INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team Runtime

Efficient Runtime Systems for Parallel Architectures

Futurs

THEME NUM

Table of contents

1. Team

Head of project team
Raymond Namyst [Professor, Université Bordeaux 1, LaBRI]

Administrative assistant
Brigitte Larue-Bourdon [Project Assistant]

Staff members
Olivier Aumage [Research Associate (CR) Inria]
Alexandre Denis [Research Associate (CR) Inria (starting from oct. 1st 2004)]
Pierre-André Wacrenier [Assistant Professor, Université Bordeaux 1, LaBRI]

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

Ph.D. students
Vincent Danjean [Ministry Grant, LIP]
Guillaume Mercier [Ministry Grant, LaBRI]
Marc Perache [CEA Grant]
Samuel Thibault [Ministry Grant, LaBRI (starting from oct. 1st 2004)]

2. Overall Objectives

2.1. Designing Efficient Runtime Systems

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

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

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

The research project centers on three main challenges:

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

Optimally exploiting new technologies. It is definitely mandatory to keep an eye over the evolution of hardware technologies (networks, processors, operating system design) to better understand the constraints imposed by real production machines and to study how to get the most out of these new technologies. On that particular point, we must undoubtedly carry on the work we have begun about interface expressiveness which allows a separation of the application requirements from the runtime system-generated optimizations. In the near future, we intend to experiment with the Infiniband and

Quadrics technologies and related potential communication optimizations. We also plan to study the scheduling of threads on SMT multi-processors.

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

2.2. Meeting the Needs of Programming Environments and Applications

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

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

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

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

3. Scientific Foundations

3.1. Runtime Systems Evolution

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

Nowadays, when intending to implement complex parallel programming environments, the use of runtime systems is unavoidable. For instance, parallel languages compilers generate code which is getting more and more complex and which relies on advanced runtime system features (e.g. the HPF Adaptor compiler [25], 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 [19] for communication handling and POSIX-threads [40] for multi-threading management. Environments such as Athapascan [26], Chant [38] or PM^2 [39] well illustrate this trend. Even though such approaches are still widespread, they do suffer from numerous limitations related

to *functional* incompatibilities between the various software components (decreased performance) and even to *implementation* incompatibilities (e.g. thread-unsafe libraries).

Several proposals (Nexus [33], Panda [43], PM^2 [39]) 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 [42], GM [18], FM [41] or LFC [24] 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 [28][37] for the Athapascan-1 [26] environment, Panda [43] for the Orca [22] compiler, PM [45] for the SCore environment, PM^2 [39] 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 [35], a computational grid-enabled multi-application environment (PadicoTM [32]) and even a multi-network version of the MPICH [8][7] library.

3.2. Current Trends

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

Even though several problems still remain unresolved so far (communication schemes optimization, reactivity to I/O events), we now have at our disposal efficient runtime systems that *do* efficiently exploit small-scale homogeneous clusters. However, the problem of mastering large-scale, hierarchical and potentially heterogeneous configurations (that is, clusters of clusters) still has to be tackled. Such configurations bring in many new problems, such as high-performance message routing in a heterogeneous context, dynamic configuration management (fault-tolerance). There are two interesting proposals in the particular case of heterogeneous clusters of clusters, namely MPICH-G2 [20] and PACX-MPI [23]. 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 [17] follows an approach similar to ours through trying to provide an efficient runtime system for such architectures. A preliminary contact has already been established in order to set up a collaboration about this topic.

Besides, even if the few aforementioned *success stories* demonstrate that current runtime systems actually improve both portability and performance of parallel environments, a lot of progress still has to be made with regards to the optimal use of runtime systems features by the higher level software layers. Those upper layers still tend to use them as mere "black-boxes". More precisely, we think that the expertise accumulated by a runtime system designer should be formalized and then transferred to the upper layers in a systematic fashion (code analysis, specialization). To our knowledge, no such work exists in the field of parallel runtime systems to date. The Compose project (LaBRI, Futurs Research Unit), has a strong expertise in the field of code analysis and code specialization. We intend to collaborate with the members of Compose and to use some of their tools in order to study the optimization potential that can be expected from this kind of approach.

