributed Computing", New York, United States, April 2013, pp. 155-166 [*DOI :* 10.1145/2462902.2462918], http://hal.inria.fr/hal-00809847

[31] Y. WANG, M. BABOULIN, J. DONGARRA, J. FALCOU, Y. FRAIGNEAU, O. LE MAITRE. *A parallel solver for incompressible fluid flows*, in "International Conference on Computational Science (ICCS 2013)", Barcelona, Italy, June 2013 [*DOI :* 10.1016/J.PROCS.2013.05.207], http://hal.inria.fr/hal-00915356

Conferences without Proceedings

[32] L. GIRAUD, F. CAPPELLO. *Resilience at extreme scale : system level, algorithmic level or both ?*, in "SIAM Conference on Computational Science and Engineering - CSE 2013", Boston, United States, SIAM, March 2013, http://hal.inria.fr/hal-00799309

Research Reports

[33] D. BARTHOU, G. GROSDIDIER, K. PETROV, M. KRUSE, C. EISENBEIS, O. PÈNE, O. BRAND-FOISSAC, C. TADONKI, R. DOLBEAU. , *Automated Code Generation for Lattice QCD Simulation*, Inria, December 2013, n^{O} RR-8417, 13 p. , http://hal.inria.fr/hal-00918812

[34] J. BEAUQUIER, P. BLANCHARD, J. BURMAN. , *Self-stabilizing Leader Election in Population Protocols over Arbitrary Communication Graphs*, September 2013, http://hal.inria.fr/hal-00867287

[35] A. FERREIRA LEITE, C. TADONKI, C. EISENBEIS, A. C. M. A. DE MELO. , *A Fine-grained Approach for Power Consumption Analysis and Prediction*, Inria, December 2013, n^o RR-8416, 12 p. , http://hal.inria.fr/hal-00918810

[36] G. FURSIN. , *Collective Mind: cleaning up the research and experimentation mess in computer engineering using crowdsourcing, big data and machine learning*, August 2013, http://hal.inria.fr/hal-00850880

Other Publications

[37] D. BARTHOU, O. BRAND-FOISSAC, R. DOLBEAU, G. GROSDIDIER, C. EISENBEIS, M. KRUSE, O. PENE, K. PETROV, C. TADONKI. , *Automated Code Generation for Lattice Quantum Chromodynamics and beyond*, 2014, http://hal.inria.fr/hal-00930288

[38] G. FURSIN. *Keynote at HPSC 2013 at NTU, Taiwan: Systematizing tuning of computer systems using crowd-sourcing and statistics*, in "HPSC - Conference on Advanced Topics and Auto Tuning in High Performance and Scientific Computing - 2013", Taipei, Taiwan, March 2013, http://hal.inria.fr/hal-00819000

[39] G. FURSIN. *Tutorial at HPSC 2013 at NTU, Taiwan: Collective Mind: novel methodology, framework and repository to crowd-source auto-tuning*, in "HPSC - Conference on Advanced Topics and Auto Tuning in High Performance and Scientific Computing - 2013", Taipee, Taiwan, March 2013, http://hal.inria.fr/hal-00819002

[40] G. FURSIN, A. W. MEMON, C. GUILLON. , *Machine Learning for Compilation and Architecture: Myth or Reality?*, 2013, http://hal.inria.fr/hal-00907143

References in notes

[41] K. AIDA, A. TAKEFUSA, H. NAKADA, S. MATSUOKA, S. SEKIGUCHI, U. NAGASHIMA. *Performance evaluation model for scheduling in a global computing system*, in "International Journal of High Performance Computing Applications", 2000, vol. 14, No. 3, pp. 268-279, http://dx.doi.org/10.1177/109434200001400308

[42] A. D. ALEXANDROV, M. IBEL, K. E. SCHAUSER, C. J. SCHEIMAN. *SuperWeb: Research Issues in JavaBased Global Computing*, in "Concurrency: Practice and Experience", June 1997, vol. 9, n^o 6, pp. 535–553

[43] L. ALVISI, K. MARZULLO. , *Message Logging: Pessimistic, Optimistic and Causal*, 2001, Proc. 15th Int'l Conf. on Distributed Computing

