IN PARTNERSHIP WITH:
CNRS

Université Paris-Sud (Paris 11)

Université des sciences et technologies de Lille (Lille 1)

Activity Report 2012

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: October 02, 2003 .

1. Members

Research Scientists

Franck Cappello [Senior Researcher, HdR]
Christine Eisenbeis [Senior Researcher]
Laura Grigori [Senior Researcher, HdR]
Grigori Fursin [Junior Researcher, since December 2011]

Faculty Members

Brigitte Rozoy [Temporary Team Leader, Professor at Paris-Sud University, HdR]
Joffroy Beauquier [Professor at Paris-Sud University, HdR]
Thomas Hérault [Associate Professor at Paris-Sud University, delegated in the Inria project/team]
Serge Petiton [Professor at University of Science and Technology of Lille, HdR]
Sylvain Peyronnet [Associate Professor at Paris-Sud University, HdR]
Marc Baboulin [Associate Professor at Paris-Sud University, Inria Research Chair, HdR]
Cédric Bastoul [Associate Professor at Paris-Sud University, HdR]
Joël Falcou [Associate Professor at Paris-Sud University]

Engineers

Taj Khan [Inria Expert Engineer, since October 2012]
Rémi Lacroix [Ingénieur jeune diplômé, since October 2012]

PhD Students

Simplice Donfack [Inria Grant]
Adrien Rémy [MESR Grant (LRI)]
Amal Khabou [MESR Grant (LRI), since October 2009]
Antoine Baldacci [CIFRE IFP, since November 2009]
Sophie Moufawad [Inria Grant, since October 2011]
Michael Kruse [Grant of Université Paris-Sud XI]
Alessandro Ferreira Leite [Brasil, cotutelle, Université Paris-Sud XI, since October 2012]
Lénaïc Bagnères [Inria Grant, since Octobre 2012]
Tatiana Martsinkevich [since April 2012]
Peva Blanchard [(Digiteo) since September 2011]
Pierre Estérie [since October 2010]

Post-Doctoral Fellows

Tobias Berka [until October 2012]
Janna Burman [since November 2011]
Riadh Fezzani [until December 12]
Maxime Hugues [until May 2012]
Mathias Jacquelin [until December 2012]
Bogdan Nicolae [until June 2012]
Ala Rezmerita [until January 2012]
Meisam Sharify [until September 2012]
Mikolaj Szydlarski [until december 2012]

Administrative Assistant

Katia Evrat [Administrative assistant]

Others

Lénaïc Bagnères [M2R internship until September 2012]
Bada Gaye [until July 2012]
Rémi Lacroix [until August 2012]
Joël POUDROUX [until July 2012]
Herman SCHINCA [until December 2012]

2. Overall Objectives

2.1. Grand-Large General Objectives

Grand-Large was evaluated in september 2012 and ends in December 2012. Additional information can be found in the 2012 Grand-Large evaluation report.

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. Scientific Foundations

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 [88] 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 [77] , Glunix [71] and Mosix [50], to cite a few. A first approach to cross administration domains was proposed by Web Computing projects such as Jet [81], Charlotte [51], Javeline [65], Bayanihan [86], SuperWeb [47], ParaWeb [58] and PopCorn [60]. 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 [49] and XtremWeb [67]. 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 [68], 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 [87], Gnutella [87] and Kazaa [76], 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 [84], CHORD [89], PASTRY [85] and TAPESTRY [94] 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 [66]. 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 [79], 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 [62]. 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 [91], pessimistic [48], causal [92]. 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 [55], MPICH-V2 [56], MPICH-Vcausal, MPICH-Vcl [57], 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 ([69]). 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. [78], [90], [66] 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. [75], [74], [52], [53], [70]) 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 [73]. 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 [61], GridSim [59], Briks [46]. 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 [80] 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* [18] [8]. 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 [16]& [25]. 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.

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 [2]. 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

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 [83].

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.

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

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. Parallel solvers for solving linear systems of equations

Participant: Laura Grigori.

In the last several years, there has been significant research effort in the development of fully parallel direct solvers for computing the solution of large unsymmetric sparse linear systems of equations. In this context, we have designed and implemented a parallel symbolic factorization algorithm, which is suitable for general sparse unsymmetric matrices. The symbolic factorization is one of the steps that is sequential and represents a memory bottleneck. The code is intended to be used with very large matrices when because of the memory usage, the sequential algorithm is not suitable. This code is available in the SuperLU_DIST, a widely used software, developed at UC Berkeley and LBNL by Professor James W. Demmel and Dr. Xiaoye S. Li. The algorithm is presented in [72]. The SuperLU_DIST is available at http://crd.lbl.gov/~xiaoye/SuperLU/ .

4.8. 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 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.

4.9. 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.

4.10. CALU for multicore architectures

Participant: Laura GRIGORI [correspondant].

The communication avoiding algorithms are implemented in the form of a portable library. In its current form, this library is designed for multicore architectures and uses a hybrid scheduling technique that exploits well the data locality and can adapt to dynamic changes in the machine. The library will be publicly available since February 2012.

See also the web page http://www-rocq.inria.fr/who/Laura.Grigori/COALA2010/coala.html.

- Version: 1.0

4.11. MIDAPACK for CMB data analysis

Participants: Laura GRIGORI [correspondant], Mikolaj SZYDLARSKI [correspondant].

Midapack is a library aiming at the crucial stages down the CMB data analysis pipeline. In its current form, the library provides tools for computing spherical harmonic transforms on heterogeneous architectures, and algorithms for finding the solution to a generalized least squares problem. The algorithms are described in [36] and [44]. See also the web page http://pages.saclay.inria.fr/laura.grigori/soft.html.

- Version: 1.0

4.12. 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 [14]. 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 [26].

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

4.13. 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 [6].

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. Communication avoiding algorithms for linear algebra

Participants: Laura Grigori, Amal Khabou, Mathias Jacquelin, Sophie Moufawad.

The focus of this research is on the design of efficient parallel algorithms for solving problems in numerical linear algebra, as solving very large sets of linear equations and large least squares problems, often with millions of rows and columns. These problems arise in many numerical simulations, and solving them is very time consuming.

Our research focuses on developing new algorithms for linear algebra problems, that minimize the required communication, in terms of both latency and bandwidth. We have introduced in 2008 two communication avoiding algorithms for computing the LU and QR factorizations, that we refer to as CALU and CAQR (joint work with J. Demmel and M. Hoemmen from U.C. Berkeley, J. Langou from C.U. Denver, and H. Xiang then at Inria) [18] [8]. Since then, we continue designing communication avoiding algorithm for other operations in both dense and sparse linear algebra. The communication avoiding algorithms are now studied by several other groups, including groups at Inria, and they start being implemented and being available in public libraries as ScaLAPACK.

