INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team grand-large

Global parallel and distributed computing

Saclay - Île-de-France

Theme : Distributed and High Performance Computing

Table of contents

1. Team **1**
2. Overall Objectives **1**
3. Scientific Foundations **2**
3.1. Large Scale Distributed Systems (LSDS) 2
3.1.1. Computing on Large Scale Global Computing systems 3
3.1.2. Building a Large Scale Distributed System 4
3.1.2.1. The resource discovery engine 4
3.1.2.2. Fault Tolerant MPI 4
3.2. Volatility and Reliability Processing 5
3.3. Parallel Programming on Peer-to-Peer Platforms (P5) 6
3.3.1. Large Scale Computational Sciences and Engineering 7
3.3.2. Experimentations and Evaluations 7
3.3.3. Languages, Tools and Interface 8
3.4. Methodology for Large Scale Distributed Systems 8
3.4.1. Observation tools 8
3.4.2. Tool for scalability evaluations 9
3.4.3. Real life testbeds: extreme realism 9
3.5. High Performance Scientific Computing 9
3.5.1. Efficient linear algebra algorithms 10
3.5.2. Preconditioning techniques 10
4. Application Domains **10**
4.1. Building a Large Scale Distributed System for Computing 10
4.2. Security and Reliability of Network Control Protocols 11
4.3. End-User Tools for Computational Science and Engineering 11
5. Software **12**
5.1. APMC-CA 12
5.2. YML 12
5.3. The Scientific Programming InterNet (SPIN) 13
5.4. V-DS 13
5.5. PVC: Private Virtual Cluster 14
5.6. OpenWP 15
5.7. Parallel solvers for solving linear systems of equations 15
6. New Results **16**
6.1. Exact algorithm for the l1-compressive sensing problem using a modified Dantzig- Wolfe method 16
6.2. Supple: a flexible probabilistic data dissemination protocol for wireless sensor networks 16
6.3. Non-self-stabilizing and self-stabilizing gathering in networks of mobile agents–the notion of speed 16
6.4. Making Population Protocols Self-stabilizing 17
6.5. Self-stabilizing synchronization in population protocols with cover times 17
6.6. Impossibility of consensus for population protocol with cover times 18
6.7. Routing and synchronization in large scale networks of very cheap mobile sensors 18
6.8. Self-Stabilizing Control Infrastructure for HPC 19
6.9. Large Scale Peer to Peer Performance Evaluations 19
6.9.1. Large Scale Grid Computing 19
6.9.2. High Performance Cluster Computing 20
6.9.3. Large Scale Power aware Computing 20
6.10. High Performance Linear Algebra on the Grid 20
6.11. Emulation of Volatile Systems 20

6.12. Exascale Systems 21
6.13. High performance scientific computing 22
6.13.1. Communication avoiding algorithms for linear algebra 22
6.13.2. Combinatorial scientific computing 23
6.13.3. Preconditioning techniques 23
6.13.4. MIcrowave Data Analysis for petaScale computers 24
7. Other Grants and Activities **24**
7.1.1. Activities starting in 2009 24
7.1.2. Other activities 24
8. Dissemination **25**
8.1.1. Research Administration 25
8.1.2. Book/Journal edition 25
8.1.3. Conference Organisation 25
8.1.4. Steering Committee membership 26
8.1.5. Program Committee membership 26
8.1.6. Session Chairing 26
9. Bibliography **26**

1. Team

Research Scientists

Franck Cappello [Research Director, HdR]
Laura Grigori [Researcher CR1, HdR]

Faculty Members

Brigitte Rozoy [Temporary Team Leader, Professor at Paris-Sud University, HdR]
Joffroy Beauquier [Professor at Paris-Sud University, HdR]
Thomas Hérault [Associate Professor at Paris-Sud University, delegated in the INRIA project/team]
Serge Petiton [Professor at University of Science and Technology of Lille, HdR]
Sylvain Peyronnet [Associate Professor at Paris-Sud University, HdR]
Marc Baboulin [Associate Professor at Paris-Sud University, since September 2010]

Technical Staff

Vincent Néri [CNRS Study Engineer]
Johan Oudinet [INRIA Expert Engineer]

PhD Students

Fatiha Bouabache [MESR Grant (LRI)]
Pawan Kumar [INRIA Grant, end September 2010]
Thomas Largillier [MESR Grant (LRI)]
Amina Guermouche [MESR Grant (LRI)]
Alexandre Borghi [MESR Grant (LRI)]
Long Qu [MESR Grant (LRI)]
Simplice Donfack [INRIA Grant]
Amal Khabou [MESR Grant (LRI), since October 2009]
Antoine Baldacci [CIFRE IFP, since November 2009]

Post-Doctoral Fellows

Federico Stivoli [PostDoc INRIA]
Ala Rezmerita [PostDoc INRIA]
Elisabeth Brunet [PostDoc INRIA, end March 2010]
Ke Wang [PostDoc INRIA, end August 2010]
François Lesueur [PostDoc Paris-Sud, end September 2010]
Thomas Ropars [PostDoc INRIA]

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).

Our approach ranges from concepts to experiments, the projects aims at:

1. models and fault-tolerant algorithms, self-stabilizing systems and wireless networks.
2. studying experimentally, and formally, the fundamental mechanisms of LSDS for high performance computing;
3. designing, implementing, validating and testing real software, libraries, middleware and platforms;
4. defining, evaluating and experimenting approaches for programming applications on these platforms.