[44] D. P. ANDERSON. , *BOINC*, 2011, http://boinc.berkeley.edu/

[45] A. BARAK, O. LA'ADAN. *The MOSIX multicomputer operating system for high performance cluster computing*, in "Future Generation Computer Systems", 1998, vol. 13, n^o 4–5, pp. 361–372

[46] A. BARATLOO, M. KARAUL, Z. M. KEDEM, P. WYCKOFF. *Charlotte: Metacomputing on the Web*, in "Proceedings of the 9th International Conference on Parallel and Distributed Computing Systems (PDCS-96)", 1996

[47] J. BEAUQUIER, C. GENOLINI, S. KUTTEN. , *Optimal reactive k-stabilization: the case of mutual exclusion. In Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing*, may 1999, pp. 199-208

[48] J. BEAUQUIER, T. HÉRAULT. , *Fault-Local Stabilization: the Shortest Path Tree.*, October 2002, Proceedings of the 21th Symposium of Reliable Distributed Systems

[49] G. BOSILCA, A. BOUTEILLER, F. CAPPELLO, S. DJILALI, G. FEDAK, C. GERMAIN, T. HÉRAULT, P. LEMARINIER, O. LODYGENSKY, F. MAGNIETTE, V. NÉRI, A. SELIKHOV. , *MPICH-V: Toward a Scalable Fault Tolerant MPI for Volatile Nodes*, 2002, in IEEE/ACM SC 2002

[50] A. BOUTEILLER, F. CAPPELLO, T. HÉRAULT, G. KRAWEZIK, P. LEMARINIER, F. MAGNIETTE. , *MPICH-V2: a Fault Tolerant MPI for Volatile Nodes based on Pessimistic Sender Based Message Logging*, November 2003, in IEEE/ACM SC 2003

[51] A. BOUTEILLER, P. LEMARINIER, G. KRAWEZIK, F. CAPPELLO. , *Coordinated Checkpoint versus Message Log for fault tolerant MPI*, December 2003, in IEEE Cluster

[52] T. BRECHT, H. SANDHU, M. SHAN, J. TALBOT. *ParaWeb: Towards World-Wide Supercomputing*, in "Proceedings of the Seventh ACM SIGOPS European Workshop on System Support for Worldwide Applications", 1996

[53] R. BUYYA, M. MURSHED. , *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*, Wiley Press, May 2002

[54] N. CAMIEL, S. LONDON, N. NISAN, O. REGEV. *The POPCORN Project: Distributed Computation over the Internet in Java*, in "Proceedings of the 6th International World Wide Web Conference", April 1997

[55] H. CASANOVA. , *Simgrid: A Toolkit for the Simulation of Application Scheduling. In Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid '01)*, May 2001, pp. 430–437

[56] K. M. CHANDY, L. LAMPORT. , *Distributed Snapshots: Determining Global States of Distr. systems*, 1985, ACM Trans. on Comp. Systems, 3(1):63–75

[57] B. O. CHRISTIANSEN, P. CAPPELLO, M. F. IONESCU, M. O. NEARY, K. E. SCHAUSER, D. WU. *Javelin: Internet-Based Parallel Computing Using Java*, in "Concurrency: Practice and Experience", November 1997, vol. 9, n^{o} 11, pp. 1139–1160

[58] J. W. DEMMEL, L. GRIGORI, M. HOEMMEN, J. LANGOU. *Communication-optimal parallel and sequential QR and LU factorizations*, in "SIAM Journal on Scientific Computing", 2012, short version of technical report UCB/EECS-2008-89 from 2008

[59] S. DOLEV. , *Self-stabilization*, 2000, M.I.T. Press

[60] G. FEDAK, C. GERMAIN, V. NÉRI, F. CAPPELLO. *XtremWeb: A Generic Global Computing System*, in "CCGRID'01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid", IEEE Computer Society, 2001, 582 p.

