Action de Recherche Amont

MASSE de DONNEES

Scientific Description of the project

1 Goal and context

The project acronym is ALPAGE, which stands for *ALgorithmique des Plates-formes À Grande Échelle* and can be translated by *Algorithms for Large-Scale Platforms*.

The project gathers several teams with different and complementary expertise, ranging from algorithm design and scheduling techniques to macro-communication primitives and routing protocols, and to peer-to-peer architectures and distributed systems.

1.1 General context

The recent evolutions in computer networks technology, as well as their diversification, yield a tremendous change in the use of these networks: applications and systems can now be designed at a much larger scale than they used to be. This scaling evolution conerns the amount of data, the number of computers, the number of users, and the geographical diversity of theses users. This race towards *large scale* computing has two major implications. First, new opportunities are offered to applications, in particular as far as scientific computing, data bases, and file sharing are concerned. Second, a large number of parallel or distributed algorithms developed for medium size systems cannot be run on large scale systems without a significant degradation of their performances. In fact, one must probably relax the constraints that a system should satisfy in order to run at a larger scale. For example, coherence protocols designed for distributed applications are too demanding in term of both message and time complexity, and must therefore be adapted to run at a larger scale. In addition, most distributed systems deployed today are characterized by a high dynamism of their entities (participants can join and leave at will), a potential instability of the large scale networks (on which concurrent applications are running), and the increasing probability of failure. Therefore, as the size of the system increases, it becomes necessary that it adapts automatically to the changes of its components, requiring a self-organization of the system when facing the arrival and departure of participants, data, or resources.

As a consequence, it becomes crucial to be able to understand and model the behavior of large scale systems, to efficiently exploit these infrastructures. More specifically, it is essential to design dedicated algorithms handling a large amount of users and/or data.

1.2 Our project within ACI Grandes Masse de Données

The scientific community learnt a lot from parallel computation, in particular to handle heterogeneous resources, both for computing time and communication bandwidth. The understanding of parallel computation was achieved thanks to a fine grained modeling of the physical network and of its communication primitives. However, the experience acquired over the last decade is not sufficient to tackle new problems occurring in large scale systems. Effectively, (1) it is unclear how the current models can be adapted to large scale systems, and (2) the current methodology requires the use of (at least partially) centralized subroutines that cannot be considered in large scale systems. More specifically, these subroutines assume the ability of gathering all the information regarding the network at a single node (topology, resource performances, etc.). This assumption is unrealistic in a general purpose large size platform, in which the nodes are unstable, and whose resource characteristics can vary abruptly over time. Basiacally, central control should be avoided as much as possible in such systems. Moreover, the proposed solutions for small to medium

size, stable, and dedicated environments do not satisfy the minimal requirements for self-organization and fault-tolerance, two properties that are unavoidable in a large scale context.

The tremendous success of peer-to-peer (P2P) applications for file sharing [14, 13, 18] led to the design of a large number of dedicated protocols, that run in a fully distributed environment. These protocols support local decisions, and the P2P services (publication, search, node insertion, etc.) are supported by a (virtual) overlay network connecting the peers over the Internet. Up to some extent, the current P2P protocols [22, 28, 23] are stable and fault-tolerant, as witnessed by their wide and intensive usage. Nevertheless, the P2P protocols are been initially designed for file sharing applications [24] and also studied in the context of general purpose distributed applications. Yet, such protocols have not been optimized for scientific applications, neither they are adapted to sophisticated data-base applications.

The main objective of our project is to federate a group of partners who have the required skills to build a bridge between, on the one hand, the world of parallel applications and grid computing, and, on the other hand, the world of large scale decentralized systems and P2P applications. We aim at designing algorithms grounded on realistic models for which it will be possible to control the performances of the algorithms, and to maintain their stability under brutal changes of the platform. We will mostly focus on the design of robust decentralized algorithms for problems that, although quite simple in essence, possess a large applicative spectrum: broadcast and multicast of large data streams, distribution of independent tasks over a large network, elementary scheduling problems, etc.

The aforementioned objectives naturally fits in ACI "Grandes Masses de Données". Effectively, our objectives envision large platforms for which the startup times induced by the deployment of the applications, and the large latency of the long-distance communications, restrict their use to applications that manipulate large volumes of data. Our project is therefore naturally fitting in Action "Efficient Algorithms" from the 2005 call, that stipulates:

> *Algorithmic has a crucial role to play for all that concerns the management of a large amount of data, and a serious research effort has to be done to improve the current (slowly convergent) algorithms, and to adapt them to multi-scale objects in the context of manipulating huge data depository. In particular, it was observed that defining abstract objects out of the experience acquired while facing practical problems enabled to design parallel and/or distributed algorithms, and an in-depth analysis of these objects should open new directions for further improvements, especially in the field of parallel programming, and computational grid. The study of distributed data structures, and the study of the communication costs, are two examples of studies from which the management of large masses of data should take benefit.*

1.3 Targeted platforms

This project is focussing on *algorithms for large scale distributed systems*, encompassing heterogeneous decentralized platforms. Here are some definitions that we will use throughout this proposal:

Stable platforms: These platforms are characterized by their large size, but their limited heterogeneousness (processor speed, bandwidth of the communication links, etc.). Moreover, the characteristics of the platform are supposed to be static (or to change at very slow rate) over time.

A typical example of such platforms is a computational grid dedicated to one user, or to a small group of well identified users, on which are running a small number of different applications, also well identified. Another example is a set of resources provided by a team of users, and shared by these users. These resources are assumed to be temporary dedicated to a specific application, or to a well identified small set of applications. Obviously, these two examples differ in the sense that although assuming a global knowledge of the platform characteristics is a reasonable assumption for a grid, it is an unrealistic assumption in the case of shared resources that can potentially evolve over time, and whose owner(s) can also potentially evolve as well.

Semi-stable platforms: These platforms are characterized by their large size, and their heterogeneousness. The resources of these platforms (topology, message routes, etc.) and their characteristics are assumed to be known even though they may change over time.

A typical example of such platform is a general purpose computational grid, or a set of resources provided by a team of users that can change significantly over time.

Unstable platform: These platforms are characterized by their large size, their high heterogeneousness, and their characteristics that can evolve rapidly over time. In particular, resources and users can join and leave at will. The size of an unstable platform is generally by one or two orders of magnitude larger than those of stable or semi-stable platforms. It is therefore very difficult, and perhaps even impossible, to get a correct view of the characteristics of an unstable platform at any time (topology, performances of the computational units, etc.). As a consequence, centralized solutions should be avoided as any central entity would immediately create a bottleneck in the system.

A typical example of such platforms are the ones on which Internet file sharing systems have been deployed.

1.4 Targeted applications

As the design of algorithms dedicated to large scale platforms is still at its early days, it is unrealistic to investigate this domain as a whole. Instead, it is preferable to start the investigation by considering simple problems, at the core of the domain. This project will therefore focus on *elementary procedures* such as data broadcasting and multicasting, the deployment of regular applications, the scheduling of these applications' tasks, etc. The applications concerned by this project are thus, mainly:

- Broadcasting and gathering. Specifically, we will investigate these problems to validate our platform models.
- Applications consisting of a large set of independent tasks, of identical (or very similar) characteristics: amount of computation, amount of data, etc. In particular, we will focus on image analysis and particle collisions [29] that will be used as test applications for our models and solutions.
- Applications consisting of a large set of tasks sharing specific files, as *parameter sweep* applications or data-base applications.