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

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

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

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

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

3. Scientific Foundations

3.1. Large Scale Distributed Systems (LSDS)

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

3.1.1. Computing on Large Scale Global Computing systems

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

Desktop Grid systems essentially extend the notion of computing beyond the frontier of administration domains. The very first paper discussing this type of systems [78] 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 [67] , Glunix [60] and Mosix [40], to cite a few. A first approach to cross administration domains was proposed by Web Computing projects such as Jet [71], Charlotte [41], Javeline [54], Bayanihan [76], SuperWeb [37], ParaWeb [47] and PopCorn [49]. These projects have emerged with Java, taking benefit of the virtual machine properties: high portability across heterogeneous hardware and OS, large diffusion of virtual machine in Web browsers and a strong security model associated with bytecode execution. Performance and functionality limitations are some of the fundamental motivations of the second generation of Global Computing systems like BOINC [39] and XtremWeb [56]. The second generation of Global Computing systems appeared in the form of generic middleware which allow scientists and programmers to design and set up their own distributed computing project. As a result, we have seen the emergence of large communities of volunteers and projects. Currently, Global Computing systems are among the largest distributed systems in the world. In the mean time, several studies succeeded to understand and enhance the performance of these systems, by characterizing the system resources in term of volatility and heterogeneity and by studying new scheduling heuristics to support new classes of applications: data-intensive, long running application with checkpoint, workflow, soft-real time etc... However, despite these recent progresses, one can note that Global Computing systems are not yet part of high performance solution, commonly used by scientists. Recent researches to fulfill the requirements of Desktop Grids for high demanding users aim at redesigning Desktop Grid middleware by essentially turning a set of volatile nodes into a virtual cluster and allowing the deployment of regular HPC utilities (batch schedulers, parallel communication libraries, checkpoint services, etc...) on top of this virtual cluster. The new generation would permit a better integration in the environment of the scientists such as computational Grids, and consequently, would broaden the usage of Desktop Grid.

The high performance potential of LSDS platforms has also raised a significant interest in the industry. Performance demanding users are also interested by these platforms, considering their cost-performance ratio which is even lower than the one of clusters. Thus, several Desktop Grid platforms are daily used in production in large companies in the domains of pharmacology, petroleum, aerospace, etc.

Desktop Grids share with Grid a common objective: to extend the size and accessibility of a computing infrastructure beyond the limit of a single administration domain. In [57], 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 [77], Gnutella [77] and Kazaa [66], 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 [74], CHORD [79], PASTRY [75] and TAPESTRY [84] 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 [55]. Existing self-stabilizing algorithms use an underlying network that is not compatible with LSDS. They assume that processors know their list of neighbors, which does not fit the P2P requirements. Our work proposes a new model for designing self-stabilizing algorithms without making this assumption, then we design, prove and evaluate overlay networks self-stabilizing algorithms in this model.

3.1.2.2. Fault Tolerant MPI

MPICH-V is a research effort with theoretical studies, experimental evaluations and pragmatic implementations aiming to provide a MPI implementation based on MPICH [69], 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 [51]. 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 [81], pessimistic [38], causal [82]. 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 [44], MPICH-V2 [45], MPICH-Vcausal, MPICH-Vcl [46], 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 ([58]). 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. [68], [80], [55] 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. [65], [64], [42], [43], [59]) for three main reasons:

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

We propose:

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

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

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

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

3.3.1. Large Scale Computational Sciences and Engineering

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

GRID scientific and engineering computing exists already since more than a decade. Since the last few years, the scale of the problem sizes and the global complexity of the applications increase rapidly. The scientific simulation approach is now general in many scientific domains, in addition to theoretical and experimental aspects, often link to more classic methods. Several applications would be computed on world-spread networks of heterogeneous computers using some web-based Application Server Provider (ASP) dedicated to targeted scientific domains. New very strategic domains, such as Nanotechnologies, Climatology or Life Sciences, are in the forefront of these applications. The development in this very important domain and the leadership in many scientific domains will depend in a close future to the ability to experiment very large scale simulation on adequate systems [63]. The P2P scientific programming is a potential solution, which is based on existing computers and networks. The present scientific applications on such systems are only concerning problems which are mainly data independents: i.e. each peer does not communicate with the others.

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

3.3.2. Experimentations and Evaluations

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

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

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

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

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

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

3.3.3. Languages, Tools and Interface

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

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

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

3.4. Methodology for Large Scale Distributed Systems

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

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

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

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

3.4.1. Observation tools

Observation tools are mandatory to understand and extract the main influencing characteristics of a distributed system, especially at large scale. Observation tools produce data helping the design of many key mechanisms in a distributed system: fault tolerance, scheduling, etc. We pursue the objective of developing and deploying a large scale observation tool (XtremLab) capturing the behavior of thousands of nodes participating to popular

Desktop Grid projects. The collected data will be stored, analyzed and used as reference in a simulator (SIMBOINC).

