INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Runtime

Efficient Runtime Systems for Parallel Architectures

Futurs

THEME NUM

Activity Report

2007

Table of contents

1. Team

Team Leader
Raymond Namyst [Professor, Université Bordeaux 1, LaBRI, HdR]

Administrative assistant
Sylvie Embolla [Project Assistant]

Staff members
Olivier Aumage [Research Associate (CR1) Inria]
Alexandre Denis [Research Associate (CR1) Inria]
Brice Goglin [Research Associate (CR2) Inria]
Guillaume Mercier [Assistant Professor, ENSEIRB, LaBRI]
Pierre-André Wacrenier [Assistant Professor, Université Bordeaux 1, LaBRI]

Research scientists (partner)
Marie-Christine Counilh [Assistant Professor, Université Bordeaux 1, LaBRI]

Research engineers
Christophe Frézier [Associate Engineer, INRIA, until Sep. 30th 2007]
Nathalie Furmento [Research Engineer, CNRS]
Cécile Romo-Glinos [Associate Engineer, INRIA, from Dec. 1st 2007]

Ph.D. students
Elisabeth Brunet [Regional Grant, LaBRI]
Mathieu Faverge [ANR Grant, LaBRI, from Nov. 2nd 2006]
François Broquedis [Ministry of Research and Technology Grant, LaBRI, from Sep. 1st 2007]
Jérôme Clet-Ortega [Ministry of Research and Technology Grant, LaBRI, from Sep. 1st 2007]
François Diakhaté [CEA Grant, LaBRI, from Sep. 1st 2007]
Stéphanie Moreaud [ANR Grant, LaBRI, from Sep. 1st 2007]
Samuel Thibault [Ministry of Research and Technology Grant, LaBRI, until Sep. 30th 2007]
François Trahay [Ministry of Research and Technology Grant, LaBRI]

2. Overall Objectives

2.1. Designing Efficient Runtime Systems

Keywords: *NUMA*, *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).

One of the main challenges encountered when designing modern runtime systems is to provide powerful abstractions, both at the programming interface level and at the implementation level, to deal with the increasing complexity of upcoming hardware architectures. While it is essential to understand – and somehow anticipate – the evolutions of hardware technologies (e.g. programmable network interface cards, multicore architectures, hardware accelerators), the most delicate task is to extract models and abstractions that will fit most of upcoming hardware features. This is a job that requires monitoring of technological development and... great experience.

Our research project centers on three main directions:

Mastering large, hierarchical multiprocessor machines The emergence of deeply hierarchical architectures based on multi-threaded multi-core chips and NUMA machines raises the need for a careful distribution of threads and data. Indeed, cache misses and NUMA penalties become more and more important with the complexity of the machine, making these constraints as important as parallelization. They require some new programming models and new tools to make the most out of these underlying architectures.

As quoted by Gao *et al.* [52], we believe it is important to expose domain-specific knowledge semantics to the various software components in order to organize computation according to the application and architecture. Indeed, the whole software stack, from the application to the scheduler, should be involved in the parallelizing, scheduling and locality adaptation decisions by providing useful information to the other components. Unfortunately, most operating systems only provide a poor scheduling API that does not allow applications to transmit valuable *hints* to the system.

That's why we investigate new approaches in the design of thread schedulers, focusing on high-level abstractions to both model hierarchical architectures and describe the structure of applications' parallelism. In particular, we have introduced the *bubble* scheduling concept [24], [17], [12] that helps to structure relations between threads in a way that can be efficiently exploited by the underlying thread scheduler. *Bubbles* express the inherent parallel structure of multithreaded applications: they are abstractions for grouping threads which "work together" in a recursive way.

Many research issues remain to be addressed, such as designing specific scheduling algorithms (favoring memory affinity, frequent synchronizations or load balance) or using our framework as a backend for OPENMP compilers.

Optimizing communications over high performance clusters and grids Using a large panel of mechanisms such as user-mode communications, zero-copy transactions and communication operation offload, the critical path in sending and receiving a packet over high speed networks has been drastically reduced over the years. Recent implementations of the MPI standard, which have been carefully designed to directly map *basic* point-to-point requests onto the underlying low-level interfaces, almost reach the same level of performance for very basic point-to-point messaging requests. However more complex requests such as non-contiguous messages are left mostly unattended, and even more so are the irregular and multi-flow communication schemes. The intent of the work on our NEWMADELEINE communication engine, for instance, is to address this situation thoroughly. The NEWMADELEINE optimization layer delivers much better performance on *complex* communication schemes with negligible overhead on basic single packet point-to-point requests. Through Mad-MPI, our proof-of-concept implementation of a subset of the MPI API, we intend to show that MPI applications can also benefit from the NEWMADELEINE communication engine.

Regarding larger scale configurations (clusters of clusters, grids), we intend to propose new models, principles and mechanisms that should allow to combine communication handling, threads scheduling and I/O event monitoring on such architectures, both in a portable and efficient way. We particularly intend to study the introduction of new runtime system functionalities to ease the development of code-coupling distributed applications, while minimizing their unavoidable negative impact on the application performance.

Integrating Communications and Multithreading Asynchronism is becoming ubiquitous in modern communication runtimes. Complex optimizations based on online analysis of the communication schemes and on the de-coupling of the request submission vs processing. Flow multiplexing or transparent heterogeneous networking also imply an active role of the runtime system request submit and process. And communication overlap as well as reactiveness are critical. Since network request cost is in the order of magnitude of several thousands CPU cycles at least, independent computations should not get blocked by an ongoing network transaction. This is even more true with the increasingly dense SMP, multicore, SMT architectures where many computing units share a few NICs. Since portability is one of the most important requirements for communication runtime systems, the usual approach to implement asynchronous processing is to use threads (such as Posix threads). Popular communication runtimes indeed are starting to make use of threads internally and also allow applications to also be multithreaded. Low level communication libraries also make use of multithreading. Such an introduction of threads inside communication subsystems is not going without troubles however. The fact that multithreading is still usually optional with these runtimes is symptomatic of the difficulty to get the benefits of multithreading in the context of networking without suffering from the potential drawbacks. We advocate the importance of the cooperation between the asynchronous event management code and the thread scheduling code in order to avoid such disadvantages. We intend to propose a framework for symbiotically combining both approaches inside a new generic I/O event manager.