1.5 Objectives

In order to provide efficient solutions for large scale systems design, we will concentrate our efforts on the following four complementary axes:

Topic 1: Large scale distributed platform modeling;

Topic 2: Overlay networks topologies;

Topic 3: Scheduling for regular parallel applications;

Topic 4: Scheduling for file-sharing applications.

Our priority will be to provide solutions that are valid and feasible in the framework of large scale platforms (i.e., they must be scalable and free of any type of bottleneck). Of course, these solutions must be adaptive, to react to the inherent dynamics of the considered systems.

The remaining of the document describes our main objectives regarding each of these four topics.

2 Project description

2.1 Models

An important part of our work will be devoted to the modeling of target platforms defined above (stable, semi-stable and unstable). More precisely, for each type of platform, our aim is to define a realistic, tractable and instantiable model.

- Realistic: for each proposed model, we aim to provide experimental results proving that the model corresponds to observed phenomena (contentions, bandwidths, interferences...)
- Tractable: in order to be considered as tractable, the model should enable to derive efficient polynomial time (optimal or approximation) algorithms for a set of fundamental problems. More precisely, Broadcast and Multicast will be considered as testbeds to validate the tractability of candidate models.
- Instantiable: one should be able to evaluate the different parameters of the model in real time. In particular, this requires platform discovery mechanisms, for both platform topology and the actual parameters (bandwidths, contentions) of the model. Real time means that the time needed to instantiate the parameters of the models should be small with respect to the evolution speed of the parameters of the platform (that will obviously not be the same for stable, semi-stable and unstable platforms).

More generally, the models we will consider strongly depend on the characteristics of the platform. On the one hand, in the case of stable platforms, costly mechanisms can be considered, since this cost will be amortized over time. On the other hand, in the case of unstable platforms, where connections and disconnections are frequent, and where communication links are not dedicated, it will be necessary to rely on less sophisticated mechanisms. Proposed algorithms for broadcast and multicast operations will strongly depend on the model, and therefore on the type of the considered platform.

2.1.1 Stable platforms

Task 1.1: Modeling and discovery mechanisms for stable platforms Recently, several models [9, 5, 6, 16, 7] have been proposed in the literature for modeling the topology and the resources of large scale distributed stable platforms. The important issues concern their ability to describe heterogeneous resources (both processing and communication resources) and the interferences between simultaneous activities. In particular, the following questions must be answered: is it possible, for a processor to be involved in more than one incoming (and/or) one outgoing communication? is it important to take into account the interferences that can occur between several distinct point to point communications? Are processing performances affected by simultaneous incoming (and/or) outgoing communications?

The second part of this task is devoted to platform discovery mechanisms. Recently, several toolboxes (ENV [26, 25] or ALNeM [20]) have been developed to find out the topology (the map) of stable platforms. It is worth noting that the aim of these toolboxes is not to discover the actual physical topology of a platform at the network packet level (that would lead to huge maps that could hardly be used to derive efficient algorithms), but rather to obtain a description of the topology of the platform at the application layer. Such a map should nevertheless be able to model the interferences than can be observed at the application layer. These maps, simple and compact, may be suitable to derive algorithms (for broadcast and multicast) and may also be scalable. Nevertheless, algorithms for building such maps are still incomplete and too slow to be considered for large scale platforms. Our aim is to make them suitable for finding the topology of large scale distributed platforms, in reasonable time (since the platforms considered in this part are stable, the running time of the discovery mechanism is not crucial, as far as it does not exceed a few hours for a large platform).

Task 1.2: Tractability analysis for the model of stable platforms In order to validate the tractability of proposed model, we will consider the broadcast of a large message over a large scale distributed but stable platform. Recently, several polynomial time algorithms have been developed for this problem [8]. These algorithms are asymptotically optimal, when the size of the message (or equivalently the number of messages) becomes arbitrarily large (more precisely, they achieve optimal throughput for broadcast operation). The first step will consist in adapting these algorithms to the platform model obtained in Task 1.1. Moreover, these algorithms require to store at a single point all topology and performance information in order to compute the set of weighting trees that will be used to broadcast messages from any source. Since this set of trees may be to large to be stored on each node for any source, algorithms for compacting routing tables will also be required.

2.1.2 Semi-stable platforms

Task 1.3: Modeling and discovery mechanisms for semi-stable platforms. In the case of semi-stable platforms, the topology of the platform does not change during the execution of an application, but the network connecting participating nodes is not dedicated, and communications involved by other transfers may affect link bandwidths. Automatic topology discovery mechanisms developed in Task 1.1 can therefore still be used in the context of semi-stable platforms, but mechanisms used to determine the actual capacities of network links (like NWS [31]) at runtime should be added. Moreover, since the topology determined in Task 1.1 does not match the actual physical topology but rather a description of the topology at the application layer, it is important to determine whether changes in link performances may affect the resulting topology (and in this case, find out how to update the map).

Task 1.4: Tractability analysis for the model of semi-stable platforms. If mechanisms for platform discovery do not require fundamental modifications when moving from stable to semi-stable platforms, algorithms for broadcasting and multicasting must be strongly reconsidered. In particular, it is not reasonable to assume that all the characteristics of the platform can be centralized at a given node to compute the set of broadcast trees, since the characteristics of the platform (and therefore the set of trees) continuously change. Therefore, it is necessary to build fully distributed algorithms for broadcasting, and these algorithms should be robust against small perturbation in network links performances. To achieve this tasks, decentralized algorithms for computing multi-flows [3, 4] and network coding [2] techniques may be used.

2.1.3 Unstable Platforms

Task 1.5: Modeling and discovery mechanisms for unstable platforms. In the case of unstable platforms, due to frequent connections and disconnections, the topology of the platform may change dramatically. It is therefore necessary to design platform discovery mechanisms that strongly differ from those proposed in Tasks 1.1 and 1.3. In order to deal with hosts volatility, peer to peer networks rely on a logical topology (overlay network) in order to ensure robustness, fault tolerance and efficient routing. In general, the overlay network is not directly related to the underlying physical network although a few peer to peer overlay have been optimized according to the physical topology (Pastry [23] and Land [1]). Ignoring the physical topology may have a strong impact on the performance the applications running on the platform. This is typically the case of the applications considered in Tasks 3.* and 4.*. Typically these applications have predictable communication schemes that should be exploited. In this context, in order to ensure time efficiency and to avoid communication resources waste, the logical and the physical topologies should be closely related. Nevertheless, due to frequent changes in the topology, it will be necessary to rely on basic and fast tools such as `ping` or `traceroute`. We may also take into account the presence in the unstable platform of stable nodes, which are continuously connected, since more sophisticated and costly algorithm may be used to determine precisely the underlying topology of induced sub-network.

Task 1.6: Tractability analysis for the model of unstable platforms. Recently, several algorithms have been proposed for collective communications on unstable platforms, such as SplitStream [10]. These algorithms consist in using several broadcast trees both in order to balance the broadcasting load and to introduce some redundancy in order to ensure fault tolerance. We propose to adapt the techniques developed in Task 1.4, based on distributed computations of multi-commodity flows [4] and network coding [2], in order to ensure both fault tolerance and efficiency.

