INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Runtime

Efficient Runtime Systems for Parallel Architectures

Futurs

THEME NUM

Table of contents

1. Team

Team Leader

Raymond Namyst [Professor, Université Bordeaux 1, LaBRI]

Administrative assistant

Brigitte Larue-Bourdon [Project Assistant]
Corinne Brisset [Project Assistant]

Staff members

Olivier Aumage [Research Associate (CR) Inria]
Alexandre Denis [Research Associate (CR) Inria]
Pierre-André Wacrenier [Assistant Professor, Université Bordeaux 1, LaBRI]

Research scientists (partner)

Marie-Christine Counilh [Assistant Professor, Université Bordeaux 1, LaBRI]

Research engineers

Nathalie Furmento [CNRS]
Christophe Frézier [Associate Engineer, INRIA (starting from Oct. 1st 2005)]
Vincent Danjean [Expert Engineer (ending Aug. 31st 2005)]

Ph.D. students

Marc Perache [CEA Grant]
Samuel Thibault [Ministry of Research and Technology Grant, LaBRI]
Elisabeth Brunet [Regional Grant, LaBRI (starting from Oct. 1st 2005)]

2. Overall Objectives

2.1. Designing Efficient Runtime Systems

Keywords: *SMT, distributed, environment, heterogeneity, parallel, runtime.*

The RUNTIME project seeks to explore the design, the implementation and the evaluation of mechanisms that will form the core of tomorrow's **parallel runtime systems**. More precisely, we propose to define, implement and validate the most generic series of runtime systems providing both an efficient and flexible foundation for building environments/applications in the field of intensive parallel computing. These runtime systems will have to allow an efficient use of parallel machines such as large scale heterogeneous and hierarchical clusters.

By *runtime systems*, we mean intermediate software layers providing the parallel applications with the required additional functionalities and dealing with the high-performance computing specific issues left unaddressed by the operating system and its peripheral device drivers. Runtime systems can thus be seen as functional extensions of operating systems and should be distinguished from high-level libraries. Note that the boundary between a runtime system and the underlying operating system is rather fuzzy since a runtime system may also feature specific extensions/enhancements to the underlying operating system (e.g. extensions to the OS thread scheduler).

The research project centers on three main challenges:

Mastering large scale heterogeneous configurations. We intend to propose new models, principles and mechanisms that should allow to combine communication handling (particularly the case of high-performance routing in heterogeneous context), threads scheduling and I/O event monitoring on such architectures, both in a portable and efficient way. We also intend to study the introduction of the necessary dynamicity, fault-tolerance and scalability properties within this new generation of runtime systems, while minimizing their unavoidable negative impact on the application performance.

Optimally exploiting new technologies. It is definitely mandatory to keep an eye over the evolutions of hardware technologies (networks, processors, operating system design) to better understand the constraints imposed by real production machines and to study how to get the most out of these new technologies. On that particular point, we must undoubtedly carry on the work we have begun about interface expressiveness which allows a separation of the application requirements from the runtime system-generated optimizations. For instance, we are currently experimenting new communication optimization techniques over the Infiniband and Quadrics network technologies. We also study the scheduling of threads on new multi-core, SMT processors.

Improving integration between environments and applications. We are interested in exploring the boundaries between runtime systems and higher level environments in order to expand the scope of our optimization techniques. Several paths will be explored concurrently: 1) the proposal of functional extensions to existing programming interfaces that will reduce the amount of unusable functionalities; 2) the exploitation of information generated by a program analyzer to improve the quality of internal runtime system heuristics; 3) the refinement of application code through a *code specializer* provided some feedback given by the runtime system at the deployment time, etc.

2.2. Meeting the Needs of Programming Environments and Applications

Keywords: *SMT, distributed, environment, heterogeneity, parallel, runtime.*

Beside those main research topics, we intend to work in collaboration with other research teams in order to *validate* our achievements (e.g. implementing the PaStiX solver on top of μPM^2), to *benefit* from external skills (e.g. use of program analyzers/specializers developed within the Compose project), to better *understand* the specific requirements of complex environments (e.g. common development of PadicoTM and μPM^2 within the framework of the RMI project from the ACI Grid) and to *combine* research efforts to solve difficult problems (e.g. study of the introduction of quality of service schemes within thread scheduling, with the future INRIA Action Mescal [formerly known as Apache]).

Among the target environments, we intend to carry on developing of the successor to the PM^2 environment, which would be a kind of technological showcase to validate our new concepts on real applications through both academic and industrial collaborations (ScAlApplix, CEA/DAM). We also plan to port standard environments and libraries (which might be a slightly sub-optimal way of using our platform) by proposing extensions (as we already did for MPI and Pthreads) in order to ensure a much wider spreading of our work and thus to get more important feedback.

Finally, most of the work proposed as part of this project is dedicated to be used as a foundation for environments and programming tools exploiting large scale computing grids. While these environments must address many issues related to long distance links properties and decentralized administration (authentication, security, deployment), they must also rely on efficient runtime systems on the "*border clusters*" in order to convert optimally the local area resources potential into application performance. We already have ongoing collaborations about this particular topic and we are currently getting in touch with other teams (CoC GRID in Germany, RNTL project with INRIA Apache project).

3. Scientific Foundations

3.1. Runtime Systems Evolution

Keywords: *cluster, communication, distributed, environment, library, multithreading, parallel.*

Nowadays, when intending to implement complex parallel programming environments, the use of runtime systems is unavoidable. For instance, parallel languages compilers generate code which is getting more and more complex and which relies on advanced runtime system features (e.g. the HPF Adaptor compiler [33], the Java bytecode Hyperion compiler [1]). They do so not only for portability purposes or for the simplicity

of the generated code, but also because some complex handling can be performed only at runtime (garbage collection, dynamic load balancing).

Parallel runtime systems have long mostly consisted of an elaborate software glue between standard libraries implementations, such as, for instance, MPI [26] for communication handling and POSIX-threads [46] for multi-threading management. Environments such as Athapascan [34], Chant [44] or PM^2 [45] well illustrate this trend. Even though such approaches are still widespread, they do suffer from numerous limitations related to *functional* incompatibilities between the various software components (decreased performance) and even to *implementation* incompatibilities (e.g. thread-unsafe libraries).

Several proposals (Nexus [39], Panda [49], PM^2 [45]) have shown that a better approach lies in the design of runtime systems that provide a tight integration of communication handling, I/O and multi-threading management. In order to get closer to an optimal solution, those runtime systems often exploit very low-level libraries (e.g. BIP [48], GM [25], FM [47] or LFC [32] for Myrinet networks) so as to control the hardware finely. It is one of the reasons that makes the design of such systems so difficult.

Many custom runtime systems have thus been designed to meet the needs of specific environments (e.g Athapascan-0 [36], [43] for the Athapascan-1 [34] environment, Panda [49] for the Orca [30] compiler, PM [51] for the SCore environment, PM^2 [45] for load balancing tools using thread migration). Somehow, because they were often intended for very similar architectures, these proposals also resulted in duplicating programming efforts.

Several studies have therefore been launched as an attempt to define some kinds of "*micro*-runtimes" (just like *micro*-kernels in the field of operating systems) that would provide a minimal set of generic services onto which a wide panel of higher-level runtime systems could be built. An example of such a micro-runtime system is μPM^2 [11]. μPM^2 integrates communication handling and multi-threading management without imposing a specific execution model. Such research approaches indeed allowed for a much better reuse of runtime systems within different programming environments. The μPM^2 platform has, for instance, been successfully used as a basis for implementing a distributed Java virtual machine [1], a Corba object broker [41], a high-performance communication framework for grids (PadicoTM [38]) and even a multi-network version of the MPICH [8], [7] library.

3.2. Current Trends

Keywords: *cluster*, *communication*, *distributed*, *environment*, *library*, *multithreading*, *parallel*.

