INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team Runtime

Efficient Runtime Systems for Parallel Architectures

Futurs

THEME 1A

Table of contents

1. Team **1**

2. Overall Objectives **1**

2.1. Designing Efficient Runtime Systems 1

2.2. Meeting the Needs of Programming Environments and Applications 2

3. Scientific Foundations **2**

3.1. Runtime Systems Evolution 2

3.2. Current Trends 3

4. Application Domains **4**

4.1. Panorama 4

5. Software **6**

5.1. PM^2 and μPM^2 6

5.2. Madeleine 7

5.3. Marcel 7

5.4. LinuxActivations 7

5.5. MPICH/MADIII 8

6. New Results **8**

6.1. A Fast implementation of MPI on clusters of clusters 8

6.2. Performance analysis of multithreaded programs 9

6.3. Impact of fault tolerance mechanisms on communication performance 9

7. Contracts and Grants with Industry **9**

7.1. Alcatel/INRIA 9

7.2. CEA/DAM 10

8. Other Grants and Activities **10**

8.1. Grid'5000 Ministry Grant 10

8.2. "ACI GRID" Ministry Grant 10

8.3. "Masse de données" Ministry Grant 10

8.4. CNRS Specific Action 10

8.5. "Grid 2002" Ministry Grant + INRIA New Investigation Grant 2003 10

8.6. NSF/INRIA 11

10. Bibliography **11**

1. Team

Head of the team

Raymond Namyst [Professor, Université Bordeaux 1, LaBRI]

Staff members

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

Research scientists (partner)

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

Ph.D. students

Vincent Danjean [Ministry Grant, LIP]
Guillaume Mercier [Ministry Grant, LaBRI]
Marc Perache [CEA Grant]

2. Overall Objectives

2.1. Designing Efficient Runtime Systems

Key words: *parallel*, *distributed*, *runtime*, *environment*, *heterogeneity*, *SMT*.

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 technology 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

Key words: *parallel*, *distributed*, *runtime*, *environment*, *heterogeneity*, *SMT*.

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 in solve difficult problems (e.g. study of the introduction of quality of service schemes within thread scheduling, with the future INRIA Action SuperGé [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

Key words: *parallel*, *distributed*, *cluster*, *environment*, *library*, *communication*, *multithreading*.

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 [18], 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 [33] for communication handling and POSIX-threads [36] for multi-threading management. Environments such as Athapascan [19], Chant [32] or PM^2 [35] 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 [26], Panda [39], PM^2 [35]) have recently 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 [38], GM [31], FM [37] or LFC [17] 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 [21][30] for the Athapascan-1 [19] environment, Panda [39] for the Orca [15] compiler, PM [41] for the SCore environment, PM^2 [35] 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 [10]. 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 [28], a computational grid-enabled multi-application environment (PadicoTM [25]) and even a multi-network version of the MPICH [7][11] library.

3.2. Current Trends

Key words: *parallel*, *distributed*, *cluster*, *environment*, *library*, *communication*, *multithreading*.

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 [34] and PACX-MPI [16]. 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 [22] 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 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* [13] 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

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

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 solution 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 improove 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 [24]).

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 [29]). Several studies have shown since then that complex execution models such as those integrating multi-threading and communication (e.g. Nexus [27][26], Athapascan [19], PM2 [35], MPI+OpenMP [20]), 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 Quadrix 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. PM^2 and μPM^2

The first released version of the PM^2 (*Parallel Multithreaded Machine*) programming environment was developed in Lille in 1995 by the GOAL Team. At that time, PM^2 was a programming environment providing an execution model especially designed for irregular parallel applications. It was principally used by research groups led by Catherine ROUCAIROL (Versailles), Jean ROMAN (Bordeaux), Denis CAROMEL (Nice) and Luc BOUGÉ (Lyon). As the group focused its research work on low-level runtime systems for clusters of SMP workstations, PM^2 gradually became a software suite composed of several libraries, the most famous being the MARCEL thread library and the MADELEINE communication library.

Recently, we have designed the first version of a low-level generic runtime system, called μPM^2, integrating thread management, communication handling and a basic distributed memory management. Its interface is limited to a remote procedure call communication mechanism and classical primitives for multithread

management. Its implementation is to some extent a kind of software glue between the MARCEL/MADELEINE libraries and an iso-address memory management component. The PM^2 environment has completely been rewritten on top of this runtime system.

Currently, the PM^2 software suite is composed of roughly 140 000 lines of C code. It is available over a wide range of architectures (Intel x86, Alpha, PowerPC, etc.), systems (Linux, Solaris, Aix, Irix, Windows 2000, etc.), and network interfaces (BIP/Myrinet, GM/Murinet, SISCI/SCI, VIA/Ethernet, SBP/Ethernet, TCP, UDP and MPI). This suite is distributed under the GPL license and can be downloaded from the http://dept-info.labri.fr/~namyst/runtime web site. This software has also been registered at the *Agence de Protection des Programmes* (Program Protection Agency). It is used by several French or foreign research projects: University of New Hampshire (P. HATCHER), Berlin (F. MUELLER), Bordeaux (J. ROMAN, Scalapplix INRIA project), Rennes (T. PRIOL, Paris INRIA project) and Besançon (J. BAHI).

