INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team Regal

Resource management in large scale distributed systems

Rocquencourt

THEME COM

Activity Report

2004

Table of contents

1. Team

Regal is a joint team between INRIA, CNRS and Paris 6 University, through the "Laboratoire d'Informatique de Paris 6", LIP6 *(UMR 7606).*

Head of project-team
Pierre Sens [Professor University of Paris 6]

Vice-head of project-team
Mesaac Makpangou [Research associate (CR) INRIA]

Administrative assistant
Sylvaine Sperte [ITA]

Staff member LIP6
Luciana Arantes [Associate professor University of Paris 6]
Bertil Folliot [Professor Univeristy of Paris 6]
Oliver Marin [Associate professor Univeristy of Paris 6]

Postdoct
Guillaume Vauvert [INRIA Post doctoral research fellow]
Ahmed Jebali [Assistant professor (ATER) University of Paris 10]

Ph. D. student
Marin Bertier [University of Paris 6]
Jean-Michel Busca [INRIA]
Ikram Chabbouh [University of Tunis]
Charles Clement [University of Paris 6]
Corina Ferdean [INRIA]
Nicolas Gibelin [INRIA]
Assia Hachichi [University of Paris 6]
Cyril Martin [University of Paris 6]
Fabio Piconni [University of Paris 6]
Julien Sopena [University of Paris 6]
Gaël Thomas [University of Paris 6]

2. Overall Objectives

Regal is a joint research team between LIP6 and INRIA-Rocquencourt. Regal focuses on management of large scale distributed systems (especially P2P and Grid architectures). A significant axis of the project work is devoted to large scale replication for applications which have strong constraints in terms of dynamicity (multi-agents applications, resource servers on the Web).

3. Scientific Foundations

Keywords: *Grid computing*, *Peer-to-peer*, *consistency*, *distributed system*, *dynamic adaptation*, *fault tolerance*, *large scale environments*, *replication*.

Scaling to large configurations is one of the major challenges addressed by the distributed system community lately. The basic idea is how to efficiently and transparently use and manage resources of millions of hosts spread over a large network. The problem is complex compared to classical distributed systems where the number of hosts is low (less than a thousand) and the inter-host links are fast and relatively reliable. In such "classical" distributed architectures, it is possible and reasonable to build a single image of the system so as to "easily" control resource allocation.

In large configurations, there is no possibility to establish a global view of the system. The underlying operating system has to make decisions (on resource allocation, scheduling ...) based only on partial and possibly wrongs view of the resources usage.

Scaling introduces the following problems :

- Continuous failures occurrence: as the number of hosts increases, the probability of a host failure converges to one. For instance if we consider a classical host MTBF (Mean Time Between Failure) equals to 13 days, in a middle scale system composed of only 10000 hosts, a failure will occur every 4 minutes. Compared to classical distributed systems, failures are more common and have to be efficiently processed.
- Asynchronous networks: clearly on the Internet network transmission delays are very dynamic and unbounded. The asynchronous nature of the Internet makes it extremely problematic to control the consistency of the system. This problem is mainly due to the Fischer, Lynch, Paterson impossibility [13] which shows that a basic feature such as consensus cannot be deterministically solved in an asynchronous system subject to only one crash failure. The fundamental difficulty is that the system has to compose on possibly wrong views of the system where hosts suspected faulty are correct, or where really faulty host are seen in a correct state.
- Failure models: classical distributed systems usually consider crash or omission failures. In a large scale context like peer-to-peer overlay networks, such an assumption is not reasonable since hosts can not only be affected by failures but can be attacked and consequently become malicious. Clearly the failure model has to be extended to deal with a possibly Byzantine behavior of hosts.

Two architectures in relation with the scaling problem have emerged during the last years :

Grid computing : Grid computing offers a model for solving massive computational problems using large numbers of computers arranged as clusters interconnected by a telecommunications infrastructure as internet, renater or VTHD. Grid computing focuses on the ability to support computation across administrative domains sets it apart from traditional distributed computing.
Grid computing has the design goal of solving problems too big for any single supercomputer, whilst retaining flexibility in order to work on multiple smaller problems. It involves sharing heterogenous resources (based on different platforms, hardware/software architectures, and computer languages), located in different places belonging to different administrative domains over a network using open standards. In short, it involves virtualizing computing resources.
On the other hand, if the number of involved hosts can be high (several thousands), the global environment is relatively controlled and users of such systems are usually considered safe and only submitted to host crash failures (typically, Byzantine failures are not considered).

Peer-to-peer overlay network : Generally, a peer-to-peer (or P2P) computer network is any network that does not rely on dedicated servers for communication but, instead, mostly uses direct connections between clients (peers). A pure peer-to-peer network does not have the notion of clients or servers, but only equal peer nodes that simultaneously function as both "clients" and "servers" with respect to the other nodes on the network.
This model of network arrangement differs from the client-server model where communication is usually relayed by the server. In a peer-to-peer network, any node is able to initiate or complete any supported transaction with any other node. Peer nodes may differ in local configuration, processing speed, network bandwidth, and storage capacity.
Different peer-to-peer networks have varying P2P overlays. In such systems, no assumption can be made on the behavior of the host and Byzantine behavior has to be considered.

Regal is interested in how to adapt distributed middleware to these large scale configurations. We target Grid and Peer-to-peer configurations. This objective is ambitious and covers a large spectrum. To reduce its spectrum, Regal focuses on fault tolerance, replication management, and dynamic adaptation. Theses areas rely on competences developed for many years by the members of the team.

Basically, Regal proposes the use of *reactive replication* to tolerate faults and to reduce the access time to get data, while adapting dynamically to the environmental constraints and the evolution of application behavior. Regal concentrates on the deployment of applications (code and data) adapted to highly distributed environments.

We concentrate on the following research themes:

Data management: the goal is to be able to deploy and locate effectively data while maintening the required level of consistency between data replicas.