Even though several problems still remain unresolved so far (communication schemes optimization, reactivity to I/O events), we now have at our disposal efficient runtime systems that *do* efficiently exploit small-scale homogeneous clusters. However, the problem of mastering large-scale, hierarchical and potentially heterogeneous configurations (that is, clusters of clusters) still has to be tackled. Such configurations bring in many new problems, such as high-performance message routing in a heterogeneous context, dynamic configuration management (fault-tolerance). There are two interesting proposals in the particular case of heterogeneous clusters of clusters, namely MPICH-G2 [27] and PACX-MPI [31]. Both proposals attempt to build virtual point-to-point connections between each pair of nodes. However, those efforts focus on very large-scale configurations (the TCP/IP protocol is used for inter-cluster communication as clusters are supposed to be geographically distant) and are thus unsuitable for exploiting configurations featuring high-speed inter-cluster links. The CoC-Grid Project [24] follows an approach similar to ours through trying to provide an efficient runtime system for such architectures. A preliminary contact has already been established in order to set up a collaboration about this topic.

Besides, even if the few aforementioned *success stories* demonstrate that current runtime systems actually improve both portability and performance of parallel environments, a lot of progress still has to be made with regards to the optimal use of runtime systems features by the higher level software layers. Those upper layers still tend to use them as mere "black-boxes". More precisely, we think that the expertise accumulated by a runtime system designer should be formalized and then transferred to the upper layers in a systematic fashion

(code analysis, specialization). To our knowledge, no such work exists in the field of parallel runtime systems to date.

The members of the RUNTIME project have an acknowledged expertise in the parallelization of complex applications on distributed architectures (combinatorial optimization, 3D rendering, molecular dynamics), the design and implementation of high performance programming environments and runtime systems (PM2), the design of communication libraries for high speed networks (Madeleine) and the design of high performance thread schedulers (Marcel, LinuxActivations).

During the last few years, we focused our efforts on the design of runtime systems for clusters of SMP nodes interconnected by high-performance networks (Myrinet, SCI, Giganet, etc). Our goal was to provide a low-level software layer to be used as a target for high-level distributed multithreaded environments (e.g. PM^2, Athapascan). A key challenge was to allow the upper software layers to achieve the full performance delivered by the hardware (low latency and high bandwidth). To obtain such a "performance portability" property on a wide range of network hardware and interfaces, we showed that it is mandatory to elaborate alternative solutions to the classical interaction schemes between programming environments and runtime systems. We thus proposed a communication interface based on the association of "transmission constraints" with the data to be exchanged and showed data transfers were indeed optimized on top of any underlying networking technology. It is clear that more research efforts will have to be made on this topic.

Another aspect of our work was to demonstrate the necessity of carefully studying the interactions between the various components of a runtime system (multiprogramming, memory management, communication handling, I/O events handling, etc.) in order to ensure an optimal behavior of the whole system. We particularly explored the complex interactions between thread scheduling and communication handling. We hence better understood how the addition of new functionalities within the scheduler could improve communication handling. In particular, we focused our study on the impact of the thread scheduler reactivity to I/O events. Some research efforts conducted by the group of Henri BAL (VU, The Netherlands), for instance, have led to the same conclusion.

Regarding multithreading, our research efforts have mainly focused on designing a multi-level "chameleon" thread scheduler (its implementation is optimized at compilation time and tailored to the underlying target architecture) and on addressing the complexity of efficiently scheduling threads on hierarchical machines like SMPs of multicore chips and NUMA machines.

Although it was originally designed to support programming environments dedicated to parallel computing (PM^2, MPI, etc.), our software is currently successfully used in the implementation of middleware such as object brokers (OmniORB, INRIA Paris project) or Java Virtual Machines (Projet Hyperion, UNH, USA). Active partnerships with other research projects made us realize that despite their different natures these environments actually share a large number of requirements with parallel programming environments as far as efficiency is concerned (especially with regard to critical operations such as multiprogramming or communication handling). An important research effort should hence be carried out to define a reference runtime system meeting a large subset of these requirements. This work is expected to have an important impact on the software development for parallel architectures.

The research project we propose is thus a logical continuation of the work we carried out over the last few years, focusing on the following directions: the quest for the best trade-off between portability and efficiency, the careful study of interactions between various software components, the use of realistic performance evaluations and the validation of our techniques on real applications.

4. Application Domains

4.1. Panorama

Keywords: *CLUMP, SMP, cluster, communication, grid, multithreading, network, performance.*

This research project takes place within the context of high-performance computing. It seeks to contribute to the design and implementation of parallel runtime systems that shall serve as a basis for the implementation of high-level parallel middleware. Today, the implementation of such software (programming environments, numerical libraries, parallel language compilers, parallel virtual machines, etc.) has become so complex that the use of portable, low-level runtime systems is unavoidable.

The last fifteen years have shown a dramatic transformation of parallel computing architectures. The expensive supercomputers built out of proprietary hardware have gradually been superseded by low-cost Clusters Of Workstations (COWs) made of commodity hardware. Thanks to their excellent performance/cost ratio and their unmatched scalability and flexibility, clusters of workstations have eventually established themselves as the today's *de-facto* standard platforms for parallel computing.

This quest for cost-effective solutions gave rise to a much wider diffusion of parallel computing architectures, illustrated by the large and steadily growing number of academic and industrial laboratories now equipped with clusters, in France (200 PCs cluster at the INRIA Rhônes-Alpes Research Unit, extension of the Alpha 512 nodes cluster (four processors per node) at CEA/DAM, Grid5000 Project, etc.), in Europe (cluster DAS-2 in the Netherlands, etc.) or in the rest of the world (the US TeraGrid Project, etc.). As a general rule, these clusters are built out of a homogeneous set of PCs interconnected with a fast system area network (SAN). Such SAN solutions (Myrinet, SCI, Giga-Ethernet, etc.) typically provide Gb/s throughput and a few microseconds latency. Commonly found computing node characteristics range from off-the-shelf PCs to high-end symmetrical multiprocessor machines (SMP) with a large amount of memory accessed through high-performance chipsets with multiple I/O buses or switches.

This increasing worldwide expansion of parallel architectures is actually driven by the ever growing need for computing power needed by numerous real-life applications. These demanding applications need to handle large amounts of data (e.g. ADN sequences matching), to provide more refined solutions (e.g. analysis and iterative solving algorithms), or to improve both aspects (e.g. simulation algorithms in physics, chemistry, mechanics, economics, weather forecasting and many other fields). Indeed, the only way to obtain a greater computing power without waiting for the next generation(s) of processors is to increase the number of computing units. As a result, the cluster computing architectures which first used to aggregate a few units quickly tended to grow to hundreds and now thousands of units. Yet, we lack the software and tools that could allow us to exploit these architectures both efficiently and in a portable manner. Consequently, large clusters do not feature to date a suitable software support to really exploit their potential as of today. The combination of several factors led in this uncomfortable situation.

First of all, each cluster is almost unique in the world regarding its processor/network combination. This simple fact makes it very difficult to design a runtime system that achieves both portability and efficiency on a wide range of clusters. Moreover, few software are actually able to keep up with the technological evolution; the others involve a huge amount of work to adapt the code due to an unsuitable internal design. We showed in [3] that the problem is actually much deeper than a mere matter of implementation optimization. It is mandatory to rethink the existing *interfaces* from a higher, semantic point of view. The general idea is that the interface should be designed to let the application "express its requirements". This set of requirements can then be mapped efficiently by the runtime system onto the underlying hardware according to its characteristics. This way the runtime system can guaranty *performance portability*. The design of such a runtime system interface should therefore begin with a thorough analysis of target applications' *specific* requirements.

Moreover, and beside semantic constraints, runtime systems should also address an increasing number of functional needs, such as fault tolerant behavior or dynamically resizable computing sessions. In addition, more specific needs should also be taken into account, for example the need for multiple independent logical communication channels in modular applications or multi-paradigm environments (e.g. PadicoTM [37]).

Finally, the special case of the CLUsters of MultiProcessors (CLUMPS) introduces some additional issues in the process of designing runtime systems for distributed architectures. Indeed, the classical execution models are not suitable because they are not able to take into account the inherent hierarchical structure of CLUMPS. For example, it was once proposed to simply expand the implementation of standard communication libraries such as MPI in order to optimize inter-processor communication within the same node (MPI/CLUMPS [42]).

Several studies have shown since then that complex execution models such as those integrating multi-threading and communication (e.g. Nexus [40], [39], Athapascan [34], PM2 [45], MPI+OpenMP [35]), are in fact much more efficient.

