INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team grand-large

Calcul parallèle et distribué à grande échelle

Saclay - Île-de-France

THEME NUM

Activity Report

2008

Table of contents

1. Team

Research Scientist

Franck Cappello [Team Leader, Research Director, HdR]
Gilles Fedak [Junior Researcher, end September 2008]
Laura Grigori [Junior Researcher CR1]

Faculty Member

Joffroy Beauquier [Professor at Paris-Sud University, HdR]
Thomas Hérault [Assistant Professor at Paris Sud University]
Serge Petiton [Professor at University of Science and Technology of Lille, HdR]
Sylvain Peyronnet [Assistant Professor at Paris Sud University]
Brigitte Rozoy [Professor at Paris-Sud University, HdR]
Sébastien Tixeuil [Professor at Paris 6 University, HdR]

Technical Staff

Roberto Podesta [INRIA Expert Engineer]
Francois Berenger [INRIA Expert Engineer, end November 2008]
Víctor Iniesta [INRIA Expert Engineer]
Vincent Neri [CNRS Study Engineer]

PhD Student

Ali Asim [MESR Grant LRI]
Fatiha Bouabache [MESR Grant (LRI)]
Matthieu Cargnelli [EADS Industrial Grant (CIFRE), thesis December 18 2008]
Laurent Choy [INRIA et Région Nord (LIFL), end September 2007]
Camille Coti [INRIA Grant]
Toussaint Guglielmi [MESR Grant (LIFL), end November 2007]
William Hoarau [MESR Grant (LRI), Thesis Mars 21 2008]
Yongjie Hu [Franco-Chine (LRI)]
Pawan Kumar [INRIA Grant]
Thomas Largillier [MESR Grant (LRI)]
Paul Malécot [INRIA Grant]
Olivier Peres [MESR Grant (LRI), Thesis September 24 2008]
Niu Qiang [China Grant for 1 year visit, end October 2008]
Benjamin Quetier [MESR Grant (LRI), Thesis September 12 2008]
Ala Rezmerita [MESR Grant (LRI)]
Amina Guermouche [MESR Grant (LRI)]
Alexandre Borghi [MESR Grant (LRI)]
Simplice Donfack [INRIA Grant]
Ye Zhang
Ling Shang [French Ministry of Foreign Affairs]

Post-Doctoral Fellow

Alok Gupta [PostDoc]
Haiwu He [PostDoc, end September 2008]
Hua Xiang [PostDoc, end August 2008]

Administrative Assistant

Katia Evrat [Administrative assistant]

2. Overall Objectives

2.1. Grand-Large General Objectives

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

The project aims at:

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

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

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

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

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

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

3. Scientific Foundations

3.1. Large Scale Distributed Systems (LSDS)

What makes a fundamental difference between recent Global Computing systems (Seti@home), Grid (EGEE, TeraGrid) and former works on distributed systems is the large scale of these systems. This characteristic becomes also true for large scale parallel computers gathering tens of thousands of CPU cores. The notion of Large Scale is linked to a set of features that has to be taken into account in these systems. An example is the system dynamicity caused by node volatility: in Internet Computing Platforms (also called Desktop Grids), a non predictable number of nodes may leave the system at any time. Some recent results also report a very low MTTI (Mean Time To Interrupt) in top level supercomputers gathering 100,000+ CPU cores. Another example of characteristics is the complete lack of control of nodes connectivity. In Desktop Grid, we cannot assume that external administrator is able to intervene in the network setting of the nodes, especially their connection to Internet via NAT and Firewalls. This means that we have to deal with the in place infrastructure in terms of performance, heterogeneity, dynamicity and connectivity. These characteristics, associated with the requirement of scalability, establish a new research context in distributed systems. The Grand-Large project aims at investigating theoretically as well as experimentally the fundamental mechanisms of LSDS, especially for the high performance computing applications.

3.1.1. Computing on Large Scale Global Computing systems

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

Desktop Grid systems essentially extend the notion of computing beyond the frontier of administration domains. The very first paper discussing this type of systems [140] 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 [124] , Glunix [115] and Mosix [89], to cite a few. A first approach to cross administration domains was proposed by Web Computing projects such as Jet [128], Charlotte [90], Javeline [108], Bayanihan [136], SuperWeb [86], ParaWeb [96] and PopCorn [100]. These projects have emerged with Java, taking benefit of the virtual machine properties: high portability across heterogeneous hardware and OS, large diffusion of virtual machine in Web browsers and a strong security model associated with bytecode execution. Performance and functionality limitations are some of the fundamental motivations of the second generation of Global Computing systems like COSM [99], BOINC [88] and XtremWeb [111]. The second generation of Global Computing systems appeared in the form of generic middleware which allow scientists and programmers to design and set up their own distributed computing project. As a result, we have seen the emergence of large communities of volunteers and projects. Currently, Global Computing systems are among the largest distributed systems in the world. In the mean time, several studies succeeded to understand and enhance the performance of these systems, by characterizing the system resources in term of volatility and heterogeneity and by studying new scheduling heuristics to support new classes of applications: data-intensive, long running application with checkpoint, workflow, soft-real time etc... However, despite these recent progresses, one can note that Global Computing systems are not yet part of high performance solution, commonly used by scientists. Recent researches to fulfill the requirements of Desktop Grids for high demanding users aim at redesigning Desktop Grid middleware by essentially turning a set of volatile nodes into a virtual cluster and allowing the deployment of regular HPC utilities (batch schedulers, parallel communication libraries, checkpoint services, etc...) on top of this virtual cluster. The new generation would permit a better integration in the environment of the scientists such as computational Grids, and consequently, would broaden the usage of Desktop Grid.

The high performance potential of LSDS platforms has also raised a significant interest in the industry. Companies like United Devices [145], Platform [130], Grid systems [151] and Datasynapse [150] propose LSDS middleware for Desktop Grids (also known as PC Grid systems). Performance demanding users are also interested by these platforms, considering their cost-performance ratio which is even lower than the one of clusters. Thus, several Desktop Grid platforms are daily used in production in large companies in the domains of pharmacology, petroleum, aerospace, etc.

Desktop Grids share with Grid a common objective: to extend the size and accessibility of a computing infrastructure beyond the limit of a single administration domain. In [112], the authors present the similarities and differences between Grid and Global Computing systems. Two important distinguishing parameters are the user community (professional or not) and the resource ownership (who own the resources and who is using them). From the system architecture perspective, we consider two main differences: the system scale and the lack of control of the participating resources. These two aspects have many consequences, at least on the architecture of system components, the deployment methods, programming models, security (trust) and more generally on the theoretical properties achievable by the system.

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

3.1.2. Building a Large Scale Distributed System

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

3.1.2.1. The resource discovery engine

A multi-users/multi-applications LSDS for computing would be in principle very close to a P2P file sharing system such as Napster [137], Gnutella [137] and Kazaa [123], except that the shared resource is the CPUs instead of files. The scale and lack of control are common features of the two kinds of systems. Thus, it is likely that solutions sharing fundamental mechanisms will be adopted, such as lower level communication protocols, resource publishing, resource discovery and distributed coordination. As an example, recent P2P projects have proposed distributed indexing systems like CAN [133], CHORD [142], PASTRY [135] and TAPESTRY [149] that could be used for resource discovery in a LSDS dedicated to computing.

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

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

A traditional way to cope with transient failures are self-stabilizing systems [110]. Existing self-stabilizing algorithms use an underlying network that is not compatible with LSDS. They assume that processors know their list of neighbors, which does not fit the P2P requirements. Our work proposes a new model for designing self-stabilizing algorithms without making this assumption, then we design, prove and evaluate overlay networks self-stabilizing algorithms in this model.

3.1.2.2. Data storage and movement

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

In this scenario of data-centric applications, the existing Desktop Grid Systems face a scalability issue. One should expect that more computing resources also provides more network bandwidth and storage capacity. On the contrary, Desktop Grids Systems like BOINC or XtremWeb rely on a centralized data service architecture. For instance, data distribution with BOINC relies on multiple HTTP servers and tasks are described as a list of files locations, which can be a potential bottleneck when scheduling tasks sharing large input files. Recent developments in content distribution such as collaborative file distribution (BitTorrent [109], Slurpie [139], Digital Fountain [98]) and P2P file sharing applications (EDonkey/Overnet [84], Kazaa [123]), has proven to be both effective, viable and *self-scalable*. The key idea, often featured as *parallel download* in the P2P applications, is to divide the file in a list of chunks. Immediately after a peer downloads a chunk from another peer, the peer serves the block for the other peers in the network, thus behaving itself as a server. Collaborative Content Distribution is a very active research topic and several promising strategies [131] such as the use of network coding [116], are proposed to improve the performance of the system. We will evaluate the efficiency of collaborative data distribution mechanism according to its impact on the overall performance of a parallel application when scheduled on Desktop Grid.

Currently there is no LSDS system dedicated to computing that allows the persistent storage of data in the participating nodes. Several LSDS systems dedicated to data storage are emerging such as OCEAN Store [120] and Ocean [105]. Storing large data sets on volatile nodes requires replication techniques. In CAN and Freenet, the documents are stored in a single piece. In OceanStore, Fastrack and eDonkey, the participants store segments of documents. This allows segment replications and the simultaneous transfer of several documents segments. In the CGP2P project, a storage system called US has been proposed. It relies on the notion of blocs (well known in hard disc drivers). Redundancy techniques complement the mechanisms and provide raid like properties for fault tolerance. Moreover, we believe that the basic blocks for building a system dedicated to data management in Desktop Grids can be found in P2P systems. We will investigate new paradigm for LSDS data management based on P2P components such as Bittorrent for data distribution and DHT for efficient and scalable data index/search, and design a new middleware, BitDew, featuring data transfer reliability, volatility tolerance, automatic data replication, multi-protocol capability, data affinity, and tuple-space like semantic.

