INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team grand-large

Calcul parallèle et distribué à grande échelle

Futurs

THEME NUM

Table of contents

1. Team

Head of the project-team

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

Topic leaders

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

Administrative assistant

Gina Grisvard [Administrative assistant INRIA-Futurs]

Permanent Members

Joffroy Beauquier [Professor at Paris-Sud University, HdR]
Franck Cappello [Research Director at INRIA-Futurs, HdR]
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, HdR]
Brigitte Rozoy [Professor at Paris-Sud University]
Sébastien Tixeuil [Assistant Professor at Paris-Sud University, HdR]

Non Permanent Position

Lamine Aouad [Teaching Assistant at Lille 1 University]
Samir Djilali [Teaching Assistant at Paris-Sud University]
Haiwu He [PostDoc INRIA]
Pierre Lemarinier [PostDoc INRIA]
Derrick Kondo [PostDoc INRIA]
Olivier Soyez [Teaching Assistant at Lille 1 University]

Ph. D. student

Ali Asim [LRI)]
Fatiha Bouabache [MESR Grant (LRI)]
Matthieu Cargnelli [EADS Industrial Grant (CIFRE)]
Laurent Choy [INRIA et Région Nord (LIFL)]
Camille Coti [INRIA(LRI)]
Philippe Gauron [MESR Grant (LRI)]
Toussaint Guglielmi [MESR Grant (LIFL)]
William Hoarau [MESR Grant (LRI)]
Benoit Hudzia [Franco-Irish Grant (LIFL)]
Yongjie Hu [Franco-Chine (LRI)]
David Ilcinkas [MESR Grant (LRI)]
Oleg Lodygensky [LaL Engineer (Laboratoire de l'Accelerateur Lineaire)]
Paul Malecot [INRIA Project Grant]
Nicolas Nisse [MNRT (LRI)]
Olivier Peres [MESR Grant (LRI)]
Benjamin Quetier [MESR Grant (LRI)]
Ala Rezmerita [MESR Grant (LRI)]
Baohua Wei [Industrial Chinese Grant (LRI)]
Ye Zhang [Industrial Chinese Grant]

Research scientist (partner)

Pierre Fraigniaud [Research Director at CNRS, HdR]
Rosaz Laurent [Assistant Professor at Paris Sud University]

Project technical staff

Julien Leduc [INRIA Exper Engineer]

Phillipe Marty [[INRIA Expert Engineer]
Vincent Neri [[CNRS Study Engineer]
Eric Rodriguez [[INRIA Associate Engineer]
Pierre Neyron [[INRIA Expert 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 [88] 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 [112] 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 [101] , Glunix [94] and Mosix [67] , to cite a few. A first approach to cross administration domains was proposed by Web Computing projects such as Jet [104] , Charlotte [68] , Javeline [84] , Bayanihan [109] , SuperWeb [64] , ParaWeb [74] and PopCorn [77]. 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 [76] , BOINC [66] and XtremWeb [87].

The high performance potential of LSDS platforms has also raised a significant interest in the industry. Companies like Entropia [83] , United Devices [118] , Platform [105] , Grid systems [123] and Datasynapse [122] 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 [89] , the authors present the similarities and differences between Grid and Global Computing systems. Two important distinguishing parameters are the user community (professional or not) and the resource ownership (who own the resources and who is using them). From the system architecture perspective, we consider two main differences: the system scale

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

3.1.2. Building a Large Scale Distributed System for Computing

This set of studies considers the XtremWeb project as the basis for research, development and experimentation. This LSDS middleware is already operational. This set gathers 4 studies aiming at improving the mechanisms and enlarging the functionalities of LSDS dedicated to computing. The first study considers the architecture of the resource discovery engine which, in principle, is close to an indexing system. The second study concerns the storage and movements of data between the participants of a LSDS. In the third study, we 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 [110] , Gnutella [110] and Kazaa [100] , 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 [106] , CHORD [114] , PASTRY [108] and TAPESTRY [121] 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 [66] 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 [122] 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 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 [97] and Ocean [82]. 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 [80] , [78]. Recently some researches consider scheduling and migrating MPI applications on Grid [113]. Other related researches concern scheduling for cycle stealing environments [107]. 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 [103] 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 [102] , 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 [81]. 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 [119] , pessimistic [65] , causal [120]. 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 [71] , MPICH-V2 [72] , MPICH-V/CL [73]. 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 ([92]). 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. [62] [115] [85] 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. [99] [98] , [69] , [70] , [93]) 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 [116]. 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 [111] , [96]. 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.

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. Methodology

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 [79] GridSim [75] , Briks [63]. 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 [103] 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 [90] and Unicore [86]. Nevertheless we can observe some convergence elements between LSDS and Grid. The paper [89] gives many details about the similarities and differences between P2P and Grid systems. From the technological perspective, the evolution of Globus to GT3 [91] 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 [91] and OGSI [117]. 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 [124]. 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) 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 [95]. 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.

5.2. BitDew

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

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

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

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

5.3. SimBOINC

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

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

5.4. XtremLab

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

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

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

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

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

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

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

5.5. MPICH-V

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

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

In addition to fault tolerant properties, MPICH-V:

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

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

5.6. 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 YvetteML. 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 C++ 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 and the OmniRPC grid computing software.

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.

5.7. The Scientific Programming InterNet (SPIN)

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

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

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

5.8. V-Grid

This project 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. We have made a virtual patform on the GDX cluster with the Vserver virtualization tool. On this platform, we have launched more than 20K virtual machines (VM) with a folding of 100 (100 VM on each physical marchine). However, some recent experiments have shown that a too high folding factor may cause a too long execution time because of some problems like swapping. Currently, we are conducting experiments on another platform based on the virtualization tool named Xen which has been strongly improved since 2 years. We expect to get better result with Xen than with Vserver.

5.9. PVC: Private Virtual Cluster

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

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

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

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

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

5.10. OpenWP

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

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

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

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

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

This environment is under construction. A prototype version is running.

5.11. FAult Injection Language (FAIL)

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

6. New Results

6.1. Large Scale Distributed Systems

6.1.1. Fault Tolerant MPICH-V

In [25] we developed a new implementation of a blocking coordinated checkpointing protocol named MPICH-Pcl. We implemented it inside the current MPICH2 distribution and inside the new Nemesis communication

driver of MPICH2. We presented the comparison of MPICH-Pcl with our precedent non-blocking coordinated checkpointing protocol implementation named MPICH-Vcl during SC06. We demonstrated that the blocking implementation outperforms the non blocking one in the context of high performance network. We also demonstrated, using 1 to 512 nodes that the synchronization of every process during the checkpoint phase does not allow this kind of protocol to scale with the increase of the number of nodes. To address efficiently high performance network clusters, we integrated our blocking coordinated checkpointing protocol to the new Nemesis communication driver. With this implementation we can rely on the network card driver directly without using the socket emulation driver.