This last issue about clusters of SMP is in fact a consequence of the current evolution of high-end distributed configurations towards more hierarchical architectures. Other similar issues are expected to arise in the future.

- The clusters hierarchical structure *depth* is increasing. The nodes themselves may indeed exhibit a hierarchical structure: because the overall memory access delay may differ (e.g. according to the proximity of the processor to the memory bank on a Non Uniform Memory Architecture) or because the computational resources are not symmetrical (e.g. multi-processors featuring the *Simultaneous Multi-Threading* technology). The challenge here is to express those characteristics as part of the execution model provided by the runtime system without compromising applications portability and efficiency on "regular" clusters.
- The widespread availability of clusters in laboratories combined with the general need for processing power usually leads to interconnect two or more clusters by a fast link to build a *cluster of clusters*. Obviously, it is likely that these interconnected clusters will be different with respect to their processor/network pair. Consequently, the interconnected clusters should *not* be considered as *merged* into one big cluster. Therefore, and beside a larger aggregated computing potential, this operation results in the addition of another level in the cluster hierarchy.
- A current approach tends to increase the number of nodes that make up the clusters (the CEA/DAM, for instance, owns a cluster of 640 4-processors nodes linked with a Quadrics network). These large clusters give rise to a set of new issues to be addressed by runtime systems. For instance, lots of low-level communication libraries do not allow a user to establish point-to-point connections between the whole set of nodes of a given configuration when the number of nodes grows beyond several dozens. It should be emphasized that this limitation is often due to physical factors of network interconnection cards (NICs), such as on-board memory amount, etc. Therefore, communication systems bypassing the constraint of a node being able to perform efficient communications only within a small neighbourhood have to be designed and implemented.
- Finally, each new communication technology brings its own new programming model. Typically, programming over a memory-mapped network such as SCI is completely different from programming over a message passing oriented network such as Myrinet. Similar observations can be made about I/O (the forthcoming Infiniband technology is likely to bring in new issues), processors and other peripheral technology. Runtime systems should consequently be openly designed from the very beginning not only to deal with such a constantly evolving set of technologies but also to be able to integrate easily and to exploit thoroughly existing as well as forthcoming *idioms*.

In this context, our research project proposal aims at designing *a new generation of runtime systems* able to provide parallel environments with most of the available processing power of cluster-like architectures. While many teams are currently working the exploitation of widely distributed architectures (grid computing) such as clusters interconnected by wide-area networks, we propose, as a complementary approach, to conduct researches dedicated to the design of high-performance runtime systems to be used as a solid foundation for high level programming environments for large parallel applications.

5. Software

5.1. Madeleine

The Madeleine library is the communication subsystem of the PM^2 software suite. This communication library is principally dedicated to the exploitation of clusters interconnected with high-speed networks, potentially of different natures. Madeleine is a *multithreaded* library both in its conception (use of lightweight processes to implement some functionalities) and in its use: Madeleine's code re-entrance enables it to be used jointly with the Marcel library. Moreover, Madeleine is a *multi-cluster* communication library that implements a concept of communication *channel* that can be either physical (that is, an abstraction of a physical network) or virtual. In that latter case, it becomes possible to build virtual heterogeneous networks. Madeleine features a message forwarding mechanism that relies on gateways when permitted by the configuration (that is, when several different networking technologies are present on the same node). Madeleine is also able to dynamically select the most appropriate means to send data according to the underlying technology (*multi-paradigms*). This is possible by specifying constraints on data to be sent ("design by contract" concept) and provides a good performance level above technologies possibly relying on very different paradigms. Madeleine relies on external software regarding deployment, session management (the *Léonie* software), or exploitation of user-given information (configuration files). Madeleine is available on various networking technologies: Myrinet, SCI, Ethernet or VIA and runs on many architectures: Linux/IA32, Linux/IA64, Linux/Alpha, Linux/Sparc, Linux/PowerPC, Solaris/Sparc, Solaris/IA32, AIX/PowerPC, WindowsNT/IA32. Madeleine and its external software roughly consists of 55000 lines of code and 116 files. This library, available within the PM^2 software is developed and maintained by Olivier AUMAGE and Raymond NAMYST.

5.2. Marcel

Marcel is the thread library of the PM^2 software suite. Marcel threads are user-level threads, which ensures a great efficiency and flexibility. Marcel exists in different *flavors* according to the platform and the needs. In order to take advantage of SMP machines, Marcel is able to use a two-level scheduler based on system kernel threads. It may also tackle the hierarchical caracteristics of nowadays super-computers (NUMA of multi-core chips for instance). The application describes affinities between the threads it launches by encapsulating them into nested bubbles, those which work on the same data for instance. Marcel then automatically distributes bubbles (and hence threads) on the hierarchy of the computer so as to beneficiate from cache effects and avoid NUMA factor penalties as much as possible. Marcel is able to perform low latency synchronisations and communications between threads using scheduler-directed polling tasks. With Linux, Marcel is also able to use activation mechanisms (see LinuxActivations) that allow to bypass classical limitations of user-level thread libraries, that is, blocking system calls. All these *flavors* are based on the same thread management core kernel and are specialized at compilation time.

While keeping the possibility to be run autonomously, Marcel combines perfectly with Madeleine and brings several mechanisms improving reactivity to communications. Specific softwares matching the needs of PM^2 are also included, allowing thread migration between homogeneous machines.

Vincent DANJEAN and Samuel THIBAULT are the main contributors to this piece of software.

5.3. LinuxActivations

LinuxActivations is a piece of software that can be used with the Marcel thread library. It is an extension of the Linux kernel allowing an efficient control of the user-level threads scheduling when they perform blocking I/O operations in the kernel. LinuxActivations is based on an extension of the *Scheduler Activations* model proposed by Anderson [28]. *Upcalls*, the opposite of system calls, are used to notify the user-level scheduler about the events triggered by the kernel. Usually, when a user-level thread within a process performs a blocking system call, the whole process is suspended. With the *upcalls* mechanism, it becomes possible to suspend only the thread responsible for the blocking call, the other threads continuing their execution. Our contribution is

to minimize the elapsed time between the detection of an I/O event in the kernel and the scheduling of the corresponding user-level thread in the application. The processing of I/O events still occur within the kernel but the decisions concerning the scheduling are now handled at the user level.

This extension is available as a Linux kernel patch, at the following URL: http://dept-info.labri.fr/~danjean/linux-activations.html. Vincent DANJEAN is the main contributor to this piece of software.

5.4. FxT : Performance analysis of multithreaded programs

FxT is a library and associated tools that can be used to analyze the performance of multithreaded programs which use a hybrid thread scheduler (i.e. a user-level scheduler on top of a kernel-level one). The Marcel thread library can take full profit from this library.

FxT is based on the offline analysis of traces (sequence of events recorded at run time). The idea is to collect simultaneously two independent traces: one within the kernel and the other in user space. Both traces are sequences of records stamped using the processor's timing registers (incremented at each clock tick). We used the *Fast Kernel Traces* (FKT [50]) library developped by Robert RUSSELL (UNH, USA) for Linux to generate kernel traces. We followed a similar design for our *Fast User Traces* library that operates in user space.

The key point is that both traces are generated very efficiently, mainly because we only record the information which is directly available at the given level. For instance, we do not try to associate the "current processor ID" to user-level events because this would require a system call each time an event is generated. The extra code inserted to generate events only contains a few assembly instructions and uses fast hardware locking instructions. In fact, the set of events collected in user space is complementary to the one collected in kernel space. Moreover, all the events are stamped with the same hardware clock. Thus, the user trace and the kernel trace can easily be merged offline into a *super-trace* in which the missing information has been computed for each event (processor ID, kernel thread ID, user thread ID, etc.)

A small library is provided in order to help the developer to record all kinds of events. We are also able to use the compiler support to records events for each entry and exit of `C` functions without requiring any modification of the source code: event probes are automatically inserted by the compiler and function's name are automatically retrieved from the symbol table of the program.

Once kernel and user traces have been collected and merged, our super-traces can be translated into the Pajé [52] format, so that all events will be graphically represented. Furthermore, Pajé has very interesting features that allows us to display the trace at any scale, to select some kinds of events to show/hide or to change the way the events are represented. This simplifies the analysis of the behavior and performance of hybrid multithreaded programs.