3.4.2. Tool for scalability evaluations

Several Grid and P2P systems simulators have been developed by other teams: SimGrid [50], GridSim [48], Briks [36]. 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 [70] but many of them considers synchronous systems where the system evolution is guided by phases. In the P2P field, ad hoc many simulators have been developed, mainly for routing in DHT. Emulation is another tool for experimenting systems and networks with a higher degree of realism. Compared to simulation, emulation can be used to study systems or networks 1 or 2 orders of magnitude smaller in terms of number of components. However, emulation runs the actual OS/middleware/applications on actual platform. Compared to real testbed, emulation considers conducting the experiments on a fully controlled platform where all static and dynamic parameters can be controlled and managed precisely. Another advantage of emulation over real testbed is the capacity to reproduce experimental conditions. Several implementations/configurations of the system components can be compared fairly by evaluating them under the similar static and dynamic conditions. Grand-Large is leading one of the largest Emulator project in Europe called Grid explorer (French funding). This project has built and used a 1K CPUs cluster as hardware platform and gathers 24 experiments of 80 researchers belonging to 13 different laboratories. Experiments concerned developing the emulator itself and use of the emulator to explore LSDS issues. In term of emulation tool, the main outcome of Grid explorer is the V-DS system, using virtualization techniques to fold a virtual distributed system 50 times larger than the actual execution platform. V-DS aims at discovering, understanding and managing implicit uncoordinated large scale phenomena. Grid Explorer is still in use within the Grid'5000 platform and serves the community of 400 users 7 days a week and 24h a day.

3.4.3. Real life testbeds: extreme realism

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

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

3.5. High Performance Scientific Computing

This research is in the area of high performance scientific computing, and in particular in parallel matrix algorithms. This is a subject of crucial importance for numerical simulations as well as other scientific and industrial applications, in which linear algebra problems arise frequently. The modern numerical simulations coupled with ever growing and more powerful computational platforms have been a major driving force behind a progress in numerous areas as different as fundamental science, technical/technological applications, life sciences.

The main focus of this research is on the design of efficient, portable linear algebra algorithms, such that solving a large set of linear equations or computing eigenvalues and eigenvectors. The characteristics of the matrices commonly encountered in this situations can vary significantly, as are the computational platforms used for the calculations.

Nonetheless two common trends are easily discernible. First, the problems to solve are larger and larger, since the numerical simulations are using higher resolution. Second, the architecture of today's supercomputers is getting very complex, and so the developed algorithms need to be adapted to these new achitectures.

A number of methods and solvers exist for solving linear systems. They can be divided into three classes: direct, iterative or semi-iterative. 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 modelling). Iterative methods as Krylov subspace iterations are less robust, but they are widely used because of their limited memory requirements and good scalability properties on sparse matrices. Preconditioners are used to accelerate the convergence of iterative methods. Semi-iterative methods such as subdomain methods are hybrid direct/iterative methods which can be good tradeoffs.

3.5.1. Efficient linear algebra algorithms

Since 2007, we work on a novel approach to dense and sparse linear algebra algorithms, which aims at minimizing the communication, in terms of both its volume and a number of transferred messages. This research is motivated by technological trends showing an increasing communication cost. Its main goal is to reformulate and redesign linear algebra algorithms so that they are optimal in an amount of the communication they perform, while retaining the numerical stability. The work here involves both theoretical investigation and practical coding on diverse computational platforms. We refer to the new algorithms as communication avoiding algorithms. In our team we have developed communication avoiding algorithms for dense LU and QR factorizations and rank revealing QR factorizations.

The theoretical investigation focuses on identifying lower bounds on communication for different operations in linear algebra, where communication refers to data movement between processors in the parallel case, and to data movement between different levels of memory hierarchy in the sequential case. The lower bounds are used to study the existing algorithms, understand their communication bottlenecks, and design new algorithms that attain them. The results obtained to date concern the LU, QR and rank revealing QR factorizations of dense matrices.

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

3.5.2. Preconditioning techniques

Solving a sparse linear system of equations is the most time consuming operation at the heart of many scientific applications, and therefore it has received a lot of attention over the years. While direct methods are robust, they are often prohibitive because of their time and memory requirements. Iterative methods are widely used because of their limited memory requirements, but they need an efficient preconditioner to accelerate their convergence. In this direction of research we focus on preconditioning techniques for solving large sparse systems.

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 [62]. 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

Participants: Sylvain Peyronnet [correspondant], Joel Falcou, Pierre Esterie, Khaled Hamidouche, Alexandre Borghi.

The APMC model checker implements the state-of-the-art approximate probabilistic model checking methods. Last year we develop a version of the tool dedicated to the CELL architecture. Clearly, it was very pedagogic, but the conclusion is that the CELL is not adapted to sampling based verification methods.

This year we develop, thanks to the BSP++ framework, a version compatible with SPM/multicores machines, clusters and hybrid architectures. This version outperforms all previous ones, thus showing the interest of both these new architectures and of the BSP++ framework.

5.2. 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/

5.3. The Scientific Programming InterNet (SPIN)

Participant: Serge Petiton [correspondant].

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

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

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

5.4. V-DS

Participant: Franck Cappello [correspondant].

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

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

We have made a virtual platform on the GDX cluster with the Vserver virtualization tool. On this platform, we have launched more than 20K virtual machines (VM) with a folding of 100 (100 VM on each physical machine). However, some recent experiments have shown that a too high folding factor may cause a too long execution time because of some problems like swapping. Currently, we are conducting experiments on another platform based on the virtualization tool named Xen which has been strongly improved since 2 years. We expect to get better result with Xen than with Vserver. Recently, we have been using the V-DS version based on Xen to evaluate at large scales three P2P middleware [73].

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

5.5. PVC: Private Virtual Cluster

Participant: Franck Cappello [correspondant].

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

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

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

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

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

5.6. OpenWP

Participant: Franck Cappello [correspondant].