System monitoring and failure detection: we envisage a service providing the follow-up of distributed information. Here, the first difficulty is the management of a potentially enormous flow of information which leads to the design of dynamic filtering techniques. The second difficulty is the asynchronous aspect of the underlying network which introduces a strong uncertainty on the collected information.

Adaptive replication: we design parameterizable techniques of replication aiming to tolerate the faults and to reduce information access times. We focus on the runtime adaptation of the replication scheme by (1) automatically adjusting the internal parameters of the strategies and (2) by choosing the replication protocol more adapted to the current context.

The dynamic adaptation of application execution support: the adaptation is declined here to the level of the execution support (in either of the high level strategies). We thus study the problem of dynamic configuration at runtime of the low support layers.

4. Application Domains

Keywords: *Internet services*, *data storage*, *multi-agent systems*.

As we already mentioned, we focus on two kinds of large scale environments : computational grids and peer-to-peer (P2P) systems. Although both environments have the same final objective of sharing large sets of resources, they initially emerged from different communities with different context assumptions and hence they have been designed differently. Grids provide support for a large number of services needed by scientific communities. They usually target thousands of hosts and hundreds of users. Peer-to-peer environments address millions of hosts with hundreds of thousands of simultaneous users but they offer limited and specialized functionalities (file sharing, parallel computation).

In peer-to-peer configurations we focus on the following applications :

- Internet services such as web caches or content distribution network (CDN) which aim at reducing the access time to data shared by many users,
- Data storage of mutable data. Data storage is a classical peer-to-peer application where users can share documents (audio and video) across the Internet. A challenge for the next generation of data sharing systems is to provide update management in order to develop large cooperative applications.

In Grid configurations we address resource management for two kinds of applications :

- Multi-agent applications which model complex cooperative behaviors.
- Application Service Provider (ASP) environments in cooperation with the DIET project of the GRAAL team.

5. Software

5.1. DARX (Dynamic Agent Replication eXtension)

Participants: Pierre Sens [correspondent], Marin Bertier, Olivier Marin.

DARX is a framework for building applications that provides adaptive fault tolerance. It relies on the fact that multi-agent platforms constitute a very strong basis for decentralized software that is both flexible and scalable, and makes the assumption that the relative importance of each agent varies during the execution. DARX groups solutions which facilitate the creation of multi-agent applications in a large-scale context. Its most important feature is adaptive replication: replication strategies are applied on a per-agent basis according to transient environment characteristics such as the criticity of the agent for the computation, the network load or the mean time between failures.

DARX constitutes a solution for the automation of adaptive replication schemes, supported by a low-level architecture which addresses scalability issues. The latter is composed of several services.

- A failure detection service which maintains the dynamic lists of all the running DARX servers as well as of the valid replicas which participate to the current application, and notifies the latter of suspected failure occurrences.
- A naming and localization service generates a unique identifier for every replica in the system, and returns the addresses of all the replicas of a same group as a response to an agent localization request.
- A system observation service monitors the behavior of the underlying distributed system: it collects low-level data by means of OS-compliant probes

5.2. Pandora

Participants: Mesaac Makpangou [correspondent], Simon Patarin.

Pandora is a general purpose monitoring platform. It offers a high level of flexibility, while still achieving good performance. Each monitoring task executed by Pandora is splitted into basic and self-contained building blocks called components. These components are chained within stacks to constitute high level tasks. Stack execution consists of components exchanging messages – data structures called "packets" – from the beginning of the stack, to the end.

Pandora provides a framework dealing mainly with packet demultiplexing, timers, threads and communication management. This allows programmers to concentrate on the precise functionalities they want to implement and promotes code reuse.

Pandora has been used by many applications:

- Monitoring of HTTP traffic,
- On-line measurement of web proxy cache efficiency,
- Analysis of the MLdonkey peer-to-peer file-sharing program.

5.3. Sabbarus

Participants: Mesaac Makpangou [correspondent], Ahmed Jebali.

The Sabbarus framework supports a replication scheme in which replicas need not be connected to the whole network all the time. Sabbarus is written on top of Ensemble [16]. It performs a simple, and compact interface to write and administrate replicated applications. The main functionalities of Sabbarus are :

- a library for developing client applications accessing replicated objects,
- a mechanism for defining specific access to replicated objects,
- a command line interface for controlling the framework.
- a mechanism for controlling the degree of divergence accepted by an application (according to its semantic).

5.4. YNVM

Participant: Bertil Folliot [correspondent].

The YNVM (YNVM is Not a Virtual Machine) is a tool based on the VVM technology to build dynamically reconfigurable systems. It provides extreme *late binding* of system/language features, unconstrained application involvement in management and use of meta-data, maximally open reflective features, and elevates concrete implementation (compiled code) to first-class status by providing dynamic (re)compilation of any part of a live system. YNVM is compatible with exsitng code bases, and application buils on top of it can be made highly reconfugrable with low effort and negligible overheads.

6. New Results

6.1. Introduction

In 2004, we focus our research on the following areas :

- Distributed algorithm for Grid configuration.
- Data storage in peer-to-peer system.
- Dynamic adaption of virtual machines.
- Deployment of decentralized applications
- Replication strategies.

6.2. Distributed algorithms

In this theme, we study the adaptation of classical distributed algorithms to large scale configurations. We actually study the adaptation of two basic blocks of distributed systems : failure detection and synchronization mechanisms.

6.2.1. Failure detection

Failure detectors are well-known as a basic building block for fault-tolerant distributed systems. Chandra and Toueg introduced in [6] the concept of *unreliable failure detector*. By adding these detectors to an asynchronous system, they have shwon that it is possible to solve the Consensus problem. The Consensus is the “greatest common denominator” of agreement problems (such as atomic broadcast or atomic commit). Fiscqher, Lynch, and Paterson [13] have shown that consensus cannot be solved deterministically in an asynchronous system that is subject to even a single crash failure. This impossibility results from the inherent difficulty of determining whether a process has actually crashed or is only “very slow”.