6.1.2. Programming the Grid

In [13], [49] we developed a framework for a parallel programming model by remote procedure calls bridging between large scale computing resource pools managed by multiple gridenabled job scheduling systems. With this system, the user can exploit not only each remote servers and clusters, but also computing resources provided with grid-enabled job scheduling systems located on different sites. This framework requires a Grid RPC system to decouple the computation in a remote node from the Grid RPC mechanism and uses document-based communication rather than connection-based communication. We implemented the proposed framework as an extension of the OmniRPC system, which is a Grid RPC system for parallel programming in a grid environment. We designed a general interface to adapt the OmniRPC system to various grid-enabled job scheduling systems easily and applied the proposed system to several grid-enabled job scheduling systems, including XtremWeb, CyberGRIP, Condor and Grid Engine. We show the preliminary performance of these implementations using a phylogenetic application. We found that the proposed system can achieve approximately the same performance as using OmniRPC and can handle interruptions in worker programs on remote nodes.

6.1.3. Private Virtual Cluster

Given current complexity of Grid technologies, the lack of security of P2P systems and the rigidity of VPN technologies make sharing resources belonging to different institutions still technically difficult. In [14] we propose a new approach called "Instant Grid" (IG), which combines various Grid, P2P and VPN approaches, allowing simple deployment of applications over different administration domains. Three main requirements should be fulfilled to make Instant Grids realistic: 1) simple networking configuration (Firewall and NAT), 2) no degradation of resource security and 3) no need to re-implement existing distributed applications. In this paper, we present Private Virtual Cluster, a low-level middleware that meets these three requirements. To demonstrate its properties, we have connected with PVC a set of firewall-protected PCs and conducted experiments to evaluate the networking performance and the capability to execute unmodified MPI applications.

6.1.4. V-Grid: a Large Scale Emulator

Virtualization tools are becoming popular in the context of Grid Computing because they allow running multiple operating systems on a single host and provide a confined execution environment. In several Grid projects, virtualization tools are envisioned to run many virtual machines per host. This immediately raises the issue of virtualization scalability. In this paper, we compare the scalability merits of 4 virtualization tools. First, from a simple experiment, we motivate the need for simple microbenchmarks. Second, we present a set of metrics and related methodologies. We propose 4 microbenchmarks to measure the different scalability parameters for the different machine resources (CPU, memory disk and network) on 3 scalability metrics (overhead, linearity and isolation). Third, we compare 4 virtual machine technologies (Vserver, Xen, UML and VMware). The results of this study demonstrate that all the compared tools present different behaviors with respect to scalability, in terms of overhead, resource occupation and isolation. Thus this work will help user to select tools according to their application characteristics

We have made a virtual patform on the GDX cluster with the Vserver virtualization tool. On this platform, we have been able to launch more than 20K virtual machines (VM) with a folding of 100 (100 VM on each physical marchine). However, some recent experiments have shown that a too big repliment factor may cause a

bad execution time because of some problems like swapping. Indeed, when you launch 100 VM on a physical machine, all the ressources are divided by 100 (memory, CPU time, bandwith, ...), so even a little application will use 100 times more memory and can made the physical machine to swap. We have also tested a bittorrent transfert on this platform and we have observed that with a 100 folding factor, the execution time have not been multiply by 100 as expected but by more than 100. It show that the continuous context changing between virtual machines cause very bad performances on hard disk cache.

6.1.5. Characterizing Intranet and Internet Desktop Grids

Desktop grids, which use the idle cycles of many desktop PC's, are currently the largest distributed systems in the world. Despite the popularity and success of many desktop grid projects, the heterogeneity and volatility of hosts within desktop grids has been poorly understood. Yet, host characterization is essential for accurate simulation and modelling of such platforms. In [12], [40] we gathered application-level traces of four real desktop grids that can be used for simulation and modelling purposes. In addition, we determined aggregate and per host statistics that reflect the heterogeneity and volatility of desktop grid resources.

Also, in [16], we studied the potential capacity of volunteer computing resource on Internet desktop grids. We analyzed measurements of about 200,000 hosts participating in a volunteer computing project. These measurements include processing power, memory, disk space, and network throughput, as well as host usage factors, user-specified limits on resource usage, and host churn. We showed that volunteer computing can support applications that are data-intensive or have high RAM or storage requirements.

6.1.6. Scheduling on Volatile and Heterogeneous Resources

Because desktop computing resources are volatile, achieving performance guarantees such as task completion rate is difficult. In [41], we investigated the use of buffering to ensure task completion rates, which is essential for soft real-time applications. In particular, we developed a model of task completion rate as a function of buffer size. We instantiated this model using parameters derived from two enterprise desktop grid data sets, evaluated the model via trace-driven simulation, and showed how this model can be used to ensure application task completion rates on enterprise desktop grid systems.

6.2. Large Scale Peer to Peer Performance Evaluations

In [28], we propose a framework dedicated to the development and the execution of parallel applications over large scale global computing platforms. A workflow programming environment is based on a new workflow language YvetteML and a HumanGRID middleware interface called YML. This language allows description of different kind of components to be allocated to GRID resources. Depending of the different targeted resources, the components may be associated to computation, data migration or other resource controls. YML is designed to have several backends for different middleware , as a welldesigned front end is developed independently of any dedicated middleware. In order to make the framework immediately useful, YML comes with preconfigure interfaces to some numerical routines and a numerical library for iterative linear algebra methods.

In [51], we have presented the integration of a multi-level scheduler in the YML architecture. It demonstrates the advantages of this architecture based on a component model and why it is well suited to develop parallel applications for Grids. Then, the under development multi-level scheduler for this framework have been presented.

Parallelizing and distributing the real symmetric eigenproblem has been extensively studied for supercomputers and highperformance clusters. Global computing introduces new constraints and we must propose new algorithms adapted to this kind of environment. In [42], we propose an explicitly restarted Lanczos algorithm combined with the Bisection and the inverse iteration methods on GRID and large clusters. The Lanczos method is very interesting when it is not possible to store the matrix in the main central memory because the components are accessed by means of matrixvector products. The success of the Lanczos algorithm is also due to its restarted scheme which limits the memory usage. The bisection algorithm, and the inverse

iteration, are particularly well suited global computing because they allow implementing the parametric parallelism paradigm also known as task-farming. We proposed our very first experimental results and analyze the effects of several parameters like the order of the Krylov subspace, the number of computed eigenpairs and computational nodes.