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

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

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

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

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

We are evaluating OpenWP from an industrial application (Amibe), used by the European aerospace company EADS. Amibe is the mesher module of jCAE[1]. Amibe generates a mesh from a CAD geometry in three steps. It first creates edges between every patch of the CAD (mesh in one dimension), then generates a surface mesh for every unfolded patch (mesh in two dimensions) and finally adds the third dimension to the mesh by projecting the 2D mesh into the original CAD surfaces. The first and third operation cannot be distributed. 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.

5.7. Parallel solvers for solving linear systems of equations

Participant: Laura Grigori.

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

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 [61]. 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. Exact algorithm for the l1-compressive sensing problem using a modified Dantzig- Wolfe method

Participants: Alexandre Borghi, Jerome Darbon, Sylvain Peyronnet.

In this work, we consider the l1-Compressive Sensing problem and presents an efficient algorithm that computes an exact solution. The idea consists in reformulating the problem such that it yields a modified Dantzig-Wolfe decomposition that allows to efficiently apply all standard simplex pivoting rules. Experimental results show the superiority of our approach compared to standard linear programming methods.

6.2. Supple: a flexible probabilistic data dissemination protocol for wireless sensor networks

Participants: Aline Carneiro Viana, Thomas Hérault, Thomas LArgillier, Sylvain Peyronnet, Fatiha Zaidi.

We propose a flexible proactive data dissemination approach for data gathering in self-organized Wireless Sensor Networks (WSN). Our protocol Supple, effectively distributes and stores monitored data in WSNs such that it can be later sent to or retrieved by a sink. Supple empowers sensors with the ability to make on the fly forwarding and data storing decisions and relies on flexible and self-organizing selection criteria, which can follow any predefined distribution law. Using formal analysis and simulation, we show that Supple is effective in selecting storing nodes that respect the predefined distribution criterion with low overhead and limited network knowledge.

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

Participants: Joffroy Beauquier, Janna Burman, Julien Clï¿½ment, Shay Kutten.

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

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

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

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

6.4. Making Population Protocols Self-stabilizing

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

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

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

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

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

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

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

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

6.6. Impossibility of consensus for population protocol with cover times

Participants: Joffroy Beauquier, Janna Burman.

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

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

Participants: Joffroy Beauquier, Brigitte Rozoy.

In a next future, large networks of very cheap mobile sensors will be deployed for various applications, going from wild life preserving or environmental monitoring up to medical or industrial system control. Each sensor will cost only a few euros, allowing a large scale deployment. They will have only a few bit of memory, no identifier, weak capacities of computation and communication, no real time clock and will be prone to failures. Moreover such networks will be fundamentally dynamic. The goal of this subject is to develop the basic protocols and algorithms for rudimentary distributed systems for such networks. The studied problems are basic ones, like data collection, synchronization (phase clock, mutual exclusion, group mutual exclusion), fault tolerance (consensus), automatic transformers, always in a context of possible failures. A well known model has already been proposed for such networks, the population protocol model. In this model, each sensor is represented by a finite state machine. Sensors are anonymous and move in an asynchronous way. When two sensors come into range of each other ("meet"), they can exchange information. One of the vast variety of motivating examples for this model is ZebraNet. ZebraNet is a habitat monitoring application in which sensors are attached to zebras in order to collect biometric data (e.g., heart rate, body temperature) and information about their behavior and migration patterns. Each pair of zebras meets from time to time. During such meetings (events), ZebraNet's agents (zebras' attached sensors) exchange data. Each agent stores its own sensor data as well as data of other sensors that were in range in the past. They upload data to a base station whenever it is nearby. It was shown that the set of applications that can be solved in the original model of population protocols is rather limited. Other models (such as some models of Delay/Disruption-Tolerant Networks - DTNs), where each node maintains links and connections even to nodes it may interact with only intermittently, do not seem to suit networks with small memory agents and a very large (and unknown) set of anonymous agents. That is why we enhance the model of population protocols by introducing a notion of "speed". We try to capture the fact that the mobile agents move at different speeds and/or have different communication ranges and/or move according to different patterns and/or visit different places with different frequencies. Intuitively, fast agents which carry sensors with large communication ranges communicate with other agents more frequently than other agents do. This notion is formalized by the notion of cover time for each agent. The cover time of an agent is the unknown number of events (pairwise meetings) in the whole system that occur (during any execution interval) before agent v meets every other agent at least once. The model we propose is somehow validated by some recent statistical results, obtained from empirical data sets regarding human or animal mobility. An important consequence of our approach is that the analytic complexity of the protocols designed in this model is possible, independently of any simulation or experimentation. For instance, we consider the fundamental problem of gathering different pieces of information, each sensed by a different anonymous mobile agent, and where the number of agents is unknown. The goal is to communicate the sensed values (as a multi-set, one value per mobile agent) to a base station, with no duplicates or losses. Gathering is a building block for many monitoring applications in networks of mobile agents. Moreover, the

gathering problem can be viewed as a special case of the routing problem in DTNs, in which there is only one destination, the base station. Then we are able to compute the complexity of solutions we propose, as well as those of solutions used in experimental projects (like ZebraNet), and to compare them. The algorithms we present are self-stabilizing. Such algorithms have the important property of operating correctly regardless of their initial state (except for some bounded period). In practice, self-stabilizing algorithms adjust themselves automatically to any changes or corruptions of the network components (excluding the algorithm's code). These changes are assumed to cease for some sufficiently long period. Self-stabilization is considered for two reasons. First, mobile agents are generally fragile, subject to failures and hard to initialize. Second, systems of mobile agents are by essence dynamic, some agents leave the system while new ones are introduced. Self-stabilization is a well adapted framework for dealing with such situations.