This software is available at the following URL: https://savannah.nongnu.org/projects/fkt/ . There are library and tools' sources and a patch for the linux kernel.

Samuel THIBAULT and Vincent DANJEAN are the main contributors to this piece of software.

5.5. MPICH-Madeleine

MPICH-Madeleine is a high-performance implementation of the MPI (*Message Passing Interface*) standard and targets hardware configurations that implies the need for multiprotocol capabilities: homogeneous SMP clusters or heterogenous clusters of clusters. It is based on a multithreaded progression engine that allows communications to progress independently from the computation. Precisely, calls to MPI routines are not required to enforce communication's progress. A side-effect of this multithreaded architecture is that our MPI implementation supports the highest level of multithreading for MPI applications, that is, `MPI_THREAD_MULTIPLE`. MPICH-Madeleine is therefore based on the Marcel user-level thread library and relies on its optimized polling mechanisms to guaranty a high level of reactivity.

Regarding low-level data transfers, MPICH-Madeleine utilizes the Madeleine generic communication library that provides us with a common limited interface above all low-level network protocols. The drawback

that arises is the impossibility to finely optimize the upper levels of the MPI implementation accordingly to the underlying low-level protocols available but thanks to the carefully designed Madeleine interface and a tight integration, the performance level achieved is similar or even better than that of highly specialized MPI implementations dedicated to a specific high-speed network.

MPICH-Madeleine is based on the popular MPICH implementation (currently MPICH1 1.2.5) but the communication engine concept is implementation-independent and could be adapted to other free implementations, such as YAMPII or Open MPI.

The software is freely available at the following URL: http://runtime.futurs.inria.fr/mpi/. MPICH-Madeleine is developped, updated and maintained by Guillaume MERCIER and Nathalie FURMENTO.

5.6. PadicoTM

PadicoTM is a high-performance communication framework for grids. It is designed to enable various middleware systems (such as CORBA, MPI, SOAP, JVM, DSM, etc.) to utilize the networking technologies found on grids. PadicoTM aims at decoupling middleware systems from the various networking ressources to reach transparent portability and flexibility: the various middleware systems use PadicoTM through a seamless virtualization of networking resources; only PadicoTM itself uses directly the networks.

PadicoTM follows a three-layer approach. The lowest layer, called the *arbitration layer*, aims at making the access to the resources cooperative rather than competitive. It enables the use of multiple middleware systems atop a single network, as needed by code coupling programming models such as parallel objects or parallel components. This layer is based on MARCEL and MADELEINE to ensure high performance. The middle layer, called the *abstraction layer*, decouples the paradigm of the programming interface from the paradigm of the network; for example, it can do dynamic client/server connections over static SPMD networks (à la MPI). The highest level layer, called the *personality layer*, gives several API called "personalities" over the abstractions. It aims at providing the middleware systems with the API they expect. It enables PadicoTM to seamlessly integrate *unmodified* middleware systems.

PadicoTM currently supports most high performance networks (Myrinet, SCI, Quadrics, etc.), communication methods for grids (plain TCP, splicing to cross firewalls, routing, tunneling). Various middleware systems are supported over PadicoTM: various CORBA implementations (omniORB, Mico), popular MPI implementations (MPICH from Argonne – actually, MPICH/PadicoTM is derived from MPICH-Madeleine —, YAMPII from the University of Tokyo), Apache Portable Runtime, JXTA from Sun (in collaboration with the PARIS project), gSOAP, Mome (DSM developped in the PARIS project), Kaffe (Java virtual machine), and Certi (HLA implementation from the ONERA).

PadicoTM was started in the PARIS project (Rennes) in 2001, in collaboration with Christian PÉREZ and migrated in RUNTIME in october 2004 together with Alexandre DENIS. The current main contributors to PadicoTM are Alexandre DENIS and Christophe FRÉZIER (RUNTIME) with some occasional contribution from Christian PÉREZ and Mathieu JAN (PARIS).

PadicoTM is composed of roughly 50 000 lines of C. It is free software distributed under the terms of the GNU General Public License, and is available for download at: http://runtime.futurs.inria.fr/PadicoTM/. It is has been hosted on InriaGForge since mid-2005 and has been downloaded 167 times (ranked 11 out of 30) since then. PadicoTM is registered at the APP under number IDDN.FR.001.260013.000.S.P.2002.000.10000.

As far as we are aware of, it is currently used by several French projects: ACI GRID HydroGrid, ACI GRID EPSN, RNTL VTHD++ and Inria ARC RedGrid. It is also used in the European FET project POP. It will be used in the ARA LEGO.

6. New Results

6.1. A Fast implementation of MPI on clusters of clusters

MPICH-Madeleine [22], our multi-protocol implementation of the MPI standard, has been improved in several ways.

First, we designed some optimization mechanisms aiming to decrease the communication time of derived datatypes for which data is stored in noncontiguous memory areas. These mechanisms have been successfully implemented and tested over our local cluster[23].

Then, we have validated the implementation of MPICH-Madeleine over Grid'5000, a platform developping a large scale nation wide infrastructure for Grid research. These validations allowed to fully tests the network functionalities of MPICH-Madeleine.

Also,we are currently working closely with the research group that is developping the MPICH software at Argonne National Laboratory in order to integrate the mechanisms created for MPICH-Madeleine into their MPICH2 software. Guillaume MERCIER is currently holding an 18-months post-doctoral position at the ANL (until August 2006).

6.2. Generic Thread Scheduling

In order to adapt our thread library MARCEL to the new hierarchical architectures (HyperThreading technology, ccNUMA), several improvements have been achieved:

- MARCEL specializes itself with respect to the underlying architecture. On monoprocessor machine, MARCEL is a very efficient pure user-level thread library. On multiprocessors, MARCEL becomes a two-level thread library so that it can exploit all the processors of the machine. If the architecture is hierarchical (SMT, ccNUMA), MARCEL allows the application to take care of this situation and to group its threads with respect to the hierarchical architecture.
- MARCEL specializes itself with respect to the underlying software. When MARCEL needs several execution flows to exploits multiprocessors machines, it choose the best method available on the target system. For example, it can use the POSIX threads on any standard Unix, but it can also use the `clone` system call on Linux systems or even the Scheduler Activations if this mechanism is available.
- MARCEL's low level mechanisms (spinlocks, atomic integers, ...) now have a generic implementation using standard `pthread` functions. This permits porting MARCEL on many architectures really quickly and easily, while still achieving quite good performance. It was tested on Opteron, Alpha, Sparc and PowerPC processors.
- MARCEL specializes itself with respect to applications' needs. Features such as traces or signals handling are included if and only if the application requires it.

Furthermore, *the thread scheduler is now able to use application directives* (regarding memory affinity between threads) to better schedule threads on hiearchical architectures (e.g. NUMA machines). It is based on a powerful *bubble mechanism* [21], [20] that allows the programmer to build hierarchical affinity sets between threads with no dependency upon the underlying architecture. The scheduler can use this information to maximize the locality of threads belonging to the same bubble while still trying to keep all the processors busy. This mechanism is generic enough for developping a wide range of schedulers, hence letting programmers try various approaches for distributing bubbles on the machine: simple top-bottom distribution, gang scheduling, work stealing, etc.

A trace analysis tool has been developped for providing off-line view of the scheduling decisions. Programmers can hence review at will how their applications got scheduled. This lets them easily discover misbehaviours so as to try and tune various scheduling approaches.

6.3. Distributed Communication Scheme Optimization

Building on our last year results on the subject of dynamic communication optimizations, we have undertaken a complete redesign of the Madeleine communication library internals. Until now, the Madeleine communication optimization engine only supported *deterministic* optimization operations, that is operations that could be decided out of informations available *both* at the send and the receive side. The new design removes this limitation and opens the engine to a large potential of new optimizations. In particular, it enables the use of opportunist optimizations where the send-side engine may decide on-the-fly to take special benefit of favourable situations, should such favourable situations be encountered. For instance, upon being about to send a packet of a particular communication flow, the send-side engine may *unilateraly* decide to aggregate this packet with another packet of another communication flow sharing the same destination *if* such a packet happens to be available, and if such an aggregation appears to be a valuable optimization.