The members of the `RUNTIME` project have an acknoledged expertise in the parallelization of complex applications on distributed architectures (combinatorial optimization, 3D rendering, molecular dynamics), the design and implementation of high performance programming environments and runtime systems (PM2), the

design of communication libraries for high speed networks (Madeleine) and the design of high performance thread schedulers (Marcel, LinuxActivations).

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

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

Regarding multithreading, our research efforts have mainly focused on designing a multi-level "chameleon" thread scheduler (its implementation is optimized at compilation time and tailored to the underlying target architecture) and on extending the *Scheduler Activations* [30] mechanism that provides a tight control on kernel threads scheduling in the presence of I/O events (to guarantee the application's reactivity).

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

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

4. Application Domains

4.1. Panorama

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

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

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

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

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

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

Moreover, and beside semantic constraints, runtime systems should also address an increasing number of functional needs, such as fault tolerant behavior or dynamically resizable computing sessions. In addition, more specific needs should also be taken into account, for example the need for multiple independent logical communication channels in modular applications or multi-paradigm environments (e.g. PadicoTM [31]).

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 [36]). Several studies have shown since then that complex execution models such as those integrating multi-threading and communication (e.g. Nexus [34][33], Athapascan [26], PM2 [39], MPI+OpenMP [27]), are in fact much more efficient.

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

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

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

5. Software

5.1. Madeleine

The Madeleine library is the communication subsystem of the PM^2 software suite. This communication library is principally dedicated to the exploitation of clusters interconnected with high-speed networks, potentially of different natures. Madeleine is a *multithreaded* library both in its conception (use of lightweight processes to implement some functionalities) and in its use: Madeleine's code re-entrance enables it to be used jointly with the Marcel library. Moreover, Madeleine is a *multi-cluster* communication library that implements a concept of communication *channel* that can be either physical (that is, an abstraction of a physical network) or virtual. In that latter case, it becomes possible to build virtual heterogeneous networks. Madeleine features a message forwarding mechanism that relies on gateways when permitted by the configuration (that is, when

several different networking technologies are present on the same node). Madeleine is also able to dynamically select the most appropriate means to send data according to the underlying technology (*multi-paradigms*). This is possible by specifying constraints on data to be sent ("design by contract" concept) and provides a good performance level above technologies possibly relying on very different paradigms. Madeleine relies on external software regarding deployment, session management (the *Léonie* software), or exploitation of user-given information (configuration files). Madeleine is available on various networking technologies: Myrinet, SCI, Ethernet or VIA and runs on many architectures: Linux/IA32, Linux/IA64, Linux/Alpha, Linux/Sparc, Linux/PowerPC, Solaris/Sparc, Solaris/IA32, AIX/PowerPC, WindowsNT/IA32. Madeleine and its external software roughly consists of 55000 lines of code and 116 files. This library, available within the PM^2 software is developed and maintained by Olivier AUMAGE, Raymond NAMYST and Guillaume MERCIER.

5.2. Marcel

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

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

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

5.3. LinuxActivations

LinuxActivations is a piece of software that can be used with the Marcel thread library. It is an extension of the Linux kernel allowing an efficient control of the user-level threads scheduling when they perform blocking I/O operations in the kernel. LinuxActivations is based on an extension of the *Scheduler Activations* model proposed by Anderson [21]. *Upcalls*, the opposite of system calls, are used to notify the user-level scheduler about the events triggered by the kernel. Usually, when a user-level thread within a process performs a blocking system call, the whole process is suspended. With the *upcalls* mechanism, it becomes possible to suspend only the thread responsible for the blocking call, the other threads continuing their execution. Our contribution is to minimize the elapsed time between the detection of an I/O event in the kernel and the scheduling of the corresponding user-level thread in the application. The processing of I/O events still occur within the kernel but the decisions concerning the scheduling are now handled at the user level.