6.8. Self-Stabilizing Control Infrastructure for HPC

Participants: Thomas Hérault, Camille Coti.

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

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

The algorithms are scalable, in the sense that all process memory, number of established communication links, and size of messages are logarithmic with the number of elements in the system. The number of synchronous rounds to build the system is also logarithmic, and the number of asynchronous rounds in the worst case is square logarithmic with the number of elements in the system. Moreover, the salf-stabilizing property of the algorithms presented induce fault-tolerance and self-adaptivity. Performance evaluation based on simulations predicts a fast convergence time (1/33s for 64K nodes), exhibiting the promising properties of such self-stabilizing approach.

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

6.9. Large Scale Peer to Peer Performance Evaluations

Participant: Serge Petiton.

6.9.1. 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 [72]. We adapted [52] 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.

6.9.2. 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 [83] 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.

6.9.3. 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 [53]. 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.

6.10. High Performance Linear Algebra on the Grid

Participants: Thomas Hérault, Camille Coti.

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

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

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

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

6.11. Emulation of Volatile Systems

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

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

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

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

6.12. Exascale Systems

Participant: Franck Cappello.

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

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

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

6.13. High performance scientific computing

Participants: Laura Grigori, Guy Atenekeng, Simplice Donfack, Amal Khabou, Federico Stivoli.

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

6.13.1. Communication avoiding algorithms for linear algebra

This research focuses on developing new algorithms for linear algebra problems, that minimize the required communication, in terms of both latency and bandwidth. We have introduced in 2008 two communication avoiding algorithms for computing the LU and QR factorizations, that we refer to as CALU and CAQR (joint work with J. Demmel and M. Hoemmen from U.C. Berkeley, J. Langou from C.U. Denver, and H. Xiang then at INRIA). Since then, we have also designed a communication avoiding algorithm for rank revealing QR. In addition, we have also extended theoretical lower bounds to sparse Cholesky factorization and identified algorithms that attain these bounds and so minimize communication. The communication avoiding algorithms are now studied by several other groups, including groups at INRIA. In particular they study their implementation on modern architectures.

In [22] we study two algorithms based on CAQR and CALU that are adapted to multicore architectures. They combine ideas to reduce communication from communication avoiding algorithms with asynchronism and dynamic task scheduling. For matrices that are tall and skinny, that is, they have many more rows than columns, the two algorithms outperform the corresponding algorithms from Intel MKL vendor library on a dual-socket, quad-core machine based on Intel Xeon EMT64 processor and on a four-socket, quad-core machine based on AMD Opteron processor. For matrices of size $m = 10^5$ and $m = 10^6$, for n varying from 10 to 1000, multithreaded CALU outperforms the corresponding routine dgetrf from Intel MKL library up to a factor of 2.3 and the corresponding routine dgetrf from AMD (ACML library) up to a factor of 5. Multithreaded CAQR outperforms by a factor of 5.3 the corresponding dgeqrf routine from MKL library. For square matrices, CALU is slightly slower than MKL dgetrf routine when $m = n < 5000$, while for $m = n = 10^4$ it is slightly faster than this routine up to a factor of 1.5 and CAQR is less performant than the corresponding routine in the other librairies.

In [32] we focus on numerical properties of CALU. To decrease the communication required in the LU factorization, CALU uses a new pivoting strategy, referred to as tournament pivoting, that may lead to a different row permutation than the classic LU factorization with partial pivoting. We have further investigated the numerical stability of CALU. Our numerical results show that tournament 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. In particular we have shown that CALU is equivalent to performing GEPP on a larger matrix formed by entries of the input matrix (sometimes slightly perturbed) and zeros.

The communication avoiding ideas have been used in [12] to decrease the number of floating point operations performed in shift and invert strategies that use the LU factorization.

In [24] we have focused on communication optimal algorithms for sparse linear algebra. Previous work has shown that a lower bound on the number of words moved between large, slow memory and small, fast memory of size M by any conventional (non-Strassen like) direct linear algebra algorithm (matrix multiply, the LU, Cholesky, QR factorizations, ...) is $\Omega(\#flops/\sqrt{(M)})$. This holds for dense or sparse matrices. There are analogous lower bounds for the number of messages, and for parallel algorithms instead of sequential algorithms.

Our goal here is to find algorithms that attain these lower bounds on interesting classes of sparse matrices. We focus on matrices for which there is a lower bound on the number of flops of their Cholesky factorization. Our Cholesky lower bounds on communication hold for any possible ordering of the rows and columns of the matrix, and so are globally optimal in this sense. For matrices arising from discretization on two dimensional and three dimensional regular grids, we discuss sequential and parallel algorithms that are optimal in terms of communication. The algorithms turn out to require combining previously known sparse and dense Cholesky algorithms in simple ways.

6.13.2. Combinatorial scientific computing

In [11] we discuss a hypergraph-based unsymmetric nested dissection ordering for reducing the fill-in incurred during Gaussian elimination. We refer to this approach as HUND. It has several important properties. It takes a global perspective of the entire matrix, as opposed to local heuristics. It takes into account the asymmetry of the input matrix by using a hypergraph to represent its structure. It is suitable for performing Gaussian elimination in parallel, with partial pivoting. This is possible because the row permutations performed due to partial pivoting do not destroy the column separators identified by the nested dissection approach. The hypergraph nested dissection approach is essentially equivalent to graph nested dissection on the matrix A^TA, but we only need the original matrix A and never form the usually denser matrix A^TA. The usage of hypergraphs in our approach is fairly standard and HUND can be implemented by calling an existing hypergraph partitioner that uses recursive bisection. Our implementation uses local reordering (CCOLAMD) to further improve the ordering. We also explain how weighted matching (MC64) can be used in this context.