The new optimization engine is designed in an open way so that new optimization schemes can be easily implemented to facilitate experiments. Optimization schemes are called “strategies” and may be conceptually expressed in terms of sequences of basic optimization operations called “tactics”. Examples of tactics include packets aggregation, packets association (for dynamic aggregation through hardware gather/scatter support), packet splitting, permutation, etc. The optimization engine actually now acts as a kind of packet scheduler.

Beside introducing packet scheduling, the new design also introduces a new approach for communication request processing. Communication requests used to be processed only during application calls to the library API commands. This could lead to a sub-optimal use of the networking hardware which might become idle in-between two application calls to the API. In the new design, the processing of communication request has been made asynchronous and is now controlled by the network interface card(s) activity. The API send commands now simply insert requests into a request list. Then, each time a card is about to become idle, the optimization engine is queried for the next packet to feed to the card, therefore ensuring that it is always kept busy as long as some requests are still pending.

A first prototype of this new design called Madeleine 4 as been implemented as part of the Master’s internship of Elisabeth Brunet, who is now doing a Ph.D in the team. It has been ported to the new MX high-performance communication interface for Myricom’s Myrinet networks.

6.4. High-Performance In-Kernel Communication

While most contributions in the high-performance communication community in the last fifteen yeards have been focused on providing fast communication to user-level applications using kernel-bypass direct access from userspace to the communication hardware, there exists some situations where having high-performance communication at hand inside the kernel is desirable. This is obviously the case when the application code using communication is located inside the kernel itself. In that case, using regular userspace communication libraries would require expensive transitions from kernel space to user space for each communication request.

Examples of in-kernel applications that would benefit from high-performance communication support include distributed storages such as network-attached disks. Network-attached disks differ from conventional hard disks by being directly plugged onto a regular communication network instead of being screwed inside a computer housing and plugged to a motherboard through a specific wiring. Such a change of being able to move hard disks outside the computer box is made possible thanks to the recent proposal of communication protocols such as the iSCSI protocol designed through the work of the IP Storage working group at IETF, for instance. The iSCSI protocol enables regular SCSI commands to be carried inside IP packets, therefore turning any IP-capable network into a virtual SCSI wiring.

In collaboration with the team of R. Russell at the InterOperability Laboratory (IOL) from the University of New Hampshire, which is involved in the iSCSI protocol standardisation process at IETF, we designed and implemented a prototype of their iSCSI communication stack on top of a kernel-enabled prototype of the Madeleine communication library called Kmadeleine, in the context of the Master’s internship of Jonathan Pagès partly in Bordeaux and partly in R.Russell’s team at IOL. We were able to run the iSCSI+Kmadeleine

software on top of TCP as well as on top of a MX/Myrinet high-performance network, and in the later case to have iSCSI benefit from the extra low latency and high bandwidth of MX/Myrinet.

6.5. FxT: Performance analysis of multithreaded programs on NUMA machines

In the context of a research contract between the INRIA RUNTIME team and the CEA/DAM (French Atomic Energy Commission), we improved our FxT software so that it fulfills the requirements of the CEA environment: it must work on Itanium NUMA machines without modifying (patching) the kernel.

This leads to a new internal architecture of FxT, allowing to take care of memory affinity when recording events. This affinity is based on a best effort algorithm: most measures are locally recorded, however on some special conditions, some measures may still be recorded in a non-local memory. Avoiding these rare bad cases would be far too much intrusive and costly.

With the requirement of not patching the kernel, it becomes difficult to track kernel-level context switch. We choose to take profit from the fact that the Marcel thread library binds one and only one kernel thread per physical processor and that NUMA nodes on CEA machines were dedicated to the running program (ie no other program runs on the machine but system daemons). Under these assumptions, FxT has been modified to be able to reconstruct the global scheduling of user threads on physical processors, even if some external events are now ignored. If a system daemon wakes up, it will be unnoticed by our tracing mechanism.

All these improvements and the adaptation of the low routines to the IA64 assembly allow us to observe the behaviour of multithreaded programs running on unmodified NUMA nodes at the CEA.

6.6. Flexible network communications on computational grids

Computational grids are defined as a large scale interconnection of computing resources — clusters of workstations or parallel supercomputer — on multiple sites. Therefore, the networking resources involved are very heterogeneous, ranging from high performance interconnection networks inside parallel computers to wide are networks between sites. The technologies of these networks are different, so are the protocols, the software stacks and the performance; even the middleware systems available differ from one network type to another — typically CORBA is available only over TCP/IP, only MPI is available over high performance networks.

PadicoTM is a communication framework that decouples middleware systems from the actual networking resource. The applications are thus able to transparently and efficiently utilize any kind of middleware (either parallel or distributed) on any network.

PadicoTM began as a *modular* framework, each module being in charge of a given paradigm adaptation. The year 2005 has been devoted to a transition towards true *software components*. The main difference is that assembly of software components are composed by a third party — the framework, not the component itself — whereas modules may use only other modules known in advance. Therefore, any combination that the user wants is doable with components; in particular, components may be designed separately and assembled later, whereas in a modular approach, the possibilities of assemblies are hardwired in the code.

The transition towards software components brought a significant improvement of flexibility and easiness to add new communication methods. We were able to add various communication methods as new components, mainly communication methods for wide area networks: various compression filters (ZIP, LZO, BZIP) and flexible socket factories to cross firewalls without compromising security (SSH tunnel, TCP splicing, relaying with dynamic routing). To take benefit of the component baseness of PadicoTM, we have shown that a *control channel* used for bootstrap and out-of-band communication is required. We proposed a novel approach for the management of this control channel that combine security, connectivity, and high performance.

For the first time in 2005, PadicoTM and MADELEINE were developed in the same INRIA project. We are beginning work to make MADELEINE and PadicoTM converge (but not merge, since MADELEINE and PadicoTM have different targets). We are currently working on embedding the generic layer of MADELEINE as a communication component for PadicoTM, so as to be able to run MADELEINE directly over PadicoTM.

Finally, we worked on widening the availability of middleware systems over PadicoTM. The Apache Portable Runtime (APR) and JXTA-C (Sun Microsystems) were ported by members of the PARIS project to enable the JuxMem environment to run over PadicoTM; benchmarks were performed with good results and published [29]. Moreover, we ported ICE (the Internet Communication Engine, announced as the successor of CORBA) and YAMPII (open source MPI implementation from the University of Tokyo) over PadicoTM with good results. Finally, we are currently discussing with the Salomé project (CEA/EDF, http://www.salome-platform.org/) to use PadicoTM to solve network-related problems (mostly connectivity and performance) encountered by Salomé when deploying on a network topology made of supercomputers with high performance internal network and restrictive security policy towards the outside, and standalone visualization workstations.

7. Contracts and Grants with Industry

7.1. PhD thesis co-supervised with CEA/DAM

3 years, 2004-2006

We did set up a collaboration with the CEA/DAM (French Atomic Energy Commission, Pierre LECA and Hervé JOURDREN, Bruyère le Chatel) on the support of nuclear simulation programs (adaptive mesh) on large clusters of SMP (thousands of processors) and on Itanium2-based NUMA machines. In September 2003, Marc PERACHE has started a PhD thesis granted by the CEA under the co-supervising of Hervé JOURDREN and Raymond NAMYST. He works on thread scheduling over clusters of SMP.

7.2. Contract between INRIA and CEA/DAM

1 year, 2004-2005

In the context of a research contract between the Runtime team and the CEA/DAM (French Atomic Energy Commission), we are working on the implementation of our hybrid thread scheduler (Marcel) on hierarchical NUMA architectures made of Intel Itanium2 processors. We particularly focus on the compliance with the POSIX Threads standard, as well as on a powerful trace mechanism for performance analysis. Vincent DANJEAN has been hired by CEA/DAM as an engineer for this purpose.

7.3. AESE Competitiveness Cluster

3 years, 2005-2008

The Runtime team is involved in the *Aeronautique, Espace, Systèmes embarqués* competitiveness cluster, labeled “worldwide scope cluster” by the Ministry of Industry, which comprises the major french actors in the field of aeronautics, space and embedded systems. The participation of Runtime is centered around the development of runtime systems designed to speed up various aeronautics simulations.