3.1.2.3. Scheduling in large scale systems

Scheduling is one of the system fundamental mechanisms. Several studies have been conducted in the context of Grid mostly considering bag of tasks, parameter sweep or workflow applications [103], [101]. Recently some researches consider scheduling and migrating MPI applications on Grid [141]. Other related researches concern scheduling for cycle stealing environments [134]. Some of these studies consider not only the dynamic CPU workload but also the network occupation and performance as basis for scheduling decisions. They often refer to NWS which is a fundamental component for discovering the dynamic parameters of a Grid. There are very few researches in the context of LSDS and no existing practical ways to measure the workload dynamics of each component of the system (NWS is not scalable).

While LSDS systems consist of volatile and heterogeneous computing resources, it is unknown exactly how volatile and heterogeneous these computing resources are. While there have been previous characterization studies on Internet-wide computing resources, these studies do not take into account causes of volatility and only report coarse aggregate statistics, such as the mean time to failure of resources. Yet, detailed resource characterization is essential for the simulation and modelling of LSDS systems in a research area where many results are obtained via simulation.

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

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

3.1.2.4. Fault Tolerant MPI

MPICH-V is a research effort with theoretical studies, experimental evaluations and pragmatic implementations aiming to provide a MPI implementation based on MPICH [126], 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 [104]. 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 [146], pessimistic [87], causal [147]. Several optimizations have been studied for the three strategies. The general context of our study is high performance computing on large platforms. One of the most used programming environments for such platforms is MPI.

Within the MPICH-V project, we have developed and published several original fault tolerant protocols for MPI: MPICH-V1 [93], MPICH-V2 [94], MPICH-Vcausal, MPICH-Vcl [95], MPICH-Pcl. The two first protocols rely on uncoordinated checkpointing associated with either remote pessimistic message logging or sender based pessimistic message logging. We have demonstrated that MPICH-V2 outperforms MPICH-V1. MPICH-Vcl implements a coordinated checkpoint strategy (Chandy-Lamport) removing the need of message logging. MPICH-V2 and Vcl are concurrent protocols for large clusters. We have compared them considering a new parameter for evaluating the merits of fault tolerant protocols: the impact of the fault frequency on the performance. We have demonstrated that the stress of the checkpoint server is the fundamental source of performance differences between the two techniques. MPICH-Vcausal implements a causal message logging protocols, removing the need for waiting acknowledgement in contrary to MPICH-V2. MPICH-Pcl is a blocking implementation of the Vcl protocol. Under the considered experimental conditions, message logging becomes more relevant than coordinated checkpoint when the fault frequency reaches 1 fault every 4 hours, for a cluster of 100 nodes sharing a single checkpoint server, considering a data set of 1 GB on each node and a 100 Mb/s network.

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

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

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

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

3.2. Volatility and Reliability Processing

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

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

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

These types of problems are not new ([113]). They have been studied deeply and many algorithmic solutions and implementations are available. What is new here and makes these old solutions not usable is scalability. Any solution involving centralization is impossible to use in practice. Previous works validated on former networks can not be reused.

3.2.1. Reliability Processing

We voluntarily presented in a separate section the volatility problem because of its specificity both with respect to type of failures and to frequency of failures. But in a general manner, as any distributed system, a global computing system has to resist to a large set of failures, from crash failures to Byzantine failures, that are related to incorrect software or even malicious actions (unfortunately, this hypothesis has to be considered as shown by DECRYPTHON project or the use of erroneous clients in SETI@HOME project), with in between, transient failures such as loss of message duplication. On the other hand, failures related accidental or malicious memory corruptions have to be considered because they are directly related to the very nature of the Internet. Traditionally, two approaches (masking and non-masking) have been used to deal with reliability problems. A masking solution hides the failures to the user, while a non-masking one may let the user notice that failures occur. Here again, there exists a large literature on the subject (cf. [125], [143], [110] for surveys). Masking techniques, generally based on consensus, are not scalable because they systematically use generalized broadcasting. The self-stabilizing approach (a non-masking solution) is well adapted (specifically its time adaptive version, cf. [122], [121], [91], [92], [114]) for three main reasons:

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

We propose:

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

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

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

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

3.3.1. Large Scale Computational Sciences and Engineering

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

GRID scientific and engineering computing exists already since more than a decade. Since the last few years, the scale of the problem sizes and the global complexity of the applications increase rapidly [144]. The scientific simulation approach is now general in many scientific domains, in addition to theoretical and experimental aspects, often link to more classic methods. Several applications would be computed on world-spread networks of heterogeneous computers using some web-based Application Server Provider (ASP) dedicated to targeted scientific domains. New very strategic domains, such as Nanotechnologies, Climatology or Life Sciences, are in the forefront of these applications. The development in this very important domain and the leadership in many scientific domains will depend in a close future to the ability to experiment very large scale simulation on adequate systems [138], [119]. The P2P scientific programming is a potential solution, which is based on existing computers and networks. The present scientific applications on such systems are only concerning problems which are mainly data independents: i.e. each peer does not communicate with the others.

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

3.3.2. Experimentations and Evaluations

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

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

Recently, we also experimented several Krylov based method, such as the Lanczos and GMRES methods on several grids, such as a French-Japanese grid using hundred of PC in France and 4 clusters at the University of Tsukuba. We also experimented on GRID5000 the same methods. We currently use several middleware such as Xtremweb, OmniRPC and Condor. We also begin some experimentations on the Tsubame supercomputer in collaboration with the TITech (Tokyo Institute of Technologies) in order to compare our grid approaches and the High performance one on an hybrid supercomputer.

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

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

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

3.3.3. Languages, Tools and Interface

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

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

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

3.4. Methodology for Large Scale Distributed Systems

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

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

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

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

3.4.1. *Observation tools*

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

3.4.2. *Tool for scalability evaluations*

Several Grid and P2P systems simulators have been developed by other teams: SimGrid [102], GridSim [97], Briks [85]. 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 [127] but many of them considers synchronous systems where the system evolution is guided by phases. In the P2P field, ad hoc many simulators have been developed, mainly for routing in DHT. Emulation is another tool for experimenting systems and networks with a higher degree of realism. Compared to simulation, emulation can be used to study systems or networks 1 or 2 orders of magnitude smaller in terms of number of components. However, emulation runs the actual OS/middleware/applications on actual platform. Compared to real testbed, emulation considers conducting the experiments on a fully controlled platform where all static and dynamic parameters can be controlled and managed precisely. Another advantage of emulation over real testbed is the capacity to reproduce experimental conditions. Several implementations/configurations of the system components can be compared fairly by evaluating them under the similar static and dynamic conditions. Grand-Large is leading one of the largest Emulator project in Europe called Grid explorer (French funding). This project has built and used a 1K CPUs cluster as hardware platform and gathers 24 experiments of 80 researchers belonging to 13 different laboratories. Experiments concerned developing the emulator itself and use of the emulator to explore LSDS issues. In term of emulation tool, the main outcome of Grid explorer is the V-DS system, using virtualization techniques to fold a virtual distributed system 50 times larger than the actual execution platform. V-DS aims at discovering, understanding and managing implicit uncoordinated large scale phenomena. Grid Explorer is still in use within the Grid'5000 platform and serves the community of 400 users 7 days a week and 24h a day.

3.4.3. *Real life testbeds: extreme realism*

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

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

3.5. High Performance Scientific Computing

Numerical simulations are important for obtaining progress in many fields, when for example the problem is too complicated for solving it only theoretically, or too expensive to prototype in the laboratory. These simulations frequently lead to solving very large sets of linear equations and large least squares problems, often with millions of rows and columns. Solving these problems is very time consuming, and the focus of our research is the design of faster algorithms. The demand for better algorithms comes from two trends. First, the problems sizes are growing as the demand for high resolution grows, and as computers get faster. Second, though computers are getting faster, their technology is developing in ways that make current algorithms less and less efficient, and so unable to solve these large problems effectively. The two challenging technology trends are massive parallelism and increased communication cost. Massive parallelism means that every computer, from our laptops to the largest supercomputers, will depend on parallel processing, with the largest computers using many thousands of processors to attain their speed (Computers have broken the Petaflop barrier in 2008). Increased communication costs means that the relative time to move data between processors is increasing *exponentially* relative to the time it to execute arithmetic operations, so conventional algorithms will spend more and more of their time communicating and less and less on computing. One of our research goals consists in developing new algorithms for solving linear systems and least squares problems, that greatly reduce the amount of time spent communicating, relative to conventional algorithms.

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

3.5.1. Efficient linear algebra algorithms

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

3.5.2. Combinatorial tools for scientific computing

This direction of research covers aspects of scientific computing that can be expressed as combinatorial problems and solved using combinatorial tools and algorithms. There are many such examples in scientific computing and an active community works on these subjects. Our research interests in this context consider the problem of structure prediction when solving sparse linear systems of equations and graph partitioning problem.