2.2 Peer to peer overlays: structures and functionalities

ALPAGE targets two main classes of applications: *(i)* data dissemination (or multicast) applications and *(ii)* grid computing applications. We are considering large-size systems. We identified the peer to peer communication paradigm as a suitable basis of our approach. Such a model is scalable and provides an attractive support for the large-scale dimension of applications targeted in the context of dynamic systems.

2.2.1 Peer to peer overlays

For the past ten years, distributed systems have been continuously growing in size. They now involve thousands, even millions of entities, scattered all over the Internet. To deal with this scalability shift, many of traditional approaches designed for small to medium size systems are no longer appropriate. The peer to peer communication paradigm has recently emerged as the key to fill this gap and provide a scalable support to the development of large-scale applications managing large amount of data. The inherent scalability of peer to peer systems rely on the following properties:

(i) Self-organization: Peer to peer systems are able to automatically re-organize themselves as peers join and leave the system.

(ii) Symmetry: As opposed to the traditional client-server approach, peers have a symmetric role in the system: they may both act as server and client. As the number of clients increases, the number of servers increases linearly so that no bottleneck is created in the system.

(iii) Fully decentralized control: Peer to peer systems do not rely on any central entity to control and manage the system. Instead, each node knows about a limited subset of the system.

Peer to peer overlays build a virtual network on top of a physical one according to a given structure (or absence of structure). In unstructured peer to peer overlays, each peer is connected to a set of other peers randomly chosen [17]. On the other side of the spectrum, structured peer to peer overlays organize peers according to a virtual name-space [28, 23]. Such overlay provide distributed hash table functionalities. Other hybrid and hierarchical overlays exist combining both approaches. Overall, such an architecture provides a scalable and efficient platform to develop large-scale applications and has generated a lot of interest in both academic and industrial worlds recently. Many applications might benefit from such architectures from file sharing applications to distributed data bases, grid computing applications and sensor networks. Most of these applications manage a large amount of data and/or resources and require efficient approaches to store, search, replicate, update and disseminate them.

This part of the project will focus on the design of solutions relying of the peer to peer paradigm in the context of the three system topologies considered in ALPAGE: stable, semi-stable and highly-dynamic. So far, most of the works done in the area of peer to peer overlays have been taking place mostly in the context of highly dynamic settings. In this project, we will also consider less dynamic systems. More specifically, we aim at studying how to exploit those topologies to optimize existing approaches.

2.2.2 Resource discovery in large-scale systems

One of the main objective of this part of ALPAGE is to provide efficient mechanism to discover resources (computing, storage and data) in large-scale systems. This functionality is crucial in the context of grid

computing applications in particular. So far, such applications have mostly relied on parallel computing programming model. They should today evolve towards distributed solutions to be able to exploit the full potential of large-scale grid computing infrastructures.

Task 2.1: Resource discovery We identified the following basic functionalities for resource location and discovery in grid computing applications: exact search, keyword-based search and range queries. In this task, we will study the combination of various peer to peer infrastructures and/or technologies. For example structured peer to peer overlays provide an efficient support for exact searches [24, 12] while unstructured networks are more adapted to keyword-based searches.

In addition, we plan to exploit the proximity potentially existing between peers (in the context of a given application) in order to improve performance. This proximity metric should be defined on a per application basis and might impact the logical links. For example we might consider topological proximity, semantic or coupling proximity metrics.

- **Topological proximity**. We measure the topological proximity as the topological distance between two peers. This can be measured for example by the number of physical links traversed to go from one peer to the other or the latency (or bandwidth) of such a route. This type of proximity has been one of the first one used to optimize peer to peer overlays [23, 21]. In fact, ignoring the topological information at the overlay level may lead to prohibitive latencies. For example, the good performances of Scribe [11], a tree-based peer to peer application-level system, relies on the fact that the underlying peer to peer overlay is optimized according to this metric. Exploiting this proximity has been identified as crucial in Task 1.5.

- **Semantic proximity**. This metric reflects an interest-based proximity between two peers according to a given application. This property has been exploited in a number of recent approaches [27, 15, 30], more specifically in the context of peer to peer file sharing systems. The basic idea is that if a system is able to cluster logically peers sharing the same interests (for example interested in the same files), the performance of search might be increased significantly. Capturing such a proximity is a tedious task that depends on the considered applications. We will consider the application of the recent approaches proposed in the context of file sharing systems to grid computing applications.

- **Coupling proximity**. Coupling proximity can be defined as a privileged relationship between peers involved in the same program which relates them either through data or code. For example, if a peer B uses the data produced by a program running on peer A, they are considered as coupled. Capturing such a proximity between peers could lead to great improvement on performance and have an impact on the tasks scheduling.

In this project, we consider simple queries (exact or keyword-based) and more complex queries. Such queries are used to discover resources and are expressed as logical expressions. The approach we plan to follow in ALPAGE is to capture and leverage the various forms of proximity exhibited by an application, taking into account the dynamics of the topology. For example topological proximity might be discovered and exploited on a long term in a static platform.

Instantaneous searches assume that peers are reactive: if the search fails, the requester may need to reiterate his request. Persistent queries enable a user to specify a request that will remain valid over a given period of time so that he will be notified later if the request could not be satisfied. Obviously, such requests are particularly useful in the context of resource discovery where peers might be waiting for a given type of resource or data to be available. We plan to extend the work done on instantaneous searches in this context. This implies to take care of the following issues: publication of newly available resources and notification to interested peers, and taking into account timing information related to validity.

Task 2.2: Small world peer to peer overlays Jon Kleinberg [19] modeled relationships in social networks using a two-dimensional grid where peers are placed according to their geographical position. Each peer

would know its direct neighbors in the grid as well as a few random long links. In [19] Kleinberg shows that it is possible to chose these links such that a greedy routing algorithm provide as good routing performance as the ones provided by a specific topology between peers. This result is actually a good starting point to implement efficient search mechanisms.

To summarize, our project will explore various peer to peer structures in order to provide efficient search functionalities on a grid computing infrastructure. More specifically we will focus on the following general objectives:

- Figure out the functionalities that one can expect from a peer to peer network in the context of grid computing. For example, connections through a firewall are not considered in file sharing applications.
- Design peer to peer overlay specifically for grid computing applications that will be more efficient than currently deployed peer to peer systems. Exploiting the topology information when it is known (geographical position, potential hierarchy) and use this information to improve performance.

2.3 Scheduling parallel applications

Scheduling applications on large scale platforms induces several problems. If a single application is executed on the platform, we typically aim at balancing the load while minimizing the communication volume. The problem is known to be hard for homogeneous parallel machines, and only gets worse for heterogeneous clusters, or even collections of such clusters. If several applications are run concurrently, and thus compete for CPU and network resources, a fair sharing of the platform must be enforced in order to optimize the execution time of each application.

The algorithms that can be designed to solve the previous problems heavily depend upon the context. Mapping and scheduling techniques must take into account the available information on application and platform parameters, or the lack thereof. Is there a master processor acting as a centralized scheduler? does the master posses an accurate knowledge of the whole platform? is the platform stable, semi-stable or unstable? The following tasks aim at providing different algorithmic techniques, depending upon the answer to the previous questions.