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

5.4. MPICH/MADIII

MPICH-Madeleine is a high-performance implementation of the MPI (*Message Passing Interface*) standard and targets hardware configurations that implies the need for multiprotocol capabilities: homogeneous SMP clusters or heterogenous clusters of clusters. It is based on a multithreaded progression engine that allows communications to progress independently from the computation. Precisely, calls to MPI routines are not required

to enforce communication's progress. A side-effect of this multithreaded architecture is that our MPI implementation supports the highest level of multithreading for MPI applications, that is, `MPI_THREAD_MULTIPLE`. MPICH-Madeleine is therefore based on the Marcel user-level thread library and relies on its optimized polling mechanisms to guaranty a high level of reactivity.

Regarding low-level data transfers, MPICH-Madeleine utilizes the Madeleine generic communication library that provides us with a common limited interface above all low-level network protocols. The drawback that arises is the impossibility to finely optimize the upper levels of the MPI implementation accordingly to the underlying low-level protocols available but thanks to the carefully designed Madeleine interface and a tight integration, the performance level achieved is similar or even better than that of highly specialized MPI implementations dedicated to a specific high-speed network.

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

The software is freely available at the following URL: http://dept-info.labri.fr/~mercier/mpi.html. MPICH-Madeleine is developped, updated and maintained by Guillaume MERCIER.

6. New Results

6.1. A Fast implementation of MPI on clusters of clusters

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

First, we designed a specific communication device for handling all shared-memory communication. This new device allows MPICH/Madeleine to fully exploit SMP nodes and its performance is very good compared to other similar devices such as those developed within MPICH-GM, SCI-MPICH, MPICH-P4 or MPICH-SHMEM. MPICH-Madeleine is now able to take into account not only the heterogeneous nature of clusters but also their hierarchical structure.

Then, we have enhanced the communication library capabilities so that it can handle dynamic applications. Typical targets of this developement are computational steering applications. A new type of communication channels (i.e. *mergeable channels*) have been introduced in Madeleine. This new feature has consequently been ported into MPICH-Madeleine leading to an MPI implementation that now supports dynamical applications.

Finally, we did validate the overall architecture of MPICH/Madeleine by conducting exhaustive performance evaluations:

- Comparisons with Madeleine have shown that the impact of the introduction of the multithreaded communication engine is not a hurdle to obtain a high performance level. Indeed, MPICH-Madeleine is able to fully exploit Madeleine'scapabilities.
- Comparisons in homogenous SMP clusters configurations with other specialized MPI implementations such as MPICH-GM (for the myrinet network), SCI-MPICH (for the Scalable Coherent Interface network) or MPICH-P4 (for the Gigabit Ethernet network) have shown that the use of a generic communication library is a sensible approach and leads to excellent results for both basic tests of the point-to-point communication routines as well as for more realistic applications such as High Performance Linpack. Precisely, the tests carried out with such an linear algebra application showed that the use of a preemptive user-level thread library was a good choice and did nott negatively impact the overall behaviour of this intensive computation application.
- Comparisons with MPICH-G2 and PACX-MPI have confirmed the efficiency of the MPICH-Madeleine software since performance evaluation are favourable to us in heterogeneous clusters of clusters cases.

An article detailling the aformentioned results was submitted to the CCGRID 2005 conference. This paper is common effort between the Runtime and the German CoC-Grid projects leading the way for further active collaboration.

Also,we are currently working closely with the research group that is developping the MPICH software at Argonne National Laboratory for integrating the mechanisms created for MPICH-Madeleine into their MPICH2 software. Guillaume MERCIER has been invited for a couple of weeks and made a first implementation of MPICH2 above the Madeleine communication library. The premiminary results are encouraging and Guillaume MERCIER has consequently been appointed as a post-doc within their team for at least one year.