Failure detector can be seen as one oracle per process. An oracle provides a list of processes that it currently suspects of having crashed. Many fault-tolerant algorithms have been proposed [15][11][2] based on unreliable failure detectors, but there are few studies of the implementation of theses detectors [19] and, to our knowledge, there is no implementation targeting dynamic and large environments such as P2P and Grid.

We propose a new failure detector implementation. This implementation is a variant of the heartbeat detector which is adaptable and can support scalable applications. Our algorithm is based on all-to-all communications where each process periodically sends an “*I am alive*” message to all processes using IP-Multicast capabilities. To provide a short detection delay, we automatically adapt the failure detection time as a function of previous receptions of “*I am alive*” messages. The *Eventually Perfect* failure detector ($\diamondsuit P$) is *reducible* to our implementation in models of partial synchrony [10][29].

Failure detectors are designed to be used over long periods where the need for *quality of detection* alters according to applications and systems evaluation. In practice, it is well known that systems are subjected to variations between long periods of instability and stability. We evaluate the maximal quality of service that the network can support in terms of detection time. Then we propose a heuristic to adapt the sending period of "*I am alive*" messages as a function of the network QoS and the application requirements.

Then, in order to adapt our detector to large configurations, we define a hierarchical version. This architecture allows to decrease the number of messages exchanged as well as the processor load. Our detectors are organized as a two level architecture: one level for intra-cluster failure detection and a second level for inter-cluster monitoring.

There are several perspectives to this work :

- The adaption of unreliable failure detectors to other large environments like peer-to-peer overlay networks.
- Generalization of our 2-level hierarchical architecture to N-level. This extension could allow us to manage Grid configurations built as an the interconnection of large clusters.

6.2.2. Synchronization

Many distributed and parallel applications benefit from Grid infrastructure, which enables the sharing of a wide variety of geographically distributed resources, acting as a single powerful computer. However, such applications usually require that their processes get exclusive access to one or more of these shared resources. Therefore, mutual exclusion becomes crucial for Grid computing.

Many distributed algorithms have been proposed to solve the problem of mutual exclusion in distributed systems, serializing concurrent accesses to a shared resource. They can be divided into two groups: *permission-based* (e.g. Lamport [18], Ricart-Agrawala [25], Singhal [27], Maekawa [20]) and *token-based* (Suzuki-Kazami [28], Raymond [24], Naimi-Trehel [22], Neilsen-Mizuno [23], Chang, Singhal and Liu [7]). The first group of algorithms are based on the principle that a node accesses the critical section only after having received permission from all the other nodes (or the majority of them [20]). In the second group of algorithms, a system-wide unique token is shared among all nodes and the possession of it gives a node the exclusive right to enter into critical section. The latter usually has a lower average message cost and often result in logarithmic message complexity O(log N) with regard to the number of nodes. The majority of O(log N) token-based algorithms are tree-based, i.e. a logical tree structure expresses the different paths of token requests and its propagation at a given time.

Since in Grid environments the number of nodes can be very large, scalability of a distributed mutual exclusion algorithm is an important feature. Considering that tree-based token mutual exclusion algorithms scale quite well, they seem to be very suitable for Grid applications. However, these algorithms do not take into account the heterogeneity of communication latency in Grid environments. For instance, the latency between machines of different clusters can be much higher than the latency between nodes within a single cluster. Consequently, the performance of those algorithms can be rather critical for Grid applications.

We propose a number of hierarchical mutual exclusion algorithms. These algorithms are extensions of Naimi-Trehel's token algorithm, reducing the latency and the number of messages exchanged between distant hosts. These extensions follow three approaches: (1) an approach based on hierarchical proxy-based approach, (2) the aggregation of requests, and (3) token preemption by closer hosts.

We compared the performance of these algorithms on an emulated Grid testbed. Actually, we study the impact of each of the extensions, showing that the combination of them can greatly improve the performance of the original algorithm.

There are several perspectives to these primar works :

- Extension of our algorithms to other synchronization mechanisms such as non exclusive locking.
- Fault tolerant locking. We are presently studying how to recover from the failure of the token owner. We have already defined a fault-tolerant version of the Naimi-Trehel algorithm and we plan to adapt this algorithm for grid configurations.

- To relax the synchrony model of the underlying network. In our previous wrok, we considered a synchronous system and we envisage to extend to a more realistic partial synchrony proposed by Chandra and Toueg in [6]

6.3. Storage in peer-to-peer systems

Although many peer-to-peer file systems have been proposed by different research groups during the last few years [8][1][17][21][9][26], only a handful are designed to scale to hundreds of thousands of nodes and to offer read-write access to a large community of users. Moreover, very few prototypes of these large-scale multi-writer systems exist to this date, and the available experimental data are still very limited.

One of the reasons for this is that, as the system grows to a very large scale, allowing updates to be made anywhere anytime while maintaining consistency, ensuring security, and achieving good performances is not an easy task. Read-only systems, such as CFS [9], are much easier to design since the time interval between meta-data updates is expected to be relatively high. This allows the extensive use of caching, since cached data are either seldom invalidated or kept until they expire. Security in a read-only system is also quite simple to implement. Digitally signing a single root block with the administrator's private key and using one-way hash functions allows clients to verify the integrity and authenticity of all file system data. Finally, consistency is hardly a problem since only a single user, the administrator, can modify the file system.

Multi-writer designs must face a number of issues not found in read-only systems, such as maintaining consistency between replicas, enforcing access control, guaranteeing that update requests are authenticated and correctly processed, and dealing with conflicting updates.