Beside those main research topics, we obviously intend to work in collaboration with other research teams in order to *validate* our achievements by integrating our results into larger software environments (MPI, OpenMP, PaStiX) and to *join* our efforts to solve difficult problems.

Among the target environments, we intend to carry on developing the successor to the PM^2 software suite, which would be a kind of technological showcase to validate our new concepts on real applications through both academic and industrial collaborations (CEA/DAM, Bull, IFP, Total). 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.

2.2. Highlights of the year

- MPICH2-NewMadeleine : an efficient multirail implementation of MPI over heterogeneous hardware.
- ForestGOMP is the first NUMA-aware implementation of OpenMP for hierarchical multiprocessors.

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 [42], 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 [34] for communication handling and POSIX-threads [58] for multi-threading management. Environments such as Athapascan [43], Chant [56] or PM^2 [57] 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).

In the past, several proposals (Nexus [49], Panda [61], PM^2 [57]) 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 [60], GM [33], MX [36], FM [59] or LFC [41] 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 [45], [55] for the Athapascan-1 [43] environment, Panda [61] for the Orca [38] compiler, PM [62] for the SCore environment, PM^2 [57] 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 [53], a high-performance communication framework for grids (PadicoTM [48]) and even a multi-network version of the MPICH [7], [6] 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 [35] and PACX-MPI [40]. 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 [32] follows an approach similar to ours through trying to provide an efficient runtime system for such architectures.

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, Quadrics, Infiniband, 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 (Project 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 (GridExplorer cluster at IDRIS, Grid'5000 project, clusters at CEA/DAM, etc.), in Europe (cluster DAS-3 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, Quadrics, Infiniband, etc.) typically provide 10Gb/s throughput and a couple of microseconds latency. Commonly found computing node characteristics range from off-the-shelf PCs to high-end symmetrical multiprocessor (SMP) or non-uniform memory access (NUMA) machines 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. DNA 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 nowadays 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 softwares are actually able to keep up with the technological evolution; while others involve a huge amount of work to adapt the code due to an unsuitable internal design. We showed in [2] 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 "expresses 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 guarantee *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 [47]).

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 [54]). Several studies have shown since then that complex execution models such as those integrating multi-threading and communication (e.g. Nexus [50], [49], Athapascan [43], PM2 [57], MPI+OpenMP [44]), 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 within the clusters (the CEA/DAM, for instance, owns a cluster of 544 16-cores nodes linked with a Quadrics network). These large clusters lead to a set of new issues to be addressed by runtime systems. For instance, a lot of low-level communication libraries do not allow users 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 or a remote DMA based network such as Infiniband. Similar observations can be made about I/O (the Infiniband technology's interoperability with Fiberchannel and iSCSI is bringing in new issues), processors and other peripheral technologies. 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 dealing with 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. NewMadeleine

The Madeleine library which had been the communication subsystem of the PM^2 software suite for almost ten years has now been replaced in production by the NewMadeleine library. NewMadeleine is primarily dedicated to the exploitation of clusters interconnected with (possibly multiple) high-speed networks, potentially of different natures. NewMadeleine is a complete redesign and rewrite of Madeleine. The new architecture is entirely modular and in the process of being completely componentified. The drivers are already available as software components.

The NEWMADELEINE SchedOpt optimizing scheduler aims at enabling the use of a much wider range of communication flow optimization techniques such as packet reordering or cross-flow packet aggregation [13]. SchedOpt targets applications with irregular, multi-flow communication schemes such as found in the increasingly common application conglomerates made of multiple programming environments and coupled pieces of code, for instance. It is designed to be programmable through the concepts of optimization *strategies* (*what* to optimize for, what the optimization goal is) expressed in terms of *tactics* (*how* to optimize to reach the optimization goal), allowing experimentations with multiple approaches or on multiple issues with regard to processing communication flows. Tactics themselves are made of basic communication flows operations such as packet merging or reordering.

Special purpose strategies have also been developped. A strategy is dedicated to heterogeneous multirail support [19]. A QoS strategy [28] is responsible for differentiated service support: it allows to use distinct optimizations and priorities for distinct communication flows.

NEWMADELEINE has strong relationships established with other software projects in the Runtime team, each of whose having been the subject of dedicated work during the last year. Indeed, NEWMADELEINE is the direct core communication library of the Mad-MPI [18] and MPICH-Madeleine modules and is being ported as a communication subsystem target for the MPICH2-Nemesis software from Argonne National Laboratory. It is built upon the PadicoTM software component model, and is now the default communication stack for clusters in PadicoTM. It fundamentally relies on the new PIOMAN module [26] and the MARCEL module for asynchronous communication request processing and progression. It now works together with the Fast User Trace module to provide post-mortem communication schemes analysis. And finally it directly depends on the recent work on *Non-Uniform Input-Ouput Access* (NUIOA) [22], [31] when run on non-uniform hierarchical architectures.

The reference software development branch of the NEWMADELEINE software consists in 62 500 lines of code. NEWMADELEINE is available on various networking technologies: Myrinet, Infiniband, Quadrics, SCI and Ethernet. This library, distributed as part of the PM^2 software is developed and maintained by Olivier AUMAGE, Elisabeth BRUNET, Nathalie FURMENTO and Raymond NAMYST. The software is freely available under the terms of the GNU General Public License version 2 at the following URL: http://runtime.futurs.inria.fr/newmadeleine/.

5.2. Marcel

MARCEL is the thread library of the PM^2 software suite. MARCEL features a two-level thread scheduler (also called N:M scheduler) that achieves the performance of a user-level thread package while being able to exploit multiprocessor machines. The architecture of MARCEL was carefully designed to support a high number of threads and to efficiently exploit hierarchical architectures (e.g. multi-core chips, NUMA machines).

The most important feature of MARCEL is its scheduler, named *BubbleSched*. BubbleSched is a framework that allows scheduling experts to implement and experiment with powerful user-level thread schedulers (http://runtime.futurs.inria.fr/marcel/bubblesched.php). It is based on high-level abstractions called *bubbles*. The application describes affinities between the threads it launches by encapsulating them into nested bubbles (those which work on the same data for instance). BubbleSched then allows to implement various advanced *bubble schedulers* that distribute bubbles (and hence threads) over the hierarchy of the computer so as to benefit from cache effects as much as possible. Memory buffer location may also be taken into account in the scheduler to avoid NUMA factor penalties [30], and other affinities may be involved such as between communication threads and physical I/O devices. A trace of the scheduling events can be recorded and used after execution for generating an animated movie showing a replay of the execution: how bubbles and threads were created, how they got distributed over the machine, how they eventually got scheduled on processors, etc. End users may hence easily try and tune various bubble schedulers for their applications, and select the most suited one.