4. Application Domains

4.1. Building a Large Scale Distributed System for Computing

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

4.2. Security and Reliability of Network Control Protocols

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

4.3. End-User Tools for Computational Science and Engineering

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

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

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

Many smart tools have to be developed to help the end-user to program such environments, using up-to-date component technologies and languages. At the actual present stage of maturity of this programming paradigm for scientific applications, the main goal is to experiment on large platforms, to evaluate and extrapolate performance, and to propose tools for the end-users; with respect to many parameters and under some specify hypothesis concerning scheduling strategies and multicast speeds [118]. We have to always replace the end-user at the center of this scientific programming. Then, we have to propose a framework to program P2P architectures which completely virtualizes the P2P middleware and the heterogeneous hardware. Our approach is based, on the one hand, on component programming and coordination languages, and on the other hand, to the development of an ASP, which may be dedicated to a targeted scientific domain. The YML framework provides a solution to the first point since it offers the YvetteML workflow language in order to orchestrate components. This is a very intuitive programming approach and it favors the re-usability of optimized and bug-free components. The abstraction of the underlying P2P middleware is also ensured by YML by means of its back-end mechanism. The end-user of YML can submit a computing task to any kind of peer connected to Internet as long as YML has a back-end in charge of the middleware which is running on this peer. Currently, YML has two back-ends for the XtremWeb and OmniRPC middleware. Another one for Condor will be soon available. The second point concerns the integration of SPIN to YML in order to get a complete programming tool which covers all the needs of the client in order to run applications (based on linear algebra methods) over the Internet. Finally, the conclusion of our work would be a P2P scientific programming methodology based on experimentations and evaluation on an actual P2P development environment.

5. Software

5.1. APMC-CA

Participant: Sylvain Peyronnet [correspondant].

APMC-CA (Cell Assisted Approximate Probabilistic Model Checker) is a new version of the APMC model checker specially dedicated for the Cell processor architecture. Using the specific features of a Cell architecture, it achieves better performances than APMC 3.0 (the previous version of the software) running on powerful standard workstations.

5.2. XtremWeb

Participant: Gilles Fedak [correspondant].

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

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

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

XtremWeb is available at http://www.xtremweb.net/

5.3. BitDew

Participant: Gilles Fedak [correspondant].

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.

BitDew is available at http://bitdew.gforge.inria.fr/

5.4. SimBOINC

Participant: Derrick Kondo [correspondant].

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

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

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

5.5. XtremLab

Participant: Gilles Fedak [correspondant].

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

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

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

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

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

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

5.6. MPICH-V

Participant: Thomas Hérault [correspondant].

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

In addition to fault tolerant properties, MPICH-V:

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

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

5.7. YML

Participants: Serge Petiton [correspondant], Nahid Emad.

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

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

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

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

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

5.8. The Scientific Programming InterNet (SPIN)

Participant: Serge Petiton [correspondant].

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

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

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

5.9. V-DS

Participant: Franck Cappello [correspondant].

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

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

We have made a virtual platform on the GDX cluster with the Vserver virtualization tool. On this platform, we have launched more than 20K virtual machines (VM) with a folding of 100 (100 VM on each physical machine). However, some recent experiments have shown that a too high folding factor may cause a too long execution time because of some problems like swapping. Currently, we are conducting experiments on another platform based on the virtualization tool named Xen which has been strongly improved since 2 years. We expect to get better result with Xen than with Vserver. Recently, we have been using the V-DS version based on Xen to evaluate at large scales three P2P middleware [132].

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

5.10. PVC: Private Virtual Cluster

Participant: Franck Cappello [correspondant].

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

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

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

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

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

This software is available at http://www.lri.fr/~rezmerit/PVC.html

5.11. OpenWP

Participant: Franck Cappello [correspondant].

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

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

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

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

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

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

5.12. FAult Injection Language (FAIL)

Participant: Sébastien Tixeuil [correspondant].

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 developed.

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

This software is available at http://www.lri.fr/~hoarau/fail/

5.13. Parallel solvers for solving linear systems of equations

Participant: Laura Grigori.

In the last several years, there has been significant research effort in the development of fully parallel direct solvers for computing the solution of large unsymmetric sparse linear systems of equations. In this context, we have designed and implemented a parallel symbolic factorization algorithm, which is suitable for general sparse unsymmetric matrices. The symbolic factorization is one of the steps that is sequential and represents a memory bottleneck. The code is intended to be used with very large matrices when because of the memory usage, the sequential algorithm is not suitable. This code is available in the SuperLU_DIST, a widely used software, developed at UC Berkeley and LBNL by Professor James W. Demmel and Dr. Xiaoye S. Li. The algorithm is presented in [117]. The SuperLU_DIST is available at http://crd.lbl.gov/~xiaoye/SuperLU/ .

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

6. New Results

6.1. Probabilistic Algorithms under Non-Deterministic Scheduler

Participants: Joffroy Beauquier, Colette Johnen.

Joffroy Beauquier and Colette Johnen have worked on the analyze of probabilistic algorithms under non-deterministic scheduler. The scheduler (also called demon or adversary) is a central notion for distributed algorithms and systems. It represents the notion that modelizes the environment. Most of the previous works consider a probabilistic scheduler (events appear randomly). But such a scheduler represents a very feeble adversary, that doesn't modelize properly a malicious environment. In this work the authors present a model that conciliates randomization (for the algorithm) and non-determinism (for the scheduler), taking into account the worst exterior behaviour.

6.2. Extensions of Self-stabilization

Participant: Sébastien Tixeuil.

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. We formulated the problem of selfish stabilization that shows how competition blends with cooperation in a stabilizing environment.

We tackled the open problem of snap-stabilization in message-passing systems. Snap-stabilization is a nice approach to design protocols that withstand transient faults. Compared to the well-known self-stabilizing approach, snap-stabilization guarantees that the effect of faults is contained immediately after faults cease to occur. Our contribution is twofold: we show that (1) snap-stabilization is impossible for a wide class of problems if we consider networks with finite yet unbounded channel capacity; (2) snap-stabilization becomes possible in the same setting if we assume bounded-capacity channels. We propose three snap-stabilizing protocols working in fully-connected networks. Our work opens exciting new research perspectives, as it enables the snap-stabilizing paradigm to be implemented in actual networks.

Self-stabilization is a strong property that guarantees that a network always resume correct behavior starting from an arbitrary initial state. Weaker guarantees have later been introduced to cope with impossibility results: probabilistic stabilization only gives probabilistic convergence to a correct behavior. Also, weak stabilization only gives the possibility of convergence. We investigated the relative power of weak, self, and probabilistic stabilization, with respect to the set of problems that can be solved. We formally proved that in that sense, weak stabilization is strictly stronger that self-stabilization. Also, we refined previous results on weak stabilization to prove that, for practical schedule instances, a deterministic weak-stabilizing protocol can be turned into a probabilistic self-stabilizing one. This latter result hints at more practical use of weak-stabilization, as such algorithms are easier to design and prove than their (probabilistic) self-stabilizing counterparts.

6.3. Self-stabilizing distributed control

Participant: Sébastien Tixeuil.

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 presented 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 proved KDP correct and evaluated its performance. We then extended KDP to handle fully generalized problem. We further extended it to handle a similarly generalized drinking philosophers problem. We described how KDP can be implemented in wireless sensor networks and demonstrate that this implementation does not jeopardize its correctness or termination properties.

We analyzed the longest increasing contiguous sequence or maximal ascending run of random variables with common uniform distribution but not independent. Their dependence is characterized by the fact that two successive random variables cannot take the same value. Using a Markov chain approach, we studied the distribution of the maximal ascending run and we developed an algorithm to compute it. This problem comes from the analysis of several self-organizing protocols designed for large-scale wireless sensor networks, and we showed how our results apply to this domain.

We quantified 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 showed 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 showed 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.

We considered gossiping among mobile agents in graphs: agents move on the graph and have to disseminate their initial information to every other agent. We focused on self-stabilizing solutions for the gossiping problem, where agents may start from arbitrary locations in arbitrary states. Self-stabilization requires (some of the) participating agents to keep moving forever, hinting at maximizing the number of agents that could be allowed to stop moving eventually. We formalized the self-stabilizing agent gossip problem, introduces the quiescence number (i.e., the maximum number of eventually stopping agents) of self-stabilizing solutions and investigates the quiescence number with respect to several assumptions related to synchrony, whiteboard availability, and anonymity.

The matching problem asks for a large set of disjoint edges in a graph. It is a problem that has received considerable attention both from the sequential and the self-stabilizing communities. Previous work has resulted in self-stabilizing algorithms for computing a maximal (1/2-approximation) matching in a general graph, as well as computing a 2/3-approximation on more specific graph types. We presented the first self-stabilizing algorithm for finding a 2/3-approximation to the maximum matching problem in a general graph. We showed that our new algorithm stabilizes in at most exponential time under a distributed adversarial daemon, and $O(n^2)$ rounds under a distributed fair daemon, where n is the number of nodes in the graph.

6.4. Security in Wireless networks

Participant: Sébastien Tixeuil.

The Sybil attack in unknown port networks such as wireless is not considered tractable. A wireless node is not capable of independently differentiating the universe of real nodes from the universe of arbitrary non-existent fictitious nodes created by the attacker. Similar to failure detectors, we propose to use *universe detectors* to help nodes determine which universe is real. We (i) define several variants of the neighborhood discovery problem under Sybil attack (ii) propose a set of matching universe detectors (iii) demonstrate the necessity of additional topological constraints for the problems to be solvable: node density and communication range; (iv) present $\mathcal{SAND}$ — an algorithm that solves these problems with the help of appropriate universe detectors, this solution demonstrates that the proposed universe detectors are the weakest detectors possible for each problem.

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

6.5. Large Scale Peer to Peer Performance Evaluations

Participant: Serge Petiton.