It is also used for teaching in Nice (F. BAUDE) and Versailles (V.D. CUNG).

5.2. 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/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.

5.3. 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. 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.

5.4. 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 a extension of the *Scheduler Activations* model proposed by Anderson [14]. *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.

5.5. MPICH/MADIII

MPICH/MadIII is an implementation of the MPI standard (*Message Passing Interface*) which is widely used for the development of parallel scientific applications. This software derives from the popular MPI implementation called MPICH (*MPI CHameleon*) which features the property that a new adaptation above a networking technology can easily be developed thanks to a layered architecture. In practice, it is only a matter of implementing a software module (a *MPICH device*) and the upper-level layers do not have to be modified. But in our case MPICH/MadIII is merely a device implemented above the MadeleineIII communication library. Some modifications to the MPICH outermost layers had to be made. The result is that MPICH/MadIII is a multithreaded MPI implementation that is able to efficiently take advantage of heterogeneous configurations (network-wisely). In particular, it becomes possible to exploit the information with regard to the hierarchical nature of a configuration.

The multithreaded architecture of MPICH/MadIII provides a high level of performance. Some experiments have demonstrated the good behaviour of MPICH/MadIII, since this multi-networks implementation shows performance levels that are similar to those obtained by implementations tailored to a specific networking technology.

This implementation is freely available at the following URL: http://dept-info.labri.fr/~mercier/mpi.html. Updates are made on a regular basis and in particular, we report quickly the most important MPICH updates within our own software (the current version is 1.2.5). A brief documentation concerning the software's use is available at the same URL.

MPICH/MadIII is used within the PADICO environment (Paris INRIA project, Rennes) and is evaluated by Professor Rehm's research group (Chemnitz, Germany). The code is developed, maintained and updated by Guillaume MERCIER.

6. New Results

6.1. A Fast implementation of MPI on clusters of clusters

We have designed a fast and heterogeneous implementation of MPICH on top of our Madeleine communication library. This implementation of MPI features a multi-protocol ability. It is also able to take advantage of heterogeneous architectures efficiently, unlike other solutions such as MPICH-G2 [34] or PACX-MPI [16] that limit themselves to using TCP/IP for inter-cluster communication. Moreover, the performance on homogeneous clusters is within the same order of magnitude as the one achieved by implementations that have been specifically designed for a single class of high-performance networks, such as SCI. See http://dept-info.labri.fr/~mercier/mpi.html.

An exhaustive performance comparison between MPICH/Madeleine and PACX-MPI has been carried out by the team led by Professor Wolfgang Rehm (Chemnitz). The goal was to evaluate the relevance of the MPICH/Madeleine design on top of high performance clusters of clusters (CoCs). The results are very good and validate our approach. We worked with Wolfgang'team to analyze the obtained results and to improve the behavior of MPICH/Madeleine on some benchmarks. We are currently writing a paper together about these experiments.

6.2. Performance analysis of multithreaded programs

We have proposed a new approach [12] 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 (incremeted at each clock tick). We used the *Fast Kernel Traces* (FKT [40]) 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 all the missing information has been computed for each event (processor ID, kernel thread ID, user thread ID, etc.)

We are currently developping a translator to convert our super-traces into the Pajé [23] format, so that it will be possible to use the powerful Pajé graphical tool to analyze the behavior and performance of hybrid multithreaded programs.

6.3. Impact of fault tolerance mechanisms on communication performance

In collaboration with Alcatel and the PARIS INRIA project, we designed a version of our Madeleine communication library that can tolerate failures, either at the network level or at the processor level. This version allows an application to continue after node/link crash. The overall mechanism is not transparent: the crash is notified to the application so that further communication attempts with the defective node can be avoided. When the node/link gets operational again, the node is reintegrated to the configuration (*hot plug*). The first part of our study was to find out the level at which the failure detection mechanisms should better be implemented. We showed that the best level depends on the underlying network driver capabilities and we did illustrate this fact on the Myrinet and the SCI networks.

During the second part of the study, we did extend Madeleine to support dynamic configurations. However, the implementation of the corresponding internal mechanisms is costly in the general case, especially over high speed system area networks. In particular, one problem is related to the cost of the extra polling operations needed to accept the arrival of a possible newcomer node. We thus designed, in cooperation with Alcatel, a specific version of Madeleine that is only able to reintegrate some previously faulty nodes (new nodes are not accepted). As expected, the corresponding implementation exhibits very good performance: the overhead of fault tolerance extra operations is neglectible.

This version of Madeleine is used as a basis for the implementation of the MOME fault-tolerant distributed shared memory subsystem (developped within the PARIS INRIA project) for high performance clusters.

7. Contracts and Grants with Industry

7.1. Alcatel/INRIA

2 years, 2002-2003

Raymond NAMYST has initiated a collaboration between the ReMaP and PARIS (Yvon JEGOU et Christine MORIN) INRIA projects and the Alcatel company. The project aims at building a high performance software bus for parallel internet routers made of clusters of smaller routers. Our work consists in designing an extension of MADELEINE in order to provide fault tolerance properties (the application should continue after a node crash, and the corresponding node should be reintegrated easily (*hot plug*) after local recovery.

