INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team grand-large

Calcul parallèle et distribué à grande échelle

Futurs

THEME NUM

Table of contents

1. Team

Head of the project-team

Franck Cappello [Research Director, INRIA-Futurs, HdR]

Administrative assistant

Gina Grisvard [Administrative assistant INRIA-Futurs]
Katia Evrat [Administrative assistant INRIA-Futurs]

Research scientist

Joffroy Beauquier [Professor at Paris-Sud University, HdR]
Franck Cappello [Research Director at INRIA-Futurs, HdR]
Gilles Fedak [Junior Researcher at INRIA-Futurs]
Laura Grigori [Junior Researcher CR1 at INRIA-Futurs]
Thomas Hérault [Assistant Professor at Paris Sud University]
Serge Petiton [Professor at University of Science and Technology of Lille, HdR]
Sylvain Peyronnet [Assistant Professor at Paris Sud University]
Brigitte Rozoy [Professor at Paris-Sud University, HdR]
Sébastien Tixeuil [Professor at Paris 6 University, HdR]

Post-doctoral fellows and non permanent position

Alok Gupta [PostDoc INRIA]
Haiwu He [PostDoc INRIA]
Mathieu Jan [PostDoc INRIA]
Derrick Kondo [PostDoc INRIA]
Pierre Lemarinier [PostDoc INRIA]
Partha Sarathi Mandal [PostDoc INRIA]
Olivier Soyez [Teaching Assistant at Lille 1 University]
Hua Xiang [PostDoc INRIA]

Ph. D. student

Ali Asim [MESR Grant LRI]
Fatiha Bouabache [MESR Grant (LRI)]
Matthieu Cargnelli [EADS Industrial Grant (CIFRE)]
Laurent Choy [INRIA et Région Nord (LIFL)]
Camille Coti [INRIA Grant]
Philippe Gauron [MESR Grant (LRI)]
Toussaint Guglielmi [MESR Grant (LIFL)]
William Hoarau [MESR Grant (LRI)]
Benoit Hudzia [Franco-Irish Grant (LIFL)]
Yongjie Hu [Franco-Chine (LRI)]
Pawan Kumar [Cordis Grant]
David Ilcinkas [MESR Grant (LRI)]
Thomas Largillier [MESR Grant (LRI)]
Paul Malecot [INRIA Grant]
Nicolas Nisse [MESR Grant (LRI)]
Olivier Peres [MESR Grant (LRI)]
Niu Qiang [China Grant for 1 year visit]
Benjamin Quetier [MESR Grant (LRI)]
Ala Rezmerita [MESR Grant (LRI)]
Ye Zhang
Ling Shang [French Ministry of Foreign Affairs]

Research scientist (external collaborator)

Pierre Fraigniaud [Research Director at CNRS, HdR]

Rosaz Laurent [Assistant Professor at Paris Sud University]

Project technical staff

Víctor Iniesta [INRIA Expert Engineer]
Julien Leduc [INRIA Exper Engineer]
Phillipe Marty [INRIA Expert Engineer]
Vincent Neri [CNRS Study Engineer]
Pierre Neyron [INRIA Expert Engineer]
Eric Rodriguez [INRIA Associate Engineer]

Student interns

Maxime Hugues [Master Student]
Pierre-Yves Aquilanti [Master Student]
Zifan Liu [Internship, Polytech'Lille]

2. Overall Objectives

2.1. Grand-Large General Objectives

Grand-Large is a research project investigating the issues raised by computing on Large Scale Distributed Systems (LSDS), where users execute different 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 on Desktop Grids and Grids, while some results also concern large scale parallel computers. Our research focuses on middleware and programming environments design, development, proof and experiments. Fundamentally, we address the issues related to computing on LSDS, gathering several methodological tools that raise themselves scientific issues: theoretical models and exploration tools (observers, simulators, emulators and real size experimental systems).

The project aims at:

1. studying experimentally, and formally, the fundamental mechanisms of LSDS for high performance computing;
2. designing, implementing, validating and testing real software, middleware and platform;
3. defining, evaluating and experimenting approaches for programming applications on these platforms.

Compared to other European and French projects, we gather skills in large scale systems formal design and validation of algorithms and protocols for distributed systems and 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 are deeply involved in the Grid5000 project (as project Director) and the future ALADDIN initiative (project scientific director).

3. Scientific Foundations

3.1. Large Scale Distributed Systems (LSDS)

What makes a fundamental difference between pioneer Global Computing systems such as Seti@home, Distributed.net and other early systems dedicated to RSA key cracking 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 Plaforms (also called Desktop Grids), a non predictable number of nodes may leave the system at any time. Some researches even consider that they may quit the system without any prior mention and reconnect the system in the same way. Another example of characteristics is the complete lack of control of nodes connectivity. 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. Systems 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 tread-offs. 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.

LSDS as a class of Grid systems, essentially extends the notion of computing beyond the frontier of administration domains. The very first paper discussing this type of systems [102] 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 [88] , Glunix [80] and Mosix [55], to cite a few. A first approach to cross administration domains was proposed by Web Computing projects such as Jet [92], Charlotte [56], Javeline [72], Bayanihan [98], SuperWeb [52], ParaWeb [62] and PopCorn [66]. 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 COSM [65], BOINC [54] and XtremWeb [76]. 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. Companies like United Devices [107], Platform [93], Grid systems [112] and Datasynapse [111] propose LSDS middleware for Desktop Grids (also known as PC Grid systems). 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.

LSDS systems (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 [77], 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.

3.1.2. Building a Large Scale Distributed System for Computing

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 [99], Gnutella [99] and Kazaa [87], 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 [95], CHORD [104], PASTRY [97] and TAPESTRY [110] 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 [74]. 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. Data storage and movement

Application data movements and storage are major issues of LSDS since a large class of computing applications requires the access of large data sets as input parameters, intermediary results or output results.

In this scenario of data-centric applications, the existing Desktop Grid Systems face a scalability issue. One should expect that more computing resources also provides more network bandwidth and storage capacity. On the contrary, Desktop Grids Systems like BOINC or XtremWeb rely on a centralized data service architecture. For instance, data distribution with BOINC relies on multiple http servers and tasks are described as a list of files locations, which can be a potential bottleneck when scheduling tasks sharing large input files. Recent developments in content distribution such as collaborative file distribution (BitTorrent [73], Slurpie [101], Digital Fountain [64]) and P2P file sharing applications (EDonkey/Overnet [75], Kazaa [87]), has proven to be both effective, viable and *self-scalable*. The key idea, often featured as *parallel download* in the P2P applications, is to divide the file in a list of chunks. Immediately after a peer downloads a chunk from another peer, the peer serves the block for the other peers in the network, thus behaving itself as a server. Collaborative Content Distribution is a very active research topic and several promising strategies [94] such as the use of network coding [81], are proposed to improve the performance of the system. We will evaluate the efficiency of collaborative data distribution mechanism according to its impact on the overall performance of a parallel application when scheduled on Desktop Grid.

Currently there is no LSDS system dedicated to computing that allows the persistent storage of data in the participating nodes. Several LSDS systems dedicated to data storage are emerging such as OCEAN Store [84] and Ocean [71]. Storing large data sets on volatile nodes requires replication techniques. In CAN and Freenet, the documents are stored in a single piece. In OceanStore, Fastrack and eDonkey, the participants store segments of documents. This allows segment replications and the simultaneous transfer of several documents segments. In the CGP2P project, a storage system called US has been proposed. It relies on the notion of blocs (well known in hard disc drivers). Redundancy techniques complement the mechanisms and provide raid like properties for fault tolerance. Moreover, we believe that the basic blocks for building a system dedicated to data management in Desktop Grids can be found in P2P systems. We will investigate new paradigm for LSDS data management based on P2P components such as Bittorrent for data distribution and DHT for efficient and scalable data index/search, and design a new middleware, BitDew, featuring data transfer reliability, volatility tolerance, automatic data replication, multi-protocol capability, data affinity, and tuple-space like semantic.

3.1.2.3. Scheduling in large scale systems

Scheduling is one of the system fundamental mechanisms. Several studies have been conducted in the context of Grid mostly considering bag of tasks, parameter sweep or workflow applications [69] , [67]. Recently some researches consider scheduling and migrating MPI applications on Grid [103]. Other related researches concern scheduling for cycle stealing environments [96]. Some of these studies consider not only the dynamic CPU workload but also the network occupation and performance as basis for scheduling decisions. They often refer to NWS which is a fundamental component for discovering the dynamic parameters of a Grid. There are very few researches in the context of LSDS and no existing practical ways to measure the workload dynamics of each component of the system (NWS is not scalable).

While LSDS systems consist of volatile and heterogeneous computing resources, it is unknown exactly how volatile and heterogeneous these computing resources are. While there have been previous characterization studies on Internet-wide computing resources, these studies do not take into account causes of volatility and only report coarse aggregate statistics, such as the mean time to failure of resources. Yet, detailed resource

characterization is essential for the simulation and modelling of LSDS systems in a research area where many results are obtained via simulation.

We propose to design, implement, and deploy a resource monitoring project called XtremLab via the BOINC software system. The goal of XtremLab will be to monitor the availability of a large fraction of the LSDS participants in an effort to paint a more detailed picture of the Internet computing landscape.