MARCEL provides a POSIX-compliant interface and a set of original extensions. It can also be compiled to provide ABI-compatibility with NTPL threads under Linux, so that multithreaded applications can use

MARCEL without being recompiled. This permits for instance to run Java applications with MARCEL. 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 NEWMADELEINE and PIOMAN, improving reactivity to communications. Specific softwares matching the needs of PM^2 are also included, allowing thread migration between homogeneous machines.

The reference software development branch of the MARCEL software consists in 61 000 lines of code. This library is developed and maintained by Samuel THIBAULT and Olivier AUMAGE. The software is freely available under the terms of the GNU General Public License version 2 at the following URL: http://runtime.futurs.inria.fr/marcel/.

5.3. ForestGOMP

FORESTGOMP is an OPENMP environment based on both the GNU OPENMP compiler (GCC/GOMP) and the MARCEL thread library. It is designed to schedule efficiently nested sets of threads (derived from nested parallel sections) over hierarchical architectures. Indeed, approaching the theoretical performance of hierarchical multicore machines requires a very careful distribution of threads and data among the underlying non-uniform architecture in order to minimize cache misses and NUMA penalties. Languages extensions such as OPENMP can really enhance the quality of thread scheduling in a portable way, by transmitting precious informations about the parallel structure of the program or the affinities between threads and data to the underlying runtime system.

The FORESTGOMP runtime generates nested MARCEL bubbles, which encapsulate threads sharing common data, each time an OPENMP parallel section is encountered. Then, thanks to BUBBLESCHED abilities, we are able to design NUMA-aware scheduling policies that dynamically map these bubbles onto the various levels of the underlying hierarchical architecture. Two ad-hoc scheduling policies have already been implemented. The *spread scheduler* [23] is designed to deal with multilevel application by consisting of a mere recursive function that greedily distribute a hierarchy of bubbles over the hierarchy of runqueues according to their computation load (either explicitly specified by the programmer, or inferred from the number of threads). The *affinity scheduler* [27], [20] enforces a distribution of threads that maximizes the proximity of threads belonging to the same parallel section, and uses a NUMA-aware work stealing strategy when load balancing is needed. We have validated this approach with highly irregular, fine grain, divide-and-conquer parallel applications.

The experiments we conducted validate this approach in terms of development easiness for the programmer, portability and performance. This is therefore a way for experts to build complex scheduling strategies that take characteristics of the application into account. Using and mixing such strategies, application programmers get a greater control on scheduling of their OPENMP programs.

François BROQUEDIS is the main contributor to this piece of software, it is distributed within the PM^2 software suite. It is freely available under the terms of the GNU General Public License version 2 at the following URL: http://runtime.futurs.inria.fr/forestgomp/index.php

5.4. Mad-MPI

Mad-MPI is a light implementation of the MPI standard. This simple, straightforward proof-of-concept implementation provides a subset of the MPI API, to allow MPI applications to benefit from the NEWMADELEINE communication engine. Mad-MPI is based on the point-to-point nonblocking posting (`isend`, `irecv`) and completion (`wait`, `test`) operations of MPI, these four operations being directly mapped to the equivalent operations of NEWMADELEINE.

Mad-MPI also implements some optimizations mechanisms for derived datatypes [51]. MPI derived datatypes deal with noncontiguous memory locations. The advanced optimizations of NEWMADELEINE allowing to reorder packets lead to a significant gain when sending and receiving data based on derived datatypes.

The Mad-MPI implementation consists in 3 000 new lines of code. It is distributed as part of the PM^2 software and is developed and maintained by Nathalie FURMENTO. The software is freely available under the terms of the GNU General Public License version 2 at the following URL: http://runtime.futurs.inria.fr/MadMPI/.

5.5. MPICH2-NewMadeleine

MPICH2-NEWMADELEINE is the successor to the old MPICH-Madeleine implementation that was derived from MPICH 1.2.7. MPICH2-NEWMADELEINE is based on the most recent MPICH2 software (1.0.6pX) and utilizes the NEWMADELEINE communication library for all network communication. As far as intranode communication are concerned, the Nemesis communication subsystem is employed [9], [14]. Nemesis is a new generic communication subsystem which goal is to address the communication needs of a wide range of programming tools and environments for clusters and parallel architectures. It has been designed to yield very low latency and high bandwidth, especially for intranode communication.

The resulting MPI implementation exhibits excellent performance, especially in the shared-memory case, which in crucial in the case of NUMA clusters. The level of performance is indeed very good and MPICH2-Nemesis compares favourably with other next-generation MPI implementations such as Open MPI or GridMPI. The latencies achieved by MPICH2-NEWMADELEINE in shared-memory are currently the best among generic MPI implementations and are extremely close to that of highly-tuned vendor-specific ports.

The NEWMADELEINE communication library has been integrated as a Nemesis network module but some architectural changes have been made to the upper layers, in particular at the ADI3 level (ch3 has thus been modified). Those changes prefigures more profound changes to come and announce the future merge between Nemesis and NEWMADELEINE. Also, all optimization mechanisms implemented in NEWMADELEINE are made available at the MPI level. For instance, MPICH2, with its NEWMADELEINE Nemesis module can use efficiently multirail configurations.

This work has been initiated by Darius BUNTINAS and Guillaume MERCIER during his postdoctoral stay at the ARGONNE NATIONAL LABORATORY (ANL). Guillaume being still an active member of the MPICH2's development and support team we have regular contacts with ARGONNE NATIONAL LABORATORY regarding the device version of Nemesis. A proposal of Associate Team has therefore been submitted at the end of 2007.

MPICH2-NEWMADELEINE , a joint development between the ANL and the Runtime Project will soon be available on Runtime's website and is developed and maintained by Darius BUNTINAS and Guillaume MERCIER and Elisabeth BRUNET.

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 resources 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 architecture is based on software components. Puk (the PadicoTM micro-kernel) implements a light-weight high-performance component model that is used to build communication stacks. Typical communications stacks built in PadicoTM follow 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 PIOMAN to ensure high performance and good interactions between threads and networking. 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. 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 (Infiniband, 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, our own MPI-Mad), Apache Portable Runtime, JXTA from Sun (in collaboration with the PARIS project), gSOAP, Mome (DSM developed 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, Christophe FRÉZIER, and François TRAHAY (RUNTIME) with some occasional contributions 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 440 times (ranked 51 out of 245 GForge projects) 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: ARA “LEGO” from the ANR, ACI GRID HydroGrid, ACI GRID EPSN and Inria ARC RedGrid. It is also used in the European FET project POP.