[61] I. FOSTER, A. IAMNITCHI. *On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing*, in "2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)", Berkeley, CA, February 2003

[62] V. K. GARG. , *Principles of distributed computing*, John Wiley and Sons, May 2002

[63] C. GENOLINI, S. TIXEUIL. , *A lower bound on k-stabilization in asynchronous systems*, October 2002, Proceedings of the 21th Symposium of Reliable Distributed Systems

[64] DOUGLAS P. GHORMLEY, D. PETROU, STEVEN H. RODRIGUES, AMIN M. VAHDAT, THOMAS E. ANDERSON. *GLUnix: A Global Layer Unix for a Network of Workstations*, in "Software Practice and Experience", 1998, vol. 28, n^{o} 9, pp. 929–961

[65] D. E. KEYES. , *A Science-based Case for Large Scale Simulation, Vol. 1, Office of Science, US Department of Energy, Report Editor-in-Chief*, July 30 2003

[66] S. KUTTEN, B. PATT-SHAMIR. , *Stabilizing time-adaptive protocols. Theoretical Computer Science 220(1)*, 1999, pp. 93-111

[67] S. KUTTEN, D. PELEG. , *Fault-local distributed mending. Journal of Algorithms 30(1)*, 1999, pp. 144-165

[68] N. LEIBOWITZ, M. RIPEANU, A. WIERZBICKI. *Deconstructing the Kazaa Network*, in "Proceedings of the 3rd IEEE Workshop on Internet Applications WIAPP'03", Santa Clara, CA, 2003

[69] M. LITZKOW, M. LIVNY, M. MUTKA. *Condor — A Hunter of Idle Workstations*, in "Proceedings of the Eighth Conference on Distributed Computing", San Jose, 1988

[70] NANCY A. LYNCH. , M. KAUFMANN (editor), *Distributed Algorithms*, 1996

[71] MESSAGE PASSING INTERFACE FORUM. , *MPI: A message passing interface standard*, June 12 1995, Technical report, University of Tennessee, Knoxville

[72] N. MINAR, R. MURKHART, C. LANGTON, M. ASKENAZI. , *The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations*, 1996

[73] H. PEDROSO, L. M. SILVA, J. G. SILVA. *Web-Based Metacomputing with JET*, in "Proceedings of the ACM", 1997

[74] B. QUÉTIER, M. JAN, F. CAPPELLO. , *One step further in large-scale evaluations: the V-DS environment*, Inria, December 2007, n^{o} RR-6365, http://hal.inria.fr/inria-00189670

[75] S. RATNASAMY, P. FRANCIS, M. HANDLEY, R. KARP, S. SHENKER. *A Scalable Content Addressable Network*, in "Proceedings of ACM SIGCOMM 2001", 2001

[76] A. ROWSTRON, P. DRUSCHEL. *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems*, in "IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)", 2001, pp. 329–350

[77] L. F. G. SARMENTA, S. HIRANO. *Bayanihan: building and studying Web-based volunteer computing systems using Java*, in "Future Generation Computer Systems", 1999, vol. 15, n^o 5–6, pp. 675–686

[78] S. SAROIU, P. K. GUMMADI, S. D. GRIBBLE. *A Measurement Study of Peer-to-Peer File Sharing Systems*, in "Proceedings of Multimedia Computing and Networking", San Jose, CA, USA, January 2002

[79] J. F. SHOCH, J. A. HUPP. *The Worm Programs: Early Experiences with Distributed Systems*, in "Communications of the Association for Computing Machinery", March 1982, vol. 25, n^o 3

[80] I. STOICA, R. MORRIS, D. KARGER, F. KAASHOEK, H. BALAKRISHNAN. *Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications*, in "Proceedings of the 2001 ACM SIGCOMM Conference", 2001, pp. 149–160

[81] G. TEL. , *Introduction to distributed algorithms*, 2000, Cambridge University Press

[82] Y.-M. WANG, W. K. FUCHS. , *Optimistic Message Logging for Independent Checkpointing in Message-Passing Systems*, 1992, pp. 147-154, Symposium on Reliable Distributed Systems

[83] Y. YI, T. PARK, H. Y. YEOM. , *A Causal Logging Scheme for Lazy Release Consistent Distributed Shared Memory Systems*, December 1998, In Proc. of the 1998 Int'l Conf. on Parallel and Distributed Systems

[84] B. Y. ZHAO, J. D. KUBIATOWICZ, A. D. JOSEPH. , *Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing*, UC Berkeley, April 2001, n^o UCB/CSD-01-1141