During 2012, our research [43] has focused on the design of the LU decomposition with panel rank revealing pivoting (LU_PRRP), an LU factorization algorithm based on strong rank revealing QR panel factorization. LU_PRRP is more stable than Gaussian elimination with partial pivoting (GEPP), with a theoretical upper bound of the growth factor of $(1+\tau b)^{(n/b)-1}$, where b is the size of the panel used during the block factorization, τ is a parameter of the strong rank revealing QR factorization, and n is the number of columns of the matrix. For example, if the size of the panel is $b = 64$, and $\tau = 2$, then $(1+2b)^{(n/b)-1} = (1.079)^{n-1} \ll 2^{n-1}$, where 2^{n-1} is the upper bound of the growth factor of GEPP. Our extensive numerical experiments show that the new factorization scheme is as numerically stable as GEPP in practice, but it is more resistant to pathological cases. The LU_PRRP factorization does only $O(n^2 b)$ additional floating point operations compared to GEPP. We have also introduced CALU_PRRP, a communication avoiding version of LU_PRRP that minimizes communication. CALU_PRRP is based on tournament pivoting, with the selection of the pivots at each step of the tournament being performed via strong rank revealing QR factorization. CALU_PRRP is more stable than CALU, the communication avoiding version of GEPP, with a theoretical upper bound of the growth factor of $(1+\tau b)^{\frac{n}{b}(H+1)-1}$, where H is the height of the reduction tree used during tournament pivoting. The upper bound of the growth factor of CALU is $2^{n(H+1)-1}$. CALU_PRRP is also more stable in practice and is resistant to pathological cases on which GEPP and CALU fail.

Our work has also focused on designing algorithms that are optimal over multiple levels of memory hierarchy and parallelism. In [32] we present an algorithm for performing the LU factorization of dense matrices that is suitable for computer systems with two levels of parallelism. This algorithm is able to minimize both the volume of communication and the number of messages transferred at every level of the two-level hierarchy of parallelism. We present its implementation for a cluster of multicore processors based on MPI and Pthreads. We show that this implementation leads to a better performance than routines implementing the LU factorization in well-known numerical libraries. For matrices that are tall and skinny, that is they have many more rows than columns, our algorithm outperforms the corresponding algorithm from ScaLAPACK by a factor of 4.5 on a cluster of 32 nodes, each node having two quad-core Intel Xeon EMT64 processors.

5.2. Preconditioning techniques for solving large systems of equations

Participants: Laura Grigori, Riadh Fezzanni, Sophie Moufawad.

A different direction of research is related to preconditioning large sparse linear systems of equations. This research is performed in the context of ANR PETALh project (2011-2012), which follows the ANR PETAL project (2008-2009). It is conducted in collaboration with Frederic Nataf from University Paris 6.

Several highly used preconditioners are for example the incomplete LU factorizations and Schwarz based approaches as used in domain decomposition. Most of these preconditioners are known to have scalability problems. The number of iterations can increase significantly when the size of the problem increases or when the number of independent domains is increased. 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 a different class of preconditioners, called direction preserving or filtering preconditioners. These preconditioners have the property of being identical to the input matrix on a given filtering vector. A judicious choice of the vector allows to alleviate the effect of low frequency modes on the convergence.

We consider in particular two classes of preconditioners. The first preconditoner is an incomplete decomposition that satisfies the filtering property [13]. The nested preconditioner has the same property for a specific vector of all ones. However the construction is different and takes advantage of a nested structure of the input matrix. The previous research on these methods considered only matrices arising from the discretization of PDEs on structured grids, where the matrix has a block tridiagonal structure. This structure imposes a sequential computation of the preconditioner and it is not suitable for the more general case of unsructured grids. Hence, while very efficient, the usage of these preconditioners was very limited. At the beginning of this research we have obtained several theoretical results for these methods that demonstrate their numerical behavior and convergence properties for cases arising from the discretization of PDEs on structured grids [13]. But the main result is the development of a generalized method [10], [11] that has two important properties: it allows the filtering property to be satisfied for any input matrix; the matrix can be reordered such that its computation is highly parallel. Experimental results show that the method is very efficient for certain classes of matrices, and shows good scalability results in terms of both problem size and number of processors. In addition to finalizing this work, our research also focused on extending the block filtering factorization to include other approximation techniques that allowed us to introduce a parameter whose tuning permits to solve very difficult problems.

5.3. MIcrowave Data Analysis for petaScale computers

Participants: Laura Grigori, Mikolaj Szydlarski, Meisam Shariffy.

Generalized least square problems with non-diagonal weights arise frequently in an estimation of two dimensional images from data of cosmological as well as astro- or geo- physical observations. As the observational data sets keep growing at Moore's rate, with their volumes exceeding tens and hundreds billions of samples, the need for fast and efficiently parallelizable iterative solvers is generally recognized.

In this work [36] we propose a new iterative algorithm for solving generalized least square systems with weights given by a block-diagonal matrix with Toeplitz blocks. Such cases are physically well motivated and correspond to measurement noise being piece-wise stationary – a common occurrence in many actual observations. Our iterative algorithm is based on the conjugate gradient method and includes a parallel two-level preconditioner (2lvl-PCG) constructed from a limited number of sparse vectors estimated from the coefficients of the initial linear system.

Our prototypical application is the map-making problem in the Cosmic Microwave Background data analysis. We show experimentally that our parallel implementation of 2lvl-PCG outperforms by a factor of up to 6 the standard one-level PCG in terms of both the convergence rate and the time to solution on up to $12,228$ cores of NERSC's Cray XE6 (Hopper) system displaying nearly perfect strong and weak scaling behavior in this regime.

5.4. Innovative linear system solvers for hybrid multicore/GPU architectures

Participant: Marc Baboulin.

The advent of new processor architectures (e.g. multicore, GPUs) requires the rethinking of most of the scientific applications and innovative methods must be proposed in order to take full advantage of current supercomputers [14].

To accelerate linear algebra solvers on current parallel machines, we introduced in public domain libraries a class of solvers based on statistical techniques. A first application concerns the solution of a square linear systems $Ax = b$. We study a random transformation of A that enables us to avoid pivoting and then to reduce the amount of communication [16]. 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. In the solver that we developed, PRBT combined with LU factorization with no pivoting take advantage of the latest generation of hybrid multicore/GPU machines and outperform existing factorization routines from current parallel library MAGMA.