6.2. Flexible and efficient multithreaded library

In order to adapt our thread library MARCEL to the new hierarchical architectures (HyperThreading technology, ccNUMA), a full rewrite as been done. This new version of MARCEL is fully modularized. This leads to several improvements:

- MARCEL's architecture mainly follows the one of Linux 2.6 in order to take benefit from linux's very low level code (like synchronisation tools and SoftIRQ mecanisms); this code is robust and generaly well optimized over a lot of computer architectures.
- MARCEL specializes itself with respect to the underlying architecture. On monoprocessor machine, MARCEL is a very efficient pure user-level thread library. On multiprocessors, MARCEL becomes a two-level thread library so that it can exploit all the processors of the machine. If the architecture is hierarchical (SMT, ccNUMA), MARCEL allows the application to take care of this situation and to group its threads with respect to the hierarchical architecture.
- MARCEL specializes itself with respect to the underlying software. When MARCEL needs several execution flows to exploits multiprocessors machines, it choose the best method available on the target system. For example, it can use the POSIX threads on any standard Unix, but it can also use the `clone` system call on Linux systems or even the Scheduler Activations if this mechanism is available.
- MARCEL specializes itself with respect to applications' needs. Features such as traces or signals handling are included if and only if the application requires it.

This new organization of our thread library allowed to port it easily on new architectures. In particular, MARCEL works on the NovaScale architecture from Bull: a cluster of ccNUMA with 16 to 32 Itanium processors per node.

Furthermore, the thread scheduler is now able to use application directives (regarding memory affinity between threads) to better schedule threads on hiearchical architectures (e.g. NUMA machines). It is based on a powerful "bubble" mechanism that allows the programmer to build hierarchical affinity sets with no dependency upon the underlying architecture. The scheduler uses this information to maximize the locality of threads belonging to the same bubble while still trying to keep all the processors busy. An article detailling this work was submitted to the RenPar'2005 conference.

6.3. Distributed Communication Scheme Optimization

With the intent to improve the optimization engine of the Madeleine communication library, we have started a new and more formal study of communication optimization algorithms. Considering the programming model of Madeleine which is message-passing oriented with incremental message building/extraction capabilities, one of the recurrent problem in the field is to decide, for a given message slice (let's call it a *packet*), whether it should better be delayed or rather sent immediately.

The former choice (delaying) would allow the aggregation of this packet with an (hypothetical) subsequent packet for the same message. If this choice is successful –a subsequent packet for the same message is packed afterward *and* aggregation of both packets happens to be possible– this choice could lead to better overall performance. On the contrary, if this choice is unsuccessful, it would cause a performance degradation. The properties of the later choice (sending immediately) are the opposite.

In either cases, making the wrong choice incurs a penalty, so, choosing the right way for each packet is important. By definition, the relevance of a choice for a given packet depends on the upcoming of the application in the near future of the expected packing operation. We therefore proposed to explore what optimizations could be derived from having the application anticipately give some information about its upcoming communication schemes.

Analysing a simplified version (with infinite buffers) of the case where aggregation requires an explicit copy into an intermediate buffer, we obtained some preliminary interesting results about aggregation optimality. In particular, considering three packets A, B, C of a given message with the following property: $0 < len(A) < len(B) < len(C)$, and the following notations: $X + Y$ for the sequentialization of packets X and Y and (XY) for the aggregation of packets X and Y, we showed that if $(AC) + B$ is more efficient than $(AB) + C$, then (ABC) is more efficient than $(AC) + B$. This result means that if a packet of a given message is elected for aggregation, any shorter packet for the same message should also be elected for aggregation.

6.4. Performance analysis of multithreaded programs

We have proposed a new approach [29][14] to analyze performance of multithreaded programs which use a hybrid thread scheduler (i.e. a user-level scheduler on top of a kernel-level one). It is based on the offline analysis of traces (sequence of events recorded at run time). The idea is to collect simultaneously two independent traces: one within the kernel and the other in user space. Both traces are sequences of records stamped using the processor's timing registers (incremented at each clock tick). We used the *Fast Kernel Traces* (FKT [44]) library developped by Robert RUSSELL (UNH, USA) for Linux to generate kernel traces. We followed a similar design for our *Fast User Traces* library that operates in user space.