In [17], we presented a performance evaluation of large scale matrix algebra applications on the Grid'5000 platform. We test the scalability of the experimental tool and some optimization techniques for large scale matrix algebra applications in grid infrastructures based on an efficient data locality, already presented for non dedicated grid platforms. This includes persistent data placement and explicit management of local memories on the computational nodes. We discussed the performances of a block-based matrix-vector product and the Gauss-Jordan method for large matrix inversion. As experimental grid middleware we use the XtremWeb system to manage the Grid'5000 computational resources. We also compare these results with those obtained on large non-dedicated computational platforms distributed on two geographic sites in France and Japan. We showed the effectiveness of the presented data placement techniques but that some constraints and limitations on the experimentation and underlying tools made scalability and realistic expectations more difficult.

In [33], we presented a classical parallel method GMRES(m) to solve large sparse linear systems utilizing a lightweight GRID system XtremWeb. GMRES(m) is one of the key methods to resolve large, non-symmetric, linear problems on this system. We discussed as well the performances of this implementation deployed on two XtremWeb networks: a local network with 128 non-dedicated PCs in Polytech-Lille of University of Sciences and Technologies of Lille in France, a remote network with 3 clusters of SCGN Grid including 91 CPUs totally in the High Performance Computing Center of University of Tsukuba in Japan. We also did the tests on the platform of supercomputer IBM SP4 of CINES in Montpellier in France. We compared the performances on the two different computing systems. The advantages and drawbacks of our implementations on this GRID computing system XtremWeb have been well explained.

6.3. Volatility and Reliability Processing

6.3.1. Self-stabilization

Stabilizing distributed systems expect all the component processes to run predefined programs that are externally mandated. In Internet scale systems, this is unrealistic, since each process may have selfish interests and motives related to maximizing its own payoff. [27] formulates the problem of selfish stabilization that shows how competition blends with cooperation in a stabilizing environment.

Also, we specify the conflict manager abstraction. Informally, a conflict manager guarantees that any two neighboring nodes can not enter their critical simultaneously (safety), and that at least one node is able to execute its critical section (progress). The conflict manager [55] problem is strictly weaker than the classical local mutual exclusion problem, where any node that requests to enter its critical section eventually does so (fairness). We argue that conflict managers are a useful mechanism to transform a large class of self-stabilizing algorithms that operate in an essentially sequential model, into self-stabilizing algorithm that operate in a completely asynchronous distributed model. We provide two implementations (one deterministic and one probabilistic) of our abstraction, and provide a composition mechanism to obtain a generic transformer. Our transformers have low overhead: the deterministic transformer requires one memory bit, and guarantees time overhead in order of the network degree, the probabilistic transformer does not require extra memory. While the probabilistic algorithm performs in anonymous networks, it only provides probabilistic stabilization guarantees. In contrast, the deterministic transformer requires initial symmetry breaking but preserves the original algorithm guarantees.

Also, in [26] we generalized the classic dining philosophers problem to separate the conflict and communication neighbors of each process. Communication neighbors may directly exchange information while conflict neighbors compete for the access to the exclusive critical section of code. This generalization is motivated by a number of practical problems in distributed systems including problems in wireless sensor networks. We present a self-stabilizing deterministic algorithm - KDP that solves a restricted version of the generalized problem where the conflict set for each process is limited to its k-hop neighborhood. Our algorithm is terminating.

We formally prove KDP correct and evaluate its performance. We then extend KDP to handle fully generalized problem. We further extend it to handle a similarly generalized drinking philosophers problem. We describe how KDP can be implemented in wireless sensor networks and demonstrate that this implementation does not jeopardize its correctness or termination properties.

A 1-adaptive self-stabilizing system is a self-stabilizing system that can correct any memory corruption of a single process in one computation step. 1-adaptivity means that if in a legitimate state the memory of a single process is corrupted, then the next system transition will lead to a legitimate state and the system will recover a correct behavior. Thus 1-adaptive self-stabilizing algorithms guarantee the very strong property that a single fault is corrected immediately and consequently that it cannot be propagated. Our aim in [19] is to study necessary and sufficient conditions to obtain that property in order to design such algorithms. In particular we show that this property can be obtained even under the distributed demon and that it can also be applied to probabilistic algorithms. We provide two self-stabilizing 1-adaptive algorithms that demonstrate how the conditions we present here can be used to design and prove 1-adaptive algorithms.

We present in [9] a generic distributed algorithm for solving silents tasks such as shortest path calculus, depth-first-search tree construction, best reliable transmitters, in directed networks where communication may be only unidirectional. Our solution is written for the asynchronous message passing communication model, and tolerates multiple kinds of failures (transient and intermittent). First, our algorithm is self-stabilizing, so that it recovers correct behavior after finite time starting from an arbitrary global state caused by a transient fault. Second, it tolerates fair message loss, finite message duplication, and arbitrary message reordering, during both the stabilizing phase and the stabilized phase. This second property is most interesting since, in the context of unidirectional networks, there exists no self-stabilizing reliable data-link protocol. A formal proof establishes its correctness for the considered problem, and subsumes previous proofs for solutions in the simpler reliable shared memory communication model.

6.3.2. Byzantine containment and resilience

As a new challenge of containing the unbounded influence of Byzantine processes in self-stabilizing protocols, we introduced a novel concept of strong stabilization. The strong stabilization relaxes the requirement of strict stabilization so that processes beyond the containment radius are allowed to be disturbed by Byzantine processes, but only a limited number of times. A self-stabilizing protocol is (t,c,f)-strongly stabilizing [46] if any process more than c hops away from any Byzantine process is disturbed at most t times in a distributed system with at most f Byzantine processes. Here c denotes the containment radius and t denotes the containment times. The possibility and the effectiveness of the strong stabilization is demonstrated using tree orientation. It is known that the tree orientation has no strictly stabilizing protocol with a constant containment radius. We first showed that the problem has no constant bound of the containment radius in a tree with two Byzantine processes even when we allow processes beyond the containment radius to be disturbed any finite number of times. Then we consider the case of a single Byzantine process and present a (1,0,1)-strongly stabilizing protocol, which achieves optimality in both containment radius and times.

Also, we study in [50] the problem of Byzantine-robust topology discovery in an arbitrary asynchronous network. We formally state the weak and strong versions of the problem. The weak version requires that either each node discovers the topology of the network or at least one node detects the presence of a faulty node. The strong version requires that each node discovers the topology regardless of faults. We focus on non-cryptographic solutions to these problems. We explore their bounds. We prove that the weak topology discovery problem is solvable only if the connectivity of the network exceeds the number of faults in the system. Similarly, we show that the strong version of the problem is solvable only if the network connectivity is more than twice the number of faults. We present solutions to both versions of the problem. Our solutions match the established graph connectivity bounds. The programs are terminating, they do not require the individual nodes to know either the diameter or the size of the network. The message complexity of both programs is low polynomial with respect to the network size.