Based, on these availability traces, we will design new simulator of Global Computing system (SIMBOINC) and investigate advanced scheduling heuristics for bag of task applications with large input data sets, application with soft real time constraints, workflow and data flow applications, long running applications which needs periodic checkpoints etc ...

3.1.2.4. 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 [90], 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 [70]. 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 [108], pessimistic [53] , causal [109]. 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 [59], MPICH-V2 [60] , MPICH-Vcausal , MPICH-Vcl [61], 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 developpers 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 ([78]). 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 DECRYPTON 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. [89] [105] [74] 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. [86] [85] , [57], [58] , [79]) 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 virtualisation 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 [106]. 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 [100] , [83]. 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 the beginning of the evaluations, we experimented with those parallel methods on a few dozen peer XtremWeb P2P Platform. 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 [68] GridSim [63] , Briks [51]. 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 [91] 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 developped, 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 inuse 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 Grid 5000 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 Grid 5000, 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

Numerical simulations are important for obtaining progress in many fields, when for example the problem is too complicated for solving it only theoretically, or too expensive to prototype in the laboratory. These simulations frequently lead to solving very large sets of linear equations and large least squares problems, often with millions of rows and columns. Solving these problems is very time consuming, and the focus of our research is the design of faster algorithms. The demand for better algorithms comes from two trends. First, the problems sizes are growing as the demand for high resolution grows, and as computers get faster. Second, though computers are getting faster, their technology is developing in ways that make current algorithms less and less efficient, and so unable to solve these large problems effectively. The two challenging technology trends are massive parallelism and increased communication cost. Massive parallelism means that every computer, from our laptops to the largest supercomputers, will depend on parallel processing, with the largest computers using many thousands of processors to attain their speed (Petaflop comuputers are expected before 2010). Increased communication costs means that the relative time to move data between processors is increasing *exponentially* relative to the time it to execute arithmetic operations, so conventional algorithms will spend more and more of their time communicating and less and less on computing. One of our research goals consists in developing new algorithms for solving linear systems and least squares problems, that greatly reduce the amount of time spent communicating, relative to conventional algorithms.

The algorithms can be based on direct methods or iterative methods. Direct methods (LU factorization for solving linear systems and QR factorization for solving least squares problems) are often preferred because of their robustness. The methods differ significantly depending on whether the matrices are dense (all nonzero entries) or sparse (very few nonzero entries, common in matrices arising from physical modeling). Iterative methods are less robust, but they are widely used because of their limited memory requirements and good scalability properties on sparse matrices. The iterative methods are mainly Krylov subspace iterations such as Conjugate Gradient for SPD matrices, GMRES or BICGSTAB for unsymmetric matrices. Preconditioning plays an important role in this context.

3.5.1. Efficient linear algebra algorithms

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 research.

3.5.2. Combinatorial tools for scientific computing

This direction of research covers aspects of scientific computing that can be expressed as combinatorial problems and solved using combinatorial tools and algorithms. There are many such examples in scientific

computing and an active community works on these subjects. Our research interests in this context consider the problem of structure prediction when solving sparse linear systems of equations and graph partitioning problem.

4. Application Domains

4.1. Building a Large Scale Distributed System for Computing

The main application domain of the Large Scale Distributed System developed in Grand-Large is high performance computing. The two main programming models associated with our platform (RPC and MPI) allow to program a large variety of distributed/parallel algorithms following computational paradigms like bag of tasks, parameter sweep, workflow, dataflow, master worker, recursive exploration with RPC, and SPMD with MPI. The RPC programming model can be used to execute concurrently different applications codes, the same application code with different parameters and library function codes. In all these cases, there is no need to change the code. The code must only be compiled for the target execution environment. LSDS are particularly useful for users having large computational needs. They could typically be used in Research and Development departments of Pharmacology, Aerospace, Automotive, Electronics, Petroleum, Energy, Meteorology industries. LSDS can also be used for other purposes than CPU intensive applications. Other resources of the connected PCs can be used like their memory, disc space and networking capacities. A Large Scale Distributed System like XtremWeb can typically be used to harness and coordinated the usage of these resources. In that case XtremWeb deploys on Workers services dedicated to provide and manage a disc space and the network connection. The storage service can be used for large scale distributed fault tolerant storage and distributed storage of very large files. The networking service can be used for server tests in real life conditions (workers deployed on Internet are coordinated to stress a web server) and for networking infrastructure tests in real like conditions (workers of known characteristics are coordinated to stress the network infrastructure between them).

4.2. Security and Reliability of Network Control Protocols

The main application domain for self-stabilizing and secure algorithms is LSDS where correct behaviours must be recovered within finite time. Typically, in a LSDS (such as a high performance computing system), a protocol is used to control the system, submit requests, retrieve results, and ensure that calculus is carried out accordingly to its specification. Yet, since the scale of the system is large, it is likely that nodes fail while the application is executing. While nodes that actually perform the calculus can fail unpredictably, a self-stabilizing and secure control protocol ensures that a user submitting a request will obtain the corresponding result within (presumably small) finite time. Examples of LSDS where self-stabilizing and secure algorithms are used, include global computing platforms, or peer to peer file sharing systems. Another application domain is routing protocols, which are used to carry out information between nodes that are not directly connected. Routing should be understood here in its most general acceptance, e.g. at the network level (Internet routing) or at the application level (on virtual topologies that are built on top of regular topologies in peer to peer systems). Since the topology (actual or virtual) evolves quickly through time, self-stabilization ensures that the routing protocol eventually provides accurate information. However, for the protocol to be useful, it is necessary that it provides extra guarantees either on the stabilization time (to recover quickly from failures) or on the routing time of messages sent when many faults occur. Finally, additional applications can be found in distributed systems that are composed of many autonomous agents that are able to communicate only to a limited set of nodes (due to geographical or power consumption constraints), and whose environment is evolving rapidly. Examples of such systems are wireless sensor networks (that are typically large of 10000+ nodes), mobile autonomous robots, etc. It is completely unrealistic to use centralized control on such networks because they are intrinsically distributed; still strong coordination is required to provide efficient use of resources (bandwidth, battery, etc.).

4.3. End-User Tools for Computational Science and Engineering

Another Grand Large application domain is Linear Algebra, which is often required to solve Large Scale Computational Science and Engineering applications. Two main approaches are proposed. First, we have to experiment and evaluate several classical stable numerical methods. Second, we have to propose tools to help end-users to develop such methods.

In addition to the classical supercomputing and the GRID computing, the large scale P2P approach proposes new computing facilities for computational scientists and engineers. Thus, it exists many applications which would use such computing facilities for long period of time . During a first period, many applications will be based on large simulations rather than classical implicit numerical methods, which are more difficult to adapt for such large problems and new programming paradigm as they generated linear algebra problems. Then, implicit method would be developed to have more accurate solutions.

Simulations and large implicit methods always have to compute linear algebra routines. So, they were our first targeted numerical methods (we also remark that the powerful worldwide computing facilities are still rated using a linear algebra benchmark http://www.top500.org). We especially focused on divide-and-conquer and block-based matrix methods to solve dense problems. We have also studied Krylov subspace methods (Lanczos, Arnoldi) and hybrid methods to solve sparse matrix problems. As these applications are utilized for many applications, it is possible to extrapolate the results to different scientific domains.

Many smart tools have to be developed to help the end-user to program such environments, using up-to-date component technologies and languages. At the actual present stage of maturity of this programming paradigm for scientific applications, the main goal is to experiment on large platforms, to evaluate and extrapolate performance, and to propose tools for the end-users; with respect to many parameters and under some specify hypothesis concerning scheduling strategies and multicast speeds [82]. We have to always replace the end-user at the center of this scientific programming. Then, we have to propose a framework to program P2P architectures which completely virtualizes the P2P middleware and the heterogeneous hardware. Our approach is based, on the one hand, on component programming and coordination languages, and on the other hand, to the development of an ASP, which may be dedicated to a targeted scientific domain. The YML framework provides a solution to the first point since it offers the YvetteML workflow language in order to orchestrate components. This is a very intuitive programming approach and it favors the re-usability of optimized and bug-free components. The abstraction of the underlying P2P middleware is also ensured by YML by means of its back-end mechanism. The end-user of YML can submit a computing task to anykind of peer connected to Internet as long as YML has a back-end in charge of the middleware which is running on this peer. Currently, YML has two back-ends for the XtremWeb and OmniRPC middleware. Another one for Condor will be soon available. The second point concerns the integration of SPIN to YML in order to get a complete programming tool which covers all the needs of the client in order to run applications (based on linear algebra methods) over the Internet. Finally, the conclusion of our work would be a P2P scientific programming methodology based on experimentations and evaluation on an actual P2P development environment.

5. Software

5.1. XtremWeb

XtremWeb is an open source middleware, generalizing global computing plarforms for a multi-user and multi-parallel programming context. XtremWeb relies on the notion of services to deploy a Desktop Grid based on a 3 tiers architecture. This architecture gathers tree main services: Clients, Coordinators and Workers. Clients submit requests to the coordinator which uses the worker resources to execute the corresponding tasks. XtremWeb decouples access to data from tasks, which allows integration with network file service.