Note that a particular instance of the problem involves applications that require the manipulation of large files, which are initially distributed across the platform. This problem is dealt with in Section 2.4.

2.3.1 Centralized scheduling

Task 3.1: Centralized scheduling of a single application. In this task, we aim at executing a large set of independent, same-size tasks. First we assume that there is a single master, that initially holds all the (data needed for all) tasks. The problem is to determine an architecture for the execution. Which processors should the master enroll in the computation? How many tasks should be sent to each participating processor? In turn, each processor involved in the execution must decide which fraction of the tasks must be computed locally, and which fraction should be sent to which neighbor (these neighbors must be determined too).

Parallelizing the computation by spreading the execution across many processors may well be limited by the induced communication volume. Rather than aiming at makespan minimization, a more relevant objective is the optimization of the throughput in steady-state mode. There are three main reasons for focusing on the steady-state operation. First is *simplicity*, as the steady-state scheduling is really a relaxation of the makespan minimization problem in which the initialization and clean-up phases are ignored. One only needs to determine, for each participating resource, which fraction of time is spent computing for which application, and which fraction of time is spent communicating with which neighbor; the actual schedule then arises naturally from these quantities. Second is *efficiency*, as steady-state scheduling provides, by

definition, a periodic schedule, which is described in compact form and is thus possible to implement efficiently in practice. Third is *adaptability*: because the schedule is periodic, it is possible to dynamically record the observed performance during the current period, and to inject this information into the algorithm that will compute the optimal schedule for the next period. This makes it possible to react on the fly to resource availability variations.

We have already designed some scheduling algorithms to optimize the steady-state throughput, but these only apply to stable platforms so far. We have ideas on how to extend the solution to deal with semi-stable platforms, e.g. adjusting the period length to cope with variations in resource performance, and re-injecting the knowledge dynamically acquired by monitoring the current execution.

Unstable platforms would require a totally different approach, where task replication would have to play a major role. How to choose the replication factor, and how to efficiently keep track of successfully executed copies? Another important criterion to consider are the *average* response time (or delay in the system), and *maximal* response time. In fact, designing multi-criteria algorithms capable of achieving a wide range of throughput/response time trade-offs would be very valuable.

To design efficient solutions for semi-stable and unstable platforms, we will use the models built up in the previous tasks. The proposed solutions will rely on a statistical vision of the platforms, and of their expected evolution throughout time.

Task 3.2: Centralized scheduling of several applications. In this task, we target the concurrent execution of several master-slave applications. As above, each application consists of a large collection of independent, same-size tasks. The applications can be of very different nature, e.g. files to be processed, images to be analyzed or matrices to be multiplied. Note that the size of the tasks, both in terms of communication volume or of computing demand, may well vary from one application to another. In fact, the relative *communication-to-computation ratio* of the applications is likely to prove an important parameter in the scheduling process.

Each application is initiated on and by a master processor, which is not necessarily the same for each application. All applications compete for CPU and network resources. The scheduler must ensure a fair management of these resources. For each application, define its throughput as the (fractional) number of tasks executed every time-unit, in steady-state mode. The overall objective is to achieve the same weighted throughput for each application. Indeed, if all applications were equally important, the scheduler would ideally process the same number of tasks for each application every time-step, thereby realizing the same throughput. However, some applications may be given more priority. For each application we define a *priority factor* that quantifies its relative worth. For instance, computing two units of load per time unit for an application with priority factor 2 is as worthwhile/profitable than computing one unit of load for an application with priority factor 1. This concept makes it possible to implement notions of application priorities for resource sharing.

The scheduling problem is more complex than the one defined in Task 3.1, because both the *communication-to-computation ratio* and the *priority factor* differ among the applications. We have investigated preliminary solutions for stable platforms, but everything is to be invented for the other two types of architectures.

2.3.2 Distributed scheduling

Task 3.3: Distributed scheduling of a single application. Even for stable platforms, it is not fully clear that we can realistically assume a global, reliable and accurate knowledge of all the platform parameters. The larger the platform, the less realistic the above assumption. In fact, for the largest platforms, it is likely that only a hierarchical knowledge of the information is available: each node only possesses the partial information related to its level in the hierarchy (see the tasks of Section 2.1 on how to acquire this local information).

We have to tackle the design and evaluation of distributed scheduling mechanisms. Each participating resource will take scheduling decisions based upon a precise information on its immediate neighborhood. Ideally, the optimal scheduling will be reached after a series of iterations aimed at local refinements of the

current solution. A first concept to investigate is that of decentralized multi-commodity flow algorithms (variants of the Awerbuch and Leighton algorithm).

It may well be the case that a decentralized approach turns out to be the only realistic solution in several applicative contexts. However, we insist that the deployment of decentralized scheduling algorithms must be very conservative. It is mandatory to guarantee that all computations are correctly executed before aiming at optimizing them by injecting local knowledge into the scheduler.

Task 3.4: Long-term extensions. Of course, we can target a distributed scheduling mechanism for the concurrent execution of several master-slave applications, but it seems a very difficult goal. Instead, we may try to deal with more complex applications, replacing the independent tasks by a series of identical Directed Acyclic Graphs (e.g., solving a succession of finite-element problems, each of them being a task graph).

Also, there are scenarios where all application tasks are not known *a priori*, but instead are submitted by the users as the execution progresses. We have to provide *on line* schedulers capable of efficiently managing such applications, most likely by extending traditional window-based strategies.

2.4 Scheduling applications sharing (large) files

We deal here with a particular instance of the previous scheduling problem. This instance involves applications that require the manipulation of large files, which are initially distributed across the platform. It may well be the case that some files are replicated. Also, some application tasks may depend upon the same files (e.g., if the files correspond to the same input data to which will be processed through different computational kernels).

The target platform is composed of several servers, with different computing capabilities, and which are linked through an interconnection network. To each server is associated a (local) data repository. Initially, the files are stored in one or several of these repositories. We assume that a file may be duplicated, and thus simultaneously stored on several data repositories. There may be restrictions on the possibility of duplicating the files (typically, each repository is not large enough to hold a copy of all the files. After having decided that server S_i will execute task T_j, the input files for T_j that are not already available in the local repository of S_i will be sent through the network. Several file transfers may occur in parallel along disjoint routes. Also, it may be possible for intermediate hosts to keep a copy of the files that they are forwarding, provided they have enough storage space to do so; this would result in increasing the number of source repositories for the corresponding files, thereby potentially speeding up the next request to access them.

Task 4.1: Centralized approach. In the first version of the problem, we assume that there is a centralized scheduler, which knows all platform and application parameters. The objective is to map each task to a server, and to schedule all the communications induced by the mapping, so as to minimize the total execution time.

There are several variants to consider. In a purely static context, the objective is to design bi-criteria algorithms that aim both at minimizing the total volume of files that are being transferred, and at carefully balancing the load of the servers. A first approach is to extend classical list heuristics, such as *min-min* or *sufferage*, to take into account that files are shared between tasks. An other idea is to adapt Web cache techniques, taking into account the fact that file accesses can be predicted more accurately than Web location accesses.

An interesting extension is that of the steady-state execution, when several instances of the same problem are pipelined on the platform. We are then faced to the same questions as in Task 3.1. The new difficulty is that we no longer deal with independent tasks but instead we have to manage tasks sharing files.

