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 at INRIA-Futurs]

Topic leaders
Franck Cappello [Middleware Design, Implementation and Test]
Joffroy Beauquier [Verification]
Serge Petiton [Large Scale Numerical Computing]

Administrative assistant
Gina Grisvard [Administrative assistant INRIA-Futurs]

Permanent Members
Joffroy Beauquier [Professor at Paris-Sud University]
Franck Cappello [Research Director at INRIA-Futurs]
Gilles Fedak [Junior Researcher at INRIA-Futurs]
Thomas Hérault [Assistant Professor at Paris Sud University]
Serge Petiton [Professor at University of Science and Technology of Lille]
Brigitte Rozoy [Professor at Paris-Sud University]
Sébastien Tixeuil [Assistant Professor at Paris-Sud University]

Non Permanent Position
Lamine Aouad [Teaching Assistant at Lille 1 University]
Samir Djilali [Teaching Assistant at Paris-Sud University]
Marin Bertier [Teaching Assistant at Paris-Sud University]

Ph. D. student
Aurelien Bouteiller [MESR Grant (LRI)]
Matthieu Cargnelli [EADS Industrial Grant (CIFRE)]
Laurent Choy [INRIA et Région Nord (LIFL)]
Philippe Gauron [MESR Grant (LRI)]
Toussaint Guglielmi [MESR Grant (LIFL)]
William Hoarau [MESR Grant (LRI)]
Benoit Hudzia [Franco-Irish Grant (LIFL)]
David Ilcinkas [MESR Grant (LRI)]
Pierre Lemarinier [MESR Grant (LRI)]
Oleg Lodygensky [LaL Engineer (Laboratoire de l'Accelerateur Lineaire)]
Nicolas Nisse [MESR Grant (LRI)]
Benjamin Quettier [MESR Grant (LRI)]
Baohua Wei [Industrial Chinese Grant (LRI)]

Research scientist (partner)
Pierre Fraigniaud [Research Director at CNRS]
Rosaz Laurent [Assistant Professor at Paris Sud University]

Project technical staff
Tangui Morlier [INRIA Associate Engineer]
Phillipe Marty [INRIA Expert Engineer]
Vincent Neri [CNRS Study Engineer]

2. Overall Objectives

2.1. Grand-Large General Objectives

Grand-Large is a Grid research project investigating the issues raised by computing on Large Scale Distributed Systems (LSDS), where participants execute different applications on shared resources belonging

to other participants, possibly geographically and administratively independent. More specifically, we consider large scale parallel and distributed computing on P2P, Global Computing and Desktop Grid systems. Our research focuses on middleware and low level programming environments design, proof and experiments. Fundamentally, we address the impact of LSDS, gathering several methodological tools: theoretical models, simulators, emulators and real size systems.

The project aims:

1. to study experimentally, and formally, the fundamental mechanisms of LSDS for high performance computing;
2. to design, implement, validate and test real software, middleware and platform;
3. to define, evaluate and experiment approaches for programming applications on these platforms.

Compared to other European and French projects, we gather skills in large scale systems (large scale scheduling, volatility tolerance, heterogeneity, inter administration domain security, etc.) acquired with the XtremWeb project (LRI, Cluster and Grid team), formal design and validation of algorithms and protocols for distributed systems (LRI, Parallelism team) and programming, evaluation, analysis and definition of programming languages and environments for parallel architectures and distributed systems (LIFL, methodologies and parallel algorithms).

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

1. the design of a middleware enlarging the application domain of Desktop Grid;
2. resource discovery engine on large scale system with volatil participants;
3. large scale storage on volatile nodes;
4. simulation of large scale scheduling;
5. fault tolerant MPI for large scale systems;
6. algorithm for large scale fault tolerance;
7. protocol verification;
8. algorithms, programming and evaluation of scientific applications on desktop Grids;
9. tools and languages for large scale computing.

These researches should have some applications in the domain of LSDS, Grid and large clusters.

At a longer term, we investigate the convergence conditions of Global Computing, P2P and Grid systems (how Grid Services can be used in Desktop Grid) and experimental tools for improving the methodology associated with research in LSDS. For example we have the responsibility of the Grid eXplorer project founded by the French ministry of research and we are deeply involved in the Grid5000 project.

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. The notion of Large Scale is linked to a set of features that has to be taken into account if the system should scale to a very high number of nodes. An example is the node volatility: 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. This feature raises many novel issues: under such assumptions, the system may be considered as fully asynchronous (it is impossible to provide bounds on message transits, thus impossible to detect some process failures), so as it is well known [68] no consensus could be achieved on such a system. Another example of feature is the complete lack of control of nodes and networks. We cannot decide when a node contributes to the system nor how. This means that we have to deal with the in place infrastructure in terms of performance, heterogeneity and dynamicity but also with the fact that any node may intermittently inject Byzantine faults. These features set up 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

Currently, largest LSDS are used for Computing (SETI@home, Folding@home, Decrypthon, etc.), file exchanges (Napster, Kazaa, eDonkey, Gnutella, etc.), networking experiments (PlanetLab, Porivo) and communication 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.

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 [90] 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 [81], Glunix [74] and Mosix [48], to cite a few. A first approach to cross administration domains was proposed by Web Computing projects such as Jet [84], Charlotte [49], Javeline [64], Bayanihan [88], SuperWeb [46], ParaWeb [55] and PopCorn [57]. 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 recent generation of Global Computing systems like COSM [1], BOINC [2] and XtremWeb [67].

The high performance potential of LSDS platforms has also raised a significant interest in the industry. Companies like Entropia [63], United Devices [95], Platform [3], Grid systems [4] and Datasynapse [5] propose LSDS middleware often known as Desktop Grid or 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 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 [69], the authors present the similarities

[1] http://www.mithral.com/
[2] http://boinc.berkeley.edu/
[3] http://www.platform.com
[4] http://www.gridsystems.com
[5] http://www.datasynapse.com

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 will 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 system for computing would be in principle very close to a P2P file sharing system such as Napster [89], Gnutella [89] and Kazaa [80], except that the ultimate 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 similar solutions will be adopted for their fundamental mechanisms 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 [85], CHORD [92], PASTRY [87] and TAPESTRY [98] 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 or reconfiguration, volatility tolerance, anonymity, limited used of the network, matching between the topologies of the underlying network and the virtual overlay network. The currently proposed architectures are not well motivated and seem to be derived from arbitrary choices.

This study has two objectives: a) compare some existing resource discovery architectures (centralized, hierarchical, fully distributed) with relevant metrics; and b) potentially propose a new protocol improving some parameters. Comparison will consider the theoretical aspects of the resource discovery engines as well as their actual performance when exposed to real experimental conditions.

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.

Several architectures exist for application parameters and results communication between the client node and the computing ones. XtremWeb uses an indirect transfer through the task scheduler which is implemented by a middle tier between client and computing nodes. When a client submits a task, it encompasses the application parameters in the task request message. When a computing node terminates a task, it transfers it to the middle tier. The client can then collect the task results from the middle tier. BOINC [] follows a different architecture using a data server as intermediary node between the client and the computing nodes. All data transfers still pass through a middle tier (the data server). DataSynapse [6] allows direct communications between the client and computing nodes. This architecture is close to the one of file sharing P2P systems. The client uploads the parameters to the selected computing nodes which return the task results using the same

[6] http://www.datasynapse.com

channel. Ultimately, the system should be able to select the appropriate transfer approach according to the performance and fault tolerance issues. We will use real deployments of XtremWeb to compare the merits of these approaches.

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 [77] and Ocean [62]. 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. We will evaluate the different proposed approaches and the how replication, affinity, cache and persistence influence the performances of computational demanding applications.