5.7. PIOMan

PIOMAN is the event detector server used by the PM^2 software suite. It aims at providing the other software components with a service that can guarantee a predefined level of “reactivity” to I/O events. It is typically used by NEWMADELEINE and PadicoTM to quickly react to network events, such as the arrival of a new packet. PIOMAN is derived from the former MARCEL event server developed by Vincent DANJEAN and thus works closely with MARCEL so as to be triggered upon context switches, processor idleness, etc.

PIOMAN is able to isolate blocking system calls on dedicated threads so that the whole process isn’t suspended. It is actually a portable alternative to the Scheduler Activations model proposed by Anderson [37] and implemented in the LinuxActivations library [10]. By isolating blocking system calls, it becomes possible to suspend only the thread responsible for the blocking call, while the other threads continue their execution. This mechanism makes PIOMAN reactive even during heavy computing phases.

PIOMAN handles both blocking and non-blocking detection methods and is able to choose the more suitable method depending on the processors’ load and the communication library’s preferences. This way, the application is reactive whatever the context is.

The level of reactivity provided by PIOMAN allows NEWMADELEINE to make communications progress in the background (by making the *rendez-vous* handshake progress for instance) and thus to fully overlap computation and communication [26], [25].

MADELEINE and NEWMADELEINE have been ported over PIOMAN. We also plan to use PIOMAN in PadicoTM and in MPI implementations such as MPICH2-Nemesis (within our collaboration with the Argonne National Laboratory) and YAMPII (within the collaboration with the University of Tokyo).

This library, distributed as part of the PM^2 software is developed and maintained by François TRAHAY. The software is freely available under the terms of the GNU General Public License version 2 at the following URL: http://runtime.futurs.inria.fr/pioman/.

5.8. Open-MX

The OPEN-MX software stack is a high-performance message passing implementation for any generic Ethernet interface. It is developped within our collaboration with Myricom, Inc. as a part of the move towards the convergence between high-speed interconnects and generic networks. This convergence has been observed for a couple of years, especially with Myricom, Quadrics or FibreChannel offering interoperability between their specific networks and Ethernet networks, but also with the latter providing features initially developed by the former.

Ethernet networks are very popular in high-performance computing (42 % of the Top500 installations). They achieve improved performance thanks to advanced features that were previously specific to proprietary networks (for instance DMA and Offloading). The need for a high-performance MPI implementation becomes critical. Unfortunately, most work towards high-performance communications focusses on high-performance interconnects such as InfiniBand or Myri-10G. Until now, people have been relying on basic MPI implementations over TCP/IP, for instance MPICH P4 or MPICH2 SSM.

These implementations suffer of TCP/IP performance limitations caused by its design for generic communications while cluster communication specificities should allow multiple optimizations. Some RDMA based implementations [39] have been proposed but they required non-standard features in the hardware and still suffer from several performance issues in the TCP/IP stack.

We are thus developing OPEN-MX [21] in collaboration with Myricom, Inc. to expose the raw Ethernet performance at the application level through a pure message passing protocol. While the goal is similar to the old GAMMA stack [46], the implementation differs since OPEN-MX relies on generic hardware and drivers and has been designed for message passing. OPEN-MX is currently in early development and already shows a good ability to expose the raw performance of any Ethernet hardware to the MPI layers.

OPEN-MX is wire-compatible with Myricom's MX protocol and interface so that any application built for MX may run on any machine without Myricom's hardware and talk other nodes running with or without the native MX stack. OPEN-MX is currently expected to be used as a networking layer for the PVFS2 parallel filesystem on the upcoming BlueGene/P machine in the Argonne laboratory. It will connect BlueGene specific nodes with generic 10 gigabit/s Ethernet boards to generic I/O nodes with Myri-10G running in native MX mode.

This work already enables the study of zero-copy communication implementation on top of generic hardware that has not been designed for such a usage. Several other optimizations such as memory copy offloading will also have to be studied to emulate zero-copy in the corner cases.

Brice GOGLIN is the main contributor to OPEN-MX. The software is already composed of more than 17 000 lines of code in the Linux kernel and in user-space. It is freely available under the terms of the GNU General Public License version 2 at the following URL: http://open-mx.gforge.inria.fr/.

6. New Results

6.1. Communication Optimization over High Speed Networks

The NEWMADELEINE communication subsystem introduces fundamental changes in communication request handling and optimizations. Traditionally, communication libraries, being synchronous, tightly link the communication requests to the application workflow, and therefore transmit incoming packets immediately to the lower network layer without any accumulation. On the contrary, NEWMADELEINE keeps accumulating packets in its optimization window while the NICs are busy. As soon as a NIC becomes idle, the optimization window is analyzed so as to generate a new ready-to-send request to be transfered through the card: NICs are exploited at their maximum (they are not overloaded when there is a high demand of transfers and under exploited when there is not) and the communication optimizations are made *just-in-time* so they closely fit the ongoing communication scheme at any given time.

When at least one of the multiplexing units becomes idle, an *optimization function* is called to elect the next request to be submitted to each idle unit. In doing so, it may select a packet to be sent from the optimization window, or for instance, synthesize a request out of several packets from that window. A wide panel of arguments may be used as an input to the optimizing function. The optimization function is to be selected among an extensible and programmable set of *strategies*. Each strategy aims at some particular optimizing goal. A strategy is itself made of one or more tactics that apply some elementary optimizing operations selected from the panel of usual operations. In particular we have experimented with multi-fragment messages with multiple policies of multi-rail packet balancing on heterogeneous high performance networks [19] and also with providing differentiated services to distinct communication flows [28] through the use of corresponding SchedOpt strategies. NEWMADELEINE showed both its usefulness in conducting such experiments and very good results in terms of latency and bandwidth, while incurring only negligible overhead on basic single packets micro-benchmarks.