Coordinator sub-services provide resource discovery, service construction, service instantiation and data repository for parameters and results. A major concern is fault tolerance. XtremWeb relies on passive replication and message logging to tolerate Clients mobility, Coordinator transient and definitive crashes and Worker volatility. An extention of XtremWeb called RPC-V features failure-tolerance of the coordinator. The Client service provides a Java API which unifies the interactions between the applications and the Coordinator. Three client applications are available: the Java API that can be used in any Java applications, a command line (shell like) interface and a web interface allowing users to easily submit requests, consult status of their tasks and retrieve results. A second major issue is the security. The origins of the treats are the applications, the infrastructure, the data (parameters and results) and the participating nodes. Currently XtremWeb provides user right managment, application sandboxing and communication encryption. XtremWeb provides a RPC interface for bag of tasks, parameter sweep, master worker and workflow applications.

XtremWeb has been tested extensively harnessing a thousand of Workers and computing a million of tasks. XtremWeb is deployed in several sites: LIFL, LIP, ID-IMAG, LRI, LAL (physique Orsay), IBBMC (biologie Orsay), Université de Guadeloupe, IFP (Institut Français du Petrol), EADS, CEA, Université du Wisconsin Madison, Université de Tsukuba (Japon), AIST (Australie), UCSD (USA). A Nederlands enterprise (AlmereGrid) maintain a Desktop Grid based on XtremWeb in the town of Almere. University of Paris Sud and University of California San Diego have used XtremWeb to gather hosts availability traces. To our knowledge, several applications has been ported to XtremWeb: Aires belonging to the HEP Auger project (LAL), a protein conformation predictor using a molecular dynamic simulator (IBBMC), CFD code (IFP), , PHYLIP Phylogenetic Package and Hand Writing recognition Application (Univ Tunisie), Application Based Iterative Methods (Hohai University, China). Several XtremWeb extention have been developped by research teams : XtremWeb-CH allows direct P2P communication between workers, YvetteML/XtremWeb allows XtremWeb to be used with a workflow frontend.

5.2. BitDew

The BitDew framework is a programmable environment for data management and distribution on computational Desktop Grids.

BitDew is a subsystem which could be easily integrated into other Desktop Grid systems (XtremWeb, BOINC, Condor etc..). Currently, Desktop Grids are mostly limited to embarrassingly parallel applications with few data dependencies. BitDew objective is to broaden the use of Desktop Grids. The BitDew framework will enable the support for data-intense parameter sweep applications, long-running applications which requires distributed checkpoint services, workflow applications and maybe in the future soft-realtime and stream processing applications.

BitDew offers programmers a simple API for creating, accessing, storing and moving data with ease, even on highly dynamic and volatile environments.

The BitDew programming model relies on 5 abstractions to manage the data : i) replication indicates how many occurrences of a data should be available at the same time on the network, ii) fault-tolerance controls the policy in presence of machine crash, iii) lifetime is an attribute absolute or relative to the existence of other data, which decides the life cycle of a data in the system, iv) placement drives movement of data according to dependency rules, v) distribution gives the runtime environment hints about the protocol to distribute the data. Programmers define for every data these simple criteria, and let the BitDew runtime environment manage operations of data creation, deletion, movement, replication, and fault-tolerance operation.

5.3. SimBOINC

SimBOINC is a simulator for heterogeneous and volatile desktop grids and volunteer computing systems. The goal of this project is to provide a simulator by which to test new scheduling strategies in BOINC, and other desktop and volunteer systems, in general. SimBOINC is based on the SimGrid simulation toolkit for simulating distributed and parallel systems, and uses SimGrid to simulate BOINC (in particular, the client CPU scheduler, and eventually the work fetch policy) by implementing a number of required functionalities.

SimBOINC simulates a client-server platform where multiple clients request work from a central server. In particular, we have implemented a client class that is based on the BOINC client, and uses (almost exactly) the client's CPU scheduler source code. The characteristics of client (for example, speed, project resource shares, and availability), of the workload (for example, the projects, the size of each task, and checkpoint frequency), and of the network connecting the client and server (for example, bandwidth and latency) can all be specified as simulation inputs. With those inputs, the simulator will execute and produce an output file that gives the values for a number of scheduler performance metrics, such as effective resource shares, and task deadline misses.

The software is available at http://simboinc.gforge.inria.fr

5.4. XtremLab

Since the late 1990's, DG systems, such as SETI@Home, have been the largest and most powerful distributed computing systems in the world, offering an abundance of computing power at a fraction of the cost of dedicated, custom-built supercomputers. Many applications from a wide range of scientific domains – including computational biology, climate prediction, particle physics, and astronomy – have utilized the computing power offered by DG systems. DG systems have allowed these applications to execute at a huge scale, often resulting in major scientific discoveries that would otherwise had not been possible.

The computing resources that power DG systems are shared with the owners of the machines. Because the resources are volunteered, utmost care is taken to ensure that the DG tasks do not obstruct the activities of each machine's owner; a DG task is suspended or terminated whenever the machine is in use by another person. As a result, DG resources are volatile in the sense that any number of factors can cause the task of a DG application to not complete. These factors include mouse or keyboard activity, the execution of other user applications, machine reboots, or hardware failures. Moreover, DG resources are heterogeneous in the sense that they differ in operating systems, CPU speeds, network bandwidth, memory and disk sizes. Consequently, the design of systems and applications that utilize these systems is challenging.

The long-term overall goal of XtremLab is to create a testbed for networking and distributed computing research. This testbed will allow for computing experiments at unprecedented scale (i.e., thousands of nodes or more) and accuracy (i.e., nodes that are at the "ends" of the Internet).

Currently, the short-term goal of XtremLab is to determine a more detailed picture of the Internet computing landscape by measuring the network and CPU availability of many machines. While DG systems consist of volatile and heterogeneous computing resources, it is unknown exactly how volatile and heterogeneous these computing resources are. Previous characterization studies on Internet-wide computing resources have not taken into account causes of volatility such as mouse and keyboard activity, other user applications, and machine reboots. Moreover, these studies often only report coarse aggregate statistics, such as the mean time to failure of resources. Yet, detailed resource characterization is essential for determining the usefulness of DG systems for various types of applications. Also this characterization is a prerequisite for the simulation and modelling of DG systems in a research area where many results are obtained via simulation, which allow for controlled and repeatable experimentation.

For example, one direct application of the measurements is to create a better BOINC CPU scheduler, which is the software component responsible for distributing tasks of the application to BOINC clients. We plan to use our measurements to run trace-driven simulations of the BOINC CPU scheduler in effort to identify ways it can be improved, and for testing new CPU schedulers before they are widely deployed.

We conduct availability measurements by submitting real compute-bound tasks to the BOINC DG system. These tasks are executed only when the host is idle, as determined by the user's preferences and controlled the BOINC client. These tasks continuously perform computation and periodically record their computation rates to file. These files are collected and assembled to create a continuous time series of CPU availability for each participating host. Utmost care will be taken to ensure the privacy of participants. Our simple, active trace method allows us to measure exactly what actual compute power a real, compute-bound application would be able to exploit. Compared to other passive measurement techniques, our method is not as susceptible to OS

idiosyncracies (e.g. with process scheduling) and takes into account keyboard and mouse activity, and host load, all of which directly impact application execution.

The XtremLab project webpage is at http://xtremlab.lri.fr

5.5. MPICH-V

Currently, MPICH-V proposes 7 protocols: MPICH-V1, MPICH-V2, MPICH-Vcl, MPICH-Pcl and 3 algorithms for MPICH-Vcausal. MPICH-V1 implements an original fault tolerant protocol specifically developed for Desktop Grids relying on uncoordinated checkpoint and remote pessimistic message logging. It uses reliable nodes called Channel Memories to store all in transit messages. MPICH-V2 is designed for homogeneous networks like clusters where the number of reliable component assumed by MPICH-V1 is too high. It reduces the fault tolerance overhead and increases the tolerance to node volatility. This is achieved by implementing a new protocol splitting the message logging into message payload logging and event logging. These two elements are stored separately on the sender node for the message payload and on a reliable event logger for the message events. The third protocol, called MPICH-Vcl, is derived from the Chandy-Lamport global snapshot algorithm. It implements coordinated checkpoint without message logging. This protocol exhibits less overhead than MPICH-V2 for clusters with low fault frequencies. MPICH-Pcl is a blocking implementation of Chandy-Lamport algorithm. It consists in exchanging messages for emptying every communication channel before checkpointing all processes. MPICH-Vcausal concludes the set of message logging protocols, implementing a causal logging. It provides less synchrony than the pessimistic logging protocols, allowing messages to influence the system before the sender can be sure that non deterministic events are logged, to the cost of appending some information to every communication. This sum of information may increase with the time, and different causal protocols, with different cut techniques, have been studied with the MPICH-V project.

The protocols developped during the first phase of the MPICH-V project are now being integrated into the two main open-source distributions of MPI, namely MPICH2 and OpenMPI. During this integration, we focus on keeping the best performances (i.e. introducing the smallest changes in the library communication driver). Eventually, the fault-tolerance properties of these two distributions should be provided by the Grand-Large project.

In addition to fault tolerant properties, MPICH-V:

1. provides a full runtime environment detecting and re-launching MPI processes in case of faults;
2. works on high performance networks such as Myrinet, Infiniband, etc (the performances are still divided by two);
3. allows the migration of a full MPI execution from one cluster to another, even if they are using different high performance networks.