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 [60], [58]. Recently some researches consider scheduling and migrating MPI applications on Grid [91]. Other related researches concern scheduling for cycle stealing environments [86]. 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). There are several strategies to deal with large scale system: introducing hierarchy or/and giving more autonomy to the nodes of the distributed system. The purpose of this research is to evaluate the benefit of these two strategies in the context of LSDS where nodes are volatile. In particular we are studying algorithms for fully distributed and asynchronous scheduling, where nodes take scheduling decisions only based on local parameters and information coming from their direct neighbors in the system topology. In order to understand the phenomena related to full distribution, asynchrony and volatility, we are building a simulation framework called V-Grid. This framework, based on the Swarm [83] multi-agent simulator, allows describing an algorithm, simulating its execution by thousands of nodes and visualizing dynamically the evolution of parameters, the distribution of tasks among the nodes in a 2D representation and the dynamics of the system with a 3D representation. We believe that visualization and experimentation are a first necessary step before any formalization since we first need to understand the fundamental characteristics of the systems before being able to model them.

3.1.2.4. Extension of MPICH-V

MPICH-V is a research effort with theoretical studies, experimental evaluations and pragmatic implementations aiming to provide a MPI implementation based on MPICH [82], 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 [61]. 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 [96], pessimistic [47], causal [97]. 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.

Whithin the MPICH-V project, we have developed and published 3 original fault tolerant protocols for MPI: MPICH-V1 [52], MPICH-V2 [53], MPICH-V/CL [54]. 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-V/CL implements a

coordinated checkpoint strategy (Chandy-Lamport) removing the need of message logging. MPICH-V2 and V/CL 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. Under the considered experimental conditions, message logging becomes more relevant than coordinated checkpoint when the fault frequency reach 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.

The next step in our research is to investigate a protocol dedicated for hierarchical desktop Grid (it would also apply for Grids). In such context, several MPI executions take place on different clusters possibly using heterogeneous networks. An automatic fault tolerant MPI for HDG or Grids should tolerate faults inside clusters and the crash or disconnection of a full cluster. We are currently considering a hierarchical fault tolerant protocol combined with a specific runtime allowing the migration of full MPI executions on clusters independently of their high performance network hardware.

The performance and volatility tolerance of MPICH-V make it attractive for :

1. large clusters;
2. clusters made from collection of nodes in a LAN environment (Desktop Grid);
3. Grid deployments harnessing several clusters;
4. and campus/industry wide desktop Grids with volatile nodes (i.e. all infrastructures featuring synchronous networks or controllable area networks).

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. There is no time for saving the state of the computation. 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 problem that arises from this way of managing computations are numerous and difficult. They can be put into two categories: synchronization and repartition problems.

- Synchronization problems (example). Suppose 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 makes 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 ([72]). 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 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 transient failures as loss of message duplication in between. On the other hand, failures related accidental or malicious memory corruptions have to be considered because they are directly related of 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. [44] [93] [65] for surveys). Masking techniques, generally based on upon consensus, because they systematically use generalized broadcasting are not scalable. The self-stabilizing approach (a non-masking solution) is well adapted (specifically its time adaptive version, cf. [79] [78], [50], [51], [73]) for three main reasons:

1. Low overhead when stabilized. Once the system is stabilized, the overhead for maintaining correction is slow because it only involves communications between neighbors.
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 all kind of application.
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 tradeoff between reliability and cost.

3.2.2. Verification of Protocols

For the past few years, a number of distributed algorithms or protocols that were published in the best conferences or scholar journals were found to be incorrect afterwards. Some have been exploited for several years, appearing to behave correctly. We do not pretend to design and implement fault free and vulnerability free systems, but we want at least to limit their failures. This goal is achieved by the formal verification, at an abstract level, of the implemented solutions. Obviously, algorithms are not to be verified by hand (incorrect algorithms were provided with proofs), but rather by verification tools we developed (MARELLA) or proof assistants. We propose that a substantial effort is done towards modelization and verification of probabilistic protocols, which offer in a large number of cases efficient and low cost solutions. We also propose to design a model that includes the environment. Indeed, computations of a distributed system are non-deterministic due to the influence of numerous external factors, such as the communication delays due to traffic overhead, the fact that failures can occur somewhere rather than somewhere else, etc. To prove a protocol independently of its environment is pointless, and this is why the environment must be part of the model.

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

Scientific applications that have traditionally performed on supercomputers may now run on a variety of heterogeneous resources geographically distributed. New grand challenge applications would have to be solved on large scale P2P systems. 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 nondeterministic. 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 a decade. Since the last few years, the scale of the problem sizes and the global complexity of the applications increase rapidly Teragrid [7]. 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, 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 Scidac [8], [76]. 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. To come at his age, P2P programming has to be able to develop parallel programming with more sophisticate dependencies between peers. It is the goal of our researches.

3.3.2. Experimentations and Evaluations

We have, first, to experiment on large P2P platforms to be able to obtain a realistic evaluation of the performance we can expect. We can also set some hypothesis on peers, networks, and scheduling to be able to have theoretical evaluations of the potential performance. We follow these two tracks. We choose 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 also choose the calculation of one matrix polynomial, which generate 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 these evaluations, we begin to experiment the same parallel methods on a few dozen peer XtremWeb P2P Platform. We plan to continue these experimentations on larger platforms to compare these results to the theoretical ones. Then, we would be able to extrapolate and obtain potential performance for some scientific applications. 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 to efficiently invert a dense matrix of size one million using a several thousand peer platform.

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.

[7] http://www.teragrid.org
[8] http://www.scidac.org

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 may develop component-based technology interface which express the dependencies between computing tasks which composed the parallel applications. Then, instead of computing task we will manage components. We introduced the YML language which allows us to express the dependencies between components, specified using XML. Nevertheless, many component criteria depend of peer characteristics and are known only at runtime. Then, we had to introduce different classes of components, depending of the level of abstraction they are concern to. A component catalogue has to be at the end-user level and another one has to be at the middleware and peer level. Then, a scheduler has to attribute a computing component to a peer with respect to the software proposed by this one, or has to decide to load new software to the targeted peer.

The YML framework and language propose a solution to develop scientific applications to P2P platform. An end-user can directly develop programs using this framework. Nevertheless, many end-users would prefer to do not program at this component and dependency graph level. Then, an interface has to be proposed, 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.

Based on the SPIN project, 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 and Technologies 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. The general research context should also considers the fundamental technological trend toward a convergence between Grid and P2P systems.

3.4.1. Metodology

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).

Grand-Large aims at gathering several complementary techniques to study the impact of large scale in LSDS: theoretical models, simulation, emulation and experimentation on real platforms. Fundamental aspects of LSDS as well as the development of middleware platforms are already existing in Grand-Large. We are also involved in the development and deployment of simulators and emulators and real platforms (testbed).

We are currently developing a simulator of LSDS called V-Grid aiming at discovering, understanding and managing implicit uncoordinated large scale phenomena. Several Grid simulators have been developed by other teams: SimGrid [59] GridSim [56], Briks [45]. 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 [83] but many of them considers synchronous systems where the system evolution is guided by phases. V-Grid is built from Swarm and adds asynchrony in the simulator, node volatility and a set of specialized features for controlling and measuring the simulation of LSDS. To exemplify the need of such simulator, we are first considering the fully distributed scheduling problem. Using V-Grid for comparing several algorithms, we have already demonstrate the need for complementary visualization tools, showing the evolution of key system parameters, presenting the distributed system topology, nodes and network global trends in a 2 dimensional shape and presenting the dynamics of the system component activity in a 3 dimensional shape. Using this last representation, we have discover unexpected large scale phenomena which would be very difficult to predict by a theoretical analysis of the simulated platform features and the scheduling algorithms.

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. This project uses a 1K CPUs cluster as hardware platform and gathers 24 experiments of 80 researchers belonging to 13 different laboratories. Experiments concern developing the emulator itself and use of the emulator to explore LSDS issues. (http://www.lri.fr/~fci/GdX/).