The Ivy system [21], for instance, stores all file system data in a set of logs using the DHash distributed hash table. In Ivy each update is stored by appending a record to a log. Since records are never removed from the logs, every client has access to the whole file system history, which greatly simplifies conflict detection and resolution. Furthermore, each Ivy user has her own log to which she appends her own updates. This has two advantages: first, writes are fast since there is no central serialization point (like Oceanstore's primary tier), and second, data cannot be overwritten by a malicious user since only the log owner can append data to it. However, as the number of users sharing a given file system increases, the number of logs that need to be traversed to satisfy a read operation also becomes larger, thus increasing network traffic. Although the number of DHash servers can grow to hundreds of thousands, the number of Ivy users sharing a given file does not scale. Another problem in Ivy is that applications have little control over the consistency of data. Although Ivy uses a consistency model similar to close-to-open consistency, applications cannot fully decide when written data are propagated to the network (this is due to the lack of a CLOSE RPC in the NFS v3 protocol specification).

Oceanstore [17] uses a completely different approach for updates handling by introducing some degree of centralization. A primary tier of nodes uses a Byzantine-fault tolerant (BFT) [5] algorithm to serialize all file system updates coming from secondary tier nodes. Since BFT is quite expensive, primary tier nodes must be highly resilient nodes located in high-bandwidth areas of the network. Oceanstore's designers assume that these nodes will be set up and maintained by a commercial service provider. Thus, Oceanstore may not be suitable for a community of cooperative users wishing to use a system which does not depend on a centralized authority. Oceanstore also takes into account network locality to optimize replica location. An introspection layer provides information about the network conditions, allowing the system to dynamically adapt itself to the current environment. Locality management is absent in Ivy.

Pangaea [26] differs from Ivy and Oceanstore in that it does not rely on a key-based routing layer. Instead, object location is achieved by maintaining a graph of live replicas through which updates are propagated by flooding a special message called harbinger. Moreover, a replica of a file or directory is created on each client that accesses the file. Although this reduces read latency, it can generate an important amount of traffic when updates are propagated.

We currently developed two file systems for peer-to-peer configurations : POST in cooperation with Rice University and Pastis.

POST is a decentralized messaging infrastructure that supports a wide range of collaborative applications, including electronic mail, instant messaging, chat, news, shared calendars and whiteboards. POST is highly resilient, secure, scalable and does not rely on dedicated servers. Instead, POST is built upon a peer-to-peer (p2p) overlay network whom participants desktop computers. POST offers three simple and general services, such as (i) secure, single-copy message storage, (ii) meta-data based on single-writer logs, and (iii) event notification.

Pastis is a completely decentralized multi-user read-write peer-to-peer file system. In our system every file is described by a modifiable inode-like structure which contains the addresses of the immutable blocks in which the file contents are stored. All data is stored using the Past distributed hash table, which is built on top of the Pastry peer-to-peer network overlay. Authentication and integrity is assured using standard cryptographic mechanisms. Our system takes advantage of the fault tolerance and locality properties of both Past and Pastry. This allows us to optimize message routing and replica retrieval so that the performance cost caused by network latency is kept at a minimum. We have also introduced a modification to the Past DHT which allows us to further increase performance when using a relaxed but nevertheless reasonable consistency model. We have developed a prototype in order to evaluate the performance of our design. Our prototype is programmed in Java and uses the FreePastry open-source implementation of Past and Pastry. It currently provides a proprietary Java interface to applications, and allows them to choose between two degrees of consistency. Preliminary results obtained suggest that our system is approximately twice as slow as NFS.

6.4. Virtual virtual machine (VVM)

The VVM team works on flexible execution environments, based on a HAL (Hardware Abstraction Layer) and a flexible dynamic compiler, free of any predefined, imposed abstractions. This high level of dynamic flexibility allows the dynamic construction of dedicated execution environments as well as their dynamic reconfiguration. To demonstrate the benefits of this approach, we used this minimal execution environment to build the JNJVM (JNJVM is Not a JVM), a dynamically adaptable Java runtime. The inherent flexibility of the JNJVM allows us to address some of the limitations of traditional rigid Java runtimes, in particular concurrency management and concurrent programming support. Other works concern A2N (Active Active Network), a minimal active node architecture that has the capability of being dynamically specializable at runtime through capsules, and C/SPAN a self-adapting Web proxy cache (based on PANDORA) that applies administrative strategies to adapt itself and react to external events. Because C/SPAN is completely flexible, even these adaptation policies can be dynamically adapted.

Within the VVM project we continue to investigate a systematic approach for building flexible, adaptable and interoperable execution environments, to free applications from the artificial limitations on reconfiguration imposed by programming environments. The HAL part of the minimal execution environment is currently being embedded in a Linux kernel-module so that our flexible Java runtime can be used on top of a traditional operating system, as a standard JVM, while preserving a maximum of flexibility. Concerning the JNJVM, we plan to further develop our flexible Java environment towards a complete flexible Java-OS and also to integrate it in the context of component based middleware, such as the Corba Component model, to illustrate the adequacy of our approach to the needs of modern distributed applications. This work is being made within the IST COACH European project.

6.5. Deployment of decentralized applications

In 2004, we propose a new platforme, Deal. Deal is a Language and System Support for Active Deployment and Management of Distributed Applications

Deal is a component-based distributed application model that matches Internet applications requirements and describes a language and a system substrate, called *Deal*, that supports this model. *Deal* simplifies greatly the release tasks for developers who want to release applications, the deployment tasks for administrators who deploy and control the execution of applications, and the application access for users who want to benefit from services delivered by running applications. *Deal* provides an Architecture and Deployment Description

Language. The *Deal* Architecture and Deployment Description Language (DAD2L) supports the basics functions of any architecture description language. These include the description of the software components and of component interfaces, the architectural configuration, and the specification of the connectors that allow components to communicate with one another. In addition, *DAD2L* permits the description of the distribution of components, the placement criteria, and the reconfiguration policy (i.e., automatic adaptation of the placement and replication of application components). Finally, *Deal* offers a system support and some supporting tools to facilitate the use of *DAD2L*.