Task 4.2: Distributed approach. The framework is the same as for Task 3.3, but the fact that tasks depend upon remote files (possibly shared) is a major complication. Indeed, local dynamic strategies, aimed at self-

stabilizing the load, will be harder to design: it will likely be difficult to come up with a reliable estimation of the cost related to communicating the files, because the information on the closest source will be neither accurate nor up-to-date.

3 Intended results

In Sections 2 to 4, we already described expected results for the different tasks. Therefore, we concentrate in this section on task allocation and schedule. We also detail the work that will be conducted by the engineers hired by the project (named CDD 1 to 3, where CDD means *Contrat à durée déterminée* and can be translated by *Fixed-term position*).

3.1 Task allocation

In the following, the three different sites are called West (Bordeaux and Rennes), East (Lyon, Grenoble and Nancy), and Paris (LRI and LIX). We refer to tasks as:

- Tasks 1.*: platform discovery and collective communication schemes
- Tasks 2.*: peer to peer systems and resource discovery
- Tasks 3.*: scheduling

Task are assigned to sites as summarized in the table below:

	EAST	PARIS	WEST
Tasks 1.*	X		X
Tasks 2.*		X	X
Tasks 3.*	X	X	

3.2 Schedule

The expected schedule for the different tasks is the following:

	0-6 months	6-12 months	12-18 months	18-24 months	24-30 months	30-36 months
Task 1.1	X	X (CDD1)	X (CDD1)			
Task 1.2	X	X	X			
Task 1.3	X	X (CDD1)	X (CDD1)			
Task 1.4		X	X (CDD2)	X (CDD2)		
Task 1.5	X	X (CDD1)	X (CDD1)			
Task 1.6				X (CDD2)	X	X
Task 2.1	X	X	X (CDD2)	X (CDD2)		
Task 2.2			X (CDD2)	X (CDD2)	X	X
Task 3.1		X (CDD1)	X (CDD1)			
Task 3.2	X	X				
Task 3.3			X	X		
Task 3.4					X	X
Task 4.1	X	X	X	X (CDD3)	X (CDD3)	
Task 4.2				X (CDD3)	X(CDD3)	X

3.3 Expected results for CDDs

3.3.1 CDD1: Platform Discovery, Stable Platforms

Most algorithms to deploy efficiently a large set of independent tasks rely on an accurate knowledge of the platform. However, as explained in section 2.1.1, most methods proposed in the network community are not suited to discover an application-level network topology. Even though several toolboxes (ENV [26, 25] or ALNEM [20]) have been developed to find the topology of stable platforms, algorithms for building such maps are still incomplete and too slow to be considered for large scale platforms.

The main task of this CDD will be to first study the scalability of these toolboxes by evaluating their performances on both real and simulated platforms. It should enable to improve these tools (by improving the underlying algorithms and coupling them with NWS [31]) and then to use them to deploy large scale applications. The hired person will start his/her work for 12 months in March 2006. He/She will spend 6 months at ID (Grenoble) and 6 months at LIP (Lyon).

3.3.2 CDD2: Collective Communications

The main task of this CDD will be to develop algorithms for collective communications. We schedule the beginning of this CDD for September 2006, so that the question of collective communication on stable platform should be solved. Therefore, his/her work will concentrate on the design of broadcast and multicast algorithms on semi-stable and unstable platforms. For semi-stable platforms, the objective is to design distributed robust algorithms, based on multi commodity flow algorithms [4] and network coding [2]. We propose to develop a prototype for broadcast and multicast onto semi stable platforms, and to validate it either through simulations or direct experimentation. The work on unstable platform will consist in improving the current version of SplitStream [10], in order to take the actual performances of the underlying network into account. The hired person will start his/her work for 12 months in September 2006. He/She will spend 6 months at LaBRI (Bordeaux) and 6 months at IRISA (Rennes).

3.3.3 CDD3: Scheduling applications sharing files

The main task of this CDD will be to develop algorithms for scheduling applications sharing files. We schedule the beginning of this CDD in September 2007, i.e. at a later stage in the project. This is because Tasks 4.1. and 4.2. build upon several results of former tasks. Task 4.1. requires input from the work on modeling and discovering the platforms, while Task 4.2. will benefit from previous work on decentralized scheduling for simpler applications. We propose to develop several heuristics and prototype software for the centralized version of the problem, both for makespan minimization and throughput optimization in steady-state mode. For the distributed approach, the objective is to design a robust distributed algorithm that aims at minimizing the communication volume, but it is not clear yet which tools will turn out useful. The hired person will start his/her work for 12 months in September 2007. He/She will spend 6 months at LIX (Paris) and 6 months at LIP (Lyon).

References

[1] I. Abraham, D. Malkhi, and O. Dobzinski. Land: Stretch (1+e) locality aware networks for dhts. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 2004)*, 2004.

[2] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. Information Theory*, 46(4), july 2000.

[3] B. Awerbuch and T. Leighton. A simple local-control approximation algorithm for multicommodity flow. In *FOCS '93: Proceedings of the 24th Conference on Foundations of Computer Science*, pages 459–468. IEEE Computer Society Press, 1993.

[4] B. Awerbuch and T. Leighton. Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks. In *STOC '94: Proceedings of the 26h ACM symposium on Theory of Computing*, pages 487–496. ACM Press, 1994.

[5] M. Banikazemi, V. Moorthy, and D. K. Panda. Efficient collective communication on heterogeneous networks of workstations. In *Proceedings of the 27th International Conference on Parallel Processing (ICPP'98)*. IEEE Computer Society Press, 1998.

[6] M. Banikazemi, J. Sampathkumar, S. Prabhu, D. Panda, and P. Sadayappan. Communication modeling of heterogeneous networks of workstations for performance characterization of collective operations. In *HCW'99, the 8th Heterogeneous Computing Workshop*, pages 125–133. IEEE Computer Society Press, 1999.

[7] O. Beaumont, L. Carter, J. Ferrante, A. Legrand, and Y. Robert. Bandwidth-centric allocation of independent tasks on heterogeneous platforms. In *International Parallel and Distributed Processing Symposium (IPDPS'2002)*. IEEE Computer Society Press, 2002.

[8] O. Beaumont, A. Legrand, L. Marchal, and Y. Robert. Pipelining broadcasts on heterogeneous platforms. *IEEE Trans. Parallel Distributed Systems*, 16(4), april 2005.

[9] F. Cappello, P. Fraigniaud, B. Mans, and A. L. Rosenberg. HiHCoHP: Toward a realistic communication model for hierarchical HyperClusters of heterogeneous processors. In *International Parallel and Distributed Processing Symposium IPDPS'2001*. IEEE Computer Society Press, 2001.

[10] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *19th ACM Symposium on Operating Systems Principles SOSP'2003*. ACM Press, 2003.

[11] M. castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE journal on Selected Area in Communications (JSAC)*, 20(8), 2002.

[12] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, Oct. 2001.

[13] edonkey. http://www.edonkey2000.com/index.html.

[14] Gnutella. http://www.gnutella.com.

[15] S. Handurukande, A.-M. Kermarrec, F. L. Fessant, and L. Massoulié. Exploiting semantic clustering in the edonkey p2p network. In *SIGOPS European workshop*, 2004.