Grand-Large members are also involved in the French Grid 5000 project which intents to deploy an experimental Grid testbed for computer scientists. This testbed may feature up to 5000 K CPUs gathering the resources of about 10 clusters geographically distributed over France. The clusters will be connected by a high speed network (Renater or/and other). Grand-Large is a leading team in Grid 5000, chairing the eGrid 5000 Specific Action of the CNRS which is intended to prepare the deployment and installation of Grid 5000. eGrid 5000 gathers about 30 engineers, researchers and team directors who have frequent meetings, discussing about the testbed security infrastructure, experiment setup, cluster coordination, experimental result storage, etc. (http://www.lri.fr/~fci/AS1/).

3.4.2. Technological Trends

The development of LSDS has followed a trajectory parallel to the one of Grid systems such as Globus [70] and Unicore [66]. Nevertheless we can observe some convergence elements between LSDS and Grid. The paper [69] gives many details about the similarities and differences between P2P and Grid systems. From the technological perspective, the evolution of Globus to GT3 [71] with the notion of Grid services is one reason of this convergence. The evolution of LSDS toward more generic and secure systems being able to provide CPU, storage and communication sharing among participants is another element of this convergence, since the notion of controllable services is likely to emerge from this perspective of more generality and flexibility.

Nowadays, Grid Computing is considering the notion of services through OGCSA [71] and OGSI [94]. A Grid service is an entity that must be auto-descriptive, dynamically published, creatable and destructible, remotely invoked and manageable (including life time cycle). The standardization effort also includes the use of well defined standards (WSDL, SOAP, UDDI...) of Web Services [9]. A typical LSDS platform gathering client nodes submitting requests to a coordination service which schedules them on a set of participating nodes can be implemented in term of services: the coordination service publishes application services and schedules their instantiations on workers; the client service requests task (association of application and parameters) executions corresponding to published application services and collects results from the coordination service; the worker service computes tasks and sends their results back to the coordination service. Note that the implementation of the coordination service can rely on sub-services such as a scheduler, a data server for parameters and results, a service repository/factory which themselves may be implemented in centralized or distributed way.

Thus we believe that LSDS could benefit from the standardization effort conducted in the Grid context by reusing the same concepts of services and by adopting the same standards (OGSA and OGSI). For example, the next version of XtremWeb will be implemented by a set of Grid services.

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)

[9] http://www.webservices.org/

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 Large Scale Programming for Computational Science and Engineering. Two main approaches are proposed. First, we have to experiment and evaluate such programming. Second, we have to develop tools for end-users.

In addition to the classical supercomputing and the GRID computing based on virtual organization, the large scale P2P approach proposes new computing facilities for computational scientists and engineers. Thus, on one hand, it exists many applications, some of them are classical, such as Computational Fluid Dynamic or Quantum Physic ones, for example, and others are news and very strategic such as Nanotechnologies, which will have to use a lot of computing power for long period of time in the close future. On another

hand, it emerges a new large scale programming paradigm for existing computers which can be accessible by scientific and engineer end-users for all classical application domains but also by new ones, such as some Non-Governmental Organisations. During a first period, many applications would 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. Nevertheless, we expected that more complex implicit methods would be adapted in the future for such programming. The potential number of peer and the planed evolution of network communications, especially multicast ones, would permit to contribute to solve some of the larger grand challenge scientific applications.

Simulations and large implicit methods would always have to compute linear algebra routines, which will be 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 will especially first focus on divide-and-conquer and block-based matrix methods to solve dense problems and on iterative 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 [75]. 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 virtualized the P2P middleware and the heterogeneous hardware. Our approach is based, on one hand, on component programming and coordination languages, and on one another hand, to the development of an ASP, which may be dedicated to a targeted scientific domain. The conclusion 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. Currently tasks concern computation but we are also considering the integration of storage and communication capabilities. 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. 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 authentication, application sandboxing and communication encryption. We have developed deployment tools for harnessing individual PCs, PCs in University or Industry laboratories and PCs in clusters. XtremWeb provides a RPC interface for bag of tasks, parameter sweep, master worker and workflow applications. Associated with MPICH-V, XtremWeb allows the execution of unchanged MPI applications on Desktop Grids.

XtremWeb has been tested extensively harnessing a thousand of Workers and computing a million of tasks. XtremWeb is deployed in several sites: University of Lille, University of Geneva, University of Tsukuba,

University of Paris Sud, University of California San Diego. In this last site, XtremWeb is the Grid engine of the Paris Sud University Desktop Grid gathering about 500 PCs. Two multi-parametric applications are to be used in production since the beginning of 2004: Aires belonging to the HEP Auger project and a protein conformation predictor using a molecular dynamic simulator.

The software, papers and presentations are available at http://www.xtremweb.net.

5.2. MPICH-V

Currently, MPICH-V proposes 6 protocols: MPICH-V1, MPICH-V2, MPICH-V/CL, 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-V/CL, 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-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.

MPICH-V3 will be studied for the Grids. It will rely on a new protocol mixing causal message logging and pessimistic remote logging of message events. This is a hierarchical protocol able to tolerate fault inside Grid sites (inside clusters) and faults of sites (the complete crash of clusters).

Another effort is pushed on the performances of MPICH-V for high-bandwidth networks. This introduces the necessity of zero-copy implementations and raises new problems with respect to the algorithms and their realization. The goal sought here is to provide fault tolerance without losing high performances.

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://www.mpich-v.net/

5.3. YML

The complexity of P2P platforms is important. An end-user cannot manage manually such complexity. We provide a set of tools designed to develop and execute large coarse grain applications on peer-to-peer systems. We developed and did the first experimentations of the YML framework for parallel programming on P2P architectures.

The main part of YML project is a 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 is a way to link components: a coordination language called Yvette. This language can express graphs of components. These graphs represent applications. The goal of this split is to manage complex coupled applications on peer-to-peer systems.

We designed a framework to take advantage of YML language. It is composed of two directories and the YML Daemon. The daemon written in JAVA uses information contained in both directories to compile and execute YML applications on top of a peer to peer system. We identify first the two main roles of the daemon. Each role relies on a specific directory. This strict separation enhances portability of applications and permits optimization during the execution stage in real-time. Currently we provide support for the XtremWeb peer to peer middleware.

To illustrate our approach, we did first experimentations for basic linear algebra routines on an XtermWeb P2P platform with a small number of peers. We did performance evaluations and discussed on the necessity to introduce a new accurate performance model for this new computing paradigm.

YML project was launched at the ASCI CNRS lab in 2001 and is developed now in collaboration with the University of Versailles. YML is under integration into SPIN to propose a GUI ASP.

5.4. 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.

5.5. V-Grid

This project is in its early stage. It started officially in September 2004. V-Grid 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-Grid 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-Grid virtualizes and schedules a full software environment for every distributed system node. V-Grid 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 realism and performance. As a consequence, we are currently studying approaches to cope with these limits.

5.6. 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.

6. New Results

6.1. Large Scale Distributed Systems

Hybrid Preemptive Scheduling of MPI Applications on the Grids [21]

Time sharing between cluster resources in Grid is a major issue in cluster and Grid integration. Classical Grid architecture involves a higher level scheduler which submits non overlapping jobs to the independent batch schedulers of each cluster of the Grid. The sequentiality induced by this approach does not fit with the expected number of users and job heterogeneity of the Grids. Time sharing techniques address this issue by allowing simultaneous executions of many applications on the same resources.

Co-scheduling and gang scheduling are the two best known techniques for time sharing cluster resources. Co-scheduling relies on the operating system of each node to schedule the processes of every application. Gang scheduling ensures that the same application is scheduled on all nodes simultaneously. Previous work has proven that co-scheduling techniques outperform gang scheduling when physical memory is not exhausted.

We compare experimentally the merits of three solutions: Co, Gang and Hybrid Scheduling, in the context of out-of-core computing, which is likely to occur in the Grid context, where many users share the same resources. The experiments show that the hybrid solution is as efficient as the co-scheduling technique when the physical memory is not exhausted, and is more efficient than gang scheduling and co-scheduling when physical memory is exhausted. We conclude that hybrid scheduling reaches slightly better or equal throughput, and offers a better fairness.

Fault Tolerance for MPI

Fault tolerance is a very important concern for critical high performance applications using the MPI library. Several protocols provide automatic and transparent fault detection and recovery for message passing systems with different impact on application performance and the capacity to tolerate a high fault rate.