The software, papers and presentations are available at http://mpich-v.lri.fr/

5.6. YML

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 sematic 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 XtermWeb 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.

5.7. The Scientific Programming InterNet (SPIN)

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), developped by the Nahid Emad team at the University of Versailles, which would be an alternative to SPIN as a component oriented integration with YML.

5.8. V-DS

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 virtualisation 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 virtualisation 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 virtualisation 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 patform 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 marchine). 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 [49].

5.9. PVC: Private Virtual Cluster

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 belog 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. PCV 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.

5.10. OpenWP

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 (generaly 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 disributed. 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.

5.11. FAult Injection Language (FAIL)

FAIL (FAult Injection Language) is a new project that was started in 2004. The goal of this project is to provide a controllable fault injection layer in existing distributed applications (for clusters and grids). A new language was designed to implement expressive fault patterns, and a preliminary implementation of the distributed fault injector based on this language was developped.

5.12. Parallel solvers for solving linear systems of equations

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 [14]. The SuperLU_DIST is available at http://crd.lbl.gov/~xiaoye/SuperLU/ .

We continue the development in implementing the numerical factorization phase using the communication avoiding ideas developed this year.

6. New Results

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

6.1. Large Scale Peer to Peer Performance Evaluations

6.1.1. 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 [41]. We adapted [27] 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.

6.1.2. 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 [45] 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.

6.1.3. 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 [28]. 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.

6.2. Volatility and Reliability Processing

6.2.1. Dependability Benchmarking

[16] One important contribution to the community that is developing Grid middleware is the definition and implementation of benchmarks and tools to assess the performance and dependability of Grid applications and the corresponding middleware. In this paper, we present an experimental study that was conducted with OGSA-DAI, a popular package of middleware that provides access to remote data resources thought a unified Web-service front-end. The results show that OGSA-DAI is quite stable and performed quite well in scalability tests, executed on Grid5000. However, we also demonstrate that OGSA-DAI WSI is currently using a SOAP container (Apache Axis1.2.1) that suffers from severe memory leaks. We show how the default configuration of OGSA-DAI is not affected by that problem, but a small change in the configuration of a Web-service may lead to very unreliable execution of OGSA-DAI.

[17] One of the topics of paramount importance in the development of Grid middleware is the impact of faults since their probability of occurrence in a Grid infrastructure and in large-scale distributed system is actually very high. In this paper we explore the versatility of a new tool for fault injection in distributed applications: FAIL-FCI. In particular, we show that not only we are able to fault-load existing distributed applications (as used in most current papers that address fault-tolerance issues), we are also able to inject *qualitative faults*, *i.e.* inject specific faults at very specific moments in the program code of the application under test. Finally, and although this was not the primary purpose of the tool, we are also able to inject specific patterns of workload, in order to stress test the application under test. Interestingly enough, the whole process is driven by a simple unified description language, that is totally independent from the language of the application, so that no code changes or recompilation are needed on the application side.

6.2.2. Robust algorithms

[33] In self-organizing systems, such as mobile ad-hoc and peer-to-peer networks, consensus is a fundamental building block to solve agreement problems. It contributes to coordinate actions of nodes distributed in an *ad-hoc* manner in order to take consistent decisions. It is well known that in classical environments, in which entities behave asynchronously and where identities are known, consensus cannot be solved in the presence of even one process crash. It appears that self-organizing systems are even less favorable because the set and identity of participants are not known. We define necessary and sufficient conditions under which fault-tolerant consensus become solvable in these environments. Those conditions are related to the synchrony requirements of the environment, as well as the connectivity of the knowledge graph constructed by the nodes in order to communicate with their peers.

[46] Properly locating sensor nodes is an important building block for a large subset of wireless sensor networks (WSN) applications. As a result, the performance of the WSN degrades significantly when misbehaving nodes report false location and distance information in order to fake their actual location. In this paper we propose a general distributed deterministic protocol for accurate identification of faking sensors in a WSN. Our scheme does *not* rely on a subset of *trusted* nodes that are not allowed to misbehave and are known to every node in the network. Thus, any subset of nodes is allowed to try faking its position. As in previous approaches, our protocol is based on distance evaluation techniques developed for WSN. On the positive side, we show that when the received signal strength (RSS) technique is used, our protocol handles at most $\lfloor \frac{n}{2} \rfloor - 2$ faking sensors. Also, when the time of flight (ToF) technique is used, our protocol manages at most $\lfloor \frac{n}{2} \rfloor - 3$ misbehaving sensors. On the negative side, we prove that no deterministic protocol can identify faking sensors if their number is $\lceil \frac{n}{2} \rceil - 1$. Thus our scheme is almost optimal with respect to the number of faking sensors. We discuss application of our technique in the trusted sensor model. More precisely our results can be used to minimize the number of trusted sensors that are needed to defeat faking ones.

[11] In this paper we present failure detectors that detect transient failures, i.e. corruption of the system state without corrupting the program of the processors. We distinguish *task* which is the problem to solve, from *implementation* which is the algorithm that solve the problem. A task is specified as a desired output of the distributed system. The mechanism used to produce this output is not a concern of the task but a concern of the implementation. In addition we are able to classify both the *distance locality* and the *history locality* property of tasks. The distance locality is related to the diameter of the system configuration that a failure detector has to maintain in order to detect a transient fault. The history locality is related to the number of consecutive system configurations that a failure detector has to maintain in order to detect a transient fault. Both the distance and history locality of a task may give the implementation designer hints concerning the techniques and resources that are required for implementing the task.

6.2.3. Self-stabilizing Algorithms

[30] In this paper, we specify the *conflict manager* abstraction. Informally, a conflict manager guarantees that any two neighboring nodes can not enter their critical simultaneously (*safety*), and that at least one node is able to execute its critical section (*progress*). The conflict manager problem is strictly weaker than the classical local mutual exclusion problem, where any node that requests to enter its critical section eventually does so (*fairness*). We argue that conflict managers are a useful mechanism to transform a large class of self-stabilizing

algorithms that operate in an essentially sequential model, into self-stabilizing algorithm that operate in a completely asynchronous distributed model. We provide two implementations (one deterministic and one probabilistic) of our abstraction, and provide a composition mechanism to obtain a generic transformer. Our transformers have low overhead: the deterministic transformer requires one memory bit, and guarantees time overhead in order of the network degree, the probabilistic transformer does not require extra memory. While the probabilistic algorithm performs in anonymous networks, it only provides probabilistic stabilization guarantees. In contrast, the deterministic transformer requires initial symmetry breaking but preserves the original algorithm guarantees.

[40] The maximal matching problem has received considerable attention in the self-stabilizing community. Previous work has given different self-stabilizing algorithms that solves the problem for both the adversarial and fair distributed daemon, the sequential adversarial daemon, as well as the synchronous daemon. In the following we present a single self-stabilizing algorithm for this problem that unites all of these algorithms in that it stabilizes in the same number of moves as the previous best algorithms for the sequential adversarial, the distributed fair, and the synchronous daemon. In addition, the algorithm improves the previous best moves complexities for the distributed adversarial daemon from $O(n^2)$ and $O(\delta m)$ to $O(m)$ where n is the number of processes, m is the number of edges, and δ is the maximum degree in the graph.

[24] Self-stabilization is a versatile technique to withstand any transient fault in a distributed system. Mobile robots (or agents) are one of the emerging trends in distributed computing as they mimic autonomous biologic entities. The contribution of this paper is threefold. First, we present a new model for studying mobile entities in networks subject to transient faults. Our model differs from the classical robot model because robots have constraints about the paths they are allowed to follow, and from the classical agent model because the number of agents remains fixed throughout the execution of the protocol. Second, in this model, we study the possibility of designing self-stabilizing algorithms when those algorithms are run by mobile robots (or agents) evolving on a graph. We concentrate on the core building blocks of robot and agents problems: naming and leader election. Not surprisingly, when no constraints are given on the network graph topology and local execution model, both problems are impossible to solve. Finally, using minimal hypothesis with respect to impossibility results, we provide deterministic and probabilistic solutions to both problems, and show equivalence of these problems by an algorithmic reduction mechanism.

6.3. Virtualization environment for large-scale Distributed Systems

V-DS has been used to study the behavior of three P2P middleware at large scales, up to 10000 peers. We mainly focus on the performance of building and maintaining P2P overlays based on: 1) Chimera which implements a Tapestry-like routing, 2) Khashmir which implements a Kademlia-like routing and 3) i3/Chord which implements the Chord routing algorithm. Notably, our results show that at large-scale only the i3/Chord is able to optimally build overlays, whatever the size of the overlay is. i3/Chord can therefore offer best performance to users for lookups (in $O(log\ n)$), while other implementations may either introduce delays or even failures in lookups.

6.4. Characterizing Intranet and Internet Desktop Grids

Desktop grids, which use the idle cycles of many desktop PC's, are currently the largest distributed systems in the world. Despite the popularity and success of many desktop grid projects, the heterogeneity and volatility of hosts within desktop grids has been poorly understood. Yet, host characterization is essential for accurate simulation and modelling of such platforms. We gathered application-level traces of four real desktop grids that can be used for simulation and modelling purposes. In addition, we determined aggregate and per host statistics that reflect the heterogeneity and volatility of desktop grid resources [18].