Our implementation of MPI, Mad-MPI, has shown that the performance of NEWMADELEINE can be obtained with MPI applications [18]. Mad-MPI has been compared to implementations of MPI for specific high performance networks, MPICH-MX and OPENMPI-MX 1.1 over MYRI-10G, and MPICH-QUADRICS over QUADRICS. This allowed us to evaluate the overhead of Mad-MPI under situations where no optimization is possible, as for example where a MPI ping-pong program exchanges single-segment messages (i.e. contiguous arrays of bytes). On both networks, Mad-MPI introduces a constant overhead of less than 0,5 μs and reaches 1155 Mbytes/s in bandwidth over MYRI-10G and 835 Mbytes/s over QUADRICS.

We have also shown the benefits of the aggregation of small messages, by comparing the performance of a multi-segments ping-pong program, with each "ping" being a sequence of independent `MPI_Isend` operations that use separate MPI communicators. We have observed that Mad-MPI is up to 70 % faster than other implementations of MPI over MX-10G, and up to 50 % faster that MPICH over QUADRICS.

Finally, we have evaluated the performance of our optimization mechanisms when using MPI derived datatypes. We used a ping-pong program which exchanges arrays of a given indexed datatype. The datatype describes a sequence of two data blocks, one small block (64 bytes) followed by a large data block (256 KBytes). Using the NEWMADELEINE scheduling strategy which aggregates all the small blocks (using messages reordering) with the *rendez-vous* requests of the large blocks, Mad-MPI exhibits a gain of about 70 % in comparison with MPICH and about 50 % with OPENMPI over MX and until about 70 % versus MPICH over QUADRICS.

6.2. Low-latency, shared-memory communication within MPICH2

The Nemesis software has been developed in order provide MPICH2 with a high-performance communication subsystem but also to assess the performance of the MPI programing model altogether. Some manufactures indeed expressed their concerns about MPI as being able to efficiently handle communication, in particular in a shared-memory context.

Bearing those concerns in mind, we developed and prototyped Nemesis in order to achieve a very low latency. We also were able to reduce drastically the amount of instructions needed to transfer messages across processes within a single node. Actually, we were able to cut the number of instructions down by a 60% factor. This work demonstrated the relevance of MPI as a programming tool for shared-memory architectures.

We plan to demonstrate the relevance of Nemesis as a communication layer for other programming environments that MPI, as well as to incorporate it into existing communication layers that do not provide efficient shared-memory support.

6.3. Thread Scheduling over Hierarchical Architectures

Exploiting full computational power of current more and more hierarchical multiprocessor machines requires a very careful distribution of threads and data among the underlying non-uniform architecture, so as to minimize the number of remote memory accesses, to favor cache affinities, or to guarantee fast completion of synchronization steps. Unfortunately, most operating systems only provide a limited thread scheduling API that does not allow applications to transmit valuable scheduling hints to the system. This is the main reason why OS schedulers usually perform poorly on NUMA architectures, and especially with irregular applications.

We have proposed to extend classical thread schedulers with high-level abstractions called *Bubbles* [64], [63], which are used to dynamically describe relations between threads in order to improve applications' performance in a portable way. Bubbles let the programmer express relations like data sharing, collective operations, good behavior with regards to co-scheduling on a SMT processor or on a NUMA machine, or more generally a particular scheduling policy need (serialization, preemption, gang scheduling, etc.). 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 developing 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. We have therefore designed a framework named *BubbleSched* that allows scheduling experts to prototype, experiment and implement user-level bubble-based thread schedulers. Non-expert programmers may even try different combinations of existing strategies to schedule the threads of their applications. Of course, such combination may still be difficult from an algorithmic point of view, but with the additional help of the associated post-mortem trace analysis tool, review at will how their applications got scheduled. This lets them easily discover misbehaviours so as to try and tune various scheduling approaches. Thus programmers can really focus on algorithmic issues rather than on gory details.

6.4. Efficient scheduling of OpenMP threads on NUMA machine

To express parallelism, scientific programmers are used to work with OPENMP, a high level parallel language, that relies on a set of annotations (including scheduling directives). While OPENMP-parallelized applications suit well SMP computers, their execution on NUMA architectures are far from being optimal, particularly when considering irregular applications. This is due to the difficulty to combine load balancing and thread/memory affinity relations. Indeed, nowadays OPENMP runtimes do not bind the application parallel structure to the underlying architecture, which may lead to not maintaining the nearness of threads and their most frequently used data.

To solve this problem, we have designed "FORESTGOMP", an extension to the GNU OPENMP runtime system that relies on the MARCEL/BUBBLESCHED thread scheduling package already described in section 5.3. This approach brings ability to define complex scheduling policies to deal with multilevel, hierarchical and fine-grain parallelism: thanks to this framework, OPENMP is now a good tool to tackle irregular applications on NUMA architectures [23].

In particular, we wrote the AFFINITY bubble-scheduler specifically designed to deal with irregular applications based on a divide and conquer scheme. The FORESTGOMP/*Affinity* approach is running the MPU application about 3 times faster than the GOMP/PTHREAD environment on a 16 cores computer [20]. On the same way, tests performed with the BT-MZ application on the same computer results in a 10.2 speedup for the FORESTGOMP/*spread* approach, while GOMP/PTHREAD reaches a speedup of 7.

6.5. 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. Since year 2005, PadicoTM is built with true *software components*. Communications methods, network access, and paradigm adaptation are implemented as components. Thus, using components as building blocks, the user may assemble communication stacks following the needs of the application and the requirements imposed by the network infrastructures. 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 control the component assembly process, and to allow advanced communication methods that require negotiation or synchronization between nodes, we have shown that a *control channel* used for bootstrap and out-of-band communication is required. We proposed [15] a novel approach for the management of this control channel as an overlay network that combine security, connectivity, and high performance.

We have started an initiative called Madico to merge PadicoTM and MADELEINE, in order to improve the integration of PadicoTM, NEWMADELEINE, and MARCEL. We have proposed a new design based on software components for NEWMADELEINE, and the implementation is ongoing. Strategies and drivers are already embedded into components. Our goal is to reach same flexibility in configuration and dynamicity at the cluster level as we have at the grid level, and to enable the use of NEWMADELEINE schedulers on large scale networks where NEWMADELEINE optimizations make sense too. Moreover, we have integrated the PadicoTM control channel infrastructure into NEWMADELEINE to get dynamicity at every level.