The conclusion of the paper “Coordinated checkpoint versus message log for fault tolerant MPI” [3], published this year in its journal version ([9]), is twofold:

- uncoordinated checkpointing tolerates higher fault frequency than coordinated checkpointing.
- the main differences between pessimistic sender based message logging and coordinated checkpointing are 1) the communication latency and 2) the performance penalty in case of faults;

This year, we conducted two studies to improve understanding and knowledge on these two aspects:

Impact of Event Logger on Causal Message Logging Protocols for Fault Tolerant MPI [35]

Causal message logging protocols have been proven the most efficient message logging technique. These protocols consist in piggybacking non deterministic events to computation message. Several protocols have been proposed in the literature. In this paper, we investigate the benefit of using a stable storage for logging message events in causal message logging protocols. To evaluate the advantage of this technique we implemented three protocols: 1) a classical causal message protocol proposed in Manetho, 2) a state of the art protocol known as LogOn, 3) a light computation cost protocol called Vcausal. We demonstrate a major impact of this stable storage for the three protocols, on the four criteria for micro benchmarks as well as for the NAS benchmark.

We evaluated the impact of the Event Logger on the three protocols for Fast-Ethernet network. We demonstrated that the presence of the Event Logger has a major impact on the four performance criteria: a) piggybacking computation cost, b) piggyback size, c) applications performance and d) fault recovery performance whatever is the causal message logging protocol considered. We also demonstrated that methods based on antecedence graph perform better than Vcausal basic techniques but use much more computation time handling causality.

Improved Message logging versus Improved coordinated checkpointing for fault tolerant MPI [34]

Pessimistic message logging increases the latency, due to additional blocking control messages. When faults occur at a high rate, coordinated checkpointing implies a higher performance penalty than message logging due to a higher stress on the checkpoint server. In this paper, we compare the performances against the previous versions and we compare the new message logging protocols against the improved coordinated checkpointing one using the NAS benchmark on a typical high performance cluster equipped with a high speed network.

We have demonstrated that recovery overhead of the coordinated version is significantly lower than the previous one without any additional fault free overhead. Latency overhead of the causal version is reduced at the cost of a slight bandwidth decrease compared to the pessimistic one. The fault free performance difference between causal and coordinated is smaller than between pessimistic and the previous coordinated, and improved coordinated tolerates higher fault frequency than the previous one, but still does not reach fault resilience of message logging techniques due to checkpoint server stress during checkpoint.

Computing on Large Scale Distributed Systems: XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid [10]

Global Computing and Desktop Grid systems belong to the class of large scale distributed systems. Their properties: very high computational, storage and communication performance potentials, very high resilience make them attractive in academia and industry as computing infrastructures in complement to more classical infrastructures such as clusters or supercomputers. However, generalizing the use of these systems in a multi-user and multi-parallel programming context involves finding solutions and providing mechanisms for many issues such as programming bag of tasks and message passing parallel applications, securing the application, the system itself and the computing nodes, deploying the systems for harnessing resources managed in different ways. In this paper, we present our experience based on the XtremWeb project towards a Computational P2P system. We describe a) the architecture of the system and its motivations, b) the parallel programming paradigms available in XtremWeb and how they are implemented, c) the deployment issues and what mechanisms are used to harness simultaneously individual resources, uncoordinated set of resources, and resources managed by batch schedulers and d) the security issue and how we address, inside XtremWeb, the protection of the computing resources. We present two multi-parametric applications to be used in production: Aires belonging to the HEP Auger project and a protein conformation predictor using a molecular dynamic simulator. To evaluate the performance and volatility tolerance, we present experiment results for bag of tasks applications and message passing applications. We show that the system can resist to massive fault and we discuss the performance of the node protection mechanism.

RPC-V: Toward Fault-Tolerant RPC for Grids with Volatile Nodes [23]

RPC is one of the programming models envisioned for the Grid. In Internet connected Large Scale Grids such as Desktop Grids, nodes and networks failures are not rare events. This paper provides several contributions, examining the feasibility and limits of fault-tolerant RPC on these platforms.

First, we characterize these Grids from their fundamental features and demonstrate that their applications scope should be safely restricted to stateless services.

Second, we present a new fault-tolerant RPC protocol associating an original combination of three-tier architecture, passive replication and message logging. We describe RPC-V, an implementation of the proposed protocol within the XtremWeb Desktop Grid middleware.

Third, we evaluate the performance of RPC-V and the impact of faults on the execution time, using a real life application on a Desktop Grid testbed assembling nodes in France and USA. We demonstrate that RPC-V allows the applications to continue their execution while key system components fail.

6.2. Large Scale Peer to Peer Performance Evaluations

Large Scale Peer to Peer Performances Evaluations with Gauss-Jordan Methods as an Example [20]

We present a large scale block-based Gauss-Jordan algorithm to invert very large dense matrices. This version proposes to exploit peer to peer platforms with increasingly sets of distributed resources. We assume that we have access to a scheduler that proposes strategies allowing persistent storage of data and migration anticipation heuristics. Under given hypotheses, we present the upper bounds of theoretical evaluation results, using different peer to peer platforms interconnected by different networks. The results show that the management and the speed of communications are crucial points for performances evaluation for such platforms. Nevertheless, we discuss that, in these cases, the classical evaluation model is not well adapted to this peer to peer computing paradigm for large scale scientific applications.

A Peer to Peer Computing Framework ; design and Performance Evaluation of YML [22]

We present the YML framework which provides supporting tools to design and execute portable parallel applications over large scale peer to peer and grid middlewares. The YML Framework defines a new parallel programming language called YvetteML which is composed of a graph language and a component model. We evaluate the performance of our framework with a simple numerical application using XtremWeb as a middleware.

A Parallel Asynchronous Hybrid Method to Accelerate Convergence of a Linear System [30]

We have worked on a method to solve linear systems described by large sparse matrices using a heterogeneous parallel network comparable to GRID architecture. This method combines the principal solving algorithm GMRES (m) implemented on a parallel computer, with Least Squares method that needs some eigenvalues calculated simultaneously and asynchronously with another machine by Arnoldi method. This hybrid method allows taking advantage of available parallelism to accelerate the convergence by decreasing considerably the number of iterations.

This method has also been implemented on a supercomputer IBM SP3, associating different groups of processors to each algorithm of it. Results show a spectacular acceleration of convergence. We observe also some matrices that do not converge with GMRES(m) method itself converge with our hybrid method.

Reliable Multicast Fault Tolerant MPI in the Grid Environment [33]

We present an implementation of a reliable multicast protocol over a fault tolerant MPI: MPICH-V2. It can provide one way to solve the problem of transferring large chunks of data between applications running on a grid with limited network links. We provide a theoretical cluster organisation and GRID network architecture to harness the performance provided by using multicasts with data compression.

Parallel Matrix Method Experimentations on Distributed P2P Platforms [36]

We present large-scale parallel matrix computing experiment on P2P platforms for several basic linear algebra methods such as matrix-matrix products and $A(A(x) + x) + x$ computations, linear system solving (Block Gauss-Jordan) and eigenvalue approximations (Givens-Householder). We explain how we adapted parallel methods for P2P platforms and we present performance evaluation with respect to several parameters.

We conclude that P2P matrix computing has, at least, to use more sophisticated scheduling strategies and middlewares to be proposed to end-users.

Restartable Tight Coupled Algorithms for SMP Cluster Architectures [38]

We analyze a hybrid method based on Krylov subspace involved in solving most sparse linear algebra problems underlying many large-scale scientific applications. We propose a strategy to realize an efficient hybrid computation among the co-methods, based on asynchronous non-blocking communications. We evaluate performance on cluster of SMP, such as IBM SP3 and SP4 for matrixmarket matrices. As conclusion, we open long term research perspectives to adapt these methods for internet connected grid architectures.