A second application is related to solving symmetric indefinite systems via LDL^T factorization for which there was no existing parallel implementation in the dense library ScaLAPACK. We developed an efficient and innovative parallel tiled algorithm for solving symmetric indefinite systems on multicore architectures [54]& [25]. This solver avoids pivoting by using a multiplicative preconditioning based on symmetric randomization. This randomization prevents the communication overhead due to pivoting, is computationally inexpensive and requires very little storage. Following randomization, a tiled LDLT factorization is used that reduces synchronization by using static or dynamic scheduling. We compare Gflop/s performance of our solver with other types of factorizations on a current multicore machine and we provide tests on accuracy using LAPACK test cases.

5.5. MILEPOST GCC: machine learning enabled self-tuning compiler

Participant: Grigori Fursin [correspondant].

Tuning compiler optimizations for rapidly evolving hardware makes porting and extending an optimizing compiler for each new platform extremely challenging. Iterative optimization is a popular approach to adapting programs to a new architecture automatically using feedback-directed compilation. However, the large number of evaluations required for each program has prevented iterative compilation from widespread take-up in production compilers. Machine learning has been proposed to tune optimizations across programs systematically but is currently limited to a few transformations, long training phases and critically lacks publicly released, stable tools.

Our approach is to develop a modular, extensible, self-tuning optimization infrastructure to automatically learn the best optimizations across multiple programs and architectures based on the correlation between program features, run-time behavior and optimizations. In this paper we describe MILEPOST GCC, the first publicly-available open-source machine learning-based compiler. It consists of an Interactive Compilation Interface (ICI) and plugins to extract program features and exchange optimization data with the cTuning.org open public repository. It automatically adapts the internal optimization heuristic at function-level granularity to improve execution time, code size and compilation time of a new program on a given architecture. Part of the MILEPOST technology together with low-level ICI-inspired plugin framework is now included in the mainline GCC.

We developed machine learning plugins based on probabilistic and transductive approaches to predict good combinations of optimizations. Our preliminary experimental results show that it is possible to automatically reduce the execution time of individual MiBench programs on various machines from GRID5000, some by more than a factor of 2, while also improving compilation time and code size. We also present a realistic multi-objective optimization scenario for Berkeley DB library using MILEPOST GCC and improve execution time by approximately 17%, while reducing compilation time and code size by 12% and 7% respectively on Intel Xeon processor.

5.6. Loop Transformations: Convexity, Pruning and Optimization

Participant: Cédric Bastoul.

High-level loop transformations are a key instrument in mapping computational kernels to effectively exploit resources in modern processor architectures. However, determining appropriate compositions of loop transformations to achieve this remains a significantly challenging task; current compilers may achieve significantly lower performance than hand-optimized programs. To address this fundamental challenge, we first present a convex characterization of all distinct, semantics-preserving, multidimensional affine transformations. We then bring together algebraic, algorithmic, and performance analysis results to design a tractable optimization algorithm over this highly expressive space. The framework has been implemented and validated experimentally on a representative set of benchmarks run on state-of-the-art multi-core platforms.

5.7. Non-self-stabilizing and self-stabilizing gathering in networks of mobile agents–the notion of speed

Participants: Joffroy Beauquier, Janna Burman, Julien Clment, Shay Kutten.

In the population protocol model, each agent is represented by a finite state machine. Agents are anonymous and supposed to move in an asynchronous way. When two agents come into range of each other (“meet”), they can exchange information. One of the vast variety of motivating examples to the population protocols model is ZebraNet. ZebraNet is a habitat monitoring application where sensors are attached to zebras and collect biometric data (e.g. heart rate, body temperature) and information about their behavior and migration patterns (via GPS). The population protocol model is, in some sense, related to cloud computing and to networks characterized by asynchrony,large scale, the possibility of failures, in the agents as well as in the communications, with the constraint that each agent is resource limited.

In order to extend the computation power and efficiency of the population protocol model, various extensions were suggested. Our contribution is an extension of the population protocol model that introduces the notion of “speed”, in order to capture the fact that the mobile agents move at different speeds and/or have different communication ranges and/or move according to different patterns and/or visit different places with different frequencies. Intuitively, fast agents which carry sensors with big communication ranges communicate with other agents more frequently than other agents do. This notion is formalized by allocating a cover time, cv, to each mobile agent v. cv is the minimum number of events in the whole system that occur before agent v meets every other agent at least once. As a fundamental example, we have considered the basic problem of gathering information that is distributed among anonymous mobile agents and where the number of agents is unknown. Each mobile agent owns a sensed input value and the goal is to communicate the values (as a multi-set, one value per mobile agent) to a fixed non-mobile base station (BS), with no duplicates or losses.

Gathering is a building block for many monitoring applications in networks of mobile agents. For example, a solution to this problem can solve a transaction commit/abort task in MANETs, if the input values of agents are votes (and the number of agents is known to BS). Moreover, the gathering problem can be viewed as a formulation of the routing problem in Disruption Tolerant Networks.

We gave different solutions to the gathering in the model of mobile agents with speed and we proved that one of them is optimal.

5.8. Making Population Protocols Self-stabilizing

Participants: Joffroy Beauquier, Janna Burman, Shay Kutten, Brigitte Rozoy.

As stated in the previous paragraph, the application domains of the population protocol model are asynchronous large scale networks, in which failures are possible and must be taken into account. This work concerns failures and namely the technique of self-stabilization for tolerating them.

Developing self-stabilizing solutions (and proving them) is considered to be more challenging and complicated than developing classical solutions, where a proper initialization of the variables can be assumed. This remark holds for a large variety of models and hence, to ease the task of the developers, some automatic techniques have been proposed to transform programs into self-stabilizing ones.

We have proposed such a transformer for algorithms in the population protocol model introduced for dealing with resource-limited mobile agents. The model we consider is a variation of the original one in that there is a non mobile agent, the base station, and that the communication characteristics (e.g. moving speed, communication radius) of the agents are considered through the notion of cover time.

The automatic transformer takes as an input an algorithm solving a static problem and outputs a self-stabilizing solution for the same problem. To the best of our knowledge, it is the first time that such a transformer for self-stabilization is presented in the framework of population protocols. We prove that the transformer we propose is correct and we make the complexity analysis of the stabilization time.

5.9. Self-stabilizing synchronization in population protocols with cover times

Participants: Joffroy Beauquier, Janna Burman, Shay Kutten, Brigitte Rozoy.