6.3.3. Sensor Networks

We quantify in [46] the amount of "practical" information (i.e. views obtained from the neighbors, colors attributed to the nodes and links) to obtain "theoretical" information (i.e. the local topology of the network up to distance k) in anonymous networks. In more details, we show that a coloring at distance 2k+1 is necessary and sufficient to obtain the local topology at distance k that includes outgoing links. This bound drops to 2k when outgoing links are not needed. A second contribution deals with color bootstrapping (from which local topology can be obtained using the aforementioned mechanisms). On the negative side, we show that (i) with a distributed daemon, it is impossible to achieve deterministic color bootstrap, even if the whole network topology can be instantaneously obtained, and (ii) with a central daemon, it is impossible to achieve distance m when instantaneous topology knowledge is limited to m-1. On the positive side, we show that (i) under the k-central daemon, deterministic self-stabilizing bootstrap of colors up to distance k is possible provided that k-local topology can be instantaneously obtained, and (ii) under the distributed daemon, probabilistic self-stabilizing bootstrap is possible for any range.

Also, we present in [24] an analysis of a MAC (Medium Access Control) protocol for wireless sensor networks. The purpose of this protocol is to manage wireless media access by constructing a Time Division Media Access (TDMA) schedule. APMC (Approximate Probabilistic Model Checker) is a tool that uses approximation-based verification techniques in order to analyse the behavior of complex probabilistic systems. Using APMC, we approximately computed the probabilities of several properties of the MAC protocol being studied, thus giving some insights about it performance.

The advent of large scale multi-hop wireless networks highlights problems of fault tolerance and scale in distributed system, motivating designs that autonomously recover from transient faults and spontaneous reconfiguration. Self-stabilization provides an elegant solution for recovering from such faults. We present complexity analysis for a family of self-stabilizing vertex coloring algorithms in the context of multi-hop wireless networks. Such coloring processes are used in several protocols for solving many different issues (clustering, synchronizing...). Overall, our results [48] show that the actual stabilization time is much smaller than the upper bound provided by previous studies. Similarly, the height of the induced DAG is much lower than the linear dependency on the size of the color domain (that was previously announced). Finally, it appears that symmetry breaking tricks traditionally used to expedite stabilization are in fact harmful when used in networks that are not tightly synchronized.

6.3.4. Benchmarking Fault-tolerance

We reviewed in [58] several existing tools for fault injection and dependability benchmarking in grids. We emphasis on the FAIL-FCI fault-injection software that has been developed in INRIA Grand Large, and a benchmark tool called QUAKE that has been developed in the University of Coimbra. We present the state-of-the-art and we explain the importance of these tools for dependability assessment of Grid-based applications and Grid middleware.

One important contribution to the community that is developing Grid middleware is the definition and implementation of benchmarks and tools to assess the performance and dependability of Grid applications and the corresponding middleware. In [59], we present an experimental study that was conducted with OGSA-DAI, a popular package of middleware that provides access to remote data resources thought a unified Web-service front-end. The results show that OGSA-DAI is quite stable and performed quite well in scalability tests, executed on Grid5000. However, we also demonstrate that OGSA-DAI WSI is currently using a SOAP container (Apache Axis1.2.1) that suffers from severe memory leaks. We show how the default configuration of OGSA-DAI is not affected by that problem, but a small change in the configuration of a Web-service may lead to very unreliable execution of OGSA-DAI.

In a network consisting of several thousands computers, the occurrence of faults is unavoidable. Being able to test the behavior of a distributed program in an environment where we can control the faults (such as the crash of a process) is an important feature that matters in the deployment of reliable programs. In [39], we extend FAIL-FCI (for Fault Injection Language, and FAIL Cluster Implementation, respectively), a software tool that permits to elaborate complex fault scenarios in a simple way, while relieving the user from writing low level code. In particular, we show that not only we are able to fault-load existing distributed applications (as used in

most current papers that address fault-tolerance issues), we are also able to inject qualitative faults, i.e. inject specific faults at very specific moments in the program code of the application under test. Finally, and although this was not the primary purpose of the tool, we are also able to inject specific patterns of workload, in order to stress test the application under test. Interestingly enough, the whole process is driven by a simple unified description language, that is totally independent from the language of the application, so that no code changes or recompilation are needed on the application side. Also, we investigate the possibility of injecting software faults in distributed java applications. Our scheme is by extending the FAIL-FCI software. It does not require any modification of the source code of the application under test, while retaining the possibility to write high level fault scenarios. As a proof of concept, we use our tool to test FreePastry, an existing java implementation of a Distributed Hash Table (DHT), against node failures.

One of the topics of paramount importance in the development of Cluster and Grid middleware is the impact of faults since their occurrence probability in a Grid infrastructure and in large-scale distributed system is actually very high. MPI (Message Passing Interface) is a popular abstraction for programming distributed computation applications. FAIL is an abstract language for fault occurrence description capable of expressing complex and realistic fault scenarios. In [36], we investigate the possibility of using FAIL to inject faults in a fault-tolerant MPI implementation. Our middleware, FAIL-MPI, is used to carry quantitative and qualitative faults and stress testing.

6.3.5. Mobile agents

We consider in [10] the task of exploring graphs with anonymous nodes by a team of non-cooperative robots, modeled as finite automata. For exploration to be completed, each edge of the graph has to be traversed by at least one robot. In this paper, the robots have no a priori knowledge of the topology of the graph, nor of its size, and we are interested in the amount of memory the robots need to accomplish exploration, We introduce the so-called reduced automata technique, and we show how to use this technique for deriving several space lower bounds for exploration. Informally speaking, the reduced automata technique consists in reducing a robot to a simpler form that preserves its "core" behavior on some graphs. Using this technique, we first show that any set of q>= 1 non-cooperative robots, requires $\Omega(log(n/q))$ memory bits to explore all n-node graphs. The proof implies that, for any set of q K-state robots, there exists a graph of size O(qK) that no robot of this set can explore, which improves the O(KO(q)) bound by Rollik (1980). Our main result is an application of this latter result, concerning terminating graph exploration with one robot, i.e., in which the robot is requested to stop after completing exploration. For this task, the robot is provided with a pebble, that it can use to mark nodes (without such a marker, even terminating exploration of cycles cannot be achieved). We prove that terminating exploration requires $\Omega(logn)$ bits of memory for a robot achieving this task in all n-node graphs.

6.4. Peer-to-peer systems design

In [29], we study the problem of the amount of knowledge about a communication network that must be given to its nodes in order to efficiently disseminate information. While previous results used particular partial information available to nodes (such as neighborhood knowledge), our approach is quantitative: we investigate the minimum total number of bits of information (minimum oracle size) that has to be available to nodes in order to perform efficient communication. It turns out that the minimum oracle size can serve as a measure of the difficulty of this task. Using this, we showed that an efficient wakeup requires strictly more information about the network than an efficient broadcast.

Kleinberg showed how to augment meshes using random edges, so that they become navigable; that is, greedy routing computes paths of polylogarithmic expected length between any pairs of nodes. Is such an augmentation is possible for all graphs? In [30], we answer negatively to this question by exhibiting a threshold on the doubling dimension, above which an infinite family of graphs cannot be augmented to become navigable. To complete our result, we prove that the special case of square meshes are always be augmentable to become navigable.