Finally, we worked on widening the availability of middleware systems over PadicoTM, and its use by other projects. 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; this will be used in the “LEGO” ANR project. We are currently working closely with the Scalapplix project-team to use PadicoTM in the EPSN (http://epsn.gforge.inria.fr) steering software to couple visualisation on a dedicated cluster and simulation running on a computational grid. The Salomé project (CEA/EDF, http://www.salome-platform.org/) is very interested in the connectivity features from PadicoTM and actually employed an intern in collaboration with RUNTIME to finalize the implementation of the PadicoTM control channel. Moreover, we ported the MPI implementation YAMPII/GridMPI (http://www.gridmpi.org/) from the University of Tokyo over PadicoTM with good results in the context of the NEGST grant (CNRS-JST) with Japan.

6.6. Reactivity to I/O events

Nowadays, communication libraries for high speed networks achieve very low latencies, close to the performance of the underlying hardware. Actually, this is true only when data transfers are done in an “undisturbed environment”, i.e. the resources (CPU, memory bus, network interface) are fully available. In real applications, the property of low latency is hard to obtain, because the runtime system (or the operating system) is unable to react to network I/O events in a short time.

We showed that, by using a centralized I/O event server, a high level of “reactivity” can be guaranteed in a portable way even during heavy computing phases. We have designed a scalable architecture for this I/O server named PIOMAN. By interacting with the thread scheduler, PIOMAN detects the completion of the communication queries (either by polling the network or waiting for interrupts) and triggers the appropriate callback as soon as possible. The progression of asynchronous communications in the background can thus be performed efficiently, allowing a full overlap of communications and computations [26], [25].

PadicoTM, MADELEINE and NEWMADELEINE are already using PIOMAN, making them reactive even when running many computing threads. We are currently developping an enhanced version of NEWMADELEINE that would optimize the communications more efficiently thanks to the multithreading support provided by PIOMAN.

6.7. NUMA-aware placement of communications

Clusters of NUMA nodes are becoming increasingly popular in high-performance computing thanks to AMD Opteron processors. It is well known that scheduling threads and placing memory properly on such nodes has a strong impact on the overall intra-node computing performance because of NUMA affinities between threads and memory. However, the impact of non-uniform access on input/output performance had not been studied extensively. Yet, it is known to have a non-negligible effect since people had been used to manually place communication benchmarks carefully on the processors and memory banks close to the network interface.

We ran an extensive study of these *Non-Uniform Input-Ouput Access* (NUIOA) effects on multi-Opteron machines. It exhibited a strong bandwidth degradation (up to 40 % among a 2 GB/s communications) when placing communication buffers on memory banks that are far from a high-performance network interface. Moreover, this impact is actually asymmetric since it only applies to buffers where it is written, not where it is read. On the latency side, the effect of placement is visible but almost negligible (hundreds of nanoseconds at most).

We then implemented in NEWMADELEINE an automatic placement strategy which queries NUMA affinities from low-level network drivers and let the communication middleware place communicating tasks and buffers accordingly. It enables performance portability by guarantying that communication performance will be as good as if manually placed. These results have been published in [22].

6.8. High-performance message passing over generic Ethernet hardware

The OPEN-MX message passing stack 5.8 is in early development but it still runs and provides a native message passing layer on any Ethernet hardware.

The API compatibility with the native Myrinet Express stack already enables existing parallel application to use OPEN-MX. Indeed, the regular MPICH-MX layer works transparently on top of OPEN-MX with satisfying performance [21]. Also, the PVFS2 storage system has been successfully tested and his under experimentation in the Argonne national laboratory in the context of the BlueGene/P installation.

7. Contracts and Grants with Industry

7.1. PhD thesis co-supervised with CEA/DAM

3 years, 2007-2010

We did set up a collaboration with the CEA/DAM (French Atomic Energy Commission, Marc PÉRACHE, 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 October 2007, François DIAKHATÉ has started a PhD thesis granted by the CEA under the co-supervision of Marc PÉRACHE and Raymond NAMYST about the use of virtualization within parallel applications over clusters of multiprocessor machines.

7.2. Contract between INRIA and Myricom

We have setup a collaboration with Myricom, Inc. (US Company building high-speed interconnect hardware and software) regarding the design and implementation of a message passing protocol on top of generic Ethernet interfaces. Myricom, Inc. provides us with high-performance networking hardware while we develop the OPEN-MX software stack to expose a high-performance MPI layer on top of any generic generic Ethernet interfaces. This contract started in July 2007 and is expected to last at least one year.

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 committee. 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. NEGST (NExt Grid Systems and Techniques)

3 years, 2006-2009.

This project is funded by the CNRS and Japan Science and Technology Agency and is led by Serge PETITON (INRIA Grand-Large) and Ken MIURA (National Institute of Informatics Center for Grid Research and Development).

It aims at promoting collaborations between Japan and France on grid computing technology. Following successful France-Japan workshops hosted by CNRS in Paris and NEREGI/NII in Tokyo, three important novel research issues have been identified: 1) Instant Grid and virtualization of grid computing resources, 2) Grid Metrics and 3) Grid Interoperability and Applications. The objective is to accelerate the intensive works of several research teams in these subjects in both countries. An international testbed including the French Grid'5000 project and its Japanese counterpart NEREGI will be use to demonstrate and validate systems, software and applications.

8.3. PHC Pessoa MAE Grant

We are setting up a collaboration with the team of Associate Professor Salvador Abreu at University of Évora, Portugal, about the design of a constraint solving engine (such as GNU Prolog) for parallel architectures. We intend to build the engine on top of the runtime systems provided in our team. We also would like to study the exploitation of new processing hardware such as the heterogeneous multicore IBM Cell which shows some interesting potential for this specific use. This contract is expected to start at the beginning of 2008.

8.4. PHC Sakura MAE Grant

We are setting up a collaboration with the team of Professor Yutaka Ishikawa at University of Tokyo, Japan, about the design and implementation of high performance communication middleware for new generation parallel computer architectures. We are aiming at studying and promoting possible convergence points between the NEWMADELEINE, PIOMAN and PadicoTM software on the RUNTIME team side and the YAMPI and GRIDMPI software on the University of Tokyo side. In particular, we expect YAMPI to benefit from the NEWMADELEINE packet scheduler when performing multi-rail communications, and we would like to experiment with the virtualization capacity of the PadicoTM to enable the IMPI (Interoperable MPI) support of GridMPI to run at full speed on high speed networks when available while maintaining "on the wire" TCP interoperability when conversing with non-PadicoTM communication stacks on the other side. This contract is expected to start at the beginning of 2008.