Experimental results on 27 medium and large size matrices with highly unsymmetric structures compare our approach to four other well-known reordering algorithms. The results show that it provides a robust reordering algorithm, in the sense that it is the best or close to the best (often within 10%) of all the other methods, in particular on matrices with highly unsymmetric structures.

6.13.3. 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. This research is performed in the context of ANR PETAL project (2009-2010), which is followed by ANR PETALh project (2011-2012). The participants at INRIA Saclay in 2010 are L. Grigori, P. Kumar and K. Wang.

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 in the previous year we have investigated further their properties. In [13], a modified tangential frequency filtering decomposition (MTFFD) preconditioner is proposed. The optimal order of the modification and the optimal relaxation parameter is determined by Fourier analysis. With the choice of optimal order of modification, the Fourier results show

that the condition number of the preconditioned matrix is $\mathcal{O}(h^{-\frac{2}{3}})$, and the spectrum distribution of the preconditioned matrix can be predicted by the Fourier results. The performance of MTFFD preconditioner is compared with tangential frequency filtering (TFFD) preconditioner on a variety of large sparse matrices arising from the discretization of PDEs with discontinuous coefficients. The numerical results show that the MTFFD preconditioner is much more efficient than the TFFD preconditioner.

Our research also focused on developing preconditioning techniques for irregular grids that are suitable for parallel computing. In [33], we introduce a class of recursive multilevel preconditioning strategies suited for solving large sparse linear systems of equations on modern day architectures. They are based on a reordering of the input matrix into a nested bordered block diagonal form, which allows a nested formulation of the preconditioners. The first one, which we refer to as nested SSOR (NSSOR), requires only the factorization of diagonal blocks at the innermost level of the recursive formulation. Hence, its construction is embarassingly parallel, and the memory requirements are very limited. Next two are nested versions of Modified ILU preconditioner with row sum (NMILUR) and colsum (NMILUC) property. We compare these methods in terms of iteration number, memory requirements, and overall solve time, with ILU(0) with natural ordering and nested dissection ordering, and MILU. We find that NSSOR compares favorably with ILU(0) with nested dissection ordering, while NMILUR and NMILUC outperform the other methods for certain matrices in our test set.

It is proved that the NSSOR method is convergent when the input matrix is SPD. The preconditioners are designed to be suitable for parallel computing.

6.13.4. MIcrowave Data Analysis for petaScale computers

In [34] we describe an algorithm for computing an inverse spherical harmonic transform suitable for Graphics Processing Unit (GPU). We use CUDA and base our implementation on a FORTRAN90 routine included in a public domain parallel package, S2HAT. We focus our attention on the two major sequential steps involved in the transforms computation, retaining the efficient parallel framework of the original code. We detail optimization techniques used to enhance the performance of the CUDA-based code and compare them with those implemented in the FORTRAN90 version. We also present performance comparisons for a single CPU plus GPU unit with the S2HAT code running on either a single or 4 processors. In particular we find that use of the latest generation of GPUs, such as NVIDIA GF100 (Fermi), can accelerate the spherical harmonic transforms by as much as 18 times with respect to S2HAT executed on one core, and by as much as 5.5 with respect to S2HAT on 4 cores, with the overall performance being limited by the Fast Fourier transforms.

The work presented here has been performed in the context of the Cosmic Microwave Background simulations and analysis. However, we expect that the developed software will be of more general interest and applicability.

7. Other Grants and Activities

7.1. Regional, National and International Actions

7.1.1. Activities starting in 2009

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

7.1.2. Other activities

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

- **ANR Cosinus project MIDAS - MIcrowave Data Analysis for petaScale computers** December 2009 - December 2012 (http://www.apc.univ-paris7.fr/APC_CS/Recherche/Adamis/MIDAS09/index.html). Collaboration with APC, University Paris 7 and Lawrence Berkeley Laboratory. This is an interdisciplinary project devised to bring together cosmologists, computational physicists, computer scientists and applied mathematiciancs to face the challenge of the tremendous volume of data as anticipated from current and forthcoming Cosmic Microwave Background (CMB) experiments. (Laura Grigori, Coordinator for INRIA Saclay, F. Cappello, J. Falcou, T. Hérault, S. Peyronnet)
- **ANR Cosinus project PETAL- PETascale ALgorithms for preconditioning for scientific applications** January 2009- December 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. Hérault, S. Peyronnet (LRI) and two foreign collaborators: J. Demmel from UC Berkeley and J. Darbon from UC Los Angeles.
- **INRIA associated team COALA with Prof. J. Demmel, UC Berkeley, 2010-2013**. This project is proposed in the context of developing Communication Optimal Algorithms for Linear Algebra. The funding covers visits in both directions.

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

8. Dissemination

8.1. Services to the Scientific Community

8.1.1. Research Administration

- Brigitte Rozoy, past position of University Vice-President and position at the research ministry.