6.6. Replication strategies

The effectiveness of replication, as a key technique to improve the clients perceived QoS when accessing large-scale distributed services, depends on decisions such as the selection of the suitable replica to assign to a client, when to create new replicas, where to place new replicas or when existing replicas become useless. Our work consists in developing a dynamic replication infrastructure aimed to improve the service delivered QoS (in terms of availability, throughput, and response time), as perceived by the clients, while consuming as little system and network resources as possible. We base our decisions for the replica management on the providers/clients requirements, aggregated in a replication contract, comprising policies for replica selection, placement, creation, migration, elimination, and consistency.

Concerning the replica selection problem, the criteria for assigning replicas to requesting clients should consider the metrics that contribute to the user-perceived QoS and some means to estimate this metrics for each replica. For example, a replicated Web service, aimed to minimize the documents transfer time, should consider link latency metric when distributing small documents and the available bandwidth for large documents (as in this case, the competing traffic becomes the primary factor limiting the document transfer time. Also, if the client requests involve intensive computing tasks on the replica server, the replica host load is an important metric to be taken into consideration. In the general case, the applications providers and the clients have different views of what a suitable replica means, according to the application characteristics, the guarantees that the application provider wishes to offer to its clients, and the clients preferences. Unfortunately, most previous works on replica selection use the same metrics for all applications and clients. This uniformity limits the efficiency of these strategies when they face the diversity of applications to be replicated and the variety of clients preferences.

Concerning replicas placement, the application providers could have different policies in terms of preferred (geographical/topological) locations where new replicas should be placed, and also in terms of statical/dynamical physical and software resources that the replica host should own. As far as we know, existing work on replica management face the same limitation of using replication criteria predefined for all applications (existing replication criteria only adapt eventually to the clients access patterns.

Our current work on replication is focused on the design and implementation of a consistency meta-model, aiming to provide fine-grained customization for replicated data objects.

Our work aims to provide consistency management customized to the semantics and different consistency requirements of a variety of Data Objects. We consider the classical model, where a Data Object encapsulates any data and provides various services to access that data. We replicate Data Objects by considering the scenario of active replication. The replica consistency management consists mainly in the right delivery of the operation calls at all replicas. We model this by following a state-machine approach. We obtain two state machines, containing the stages met by an operation call, at its replica of issuance respectively at a peer, before being executed all over. We transpose this representation within a framework, wherein different components perform different transitions. The transitions are parameterized by the semantics and by the consistency needs of the operation calls . The semantics indicates if this call is a read or a write and, in the latter case, its impact on the data that it modifies. The consistency needs are expressed in terms of the divergence tolerated with respect to the real object, the instant when the call should be made visible to peers, the call's ordering and conflicting relations.

7. Other Grants and Activities

7.1. National initiatives

7.1.1. Distributed algorithms and application CNRS Grant

Members: LIAFA, IRISA (Adept Team), LRI, Regal, EPFL

Objectives: This working group leaded by Hugues Fauconier and Carole Delporte (LIAFA) focuses on the application of theorical results in distributed algorithms on the large scale environments

7.1.2. ENWEG CNRS Grant

Members: LRI, CEA, ID-IMAG, INRIA Sophia, IRISA, LABRI, LIFC, LIFL, LIP, LIP6, LORIA, LRI, PRISM

Objectives: This project leaded by Franck Cappello (LRI) aims to identify issues (scientific and technical) and proposes solutions following the overall goal of building an experimental Grid platform gathering nodes geographically distributed in France.

7.1.3. Grid Data Service - ACI MD (2003-2005)

Members: IRISA (Paris Team), ENS-Lyon (LIP - Remap Team), Regal

Objectives: The goal of this project is to propose an approach where the grid computation is decoupled from data management, by building a data sharing service adapted to the constraints of scientific grid computations. The main goal of this project is to specify, design, implement and evaluate a data sharing service for mutable data and integrate it into the DIET ASP environment developed by the ReMaP team of LIP. This service will be built using the generic JuxMem platform for peer-to-peer data management (currently under development within the PARIS team of IRISA, Rennes). The platform will be used to implement and compare multiple replication and data consistency strategies defined together by the PARIS team (IRISA) and by the REGAL team of LIP6.

7.1.4. Data Grid eXplorer (2003-2005)

Members: IMAG-ID, Laria, LRI, LAAS, LORIA, LIP Ens-Lyon, LIFL, INRIA Sophie Antipolis, LIP6, IBCP, CEA, IRISA INRIA Rocquencourt

Objectives: The goal of Data Grid Explorer is to build an emulation environment to study large scale configurations. Today, it is difficult to evaluate new models for data placement and caching, network content distribution, peer-to-peer systems, etc. Options include writing simulation environments from scratch, employing detailed packet-level simulation environments such as NS, local testing within a controlled cluster setting, or deploying live code across the Internet or a Testbed. Each approach has a number of limitations. Custom simulation environments typically simplify network and failure characteristics. Packet-level simulators add more realism but limit system scalability to a few hundred of simultaneous nodes. Cluster-based deployment adds another level of realism by allowing the evaluation of real code, but unfortunately the network is highly over-provisioned and uniform in its performance characteristics. Finally, live Internet and Testbed deployments provide the most realistic evaluation environment for wide-area distributed services. Unfortunately, there are significant challenges to deploying and evaluating real code running at a significant number of Internet sites. The main benefit of emulation is the ability to reproduce experimental conditions and results.
The project is structured horizontally into transverse working groups: Infrastructure, Emulation, Network, and Applications. The Regal team is leader for the Emulation working group.

7.1.5. ACI Grid DataGraal (2002 - 2004)

Members: Regal (coordinator), projet PARIS (IRISA), LRI, Remap team (LIP Lyon), LISI (Lyon), IMAG-ID, IMAG-LSR, SMIS team, CEA, CESR, LIRMM,

Objectives: DataGraal is a discussion forum about data management, systems and applications for large scale environments. The motivation is to benefit from the experiences of different communities: system, database, and scientific applications.
The objective is to understand the link between grid computing and distributed information systems. We investigate the impact of the large scale environment, which represents a challenge, at the same time for calculation on grid and the distributed information systems. We aim: (1) to specify the useful concepts and the common approaches on mobility, availability and distribution of huge amount of data, (2) to make emerge some federative projects from these communities.