Experimentations and Programming Paradigms for Matrix Computing on Peer to Peer Grid [37]

We discuss the peer to peer solution for the deployment of large scale numerical applications on distributed resources. We use the XtremWeb P2P middleware for managing our computational resources. This global computing platform, such as much current grid deployments, is currently dedicated to multi-parameters applications. Our goal is to use it for executing numerical parallel applications that require communications between tasks. We discuss the performances of a dense block based matrix-vector product deployed for two XtremWeb networks; a local one with 128 non dedicated PCs, and for 261 PCs distributed on two geographic sites: University of Lille I and Paris XI University at Orsay. We give an investigation of the scheduling schemes well adapted for such systems and applications. We also present the performances of out-of-core products and a new programming paradigm using out-of-core technique for peer to peer numerical applications.

6.3. Volatility and Reliability Processing

Wireless sensor networks benefit from communication protocols that reduce power requirements by avoiding frame collision. Time Division Media Access methods schedule transmission in slots to avoid collision, however these methods often lack scalability when implemented in *ad hoc* networks subject to node failures and dynamic topology. [31] reports a distributed algorithm for TDMA slot assignment that is self-stabilizing to transient faults and dynamic topology change. The expected local convergence time is $O(1)$ for any size network satisfying a constant bound on the size of a node neighborhood.

In large scale multi-hops wireless networks, flat architectures are not scalable. In order to overcome this major drawback, clusterization is introduced in order to support a self-organization and enable hierarchical routing. When dealing with multi-hops wireless networks the robustness is a main issue since such networks are deployed in hostile environment.

Self-stabilization in Self-organized Wireless Multi-hop Networks [41]

We study the problem of self-organization of multi-hops wireless networks and present a distributed algorithm that is self-stabilizing to transient fault and dynamic topology changes. The expected convergence time is $O(1)$ for any size of network satisfying a constant bound on the node degree (*i.e.*, the size of a node's neighborhood).

Mutual exclusion

The mutual exclusion problem is fundamental in distributed computing, since it permits processors that compete to access a shared resource to be able to synchronize and get exclusive access to the resource (i.e. execute their critical section). It is well known that providing self-stabilization in general uniform networks (e.g. anonymous rings of arbitrary size) can only be probabilistic.

However, all existing uniform probabilistic self-stabilizing mutual exclusion algorithms designed to work under an unfair distributed scheduler (that may choose processors to execute their code in an arbitrary manner) suffer from the following common drawback: once stabilized, there exists no upper bound on time between two successive executions of the critical section at a given processor.

Self-stabilizing Mutual Exclusion with Arbitrary Scheduler [12]

We present the first self-stabilizing algorithm that guarantees such a bound ($O(n^3)$, where n is the network size) while working using an unfair distributed scheduler. Our algorithm works in an anonymous unidirectional ring of any size and has a polynomial expected stabilization time.

Optimal Randomized Self-stabilizing Mutual Exclusion in Synchronous Rings [24]

We propose several self-stabilizing protocols for unidirectional, anonymous, and uniform synchronous rings of arbitrary size, where processors communicate by exchanging messages. When the size of the ring n is unknown, we better the service time by a factor of n (performing the best possible complexity for the stabilization time and the memory consumption). When the memory size is known, we present a protocol that is optimal in memory (constant and independent of n), stabilization time, and service time (both are in $\Theta(n)$).

Stabilizing Inter-domain Routing in the Internet [11]

This paper reports the first self-stabilizing Border Gateway Protocol (BGP). BGP is the standard inter-domain routing protocol in the Internet.

The routing instability in the Internet can occur due to errors in configuring the routing data structures, the routing policies, transient physical and data link problems, software bugs, and memory corruption. This instability can increase the network latency, slow down the convergence of the routing data structures, and can also cause the partitioning of networks. Most of the previous studies concentrated on routing policies to achieve the convergence of BGP while the oscillations due to transient faults were ignored.

The purpose of self-stabilizing BGP is to solve the routing instability problem when this instability results from transient failures. The self-stabilizing BGP presented here provides a way to detect and automatically recover from this type of faults. Our protocol is combined with an existing protocol to make it resilient to policy conflicts as well.

Auto-stabilisation et Protocoles Réseaux [18]

This paper surveys works that propose self-stabilizing solutions to problems arising in Compute Networks, such as routing and transport layers, or network control protocols. We also review techniques to design self-stabilizing protocols, and mechanisms that reduce the stabilization time when the number of hitting faults is small.

Occurence preserving congruences and associated graphs [8]

We regard occurrence preserving congruences on the right as a generalization of Mazurkiewicz traces. We study the relations between the prefix-graph of a trace, which is a pertinent representation of a trace and of its prefixes, and the ideal-graph which is a representation of the order associated with the trace. We give examples showing the differences between Mazurkiewicz traces and this generalization. Then we study the properties and the relations between these two graphs. To finish with, we relate the Post correspondence problem with such congruences.

Traditionally, self-stabilization is considered as suffering two main drawbacks. The first is that a self-stabilizing system recovers only after some time, meaning that during that time the behaviour is not correct. The second is that a process can never know whether or not the system is stabilized. There is little we can do about the first problem because it is inherently bound to the very definition of self-stabilization. We simply have to pay a much smaller overhead than for robust algorithms approach, e.g., consensus, and the payload is the temporary loss of safety. We tried to solve the observation problem by using much less safe memory than previous approaches. In particular, we raised the question of whether or not stabilization detection is feasible if only one node has some stable memory. Our main result [7] can be stated as follows: if there exists a self-stabilizing distributed solution for some problem in some network, then there exists a self-stabilizing distributed solution (not necessarily the same) for the same problem in the same network that can be observed by an observer located at a particular node. Moreover the observable self-stabilizing solution can be effectively built from the simply self-stabilizing solution in a canonical way, and therefore it can be built into a compiler.

6.4. Nation Wide Experimental Platforms (testbed)

6.4.1. Grid'5000

The Grid is envisioned to become a main infrastructure to provide seamless and transparent access to computing, storage, communications and service facilities to Internet users. After a first experimentation phase, generally with a low number of resources, new projects are unveiled with the objective to build large scale Grids combining hundreds of computers around the world for thousands of users. The European EGEE project is one example in Europe.

However, Grids are very complex objects because they are fundamentally distributed systems gathering complex and potentially volatile nodes, featuring a deep software stack and connected by possibility asynchronous (best effort) networks. These systems are so complex that it is not known if one can model their behaviour with enough accuracy to predict their properties (performance, fault tolerance, security, QoS) without realizing actual experimentations. Thus observations of real Grid, experimentations with real conditions, phenomena isolation and behaviours understanding are certainly important steps towards accurate models. In that perspective, experimental testbed are fundamental methodological tools, allowing experimentation and observation of large scale phenomena in Grid and their applications.

The Grid 5000 project aims at building and developing a nation wide highly reconfigurable experimental testbed allowing a large variety of experiments on all the different layers of the software stack between the users and the hardware. Grid 5000 seeks to ease and support experimentations and to provide rigorous control and measurement mechanisms.

In its current state, this instrument for Grid researchers is built gathering the resources of 8 computers centres (Grid 5000 sites), connected by RENATER (the French national network for research and education), offering to the users thousands of CPUs. Each sites host a PC clusters providing from 256 to 1000 nodes (CPUs). The Grid 5000 control and provisioning environment allows to configure and install a full software stack on each Grid 5000 cluster nodes. This will give the users the unique capability to setup the exact software environment required for his experiment. Thus, the user may specify the OS, network protocols, middleware, runtimes, application and more generally all components of the software stack needed for his experiment. In addition to this configuration capabilities, Grid 5000 will offer a set of tools controlling the experimental conditions during the execution of the experiment. Basically, the user will be able to start and stop every Grid 5000 nodes, on demand.