8.1.2. Book/Journal edition

- Joffroy Beauquier, PPL, Parallel Processing Letters
- Franck Cappello, Cluster Computing Journal, Springer, Netherlands, since 2006
- Franck Cappello, Journal of Grid Computing, Springer Netherlands, since 2003
- Franck Cappello, Journal of Grid and utility computing, Inderscience, since 2004

8.1.3. Conference Organisation

- Franck Cappello, Program co-chair, IEEE CCGRID'2009
- Franck Cappello, Area co-chair, IEEE-ACM Supercomputing '09
- Franck Cappello, General co-chair, IPDPS/PCGrid'09
- Sylvain Peyronnet, Organizing committee, CSDM 2010

- Sylvain Peyronnet, Organizing committee, CSDM 2011
- Laura Grigori, Member of Organizing Committee of 5th SIAM Workshop on Combinatorial Scientific Computing (CSC).

8.1.4. Steering Committee membership

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

8.1.5. Program Committee membership

- Joffroy Beauquier SSS 2009
- Joffroy Beauquier SAR/SSI 2009
- Joffroy Beauquier Sirocco 2009
- Joffroy Beauquier Sirocco 2010
- Sylvain Peyronnet, ISVC 2010 %item Franck Cappello, 3rd Workshop on
- Laura Grigori, Supercomputing 2010
- Laura Grigori, IEEE International Parallel and Distributed Processing Symposium, IPDPS 2011.
- Laura Grigori, Parallel Algorithms and Software for Sparse Linear Algebra Computations, Special session of PDP 2010.
- Laura Grigori, Parallel Matrix Algorithms and Applications PMAA2010.

8.1.6. Session Chairing

- Sylvain Peyronnet, CSDM 2010, , 29/10/2010

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", November 2006, vol. 20, n^{o} 4, p. 481-494.

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

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

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

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

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

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

Publications of the year

Articles in International Peer-Reviewed Journal

[8] E. AGULLO, C. COTI, T. HÉRAULT, J. LANGOU, S. PEYRONNET, A. REZMERITA, F. CAPPELLO, J. DONGARRA. *QCG-OMPI: MPI applications on grids*, in "Future Generation Computer Systems", 2010.

[9] A. BORGHI, J. DARBON, S. PEYRONNET. *Exact optimization for the l1-Compressive Sensing problem using a modified Dantzig-Wolfe method*, in "Theoretical Computer Science", 2010, vol. In Press, Accepted Manuscript, - [*DOI :* DOI: 10.1016/J.TCS.2010.10.032], http://www.sciencedirect.com/science/article/B6V1G-51J9DRR-1/2/c10fbd8b2e027fefaed3059aad668d9c.

[10] J. CLÉMENT, S. MESSIKA, B. ROZOY. *Observer des Algorithmes Auto-Stabilisants : le coût*, in "Technique et Science Informatiques", 2010.

[11] L. GRIGORI, E. BOMAN, S. DONFACK, T. DAVIS. *Hypergraph-based unsymmetric nested dissection ordering for sparse LU factorization*, in "SIAM Journal on Scientific Computing", 2010.

[12] L. GRIGORI, D. NUENTSA, H. XIANG. *Saving flops in LU-based Shift and Invert Strategies*, in "Journal of Computational and Applied Mathematics", 2010, vol. 234, n^{o} 12, p. 3216 - 3225.

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

International Peer-Reviewed Conference/Proceedings

[14] E. AGULLO, C. COTI, J. DONGARRA, T. HÉRAULT, J. LANGOU. *QR Factorization of Tall and Skinny Matrices in a Grid Computing Environment*, in "24th IEEE International Parallel & Distributed Processing Symposium (IPDPS'10)", Atlanta, Ga, April 2010, to appear.

[15] J. BEAUQUIER, J. BURMAN. *Self-stabilizing Synchronization in Mobile Sensor Networks with Covering*, in "DCOSS", 2010, p. 362-378.

[16] J. BEAUQUIER, J. BURMAN, J. CLÉMENT, S. KUTTEN. *On utilizing speed in networks of mobile agents*, in "PODC", 2010, p. 305-314.

[17] A. BORGHI, J. DARBON, S. PEYRONNET. *L1 compressive sensing: Exact optimization a la Dantzig-Wolfe*, in "Signal Processing Systems (SIPS), 2010 IEEE Workshop on", IEEE, 2010, p. 7–12.

[18] A. BORGHI, J. DARBON, S. PEYRONNET, T. F. CHAN, S. OSHER. *A Compressive Sensing Algorithm for Many-Core Architectures*, in "Advances in Visual Computing - 6th International Symposium, ISVC 2010", 2010, p. 678-686.

[19] F. BOUABACHE, T. HÉRAULT, S. PEYRONNET, F. CAPPELLO. *Planning Large Data Transfers in Institutional Grids*, in "Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing", Washington, DC, USA, CCGRID '10, IEEE Computer Society, 2010, p. 547–552, http://dx.doi.org/10.1109/CCGRID.2010.68.

[20] A. BOURKI, G. CHASLOT, M. COULM, V. DANJEAN, H. DOGHMEN, T. HÉRAULT, J.-B. HOOCK, A. RIMMEL, F. TEYTAUD, O. TEYTAUD, P. VAYSSIÈRE, Z. YU. *Scalability and Parallelization of Monte-Carlo Tree Search*, in "The International Conference on Computers and Games 2010", Japon Kanazawa, 2010, http://hal.inria.fr/inria-00512854/en.