[16] B. Hong and V. Prasanna. Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput. In *International Parallel and Distributed Processing Symposium IPDPS'2004*. IEEE Computer Society Press, 2004.

[17] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In *ACM/IFIP/USENIX 5th International Middleware Conference (Middleware)*, Toronto, Canada, October 2004.

[18] KazAa. www.kazaa.com.

[19] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *32nd ACM Symposium on Theory of Computing*, 2000.

[20] A. Legrand, F. Mazoit, and M. Quinson. An application-level network mapper. Technical Report 2002-09, Ecole Normale Superieure de Lyon, Feb. 2002.

[21] L. Massoulié, A.-M. Kermarrec, and A. Ganesh. Network awareness and failure resilience in self-organizing overlay networks. In *22nd Symposium on Reliable Distributed Systems*, 2003.

[22] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, Aug. 2001.

[23] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.

[24] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP'01)*, Chateau Lake Louise, Banff, Canada, Oct. 2001.

[25] G. Shao. *Adaptive scheduling of master/worker applications on distributed computational resources*. PhD thesis, Dept. of Computer Science, University Of California at San Diego, 2001.

[26] G. Shao, F. Berman, and R. Wolski. Master/slave computing on the grid. In *Heterogeneous Computing Workshop HCW'00*. IEEE Computer Society Press, 2000.

[27] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM'03*, 2003.

[28] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM 2001*, San Diego, USA, Aug. 2001.

[29] I. Stokes-Rees, A. Tsaregorodtsev, and V. Garonne. DIRAC: A scalable lightweight architecture for high throughput computing. In *Fifth IEEE/ACM International Workshop on Grid Computing Grid 2004*. IEEE Computer Society Press, 2004.

[30] S. Voulagaris and M. van Steen. Epidemic-style management of semantic overlays for content-based searching. In *EuroPar'05*, 2005.

[31] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768, 1999.

4 Bibliographical references of the researchers involved in the project

4.1 EAST site

Arnaud Legrand
Arnaud Legrand received the PhD degree from École normale supérieure de Lyon in 2003. He is currently a CNRS permanent researcher in the ID-IMAG laboratory in Grenoble. He is mainly interested in parallel algorithm design for heterogeneous platforms and in scheduling techniques.

- C. Banino, O. Beaumont, L. Carter, J. Ferrante, A. Legrand, and Y. Robert. Scheduling strategies for master-slave tasking on heterogeneous processor platforms. *IEEE Trans. Parallel Distributed Systems*, 15(4):319–330, 2004.
- A. Legrand and M. Quinson. Automatic deployment of the Network Weather Service using the Effective Network View. *High Performance Grid Computing workshop (HPGC'04)*.
- O. Beaumont, A. Legrand, L. Marchal, and Y. Robert. Scheduling strategies for mixed data and task parallelism on heterogeneous clusters. *Parallel Processing Letters*, 2003.
- H. Casanova, A. Legrand, L. Marchal. Scheduling Distributed Applications: the SimGrid Simulation Framework. *Proceedings of the third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)* .

- O. Beaumont, A. Legrand, F. Rastello, and Y. Robert. Dense linear algebra kernels on heterogeneous platforms: redistribution issues. *Parallel Computing*, 28:155–185, 2002.

Martin Quinson
Martin Quinson received in 2003 the Ph.D. degree in Computer Science from the ENS-Lyon, France. In 2004, he worked at University of California, Santa Barbara in the team of Professor Wolski on the NWS project on grid platform monitoring. He is currently an Assistant Professor at the University Henri Poincaré, Nancy-I since 2005. Martin Quinson's main research interests include network monitoring and tomography, distributed application developement and grid middlewares.

- Martin Quinson. Dynamic Performance Forecasting for Network-Enabled Servers in a Metacomputing Environment. *International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS'02), in conjunction with IPDPS'02*.
- Arnaud Legrand, Frédéric Mazoit and Martin Quinson. An Application-Level Network Mapper. *Research Repport RR-LIP2003-09, École Normale Supérieure de Lyon, 2002*
- Philippe Combes, Frédéric Lombard, Martin Quinson and Frédéric Suter. A Scalable Approach to Network Enabled Servers. *Proceedings of the 7th Asian Computing Science Conference, 2002*.
- Arnaud Legrand and Martin Quinson. Automatic deployment of the Network Weather Service using the Effective Network View. *High Performance Grid Computing workshop (HPGC'04)*.
- Eddy Caron, Frédéric Desprez, Martin Quinson and Frédéric Suter. Performance Evaluation of Linear Algebra Routines for Network Enabled Servers. *The International Journal on High Performance Computing and Applications, 2004*.

Yves Robert
Yves Robert received the PhD degree from Institut National Polytechnique de Grenoble in 1986. He is currently a full professor in the Computer Science Laboratory LIP at ENS Lyon. He is the author of four books, 90 papers published in international journals, and 110 papers published in international conferences. His main research interests are scheduling techniques and parallel algorithms for clusters and grids. He is a senior member of IEEE and the IEEE Computer Society, and serves as an associate editor of *IEEE Transactions on Parallel and Distributed Systems*.

- O. Beaumont, V. Boudet, A. Petitet, F. Rastello, and Y. Robert. A proposal for a heterogeneous cluster ScaLAPACK (dense linear solvers). *IEEE Trans. Computers*, 50(10):1052–1070, 2001.
- O. Beaumont, V. Boudet, F. Rastello, and Y. Robert. Matrix multiplication on heterogeneous platforms. *IEEE Trans. Parallel Distributed Systems*, 12(10):1033–1051, 2001.
- O. Beaumont, A. Legrand, L. Marchal, and Y. Robert. Steady-state scheduling on heterogeneous clusters. *Int. J. of Foundations of Computer Science*, 16(2), 2005.
- A. Legrand, H. Renard, Y. Robert, and F. Vivien. Mapping and load-balancing iterative computations on heterogeneous clusters with shared links. *IEEE Trans. Parallel Distributed Systems*, 15(6):546–558, 2004.
- O. Beaumont, L. Marchal, and Y. Robert. Complexity results for collective communications on heterogeneous platforms. *Int. Journal of High Performance Computing Applications*, 2006, to appear.

Frédéric Vivien
Frédéric Vivien was born in 1971 in Saint-Brieuc, France. He obtained a PhD thesis from École normale supérieure de Lyon in 1997. From 1998 to 2002, he was an associate professor at Louis Pasteur University of Strasbourg. He spent the year 2000 working in the Computer Architecture Group of the MIT Laboratory for Computer Science. He is currently a full researcher from INRIA. His main research interests are scheduling techniques, parallel algorithms for clusters and grids, and automatic compilation/parallelization techniques.