Grid 5000 is a multi-institutions project, gathering funding from the French Ministry of research, INRIA, CNRS, University and several regional councils. The direction of the project is ensured by a Steering Committee (SC) involving the director of the ACI Grid, Thierry Priol, The director of the ACI Grid scientific committee, Brigitte Plateau, the director of RENATER, Dany Vandromme and all leaders of Grid 5000 sites. The project is implemented by a team of engineers belonging to the technical committee (TC). More than hundred researchers (permanents and Ph. D. students) will use this instrument involving about 50 engineers (10 at full time).

Regarding the INRIA, Grid'5000 is a collaborative effort of several INRIA projects (by alphabetical order): Apache, Caiman, Grand-Large, Oasis, Paris, Remap, Reso, Runtime, Scalaplix. Thus several Research Units are involved (by alphabetical order): Futurs, Rennes, Rhone Alpes and Sophia.

The role of Grand-Large in Grid 5000 is first the direction of the project, providing a vision, chairing the Steering Committee, preparing roadmaps and decisions to be discussed by the SC, preparing the SC meetings, etc. Second, Grand-Large, chairs the Technical Committee in charge of implementing the decisions of the SC and giving information to the SC to help the decision process. By being a central member in the SC and TC, Grand-Large plays a major role in Grid 5000.

One Expert Engineer funded by the INRIA is associated with this project at Orsay for its every day configuration and maintenance. Several Engineers from IDRIS and LRI have participated to the original design of Grid 5000 and help the Expert Engineer.

6.4.2. Grid eXplorer

Large scale distributed systems like P2P systems, Sensor Networks and Desktop Grids exhibit complex behaviours, difficult to understand because they fundamentally gather a large set of volatile nodes connected by an asynchronous network. Most of the well known techniques in distributed systems for fault tolerance do not work at this scale because their complexity is too high, they do not accept fault during some stabilisation phase or because the system is evolving in size too rapidly and too strongly. Like for the Grid, large scale distributed systems are complex to model and require prior experimentations and observations

To offer a respond to this challenge, the Grid eXplorer project aims at providing a large scale distributed system emulator. It consists first in building the emulator gathering hardware and software components and

developing the unavailable software. In term of hardware, the project seeks to install a 1000 CPUs cluster using a non blocking Ethernet network as well as a non blocking high speed network for a subset of the cluster. As software, the emulation environment will allow users to configure all layers of the software stack for every experiment. In addition to this feature, shared with Grid 5000, Grid eXplorer will provide network emulators, virtualization mechanisms as well as fault injectors. The second part of the project is to address a variety of large scale distributed system issues by experimenting actual applications, distributed systems, OS and network protocols and testing new ones.

Grid eXplorer complements Grid 5000 by providing an experimental environment where the user has the capacity to control the network experimental conditions.

This project is supported by several funding sources: the ministry of research through the ACI *Masses de données* (Data Grid Explorer), INRIA, CNRS and the *Ile de France* regional council. The total budget of this project is about 2 Millions Euros.

Regarding the INRIA, Grid eXplorer is a collaborative effort of several INRIA projects (by alphabetical order): Apache, Grand-Large, Regal and Reso. Thus several Research Units are involved (by alphabetical order): Futurs, Rhone Alpes and Rocquencourt.

Grand-Large co-initiated this project, leads the ACI *Masse de données* project and managed the initial procurement as well as the hosting of the cluster, actually installed in the IDRIS laboratory at Orsay. One engineer is associated with this project at Orsay for its every day configuration and maintenance. This engineer is funded by the ACI *Masse de données*. Several Engineers from IDRIS and LRI have participated to the original design of Grid Explorer and help the Expert Engineer.

7. Other Grants and Activities

7.1. Regional, National and International Actions

- ACI Data Mass Grid eXplorer, 3 years, head: F. Cappello, chair: Serge Petiton
- Specific Action of CNRS enabling Grid5000, 1 year, F. Cappello
- Global Computing: Augernome XtremWeb, Multi-Disciplinary Project (University of Paris XI), 4 years, sub-projet chair: Franck Cappello
- CNRS-Urbana Champaign Collaboration Program, 1 year, F. Cappello. Visit of Geraud Krawezik at Urbana in November 2003.
- ACI GRID CGP2P: Global Peer to Peer Computing, 3 years, head: F. Cappello
- ACI GRID 2. head: Jean Louis Pazat, sub-topic chair: F. Cappello, Serge Petiton
- ACI DataGraal. head: Pierre Sens, sub-topic chair: F. Cappello
- Specific Action of CNRS "Analyse Structurelle and Algorithmique des Reseaux Dynamiques" (DYNAMO), 1 year, head: P. Fraigniaud, Serge Petiton
- INRIA Associated Team "F-J Grid" with University of Tsukuba, 1 year, head: Franck Cappello
- ACI "GRID'5000", 3 years, head: Franck Cappello.
- CIFRE EADS, 3 years, (still in discussion), head: Franck Cappello.
- INRIA funding, MPI-V, collaboration with UTK, LALN and ANL, head: Franck Cappello
- Sakura program with University of Tsukuba, 1 year, head: Gilles Fedak
- Regional Council "Grid eXplorer", 1 year, co-chair: Franck Cappello
- Mobicoop (Agents mobiles cooperatifs pour la recherche dinformations dans des reseaux non fiables) CNRS JemSTIC action, 2 years, head: S. Tixeuil.
- STAR (Stabilisation des reseaux fondes sur la technologie Internet), CNRS JemSTIC action, 2 years, head: S. Tixeuil.
- ACI Sécurité FRAGILE, 3 years, head : S. Tixeuil, P. Fraigniaud
- ACI Sécurité SR2I, 3 years, subproject chair: S. Tixeuil
- P2P Project of ACI "Masse de Donnees" : P. Fraigniaud

7.2. Industrial Contacts

- CEA DAM, between CESTA Bordeaux and LIFL, 8 months, started in August 2002, head: Serge Petiton
- GIE EADS, Thesis founding (CIFRE) for Mathieu Caragnelli, from November 2004, 3 years. Title: Grid Services for semantics.

8. Dissemination

8.1. Services to the Scientific Community

- HPDC 13 program committee at Chicago, Franck Cappello, March 18, 2004,
- PaCT, Parallel Computing Technologies: program committee Franck Cappello, 2005.
- SC 04, ACM/IEEE SuperComputing 2004, Conference on High Performance Computing, Networking and Storage: program committee Franck Cappello, 2004.
- HiPC, ACM/IEEE International Conference on High Performance Computing: program committee Franck Cappello, 2004.
- GP2PC 2004 workshop organisation in conjunction with IEEE/ACM CCGRID 2004, Franck Cappello, Chicago, 2004
- GP2PC 2004 workshop program committee member Serge Petiton, Chicago March 2004,
- HeterPar program committee member Serge Petiton, Cork Ireland July 2004,
- INRIA-Japon meeting, Rocquencourt, Franck Cappello, Mai 26, Rocquencourt, 2004,
- Participation to HPC ASIA and NAREGI Workshop to prepare the French delegation on Grid to Japan, Franck Cappello, Tokyo, July 2004,.
- Discussions Titech researchers to prepare the French delegation on Grid to Japan, Franck Cappello, Tokyo, July 2004
- Discussions ITBL (JAERI) researchers to prepare the French delegation on Grid to Japan, Franck Cappello, Tokyo, July 2004
- INRIA meeting with JSPS director with Gilles Kan, at Roquencourt, Franck Cappello, September 3, Roquencourt, 2004,
- Meeting to prepare the visit of the French delegation to Japon with Ken Miura at Paris, Franck Cappello, September 30, Paris, 2004,
- French Grid Researcher delegation in Japon, Franck Cappello (preparation and management, December 10 to 16, Tokyo, 2004
- Serge Petiton represents the STIC/CNRS for the work group "calcul scientifique" of the ministry of research.
- Serge Petiton represents the STIC for the ORAP scientific council.
- HCW, IEEE Heterogeneous Computing Workshop:Franck Cappello, 2005. EGC, European Grid Conference:Franck Cappello, 2005.