7.1.6. Gedeon - ACI MD (2004-2006)

Members: IMAG-ID, IMAG-LSR, IBCP, Regal

Objectives: File systems (FS) are commonly used to store data. Especially, they are intensively used in the community of large scientific computing (astronomy, biology, weather prediction) which needs the storage of large amouts of data in a distributed manner. In a GRID context (cluster of clusters), traditional distributed file systems have been adapted to manage a large number of hosts (like the Andrew File System). However, such file systems remain inadequate to manage huge data. They are suited for traditional unix (small) files. Thus, the grain of distribution is typically an entire file and not a piece of file which is essential for large files. Furthermore, the tools for managing data (e.g, interrogation, duplication, consistency) are unsuited for large sizes.
Database Management Systems (DBMS) provides different abstraction layers, high level languages for data interrogation and manipulation etc. However, the imposed data structuration, the low distribution, and the usually monolithic architecture of DBMSs limit their utilisation in the scientific computing context.
The main idea of the Gedeon project is to merge the functions of file systems and DBMS, focusing on structuration of meta-data, duplication and coherency control. Our goal is NOT to build a DBMS describing a set of files. We will study how database management services can be used to improve the efficiency of file access and to increase the functionality provided to scientific programmers.

7.2. European initiatives

7.2.1. IST Coach (2002-2004)

Members: T-Systems, Humboldt Univeristaet zu Berlin, Intracom, Lucent, Thales, CNRS, ObjectSecurity, LIP6, LIFL, FOKUS

Objectives: COACH is concerned with the rapid and cost effective development of large scale and mission critical distributed applications by using software components. These components are based on the OMG's CORBA Component Model (CCM), which is a specification for creating distributed, server-side scalable, component-based, language-neutral, transactional, multi-user, and secure applications. The key objective is to build a component framework that is well integrated with state of the art software engineering techniques like the OMG's Model Driven Architecture. The framework can rapidly transform models, architecture and design level components, as well as policies to execution level and deploy them efficiently and securely on distributed hardware platforms. This allows the developer to concentrate on the business logic instead of reinventing technical infrastructure and to reuse existing components, thus increasing software quality and greatly reducing development costs and time to market.

8. Dissemination

8.1. Program committees and responsibilities

Luciana Arantes was member of the program committee of:

- *The 2003 International Multiconference in Computer Science and Computer Engineering*. June 2003, Las Vegas, USA.
- CDUR'2005 - *Journées Francophones sur la Cohérence des Données en Univers Réparti*, 2005, France

Bertil Folliot was member of the program committee of:

- 4th International Symposium on Parallel and Distributed Computing, Lille, July 2005.
- IADIS International Conference on Applied Computing, Lisbonne, Portugal, March 2004.
- 3nd International Symposium on Parallel and Distributed Computing, Cork, Irlande, July 2004.
- IFIP Symposium on Computer Architecture and High Performance Computing, Foz do Iguaçu, Brésil, October 2004.

Mesaac Makpangou was member of the program committee of:

- Medianet'2004 - *International Conference on Intelligent Access of Multimedia Documents on Internet* , June 2002, Sousse, Tunisie.

Pierre Sens was member of the program committee of:

- AC'2004 - *IADIS Applied Computing 2004 Conference*
- CFSE'2005- *Conférence française sur les systèmes d'Exploitation*. 2005, France.
- CDUR'2005 - *Journées Francophones sur la Cohérence des Données en Univers Réparti*, 2005, France

P. Sens is in charge of cooperation between University of Paris 6 and the Oradea University (Roumania).
P. Sens is vice-chair of the French Chapter of ACM SIGOPS

8.2. PhD reviews

Bertil Folliot was PhD rewiever of :

- Cyril Guilloud, “Traçage flexible d’exécutions de programmes parallèles”, Thèse de Doctorat de l’Institut National Polytechnique de Grenoble (advisors: Jacques Chassin de Kergommeaux, Brigitte Plateau), Grenoble, février 2004.
- Patricia Pascal, “Gestion de ressources pour des services déportés sur des grappes d’ordinateurs avec qualité de service garantie”, Thèse de Doctorat de l’INSA Toulouse (advisors : Gérard Authie, Thierry Monteil), Toulouse, novembre 2004.
- Romain Lenglet, “Composition flexible et efficace de transformations de programmes”, Thèse de Doctorat de l’Institut National Polytechnique de Grenoble (advisors : Daniel Hagimont, Thierry Coupaye), novembre 2004.

Pierre Sens was PhD rewiever of :

- Amed Mokhtar, “Réplication de données dans les réseaux hybrides pour améliorer les performances”, IRISA, Univeristé Rennes 1, January 2004, (advisor: G. Rubato)
- Goeffroy Vallée, “Conception d’un ordonnanceur de processus adaptable pour la gestion globale des ressources dans les grappes de calculateurs : mise en ½uvre dans le système d’exploitation Kerrighed”, IRISA, Univeristé Rennes 1, March 2004 (advisor: C. Morin)
- Philippe Raipin, “Accord tolérants les fautes dans systèmes répartis synchrones et asynchrones”, IRISA, Univeristé Rennes 1, October 2004 (advisors: M. Hurfin, M. Raynal)
- Corine Marchand, “Detection de fautes dans les réseaux ad-hoc”, IMAG-ID, Institut National Polytechnique de Grenoble, December 2004 (advisors: B. Plateau, J-M. Vincent)