Synchronization is widely considered as an important service in distributed systems which may simplify protocol design. Phase clock is a general synchronization tool that provides a form of a logical time. We have developed a self-stabilizing phase clock algorithm suited to the model of population protocols with cover time. We have shown that a phase clock is impossible in the model with only constant-state agents. Hence, we assumed an existence of resource unlimited agent - the base station. The clock size and duration of each phase of the proposed phase clock tool are adjustable by the user. We provided application examples of this tool and demonstrate how it can simplify the design of protocols. In particular, it yields a solution to Group Mutual Exclusion problem.

5.10. Impossibility of consensus for population protocol with cover times

Participants: Joffroy Beauquier, Janna Burman.

We have extended the impossibility result for asynchronous consensus of Fischer, Lynch and Paterson (FLP) to the asynchronous model of population protocols with cover times. We noted that the proof of FLP does not apply. Indeed, the key lemma stating that two successive factors in an execution, involving disjoint subsets of agents, commute, is no longer true, because of the cover time property. Then we developed a completely different approach and we proved that there is no general solution to consensus for population protocols with cover times, even if there is a single possible crash. We noted that this impossibility result also applies to randomized asynchronous consensus, contrary to what happens in the classical message-passing or shared memory communication models, in which the problem is solvable inside some bounds on the number of faulty processes. Then, for circumventing these impossibility results, we introduced the phase clock oracle and the S oracle, and we shown how they allow to design solutions.

5.11. Routing and synchronization in large scale networks of very cheap mobile sensors

Participants: Joffroy Beauquier, Brigitte Rozoy.

In a next future, large networks of very cheap mobile sensors will be deployed for various applications, going from wild life preserving or environmental monitoring up to medical or industrial system control. Each sensor will cost only a few euros, allowing a large scale deployment. They will have only a few bit of memory, no identifier, weak capacities of computation and communication, no real time clock and will be prone to failures. Moreover such networks will be fundamentally dynamic. The goal of this subject is to develop the basic protocols and algorithms for rudimentary distributed systems for such networks. The studied problems are basic ones, like data collection, synchronization (phase clock, mutual exclusion, group mutual exclusion), fault tolerance (consensus), automatic transformers, always in a context of possible failures. A well known model has already been proposed for such networks, the population protocol model. In this model, each sensor is represented by a finite state machine. Sensors are anonymous and move in an asynchronous way. When two sensors come into range of each other ("meet"), they can exchange information. One of the vast variety of motivating examples for this model is ZebraNet. ZebraNet is a habitat monitoring application in

which sensors are attached to zebras in order to collect biometric data (e.g., heart rate, body temperature) and information about their behavior and migration patterns. Each pair of zebras meets from time to time. During such meetings (events), ZebraNet's agents (zebras' attached sensors) exchange data. Each agent stores its own sensor data as well as data of other sensors that were in range in the past. They upload data to a base station whenever it is nearby. It was shown that the set of applications that can be solved in the original model of population protocols is rather limited. Other models (such as some models of Delay/Disruption-Tolerant Networks - DTNs), where each node maintains links and connections even to nodes it may interact with only intermittently, do not seem to suit networks with small memory agents and a very large (and unknown) set of anonymous agents. That is why we enhance the model of population protocols by introducing a notion of "speed". We try to capture the fact that the mobile agents move at different speeds and/or have different communication ranges and/or move according to different patterns and/or visit different places with different frequencies. Intuitively, fast agents which carry sensors with large communication ranges communicate with other agents more frequently than other agents do. This notion is formalized by the notion of cover time for each agent. The cover time of an agent is the unknown number of events (pairwise meetings) in the whole system that occur (during any execution interval) before agent v meets every other agent at least once. The model we propose is somehow validated by some recent statistical results, obtained from empirical data sets regarding human or animal mobility. An important consequence of our approach is that the analytic complexity of the protocols designed in this model is possible, independently of any simulation or experimentation. For instance, we consider the fundamental problem of gathering different pieces of information, each sensed by a different anonymous mobile agent, and where the number of agents is unknown. The goal is to communicate the sensed values (as a multi-set, one value per mobile agent) to a base station, with no duplicates or losses. Gathering is a building block for many monitoring applications in networks of mobile agents. Moreover, the gathering problem can be viewed as a special case of the routing problem in DTNs, in which there is only one destination, the base station. Then we are able to compute the complexity of solutions we propose, as well as those of solutions used in experimental projects (like ZebraNet), and to compare them. The algorithms we present are self-stabilizing. Such algorithms have the important property of operating correctly regardless of their initial state (except for some bounded period). In practice, self-stabilizing algorithms adjust themselves automatically to any changes or corruptions of the network components (excluding the algorithm's code). These changes are assumed to cease for some sufficiently long period. Self-stabilization is considered for two reasons. First, mobile agents are generally fragile, subject to failures and hard to initialize. Second, systems of mobile agents are by essence dynamic, some agents leave the system while new ones are introduced. Self-stabilization is a well adapted framework for dealing with such situations.

5.12. Self-Stabilizing Control Infrastructure for HPC

Participants: Thomas Hérault, Camille Coti.

High performance computing platforms are becoming larger, leading to scalability and fault-tolerance issues for both applications and runtime environments (RTE) dedicated to run on such machines. After being deployed, usually following a spanning tree, a RTE needs to build its own communication infrastructure to manage and monitor the tasks of parallel applications. Previous works have demonstrated that the Binomial Graph topology (BMG) is a good candidate as a communication infrastructure for supporting scalable and fault-tolerant RTE.

In this work, we presented and analyzed a self-stabilizing algorithm to transform the underlying communication infrastructure provided by the launching service (usually a tree, due to its scalability during launch time) into a BMG, and maintain it in spite of failures. We demonstrated that this algorithm is scalable, tolerates transient failures, and adapts itself to topology changes.

The algorithms are scalable, in the sense that all process memory, number of established communication links, and size of messages are logarithmic with the number of elements in the system. The number of synchronous rounds to build the system is also logarithmic, and the number of asynchronous rounds in the worst case is square logarithmic with the number of elements in the system. Moreover, the salf-stabilizing property of the algorithms presented induce fault-tolerance and self-adaptivity. Performance evaluation based on simulations

predicts a fast convergence time (1/33s for 64K nodes), exhibiting the promising properties of such self-stabilizing approach.

We pursue this work by implementing and evaluating the algorithms in the STCI runtime environment to validate the theoretical results.