6.5.1. Large Scale Grid Computing

Recent progress has made possible to construct high performance distributed computing environments, such as computational grids and cluster of clusters, which provide access to large scale heterogeneous computational resources. Exploration of novel algorithms and evaluation of performance is a strategic research for the future of computational grid scientific computing for many important applications [129]. We adapted [106] an explicit restarted Lanczos algorithm on a world-wide heterogeneous grid platform. This method computes one or few eigenpairs of a large sparse real symmetric matrix. We take the specificities of computational resources into account and deal with communications over the Internet by means of techniques such as out-of-core and data persistence. We also show that a restarted algorithm and the combination of several paradigms of parallelism are interesting in this context. We perform many experimentations using several parameters related to the Lanczos method and the configuration of the platform. Depending on the number of computed Ritz eigenpairs, the results underline how critical the choice of the dimension of the working subspace is. Moreover, the size of platform has to be scaled to the order of the eigenproblem because of communications over the Internet.

6.5.2. High Performance Cluster Computing

Grid computing focuses on making use of a very large amount of resources from a large-scale computing environment. It intends to deliver high-performance computing over distributed platforms for computation and data-intensive applications. We propose [148] an effective parallel hybrid asynchronous method to solve large sparse linear systems by the use of a Grid Computing platform Grid5000. This hybrid method combines a parallel GMRES(m) (Generalized Minimum RESidual) algorithm with the Least Square method that needs some eigenvalues obtained from a parallel Arnoldi algorithm. All of these algorithms run on the different processors of the platform Grid5000. Grid5000, a 5000 CPUs nation-wide infrastructure for research in Grid computing, is designed to provide a scientific tool for computing. We discuss the performances of this hybrid method deployed on Grid5000, and compare these performances with those on the IBM SP series supercomputers.

6.5.3. Large Scale Power aware Computing

Energy conservation is a dynamic topic of research in High Performance Computing and Cluster Computing. Power-aware computing for heterogeneous world-wide Grid is a new track of research. We have studied and evaluated the impact of the heterogeneity of the computing nodes of a Grid platform on the energy consumption. We propose to take advantage of the slack-time caused by the heterogeneity in order to save energy with no significant loss of performance by using Dynamic Voltage Scaling (DVS) in a distributed eigensolver [107]. We show that using DVS only during the slack-time does not penalize the performances but it does not provide significant energy savings. If DVS is applied to all the execution, we get important global and local energy savings (respectively up to 9% and 20%) without a significant rise of the wall-clock times.

6.6. Combining Annotation Language and Workflow Environments for Porting Existing Applications on Grids

Participant: Franck Cappello.

Many Industrial companies are looking for programming environments to port their existing applications to the grid. Workflow environments are an appealing solution for them because they match the architectural features of grids (hierarchy, heterogeneity, dynamism). However, current workflow environments require of programmers significant efforts to adapt existing applications. In [33], we propose the OpenWP programming and runtime environment, in order to ease the adaptation and execution of existing applications on grids. OpenWP essentially allows the programmer: 1) expressing the parallelism and distribution in existing codes using directives and 2) executing the applications on grids using existing workflow engines. This paper presents the OpenWP environment in details and evaluates its performance from a non trivial industrial mesher application used by an aerospace company.

6.7. Characterizing Intranet and Internet Desktop Grids

Participant: Gilles Fedak.

Desktop Grids use the computing, network and storage resources from idle desktop PC's distributed over multiple-LAN's or the Internet to compute a large variety of resource-demanding distributed applications. While these applications need to access, compute, store and circulate large volumes of data, little attention has been paid to data management in such large-scale, dynamic, heterogeneous, volatile and highly distributed Grids. In most cases, data management relies on ad-hoc solutions, and providing a general approach is still a challenging issue.

To address this problem, we propose in [44] the BitDew framework, a programmable environment for automatic and transparent data management on computational Desktop Grids. This paper describes the BitDew programming interface, its architecture, and the performance evaluation of its runtime components. BitDew relies on a specific set of meta-data to drive key data management operations, namely life cycle, distribution, placement, replication and fault-tolerance with a high level of abstraction. The Bitdew runtime environment is a flexible distributed service architecture that integrates modular P2P components such as DHT's for a distributed data catalog and collaborative transport protocols for data distribution. Through several examples, we describe how application programmers and Bitdew users can exploit Bitdew's features. The performance evaluation demonstrates that the high level of abstraction and transparency is obtained with a reasonable overhead, while offering the benefit of scalability, performance and fault tolerance with little programming cost.

In [43], we propose a multi-protocol file transfer service, based on BitDew, which supports client/server and P2P protocols, as well as Wide Area Storage such as Amazon S3. We describe the mechanisms used to ensure file transfer monitoring and reliability. We explain how to plug-in new or existing protocols and we give evidence of the versatility of the framework by implementing the HTTP, FTP and BitTorrent protocols and access to the Amazon S3 and IBP Wide Area Storage.

In [34] and [16] we discuss how BitTorrent data distribution protocol can be adapted to Desktop Grid computing environments, particularly to the BOINC platform. To date, Desktop Grid systems have focused primarily on utilizing spare CPU cycles, yet have neglected to take advantage of client network capabilities. We integrated the BitTorrent protocol within the BOINC middleware and run middle scale experiments We measured the impact of the BitTorrent components in both the BOINC client and server, and compared it with the original BOINC. We proved that the BitTorrent client has a negligible influence in the client's computation time, even when it is seeding during half that time, and determined the specific overhead in clients and servers. It allowed us to discover an abnormal result in the server network output that should be further analyzed.

Service grids and desktop grids are both promoted by their supportive communities as great solutions for solving the available compute power problem and helping to balance loads across network systems. Little work, however, has been undertaken to blend these two technologies together in an effort to create one vast and seamless pool of resources. In [26], [36], [46] and [14], we present a new FP7 infrastructures project, entitled Enabling Desktop Grids for e-Science (EDGeS), that is building technological bridges to facilitate service and desktop grid interoperability. We provide a taxonomy for existing state of the art grid systems and background into service grids, such as EGEE and volunteer computing platforms, such as BOINC and XtremWeb. We describe our approach within three themes for identifying translation technologies for porting applications between service grids and desktop grids and vice versa. The individual themes discuss the actual bridging technologies employed, the distributed data issues surrounding deployment and application development and user access issues. In [50], we give a detailed presentation of the BOINC to EGEE bridge, and in [31], we address the security issues when bridging Service Grid with Desktop Grid. We present how to bridge EGEE resources with our XtremWeb platform using the gliding-in mechanism and we describe a new Desktop Grid security model to bridge this anonymous environment to the strongly securized Service Grid one. Finally we propose an implementation of this security model in the XtremWeb middleware and report on performance evaluation.

6.8. Grids and High Performance Computing

Participants: Thomas Hérault, Sylvain Peyronnet.

[29] As High Performance platforms (Clusters, Grids, etc.) continue to grow in size, the average time between failures decreases to a critical level. An efficient and reliable fault tolerance protocol plays a key role in High Performance Computing. Rollback recovery is the most common fault tolerance technique used in High Performance Computing and especially in MPI applications. This technique relies on the reliability of the checkpoint storage. Most of the rollback recovery protocols assume that the checkpoint servers machines are reliable. However, in a grid environment any unit can fail at any moment, including components used to connect different administrative domains. Such failures lead to the loss of a whole set of machines, including the more reliable machines used to store the checkpoints in this administrative domain. Thus it is not safe to rely on the high MTBF (Mean Time Between Failures) of specific machines to store the checkpoint images. In this work, we have introduced a new coordinated checkpoint protocol, which tolerates checkpoint server failures and clusters failures, and ensures a checkpoint storage reliability in a grid environment. To provide this reliability the protocol is based on a replication process. We proposed new hierarchical replication strategies, with two different degrees of hierarchy, adapted to the topology of cluster of clusters. Our solution exploits the locality of checkpoint images in order to minimize inter-cluster communication. We evaluated the effectiveness of our two hierarchical replication strategies through simulations against several criteria such as topology and scalability .

[35] Institutional grids consist of the aggregation of clusters belonging to different administrative domains to build a single parallel machine. To run an MPI application over an institutional grid, one has to address many challenges. One of the first problems to solve is the connectivity of the different nodes not belonging to the same administrative domain. Techniques based on communication relays, dynamic port opening, among others. have been proposed. In this work, which was supported by the European FP6 STREP QosCosGrid, we proposed a set of Grid or Web Services to abstract this connectivity service, and we evaluated the performances of this new level of communication for establishing the connectivity of an MPI application over an experimental grid.

[15] This article is the journal version of the work done in 2007 with Darius Buntinas. The description of the work appears in the RAWEB 2007.

6.9. Verifying Distributed Monte Carlo Methods

Participant: Sylvain Peyronnet.

[19] We presents several related methods for drawing traces. First, we design a method that show how to draw traces uniformly at random in large models composed of several components. We also designed a method for drawing traces according to a given coverage criterion is presented, to- gether with a notion of randomized coverage satisfaction. These methods rely on combinatorial algorithms, based on a representation of the model by an automaton or by a product of several automata, synchronised or not.

[27] We designed APMC-CA (Cell Assisted Approximate Probabilistic Model Checker), a new version of the APMC model checker specially dedicated for the Cell processor architecture. We show that using the specific features of a Cell architecture, we can achieve better performances than APMC 3.0 running on powerful standard workstations. We also demonstrated the difficulties raised by the powerfull, yet constraint environment of the Cell, and its limitations with respect to memory access, when applied to simulation and model checking.

6.10. High performance scientific computing

Participant: Laura Grigori.