While desktop grids offer a high return on investment, one critical issue is the validation of results returned by participating hosts. Several mechanisms for result validation have been previously proposed. However, the characterization of errors is poorly understood. To study error rates, [38] we implemented and deployed a desktop grid application across several thousand hosts distributed over the Internet. We then analyzed the

results to give quan- titative and empirical characterization of errors stemming from input or output (I/O) failures. We find that in practice, error rates are widespread across hosts but occur relatively infrequently. Moreover, we find that er- ror rates tend to not be stationary over time nor correlated between hosts. In light of these characterization results, we evaluated state-of- the-art error detection mechanisms and describe the trade-offs for using each mechanism.

6.5. Data Management on Volatile and Heterogeneous Resources

Data-centric applications are still a challenging issue for Large Scale Distributed Computing Systems. The emergence of new protocols and softwares for collaborative content distribution over Internet offers a new opportunity for efficient and fast delivery of high volume of data. In [20] we present an evaluation of the BitTorrent protocol for Computational Desktop Grids. We first present a prototype of a generic subsystem dedicated to data management and designed to serve as a building block for any Desktop Grid System. Based on this prototype we conduct experimentations to evaluate the potential of BitTorrent compared to a classical approach based on FTP data server. The preliminary results obtained with a 65-nodes cluster measure the basic characteristics of BitTorrent in terms of latency and bandwidth and evaluate the scalability of BitTorrent for the delivery of large input files. Moreover, we show that BitTorrent has a considerable latency overhead compared to FTP but clearly outperforms FTP when distributing large files or files to a high number of nodes. Tests on a synthetic application show that BitTorrent significantly increases the computation/communication ratio of the applications eligible to run on a Desktop Grid System.

In [25], we present a A Distributed and Replicated Service for Checkpoint Storage.

6.6. Peer to Peer Resource Discovery

A new model for self-stabilizing algorithm design and evaluation has been proposed in. As an illustration of this new model, an algorithm to build a self-stabilizing spanning tree over an IP-based P2P network has been proposed and evaluated. Thus, this contribution provides a first overlay network on which we can build Resource Discovery P2P services. Other topologies are now studied, and we focus our work on the probabilistic evaluation of the theoretical and practical performances.

6.7. Fault Tolerant MPI

6.7.1. Grid-Enabling OpenMPI

As a first contribution to the OpenMPI library and runtime environment, we have proposed a set of efficient Grid Services for inter-cluster communication and integrated these services in the OpenMPI library [29]. This first step towards a fully grid-enabled, fault-tolerant MPI for the Grid is part of our work within the QosCosGrid European project. The first measurements demonstrated a startup overhead orders of magnitude lower than the state-of-the-art MPICH-G2, and efficient and transparent brokering services for the inter-cluster communication. Many relaying techniques, developped for the RPC-V project have been integrated in these services.

6.7.2. New Fault Tolerant Protocols for MPICH-V

Our collaboration with the MPICH team at ANL also have produced a new Fault-Tolerant protocol for the last MPICH driver, called Nemesis. This work is currently under review for publication in the FGCS journal. The new protocol proposed is an optimisation of the implementation presented during the SC'06 conference. We compare the efficiency of complex implementations of coordinated checkpointing with traditional implementations using blocking synchronizations to flush the communication channels before taking the checkpoint images. Surprisingly, the simple approach seems to demonstrate better performances when dealing with reasonable sized clusters.

6.7.3. Fault Tolerant Runtime Environments

Within our collaboration with the OpenMPI team at UTK, we are focusing our work on fault tolerant runtime environments. This work is still in its preliminary stage, but major contributions on the topology used to connect the runtime environment components with each other are expected. The topology presented are expected to exhibit a better scalability of the runtime environment deployement (which is a bottleneck in the current startup time of OpenMPI), and a better resiliance to failures.

6.8. High performance scientific computing

The focus of this research is on the design of faster algorithms for 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.

6.8.1. Communication avoiding algorithms for LU and QR factorizations

The first results in this project were obtained in the context of the parallel QR factorization of a tall skinny matrix $m \times b$. This mathematical operation is a bottleneck, for example, in a variety of material science simulation codes like the 2006 Gordon Bell Prize winning QBox code of Francois Gygi et al. A new algorithm was developed that substantially decreases the number of messages exchanged during the factorization. If we use one of the standard algorithms as implemented in ScaLAPACK, then the number of messages is $O(b \log_2 p)$. The new approach for performing this factorization leads to $O(\log_2 p)$ number of necessary messages, at the cost of some redundant computation. This is obtained by a formulation in which the QR decomposition is reorganized into a tree-like computation. The reduction in the number of messages is in particular interesting since the number of messages is independent of b, the horizontal dimension of the input matrix. The developed code is up to one order of magnitude faster than the corresponding ScaLAPACK code.

The same techniques are applied to the QR and LU factorization with partial pivoting of a dense matrix. The new algorithms overcome the latency bottleneck of the classic factorizations, as implemented in ScaLAPACK. Our goal is also to extend these techniques to sparse matrices.

The papers describing these algorithms are in preparation [35]. This research is in collaboration with James Demmel and Mark Hoemmen from U.C. Berkeley, USA, and Julien Langou from University of Denver, Colorado, USA.

6.8.2. Preconditioning large systems of equations

The incomplete version of such factorization can be used to construct very efficient preconditioners for iterative methods. In the future, we will apply the incomplete factorization in preconditioning. Currently, for the saddle point problem arising from PDE or optimization, we designed preconditioners based on Kronecker product approximation or Schilder factorization to accelerate the convergence. (H. Xiang, L. Grigori, Kronecker product approximation preconditioner for convection-diffusion model problems, manuscript [44]; Q. Niu, L. Grigori, H. Xiang, A class of constraint type preconditioners for regularized saddle point problems, manuscript). To understand the perturbation sensitivity, we investigate the condition number of the problem, such as Sylvester equation and Lyapunov equation and we also examine the backward error of an algorithm [21].

6.8.3. Combinatorial tools for scientific computing

This direction of research covers aspects of scientific computing that can be expressed as combinatorial problems and solved using combinatorial tools and algorithms. There are many such examples in scientific computing. In this context, we have adressed and solved several open questions in the structure prediction problem for solving sparse systems of equations [13].

We have also worked on another aspect that refers to the graph partitioning algorithms used as front-end ordering preceding the factorization. This is a collaboration with Guy Atenekeng Kahou (IRISA, Rennes) and Masha Sosonkina (Ames Laboratory, USA), that aims at partitioning a matrix into a block-diagonal form, such that any two consecutive blocks overlap. The partitioned matrix is suitable for applying the explicit formulation of the Multiplicative Schwarz Preconditioner developed by Atenekeng, Kamgnia and Philippe. The paper describing this algorithm is submitted to Parallel Computing.

7. Other Grants and Activities

7.1. Regional, National and International Actions

- PECO-NEI RFR with Eastern Europe Countries, 2006 - 2009, PI L. Grigori
- INRIA Associated Team "F-J Grid" with University of Tsukuba, head: Franck Cappello
- ACI "GRID'5000", 3 years (2003-2007), head: Franck Cappello.
- CIFRE EADS, 3 years (2005-2008), head: Franck Cappello.
- INRIA funding, MPI-V, collaboration with UTK, LALN and ANL, head: Franck Cappello
- Regional Council "Grid eXplorer", 3 years (2006-2009), co-chair: Franck Cappello
- ACI Sécurité FRAGILE, 3 years (2004-2007), head: S. Tixeuil
- ACI Sécurité SR2I, 3 years (2004-2007), subproject chair: S. Tixeuil
- ANR Jeunes chercheurs XtremLab : G. Fedak, 3 years (2005-2008)
- ANR CIS Project FF2A3, 3 years (2007 - 2010), PI F. Hecht, subproject head L. Grigori
- AURORA project France-Norway 1 year, "Self-stabilization and Sensor Networks", chair: S. Tixeuil
- European CoreGrid Network of Excellence, 4 years (2004 -2008), subtask head: S. Tixeuil, subproject heads: G. Fedak, T. Herault, S. Tixeuil
- European STREP (6th FP pri5-IST), 3 years (2006-2008). Quasi-Opportunistic Supercomputing for Complex Systems in Grid Environments (QosCosGrid), managment board representative: Franck Cappello, task head: Thomas Herault
- European project. Grid4All, 3 years (2006-2008), managment board representative: Franck Cappello, task head: Gilles Fedak
- NEGST Project (CNRS-JST), 3 years (2006-2008), http://www2.lifl.fr/MAP/negst/ , head(french side): Serge Petiton
- CARRIOCAS, Pole de Competitivité System@tic, 3 years (2006-2009), http://www.carriocas.org/, Franck Cappello
- HipCal, ANR CIS, 3 years (2006-2009), http://hipcal.lri.fr/wakka.php?wiki=PagePrincipale, Franck Cappello

8. Dissemination

8.1. Services to the Scientific Community

8.1.1. Book/Journal edition

- Franck Cappello, Thomas Hérault, and Jack Dongarra, EuroPVM/MPI2007, Recent Advances in Parallel Virtual Machine and Message Passing Interface ", Lecture Notes in Computer Science, Springer , September 2007.