5.13. Large Scale Peer to Peer Performance Evaluations

Participant: Serge Petiton.

5.13.1. Large Scale Grid Computing

Recent progress has made possible to construct high performance distributed computing environments, such as computational grids and cluster of clusters, which provide access to large scale heterogeneous computational resources. Exploration of novel algorithms and evaluation of performance is a strategic research for the future of computational grid scientific computing for many important applications [82]. We adapted [63] an explicit restarted Lanczos algorithm on a world-wide heterogeneous grid platform. This method computes one or few eigenpairs of a large sparse real symmetric matrix. We take the specificities of computational resources into account and deal with communications over the Internet by means of techniques such as out-of-core and data persistence. We also show that a restarted algorithm and the combination of several paradigms of parallelism are interesting in this context. We perform many experimentations using several parameters related to the Lanczos method and the configuration of the platform. Depending on the number of computed Ritz eigenpairs, the results underline how critical the choice of the dimension of the working subspace is. Moreover, the size of platform has to be scaled to the order of the eigenproblem because of communications over the Internet.

5.13.2. High Performance Cluster Computing

Grid computing focuses on making use of a very large amount of resources from a large-scale computing environment. It intends to deliver high-performance computing over distributed platforms for computation and data-intensive applications. We propose [93] an effective parallel hybrid asynchronous method to solve large sparse linear systems by the use of a Grid Computing platform Grid5000. This hybrid method combines a parallel GMRES(m) (Generalized Minimum RESidual) algorithm with the Least Square method that needs some eigenvalues obtained from a parallel Arnoldi algorithm. All of these algorithms run on the different processors of the platform Grid5000. Grid5000, a 5000 CPUs nation-wide infrastructure for research in Grid computing, is designed to provide a scientific tool for computing. We discuss the performances of this hybrid method deployed on Grid5000, and compare these performances with those on the IBM SP series supercomputers.

5.13.3. Large Scale Power aware Computing

Energy conservation is a dynamic topic of research in High Performance Computing and Cluster Computing. Power-aware computing for heterogeneous world-wide Grid is a new track of research. We have studied and evaluated the impact of the heterogeneity of the computing nodes of a Grid platform on the energy consumption. We propose to take advantage of the slack-time caused by the heterogeneity in order to save energy with no significant loss of performance by using Dynamic Voltage Scaling (DVS) in a distributed eigensolver [64]. We show that using DVS only during the slack-time does not penalize the performances but it does not provide significant energy savings. If DVS is applied to all the execution, we get important global and local energy savings (respectively up to 9% and 20%) without a significant rise of the wall-clock times.

5.14. High Performance Linear Algebra on the Grid

Participants: Thomas Hérault, Camille Coti.

Previous studies have reported that common dense linear algebra operations do not achieve speed up by using multiple geographical sites of a computational grid. Because such operations are the building blocks of most scientific applications, conventional supercomputers are still strongly predominant in high-performance computing and the use of grids for speeding up large-scale scientific problems is limited to applications exhibiting parallelism at a higher level.

In this work, we have identified two performance bottlenecks in the distributed memory algorithms implemented in ScaLAPACK, a state-of-the-art dense linear algebra library. First, because ScaLAPACK assumes a homogeneous communication network, the implementations of ScaLAPACK algorithms lack locality in their communication pattern. Second, the number of messages sent in the ScaLAPACK algorithms is significantly greater than other algorithms that trade flops for communication.

This year, we presented a new approach for computing a QR factorization one of the main dense linear algebra kernels of tall and skinny matrices in a grid computing environment that overcomes these two bottlenecks. Our contribution is to articulate a recently proposed algorithm (Communication Avoiding QR) with a topology-aware middleware (QCG-OMPI) in order to confine intensive communications (ScaLAPACK calls) within the different geographical sites.

An experimental study conducted on the Grid5000 platform shows that the resulting performance increases linearly with the number of geographical sites on large-scale problems (and is in particular consistently higher than ScaLAPACKs).

5.15. Emulation of Volatile Systems

Participants: Thomas Largillier, Benjamin Quetier, Sylvain Peyronnet, Thomas Hérault, Franck Cappello.

In the process of developping grid applications, people need to often evaluate the robustness of their work. Two common approaches are simulation, where one can evaluate his software and predict behaviors under conditions usually unachievable in a laboratory experiment, and experimentation, where the actual application is launched on an actual grid. However simulation could ignore unpredictable behaviors due to the abstraction done and experimation does not guarantee a controlled and reproducible environment, and simulation often introduces a high level of abstraction that make the discovery and study of unexpected, but real, behaviors a rare event.

In this work, we proposed an emulation platform for parallel and distributed systems including grids where both the machines and the network are virtualized at a low level. The use of virtual machines allows us to test highly accurate failure injection since we can destroy virtual machines, and network virtualization provides low-level network emulation. Failure accuracy is a criteria that evaluates how realistic a fault is. The accuracy of our framework has been evaluated through a set of micro benchmarks and a very stable P2P system called Pastry.

We are in the process of developping a fault injection tool to work with the platform. it will be an extension of the work started in the tool Fail. The interest of this work is that using Xen virtual machines will allow to model strong adversaries since it is possible to have virtual machines with shared memory. These adversaries will be stronger since they will be able to use global fault injection strategies.

5.16. Exascale Systems

Participant: Franck Cappello.

Over the last 20 years, the open-source community has provided more and more software on which the world's high-performance computing systems depend for performance and productivity. The community has invested millions of dollars and years of effort to build key components. Although the investments in these separate software elements have been tremendously valuable, a great deal of productivity has also been lost because of the lack of planning, coordination, and key integration of technologies necessary to make them work together smoothly and efficiently, both within individual petascale systems and between different systems. A repository gatekeeper and an email discussion list can coordinate open-source development within a single project, but there is no global mechanism working across the community to identify critical holes in the overall software environment, spot opportunities for beneficial integration, or specify requirements for more careful coordination. It seems clear that this completely uncoordinated development model will not provide the software needed to support the unprecedented parallelism required for peta/exascale computation on millions of cores, or the flexibility required to exploit new hardware models and features, such as transactional

memory, speculative execution, and GPUs. We presented a rational promoting that the community must work together to prepare for the challenges of exascale computing, ultimately combing their efforts in a coordinated International Exascale Software Project.