8.5. Associate Team Program with ANL

We submitted a proposal in order to create an Associate Team between our group and the Radix Lab located at Argonne National Laboratory (Chicago, USA). The Radix Lab is involved in research about operating systems, parallel programming and MPI implementations. In particular, they developed the MPICH2 implementation. Since Guillaume MERCIER spent 18 months as a post-doc in this group, it seemed rather natural to apply to the “Associate Team Program” because we share a lot of research topics and we could efficiently pool our resources regarding parallel programming models and implementations. We expect to efficiently make converge the Nemesis and NEWMADELEINE software in order to create a new high-performance runtime system tailored for the MPICH2 implementation but that could benefit other current implementations. Since the MPICH2 group currently doesn’t develop its own thread library our knowledge in the field of multithreading will be of great help in order to better address this complicated matter. In the long run, we also expect to work on a possible new model that would address the issues raised with the NUMA and multi-cores architectures.

8.6. Committees

Raymond NAMYST was co-chair of the EXPGRID (Experimental Grid testbeds for the assessment of large-scale distributed applications and tools) workshop held in conjunction with the 15th International Symposium on High Performance Distributed Computing (HPDC-15). He is also member of the *EuroPVMMPI 2007*, *EuroPVMMPI2008*, *Cluster 2008*, *CCGRID’08* and *RenPar 2008* program committees.

8.7. Invitations

Brice GOGLIN has been invited at the CLUSTERVISION Users’ Group in Cheltenham, UK, in October 2007 to present Runtime and ScAlApplix project research activities using latest hardware available in Grid’5000 in Bordeaux.

Elisabeth BRUNET and Guillaume MERCIER have visited Argonne National Laboratory for a two-weeks period in September 2007. The goal of this visit was twofold: firstly Elisabeth BRUNET made a presentation of the NEWMADELEINE communication library and its mechanisms. Secondly, this visit was an opportunity to discuss about setting up an “Associate Team” between Runtime and the Radix Lab, responsible for the development of the MPICH2 software.

Alexandre DENIS and Guillaume MERCIER have been invited to visit Prof. Ishikawa's group in Tokyo University in Japan. Informal contacts between the Runtime group and this Japanese group have going for a couple of months and this two-weeks visit that took place in April 2007 was an opportunity to discuss about strengthening our collaboration. After this visit, both groups decided to apply to the Sakura program, as well as to the Ayame program. A proposal of Associate Team has also been submitted. During this visit Alexandre DENIS successfully managed to integrate the Japanese GRIDMPI over his PadicoTM communication software. We also started some discussion about the support of IMPI into the MPICH2 implementation.

9. Dissemination

9.1. Reviews

Olivier AUMAGE was involved in the paper reviewing process of the Transaction on Parallel and Distributed Systems IEEE journal and the EuroPVM/MPI conference.

Brice GOGLIN was involved in the paper reviewing process of the International Conference on Cluster Computing (Cluster 2007).

Alexandre DENIS was involved in the paper reviewing process of the International Conference on Cluster Computing (Cluster 2007) and the EuroPVM/MPI 2007 conference.

Raymond NAMYST has reviewed 2 PhD Thesis during the year 2007.

9.2. Seminars

Raymond NAMYST gave several seminars about multi-core programming at the CEA (Mar. 2007), the LaBRI (Apr. 2007) and the ENS Lyon (Oct. 2007).

Olivier AUMAGE gave a seminar about NEWMADELEINE at the LIG/Montbonnot (Mar. 2007). He also gave a seminar for the members of the LEGO grant at the ANR mid-term evaluation meeting in Paris (Sep. 2007).

François TRAHAY gave a seminar about PIOMAN at the LaBRI (Nov. 2007).

9.3. 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.

Brice GOGLIN gave a course on “Parallel and Distributed Systems” 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, 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^o 1, January 2002, p. 43-54, http://runtime.futurs.inria.fr/Download/Publis/AumBouDenEyrMehMerNamPry01CC.ps.gz.

[3] O. AUMAGE, L. BOUGÉ, L. EYRAUD, R. NAMYST. *Communications efficaces au sein d'une interconnexion hétérogène de grappes : Exemple de mise en oeuvre dans la bibliothèque Madeleine*, in "Calcul réparti à grande échelle", F. BAUDE (editor), ISBN 2-7462-0472-X, Hermès Science Paris, 2002.

[4] 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^o 4, April 2002, p. 607–626, http://runtime.futurs.inria.fr/Download/Publis/clustercomputing2k1.ps.gz.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] D. BUNTINAS, G. MERCIER, W. GROPP. *Implementation and Shared-Memory Evaluation of MPICH2 over the Nemesis Communication Subsystem*, in "Recent Advances in Parallel Virtual Machine and Message Passing Interface: Proc. 13th European PVM/MPI Users Group Meeting, Bonn, Germany", September 2006.

[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.

Year Publications

Doctoral dissertations and Habilitation theses

[12] S. THIBAULT. *Ordonnancement de processus légers sur architectures multiprocesseurs hiérarchiques : BubbleSched, une approche exploitant la structure du parallélisme des applications.*, Ph. D. Thesis, Univ. Bordeaux 1, December 2007.

Articles in refereed journals and book chapters

[13] E. BRUNET. *NewMadeleine : ordonnancement et optimisation de schémas de communication haute performance (version étendue de Perpi'06).*, in "Technique et Science Informatiques", To appear, 2008.

[14] D. BUNTINAS, G. MERCIER, W. GROPP. *Implementation and Evaluation of Shared-Memory Communication and Synchronization Operations in MPICH2 using the Nemesis Communication Subsystem*, in "Parallel Computing, Selected Papers from EuroPVM/MPI 2006", vol. 33, n^{o} 9, September 2007, p. 634–644.

[15] A. DENIS. *Meta-communications in component-based communication frameworks for grids*, in "Cluster Computing", vol. 10, n^{o} DOI 10.1007/s10586-007-0036-5, June 2007, p. 253-263.

[16] B. GOGLIN, O. GLÜCK, P. V.-B. PRIMET. *Interaction efficace entre les réseaux rapides et le stockage distribué dans les grappes de calcul*, in "Technique et Science Informatiques", To appear, 2008.

[17] S. THIBAULT, R. NAMYST, P.-A. WACRENIER. *BubbleSched: une plate-forme pour la conception d'ordonnanceurs de threads portables sur machines multiprocesseurs hiérarchiques*, in "Technique et Science Informatiques", To appear, 2008.