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

6.10.1. Communication avoiding algorithms for LU and QR factorizations

This research focuses on developing new algorithms for linear algebra problems, that minimize the required communication, in terms of both latency and bandwidth. The results we have obtained to date concern algorithms for solving dense linear systems and dense least squares problems. This research is joint work with J. Demmel and M. Hoemmen from U.C. Berkeley, J. Langou from C.U. Denver, and H. Xiang from INRIA Saclay.

In [67] we present parallel and sequential dense QR factorization algorithms that are both *optimal* (sometimes only up to polylogarithmic factors) in the amount of communication (latency and bandwidth) they require, and *numerically as stable* as conventional Householder QR. The first set of algorithms, “Tall Skinny QR (TSQR), are for matrices for which the number of rows is much larger than the number of columns, and which have their rows distributed over processors in a one-dimensional (1-D) block row layout. The second set of algorithms, “Communication-Avoiding QR (CAQR), are for general rectangular matrices distributed using a two-dimensional (2-D) block cyclic layout.

In [47] we present a Communication-Avoiding LU factorization (CALU) algorithm for computing in parallel the LU factorization of a dense matrix A distributed in a two-dimensional (2D) layout. To decrease the communication required in the LU factorization, CALU uses a new pivoting strategy, referred to as ca-pivoting, that may lead to a different row permutation than the classic LU factorization with partial pivoting. Our numerical results show that ca-pivoting scheme is stable in practice. We observe that it behaves as a threshold pivoting, and in practical experiments $|L|$ is bounded by 3, while in LU factorization with partial pivoting, $|L|$ is bounded by 1,where $|L|$ denotes the matrix of absolute values of the entries of L. Extensive testing on many different matrices always resulted in residuals $||Ax - b||$ comparable to those from conventional partial pivoting.

To prove optimality of the algorithms, we extend in [68] known lower bounds on communication bandwidth for sequential and parallel versions of conventional $\Theta(n^3)$ matrix multiplication (see Hong and Kung and Irony, Toledo, and Tiskin) to also provide latency lower bounds, and show that these bounds also apply to $\Theta(n^3)$ implementations of dense LU and QR decompositions. Showing that the bounds apply to LU is easy, but QR is more complicated. With an optimal choice of matrix layout, CALU and CAQR attain both the bandwidth and the latency lower bounds (sometimes only up to polylogarithmic factors). In other words, the algorithms have significantly lower latency cost in the parallel case, and significantly lower latency and bandwidth costs in the sequential case, than conventional algorithms as for example implemented in LAPACK and ScaLAPACK. We note that although our new algorithms perform slightly more floating point operations than LAPACK and ScaLAPACK, they have the same highest order terms in their floating point operation counts.

6.10.2. Combinatorial tools for scientific computing

This direction of research covers aspects of scientific computing that can be expressed as combinatorial problems and solved using combinatorial tools and algorithms. There are many such examples in scientific computing. In this context, we have addressed and solved several open questions in the structure prediction problem for solving sparse systems of equations [20].

We have also worked on reordering algorithms used as front-end ordering preceding the LU factorization for sparse matrices. Although this problem is NP-complete, in practice there are several efficient fill reducing heuristics. Fill refers to an element which is zero in the input matrix A and becomes nonzero in the factors L and U. We have exploited the usage of hypergraphs in this context [73] that take into account the asymmetry of the input matrix. Reordering for unsymmetric matrices is considered to be an open problem, due to the difficulty of generalizing concepts from symmetric matrices to unsymmetric matrices.

We have also worked on another aspect that refers to the graph partitioning algorithms used as front-end ordering preceding the factorization [21]. This is a collaboration with Guy Atenekeng Kahou (IRISA, Rennes) and Masha Sosonkina (Ames Laboratory, USA), that aims at partitioning a matrix into a block-diagonal form, such that any two consecutive blocks overlap. The partitioned matrix is suitable for applying the explicit formulation of the Multiplicative Schwarz Preconditioner developed by Atenekeng, Kamgnia and Philippe.

6.10.3. Preconditioning techniques

A different direction of research is related to preconditioning large sparse linear systems of equations. In this research we consider different preconditioners based on incomplete factorizations. The tangential filtering is an incomplete factorization technique where it is possible to ensure that the factorization will coincide with the original matrix for some specified vector.

Recent research has shown that ILU combined with tangential filtering leads to very efficient preconditioner for matrices arising from the discretization of scalar equations on structured grids and we have investigated further their properties [75], [80] . We also designed preconditioners based on Kronecker product approximation or Schilder factorization for saddle point problems arising from PDE or optimization [77].

7. Other Grants and Activities

7.1. Regional, National and International Actions

7.1.1. Activities starting in 2008

- **ANR SPADES** Coordinated by LIP-ENS Lyon. (Sylvain Peyronnet, Franck Cappello)
- **Défi ANR SECSI** Participant to this challenge. From September 2008 to August 2010. Managed by the SAIC. (Thomas Hérault, Sylvain Peyronnet, Sébastien Tixeuil)
- **EDGeS: Enabling Desktop Grids for e-Science**, Founded by EU-FP7 for 24 month in 2008 and 2009. 236 KEuros budget. Participants: SZTAKI, INRIA, CIEMAT, Fundecyt, UoW, CU, FCTUC. Responsible for: “JRA1: Service Grids-Desktop Grids Bridges Technologies” (Gilles Fedak, Thomas Hérault)

- **ANR Cosinus project PETAL- PETascale ALgorithms for preconditioning for scientific applications** 2008-2010. Collaboration with Laboratoire Lions - Universite 6, IFP, INRIA Bordeaux and CEA, UC Berkeley and Argonne. The goal is to investigate preconditioning techniques on multicore architectures and apply them on real world applications from IFP, CEA and Argonne. (Laura Grigori, Principal Investigator)
- **Digiteo DIM-08 project X-Scale-NL – Scheduling and numerical libraries enabling scientific applications on petascale machines** 2008-2011. Funding for a Phd student and travel (114000 euros). Participants: Laura Grigori (Principal Investigator), F. Cappello (INRIA), T. Herault, S. Peyronnet (LRI) and two foreign collaborators: J. Demmel from UC Berkeley and J. Darbon from UC Los Angeles.

7.1.2. Other activities

- **PECO-NEI** RFR with Eastern Europe Countries, 2006 - 2009, PI L. Grigori
- INRIA Associated Team **"F-J Grid"** with University of Tsukuba, head: Franck Cappello
- **CIFRE EADS**, 3 years (2005-2008), head: Franck Cappello.
- INRIA funding, **MPI-V**, collaboration with UTK, LALN and ANL, head: Franck Cappello
- Regional Council **"Grid eXplorer"**, 3 years (2006-2009), co-chair: Franck Cappello
- ANR Jeunes chercheurs **XtremLab**: G. Fedak, 3 years (2005-2008)
- ANR CIS Project **FF2A3**, 3 years (2007 - 2010), PI F. Hecht, subproject head L. Grigori
- **AURORA project France-Norway** 1 year, "Self-stabilization and Sensor Networks", chair: S. Tixeuil
- **European CoreGrid Network of Excellence**, 4 years (2004 -2008), subtask head: S. Tixeuil, subproject heads: G. Fedak, T. Herault, S. Tixeuil
- **European STREP** (6th FP pri5-IST), 3 years (2006-2008). Quasi-Opportunistic Supercomputing for Complex Systems in Grid Environments (QosCosGrid), managment board representative: Franck Cappello, task head: Thomas Herault
- European project. **Grid4All**, 3 years (2006-2009), managment board representative: Franck Cappello, task head: Gilles Fedak
- **NEGST Project** (CNRS-JST), 3 years (2006-2008), http://www2.lifl.fr/MAP/negst/ , head(french side): Serge Petiton
- **CARRIOCAS**, Pole de Competitivité System@tic, 3 years (2006-2009), http://www.carriocas.org/, Franck Cappello
- **HipCal**, ANR CIS, 3 years (2006-2009), http://hipcal.lri.fr/wakka.php?wiki=PagePrincipale, Franck Cappello

8. Dissemination

8.1. Services to the Scientific Community

8.1.1. Book/Journal edition

- Laura Grigori, Guest editor in charge of a Special Issue of Parallel Computing on Parallel Matrix Algorithms and Applications.

8.1.2. Conference Organisation

- Franck Cappello, Workshop Chair, IEEE/ACM CCGRID'2008

- Franck Cappello, Program vice-Chair, Mardi Gras Conference 2008,
- Franck Cappello, General Chair IEEE APSCC'2008, Yilan, Taiwan.
- Sébastien Tixeuil, Program co-chair, Algotel 2008
- Laura Grigori, Organizer with J. Demmel (UC Berkeley) of a Minisymposium entitled "Communication avoiding algorithms" at the SIAM Conference on Parallel Processing May 2008. The minisymposium consists of 8 talks of 30 minutes each.
- Laura Grigori, Organizer with S. Toledo (Tel-Aviv University) of a minisymposium "Progress in Orthogonal Factorizations (QR and SVD)" at the SIAM Conference on Parallel Processing May 2008. The minisymposium consists of 4 talks of 30 minutes each.