Over the past few years resilience has became a major issue for high-performance computing (HPC) systems, in particular in the perspective of large petascale systems and future exascale systems. These systems will typically gather from half a million to several millions of central processing unit (CPU) cores running up to a billion threads. From the current knowledge and observations of existing large systems, it is anticipated that exascale systems will experience various kind of faults many times per day. It is also anticipated that the current approach for resilience, which relies on automatic or application level checkpoint/restart, will not work because the time for checkpointing and restarting will exceed the mean time to failure of a full system. This set of projections leaves the community of fault tolerance for HPC systems with a difficult challenge: finding new approaches, which are possibly radically disruptive, to run applications until their normal termination, despite the essentially unstable nature of exascale systems. Yet, the community has only five to six years to solve the problem. In order to start addressing this challenge, we synthesized the motivations, observations and research issues considered as determinant of several complimentary experts of HPC in applications, programming models, distributed systems and system management.

As a first step to adress the resilience challenge, we conducted a comprehensive study of the state of the art . The emergence of petascale systems and the promise of future exascale systems have reinvigorated the community interest in how to manage failures in such systems and ensure that large applications, lasting several hours or tens of hours, are completed successfully. Most of the existing results for several key mechanisms associated with fault tolerance in high-performance computing (HPC) platforms follow the rollback-recovery approach. Over the last decade, these mechanisms have received a lot of attention from the community with different levels of success. Unfortunately, despite their high degree of optimization, existing approaches do not fit well with the challenging evolutions of large-scale systems. There is room and even a need for new approaches. Opportunities may come from different origins: diskless checkpointing, algorithmic-based fault tolerance, proactive operation, speculative execution, software transactional memory, forward recovery, etc. We provided the following contributions: (1) we summarize and analyze the existing results concerning the failures in large-scale computers and point out the urgent need for drastic improvements or disruptive approaches for fault tolerance in these systems; (2) we sketch most of the known opportunities and analyze their associated limitations; (3) we extract and express the challenges that the HPC community will have to face for addressing the stringent issue of failures in HPC systems.

6. Partnerships and Cooperations

6.1. Regional Initiatives

6.1.1. Activities starting in 2009

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

6.1.2. Other activities

- **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).
- **ANR SPADES** Coordinated by LIP-ENS Lyon. (Sylvain Peyronnet, Franck Cappello, Ala Rezmerita)
- **Défi ANR SECSI** Participant to this challenge. From September 2008 to August 2010. Managed by the SAIC. (Thomas Hérault, Sylvain Peyronnet, Sébastien Tixeuil)

- **ANR Cosinus project MIDAS - MIcrowave Data Analysis for petaScale computers** December 2009 - December 2012 (http://www.apc.univ-paris7.fr/APC_CS/Recherche/Adamis/MIDAS09/index.html). Collaboration with APC, University Paris 7 and Lawrence Berkeley Laboratory. This is an interdisciplinary project devised to bring together cosmologists, computational physicists, computer scientists and applied mathematiciancs to face the challenge of the tremendous volume of data as anticipated from current and forthcoming Cosmic Microwave Background (CMB) experiments. (Laura Grigori, Coordinator for Inria Saclay, F. Cappello, J. Falcou, T. Hérault, S. Peyronnet)
- **ANR Cosinus project PETALh - PETascale ALgorithms for preconditioning for scientific applications** January 2011- December 2012. Collaboration with Laboratoire Lions - Universite 6, IFP, Inria Bordeaux and CEA, UC Berkeley and Argonne. The goal is to investigate preconditioning techniques on multicore architectures and apply them on real world applications from IFP, CEA and Argonne. (Laura Grigori, Principal Investigator)
- **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)

  PI L. Grigori
- Inria Associated Team **"F-J Grid"** with University of Tsukuba, head: Franck Cappello
- Inria funding, **MPI-V**, collaboration with UTK, LALN and ANL, head: Franck Cappello
- ANR CIS Project **FF2A3**, 3 years (2007 - 2010), PI F. Hecht, subproject head L. Grigori
- **HipCal**, ANR CIS, 3 years (2006-2009), http://hipcal.lri.fr/wakka.php?wiki=PagePrincipale, Franck Cappello

6.2. International Initiatives

6.2.1. Inria Associate Teams

- **Inria associated team COALA with Prof. J. Demmel, UC Berkeley, 2010-2013**. This project is proposed in the context of developing Communication Optimal Algorithms for Linear Algebra. The funding covers visits in both directions. The following visits of PhD students and postdoctoral researcher took place in the context of this associated team:
  - Visit of M. Jacquelin to UC Berkeley (August 2011, for 1 month).
  - Visit of S. Moufawad (November 2012, for 1 month).

6.3. European Initiatives

6.3.1. Collaborations in European Programs, except FP7

Program: ITEA2

Project acronym: MANY

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

Duration: 01/09/2011 - 31/08/2014

Coordinator: XDIN AB (formerly ENEA)

Other partners: Universitat Auto`noma de Barcelona (UAB), CEPHIS group (Spain), CAPS-Entreprise, (France), Inria, Grand Large (France), Institut Mines-Te´le´com/Te´le´com Sud Paris (IMT/TSP), Computer Science Department (France), THALES Communications & Security, (France), XDIN AB, (Sweden), ETRI, (Korea), Seven Core Co, Ltd, (Korea), TestMidas Co, Ltd, (Korea), ST-Ericsson BV, (Netherlands), Vector Fabrics BV, (Netherlands), Technische Universiteit Eindhoven, (Netherlands), University of Mons (UMONS), POLE-TI (Belgium)

Abstract: The ability to reuse existing software code has grown in importance as software applications become more complex. With the arrival of many-core semiconductor architectures, application developers face an additional problem: how to rewrite software applications to exploit the increased parallel processing available. The MANY project is developing an improved programming environment for the embedded-systems realm; one which will facilitate faster development of applications for a variety of hardware platforms. (Cédric Bastoul, Léna”ic Bagnères, Taj Khan)

6.4. International Research Visitors

6.4.1. Internships

German SCHINCA (*Date_begin_end ???*)

Subject: Minimizing communication in scientific computing

Institution: University of Buenos Aires (Argentina)

German SCHINCA (*Date_begin_end ???*)

Alessandro Ferreira Leite (October 2012-December 2012)

Subject: Energy issues in Cloud Computing

Institution: University of Brasilia (Brasil)

7. Dissemination

7.1. Scientific Animation

Christine Eisenbeis

- NPC 2012, the 9th Ifip International Conference on Network and Parallel Computing, Gwangju, Korea, September 6-8, 2012, program committee;
- UCNC 2012, 11th International Conference on Unconventional Computation and Natural Computation, Orléans, September 3-7, 2012, program committee;
- member of IFIP WG 10.3;
- IJPP (International Journal on Parallel Programming) editorial board.