- Toshimitsu Masuzawa, Sébastien Tixeuil, editors. Stabilization, Safety, and Security in Distributed Systems. LNCS 4838, 2007.
- Toshimitsu Masuzawa, Sébastien Tixeuil, editors. Reliability, Availability, and Security. INRIA, 2007.

8.1.2. Conference Organisation

The Grand-Large team has organized with the support of INRIA the international conference EuroPVM/MPI'07 (14th European PVM/MPI Users' Group Meeting). This event has been a success from the scientific point of view (with 25 sessions of high quality presentations, 37 selected papers), as well as from the attendance point of view (120 participants, from all countries of the world, including more than 40 american researchers and industrials). The conference highights include 6 keynotes speaches (Tony Hey, first MPI proposal co-author, Satoshi Matsuoka, Rusty Lusk, Bernd Mohr, Al Geist and George Bosilca), 3 tutorials and 4 best papers. The conference was sponsored by 9 partners who provided a major part of the finances for the event. It was held in the center of Paris, at the Salons espace Saint Honore. Full details on the conference can be found on its web page http://pvmmpi07.lri.fr.

- Franck Cappello, Workshop Chair, IEEE/ACM CCGRID'2008
- Franck Cappello, Program vice-Chair, Mardi Gras Conference 2008,
- Sébastien Tixeuil, Program co-chair, Algotel 2008
- Sébastien Tixeuil, Program co-chair, Ninth International Symposium on Stabilization, Safety, and Security (SSS 2007),
- Sébastien Tixeuil, Program co-chair, First International Workshop on Reliability, Availability, and Security (WRAS 2007)
- Franck Cappello, Program co-chair, HotP2P'2007, Fourth International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P) in conjunction with the IEEE International Parallel & Distributed Processing Symposium, 2007
- Franck Cappello, General co-chair, Grid and Pervasive Computing, 2007
- Thomas Herault, Franck Cappello, Program co-chairs, EuroPVM/MPI, Paris, Sept, 2007, 2007
- Gilles Fedak, Publicity chair, 14th European PVM/MPI Users' Group Conference, Paris 2007

8.1.3. Editorial Committee membership

- Franck Cappello, Cluster Computing Journal, Springer, Netherlands
- Franck Cappello, Journal of Grid Computing, Springer Netherlands
- Franck Cappello, Journal of Grid and utility computing, Inderscience
- Franck Cappello, Scientific Programming Journal Special Issue on Grids and Worldwide Computing, IOS Press, Winter 2005
- Franck Cappello, "Technique et Science Informatiques", 2001-2005
- Sébastien Tixeuil, "Technique et Science Informatiques", 2005-
- P. Fraigniaud, Theory of Computing Systems (TOCS), Springer,
- P. Fraigniaud, Journal of Interconnection Networks (JOIN), World Scientific,

8.1.4. Steering Committee membership

- Franck Cappello, IEEE/ACM HPDC
- Franck Cappello, IEEE/ACM CCGRID
- P. Fraigniaud, International Symposium on Theoretical Aspects of Computer Science (STACS).
- P. Fraigniaud, ACM Symposium on Parallelism in Algorithms and Architectures (SPAA).

- P. Fraigniaud, International symposium on Distributed Computing (DISC).

8.1.5. Program Committee membership

- Franck Cappello, Mardi Gras 2008
- Franck Cappello, SIMUTools 2008
- Franck Cappello, HCW , 17th Heterogeneous Computing Workshop (HCW'08) 2008
- Franck Cappello, IPTPS 2008
- Franck Cappello, HotP2P 2008
- Franck Cappello, ACM HPDC 17th IEEE International conference on High Performance Distributed Computing, 2008
- Franck Cappello, IEEE/ACM SC 2008, 21th IEEE/ACM International Conference for High Performance Computing and Communications (SC07) -Austin Texas, 2008
- Franck Cappello, IEEE/ACM CCGRID 8th IEEE International Symposium on Cluster Computing and the Grid.Lyon, France , May 2008
- Sébastien Tixeuil, DISC 2008
- Sébastien Tixeuil, Algotel 2008, co-chair
- Sébastien Tixeuil, Sirocco 2008
- Gilles Fedak, Renpar 2008
- Gilles Fedak, Euromicro PDP 2008
- Franck Cappello, OPODIS'2007, 10th International Conference On Principles Of Distributed Systems, Guadeloupe, December 17-20th, 2007
- Franck Cappello, 2007 International Conference on Complex Open Distributed Systems (CODS'2007), China, 22-24 July 2007
- Franck Cappello, GridNets 2007 – Lyon, October 17-19
- Franck Cappello, Europar'2007 – Rennes, Aug 28-31, 2007
- Franck Cappello, Grid'2007 – The 8th IEEE International Conference on Grid Computing (Grid 2007), Austin, Texas, September 2007
- Franck Cappello, SC'2007 – 20th IEEE/ACM International Conference for High Performance Computing and Communications (SC07) - Reno, Nevada, Nov 12-16, 2007
- Franck Cappello, CCGRID'2007 – 7th IEEE International Symposium on Cluster Computing and the Grid. Rio de Janeiro - Brasil / May 14-17, 2007.
- Franck Cappello, HCW'2007 – 16th Heterogeneous Computing Workshop (HCW'07), Long Beach, California, March 25 2007, June 2007.
- Franck Cappello, HPDC'2007 – 16th IEEE International conference on High Performance Distributed Computing, June 2007.
- Franck Cappello, ICDCS'2007 – 27th IEEE International Conference on Distributed Computing Systems, July 2007
- Sébastien Tixeuil, Algotel 2007, April 2007.
- Sébastien Tixeuil, GPC 2007, June 2007.
- Sébastien Tixeuil, AlgoSensors 2007, July 2007.
- Sébastien Tixeuil, CODS 2007.
- Sébastien Tixeuil, CoreGRID Workshop on Grid Programming Model, Grid and P2P Systems Architecture, Grid Systems, Tools and Environments, 2007.

- Sébastien Tixeuil, SSS 2007, November 2007, co-Chair.
- Sébastien Tixeuil, OPODIS 2007
- Joffroy Beauquier, PODC 2007, August 2007.
- Gilles Fedak, GP2PC'2007, "Global and Peer to Peer Computing", in association with CCGRID'2007, Rio de Janeiro - Brasil / May 14-17, 2007.
- Gilles Fedak, International Workshop on Hot Topics in Mobile Peer-to-Peer Networks (HOT-MP2P) Xi'An, China, September 10-14, 2007
- Gilles Fedak, Workshop on Modeling, Simulation and Optimization of Peer-to-peer environments (MSOP2P) associé à EuroMicro PDP 2007
- Derrick Kondo, GP2PC'2007, "Global and Peer to Peer Computing", in association with CCGRID'2007, Rio de Janeiro - Brasil / May 14-17, 2007.

8.1.6. School and Workshop organization

- Franck Cappello and Gilles Fedak, General Co-chairs Workshop on Large-Scale and Volatile Desktop Grids (PCGrid 2007) in conjunction with IPDPS, Miami, USA, 2008
- Derrick Kondo, Program chair, PCGrid'2007, First Workshop on Large-Scale and Volatile Desktop Grids (PCGrid) in conjunction with the IEEE International Parallel & Distributed Processing Symposium, 2007
- Laura Grigori, Co-chair of Workshop on Combinatorial tools for parallel sparse matrix computations, September, 2007, Gdansk, Poland. Co-organized with Masha Sosonkina (Ames Laboratory). In conjunction with PPAM'07 Conference and publication of selected papers in LNCS proceedings.
- Serge Petiton, Organization (on the French side) of the French-Japanese PAAP workshop at RIKEN, Tokyo, November 1st and 2nd 2007 [1]
- Serge Petiton, Organization (on the French side) of the 2nd French-Japanese NEGST workshop at NII, Tokyo, May 28th-29th 2007 [2]
- Gilles Fedak, General chair , XtremWeb User Group Meeting, Tunisie 2007
- Gilles Fedak, Program chair, Workshop on Large-Scale and Volatile Desktop Grids (PCGrid 2007), in conjunction with IPDPS, Long Beach, USA, 2007
- Gilles Fedak, Co-chair Global and Peer-to-Peer Computing, in conjunction with CCGRID with M. Sato (Univ. Tsukuba), Rio de Janeiro, Brazil, 2007 .