Publications in Conferences and Workshops

[18] O. AUMAGE, E. BRUNET, N. FURMENTO, R. NAMYST. *NewMadeleine: a Fast Communication Scheduling Engine for High Performance Networks*, in "CAC 2007: Workshop on Communication Architecture for Clusters, held in conjunction with IPDPS 2007, Long Beach, California, USA", Also available as LaBRI Report 1421-07 and INRIA RR-6085, March 2007, http://hal.inria.fr/inria-00127356.

[19] O. AUMAGE, E. BRUNET, G. MERCIER, R. NAMYST. *High-Performance Multi-Rail Support with the NewMadeleine Communication Library*, in "HCW 2007: the Sixteenth International Heterogeneity in Computing Workshop, held in conjunction with IPDPS 2007, Long Beach, California, USA", March 2007, http://hal.inria.fr/inria-00126254.

[20] F. BROQUEDIS, F. DIAKHATÉ, S. THIBAULT, O. AUMAGE, R. NAMYST, P.-A. WACRENIER. *Scheduling Dynamic OpenMP Applications over Multicore Architectures*, in "Proceedings of the 22th International Parallel and Distributed Processing Symposium (IPDPS 2008), Miami, FL", Submitted, IEEE, April 2008.

[21] B. GOGLIN. *Design and Implementation of Open-MX: High-Performance Message Passing over generic Ethernet hardware*, in "CAC 2008: Workshop on Communication Architecture for Clusters, held in conjunction with IPDPS 2008, Miami, FL", Submitted, IEEE, April 2008.

[22] S. MOREAUD, B. GOGLIN. *Impact of NUMA Effects on High-Speed Networking with Multi-Opteron Machines*, in "The 19th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2007), Cambridge, Massachussetts", November 2007, http://hal.inria.fr/inria-00175747.

[23] S. THIBAULT, F. BROQUEDIS, B. GOGLIN, R. NAMYST, P.-A. WACRENIER. *An Efficient OpenMP Runtime System for Hierarchical Architectures*, in "International Workshop on OpenMP (IWOMP), Beijing,China", 6 2007, p. 148–159, http://hal.inria.fr/inria-00154502.

[24] S. THIBAULT, R. NAMYST, P.-A. WACRENIER. *Building Portable Thread Schedulers for Hierarchical Multiprocessors: the BubbleSched Framework*, in "EuroPar, Rennes,France", ACM, 8 2007, http://hal.inria.fr/inria-00154506.

[25] F. TRAHAY, E. BRUNET, A. DENIS. *A multithreaded communication engine for multicore architectures*, in "CAC 2008: Workshop on Communication Architecture for Clusters, held in conjunction with IPDPS 2008, Miami, FL", Submitted, IEEE, April 2008.

[26] F. TRAHAY, A. DENIS, O. AUMAGE, R. NAMYST. *Improving Reactivity and Communication Overlap in MPI using a Generic I/O Manager*, in "EuroPVM/MPI", F. CAPPELLO, T. HERAULT, J. DONGARRA (editors), Lecture Notes in Computer Science, vol. Recent Advances in Parallel Virtual Machine and Message Passing Interface, n^{o} 4757, Springer, 2007, p. 170-177, http://hal.inria.fr/inria-00177167.

Internal Reports

[27] F. BROQUEDIS. *De l'exécution structurée de programmes OpenMP sur architectures hiérarchiques*, Mémoire de DEA, June 2007, http://hal.inria.fr/inria-00177150.

[28] J. CLET-ORTEGA. *Ordonnancement et qualité de service pour réseaux rapides*, Mémoire de DEA, June 2007, http://hal.inria.fr/inria-00177230.

[29] F. DIAKHATÉ. *Reconstruction parallèle de surfaces par partition de l'unité hiérarchique*, Mémoire de DEA, June 2007, http://runtime.futurs.inria.fr/Download/Publis/Dia07Master.pdf.

[30] S. JEULAND. *Ordonnancement de threads dirigé par la mémoire sur architecture NUMA*, Mémoire de DEA, September 2007, http://hal.inria.fr/inria-00177129.

[31] S. MOREAUD. *Impact des architectures multiprocesseurs sur les communications dans les grappes de calcul : de l'exploration des effets NUMA au placement automatique*, Mémoire de DEA, June 2007, http://hal.inria.fr/inria-0017749.

References in notes

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

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

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

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

[36] *Myrinet Express (MX): A High Performance, Low-Level, Message-Passing Interface for Myrinet*, 2006, http://www.myri.com/scs/.

[37] 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^{o} 1, February 1992, p. 53-79.

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

[39] P. BALAJI, H.-W. JIN, K. VAIDYANATHAN, D. K. PANDA. *Supporting iWARP Compatibility and Features for Regular Network Adapters*, in "Proceedings of the Workshop on Remote Direct Memory Access (RDMA): Applications, Implementations, and Technologies (RAIT); held in conjunction with the IEEE International Confer ence on Cluster Computing, Boston, MA", September 2005.

[40] 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.

[41] 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.ist.psu.edu/bhoedjang98lfc.html.

[42] 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.

[43] 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.

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

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

[46] G. CIACCIO, G. CHIOLA. *GAMMA and MPI/GAMMA on GigabitEthernet*, in "Proceedings of 7th EuroPVM-MPI conference, Balatonfured, Hongrie", Lecture Notes in Computer Science, vol. 1908, Springer Verlag, Septembre 2000.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] N. Furmento, G. Mercier. *Optimisation Mechanisms for MPICH-Madeleine*, Also available as LaBRI Report 1362-05, Technical Report, n^{o} 0306, INRIA, July 2005, http://hal.inria.fr/inria-00069874.

[52] G. R. Gao, T. Sterling, R. Stevens, M. Hereld, W. Zhu. *Hierarchical multithreading: programming model and system software*, in "20th International Parallel and Distributed Processing Symposium (IPDPS)", April 2006.

[53] J.-M. Geib, C. Gransart, P. Merle. *CORBA : des concepts à la pratique*, Inter-Editions, 1997.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] B. Nichols, D. Buttlar, J. Farrell. *Pthreads Programming: POSIX Standard for Better Multiprocessing*, 1996.

[59] S. Pakin, V. Karamcheti, A. Chien. *Fast Messages (FM: Efficient, Portable Communication for workstation cluster and Massively-Parallel Processors*, in "IEEE Concurrency", 1997.

[60] 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.

[61] 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.

[62] 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.

[63] 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/.

[64] 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/.