Laura Grigori

- Member of Program Committee of IEEE International Parallel and Distributed Processing Symposium IPDPS 2012.
- Member of Program Committee of IEEE/ACM SuperComputing SC12 Conference, 2012.
- Member of the Best Paper Awards and Best Student Paper Award Jury, IEEE/ACM Supercomputing 2012, Conflict Chair for Algorithms, Supercomputing 2012.
- Member of Program Committee of HiPC 2012, 19th IEEE Int’l Conference on High Performance Computing.

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. 6 heures de cours.

Master: L. Grigori teaches in "Calcul Haute Performance" of M2 recherche NSI of University Paris Sud 11.

Polytech 5th year: L. Grigori teaches the "Parallel Computing" class.

7.2.2. Supervision

HdR : Marc Baboulin, “Fast and reliable solutions for numerical linear algebra solvers in high-performance computing”, université Paris-Sud 11, 5 décembre 2012.

HdR : Cédric Bastoul, “Contributions à l’optimisation des programmes de haut niveau”, université Paris-Sud 11, 12 décembre 2012.

PhD: Simplice Donfack, Solving Systems of Equations on Petascale Machines, University Paris Sud 11, 7 March 2012, PhD Supervisor: L. Grigori

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

PhD in progress: Sophie Moufawad, Communication Avoiding preconditioners, University Paris Sud 11, Supervisor: L. Grigori

7.3. Popularization

Christine Eisenbeis a contribué au manuel scolaire d’enseignement de l’informatique en terminale S [39]. Elle est intervenue à la table ronde “Qu’en disent les femmes scientifiques” du colloque "Effets de genre dans les sciences et les technologies", organisé par le Centre Interdisciplinaire d’Étude de l’Évolution des Idées, des Sciences et des Techniques (CIEEIST, Centre d’Alembert) à Orsay les 9 et 10 mai 2012.

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, p. 14-24.

[2] 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, p. 685–699.

[3] 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, p. 481-494.

[4] 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, p. 319–333.

[5] 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.

[6] 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, p. 296-327, 10.1007/s10766-010-0161-2, http://dx.doi.org/10.1007/s10766-010-0161-2.

[7] 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° 3, p. 1289-1314.

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

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

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

[11] L. GRIGORI, F. NATAF. *Generalized Filtering Decomposition*, Inria, March 2011, n° RR-7569, 8, http://hal.inria.fr/inria-00576894/en.

[12] 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.

[13] Q. NIU, L. GRIGORI, P. KUMAR, F. NATAF. *Modified tangential frequency filtering decomposition and its Fourier analysis*, in "Numerische Mathematik", 2010, vol. 116, n° 1, p. 123-148.

[14] S. TOMOV, J. DONGARRA, M. BABOULIN. *Towards dense linear algebra for hybrid GPU accelerated manycore systems*, in "Parallel Computing", 2010, vol. 36, n° 5&6, p. 232–240.

[15] 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

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

[17] M. BAHI, C. EISENBEIS. *Impact of Reverse Computing on Information Locality in Register Allocation for High Performance Computing*, in "International Journal of Parallel Programming", August 2012 [*DOI :* 10.1007/s10766-012-0212-y], http://hal.inria.fr/hal-00776246.

[18] 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.

[19] A. DENISE, M.-C. GAUDEL, S.-D. GOURAUD, R. LASSAIGNE, J. OUDINET, S. PEYRONNET. *Coverage-biased random exploration of large models and application to testing*, in "Software Tools for Technology Transfer (STTT)", 2012, vol. 14, n° 1, p. 73-93, http://hal.inria.fr/inria-00560621.

[20] P. ESTÉRIE, M. GAUNARD, J. FALCOU, J.-T. LAPRESTÉ. *Exploiting Multimedia Extensions in C++: A Portable Approach*, in "Computing in Science & Engineering", 2012, vol. 14, n° 5, p. 72–77.

[21] K. HAMIDOUCHE, F. M. MENDOCA, J. FALCOU, A. C. M. A. DE MELO, D. ÉTIEMBLE. *Parallel Smith-Waterman Comparison on Multicore and Manycore Computing Platforms with BSP++*, in "International Journal of Parallel Programming", 2012, p. 1–26.

[22] E. PARK, J. CAVAZOS, L.-N. POUCHET, C. BASTOUL, A. COHEN, P. SADAYAPPAN. *Predictive Modeling in a Polyhedral Optimization Space*, in "International Journal of Parallel Programming", 2012, Accepted for publication.

[23] V.-T. TRAN, B. NICOLAE, G. ANTONIU. *Towards Scalable Array-Oriented Active Storage: the Pyramid Approach*, in "ACM Operating Systems Review (OSR)", 2012, vol. 46, n° 1, p. 19-25 [*DOI :* 10.1145/2146382.2146387], http://hal.inria.fr/hal-00640900.

International Conferences with Proceedings

[24] 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 "1st International IBM Cloud Academy Conference - ICA CON 2012", Research Triangle Park, North Carolina, United States, 2012, http://hal.inria.fr/hal-00684866.

[25] 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, p. 14-24.

[26] 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, p. 17–26.

[27] L. BAUTISTA GOMEZ, B. NICOLAE, N. MARUYAMA, F. CAPPELLO, S. MATSUOKA. *Scalable Reed-Solomon-based Reliable Local Storage for HPC Applications on IaaS Clouds*, in "Euro-Par '12: 18th International Euro-Par Conference on Parallel Processing", Rhodes, Greece, May 2012, http://hal.inria.fr/hal-00703119.

[28] L. BLIN, J. BURMAN, N. NISSE. *Brief Announcement: Distributed Exclusive and Perpetual Tree Searching*, in "DISC - 26th International Symposium on Distributed Computing", Salvador, Brazil, 2012, http://hal.inria.fr/hal-00741982.

[29] L. BLIN, J. BURMAN, N. NISSE. *Nettoyage perpétuel de réseaux*, in "14èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel)", La Grande Motte, France, F. MATHIEU, N. HANUSSE (editors), 2012, 4, http://hal.inria.fr/hal-00687134.

[30] M. E. M. DIOURI, O. GLÜCK, L. LEFÈVRE, F. CAPPELLO. *Energy considerations in Checkpointing and Fault tolerance protocols*, in "2nd International Workshop on Fault Tolerance for HPC at eXtreme Scale (FTXS), co-located with the 42th International conference on Dependable Systems and Networks (DSN 2012)", Boston, United States, June 2012, http://hal.inria.fr/hal-00748006.