8. Other Grants and Activities

8.1. “Calcul Intensif et Grilles de Calcul” ANR projects

3 years, 2005-2007

The National Agency for Research (ANR) has launched a program called CIGC about the development of High Performance computing and Grids. In 2005, twelve research proposals have been selected by the national comittee. We participate to three of these projects (granted each a three-years funding):

LEGO Grid infrastructure and middleware is now a mature technology; however, grid programming and use is still a very complex task, because each middleware only take into account one paradigm: MPI, RPC, workflow, master-slave, shared data, ... Thus a new model must be learnt for each kind of application. Current high performance computing application are becoming multi-paradigm. The

aim of LEGO is to propose and to implement a multi-paradigm programming model (component, shared data, master-slave, workflow) comprising state of the art grid programming. It will use efficient scheduling, deployment, and an adequate communication layer. The model will be designed to cope with three kinds of classical high performance computing applications: climate modeling, astronomy simulation, and matrix computation.

NUMASIS Adapting and Optimizing Applicative Performance on NUMA Architectures Design and Implementation with Applications in Seismology. Future generations of multiprocessor machines will rely on a NUMA architecture featuring multiple memory levels as well as nested computing units. To achieve most of the hardware's performance, parallel applications need powerful software to carefully distribute processes and data so as to limit non-local memory accesses. The NUMASIS project aims at evaluating the functionalities provided by current operating systems and middleware in order to point out their limitations. It also aims at designing new methods and mechanisms for an efficient scheduling of processes and a clever data distribution on such platforms. The target application domain is seismology, which is very representative of the needs of computer-intensive scientific applications.

PARA The peak performance improvement of the new microprocessor generation comes from an increase in the degrees and the multiple levels of parallelism: multithread/multicore, multiple and complex vector units. This increase in the number of way to express parallelism leads to reconsider the usual code optimization techniques. The goal of project PARA is to study and develop new optimization methods for an optimal use of the different parallelism levels. Target architectures will be both new generation of generic processors and more specialized systems (GPU, processor Cell, APE). The idea is to combine microbenchmarking techniques (dynamic and detailed analyses of small code kernels) with adaptative code generation (iterative optimizations expressed by metaprograms). Our reference code will come from the numeric simulation field (fluid mechanics, geophysics and QCD) and from cryptology (mainly cryptoanalysis).

8.2. Grid'5000 Ministry Grant

3 years, 2003-2005

The ACI GRID initiative, managed by the Ministry of Research, aims at boosting the involvement of French research teams in Grid research, which requires considerable coordination efforts to bring experts from both computer science and applied mathematics. In 2003, a specific funding as been allocated to set up an experimental National Grid infrastructure, called Grid'5000. It aims at building a 5000 processors Grid infrastructure using ten different sites in France interconnected by the RENATER research network. The Bordeaux site has been selected to become one of these sites (120 kEuros granted from ACI GRID, 300 kEuros from INRIA). Four local research teams are involved in this project. Raymond NAMYST is the local coordinator of Grid'5000. He did also coordinate the writing of the grant request submitted to the Regional Council of Aquitaine. This request has been accepted and the grant amount is 650 kEuros for two years.

8.3. "Masse de données" Ministry Grant

3 years, 2003-2006.

The project is named Data Grid Explorer (led by Franck CAPPELLO, LRI) and aims to build a large testbed in order to emulate Grid/P2P systems. This emulator is based on a large cluster (1K CPU cluster), a database of experimental mesurements and a set of tools for experiments and result analysis. Our goal is to design a runtime system providing measurement tools over a configurable multi-level scheduler and a configurable high performance communication layer.

9. Dissemination

9.1. Schools

Raymond NAMYST has been invited to give a lecture ($12 \times 1h30$) on *Efficient Programming on Parallel Architectures* at the CEA-EDF-INRIA summer school (june 2006) devoted to *Optimizing Scientific Applications on New Generation High Performance Hardware*.

9.2. Committees

Raymond NAMYST was part of the *RenPar 2005 Conference* and *Coset 2005 Workshop* program committees.

9.3. Invitations

Guillaume MERCIER has been appointed as a post-doctoral researcher for 18 months at Argonne National Laboratory (Chicago, USA) in William GROPP and Ewing LUSK's group which leads MPICH2 development. The current MPICH2 low-level architecture has undergone important changes, since a new communication channel, called NEMESIS will replace the other available ADI3 channels. Guillaume MERCIER, along with William GROPP and Darius BUNTINAS, designed and implemented this NEMESIS software. But the ADI3 itself is on the verge to more dramatic changes since the development of a device version of NEMESIS will soon be undertaken. Since our Madeleine 4 communication library and the MPICH2 NEMESIS channel possess common goals and design features, we believe that this is a very good opportunity to share experience about these two pieces of software and to influence altogether the design of MPICH2. To this end, Raymond NAMYST and Olivier AUMAGE, have been invited at the ANL and presented their current works to the designers of NEMESIS. Also, Guillaume MERCIER, as a member of the MPICH2 development team and as a NEMESIS designer, will visit theRUNTIME project for a 6-months term (from March until August 2006). He will work on the NEMESIS/Madeleine 4 convergence, and also will take advantage of our group's expertise in the high-performance networks department in order to develop new networks modules for NEMESIS: both Quadrics and Infiniband are concerned but Madeleine is the eventual target. Guillaume will also use the extensive knowledge of the team with regard to multithreading in order to better support the thread-dependant features in the current MPICH2 implementation.

Raymond NAMYST has been invited to give a talk at the HPC international Workshop (High Performance Computing) organized by Bull and CEA on "Designing high performance runtime systems for clusters of multiprocessors".

9.4. Reviews

Olivier AUMAGE was involved in the paper reviewing process of the Transaction on Computer Systems IEEE journal, the CCGrid 2005 conference and the Heterogeneous Computing Workshop 2006.

Alexandre DENIS was involved in the paper reviewing process of the International Symposiom High-Performance Distributed Computing (HPDC 2005), the International Symposium on Parallel and Distributed Computing (ISPDC 2005), the SuperComputing Conference (SC|05), the International Parallel & Distributed Processing Symposium (IPDPS 2006) and the ICC'06 General Symposium.

Raymond NAMYST has reviewed 3 PhD Thesis during year 2005.

9.5. Seminars

Olivier AUMAGE gave a seminar about the Madeleine High-Performance Communication Library at the University of New Hampshire and at Argonne National Laboratory (Jul. 2005).

Olivier AUMAGE and Raymond NAMYST did participate to the "Future Generation Grids" seminar in Dagstuhl (Nov. 2004) and to a collaboration seminar at CEA-DAM near Paris (Feb. 2005).

Alexandre DENIS gave a seminar on “Network communications in Grids” to visitors from the *Salomé* project (CEA-EDF) at LaBRI (Dec. 2005).

Raymond NAMYST gave a seminar about “Efficient Thread Scheduling over hierarchical multiprocessor machines” at the Argonne National Laboratory (Jul. 2005).

9.6. Expertise

Alexandre DENIS was involved as an expert in the ARA MDMSA (Masse de données: Modélisation, Simulation, Applications) project selection process for the ANR 2005 call.

Raymond NAMYST was involved as an expert in the ARA CIGC (Calcul Intensif et Grilles de Calcul) project selection process (2005 campaign).

9.7. Teaching

Olivier AUMAGE gave a course on “Network Architecture and Related Systems” in the Master of Science at the University Bordeaux 1. He gave a course about “High-Performance Communication Supports” and a course on “Programming Languages for Parallelism” at the ENSEIRB engineering school.

Alexandre DENIS gave a course on “System and Middleware for Parallel and Distributed Computing” in the Master of Science at the University of Bordeaux 1.

Raymond NAMYST holds a professor position at the University Bordeaux 1 and gave several courses related to operating systems and networks. He also gave a course on “Fast Network Protocols” at the ENSEIRB engineering school.