- P. Boulet, J. Dongarra, F. Rastello, Y. Robert, and F. Vivien. Algorithmic issues on heterogeneous computing platforms. *Parallel Processing Letters*, 9(2):197–213, 1999.
- A. Giersch, Y. Robert, and F. Vivien. Scheduling tasks sharing files from distributed repositories. In *Euro-Par-2004: International Conference on Parallel Processing*, LNCS 3149, pages 148–159. Springer Verlag, 2004.
- A. Giersch, Y. Robert, and F. Vivien. Scheduling tasks sharing files on heterogeneous master-slave platforms. *Journal of Systems Architecture*, 2005, to appear.
- A. Legrand, H. Renard, Y. Robert, and F. Vivien. Mapping and load-balancing iterative computations on heterogeneous clusters with shared links. *IEEE Trans. Parallel Distributed Systems*, 15(6):546–558, 2004.
- H. Renard, Y. Robert, and F. Vivien. Data redistribution algorithms for heterogeneous processor rings. *Int. Journal of High Performance Computing Applications*, 2006, to appear.

4.2 PARIS site

Philippe Baptiste
Philippe Baptiste is 33 years old. He holds a research position at CNRS, and is associate professor at Ecole Polytechnique. His main research interests are: operations research, constraint programming and scheduling theory. He has published more than 25 papers in international journals. He is also working with Ilog, Thales, and Eurocontrol, on the resolution of real life optimization problems. He has received the Goldstine Fellowship (IBM research), the "Cor Baayen Award" (ERCIM, European Research Consortium for Informatics and Mathematics) and the "Prix Robert Faure" (SociétFr. de Rech. Op.). Philippe is a member of the editorial board of "Journal of Scheduling", "Operations Research Letters" and "Discrete Optimization".

- Ph. Baptiste, C. Le Pape, and W. Nuijten. Constraint-Based Scheduling. Kluwer Academic Publishers, 2001.
- Ph. Baptiste, Peter Brucker, Sigrid Knust, and Vadim G. Timkovsky. Ten notes on equal-processing-time scheduling. 4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies, Volume 2, pages 111-127, 2004.
- A. Jouglet, Ph. Baptiste and J. Carlier. Branch-and-Bound Algorithms for Total Weighted Tardiness. In HandBook of Scheduling: Algorithms, Models and Performance Analysis, coordinated by J. Leung, CRC Press, USA, 2004
- Ph. Baptiste and C. Le Pape. Scheduling a Single Machine to Minimize a Regular Objective Function under Setup Constraints. Discrete Optimization 2, pages 83-99, 2005.
- Ph. Baptiste and S. Demassey. Tight LP Bounds for Resource Constrained Project Scheduling. OR Spectrum, 26: 251-262, 2004.

Christoph Dürr
Christoph Dürr did his PhD at the LRI, Université de Paris-Sud on classical, deterministic algorithms for problems that raise from quantum cellular automata. Then he worked at ICSI, Berkeley, on discrete tomography: the problem of reconstruction discrete two-dimensional objects from orthogonal one-dimensional projections. After some research activities in quantum computing, he is now into scheduling, where he studies the problems of maximizing the throughput in overloaded systems or maximizing the maxflow in router queues.

- Konstantin Artiouchine, Philippe Baptiste, and Christoph Dürr, "Runway scheduling with holding loop", In proceedings of Discrete Optimization Methods in Production and Logistics (DOM), pp. 96-101, Omsk-Irkutsk, Russia, 2004.
- Philippe Baptiste, Marek Chrobak, Christoph Dürr, Wojciech Jawor and Nodari Vakhania, "Preemptive Scheduling of Equal-Length Jobs to Maximize Weighted Throughput", Operation Research Letters, vol. 32(3), pp. 258-264, 2004.
- Philippe Baptiste, Christoph Dürr, Marek Chrobak and Francis Sourd. "Preemptive Multi-Machine Scheduling of Equal-Length Jobs to Minimize the Average Flow Time", Presentation at Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP), 2005.
- Marek Chrobak, Christoph Dürr, Wojciech Jawor, Lukasz Kowalik, Maciej Kurowski, "A Note on Scheduling Equal-Length Jobs to Maximize Throughput", to appear in Journal of Scheduling.

Pierre Fraigniaud
Pierre Fraigniaud received the Ph.D. degree in Computer Science from the ENS-Lyon, France, in 1990. He is currently Directeur de Recherches at CNRS, and located at the CS Department of Univ. Paris-Sud (LRI), where he is leading the research group "Graph Theory and Fundamental Aspects of Communications" (GraFComm). He is also associated member of the INRIA project "Grand Large" on global parallel and distributed computing. Pierre Fraigniaud's main research interests include several aspects of communication networks, parallel and distributed systems, and telecommunication systems. He is member of the Editorial Board of Theory of Computing Systems (TOCS), and the Journal of Interconnection Networks (JOIN). He is currently acting as Program Chair for the 19th Int. Symp. on Distributed Computing (DISC 2005), and acted as Program Chair for the 13th ACM Symp. on Parallel Algorithms and Architectures (SPAA 2001). He is member of the Steering Committee of the series of Int. Symp. on Theoretical Aspects of Computer Science (STACS).

- R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg, "Label-Guided Graph Exploration by a Finite Automaton", In 32nd Int. Colloquium on Automata, Languages and Programming (ICALP), 2005.
- F. Cappello, P. Fraigniaud, B. Mans and A. L. Rosenberg, "An Algorithmic Model for Heterogeneous Hyper-Clusters: Rationale and Experience", International Journal of Foundations of Computer Science 16, pages 195-216, 2005.
- P. Fraigniaud, "A New Perspective on the Small-World Phenomenon: Greedy Routing in Tree-Decomposed Graphs", In 13th Annual European Symposium on Algorithms (ESA), 2005.
- P. Fraigniaud, and P. Gauron, "D2B: a de Bruijn Based Content-Addressable Network", To appear in Theoretical Computer Science, Special Issue on Complex Networks.
- P. Fraigniaud, B. Mans and A. L. Rosenberg, "Efficient Trigger-Broadcast in Heterogeneous Clusters", Journal of Parallel and Distributed Computing 65, pages 628-642, 2005.

Sébastien Tixeuil
Sébastien Tixeuil received the "Magistère d'Informatique Appliquée" from University Pierre and Marie Curie (France) in 1995, and his M.Sc and Ph.D. in Computer Science from the University of Paris Sud (France) in 1995 and 2000, respectively. In 2000, he became faculty at the University Paris Sud, where he is now an Assistant Professor (Maître de Conférences). He is member of the INRIA project "Grand Large" since its creation in 2003. His research interests include self-stabilization, and fault-tolerance in distributed systems.

- Pierre Fraigniaud, David Ilcinkas, Sergio Rajsbaum, and Sébastien Tixeuil, "Space lower bounds for graph exploration via reduced automata", In proceedings of 12th International Colloquium on Structural Information and Communication Complexity (SIROCCO), LNCS 3499, pages 140-154, 2005.
- Ajoy K. Datta, Maria Gradinariu, and Sébastien Tixeuil, "Self-stabilizing Mutual Exclusion with Arbitrary Scheduler", The Computer Journal 47(3), pages 289-298, 2004.
- Colette Johnen, Franck Petit and Sébastien Tixeuil, "Auto-stabilisation et Protocoles Réseaux", Technique et Science Informatiques 23(8), pages 1027-1056, 2004.
- Philippe Duchon, Nicolas Hanusse, and Sébastien Tixeuil, "Optimal Randomized Self-stabilizing Mutual Exclusion in Synchronous Rings", In proceedings of the 18th Symposium on Distributed Computing (DISC), LNCS 3274, pages 216-229, 2004.
- Bertrand Ducourthial and Sébastien Tixeuil, "Self-stabilization with Path Algebra", Theoretical Computer Science 293(1), pages 219-236, 2003.