[31] S. DONFACK, L. GRIGORI, W. D. GROPP, V. KALE. *Hybrid static/dynamic scheduling for already optimized dense matrix factorization*, in "IEEE International Parallel and Distributed Processing Symposium IPDPS", 2012.

[32] S. DONFACK, L. GRIGORI, A. KHABOU. *Communication avoiding through a multilevel LU factorization*, in "Proceedings of EuroPar Conference", 2012.

[33] M. DORIER, G. ANTONIU, F. CAPPELLO, M. SNIR, L. ORF. *Damaris: How to Efficiently Leverage Multicore Parallelism to Achieve Scalable, Jitter-free I/O*, in "CLUSTER - IEEE International Conference on Cluster Computing", Beijing, China, IEEE, September 2012, http://hal.inria.fr/hal-00715252.

[34] P. ESTÉRIE, M. GAUNARD, J. FALCOU, J.-T. LAPRESTÉ, B. ROZOY. *Boost. SIMD: generic programming for portable SIMDization*, in "Proceedings of the 21st international conference on Parallel architectures and compilation techniques", ACM, 2012, p. 431–432.

[35] G. FABBIAN, M. SZYDLARSKI, R. STOMPOR, L. GRIGORI, J. FALCOU. *Spherical Harmonic Transforms with S2HAT (Scalable Spherical Harmonic Transform) Library*, in "Astronomical Data Analysis Software and Systems XXI", 2012, vol. 461, 61.

[36] L. GRIGORI, R. STOMPOR, M. SZYDLARSKI. *A parallel two-level preconditioner for Cosmic Microwave Background map-making*, in "Proceedings of the ACM/IEEE Supercomputing SC12 Conference", 2012.

[37] B. NICOLAE, F. CAPPELLO. *A Hybrid Local Storage Transfer Scheme for Live Migration of I/O Intensive Workloads*, in "HPDC'12: The 21st International ACM Symposium on High-Performance Parallel and Distributed Computing", Delft, Netherlands, January 2012, http://hal.inria.fr/hal-00686654.

[38] H. YE, L. LACASSAGNE, D. ÉTIEMBLE, L. CABARET, J. FALCOU, A. ROMERO, O. FLORENT. *Impact of high level transforms on high level synthesis for motion detection algorithm*, in "Design and Architectures for Signal and Image Processing (DASIP), 2012 Conference on", IEEE, 2012, p. 1–8.

Scientific Books (or Scientific Book chapters)

[39] G. DOWEK, J.-P. ARCHAMBAULT, E. BACCELLI, C. CIMELLLI, A. COHEN, C. EISENBEIS, T. VIÉVILLE, B. WACK. *Informatique et Sciences du Numérique - Spécialité ISN en Terminale S*, Eyrolles, August 2012, 303, http://hal.inria.fr/hal-00765220.

Research Reports

[40] J. BEAUQUIER, J. BURMAN, L. ROSAZ, B. ROZOY. *Non-deterministic Population Protocols (Extended Version)*, Université Paris-Sud 11 and CNRS and Inria, September 2012, http://hal.inria.fr/hal-00736261.

[41] 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*, Inria, October 2012, n^{o} RR-7950, http://hal.inria.fr/hal-00696154.

[42] M. DORIER, G. ANTONIU, F. CAPPELLO, M. SNIR, L. ORF. *Damaris: Leveraging Multicore Parallelism to Mask I/O Jitter*, Inria, April 2012, n^{o} RR-7706, 36, http://hal.inria.fr/inria-00614597.

[43] A. KHABOU, J. DEMMEL, L. GRIGORI, M. GU. *LU factorization with panel rank revealing pivoting and its communication avoiding version*, Inria, January 2012, n° RR-7867, http://hal.inria.fr/hal-00723564.

[44] M. SZYDLARSKI, P. ESTÉRIE, J. FALCOU, L. GRIGORI, R. STOMPOR. *Parallel Spherical Harmonic Transforms on heterogeneous architectures (GPUs/multi-core CPUs)*, Inria, May 2012, n° RR-7635, 31, http://hal.inria.fr/inria-00597576.

Other Publications

[45] C. BASTOUL. *Contributions to High-Level Program Optimization. Habilitation Thesis. Paris-Sud University, France*, December 2012.

References in notes

[46] 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, p. 268-279, http://dx.doi.org/10.1177/109434200001400308.

[47] 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° 6, p. 535–553.

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

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

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

[51] 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.

[52] 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, p. 199-208.

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

[54] D. BECKER, M. BABOULIN, J. DONGARRA. *Reducing the amount of pivoting in symmetric indefinite systems*, 2011, University of Tennessee Technical Report ICL-UT-11-06 and Inria Research Report 7621, to appear in the proceedings of 9th International Conference on Parallel Processing and Applied Mathematics, September 2011.

[55] 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.

[56] 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.

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

[58] 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.

[59] 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.

[60] 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.

[61] 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, p. 430–437.

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

[63] L. CHOY, S. G. PETITON, M. SATO. *Resolution of large symmetric eigenproblems on a world wide grid*, in "Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007)", Rio de Janeiro, Brazil, IEEE Computer Society, May 2007, p. 301-308.

[64] L. CHOY, S. G. PETITON, M. SATO. *Toward power-aware computing with dynamic voltage scaling for heterogeneous platforms*, in "Sixth International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (HeteroPar) in conjunction with the 2007 IEEE International Conference on Cluster Computing (Cluster07)", Austin, Texas USA, IEEE Computer Society Press, September 2007.

[65] 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, p. 1139–1160.

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

[67] 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.

[68] 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.

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

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

[71] 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, p. 929–961.

[72] 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, p. 1289-1314.

[73] 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.

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

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

[76] 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.

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

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

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

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

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

[82] S. G. PETITON, L. CHOY. *Eigenvalue Grid and Cluster Computations, Using Task Farming Computing Paradigm and Data Persistency*, in "SIAM conference on Computational Science & Engineering (CSE'07)", Costa Mesa, California, USA, February 2007.

[83] 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.

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

[85] 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, p. 329–350.

[86] 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, p. 675–686.

[87] 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.

[88] 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.

[89] 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, p. 149–160.

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

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

[92] 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.

[93] Y. ZHANG, G. BERGÈRE, S. G. PETITON. *A parallel hybrid method of GMRES on Grid System*, in "Workshop on High Performance Grid Computing (HPGC'07), jointly published with IPDPS'07 proceedings", Long Beach, California, USA, March 2007.

[94] 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.