[21] A. CARNEIRO VIANA, T. HÉRAULT, T. LARGILLIER, S. PEYRONNET, F. ZAÏDI. *Supple: a flexible probabilistic data dissemination protocol for wireless sensor networks*, in "Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems", New York, NY, USA, MSWIM '10, ACM, 2010, p. 385–392, http://doi.acm.org/10.1145/1868521.1868586.

[22] S. DONFACK, L. GRIGORI, A. KUMAR GUPTA. *Implementing communication-avoiding LU and QR factorizations for tall and skinny matrices on multicore architectures*, in "Proceedings of 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2010)", Atlanta, USA, April 19-23 2010.

[23] G. FEDAK, J.-P. GELAS, T. HÉRAULT, V. INIESTA, D. KONDO, L. LEFÈVRE, P. MALECOT, L. NUSSBAUM, A. REZMERITA, O. RICHARD. *DSL-Lab: a Low-power Lightweight Platform to Experiment on Domestic Broadband Internet*, in "9th International Symposium on Parallel and Distributed Computing (ISPDC'2010)", Turquie Istanbul, Jul 2010, http://hal.inria.fr/inria-00502888/en.

[24] L. GRIGORI, P.-Y. DAVID, J. DEMMEL, S. PEYRONNET. *Brief announcement: Lower bounds on communication for direct methods in sparse linear algebra*, in "ACM Symposium on Parallelism in Algorithms and Architectures", 2010.

[25] L. GRIGORI, P.-Y. DAVID, J. DEMMEL, S. PEYRONNET. *Brief announcement: Lower bounds on communication for sparse Cholesky factorization of a model problem*, in "Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures", New York, NY, USA, SPAA '10, ACM, 2010, p. 79–81, http://doi.acm.org/10.1145/1810479.1810496.

[26] K. HAMIDOUCHE, A. BORGHI, P. ESTERIE, J. FALCOU, S. PEYRONNET. *Three High Performance Architectures in the Parallel Approximate Probabilistic Model Checking Boat*, in "Proceedings of the 9th International Workshop on Parallel and Distributed Methods in verifiCation", 2010.

[27] S. HEMON, T. LARGILLIER, S. PEYRONNET. *Partial Ranking of Products for Recommendation Systems*, in "E-Commerce and Web Technologies: 11th International Conference, EC-Web 2010", Bilbao, Spain, September 1-3 2010, p. 265–277.

[28] T. LARGILLIER, S. PEYRONNET. *Lightweight Clustering Methods for Webspam Demotion*, in "Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on", Los Alamitos, CA, USA, IEEE Computer Society, 2010, p. 98-104 [*DOI :* 10.1109/WI-IAT.2010.243].

[29] T. LARGILLIER, S. PEYRONNET. *Using patterns in the behavior of the random surfer to detect Webspam beneficiaries*, in "1st International Workshop on Web Intelligent Systems and Services", 2010.

[30] T. LARGILLIER, G. PEYRONNET, S. PEYRONNET. *SpotRank: a robust voting system for social news websites*, in "Proceedings of the 4th workshop on Information credibility", New York, NY, USA, WICOW '10, ACM, 2010, p. 59–66, http://doi.acm.org/10.1145/1772938.1772951.

[31] F. LESUEUR, A. REZMERITA, T. HÉRAULT, S. PEYRONNET, S. TIXEUIL. *SAFE-OS: a Secure and Usable Desktop Operating System*, in "Proceedings of CRiSIS 2010", Montréal, Canada, October 2010.

Research Reports

[32] L. GRIGORI, J. DEMMEL, H. XIANG. *CALU: a communication optimal LU factorization algorithm*, UCB/EECS-2010-29, 2010.

[33] L. GRIGORI, P. KUMAR, F. NATAF, K. WANG. *A class of multilevel parallel preconditioning strategies*, INRIA, Oct 2010, n^{o} RR-7410, http://hal.inria.fr/inria-00524110/en.

[34] I. HUPCA, J. FALCOU, L. GRIGORI, R. STOMPOR. *Spherical harmonic transform with GPUs*, INRIA, Oct 2010, n^{o} RR-7409, http://hal.inria.fr/inria-00522937/en.

[35] P. KUMAR, L. GRIGORI, Q. NIU, F. NATAF. *Fourier Analysis of Modified Nested Factorization Preconditioner for Three-Dimensional Isotropic Problems*, Jan 2010, http://hal.inria.fr/inria-00448291/en.

References in notes

[36] K. AIDA, A. TAKEFUSA, H. NAKADA, S. MATSUOKA, S. SEKIGUCHI, U. NAGASHIMA. *Performance evaluation model for scheduling in a global computing system*, in "International Journal of High Performance Computing Applications", 2000, vol. 14, No. 3, p. 268-279, http://dx.doi.org/10.1177/109434200001400308.

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

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

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

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

[41] 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.

[42] 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.

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

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

[45] A. BOUTEILLER, F. CAPPELLO, T. HÉRAULT, G. KRAWEZIK, P. LEMARINIER, F. MAGNIETTE. *MPICH-V2: a Fault Tolerant MPI for Volatile Nodes based on Pessimistic Sender Based Message Logging*, November 2003, in IEEE/ACM SC 2003.

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

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

[52] 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.

[53] 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.

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

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

[56] 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.

[57] 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.

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

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

[60] 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", 1998, vol. 28, n^{o} 9, p. 929–961.

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

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

[63] 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.

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

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

[66] 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.

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

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

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

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

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

[72] 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.

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

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

[75] 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.

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

[77] 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.

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

[79] 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.

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

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

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

[83] 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.

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