8.3. Teaching

- Luciana Arantes :
  - Principles of operating systems in Licence d’Informatique, Université Paris 6
  - Operating systems kernel in Maîtrise d’Informatique, Université Paris 6
  - Responsible for projects in operating system, Maîtrise d’Informatique, Université Paris 6
- Bertil Folliot :
  - Principles of operating systems in Licence d’Informatique, Université Paris 6
  - Distributed algorithms and systems in Magistère d’Informatique, Université Paris 6
  - Distributed programming, DESS DLS, Université Paris 6
  - Distributed systems and client/serveur in Maîtrise d’Informatique, DEA Systèmes Répartis, Université Paris 6
  - Responsible for the DEA Systèmes Informatique Répartis, Université Paris 6, CNAM, ENST
  - Projects in distributed programming, Maîtrise d’Informatique, Université Paris 6
- Mesaac Makpangou
  - Distributed systems, DESS IRS, Université de Versailles Saint-Quentin en Yvelines

  - Performance of distributed systems, DEA MISI, Université Versailles Saint-Quentin
  - Large Scale Data sharing, DEA IFA, Université de Marne-La-Vallee
  - Distributed systems and applications, Institut Superieur de Technologies et de management (ISTM)
  - Systems and networks, Master, Pôle Universitaire Leonard de Vinci

- Oliver Marin

  - Operating system programming, Master d'Informatique, Univeristé Paris 6
  - Operating system Principles, Licence d'Informatique, Univeristé Paris 6 Université Paris 6
  - Parallel and distributed systems,Master d'Informatique, Université Paris 6
  - Client/server arcitecture in Licence professionelle d'Informatique, Université Paris 6

- Pierre Sens

  - Principles of operating systems in Licence d'Informatique, Université Paris 6
  - Operating systems kernel in Maîtrise d'Informatique, Université Paris 6
  - Responsible for Operating System and Network part of the Maîtrise d'Informatique, Université Paris 6
  - Animation of the working group on advances in distributed system, DEA SIR, Université Paris 6

9. Bibliography

Major publications by the team in recent years

[1] A. ADYA, W. BOLOSKY, M. CASTRO, R. CERMAK, J. R. DOUCEUR, J. HOWELL, M. LORCH, W. R. P.. *FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment*, in "5th Symposium on Operating Systems Design and Implementation (OSDI 2002)", December 2002.

[2] M. K. AGUILERA, W. CHEN, S. TOUEG. *Using the heartbeat failure detector for quiescent reliable communication and consensus in partitionable networks*, in "Theoretical Computer Science", vol. 220, n° 1, June 1999, p. 3–30.

[3] B. BERSHAD, S. SAVAGE, P. PARDYAK, E. SIRER, M. FIUCZYNSKI, D. BECKER, C. CHAMBERS, S. EGGERS. *Extensibility, safety and performance in the SPIN operating system*, in "ACM Operating Systems Review", vol. 29, n° 5, 1995, p. 267-284.

[4] K. P. BIRMAN. *Replication and Fault-Tolerance in the ISIS System*, in "ACM Operating Systems Review", Proc. 10th ACM Symp. on Operating System Principles, Orcas Island, WA, USA, December 1985, vol. 19, n° 5, 1985, p. 79-86.

[5] M. CASTRO, LISKOV. *Practical Byzantine fault tolerance*, in "Proc. of USENIX Symposium on Operating Systems Design and Implementation (OSDI 1999)", 1999.

[6] T. D. CHANDRA, S. TOUEG. *Unreliable failure detectors for reliable distributed systems*, in "Journal of the ACM", 1996.

[7] I. CHANG, M. SINGHAL, M. T. LIU. *An improved O(Log N) Mutual Exclusion Algorithm*, in "Proceedings of the 1990 International Conference on Parallel Proces sing", August 1990, p. 295-302.

[8] I. CLARKE, O. SANDBERG, B. WILEY, T. HONG. *Freenet: A distributed anonymous information storage and retrieval system*, in "Proc. of the Workshop on Design Issues in Anonymity and Unobser vability", July 2000.

[9] F. DABEK, M. F. KAASHOEK, D. KARGER, D. MORRIS, S. I.. *Wide-area cooperative storage with CFS*, in "Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)", October 2001.

[10] D. DOLEV, C. DWORK, L. STOCKMEYER. *On the Minimal Synchronism Needed for Distributed Consensus*, in "Journal of the ACM", vol. 34, n° 1, 1987, p. 77–97.

[11] D. DOLEV, R. FRIEDMAN, I. KEIDAR, D. MALKHI. *Failure detectors in omission failure environments*, Technical report, n° TR96-1608, Cornell University, Computer Science Department, September 1996.

[12] D. R. ENGLER, M. F. KAASHOEK, J. JAMES O'TOOLE. *Exokernel: An Operating System Architecture for Application-Level Resource Management*, in "Proceedings of the 15th Symposium on Operating Systems Principles (15th SOSP'95), Operating Systems Review, Copper Mountain, CO", Published as Proceedings of the 15th Symposium on Operating Systems Principles (15th SOSP'95), Operating Systems Review, volume 29, number 5, ACM SIGOPS, dec 1995, p. 251-266.

[13] M. J. FISCHER, N. A. LYNCH, M. S. PATERSON. *Impossibility of distributed consensus with one faulty process*, in "Journal of the ACM", vol. 32, n^o 2, apr 1985, p. 374-382.

[14] R. GUERRAOUI. *Indulgent Algorithms*, in "Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC-00), NY", ACM Press, jul 16-19 2000, p. 289-298.

[15] R. GUERRAOUI, M. LARREA, A. SCHIPER. *Non Blocking Atomic Commitement with an Unreliable Failure Detector*, Technical Report, June 1995, http://www.research.ec.org/broadcast/trs/papers/86.ps.

[16] M. HAYDEN. *The Ensemble System*, Technical report, n^o TR98-1662, Cornell University, January 1998.

[17] J. KUBIATOWICZ, D. BINDEL, Y. CHEN, S. CZERWINSKI, D. EATON, R. GUMMADI, S. RHEA, H. WEATHERSPOON, C. WEIMER, Z. B.. *Oceanstore: An architecture for global-scale persistent store*, in "Proc. of ASPLOS'2000", November 2000.