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

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

Once kernel and user traces have been collected and merged, our super-traces can be translated into the Pajé [46] format, so that all events will be graphically represented. Furthermore, Pajé has very interesting features that allows us to display the trace at any scale, to select some kinds of events to show/hide or to change the way the events are represented. This simplifies the analysis of the behavior and performance of hybrid multithreaded programs.

7. Contracts and Grants with Industry

7.1. PhD thesis co-supervised with CEA/DAM

3 years, 2004-2006

We did set up a collaboration with the CEA/DAM (French Atomic Energy Commission, Pierre LECA and Hervé JOURDREN, Bruyère le Chatel) on the support of nuclear simulation programs (adaptive mesh) on large clusters of SMP (thousands of processors) and on Itanium2-based NUMA machines. Marc PERACHE (a former student of Raymond NAMYST at ENS-Lyon) completed his Master project at CEA/DAM dealing with the implementation of a Hydrodynamics Simulation application on top of PM^2. He has started a PhD thesis granted by the CEA under the co-supervising of Hervé JOURDREN and Raymond NAMYST (start in september 2003).

7.2. Contract between INRIA and CEA/DAM

1 year, 2004-2005

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

8. Other Grants and Activities

8.1. Grid'5000 Ministry Grant

3 years, 2003-2005

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

8.2. "Masse de données" Ministry Grant

3 years, 2003-2006.

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

8.3. CNRS Specific Action

1 year, 2004.

Raymond NAMYST has coordinated a national "CNRS specific action" on "Grid Programming Methodologies: what directions for future research?" By gathering all the main French actors in Grid Computing

(Algorithms, Runtime Systems, Networks, etc.), the goal was to explore the actual trends in the many related research topics and to try to bring out a “programming methodology” everyone would agree with.

9. Dissemination

9.1. Committees

Olivier AUMAGE was part of the *High-Performance Conference* (HIPC 2004) program committee in the systems software track.

Raymond NAMYST is part of the *RenPar 2005* program committee.

9.2. Invitations

Guillaume MERCIER was invited for two weeks by William GROPP’s group in Argonne National Laboratory (Chicago, USA) in order to present his work on a multiprotocol implementation of MPI. This team leads the development of the MPICH software which is the most popular existing MPI implementation. As a result of this visit, Guillaume was appointed as a post-doc within the team for a one-year term.

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

Raymond NAMYST has been invited to join a french mission visiting Japan research teams working on the topic Grid and Cluster computing. This mission was co-organized by Japanese National Institute of Informatics (NII), INRIA and French Embassy in Japan. Raymond NAMYST gave a talk at AIST (National Institute of Advanced Industrial Science and Technology) on “MPICH-Madeleine: A fast Multiprocotol implementation of MPI”.

9.3. Reviews

Olivier AUMAGE was involved in the paper reviewing process of the Transaction on Parallel and Distributed Systems IEEE journal, the ICS2004 Workshop on COmponent Models and Systems for Grid Applications and the Cluster 2004 conference.

Raymond NAMYST has reviewed 5 PhD Thesis (Rennes, Grenoble, Amiens, Lyon) from october 2003 to december 2004.

9.4. Seminars

Olivier AUMAGE gave a seminar on “Distributed Computing, the Communication Point of View” at the CEA (French Atomic Energy Commission).

Raymond NAMYST has been invited to give a seminar on “Advanced multitheading techniques” at the CEA (Bordeaux).

9.5. Expertise

Olivier AUMAGE was involved as an expert in the ACI-Grid project selection process for the Grid 5000 call.

9.6. Teaching

Olivier AUMAGE gave a course on “Network Architecture and Related Systems” in the Master of Science at the University Bordeaux 1.

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

10. Bibliography

Major publications by the team in recent years

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

[2] O. AUMAGE. *Madeleine : une interface de communication performante et portable pour exploiter les interconnexions hétérogènes de grappes.*, 154 pages, Thèse de Doctorat, spécialité informatique, École normale supérieure de Lyon, 46, allée d'Italie, 69364 Lyon cedex 07, France, September 2002.