10. Bibliography

Major publications by the team in recent years

[1] G. ANTONIU, L. BOUGÉ, P. HATCHER, M. MACBETH, K. MCGUIGAN, R. NAMYST. *The Hyperion system: Compiling multithreaded Java bytecode for distributed execution*, in "Parallel Computing", vol. 27, October 2001, p. 1279–1297, http://www.irisa.fr/paris/Biblio/Papers/Antoniu/AntBouHatBetGuiNam01ParCo.ps.gz.

[2] O. AUMAGE. *Madeleine : une interface de communication performante et portable pour exploiter les interconnexions hétérogènes de grappes.*, 154 pages, Thèse de Doctorat, spécialité informatique, École normale supérieure de Lyon, 46, allée d’Italie, 69364 Lyon cedex 07, France, September 2002, http://runtime.futurs.inria.fr/Download/Publis/Aum02These.ps.gz.

[3] O. AUMAGE, L. BOUGÉ, A. DENIS, L. EYRAUD, J.-F. MÉHAUT, G. MERCIER, R. NAMYST, L. PRYLLI. *A Portable and Efficient Communication Library for High-Performance Cluster Computing (extended version)*, in "Cluster Computing", vol. 5, n° 1, January 2002, p. 43-54, http://runtime.futurs.inria.fr/Download/Publis/AumBouDenEyrMehMerNamPry01CC.ps.gz.

[4] O. AUMAGE, L. BOUGÉ, L. EYRAUD, R. NAMYST. *Calcul réparti à grande échelle*, I.–U. D. N.–I. FRANÇOISE BAUDE (editor). , ISBN 2-7462-0472-X, chap. Communications efficaces au sein d’une interconnexion hétérogène de grappes : Exemple de mise en oeuvre dans la bibliothèque Madeleine, Hermès Science Paris, 2002.

[5] O. AUMAGE, L. BOUGÉ, J.-F. MÉHAUT, R. NAMYST. *Madeleine II: A Portable and Efficient Communication Library for High-Performance Cluster Computing*, in "Parallel Computing", vol. 28, n° 4, April 2002, p. 607–626, http://runtime.futurs.inria.fr/Download/Publis/clustercomputing2k1.ps.gz.

[6] O. AUMAGE, L. EYRAUD, R. NAMYST. *Efficient Inter-Device Data-Forwarding in the Madeleine Communication Library*, in "Proc. 15th Intl. Parallel and Distributed Processing Symposium, 10th Heterogeneous Computing Workshop (HCW 2001), San Francisco", Extended proceedings in electronic form only, Held in conjunction with IPDPS 2001, April 2001, 86, http://runtime.futurs.inria.fr/Download/Publis/AumEyrNam00HCW2001.ps.gz.

[7] O. AUMAGE, G. MERCIER. *MPICH/MadIII: a Cluster of Clusters Enabled MPI Implementation*, in "Proc. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003), Tokyo", IEEE, May 2003, p. 26–35, http://runtime.futurs.inria.fr/Download/Publis/AumMer03CCGRID.ps.gz.

[8] O. AUMAGE, G. MERCIER, R. NAMYST. *MPICH/Madeleine: a True Multi-Protocol MPI for High-Performance Networks*, in "Proc. 15th International Parallel and Distributed Processing Symposium (IPDPS 2001), San Francisco", Extended proceedings in electronic form only., IEEE, April 2001, 51, http://runtime.futurs.inria.fr/Download/Publis/AumMerNam01IPDPS2001.ps.gz.

[9] L. BOUGÉ, P. HATCHER, R. NAMYST, C. PÉREZ. *A multithreaded runtime environment with thread migration for a HPF data-parallel compiler*, in "The 1998 Intl Conf. on Parallel Architectures and Compilation Techniques (PACT '98), Paris, France", IFIP WG 10.3 and IEEE, October 1998, p. 418-425, ftp://ftp.ens-lyon.fr/pub/LIP/Rapports/RR/RR1998/RR1998-43.ps.Z.

[10] V. DANJEAN, R. NAMYST, R. RUSSELL. *Linux Kernel Activations to Support Multithreading*, in "Proc. 18th IASTED International Conference on Applied Informatics (AI 2000), Innsbruck, Austria", IASTED, February 2000, p. 718-723, http://runtime.futurs.inria.fr/Download/Publis/DanNamRus00IASTED.ps.gz.

[11] R. NAMYST. *Contribution à la conception de supports exécutifs multithreads performants*, Habilitation à diriger des recherches, Université Claude Bernard de Lyon, pour des travaux effectués à l'école normale supérieure de Lyon, December 2001, http://runtime.futurs.inria.fr/Download/Publis/NamystHDR.pdf.

Doctoral dissertations and Habilitation theses

[12] V. DANJEAN. *Contribution à l'élaboration d'ordonnanceurs de processus légers performants et portables pour architectures multiprocesseurs*, 156 pages, Thèse de Doctorat, spécialité informatique, École normale supérieure de Lyon, 46, allée d'Italie, 69364 Lyon cedex 07, France, December 2004, http://tel.ccsd.cnrs.fr/tel-00009541.

[13] G. MERCIER. *Communications à hautes performances portables en environnements hiérarchiques, hétérogènes et dynamiques*, 168 pages, Thèse de Doctorat, spécialité informatique, Université de Bordeaux 1, Domaine Universitaire, 351 Cours de la libération, 33405 Talence Cedex, December 2004, http://tel.ccsd.cnrs.fr/documents/archives0/00/00/90/54/.

Articles in refereed journals and book chapters

[14] V. DANJEAN, P.-A. WACRENIER. *Mécanismes de traces efficaces pour programmes multithreadés*, in "TSI, (Technique et Science Informatiques)", To appear, 2005, http://dept-info.labri.fr/~danjean/publications.html#DanWac05TSI.

[15] A. DENIS, S. LACOUR, C. PÉREZ, T. PRIOL, A. RIBES. *Engineering the Grid: status and perspective*, B. D. MARTINO, J. DONGARRA, A. HOISIE, L. T. YANG, H. ZIMA (editors). , ISBN: 1-58883-038-1, chap.

Programming the Grid with components: models and runtime issues, American Scientific Publishers, 2005, http://www.aspbs.com/grid.html.

[16] C. MORIN, A. DENIS, R. NAMYST, O. AUMAGE, R. LOTTIAUX. *Encyclopédie de l'informatique*, À paraître, chap. Des réseaux de calculateurs aux grilles de calcul, Vuibert, 2005.

Publications in Conferences and Workshops

[17] O. AUMAGE, J. M. BAHI, S. CONTASSOT-VIVIER, R. COUTURIER, A. DENIS, R. NAMYST, G. PAPAURÉ, C. PÉREZ, M. SAUGET. *Alta: Asynchronous Loss Tolerant Algorithms for Grid Computing*, in "3rd International workshop on Parallel Matrix Algorithms and Applications (PMAA'04), Marseille, France", CIRM, October 2004.

[18] F. CAPPELLO, E. CARON, M. DAYDE, F. DESPREZ, E. JEANNOT, Y. JEGOU, S. LANTERI, J. LEDUC, N. MELAB, G. MORNET, R. NAMYST, P. PRIMET, O. RICHARD. *Grid'5000: a large scale, reconfigurable, controlable and monitorable Grid platform*, in "Grid'2005 Workshop, Seattle, USA", To appear, IEEE/ACM, November 13-14 2005.

[19] A. DENIS, O. AUMAGE, R. HOFMAN, K. VERSTOEP, T. KIELMANN, H. BAL. *Wide-Area Communication for Grids: An Integrated Solution to Connectivity, Performance and Security Problems*, in "Proc. of the Thirteenth IEEE International Symposium on High-Performance Distributed Computing (HPDC'13), Honolulu, Hawaii", 10 pages, IEEE, June 2004, http://hal.inria.fr/inria-00000126.

[20] S. THIBAULT. *A Flexible Thread Scheduler for Hierarchical Multiprocessor Machines*, in "Second International Workshop on Operating Systems, Programming Environments and Management Tools for High-Performance Computing on Clusters (COSET-2), Cambridge / USA", ICS / ACM / IRISA, 06 2005, http://hal.inria.fr/inria-00000138/en/.