8.2. Participation to Workshops, Seminars and Miscellaneous Invitations

8.2.1. Invited International Conference

- Franck Cappello "Towards an International "Computer Science Grid", KEYNOTE talk, IEEE/ACM CCGRID'2007, Rio, Brazil, 2007
- Franck Cappello "Towards a third generation of Desktop Grids", KEYNOTE talk, GCP'2007, Paris, 2007
- Franck Cappello "Towards an International "Computer Science Grid", KEYNOTE talk, Symposium on Grid, Delft, 2007
- Franck Cappello "Progress in Automatic Fault Tolerance MPI: the MPICH-V dinasty", KEYNOTE talk, IEEE WETICE, Paris, 2007
- Gilles Fedak "Towards Data-Intensive Applications on XtremWeb", XtremWeb Users Group Workshop, Hammamet, Tunisie Février 2007

8.2.2. Invited National Conference

- Gilles Fedak "DSLLab : Plate-forme d'expérimentation pour les systèmes distribués à large échelle sur Internet haut-débit", Colloque ANR JCJC Montpellier 2007, France

8.2.3. Schools, Workshops

- Laura Grigori, Tutorial at Ecole d'ete en Automatique, Faculte d'Automatique et Ordinateurs, Bucarest, Roumanie, May 21-25, 2007.
- Franck Cappello, 1st PAAP Workshop at RIKEN Tokyo, November 1st 2007, "OpenWP: a directive based Workflow language and execution framework"
- Serge Petiton, 1st PAAP Workshop at RIKEN Tokyo, November 1st 2007, Key note speech "Toward PetascaleProgramming and Computing, challenges and collaborations" and final speech
- Laurent Choy, 2nd NEGST Workshop at NII Tokyo, May 28th 2007, "Resolution of large symmetric eigenproblems on a world-wide grid"
- Serge Petiton, 2nd NEGST Workshop at NII Tokyo, May 29th 2007, "The NEGST project in France, Past, Present and Future"

8.2.4. Seminaries

- Gilles Fedak, Advance in the Grid4All Scheduling Service, Grid4all Consortium meeting, Barcelone, Juin 2007
- Laura Grigori, Seminar in the serie organized jointly by the Numerical Analysis Group at Rutherford Appleton Laboratory and the Numerical Analysis Group at Oxford, October 31 to November 2, 2007.
- Laura Grigori, Seminar at US France Young Scientists Symposium, organized by the French Embassy in US, October 2007.
- Laura Grigori, at Universite de Lille, Department of Mathematics, Equipe d'Analyse Numerique/ EDP, June 2007.
- Serge Petiton, University College Dublin, October 2007
- Serge Petiton, 21th Orap forum, June 7th juin,
- Serge Petiton, UCSD, "Global GRID Linear Algebra Progamming and Computing", February 2007

9. Bibliography

Major publications by the team in recent years

[1] R. BOLZE, F. CAPPELLO, E. CARON, M. DAYDÉ , F. DESPREZ, E. JEANNOT, Y. JÉGOU, S. LANTERI, J. LEDUC, N. MELAB, G. MORNET, R. NAMYST, P. PRIMET, B. 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", vol. 20, n^{o} 4, November 2006, p. 481-494.

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

[3] F. CAPPELLO, S. DJILALI, G. FEDAK, T. HERAULT, 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.

[4] T. HERAULT, 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.

[5] T. HERMAN, S. TIXEUIL. *A Distributed TDMA Slot Assignment Algorithm for Wireless Sensor Networks*, in "Proceedings of the First Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors'2004), Turku, Finland", Lecture Notes in Computer Science, n^{o} 3121, Springer-Verlag, July 2004, p. 45-58.

[6] W. HOARAU, S. TIXEUIL, F. VAUCHELLES. *FAIL-FCI: Versatile Fault-Injection*, in "Future Generation Computer Systems", vol. 23, n^{o} 7, 2007, p. 913-919.

[7] C. JOHNEN, L. O. ALIMA, A. K. DATTA, S. TIXEUIL. *Optimal Snap-stabilizing Neighborhood Synchronizer in Tree Networks*, in "Parallel Processing Letters", vol. 12, n^{o} 3 & 4, 2002, p. 327–340.

[8] T. MASUZAWA, S. TIXEUIL. *On Bootstrapping Topology Knowledge in Anonymous Networks*, in "Eighth International Symposium on Stabilization, Safety, and Security on Distributed Systems (SSS 2006), Dallas, Texas", A. K. DATTA, M. GRADINARIU (editors), Lecture Notes in Computer Science, Springer Verlag, November 2006, p. 454-468.

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

Year Publications

Books and Monographs

[10] F. CAPPELLO, T. HERAULT, J. DONGARRA (editors). *Recent Advances in Parallel Virtual Machine and Message Passing Interface, 14th European PVM/MPI User's Group Meeting*, LNCS, n^{o} 4757, Springer, Paris, France, October 2007.

Articles in refereed journals and book chapters

[11] J. BEAUQUIER, S. DELAËT, S. DOLEV, S. TIXEUIL. *Transient Fault Detectors*, in "Distributed Computing", 2007, to appear.

[12] F. CAPPELLO, G. FEDAK, T. MORLIER, O. LODYGENSKY. *Des systèmes client-serveur aux systèmes pair à pair*, in "Encyclopédie de l'informatique et des systèmes d'information", Vuibert, 2007.

[13] L. GRIGORI, M. COSNARD, E. G. NG. *On the Row Merge Tree for Sparse LU Factorization with Partial Pivoting*, in "BIT Numerical Mathematics Journal", vol. 47, n^{o} 1, 2007, p. 45-76.

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

[15] L. GRIGORI, X. S. LI. *Towards an Accurate Performance Modeling of Parallel Sparse Factorization*, in "Applicable Algebra in Engineering, Communication, and Computing Journal", vol. 18, n^{o} 3, 2007, p. 241-261.

[16] W. HOARAU, S. TIXEUIL, N. RODRIGUES, D. SOUSA, L. SILVA. *Benchmarking the OGSA-DAI middleware*, in "Integrated Research in GRID COmputing", Springer Verlag, 2007.

[17] W. HOARAU, S. TIXEUIL, F. VAUCHELLES. *FAIL-FCI: Versatile Fault-Injection*, in "Future Generation Computer Systems", 2007.

[18] D. KONDO, G. FEDAK, F. CAPPELLO, A. A. CHIEN, H. CASANOVA. *Resource Availability in Enterprise Desktop Grids*, in "Future Generation Computer Systems", vol. 23, n^{o} 7, August 2007, p. 888-903.

[19] S. TIXEUIL. *Fault-tolerant distributed algorithms for scalable systems*, in "Wireless Ad Hoc and Sensor Networks", H. LABIOD (editor), ISBN: 978 1 905209 86, ISTE, October 2007.

[20] B. WEI, G. FEDAK, F. CAPPELLO. *Towards Efficient Data Distribution on Computational Desktop Grids with BitTorrent*, in "Future Generation Computer Systems", vol. 23, n^{o} 7, August 2007, p. 983-989.

[21] H. XIANG, Y. WEI. *On normwise structured backward errors for saddle point systems*, in "SIAM Journal on Matrix Analysis and Applications", vol. 29, 2007, p. 838-849.

Publications in Conferences and Workshops

[22] J. BEAUQUIER, J. CLÉMENT, L. ROSAZ, S. MESSIKA, B. ROZOY. *The counting problem in mobile sensor networks*, in "Disc 07, Lemenos, Cyprus", LNCS, September 2007.

[23] J. BEAUQUIER, Y.-J. HU. *Intrusion detection based on distance combination*, in "CESSE07, Venice, Italy", World Acacemy of Sciences, WAS, November 2007.

[24] L. BLIN, M. G. POTOP-BUTUCARU, S. TIXEUIL. *On the self-stabilization of mobile robots in graphs*, in "International Conference on Principles of Distributed Systems (OPODIS 2007)", Springer-Verlag, December 2007.

[25] F. BOUABACHE, T. HERAULT, G. FEDAK, F. CAPPELLO. *A Distributed and Replicated Service for Checkpoint Storage*, in "CoreGRID Workshop on Grid programming model, Grid and P2P systems architecture and Grid systems, tools and environments, Heraklion, Greece", 2007.

[26] M. CADILHAC, T. HERAULT, R. LASSAIGNE, R. LASSAIGNE, S. PEYRONNET, S. TIXEUIL. *Evaluating Complex MAC Protocols for Sensor Networks with APMC*, in "Proceedings of the 6th International Workshop on Automated Verification of Critical Systems (AVoCS 2006), Nancy, France", ENTCS, vol. 185, July 2007, p. 33–46.

[27] 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.

[28] 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.

[29] C. COTI, A. REZMERITA, T. HERAULT, F. CAPPELLO. *Grid Services for MPI*, in "Proceedings of the 14th European PVM/MPI User's Group Meeting, Paris, France", F. CAPPELLO, J. DONGARRA, T. HERAULT (editors), LNCS, n^{o} 4757, LNCS, October 2007, p. 388-389.

[30] M. GRADINARIU, S. TIXEUIL. *Conflict Managers for Self-stabilization without Fairness Assumption*, in "Proceedings of the International Conference on Distributed Computing Systems (ICDCS 2007)", IEEE, June 2007, 46.

[31] F. G. P. GREVE, S. TIXEUIL. *Condições de Conectividade para Realização de Acordo Tolerante a Falhas em Sistemas Auto-Organizàveis*, in "Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2007)", May 2007.

[32] F. GREVE, S. TIXEUIL. *Connaissance vs. Synchronie pour l'Accord Tolérant aux Pannes dans les Réseaux Inconnus*, in "Proceedings of Algotel 2007, Oléron", INRIA (editor), May 2007.

[33] F. GREVE, S. TIXEUIL. *Knowledge Connectivity vs. Synchrony Requirements for Fault-Tolerant Agreement in Unknown Networks*, in "Proceedings of IEEE International Conference on Dependable Systems and networks (DSN 2007)", IEEE, June 2007, p. 82-91.