[18] L. LAMPORT. *Time, Clocks, and the Ordering of Events in a Distributed System*, in "CACM: Communications of the ACM", vol. 21, 1978.

[19] M. LARREA, A. FERNÁNDEZ, S. ARÉVALO. *Optimal Implementation of the Weakest Failure Detector for Solving Consensus*, in "Proc. of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC-00), NY", ACM Press, July 16–19 2000, p. 334–334.

[20] M. MAEKAWA. *A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems*, in "ACM Transactions on Computer Systems", vol. 3, n^o 2, May 1985, p. 145–159.

[21] A. MUTHITACHAROEN, R. MORRIS, T. M. GIL, C. B.. *Ivy: A Read/Write Peer-to-Peer File System*, in "Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI 2002)", 2002.

[22] M. NAIMI, M. TREHEL, A. ARNOLD. *A Log (N) Distributed Mutual Exclusion Algorithm Based on Path Reversal*, in "Journal of Parallel and Distributed Computing", vol. 34, n^o 1, 10 April 1996, p. 1–13.

[23] M. L. NEILSEN, M. MIZUNO. *A Dag-Based Algorithm for Distributed Mutual Exclusion*, in "Proceedings of the 11th International Conference on Distributed Com puting Systems (ICDCS), Washington, DC", 1991, p. 354–360.

[24] K. RAYMOND. *A tree-based algorithm for distributed mutual exclusion*, in "ACM Transactions on Computer Systems (TOCS)", vol. 7, n^o 1, 1989, p. 61–77.

[25] G. RICART, A. AGRAWALA. *An Optimal Algorithm for Mutual Exclusion in Computer Networks*, in "CACM: Communications of the ACM", vol. 24, 1981.

[26] Y. SAITO, C. KARAMANOLIS, M. KARLSSON, M. M.. *Taming aggressive replication in the pangaea wide-area file system*, in "5th Symp. on Op. Sys. Design and Implementation (OSDI 2002)", December 2002.

[27] M. SINGHAL. *A dynamic information structure for Mutual Exclusion algorithm for Distrib uted Systems*, in "IEEE Transactions on Parallel Distributed Systems", vol. 3, n^o 1, 1992, p. 121–125.

[28] I. SUZUKI, T. KASAMI. *A distributed mutual exclusion algorithm*, in "ACM Transactions on Computer Systems (TOCS)", vol. 3, n° 4, 1985, p. 344–349.

[29] P. VERISSIMO, A. CASIMIRO, C. FETZER. *The timely computing base: Timely actions in the presence of uncertain timeliness*, in "Proc. of the Int'l Conf. on Dependable Systems and Networks, New York City, USA", IEEE Computer Society Press, june 2000, p. 533-542.

[30] R. VAN RENESSE, K. P. BIRMAN, S. MAFFEIS. *Horus: A flexible Group Communication System*, in "Communication of the ACM", vol. 39, n° 4, apr 1996, p. 76-83.

Doctoral dissertations and Habilitation theses

[31] F. OGEL. *Environnements d'exécution dynamiquement adaptables*, Ph. D. Thesis, May 2004.

Articles in referred journals and book chapters

[32] F. OGEL, G. THOMAS, A. GALLAND, B. FOLLIOT. *MVV : une Plate-forme à Composants Dynamiquement Reconfigurables — La Machine Virtu elle Virtuelle*, in "Numéro Spécial Technique et Science Informatiques (TSI)", J.-L. GIAVITTO (editor)., (to appear), Hermes Science, 2004.

Publications in Conferences and Workshops

[33] M. BERTIER, L. A. P. SENS. *Hierarchical token based mutual exclusion algorithms*, in "Proceedings of the 4th IEEE/ACM International Symposium on Clu ster Computing and the Grid (CCGrid '04), Chicago (USA)", IEEE Society Press, April 2004.

[34] J.-M. BUSCA, M. BERTIER, F. BELKOUCH, P. SENS, L. ARANTES. *Performance Evaluation of a Quorum-Based State-Machine Replication Algorithm for Computing Grids*, in "Proc of the 16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD ' 04)", October 2004.

[35] I. CHABBOUH, M. MAKPANGOU. *A Configuration Tool for Caching Dynamic Pages*, in "9th International Workshop WCW 2004 Beijing, China, October 18-20, 2004 Proceedings", Springer, 2004.

[36] M. M. CORINA FERDEAN. *A Generic and Flexible Model for Replica Consistency Management*, in "1st International Conference on Distributed Computing and Internet Technology (ICDCIT 2004)", 2004.

[37] M. M. CORINA FERDEAN, N. GIBELIN. *The Contract Substrate for Adaptive Replication of Data Objects*, in "2nd International Conference on Intelligent Access of Multimedia Documents on Internet (MediaNet'04)", 2004.

[38] A. HACHICHI, C. MARTIN, G. THOMAS, S. PATARIN, B. FOLLIOT. *Reconfigurations dynamiques de services dans un intergiciel à composants CORBA CCM*, in "1ère Conférence Francophone sur le Déploiement et la (Re) Configuration de Logiciels (DECOR'04), Grenoble, France", october 2004.

[39] F. OGEL, B. FOLLIOT, G. THOMAS. *A Step Toward Ubiquitous Computing: An Efficient Flexible Micro-ORB*, in "proceedings of the 11th ACM SIGOPS European Workshop, Leuven, Belgium", sept. 2004, p. 176–181.

[40] S. Patarin, M. Makpangou. *Pandora : une palte-forme efficace pour la construction d'applications autonomes*, in "1ère Conférence Francophone sur le Déploiement et la (Re) Configuration de Logiciels (DECOR'04), Grenoble, France", october 2004.

[41] F. Picconi, J.-M. Busca, P. Sens. *Exploiting network locality in a decentralized read-write peer-to-peer file systems*, in "Proc of the 10th International Conference on Parallel and Distributed Systems (ICPADS '04), Newport Beach (USA)", IEEE Society Press, July 2004.