[32] Search games are attractive for their correspondence with classical width parameters. For instance, the invisible search number of a graph is equal to its pathwidth plus 1, and the visible search number of a graph is equal to its treewidth plus 1. The connected variants of these games ask for search strategies that are connected, i.e., at every step of the strategy, the searched part of the graph induces a connected subgraph. We focus on monotone search strategies, i.e., strategies for which every node is searched exactly once. It is known that the monotone connected visible search number of an n-node graph is at most O(log n) times its visible search number. First, we prove that this logarithmic bound is tight. Second, we prove that, as opposed to the non-connected variant of visible graph searching, recontamination helps for connected visible search.

In [31], we give a constructive proof of the equality between treewidth and connected treewidth, which finds applications in graph searching problems. More precisely, we describe an $O(nk^3)$-time algorithm that, given any n-node width-k tree-decomposition of a connected graph G, returns a connected tree-decomposition of G of width $\leq k$.

[22] addresses the graph searching problem in a distributed setting. We describe a distributed protocol that enables searchers with logarithmic size memory to clear any network, in a fully decentral- ized manner. The search strategy for the network in which the searchers are launched is computed online by the searchers themselves without knowing the topology of the network in advance. It performs in an asynchronous environment, i.e., it implements the necessary synchronization mechanism in a decentralized manner. In every network, our protocol performs a connected strategy using at most $k + 1$ searchers, where k is the minimum number of searchers required to clear the network in a monotone connected way, computed in the centralized and synchronous setting.

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
- 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, 2 years, head: Gilles Fedak
- Regional Council "Grid eXplorer", 1 year, co-chair: Franck Cappello
- ACI Sécurité FRAGILE, 3 years (2004-2007), head: S. Tixeuil
- ACI Sécurité SR2I, 3 years (2004-2007), subproject chair: S. Tixeuil
- P2P Project of ACI “Masse de Donnees”: P. Fraigniaud
- ANR Jeunes chercheurs XtremLab : G. Fedak
- ANR Masses de Données ALPAGE, 3 years (2005-2008), sub-topic chair: P. Fraigniaud

- AURORA project France-Norway 1 year, "Self-stabilization and Sensor Networks", chair: S. Tixeuil
- European CoreGrid Network of Excellence, subtask head: S. Tixeuil, sub-project heads: G. Fedak, T. Herault, S. Tixeuil
- European STREP (6th FP pri5-IST). Quasi-Opportunistic Supercomputing for Complex Systems in Grid Environments (QosCosGrid), managment board representative: Franck Cappello, task head: Thomas Herault
- European project. Grid4All, managment board representative: Franck Cappello, task head: Gilles Fedak

7.2. Industrial Contacts

- 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

8.1.1. Book/Journal edition

- Franck Cappello, Mitsuhisa Sato, Adriana Iamnitchi, special issue on "Global and Peer-to-Peer Computing", JoGC, Journal of Grid Computing, 2006

8.1.2. Conference Organisation

- Franck Cappello, HPDC'2006, "High Performance Distributed Computing", Paris, June 19-23, 2006

8.1.3. Editorial Committee membership

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

8.1.4. Steering Committee membership

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

8.1.5. Program Committee membership

- Franck Cappello, Workshop on XEN in High-Performance Cluster and Grid Computing Environments as part of The Fourth International Symposium on Parallel and Distributed Processing and Applications (ISPA'2006). Sorrento, Italy
- Franck Cappello, SC'2006 – International Conference and Supercomputing and Networking, Tempa, USA, November 11-17, 2006.
- Franck Cappello, IPDPS'2006 – 20th Annual IEEE International Parallel and Distributed Processing Symposium, Rhodes, Grece, April 3-6, 2006.
- Franck Cappello, HCW 2006 – 14th Heterogeneous Computing Workshop, Rodes Island, Greece, April 25-29, 2006
- Franck Cappello, VECPAR'2006 –7th International Meeting High Performance Computing for Computational Science, Rio de Janeiro, Brazil, July 10-12, 2006
- Sébastien Tixeuil, ICDCS'2006 – 26th IEEE International Conference on Distributed Computing Systems, Lisboa, Portugal, July 4-7, 2006
- Franck Cappello, HotP2P'06, Hot Topic in P2P System, Greece – 2006
- Sébastien Tixeuil, Algotel 2006 – 2006
- Sébastien Tixeuil, HPDC 2006, Poster Chair – 2006
- Sébastien Tixeuil, DISC 2006, Stockholm Sweden, September 18-20 – 2006.
- Sébastien Tixeuil, SSS 2006, Dallas, texas, November 17-19 – 2006.
- Sébastien Tixeuil, SWAN 2006, Bangalore, India, December – 2006.
- Franck Cappello, GP2PC'06, Singapor, April 2006
- Franck Cappello, ECG'2006, European Grid Conference, June 7-8, 2006.
- Franck Cappello, ICCP'2006, THE 2006 INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, Columbus, Ohio, USA, August 14-18, 2006.
- Franck Cappello, Grid'2006, Barcelona, Spain, September 2006
- P. Fraigniaud, 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rhodes Island, Greece, 25-29 April, 2006. http://www.ipdps.org/
- P. Fraigniaud, 26th International Conference on Distributed Computing Systems (ICDCS), Lisboa, Portugal, July 4-7, 2006. http://icdcs2006.di.fc.ul.pt/
- P. Fraigniaud, 12th European Conference on Parallel Computing (Euro-Par), Dresden, Germany, Aug. 29 - Sept. 1, 2006. http://www.europar2006.de/
- P. Fraigniaud, 24th IASTED Conference on Parallel and Distributed Computing and Networks (PDCN), Innsbruck, Austria, February 14-16, 2006. http://www.iasted.org/conferences/2006/Innsbruck/pdcn.htm
- P. Fraigniaud, 13th Colloquium on Structural Information and Communication Complexity (SIROCCO), Chester, UK, July 3-5, 2006. http://sirocco06.csc.liv.ac.uk/
- Derrick Kondo GP2PC'2006, "Global and Peer to Peer Computing", in association with CCGRID'2006, Singapore, 16-19 May 2006.
- Serge Petiton GP2PC'2006, "Global and Peer to Peer Computing", in association with CCGRID'2006, Singapore, 16-19 May 2006.
- Gilles Fedak GP2PC'2006, "Global and Peer to Peer Computing", in association with CCGRID'2006, Singapore, 16-19 May 2006.
- Joffroy Beauquier, SSS 2006, November 20O6.
- Joffroy Beauquier, OPODIS 2006, December 20O6.
- Gilles Fedak, Rencontres du parallélisme Renpar, Perpignan 2006

8.1.6. School and Workshop organization

- Sébastien Tixeuil Ecotel 2006 (Ecole d'hiver des télécommunications), program comittee co-chair
- Sébastien Tixeuil, Second Coregrid Workshop on Grid and P2P System Architecture, Paris, 16-17 january 2006.
- Gilles Fedak , GP2PC'2006, "Global and Peer to Peer Computing", in association with CCGRID'2006, Singapore, 16-19 May 2006.