[34] L. GRIGORI, X. S. LI. *Towards An Accurate Performance Modeling of Parallel Sparse LU Factorization*, in "SIAM Conference on Computational Science and Engineering", Asbtract Proceedings, 2007.

[35] L. GRIGORI, H. XIANG, J. DEMMEL, M. HOEMMEN, J. LANGOU. *Communication avoiding dense QR and LU factorizations*, in "Matrix Analysis and Applications (M2A07)", Abstract Proceedings, 2007.

[36] T. HERAULT, P. LEMARINIER, O. PERES, L. PILARD, J. BEAUQUIER. *A Model for Large Scale Self-Stabilization*, in "Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium, Long Beach, CA, USA", vol. 1, n^{o} 10, IEEE, March 2007.

[37] A. IOSUP, M. JAN, O. SONMEZ, D. EPEMA. *On the Dynamic Resource Availability in Grids*, in "Proc. of the 8th IEEE/ACM International Conference on Grid Computing (Grid 2007), Austin, TX, USA", September 2007, p. 26-33.

[38] D. KONDO, F. ARAUJO, P. MALECOT, P. DOMINGUES, L. M. SILVA, G. FEDAK, F. CAPPELLO. *Characterizing Result Errors in Internet Desktop Grids*, in "European Conference on Parallel and Distributed Computing EuroPar'07, Rennes, France", August 2007.

[39] X. S. LI, J. DEMMEL, L. GRIGORI, M. GU, J. XIA, S. JARDIN, C. SOVINEC, L.-Q. LEE. *Enhancing scalability of sparse direct methods*, in "Journal of Physics : Conference Series", vol. 78, 2007.

[40] F. MANNE, M. MJELDE, L. PILARD, S. TIXEUIL. *A New Self-Stabilizing Maximal Matching Algorithm*, in "Proceedings of the 14th International Colloquium on Structural Information and Communication Complexity (Sirocco 2007)", vol. 4474, Springer Verlag, June 2007, p. 96-108.

[41] 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.

[42] B. QUÉTIER, T. HERAULT, V. NERI, F. CAPPELLO. *Virtual Parallel Machines Through Virtualization: Impact on MPI Executions*, in "Proceedings of the 14th European PVM/MPI User's Group Meeting, Paris, France", F. CAPPELLO, J. DONGARRA, T. HERAULT (editors), LNCS, n^{o} 4757, LNCS, October 2007, p. 378-379.

[43] C. RANDRIAMARO, O. SOYEZ, G. UTARD, F. WLAZINSKI. *Dynamic distribution for data storage in a P2P network*, in "International conference on Grid and Pervasive Computing, Paris, France", May 2007.

[44] H. XIANG, L. GRIGORI. *Kronecker product approximation preconditioner for convection-diffusion model problems*, in "International Conference on Preconditioning Techniques", Abstract Proceedings, July 2007.

[45] 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.

Internal Reports

[46] S. DELAËT, P. S. MANDAL, M. ROKICKI, S. TIXEUIL. *Deterministic Secure Positioning in Wireless Sensor Networks*, Research Report, n^{o} 6326, INRIA, 10 2007, http://hal.inria.fr/inria-00179056.

[47] F. GREVE, S. TIXEUIL. *Knowledge Connectivity vs. Synchrony Requirements for Fault-Tolerant Agreement in Unknown Networks*, Research Report, n^{o} 6099, INRIA, January 2007, http://hal.inria.fr/inria-00123020.

[48] F. MANNE, M. MJELDE, L. PILARD, S. TIXEUIL. *A New Self-Stabilizing Maximal Matching Algorithm*, Research Report, n^{o} 6111, INRIA, January 2007, http://hal.inria.fr/inria-00127899.

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

Miscellaneous

[50] P. SERGE, D. MICHEL, C. CHRISTOPHE, F. CAPPELLO, N. THIERRY, P. BRIGITTE, R. MARIE-MADELEINE. *Supercalculateurs au Japon*, Written in french, June 2007.

References in notes

[51] K. AIDA, A. TAKEFUSA, H. NAKADA, S. MATSUOKA, S. SEKIGUCHI, U. NAGASHIMA. *Performance evaluation model for scheduling in a global computing system*, vol. 14, No. 3, 2000.

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

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

[54] D. ANDERSON. *BOINC*, http://boinc.berkeley.edu/.

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

[56] 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.

[57] 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.

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

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

[60] A. BOUTEILLER, F. CAPPELLO, T. HERAULT, 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.

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

[62] 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.

[63] 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.

[64] J. W. BYERS, M. LUBY, M. MITZENMACHER, A. REGE. *A Digital-Fountain Approach to Reliable Distribution of Bulk Data*, in "proc. of the ACL SIGCOMM", 1998.

[65] COSM. *Mithral Communications & Design Inc.*, http://www.mithral.com/.

[66] 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.

[67] J. CAO, STEPHEN A. JARVIS, S. SAINI, GRAHAM R. NUDD. *GridFlow: Workflow Management for Grid Computing*, in "Proceedings of the Third IEEE/ACM Internation Symposium on Cluster Computing and the Grid", May 2003.

[68] 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.

[69] H. CASANOVA, A. LEGRAND, D. ZAGORODNOV, F. BERMAN. *Heuristics for Scheduling Parameter Sweep Applications in Grid Environments*, in "Proceedings of the Ninth Heterogeneous Computing Workshop", IEEE Computer Society Press, 2000, p. 349-363.

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

[71] Y. CHEN, J. EDLER, A. GOLDBERG, A. GOTTLIEB, S. SOBTI, P. YIANILOS. *A prototype implementation of archival intermemory. In Proceedings of ACM Digital Libraries. ACM, August 1999.*.

[72] 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", vol. 9, n^{o} 11, November 1997, p. 1139–1160.

[73] B. COHEN. *Incentives Build Robustness in BitTorrent*, in "Workshop on Economics of Peer-to-Peer Systems, Berkeley", 2003.

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

[75] EDONKEY. *EDonkey, Overnet Homepage*, January 2002, http://mldonkey.sourceforge.net/.

[76] G. FEDAK, C. GERMAIN, V. NERI, 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.

[77] 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.

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

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

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

[81] C. GKANTSIDIS, P. RODRIGUEZ. *Network Coding for Large Scale Content Distribution*, in "Proceedings of IEEE/INFOCOM 2005, Miami, USA", March 2005.

[82] B. HUDZIA. *Use of Multicast in P2P Network thought Integration in MPICH-V2*, Technical report, Master of Science Internship, Pierre et Marie Curie University, September 2003.

[83] 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.

[84] J. KUBIATOWICZ, D. BINDEL, Y. CHEN, P. EATON, D. GEELS, R. GUMMADI, S. RHEA, H. WEATHERSPOON, W. WEIMER, C. WELLS, B. ZHAO. *OceanStore: An Architecture for Global-scale Persistent Storage*, in "Proceedings of ACM ASPLOS", ACM, November 2000.

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

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

[87] 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.

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

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

[90] MESSAGE PASSING INTERFACE FORUM. *MPI: A message passing interface standard. Technical report, University of Tennessee, Knoxville, June 12, 1995. 16*.

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

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

[93] PLATFORM. *Platform Computing - Accelerating Intelligence - Grid Computing*, http://www.platform.com.

[94] D. QIU, R. SRIKANT. *Modeling and Performance analysis of BitTorrent-like Peer-to-Peer Networks*, in "SIGCOMM Comput. Commun. Rev.", vol. 34, n^{o} 4, 2004, p. 367–378.

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

[96] A. L. ROSENBERG. *Guidelines for Data-Parallel Cycle-Stealing in Networks of Workstations I: On Maximizing Expected Output*, in "Journal of Parallel Distributed Computing", vol. 59, n^{o} 1, 1999, p. 31-53.

[97] 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.

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

[99] 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.

[100] SCIDAC. *SciDAC*, http://www.scidac.org.

[101] R. SHERWOOD, R. BRAUD, B. BHATTACHARJEE. *Slurpie: A Cooperative Bulk Data Transfer Protocol*, in "Proceedings of IEEE INFOCOM", March 2004.

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

[103] O. SIEVERT, H. CASANOVA. *Policies for Swapping MPI Processes. HPDC 2003: 104-113.*

[104] 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.

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

[106] TERAGRID. *TeraGrid*, http://www.teragrid.org.

[107] B. UK, M. TAUFER, T. STRICKER, G. SETTANNI, A. CAVALLI. *Implementation and Characterization of Protein Folding on a Desktop Computational Grid - Is Charmm a Suitable Candidate for the United Devices Metaprocessor*, Technical report, n^{o} 385, ETH Zurich, Institute for Comutersystems, October 2002.

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

[109] Y. YI, T. PARK, H. Y. YEOM. *A Causal Logging Scheme for Lazy Release Consistent Distributed Shared Memory Systems. In Proc. of the 1998 Int'l Conf. on Parallel and Distributed Systems, Dec. 1998. 1.*

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

[111] DATASYNAPSE. *Application Virtualization - DataSynapse Inc.*, http://www.datasynapse.com.

[112] GRIDSYSTEMS. *GRIDSYSTEMS - The Open Fabric of Virtualization*, http://www.gridsystems.com.