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
- Sébastien Tixeuil, "Technique et Science Informatiques", since 2005

8.1.4. Steering Committee membership

- Franck Cappello, IEEE/ACM HPDC
- Franck Cappello, IEEE/ACM CCGRID

8.1.5. Program Committee membership

- Laura Grigori, Parallel Matrix Algorithms and Applications (PMAA08) 2008
- Franck Cappello, The 28th International Conference on Distributed Computing Systems (ICDCS 2008), June 17 - 20, 2008, Beijing, China
- Franck Cappello, Mardi Gras Conference 2008 - Workshop on Grid-Enabling Applications, Jan 30 - Feb 2, 2008
- Franck Cappello, SIMUTools 2008: International Conference on Simulation Tools and Techniques
- Franck Cappello, HCW , 17th Heterogeneous Computing Workshop (HCW'08) 2008
- Franck Cappello, IPTPS 2008
- Franck Cappello, HotP2P 2008
- Franck Cappello, ACM HPDC 17th IEEE International conference on High Performance Distributed Computing, 2008
- Franck Cappello, IEEE/ACM SC 2008, 21th IEEE/ACM International Conference for High Performance Computing and Communications (SC08) - Austin Texas, 2008
- Franck Cappello, The 9th IEEE/ACM International Conference on Grid Computing (Grid 2008), Tsukuba, Japan
- Franck Cappello, IEEE/ACM CCGRID 8th IEEE International Symposium on Cluster Computing and the Grid.Lyon, France , May 2008
- Sébastien Tixeuil, OPODIS 2008, Program co-Chair
- Sébastien Tixeuil, DISC 2008
- Sébastien Tixeuil, Algotel 2008, Program co-Chair
- Sébastien Tixeuil, SSS 2008
- Sébastien Tixeuil, Sirocco 2008
- Sébastien Tixeuil, WAMAN 2008

- Sébastien Tixeuil, CODS 2008
- Sylvain Peyronnet, ISVC 2008
- Sylvain Peyronnet, ICCP 2008
- Thomas Hérault, ISPDC 2008
- Thomas Hérault, VECPAR 2008
- Gilles Fedak, Renpar 2008
- Gilles Fedak, Euromicro PDP 2008

8.1.6. School and Workshop organization

- Franck Cappello and Gilles Fedak, General Co-chairs Workshop on Large-Scale and Volatile Desktop Grids (PCGrid 2008) in conjunction with IPDPS, Miami, USA, 2008
- Gilles Fedak, General chair of 2nd XtremWeb User Group Meeting, Orsay, 2008
- Gilles Fedak, Co-chair of Global and Peer-to-Peer Computing, in conjunction with CCGRID, with M. Sato (Univ. Tsukuba), Lyon, France, may 2008

8.2. Invited Conferences

8.2.1. Invited International Conference

- Laura Grigori, Invited plenary talk at "IBM a l'ere du Petaflops, day organized by IBM Research in Paris, 18 October 2008.
- Laura Grigori, Invited talk at US Europe Young Scientists Symposium, October 13-15, 2008, organized at Oak Ridge National Laboratory, USA. Participation on invitation only.
- Gilles Fedak, Invited talk at 3rd Workshop on the Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE-CN'08), High Performance Distributed Computing (HPDC'08) [45]
- Gilles Fedak, Bridging XtremWeb with the EGEE Grid, Invited talk at First EDGeS User Forum and Industry Forum, Orsay, France
- Gilles Fedak, Implementing New File Transfer Protocols in BitDew: Amazon S3 Case Study, Invited talk at XW'08: 2nd XtremWeb Users Group Workshop, Orsay, France
- Franck Cappello, "Fault Tolerance & PetaScale Systems: Current Knowledge, Challenges and Opportunities", KEYNOTE talk, Europar, Spain, Aug. 2008
- Franck Cappello, "Fault Tolerance & PetaScale Systems: Current Knowledge, Challenges and Opportunities", KEYNOTE talk, EuroPVM/MPI, Dublin, Sept. 2008
- Franck Cappello, "French National Grid Testbed: Grid 5000", KEYNOTE talk, DCABES 2008, Dalian, China, July 2008
- Franck Cappello, "Fault Tolerance & PetaScale Systems: Current Knowledge, Challenges and Opportunities", Invited talk, Clusters and Computational Grids for Scientific Computing 2008, Highland Lake Inn, Asheville, USA, Sept. 2008
- Franck Cappello, "Fault Tolerance & PetaScale Systems: Current Knowledge, Challenges and Opportunities", Invited talk, HPC Conference, Cetraro, June. 2008
- Franck Cappello, "Grid'5000", Invited talk, EuroVO workshop on Grid, Garching, April 2008
- Franck Cappello, "Computer Science Grids and research in Grid Computing", KEYNOTE talk, 3rd EGEE User Forum, Clermont-Ferrand, Feb. 2008

8.2.2. Seminaries

- Laura Grigori, Seminar at Colorado University at Denver, Mathematics Department.

8.2.3. Challenges

- Gilles Fedak, Large-Scale Bioinformatic Computing on Data Desktop Grid, participation to a challenge in First IEEE International Scalable Computing Challenge (SCALE 2008) along with CCGRID'08 [48]

8.2.4. Scientific Popularization

- “Planification Monté-Carlo Parallèle à Base de Bandits et Application au Jeu de Go”. Article in the periodical publication “Plein-Sud spécial recherche”, edited by the university Paris Sud-XI

9. Bibliography

Major publications by the team in recent years

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

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

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

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

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

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

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

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

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

Year Publications

Doctoral Dissertations and Habilitation Theses

[10] M. Cargnelli. *OpenWP: Etude et Extension des Technologies de Workflows pour le Calcul Haute Performance sur Grille*, Ph. D. Thesis, LRI, Univ Paris-Sud, December 2008.

[11] W. Hoarau. *Injection de fautes dans les systèmes distribués*, Ph. D. Thesis, LRI, Univ. Paris-Sud, March 2008, http://www.lri.fr/~hoarau/fichiers/these-hoarau.pdf.

[12] O. Peres. *Construction de topologies autostabilisante dans les systèmes à grande échelle*, Ph. D. Thesis, LRI, Univ. Paris-Sud, September 2008.

[13] B. Quétier. *EmuGrid : étude de mécanismes de virtualisation pour l'émulation conforme de Grilles à grande échelle*, Ph. D. Thesis, LRI, Univ. Paris-Sud, September 2008.

Articles in International Peer-Reviewed Journal

[14] Z. Balaton, Z. Farkas, G. Gombás, P. Kacsuk, R. Lovas, A. C. Marosi, A. Emmen, G. Terstyanszky, T. Kiss, I. Kelley, I. Taylor, O. Lodygensky, M. Cárdenas-Montes, G. Fedak, F. Araujo. *EDGeS: the Common Boundary Between Service and Desktop Grids*, in "Parallel Processing Letters", Edges, Similar versions of the paper presenting the EDGeS project have been published in several conferences (Chinagrig, Ibergrid, Coregrid), vol. 18, n^{o} 3, September 2008, p. 433–453.

[15] D. Buntinas, 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 Protocols*, in "Future Generation Comp. Syst.", vol. 24, n^{o} 1, 2008, p. 73-84.

[16] F. Costa, L. Silva, G. Fedak, I. Kelley. *Optimizing Data Distribution in Desktop Grid Platforms*, in "Parallel Processing Letters", extended version of paper at PCGRID 08, vol. 18, n^{o} 3, September 2008, p. 391–410.

[17] P. Danturi, M. Nesterenko, S. Tixeuil. *Self-stabilizing Philosophers with Generic Conflicts*, in "ACM Transactions of Adaptive and Autonomous Systems (TAAS)", 2008.

[18] A. Dasgupta, S. Ghosh, S. Tixeuil. *An Exercise in Selfish Stabilization*, in "ACM Transactions of Adaptive Autonomous Systems (TAAS)", 2008, http://hal.inria.fr/inria-00335919/en/.

[19] M.-C. Gaudel, A. Denise, S.-D. Gouraud, R. Lassaigne, J. Oudinet, S. Peyronnet. *Coverage-biased Random Exploration of Models*, in "Electr. Notes Theor. Comput. Sci.", vol. 220, n^{o} 1, 2008, p. 3-14.

[20] L. Grigori, J. Gilbert, M. Cosnard. *Structure Prediction for Sparse Gaussian Elimination with Partial Pivoting*, in "SIAM Journal on Matrix Analysis and Applications", vol. 30, n^{o} 4, 2008.

[21] G. A. Kahou, L. Grigori, M. Sosonkina. *A Partitioning Algorithm for Block Diagonal Matrices with Overlap*, in "Parallel Computing", vol. 34, n^{o} 6, 2008.

[22] R. LASSAIGNE, S. PEYRONNET. *Probabilistic Verification and Approximation*, in "Annals of Pure and Applied Logic", vol. 152, n$^{\text{o}}$ 1–3, 2008, p. 122-131.

[23] T. MASUZAWA, S. TIXEUIL. *On Bootstrapping Topology Knowledge in Anonymous Networks*, in "ACM Transactions on Adaptive and Autonomous Systems (TAAS)", 2008.

[24] N. MITTON, K. PAROUX, B. SERICOLA, S. TIXEUIL. *Ascending runs in dependent uniformly distributed random variables: Application to wireless networks*, in "Methodology and Computing in Applied Probability", 2008, https://hal.inria.fr/inria-00239348.

[25] 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 "J. Grid Comput.", vol. 6, n$^{\text{o}}$ 2, 2008, p. 141-157.

International Peer-Reviewed Conference/Proceedings

[26] Z. BALATON, Z. FARKAS, G. GOMBÁS, P. KACSUK, R. LOVAS, A. C. MAROSI, A. EMMEN, G. TERSTYANSZKY, T. KISS, I. KELLEY, I. TAYLOR, O. LODYGENSKY, M. CÁRDENAS-MONTES, G. FEDAK, F. ARAUJO. *EDGeS: the Common Boundary Between Service and Desktop Grids*, in "Proceedings of the CoreGrid Integration Workshop (CGIW08), Hersonissos-Crete, Greece", Similar versions of the paper presenting the EDGeS project have been published in several conferences., April 2008.