8.1.7. Session Chairing

- Sébastien Tixeuil, Session 1: Fault Tolerance, Second CoreGRID workshop on P2P and System Architectures, Paris, France, 16-17 January, 2006.
- Sébastien Tixeuil, Session 4, DISC, Stockholm, Sweden, September 18-20, 2006.
- Sébastien Tixeuil, Session Sensor Networks, SSS, Dallas, Texas, USA, November 17-19, 2006.

8.2. Participation to Workshops, Seminars and Miscellaneous Invitations

8.2.1. Invited International Conference

- Franck Cappello "Grid'5000, motivations, status and early results", Grid@Asia workshop, Seoul, Corea, December 13, 2006
- Franck Cappello "When Scale Reactivates Research in Distributed Computing: Grid'5000, Instant Grid and MPI-V", STIC-Amsud meeting, Santiago, Chile, October 18-20, 2006
- Franck Cappello "Grid'5000, motivations, status and early results", Workshop of the Grille Academic Tunisienne pour la Recherche Scientifique, Tunis, Tunisia, Oct., 2006
- Franck Cappello "Grid'5000, motivations, status and early results", HPC Conference, Cetraro, July, 2006
- Franck Cappello "An Update of Grid5000 and a Focus on a Fault Tolerant MPI Experiemnt", Clusters and Computational Grids for Scientific Computing 2006, Highland Lake Inn, Ashville USA September, 2006
- Franck Cappello "Grid and Utility Computing: Do they really mean Pervasive Services?", ICPS 2006 panel session, June 2006
- Franck Cappello "Grid 5000: the need for experimental platform for Grid research", ExpGrid Workshop Panel, Paris, June 2006.
- Franck Cappello "Grid'5000, motivations, status and early results", Workshop new trends in HPDC, Amsterdam, March, 2006

8.2.2. Invited National Conference

- Franck Cappello "Grid'5000, motivation, état et résultats récdents", journée Notere, Toulouse, Juin 2006.
- Franck Cappello "P2P et dépendance", journée JSSI, Paris, Mai 2006.
- Franck Cappello "Grid'5000, motivation, état et résultats récdents", journée GrilBio, Lyon, Mai 2006.
- Franck Cappello "Grid'5000, motivation, état et résultats récdents", Ecole Grid'5000, Grenoble, Mars 2006.

8.2.3. Schools, Workshops

- Franck Cappello, "Les Grilles : les défits du calcul distribuée", Alcatel Marcoussis seminar, November 9, 2006.

- P. Fraigniaud, "Graph exploration and graph searching", Descrete Mathematics Summer School, Valparaíso, Chili, 9-13 Jan, 2006.
- P. Fraigniaud, "Aspects fondamentaux des réseaux décentralisés", Ecole de printemps GRID et P2P, Crans-Montana, Suisse, 6-10 mars 2006.
- P. Malécot, "Les Grilles de PC: expériences avec BOINC et XtremWeb", présentation invitée, JTR 2006: "Infrastructure de grille, aspects systèmes et réseaux", Lyon, 16-18 octobre 2006

8.2.4. Seminaries

- Derrick Kondo, "Dependability on Desktop Grids", Coregrid meeting, Nice, France, November 2006.
- Derrick Kondo, "Resource Availability in Enterprise Desktop Grids", Second CoreGrid Workshop on Grid and Peer-to-Peer Systems Architecture, Paris, France, January 2006.
- Derrick Kondo, "Towards Soft Real-Time Applications on Enterprise Desktop Grids", Second CoreGrid Workshop on Grid and Peer-to-Peer Systems Architecture, Paris, France, January 2006.
- Gilles Fedak, "Scheduling in Desktop Grids", Grid4all Consortium meeting, Paris, Décembre 2006
- Gilles Fedak, "Efficient Data Distribution with BitTorrent for Computational Desktop Grids", Université de Tsukuba, Octobre 2006
- Gilles Fedak et Derrick Kondo, "Desktop Grid middleware and scheduling for the Grid4all project", Grid4all kickoff meeting, Stockholm, Juin 2006
- G. Fedak et P. Domingues (UCO), "Dependability mechanisms for Desktop Grid Computing", CoreGrid Integration Workshop, Paris, Janvier 2006
- Sébastien Tixeuil, "Toward Self-stabilizing Large Scale Systems", Kent State University Colloquium, 15 Novembre 2006.
- Sébastien Tixeuil, "Toward Self-stabilizing Large Scale Systems", ENS Cachan, 17 Octobre 2006.
- Camille Coti, "MPICH-PcL vc MPICH-VcL", "Journées data grix eXplorer", IDRIS, Orsay, 13 octobre 2006

9. Bibliography

Major publications by the team in recent years

[1] 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 "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^{o} 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.

Year Publications

Articles in refereed journals and book chapters

[7] J. BEAUQUIER, L. PILARD, B. ROZOY. *Observing locally self-stabilization in a probabilistic way*, in "Journal of Aerospace Computing, Information, and Communication (JACIC)", 2006, to appear.

[8] R. BOLZE, ET AL.. *Grid5000: A Large Scale Highly Reconfigurable Experimental Grid Testbed*, in "International Journal on High Peerformance Computing and Applications", 2006.

[9] S. DELAËT, B. DUCOURTHIAL, S. TIXEUIL. *Self-stabilization with r-operators revisited*, in "Journal of Aerospace Computing, Information, and Communication", 2006.

[10] P. FRAIGNIAUD, D. ILCINKAS, S. RAJSBAUM, S. TIXEUIL. *Shimon Even Festschrift*, Lecture Notes in Computer Science, chap. The reduced automata technique for graph exploration space lower bounds, n^{o} 3895, Springer-Verlag Berlin Heidelberg, 2006, p. 1-26.

[11] W. HOARAU, L. SILVA, S. TIXEUIL. *Integrated Research in Grid Computing*, CoreGRID, chap. Fault-injection and Dependability Benchmarking for GRID COmputing Middleware, Springer Verlag, 2006.

[12] D. KONDO, G. FEDAK, F. CAPPELLO, A. CHIEN, H. CASANOVA. *Resource Availability in Enterprise Desktop Grids*, in "Future Generation Computer Systems", 2006.

[13] Y. NAKAJIMA, M. SATO, F. CAPPELLO. *Integrating Computing Resources on Multiple Grid-enabled Job Scheduling Systems Through a Grid RPC System*, in "Journal of Grid Computing", 2006.