[21] S. THIBAULT. *Un ordonnanceur flexible pour machines multiprocesseurs hiérarchiques*, in "16ème Rencontres Francophones du Parallélisme 16ème Rencontres Francophones du Parallélisme, Le Croisic / France", ACM/ASF - École des Mines de Nantes, 04 2005, http://hal.inria.fr/inria-00000137/en/.

Internal Reports

[22] N. FURMENTO, G. MERCIER. *MPICH/Madeleine Installer's, User's and Developer's Guide*, Also available as LaBRI Report 1375-05, Technical Report, n° 0316, INRIA, December 2005, http://www.inria.fr/rrrt/rt-0316.html.

[23] N. FURMENTO, G. MERCIER. *Optimisation Mechanisms for MPICH/Madeleine*, Also available as LaBRI Report 1362-05, Technical Report, n° 0306, INRIA, July 2005, http://www.inria.fr/rrrt/rt-0306.html.

Bibliography in notes

[24] *Cluster-of-Clusters(CoC)-Grid Project*, http://www.tu-chemnitz.de/informatik/RA/cocgrid/.

[25] *GM information from Myricom*, http://www.myri.com/scs/.

[26] *MPI: A Message-Passing Interface Standard*, Message Passing Interface Forum, June 1995, http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html.

[27] *PICH-G2: a Grid-enabled Implementation of MPI*, http://www3.niu.edu/mpi/.

[28] T. ANDERSON, B. BERSHAD, E. LAZOWSKA, H. LEVY. *Scheduler Activations: Effective Kernel Support for the User-Level Management of Parallelism*, in "ACM Transactions on Computer Systems", vol. 10, n° 1, February 1992, p. 53-79.

[29] G. ANTONIU, M. JAN, D. A. NOBLET. *Enabling the P2P JXTA Platform for High-Performance Networking Grid Infrastructures*, in "Proc. of the first Intl. Conf. on High Performance Computing and Communications (HPCC '05), Sorrento, Italy", B. D. MARTINO, L. T. YANG, O. F. RANA, J. DONGARRA (editors). , Lect. Notes in Comp. Science, n° 3276, Springer-Verlag, September 2005, p. 429-440.

[30] H. BAL, F. KAASHOEK, A. TANENBAUM. *ORCA: A language for parallel programming of distributed systems*, in "IEEE Transactions on Software Engineering", vol. 18, n° 3, Mar 1992, p. 190-205.

[31] T. BEILSEL, E. GABRIEL, M. RESCH. *An Extension to MPI for Distributed Computing on MPP's*, in "EuroPVM/MPI '97: Recent Advances in Parallel Virtual Machine and Message Passing Interface, Cracow, Pologne", M. BUBACK, J. DONGARRA, J. WASNIEWSKI (editors). , Lecture Notes in Computer Science, vol. 1332, Springer Verlag, novembre 1997, p. 75-83.

[32] R. BHOEDJANG, T. RUHL, H. BAL. *LFC: A Communication Substrate for Myrinet*, in "Fourth Annual Conference of the Advanced School for Computing and Imaging, Lommel, Belgium", June 1998, http://citeseer.ifi.unizh.ch/bhoedjang98lfc.html.

[33] T. BRANDES, F. ZIMMERMANN. *ADAPTOR: A Transformation Tool for HPF Programs*, in "Proceedings of the Conference on Programming Environments for Massively Parallel Distributed Systems", Birkhauser Verlag, April 1994, p. 91-96.

[34] J. BRIAT, I. GINZBURG, M. PASIN, B. PLATEAU. *Athapascan Runtime : Efficiency for Irregular Problems*, in "Proceedings of the Euro-Par '97 Conference, Passau, Germany", Lecture Notes in Computer Science, vol. 1300, Springer Verlag, août 1997, p. 590–599.

[35] F. CAPPELLO, D. ETIEMBLE. *MPI versus MPI+OpenMP on IBM SP for the NAS Benchmarks*, in "Supercomputing", 2000.

[36] M. CHRISTALLER. *Athapascan-0 : vers un support exécutif pour applications parallèles irrégulières efficacement portables*, Ph. D. Thesis, Université Joseph Fourier, Grenoble I, Nov 1996.

[37] A. DENIS, C. PÉREZ, T. PRIOL. *PadicoTM: An Open Integration Framework for Communication Middleware and Runtimes*, in "Future Generation Computer Systems", vol. 19, 2003, p. 575–585, http://www.irisa.fr/paris/Biblio/Papers/Denis/DenPerPri03FGCS.pdf.

[38] A. DENIS, C. PÉREZ, T. PRIOL. *PadicoTM: An Open Integration Framework for Communication Middleware and Runtimes*, in "IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), Berlin, Germany", IEEE Computer Society, May 2002, p. 144-151,

http://www.irisa.fr/paris/Biblio/Papers/Denis/DenPerPri02CCGRID.ps.

[39] I. FOSTER, J. GEISLER, C. KESSELMAN, S. TUECKE. *Managing Multiple Communication Methods in High-performance Networked Computing Systems*, in "Journal of Parallel and Distributed Computing", vol. 40, 1997, p. 35–48.

[40] I. FOSTER, C. KESSELMAN, S. TUECKE. *The Nexus approach to integrating multithreading and communication*, in "Journal of Parallel and Distributed Computing", vol. 37, 1996, p. 70-82.

[41] J.-M. GEIB, C. GRANSART, P. MERLE. *CORBA : des concepts à la pratique*, Inter-Editions, 1997.

[42] P. GEOFFRAY, L. PRYLLI, B. TOURANCHEAU. *BIP-SMP: High Performance message passing over a cluster of commodity SMPs*, in "Supercomputing (SC '99), Portland, OR", Electronic proceedings only, November 1999.

[43] I. GINZBURG. *Athapascan-0b: Intégration efficace et portable de multiprogrammation légère et de communications*, Thèse de doctorat, Institut National Polytechnique de Grenoble, LMC, Sep 1997.

[44] M. HAINES, D. CRONK, P. MEHROTRA. *On the design of Chant: A talking threads package*, in "Proc. of Supercomputing'94, Washington", November 1994, p. 350-359.

[45] R. NAMYST. *PM2 : un environnement pour une conception portable et une exécution efficace des applications parallèles irrégulières*, Thèse de doctorat, Univ. de Lille 1, January 1997.

[46] B. NICHOLS, D. BUTTLAR, J. FARRELL. *Pthreads Programming: POSIX Standard for Better Multiprocessing*, 1996.

[47] S. PAKIN, V. KARAMCHETI, A. CHIEN. *Fast Messages (FM: Efficient, Portable Communication for workstation cluster and Massively-Parallel Processors*, in "IEEE Concurrency", 1997.

[48] L. PRYLLI, B. TOURANCHEAU. *BIP: A new protocol designed for High-Performance networking on Myrinet*, in "1st Workshop on Personal Computer based Networks Of Workstations (PC-NOW '98), Orlando, USA", Lecture Notes in Computer Science, vol. 1388, Springer-Verlag, Held in conjunction with IPPS/SPDP 1998. IEEE, mars 1998, p. 472-485.

[49] T. RUHL, H. E. BAL, R. A. BHOEDJANG, K. G. LANGENDOEN, G. D. BENSON. *Experience with a Portability Layer for Implementing Parallel Programming Systems*, in "International Conference on Parallel and Distributed Processing Techniques and Applications, Sunnyvale, CA", August 1996, p. 1477-1488.

[50] R. RUSSELL, M. CHAVAN. *Fast Kernel Tracing: A Performance Evaluation Tool for Linux*, in "Proceedings of the 19th IASTED International Conference on Applied Informatics, Innsbruck, Austria", February 2001, p. 19-22.

[51] H. TEZUKA, A. HORI, Y. ISHIKAWA, M. SATO. *PM: An Operating System Coordinated High Performance Communication Library*, in "Proceedings of High Performance Computing and Networks (HPCN'97)", Lecture Notes in Computer Science, vol. 1225, Springer Verlag, Avril 1997, p. 708-717.

[52] J. C. DE KERGOMMEAUX, B. DE OLIVEIRA STEIN. *Pajé: an Extensible Environment for Visualizing Multi-Threaded Programs Executions*, in "Proceedings of EuroPar2000, Munich, Allemagne", 2000.