[27] A. BORGHI, T. HERAULT, R. LASSAIGNE, S. PEYRONNET. *Cell Assisted APMC*, in "Fifth International Conference on the Quantitative Evaluaiton of Systems (QEST 2008), Saint-Malo, France", IEEE Computer Society, 14-17 September 2008, p. 75-76.

[28] F. BOUABACHE, T. HERAULT, G. FEDAK, F. CAPPELLO. *Hierarchical Replication Techniques to Ensure Checkpoint Storage Reliability in Grid Environment*, in "Poster in The sixth ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-08), Doha", 2008.

[29] F. BOUABACHE, T. HERAULT, G. FEDAK, F. CAPPELLO. *Hierarchical Replication Techniques to Ensure Checkpoint Storage Reliability in Grid Environments*, in "Proceedings of 8th IEEE International Symposium on Cluster Computing and the Grid CCGRID'08, Lyon, France", may 2008, p. 475–483.

[30] Z. BOUZID, M. G. POTOP-BUTUCARU, S. TIXEUIL. *Byzantine-resilient Convergence in Oblivious Robot Networks*, in "International Conference on Distributed Systems and Networks (ICDCN 2009)", January 2009.

[31] G. CAILLAT, G. FEDAK, H. HE, O. LODYGENSKY, E. URBAH. *Towards a Security Model to Bridge Internet Desktop Grids and Service Grids*, in "Proceedings of the Euro-Par 2008 Workshops (LNCS), Workshop on Secure, Trusted, Manageable and Controllable Grid Services (SGS'08), Las Palmas de Gran Canaria, Spain", August 2008.

[32] F. CAPPELLO. *Fault Tolerance for PetaScale Systems: Current Knowledge, Challenges and Opportunities*, in "PVM/MPI", A. L. LASTOVETSKY, T. KECHADI, J. DONGARRA (editors), Lecture Notes in Computer Science, vol. 5205, Springer, 2008.

[33] M. CARGNELLI, G. ALLÉON, F. CAPPELLO. *OpenWP: combining annotation language and workflow environments for porting existing applications on grids*, in "Proceedings of IEEE GRID 2008, Tsukuba", IEEE Press, October 2008, p. 176-183.

[34] F. Costa, L. Silva, G. Fedak, I. Kelley. *Optimizing the Data Distribution Layer of BOINC with BitTorrent*, in "Proceedings of IPDPS'08, 2nd Workshop on Desktop Grids and Volunteer Computing Systems (PCGrid 2008), Miami, Florida", apr 2008, p. 1–8.

[35] C. Coti, T. Herault, S. Peyronnet, A. Rezmerita, F. Cappello. *Grid Services for MPI*, in "8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008), Lyon, France", IEEE Computer Society, 19-22 May 2008, p. 417-424.

[36] M. Cárdenas-Montes, A. Emmen, A. C. Marosi, F. Araujo, G. Gombás, G. Fedak, I. Kelley, I. Taylor, O. Lodygensky, P. Kacsuk, R. Lovas, T. Kiss, Z. Balaton, Z. Farkas, G. Terstyanszky. *EDGeS: Bridging Desktop and Service Grids*, in "Proceedings of IBERGRID, 2nd Iberian Grid Infrastructure Conference, Porto, Portugal", Similar versions of the paper presenting the EDGeS project have been published in several conferences., May 2008, p. 212–226.

[37] S. Delaët, P. S. Mandal, M. A. Rokicki, S. Tixeuil. *Deterministic Secure Positioning in Wireless Sensor Networks*, in "Proceedings of the IEEE Conference on Distributed Computing in Sensor Systems", S. E. Nikoletseas, B. S. Chlebus, D. B. Johnson, B. Krishnamachari (editors), Lecture Notes in Computer Science, vol. 5067, Springer, 2008, p. 469-477.

[38] S. Delaët, S. Devismes, M. Nesterenko, S. Tixeuil. *Brief Announcement: Snap-Stabilization in Message-Passing Systems*, in "Principles of Distributed Computing (PODC 2008)", August 2008, https://hal.inria.fr/inria-00248465.

[39] S. Delaët, S. Devismes, M. Nesterenko, S. Tixeuil. *Snap-Stabilization in Message-Passing Systems*, in "International Conference on Distributed Systems and Networks (ICDCN 2009)", January 2009, https://hal.inria.fr/inria-00248465.

[40] S. Delaët, P. S. Mandal, M. A. Rokicki, S. Tixeuil. *Deterministic Secure Positioning in Wireless Sensor Networks*, in "Proceedings of the ACM/IEEE International Conference on Distributed Computing in Sensor Networks (DCOSS 2008)", Lecture Notes in Computer Science, Springer-Verlag, June 2008, https://hal.inria.fr/inria-00179056.

[41] S. Devismes, S. Tixeuil, M. Yamashita. *Weak vs. Self vs. Probabilistic Stabilization*, in "Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS 2008), Beijin, China", June 2008, http://hal.inria.fr/inria-00189952/fr/.

[42] G. Fedak, H. He, F. Cappello. *A File Transfer Service with Client/Server, P2P and Wide Area Storage Protocols*, in "Globe", A. Hameurlain (editor), Lecture Notes in Computer Science, vol. 5187, Springer, 2008, p. 1-11.

[43] G. Fedak, H. He, F. Cappello. *A File Transfer Service with Client/Server, P2P and Wide Area Storage Protocols*, in "Proceedings of the First International Conference on Data Management in Grid and P2P Systems (Globe'2008), Turin, Italy", LNCS, Springer Verlag, September 2008, p. 1–11.

[44] G. Fedak, H. He, F. Cappello. *BitDew: A Programmable Environment for Large-Scale Data Management and Distribution*, in "Proceedings of the ACM/IEEE SuperComputing Conference (SC'08), Austin, USA", November 2008.

[45] G. FEDAK, H. HE, F. CAPPELLO. *Distributing and Managing Data on Desktop Grids with BitDew*, in "Proceedings of High Performance Distributed Computing (HPDC'08), 3rd Workshop on the Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE-CN'08), Boston, USA", Invited talk, June 2008, p. 63–64.

[46] G. FEDAK, H. HE, O. LODYGENSKY, Z. BALATON, Z. FARKAS, G. GOMBÁS, P. KACSUK, R. LOVAS, A. C. MAROSI, I. KELLEY, I. TAYLOR, G. TERSTYANSZKY, T. KISS, M. CÁRDENAS-MONTES, A. EMMEN, F. ARAUJO. *EDGeS: A Bridge Between Desktop Grids and Service Grids*, in "IEEE computing society Proceeding of the 3rd ChinaGrid Annual Conference, Dunhuang, Gansu, China", Similar versions of the paper presenting the EDGeS project have been published in several conferences., August 2008, p. 1–9.

[47] L. GRIGORI, J. DEMMEL, H. XIANG. *Communication Avoiding Gaussian Elimination*, in "SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, Piscataway, NJ, USA", IEEE Press, 2008, p. 1–12, http://doi.acm.org/10.1145/1413370.1413400.

[48] H. HE, G. FEDAK, F. CAPPELLO. *Large-Scale Bioinformatic Computing on Data Desktop Grid*, in "First IEEE International Scalable Computing Challenge (SCALE 2008) along with CCGRID'08, Lyon, France", Challenge, May 2008.

[49] M. HUGUES, S. G. PETITON. *A Matrix Inversion Method with YML/OmniRPC on a Large Scale Platform*, in "VECPAR", J. M. L. M. PALMA, P. AMESTOY, M. J. DAYDÉ, M. MATTOSO, J. C. LOPES (editors), Lecture Notes in Computer Science, vol. 5336, Springer, 2008, p. 95-108.

[50] P. KACSUK, Z. FARKAS, G. FEDAK. *Towards Making BOINC and EGEE Interoperable*, in "Proceedings of 4th IEEE International Conference on e-Science (e-Science 2008), International Grid Interoperability and Interoperation Workshop 2008 (IGIIW 2008), Indianapolis, USA", December 2008.

[51] F. MANNE, M. MJELDE, L. PILARD, S. TIXEUIL. *A self-stabilizing 2/3-approximation algorithm for the maximum matching problem*, in "Stabilization, Safety, and Security of Distributed Systems, 10th International Symposium (SSS 2008), Detroit", S. S. KULKARNI, A. SCHIPER (editors), Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, November 2008.

[52] T. MASUZAWA, S. TIXEUIL. *Quiescence of Self-stabilizing Gossiping among Mobile Agents in Graphs*, in "Proceedings of 15th International Colloquium on Structural Information and Communication Complexity (Sirocco 2008), Villars-sur-Ollon, Switzerland", Lecture Notes in Computer Science, Springer-Verlag, June 2008.

[53] T. MASUZAWA, S. TIXEUIL. *Strong Stabilization: Bounding Times Affected by Byzantine Processes in Stabilization*, in "Asian Association for Algorithms and Computation annual meeting (AAAC 2008), Hong Kong", April 2008.

[54] A. VORA, M. NESTERENKO, S. TIXEUIL, S. DELAËT. *Universe Detectors for Sybil Defense in Ad Hoc Wireles Networks*, in "International Conference on Stabilization, Safety, and Security (SSS 2008)", Lecture Notes in Computer Science, Springer-Verlag, November 2008.

[55] Y. ZHANG, G. BERGÈRE, S. G. PETITON. *Grid Computing: A Case Study in Hybrid GMRES Method*, in "NPC", J. CAO, M. LI, M.-Y. WU, J. CHEN (editors), Lecture Notes in Computer Science, vol. 5245, Springer, 2008, p. 286-296.