[14] B. QUETIER, V. NERI, F. CAPPELLO. *Scalability Comparison of Four Host Virtualization Tools*, in "Journal of Grid Computing", 2006.

[15] S. TIXEUIL. *Réseaux mobiles ad hoc et réseaux de capteurs sans fils*, Information, Commande, Communication, chap. Algorithmique répartie tolérante aux pannes dans les systèmes à grande échelle, Lavoisier, March 2006, p. 251-284.

Publications in Conferences and Workshops

[16] D. ANDERSON, G. FEDAK. *The Computational and Storage Potential of Volunteer Computing*, in "Proceedings of the 6th International Symposium on Cluster Computing and the Grid (CCGRID'06), Singapore, USA", May 2006.

[17] L. M. AOUAD, S. G. PETITON. *Parallel Basic Matrix Algebra on the Grid'5000 Large Scale Distributed Platform*, in "Cluster'06. The 2006 IEEE International Conference on Cluster Computing, Barcelona", September 25th-28th 2006.

[18] J. BEAUQUIER, S. DELAËT, S. HADDAD. *A 1-Strong Self-Stabilizing Transformer*, in "Eighth International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), Dallas, Texas", A. K. DATTA, M. GRADINARIU (editors). , Lecture Notes in Computer Science, November 2006.

[19] J. BEAUQUIER, S. DELAËT, S. HADDAD. *Necessary and Sufficient Conditions for 1-adaptivity*, in "Proceedings of IEEE International Conference on Paralle and Distributed Systems (IPDPS 2006)", April 2006.

[20] J. BEAUQUIER, C. JOHNEN, S. MESSIKA. *All k-bounded policies are equivalent for self-stabilization*, in "Eighth International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), Dallas, Texas", A. K. DATTA, M. GRADINARIU (editors). , Lecture Notes in Computer Science, November 2006, http://www.lri.fr/parall/publis/messika/sss06.pdf.

[21] J. BEAUQUIER, C. JOHNEN, S. MESSIKA. *Brief Announcement: Computing automatically the stabilization time against the worst and the best schedulers*, in "Proceedings of DISC'06, LNCS 4167", S. DOLEV (editor). , Springer Verlag, 2006, p. 443-447, http://www.lri.fr/parall/publis/messika/discbrief06.pdf.

[22] L. BLIN, P. FRAIGNIAUD, N. NISSE, S. VIAL. *Distributed Chasing of Network Intruders*, in "13th Colloquium on Structural Information and Communication Complexity (SIROCCO), Chester, UK", July 3 - 5 2006.

[23] M. CADILHAC, T. HERAULT, P. LEMARINIER. *Message Relaying Techniques for Computational Grids and their Relations to Fault Tolerant Message Passing for the Grid*, in "Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture, Paris, France", January 2006.

[24] M. CADILHAC, T. HÉRAULT, R. LASSAIGNE, S. PEYRONNET, S. TIXEUIL. *Evaluating complex MAC protocols for sensor networks with APMC*, in "Proceedings of AVOCS 2006, Nancy", September 2006.

[25] C. COTI, T. HERAULT, P. LEMARINIER, L. PILARD, A. REZMERITA, E. RODRIGUEZ, F. CAPPELLO. *Blocking vs. Non-Blocking Coordinated Checkpointing for Large-Scale Fault Tolerant MPI*, in "Int. Conf. for High Performance Computing, Networking, Storage and Analysis (SC2006), Tampa, USA", IEEE/ACM, November 2006.

[26] P. DANTURI, M. NESTERENKO, S. TIXEUIL. *Self-stabilizing Philosophers with Generic Conflicts*, in "Eighth International Symposium on Stabilization, Safety, and Security on Distributed Systems (SSS 2006), Dallas, Texas", A. K. DATTA, M. GRADINARIU (editors). , Lecture Notes in Computer Science, Springer Verlag, November 2006, to appear.

[27] A. DASGUPTA, S. GHOSH, S. TIXEUIL. *Selfish Stabilization*, in "Eighth International Symposium on Stabilization, Safety, and Security on Distributed Systems (SSS 2006), Dallas, Texas", A. K. DATTA, M. GRADINARIU (editors). , Lecture Notes in Computer Science, Springer Verlag, November 2006, to appear.

[28] O. DELANNOY, N. EMAD, S. PETITON. *Workflow Global Computing with YML*, in "GRID 2006, Barcelona", September 28th-29th 2006.

[29] P. FRAIGNIAUD, D. ILCINKAS, A. PELC. *Oracle size: a new measure of difficulty for communication tasks*, in "25th ACM Symposium on Principles Of Distributed Computing (PODC), Denver, Colorado, USA", July 23-26 2006.

[30] P. FRAIGNIAUD, E. LEBHAR, Z. LOTKER. *A Doubling Dimension Threshold Theta(loglog n) for Augmented Graph Navigability*, in "14th Annual European Symposium on Algorithms (ESA), Zurich, Switzerland", 11-13 September 2006.

[31] P. FRAIGNIAUD, N. NISSE. *Connected treewidth and connected graph searching*, in "7th Latin American Symposium (LATIN), LNCS3887", 2006, p. 479-490.

[32] P. FRAIGNIAUD, N. NISSE. *Monotony Properties of Connected Visible Graph Searching*, in "32nd Int. Workshop on Graph-Theoretic Concepts in Computer Science, Bergen, Norway", June 22-24 2006.

[33] H. HE, G. BERGERE, Z. WANG, S. PETITON. *Solving linear systems on global computing architecture with GMRES method*, in "International Conference on Distributed Computing and Applications for Business Engineering and Sciences, Hangzhou, China", October 12-15 2006.

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

[35] T. HERAULT, P. LEMARINIER, O. PERES, L. PILARD, J. BEAUQUIER. *Self-Stabilizing Spanning Tree Algorithm for Large Scale Systems (Brief Announcement)*, in "Eighth International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), Dallas, Texas", A. K. DATTA, M. GRADINARIU (editors). , Lecture Notes in Computer Science, November 2006.

[36] W. HOARAU, P. LEMARINIER, T. HERAULT, E. RODRIGUEZ, S. TIXEUIL, F. CAPPELLO. *FAIL-MPI: How fault-tolerant is fault-tolerant MPI?*, in "Proceedings of Cluster 2006, Barcelona, Spain", September 2006.

[37] W. HOARAU, S. TIXEUIL. *Easy fault injection and stress testing with FAIL-FCI*, in "Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture, Paris, France", January 2006.

[38] W. HOARAU, S. TIXEUIL, N. MORENO, D. SOUSA, L. SILVA. *Benchmarking the OGSA-DAI Middleware*, in "Second Coregrid Integration Workshop, Krakow, Poland", October 2006.