4.3 WEST site

Olivier Beaumont
Olivier Beaumont obtained a PhD thesis from the Université de Rennes in 1999, and the *Habilitation à diriger des Recherches* from the Université de Bordeaux in 2004. He is currently an associate professor in the LaBRI laboratory in Bordeaux. His main research interests are parallel algorithms on distributed memory architectures.

- Cyril Banino, Olivier Beaumont, Larry Carter, Jeanne Ferrante, Arnaud Legrand, and Yves Robert. Scheduling strategies for master-slave tasking on heterogeneous processor platforms. *IEEE Trans. Parallel Distributed Systems*, 15(4):319–330, 2004.
- Olivier Beaumont, Arnaud Legrand, Loris Marchal, and Yves Robert. Pipelining broadcasts on heterogeneous platforms. *IEEE Trans. Parallel Distributed Systems*, 16(4):300-313, 2005.
- Olivier Beaumont, Henri Casanova, Arnaud Legrand, Yves Robert, and Yang Yang. Scheduling divisible loads on star and tree networks: results and open problems. *IEEE Trans. Parallel Distributed Systems*, 16(3):207-218, 2005.
- Olivier Beaumont, Arnaud Legrand, and Yves Robert. The master-slave paradigm with heterogeneous processors. *IEEE Trans. Parallel Distributed Systems*, 14(9):897–908, 2003.

Philippe Duchon
Philippe Duchon was born in 1969. He received his Ph.D. in Computer Science from the Université Bordeaux 1 in 1998, and currently holds a position as associate professor in the LaBRI laboratory in Bordeaux. His research interests range from enumerative combinatorics to the study of randomized algorithms and stochastic processes.

- P. Duchon, P. Flajolet, G. Louchard, G. Schaeffer, 2004. "Boltzmann Samplers for the Random Generation of Combinatorial Structures". Combinatorics, Probability and Computing 13 No 4-5, pp 577-625.

- P. Duchon, N. Hanusse, E. Lebhar, N. Schabanel, 2004. "Could Any Graph be turned into a Small World?", à paraître dans Theoretical Computer Science (Rapport de recherche LIP 2004-62).
- P. Duchon, N. Hanusse, N. Saheb, A. Zemmari, 2004. "Broadcast in the Rendezvous Model", à paraître dans Information Processing Letters (version courte dans les actes de STACS 2004, Lecture Notes in Computer Science 2996, pp 559-570).
- P. Duchon, N. Hanusse, S. Tixeuil, 2004. "Optimal Randomized Self-stabilizing Mutual Exclusion on Synchronous Rings", actes de DISC 2004 (Lecture Notes in Computer Science 3274, pp 216-229).

Cyril Gavoille
Cyril Gavoille received the PhD from École Normale Supérieure de Lyon in 1996. He is currently a full professor at Bordeaux University in the Computer Science Laboratory LaBRI. He is the author of 20 international journal papers and of 40 international conference papers. His main research interests are distributed computing and data-structures, specially routing algorithms. He was chairman of 6th Colloquium on Structural Information and Communication Complexity (SIROCCO) in 1999, and has participed in 17 international program committee conferences (PODC, PDCN, HiPC, IWDC for 2005).

- Pierre Fraigniaud, Cyril Gavoille, and Christophe Paul. Eclecticism shrinks even small worlds. In 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC), pages 169-178. ACM Press, July 2004.
- Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. In 16th Annual ACM Symposium on Parallel Algorithms and Architecture (SPAA), pages 20-24. ACM Press, July 2004.
- Yon Dourisboure and Cyril Gavoille. Sparse additive spanners for bounded tree-length graphs. In 11th International Colloquium on Structural Information & Communication Complexity (SIROCCO), volume 3104 of Lecture Notes in Computer Science, pages 123-137. Springer, June 2004.
- Pierre Fraigniaud and Cyril Gavoille. Header-size lower bounds for end-to-end communication in memoryless networks. Computer Networks, 2004.

Nicolas Hanusse
Nicolas Hanusse received the PhD degree from Université Bordeaux I in 1997. He is currently a CNRS permanent researcher in the LaBRI laboratory of Bordeaux. He is mainly interested in distributed computing and graph algorithms.

- Nicolas Hanusse, Dimitris Kavvadias, Evangelos Kranakis and Danny Krizanc, Memoryless Search Algorithm in Networks with Faulty Advice, IFIP-WCC'2002 (World Computer Congress); Track - International Conference on Theoretical Computer Science, pages 206-216, Montreal, 2002.
- Philippe Duchon, Nicolas Hanusse, Sébastien Tixeuil, Optimal Randomized Self-stabilizing Mutual Exclusion on Synchronous Rings, Proceedings of DISC'2004,pp216–229, 2004
- P. Duchon, N. Hanusse, N. Saheb, A. Zemmari, 2004. "Broadcast in the Rendezvous Model", à paraître dans Information and Computation (version courte dans les actes de STACS 2004, Lecture Notes in Computer Science 2996, pp 559-570).
- N. Hanusse, E. Kranakis, D. Krizanc, Searching with Mobile Agents in Networks with Liars, Discrete Applied Mathematics, Vol 137, Issue 1, pages 69-85, 2004
- P. Duchon, N. Hanusse, E. Lebhar, N. Schabanel, 2004. "Could Any Graph be turned into a Small World?", à paraître dans Theoretical Computer Science (Rapport de recherche LIP 2004-62).

Anne-Marie Kermarrec
Anne-Marie Kermarrec obtained her Ph.D. from the University of Rennes in October 1996 in the area of fault-tolerant distributed shared memory systems. She also spent a year (Oct.96-Oct97) in the Computer Systems group of Vrije Universiteit, Amsterdam, working in the Globe project in collaboration with Andrew Tanenbaum and Maarten van Steen. In 1997, she became an assistant professor at the University of Rennes. In 2000, she joined Microsoft Research in Cambridge and worked in the area of peer to peer computing. Since February 2004, she is a senior researcher at INRIA, working in the PARIS project. Her research interests include large-scale distributed systems, peer to peer overlay networks and applications. In this context, she has many collaborations with foreign labs (EPFL Lausanne, Vrije Universiteit Amsterdam, Microsoft Research Cambridge and University of Bologna.

- Patrick Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Massoulié. From Epidemics to Distributed Computing. IEEE Computer, 37(5):60-67, May 2004.
- Patrick Eugster, Pascal Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The Many Faces of Publish/Subscribe. ACM computing Surveys, 35(2), June 2003.
- Patrick Eugster, Sidath Handurukande, Rachid Guerraoui, Anne-Marie Kermarrec, and Petr Kouznetsov. Lightweight Probabilistic Broadcast. ACM Transaction on Computer Systems, 21(4), November 2003.
- Sidath Handurukande, Anne-Marie Kermarrec, Fabrice Le Fessant, and Laurent Massoulié. Exploiting Semantic Clustering in the eDonkey P2P network. In SIGOPS European Workshop, Leuven, Belgium, pages 109-114, September 2004.
- Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: High-Bandwidth Multicast in Cooperative Environments. In Symposium on Operating System principles (SOSP 2003), Bolton Landing, NY, October 2003