8.2. Participation to Workshops, Seminars and Miscellaneous Invitations

- Presentation of Grand-Large and ACI CGP2P activities, France/UK Workshop on Grid at London, Franck Cappello,, November 3, London, 2003,
- presentation of Grid 5000, France Japan workshop on Grid at Paris, Franck Cappello, March 8, Paris, 2004,
- Invited presentation of XtremWeb, Geret symposium at Nancy on P2P systems, Franck Cappello, Mars 25, Nancy, 2004,
- invited seminar of Grid 5000 update, « RTP réseau » meeting at Carcassonne, Franck Cappello, Carcassonne, April 9 2004,
- « MPICH-V project», invited seminar at LIP, Franck Cappello, Mai 18, Lyon, 2004
- presentation of « Desktop Grid », Spring School Druide 2004, Franck Cappello, Mai 23, La colle sur loup, 2004
- Invited presentation « Grids and networking», « Internet Nouvelle Génération » summer school, Franck Cappello with Pascale Primet, June 14, Nancy, 2004,
- presentation of XtremWeb and Grid 5000 projects, Grid-Use 2004 workshop, Franck Cappello, Gilles Fedak and Tangui Morlier, June 21 to 25, Nancy, 2004.
- Participation to HPC ASIA and NAREGI Workshop to prepare the French delegation on Grid to Japan, Franck Cappello, Tokyo, July 2004,.
- « Grid Research in France »,Invited seminar at NAREGI, Discussions NAREGI researchers to prepare the French delegation on Grid to Japan, Franck Cappello, Tsukuba, July 2004.
- « Grid 5000 and Grid eXplorer», Invited seminar at AIST: Discussions AIST researchers to prepare the French delegation on Grid to Japan, Franck Cappello, Tsukuba, July 2004
- « Out-of-core preemptive scheduling of MPI applications for the Grid », Invited presentation, Scheduling Workshop, Franck Cappello, August 28 to 30, Aussois, 2004,
- Presentation of Grid 5000, Meeting with Vice president Reseach of Microsoft at Roquencourt with Gilles Kan, Franck Cappello, September 22, Roquencourt, 2004,
- « The MPICH-V project », invited presentation, Workshop CCGDS (Dongarra) at Lyon, Franck Cappello, September 26 to 29, Lyon, 2004,
- Meeting to prepare the visit of the French delegation to Japon with Ken Miura at Paris, Franck Cappello, September 30, Paris, 2004,
- « Grid 5000 », Invited Intech seminar, Inria Sophia Antipolis, Franck Cappello, October 21, 2004,
- Reims University invited Seminar, « Grid 5000 », Franck Cappello, Octobre 28, Sophia Antipolis, 2004.
- Invited presentation of « MPICH-V project » and « Grid 5000 », Dagstulh Seminar, Franck Cappello, September 29 to November 2, Dagstulh, 2004.
- « MPICH-V, Grid eXplorer, Grid 5000 », Invited seminar at ICL, University of Tennessee at Knoxville, USA, November 5, 2004.
- French Grid Researcher delegation in Japon, Franck Cappello (preparation and management, December 10 to 16, Tokyo, 2004
- presentation of « MPICH-V », Spring School Druide 2004, Thomas Hérault, Mai 25, La colle sur loup, 2004
- presentation of « RPC-V », Dagstulh Seminar, Thomas Hérault, September 29 to November 2, Dagstulh, 2004.
- Nahid Emad and Serge Petiton. A Dynamic Approcah to Solve a Large non-Hermitian Eigenproblem, Applied Mathematic Colloquium, Colombia University [42], New-York, NY, USA, April 13, 2004.
- Serge Petiton. Parallel Matrix Computing on P2P Platforms, GRID Colloquium [43], University of Tsukuba, Japon, September 27, 2004.

9. Bibliography

Major publications by the team in recent years

[1] G. BOSILCA, A. BOUTEILLIER, F. CAPPELLO, S. DJILALI, G. FEDAK, C. GERMAIN, T. HERAULT, P. LEMARINIER, O. LODYGENSKY, F. MAGNIETTE, V. NERI, A. SELHIKOV. *MPICH-V: Toward a Scalable Fault Tolerant MPI for Volatile Nodes*, in "proceedings of ACM/IEEE International Conference on Supercomputing", 2002.

[2] A. BOUTEILLER, G. KRAWEZIK, P. LEMARINIER, F. CAPPELLO. *MPICH-V3: A hierarchical fault tolerant MPI for Multi-Cluster Grids*, IEEE/ACM SC 2003, Phoenix USA, November 2003.

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

[4] 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.

[5] 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° 3 & 4, 2002, p. 327–340.

[6] O. LODYGENSKY, G. FEDAK, F. CAPPELLO, V. NERI, M. LIVNY, D. THAIN. *XtremWeb and Condor : sharing resources between Internet connected Condor pools*, in "GP2PC 2003 Workshop, Tokyo, Japan", IEEE/ACM CCGRID2003, May 12-15 2003.

Articles in referred journals and book chapters

[7] J. BEAUQUIER, L. PILARD, B. ROZOY. *Observing locally self-stabilization*, in "Journal of High Speed networks", to appear, 2004.

[8] I. BIERMANN, L. ROSAZ, B. ROZOY. *Occurence preserving congruences and associated graphs*, in "Rairo Informatique théorique et applications", to appear, 2004.

[9] A. BOUTEILLER, P. LEMARINIER, G. KRAWEZIK, F. CAPPELLO. *Coordinated checkpoint versus message log for fault tolerant MPI*, in "Int. Journal of High Performance Computing and Networking (IJHPCN)", n° 3, September 2004.

[10] 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.

[11] Y. CHEN, A. K. DATTA, S. TIXEUIL. *Stabilizing Inter-domain Routing in the Internet*, in "Journal of High Speed Networks", to appear, 2004.

[12] A. K. DATTA, M. GRADINARIU, S. TIXEUIL. *Self-stabilizing Mutual Exclusion with Arbitrary Scheduler*, in "The Computer Journal", vol. 47, n° 3, 2004, p. 289-298.

[13] K. DIKS, P. FRAIGNIAUD, E. KRANAKIS, A. PELC. *Tree exploration with little memory*, in "Journal of Algorithms", vol. 51, n° 1, 2004, p. 38–63.

[14] P. FRAIGNIAUD. *A Note on Line Broadcast in Digraphs under the Edge-Disjoint Paths Mode*, in "Discrete Applied Mathematics", vol. 144, 2004, p. 320–323.

[15] P. FRAIGNIAUD, C. GAVOILLE. *Header-Size Lower Bounds for End-to-End Communication in Memoryless Networks*, in "Computer Network", to appear, 2004.

[16] P. FRAIGNIAUD, D. ILCINKAS, G. PEER, A. PELC, D. PELEG. *Graph exploration by a finite automaton*, in "Theoretical Computer Science", to appear, 2004.

[17] P. FRAIGNIAUD, B. MANS, A. ROSENBERG. *Efficient Trigger-Broadcasting in Heterogeneous Clusters*, in "Journal on Parallel and Distributed Computing", to appear, 2004.

[18] C. JOHNEN, F. PETIT, S. TIXEUIL. *Auto-stabilisation et Protocoles Réseaux*, in "Technique et Science Informatiques", vol. 23, n° 8, 2004, p. 1027-1056.

[19] F. MAGNIETTE, L. PILARD, B. ROZOY. *A method for the verification of distributed and synchronized algorithm*, in "International Journal of Production Research", to appear, 2004.

[20] S. PETITON, L. AOUAD. *Large Scale Peer to Peer Performances Evaluations with Gauss-Jordan Methods as an Example*, in "Computer Science 3019", 2004.

Publications in Conferences and Workshops

[21] A. BOUTEILLER, H.-L. BOUZIANE, P. LEMARINIER, T. HERAULT, F. CAPPELLO. *Hybrid Preemptive Scheduling of MPI Applications on the Grids*, in "5th IEEE/ACM International Workshop on Grid Computing (Grid 2004), Pittsburgh USA", IEEE CS Press/ACM, November 2004.