[3] O. AUMAGE, L. BOUGÉ, A. DENIS, L. EYRAUD, J.-F. MÉHAUT, G. MERCIER, R. NAMYST, L. PRYLLI. *A Portable and Efficient Communication Library for High-Performance Cluster Computing (extended version)*, in "Cluster Computing", Special Issue: Selected Papers from the IEEE Cluster 2000 Conference. Extended version of ., vol. 5, nº 1, January 2002, p. 43-54, http://dept-info.labri.fr/~aumage/Runtime/Publis/Download/AumBouDenEyrMehMerNamPry01CC.ps.gz.

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

[5] O. AUMAGE, L. BOUGÉ, J.-F. MÉHAUT, R. NAMYST. *Madeleine II: A Portable and Efficient Communication Library for High-Performance Cluster Computing*, in "Parallel Computing", vol. 28, nº 4, April 2002, p. 607–626.

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

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

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

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

[10] V. DANJEAN, R. NAMYST, R. RUSSELL. *Linux Kernel Activations to Support Multithreading*, in "Proc. 18th IASTED International Conference on Applied Informatics (AI 2000), Innsbruck, Austria", IASTED, February 2000, p. 718-723, http://dept-info.labri.fr/~aumage/Runtime/Publis/Download/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://dept-info.labri.fr/~aumage/Runtime/Publis/Download/NamystHDR.pdf.

Doctoral dissertations and Habilitation theses

[12] V. DANJEAN. *Contribution à l'élaboration d'ordonnanceurs de processus légers performants et portables pour architectures multiprocesseurs*, 156 pages, Thèse de Doctorat, spécialité informatique, École normale supérieure de Lyon, 46, allée d'Italie, 69364 Lyon cedex 07, France, December 2004, http://dept-info.labri.fr/~danjean/publications.html#Dan04Thesis.

[13] G. MERCIER. *Communications à hautes performances portables en environnements hiérarchiques, hétérogènes et dynamiques*, 168 pages, Thèse de Doctorat, spécialité informatique, Université de Bordeaux 1, Domaine Universitaire, 351 Cours de la libération, 33405 Talence Cedex, December 2004.

Articles in referred journals and book chapters

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

Publications in Conferences and Workshops

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

[16] A. DENIS, O. AUMAGE, R. HOFMAN, K. VERSTOEP, T. KIELMANN, H. BAL. *Wide-Area Communication for Grids: An Integrated Solution to Connectivity, Performance and Security Problems*, in "Proc. of the Thirteenth IEEE International Symposium on High-Performance Distributed Computing (HPDC'13), Honolulu, Hawaii", 10 pages, IEEE, June 2004, http://dept-info.labri.fr/~aumage/Publications/Download/hpdc2004.html.

Bibliography in notes

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

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

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

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

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

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

[23] 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.

[24] R. BHOEDJANG, T. RUHL, H. BAL. *LFC: A Communication Substrate for Myrinet*, 1998, http://citeseer.nj.nec.com/bhoedjang98lfc.html.

[25] 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.

[26] J. BRIAT, B. P. I. GINZBURG. *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.

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

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

[29] V. DANJEAN. *Mécanismes de traces efficaces pour programmes multithreadés*, in "Actes des Rencontres francophones du parallélisme (RenPar 15), La Colle sur Loup (France)", October 2003.

[30] V. DANJEAN, R. NAMYST. *Controling Kernel Scheduling from User Space: an Approach to Enhancing Applications' Reactivity to I/O Events*, in "Proceedings of the 2003 International Conference on High Performance Computing (HiPC '03), Hyderabad, India", December 2003, http://dept-info.labri.fr/~aumage/Runtime/Publis/Download/DanNam03HIPC.pdf.

[31] A. DENIS, C. P?REZ, T. PRIOL. *PadicoTM: An Open Integration Framework for Communication Middleware and Runtimes*, in "Future Generation Computer Systems", vol. 19, 2003, p. 575–585.

[32] 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.

[33] 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.

[34] 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.

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

[36] 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.

[37] 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.

[38] 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.

[39] 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.

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

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

[42] 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.

[43] 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.

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

[45] 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.

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