[56] Y. ZHANG, G. BERGÈRE, S. G. PETITON. *Large Scale Parallel Hybrid GMRES Method for the Linear System on Grid System*, in "ISPDC", IEEE Computer Society, 2008, p. 244-249.

Scientific Books (or Scientific Book chapters)

[57] F. BOUABACHE, T. HERAULT, G. FEDAK, F. CAPPELLO. *Checkpointing and Monitoring*, in "A Distributed and Replicated Service for Checkpoint Storage", CoreGRID Books: Making Grids Work, extended version of the paper "Hierarchical Replication Techniques to Ensure Checkpoint Storage Reliability in Grid Environments" at CCGRID 2008, vol. 7, M. Danelutto and P. Fragopoulou and V. Getov, eds. Springer, 2008, p. 293–306.

[58] F. CAPPELLO, H. BAL. *Toward an International "Computer Science Grid"*, in "High Performance Computing and Grids in Action", L. GRANDINETTI (editor), vol. 16, March 2008.

[59] F. CAPPELLO, G. FEDAK, D. KONDO, P. MALÉCOT, A. REZMERITA. *Desktop Grids : From Volunteer Distributed Computing to High Throughput Computing Production Platforms*, in "Handbook of Research on Scalable Computing Technologies", K.-C. LI, C.-H. HSU, L. T. YANG, J. DONGARRA, H. ZIMA (editors), Accepted, to be published in 2009, IGI Global, 2009.

Books or Proceedings Editing

[60] F. CAPPELLO, T. HERAULT, J. DONGARRA (editors). *Special Issue on best papers of EuroPVM/MPI'07, Parallel Computing journal*, Elsevier, 2009.

[61] H. JIN, C.-H. HSU, F. CAPPELLO (editors). *Special Issue on "Peer-to-Peer Grid Technologies", Future Generation Computer Systems journal*, Elsevier, 2009.

Research Reports

[62] A. ANDRZEJAK, A. REINEFELD, F. SCHINTKE, T. SCHÜTT, C. MASTROIANNI, P. FRAGOPOULOU, D. KONDO, P. MALÉCOT, G. COSMIN SILAGHI, L. MOURA SILVA, P. TRUNFIO, D. ZEINALIPOUR-YAZTI, E. ZIMEO. *Grid Architectural Issues: State-of-the-art and Future Trends*, Technical report, n^{o} WHP-0004, Institute on Architectural Issues: Scalability, Dependability, Adaptability, CoreGRID - Network of Excellence, May 2008, http://www.coregrid.net/mambo/images/stories/WhitePapers/whp-0004.pdf.

[63] S. BERNARD, S. DEVISMES, M. G. POTOP-BUTUCARU, S. TIXEUIL. *Bounds for Self-stabilization in Unidirectional Networks*, Technical report, INRIA, May 2008.

[64] F. COSTA, L. SILVA, G. FEDAK, I. KELLEY. *Optimizing the Data Distribution Layer of BOINC with BitTorrent*, Technical report, n^{o} TR-0139, Institute on Architectural issues: scalability, dependability, adaptability, CoreGRID Technical Report, June 2008.

[65] C. COTI, T. HERAULT, F. CAPPELLO. *MPI Applications on Grids: A Topology-Aware Approach*, RR-6633, Rapport de recherche, INRIA, 2008, http://hal.inria.fr/inria-00319241/en/.

[66] S. DELAËT, S. DEVISMES, M. NESTERENKO, S. TIXEUIL. *Snap-Stabilization in Message-Passing Systems*, Research Report, n^{o} 6446, INRIA, February 2008, https://hal.inria.fr/inria-00248465.

[67] J. DEMMEL, L. GRIGORI, M. HOEMMEN, J. LANGOU. *Communication-avoiding parallel and sequential QR and LU factorizations: theory and practice*, LAWN #204, Technical report, n^{o} UCB/EECS-2008-89, University of California Berkeley, EECS Department, 2008.

[68] J. DEMMEL, L. GRIGORI, M. HOEMMEN, J. LANGOU. *Communication-optimal parallel and sequential QR and LU factorizations*, Technical report, n^{o} short version of UCB/EECS-2008-89, arxiv, 2008.

[69] J. DEMMEL, L. GRIGORI, M. HOEMMEN, J. LANGOU. *Implementing communication optimal parallel and sequential QR factorizations*, Technical report, n^{o} short version of UCB/EECS-2008-89, arxiv, 2008.

[70] G. FEDAK, H. HE, F. CAPPELLO. *BitDew: A Programmable Environment for Large-Scale Data Management and Distribution*, RR-6427, Technical report, n^{o} 6427, INRIA, jan 2008, http://hal.inria.fr/inria-00216126/en/.

[71] G. FEDAK, O. LODYGENSKY, Z. FARKAS. *Prototypes of Bridge from Desktop Grids to Service Grids*, Technical report, Deliverable JRA1.1, EDGeS project European Union, 2008.

[72] G. FEDAK, O. LODYGENSKY, Z. FARKAS. *Prototypes of Bridge from Service Grids to Desktop Grids*, Technical report, Deliverable JRA1.2, EDGeS project European Union, 2008.

[73] L. GRIGORI, E. BOMAN, S. DONFACK, T. DAVIS. *Hypergraph-based unsymmetric nested dissection ordering for sparse LU factorization*, RR-6520, Rapport Technique, n^{o} TR 6520, INRIA, February 2008, http://hal.inria.fr/inria-00271394/en/.

[74] L. GRIGORI, J. DEMMEL, H. XIANG. *Communication Avoiding Gaussian Elimination*, RR-6523, Rapport de recherche, INRIA, 2008, http://hal.inria.fr/inria-00277901/en/.

[75] L. GRIGORI, F. NATAF, Q. NIU. *Two sides tangential filtering decomposition*, RR-6554, Rapport de recherche, n^{o} TR 6554, INRIA, 2008, http://hal.inria.fr/inria-00286595/en/.

[76] L. GRIGORI, D. W. NUENTSA, H. XIANG. *Saving Flops in LU Based Shift-and-Invert Strategy*, RR-6553, Rapport de recherche, INRIA, 2008, http://hal.inria.fr/inria-00286417/en/.

[77] L. GRIGORI, H. XIANG. *Kronecker product approximation preconditioners for convection-diffusion model problems*, Submitted to Numerical Linear Algebra, in revision, Rapport Technique, n^{o} TR 6536, INRIA, March 2008, http://hal.inria.fr/inria-00268301/en/.

[78] T. MASUZAWA, S. TIXEUIL. *Quiescence of Self-stabilizing Gossiping among Mobile Agents in Graphs*, RR-6458, Rapport de recherche, INRIA, 2008, http://hal.inria.fr/inria-00260011/en/.

[79] N. MITTON, K. PAROUX, B. SERICOLA, S. TIXEUIL. *Ascending runs in dependent uniformly distributed random variables: Application to wireless networks*, Research Report, n^{o} 6443, INRIA, February 2008, https://hal.inria.fr/inria-00239348.

[80] Q. NIU, L. GRIGORI, P. KUMAR, F. NATAF. *Modified tangential frequency filtering decomposition and its Fourier analysis*, RR-6662, Rapport de recherche, n^{o} TR 6662, INRIA, 2008, http://hal.inria.fr/inria-00324378/en/.

[81] A. Vora, M. Nesterenko, S. Tixeuil, S. Delaët. *Universe Detectors for Sybil Defense in Ad Hoc Wireless Networks*, Technical report, CoRR:abs/0805.0087, 2008.

[82] A. Vora, M. Nesterenko, S. Tixeuil, S. Delaët. *Universe Detectors for Sybil Defense in Ad Hoc Wireles Networks*, Technical report, INRIA, May 2008.

Other Publications

[83] D. Tromeur-Dervout, L. Grigori, B. Philippe, A. Sameh. *Special Issue of Parallel Computing on Parallel Matrix Algorithms and Applications*, Guest Editor, 2008.

References in notes

[84] *EDonkey Homepage*, http://mldonkey.sourceforge.net/, http://mldonkey.sourceforge.net/.

[85] K. Aida, A. Takefusa, H. Nakada, S. Matsuoka, S. Sekiguchi, U. Nagashima. *Performance evaluation model for scheduling in a global computing system*, in "International Journal of High Performance Computing Applications", vol. 14, No. 3, 2000, p. 268-279, http://dx.doi.org/10.1177/109434200001400308.

[86] 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.

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

[88] D. Anderson. *BOINC*, http://boinc.berkeley.edu/.

[89] 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.

[90] 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.

[91] 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.

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

[93] G. Bosilca, A. Bouteiller, F. Cappello, S. Djilali, G. Fedak, C. Germain, T. Herault, P. Lemarinier, O. Lodygensky, F. Magniette, V. Néri, A. Selikhov. *MPICH-V: Toward a Scalable Fault Tolerant MPI for Volatile Nodes, in IEEE/ACM SC 2002*.

[94] 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.

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

[96] 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.

[97] 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.

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

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

[100] 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.

[101] 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.

[102] 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.

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

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

[105] 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.*.

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

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

[108] 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.

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

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

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

[112] 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.

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

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

[115] 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.

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

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

[118] 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.

[119] 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.

[120] J. D. 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.

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

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

[123] 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.

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

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

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

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

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

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

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

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

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

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

[134] 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.

[135] 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.

[136] 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.

[137] 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.

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

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

[140] 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.

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

[142] 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.

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

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

[145] 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.

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

[147] 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.*

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

[149] 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.

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

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