[22] O. DELANNOY, S. PETITON. *A Peer to Peer Computing Framework ; design and Performance Evaluation of YML*, in "HeteroPpar 2004, Cork, Irlande", 2004 2004.

[23] S. DJILALI, T. HERAULT, O. LODYGENSKY, T. MORLIER, F. CAPPELLO. *RPC-V: Toward Fault-Tolerant RPC for Grids with Volatile Nodes*, in "IEEE/ACM SC 2004, Pittsburg, USA", 2004.

[24] P. DUCHON, N. HANUSSE, S. TIXEUIL. *Optimal Randomized Self-stabilizing Mutual Exclusion in Synchronous Rings*, in "Proceedings of the 18th Symposium on Distributed Computing (DISC(2004), Amsterdam, The Nederlands", Lecture Notes in Computer Science, n° 3274, Springer Verlag, October 2004, p. 216-229.

[25] P. DUCHON, N. HANUSSE, S. TIXEUIL. *Protocoles auto-stabilisants synchrones d'exclusion mutuelle pour les anneaux anonymes*, in "Proceedings of Rencontres Francophones sur l'Algorithmique des Communications (Algotel'2004), Batz sur Mer, France", ENST Bretagne, 2004.

[26] P. FRAIGNIAUD, L. GASIENIEC, D. KOWALSKI, A. PELC. *Collective Tree Exploration*, in "6th Latin American Theoretical Informatics Symposium (LATIN), Buenos Aires", April 2004.

[27] P. FRAIGNIAUD, C. GAVOILLE, C. PAUL. *Eclecticism Shrinks Even Small Worlds*, in "proceedings of the 23rd ACM Symp. on Principles of Distributed Computing (PODC)", To appear, 2004.

[28] P. FRAIGNIAUD, D. ILCINKAS. *Digraphs Exploration with Little Memory*, in "21st Symposium on Theoretical Aspects of Computer Science (STACS), Montpellier", March 2004.

[29] P. FRAIGNIAUD, D. ILCINKAS, G. PEER, A. PELC, D. PELEG. *Graph Exploration by a Finite Automaton*, in "29th Symposium on Mathematical Foundations of Computer Science (MFCS)", to appear, 2004.

[30] H. HE, G. BERGÈRE, S. PETITON. *A Parallel Asynchronous Hybrid Method to Accelerate Convergence of a Linear System*, in "International Symposium on Distributed Computing and Application to Business, Engineering and Science, Wuhan, China", September 2004.

[31] 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° 3121, Springer-Verlag, July 2004, p. 45-58.

[32] T. HERMAN, S. TIXEUIL. *Un Algorithme TDMA Réparti pour les réseaux de capteurs*, in "Proceedings of Rencontres Francophones sur l'Algorithmique des Communications (Algotel'2004), Batz sur Mer, France", ENST Bretagne, 2004.

[33] B. HUDZIA, S. PETITON. *Reliable multicast fault tolerant MPI in the Grid environment*, in "International Conference GRIDnet, San Jose, CA, USA", October 2004.

[34] P. LEMARINIER, A. BOUTEILLER, T. HERAULT, G. KRAWEZIK, F. CAPPELLO. *Improved Message Logging versus Improved Coordinated Checkpointing for fault tolerant MPI*, in "Proceedings of the 6th international conference on Cluster Computing (CLUSTER'04), San Diego, USA", IEEE, September 2004.

[35] P. LEMARINIER, B. COLLIN, A. BOUTEILLER, T. HERAULT, F. CAPPELLO. *Impact of Event Logger on Causal Message Logging Protocols for Fault Tolerant MPI*, in "Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS 05)", to appear, 2005.

[36] S. PETITON, L. AOUAD, L. CHOY. *Parallel Matrix Method Experimentations on Distributed P2P Platforms*, in "3th international workshop on Parallel Matrix Algorithms and Applications, Luminy, France", October 2004.

[37] S. PETITON, L. AOUAD. *Experimentations and Programming Paradigms for Matrix Computing on Peer to Peer Grid*, in "proceedings of the Grid2004 Workshop, Pittsuburgh, PA, USA", November 2004.

[38] A. SEDRAKIAN, S. PETITON, N. EMAD. *Restartable Tight Coupled Algorithms for SMP Cluster Architectures*, in "Parallel and Distributed Computing and Systems (PDCS), Camnbridge, MA, USA", November 2004.

Internal Reports

[39] J. BEAUQUIER, L. PILARD, B. ROZOY. *Observing locally self-stabilization*, Technical report, n° 1387, University of Paris Sud XI, May 2004.

[40] P. DUCHON, N. HANUSSE, S. TIXEUIL. *Optimal self-stabilizing mutual exclusion on synchronous rings*, Technical report, n° 1389, University of Paris Sud XI, Laboratoire de Recherche en Informatique, June 2004.

[41] N. MITTON, E. FLEURY, I. GUÉRIN-LASSOUS, S. TIXEUIL. *Self-stabilization in Self-organized Wireless Multi-hop Networks*, Technical report, INRIA, December 2004, http://www.inria.fr/rrrt/rr-5426.html.

Miscellaneous

[42] N. EMAD, S. PETITON. *A Dynamic Approcah to Solve a Large non-Hermitian Eigenproblem*, Applied Mathematic Colloquium, Colombia University, April 13 2004.

[43] S. PETITON. *Parallel Matrix Computing on P2P Platforms*, GRID Colloquium, September 27 2004.

Bibliography in notes

[44] N. A.LYNCH. M. KAUFMANN (editor). *Distributed Algorithms*, 1996.

[45] 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.

[46] 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° 6, June 1997, p. 535–553.

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

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

[49] 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.

[50] 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.

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

[52] 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.

[53] 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.

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

[55] 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.

[56] 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.

[57] 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.

[58] J. CAO, S. A. JARVIS, S. SAINI, G. 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.

[59] 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.

[60] 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. C. S. PRESS (editor)., 2000, p. 349-363.

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

[62] 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.*.

[63] A. CHIEN, B. CALDER, S. ELBERT, K. BHATIA. *Entropia: Architecture and Performance of an Enterprise Desktop Grid System*, in "Journal of Parallel and Distributed Computing", vol. 63, n° 5, 2003, p. 597–610.

[64] 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° 11, November 1997, p. 1139–1160.

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

[66] D. W. ERWIN. *UNICORE - a Grid computing environment. Concurrency and Computation: Practice and Experience 14(13-15): 1395-1410 (2002).*

[67] 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.

[68] M. J. FISCHER, N. A. LYNCH, M. S. PATERSON. *Impossibility of Distributed Consensus with one Faulty Process*, in "Journal of the ACM", vol. 32, n° 2, April 1985, p. 374–382.

[69] 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.

[70] I. FOSTER, C. KESSELMAN. *Globus: A metacomputing infrastructure toolkit, Internat. J. Supercomput. Appl. 11, 2 (1997), 115#128.*.

[71] I. FOSTER, C. KESSELMAN, J. NICK, S. TUECKE. *The physiology of the grid: An open grid services architecture for distributed systems integration. Technical report, Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.*.

[72] V. K. GARG. *Principles of distributed computing. John Wiley and Sons; ISBN: 0471036005; (May 2002).*.

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

[74] 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° 9, 1998, p. 929–961.

[75] 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.

[76] 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.

[77] 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.

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

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

[80] 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.

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

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

[83] N. MINAR, R. MURKHART, C. LANGTON, M. ASKENAZI. *The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations*, 1996, http://www.santafe.edu/projects/swarm/overview/overview.html.

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

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

[86] 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° 1, 1999, p. 31-53.

[87] 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.

[88] 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° 5–6, 1999, p. 675–686.

[89] 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.

[90] 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° 3, March 1982.

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

[92] 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.

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

[94] S. TUECKE, K. CZAJKOWSKI, I. FOSTER, S. G. J. FREY, C. KESSELMAN. *Grid Service Specification. Draft 3, Global Grid Forum, July 2002..*

[95] 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° 385, ETH Zurich, Institute for Comutersystems, October 2002.

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

[97] 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.*

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