[39] W. HOARAU, S. TIXEUIL, F. VAUCHELLES. *Fault Injection in Distributed Java Applications*, in "International Workshop on Java for Parallel and Distributed Computing (joint with IPDPS 2006), Greece", IEEE, April 2006, to appear.

[40] D. KONDO, G. FEDAK, F. CAPPELLO, A. CHIEN, H. CASANOVA. *On Resource Volatility in Enterprise Desktop Grids*, in "Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing (eScience'06), Amsterdam, Netherlands", December 2006.

[41] D. KONDO, B. KINDARJI, G. FEDAK, F. CAPPELLO. *Towards Soft Real-Time Applications on Enterprise Desktop Grids*, in "Proceedings of the 6th IEEE Symposium on Cluster Computing and the Grid (CCGRID '06), Singapore", May 2006.

[42] C. LAURENT, S. G. PETITON, M. SATO. *Solving symmetric eigenproblems on grid and global computing environment*, in "4th International Workshop on Parallel Matrix Algorithms and Applications (PMAA'06),, IRISA, Rennes, France", September, 7th-9th 2006.

[43] P. MALÉCOT, D. KONDO, G. FEDAK. *XtremLab: A System for Characterizing Internet Desktop Grids*, in "Poster in International Symposium on High Performance Distributed Computing HPDC'06, Paris, France", June 2006.

[44] P. MALÉCOT, D. KONDO, G. FEDAK. *XtremLab: Une plateforme pour l'observation et la caractérisation des grilles de PC sur Internet*, in "Rencontres francophones du parallélisme (Renpar'06), Perpignan, France", October 2006.

[45] T. MASUZAWA, S. TIXEUIL. *Bounding the Impact of Unbounded Attacks in Stabilization*, in "Eighth International Symposium on Stabilization, Safety, and Security on Distributed Systems (SSS 2006), Dallas, Texas", A. K. DATTA, M. GRADINARIU (editors). , Lecture Notes in Computer Science, Springer Verlag, November 2006, to appear.

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

[47] N. MITTON, E. FLEURY, I. GUÉRIN-LASSOUS, B. SÉRICOLA, S. TIXEUIL. *Convergence dans les réseaux sans fil*, in "Proceedings of Algotel 2006", INRIA, May 2006, to appear.

[48] N. MITTON, E. FLEURY, I. GUÉRIN-LASSOUS, B. SÉRICOLA, S. TIXEUIL. *On Fast Randomized Colorings in Sensor Networks*, in "Proceedings of ICPADS 2006", IEEE Press, July 2006, to appear.

[49] Y. NAKAJIMA, M. SATO, Y. AIDA, T. BOKU, F. CAPPELLO. *Integrating Computing Resources on Multiple Grid-enabled Job Scheduling Systems Through a Grid RPC System*, in "CCGRID, Singapore", 2006.

[50] M. NESTERENKO, S. TIXEUIL. *Discovering Network Topology in the Presence of Byzantine Nodes*, in "Prooceedings of Sirocco 2006, Chester, UK", Springer Verlag, July 2006, to appear.

[51] S. NOEL, O. DELANNOY, N. EMAD, P. MANNEBACK, S. PETITON. *A multi-level scheduler for the Grid computing YML framework*, in "EuroPAR 2006 - CoreGRID Workshop, Dresden", August 28th-29th 2006.

[52] S. G. PETITON, L. M. AOUAD. *Distributed Out-of-Core Parallel Linear Algebra on Grid5000 Heterogeneous Platform*, in "12th siam conference on Parallel Processing for Scientific Computing, San Francisco, USA", February 21th-24th 2006.

[53] S. TIXEUIL, W. HOARAU, L. SILVA. *An Overview of Existing Tools for Fault-Injection and Dependability Benchmarking in Grids*, in "Second CoreGRID Workshop on Grid and Peer to Peer Systems Architecture, Paris, France", January 2006.

Internal Reports

[54] J. BEAUQUIER, C. JOHNEN, S. MESSIKA. *Computing automatically the stabilization time against the worst and the best schedulers*, Technical report, n^{o} 1448, Université Paris-Sud XI, May 2006, http://www.lri.fr/Rapports-internes/2006/RR1448.pdf.

[55] M. GRADINARIU, S. TIXEUIL. *Conflict Managers for Self-stabilization without Fairness Assumption*, Technical report, n^{o} 1459, LRI, Université Paris Sud, September 2006.

[56] T. HERAULT, W. HOARAU, P. LEMARINIER, E. RODRIGUEZ, S. TIXEUIL. *FAIL-MPI: How fault-tolerant is fault-tolerant MPI?*, Technical report, n^{o} 1450, Université Paris-Sud XI, June 2006.

[57] T. HERAULT, P. LEMARINIER, O. PERES, L. PILARD, J. BEAUQUIER. *Self-Stabilizing Spanning Tree Algorithm for Large Scale Systems*, Technical report, n^{o} 1457, LRI, 2006.

[58] W. HOARAU, L. SILVA, S. TIXEUIL. *An Overview of Existing Tools for Fault-Injection and Dependability Benchmarking in Grids*, Technical report, n^{o} 0041, CoreGRID, October 2006.

[59] W. HOARAU, S. TIXEUIL, N. RODRIGUES, D. SOUSA, L. SILVA. *Benchmarking the OGSA-DAI Middleware*, Technical report, n^{o} 0060, CoreGRID, October 2006.

[60] M. NESTERENKO, S. TIXEUIL. *Bounds on Topology Discovery in the Presence of Byzantine Faults*, Technical report, n^{o} TR-KSU-CS-2006-01, Dept. of Computer Science, Kent State University, 2006, http://www.cs.kent.edu/techreps/TR-KSU-CS-2006-01.pdf.

[61] S. TIXEUIL. *Vers l'Auto-stabilisation des Systèmes à Grande Echelle*, Habilitation à Diriger les Recherches, Université Paris-Sud XI, Orsay, France, May 2006.

References in notes

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

[63] 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.

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

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

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

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

[68] 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.

[69] 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.

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

[71] 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*.

[72] 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.

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

[74] 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.

[75] 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.

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

[77] 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.

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

[79] 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.

[80] 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.

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

[82] 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.*.

[83] 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^{o} 5, 2003, p. 597–610.

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

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

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

[87] 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.

[88] 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^{o} 2, April 1985, p. 374–382.

[89] 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.

[90] I. FOSTER, C. KESSELMAN. *Globus: A metacomputing infrastructure toolkit, Internat. J. Supercomput. Appl. 11, 2 (1997), 115128..*

[91] 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..*

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

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

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

[95] 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.

[96] 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.

[97] 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.

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

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

[100] 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.

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

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

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

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

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

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

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

[108] 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.

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

[110] 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.

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

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

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

[114] 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.

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

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

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

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

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

[120] 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.*

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

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

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

[124] WEBSERVICES. *WebServices.Org*, http://www.webservices.org/.