7.2. CEA/DAM

We are in the process of setting 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).

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. Four local research teams are involved in this project. Raymond NAMYST is the local coordinator of Grid'5000.

8.2. "ACI GRID" Ministry Grant

2 years, 2002-2003.

We are involved in a Cooperative Research Initiative (ACI in French) on the *Globalization of Data and Computing Resources* (GRID). The project is named RMI (led by Christian PEREZ, IRISA) and focuses on the support of object-based distributed applications on computational grids. More precisely, the main goal is to design a multi-applications platform (PadicoTM) able to take advantage of modern networking hardware in a transparent manner (communication multiplexing). The implementation of PadicoTM is built on top of our μPM^2 runtime system. This initiative is a good opportunity for us to evaluate the suitability of μPM^2 within a real life Grid context.

8.3. "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.

8.4. CNRS Specific Action

1 year, 2004.

Raymond NAMYST coordinates 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 is to explore the actual trends in the many related research topics and to try to bring out a "programming methodology" everyone would agree with.

8.5. "Grid 2002" Ministry Grant + INRIA New Investigation Grant 2003

1 year, 2003

Christian PEREZ (IRISA, Rennes), Jacques BAHI (LIFC, Besançon) and Raymond NAMYST (LaBRI, Bordeaux) have proposed to start a study about "*Asynchronous Iterative Algorithms Using Variable Reliability Network Protocols*". This work is supported by both the ACI GRID 2002 and the ARC INRIA 2003.

8.6. NSF/INRIA

Contract with UNH since 1999.
Luc BOUGÉ and Raymond NAMYST have initiated a collaboration with the group of Philip HATCHER (UNH, USA) in 1999. The project (C*IT) has been supported by INRIA and NSF (for 2 years), was centered around the use of distributed multithreaded runtimes for the support of parallel applications developed using a high-level parallel language (C*, HPF). The project was successful and the collaboration was thus extended by an additional 2-years period. The second project was aiming at designing a runtime system for a distributed Java virtual machine for clusters. Within this framework, we are also working with Robert RUSSELL on the design of communication libraries for high speed networks as well as on new software profiling tools for Linux and PM^2 (*Fast Kernel Traces* and *Fast User Traces* tools).

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 », volume 27, October, 2001, pages 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.*. Thèse de Doctorat, spécialité informatique, École normale supérieure de Lyon, 46, allée d'Italie, 69364 Lyon cedex 07, France, September, 2002, 154 pages.

[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 », number 1, volume 5, January, 2002, pages 43-54, http://dept-info.labri.fr/~namyst/runtime/biblio/Aumage/AumBouDenEyrMehMerNamPry01CC.ps.gz, Special Issue: Selected Papers from the IEEE Cluster 2000 Conference. Extended version of ..

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

[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 », number 4, volume 28, April, 2002, pages 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) », Held in conjunction with IPDPS 2001, pages 86, San Francisco, April, 2001, http://dept-info.labri.fr/~namyst/runtime/biblio/Aumage/AumEyrNam00HCW2001.ps.gz, Extended proceedings in electronic form only.

[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) », IEEE, pages 51, San Francisco, April, 2001, http://dept-info.labri.fr/~namyst/runtime/biblio/Aumage/AumMerNam01IPDPS2001.ps.gz, Extended proceedings in electronic form only..

[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) », IFIP WG 10.3 and IEEE, pages 418-425, Paris, France, October, 1998, ftp://ftp.ens-lyon.fr/pub/LIP/Rapports/RR/RR1998/RR1998-43.ps.Z.

[9] V. DANJEAN, R. NAMYST, R. RUSSELL. *Linux Kernel Activations to Support Multithreading.* in « Proc. 18th IASTED International Conference on Applied Informatics (AI 2000) », IASTED, pages 718-723, Innsbruck, Austria, February, 2000, http://dept-info.labri.fr/~namyst/runtime/biblio/Danjean/DanNamRus00IASTED.ps.gz.

[10] 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/~namyst/runtime/biblio/Namyst/NamystHDR.pdf.

Publications in Conferences and Workshops

[11] 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) », IEEE, pages 26–35, Tokyo, May, 2003.

[12] 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.

[13] 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/~namyst/runtime/biblio/Danjean/DanNam03HIPC.pdf.

Bibliography in notes

[14] 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 », number 1, volume 10, February, 1992, pages 53-79.

[15] H. BAL, F. KAASHOEK, A. TANENBAUM. *ORCA: A language for parallel programming of distributed systems.* in « IEEE Transactions on Software Engineering », number 3, volume 18, Mar, 1992, pages 190-205.

[16] 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 », series Lecture Notes in Computer Science, volume 1332, Springer Verlag, M. BUBACK, J. DONGARRA, J. WASNIEWSKI, editors, pages 75-83, Cracow, Pologne, novembre, 1997.

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

[18] 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, pages 91-96, April, 1994.

[19] J. BRIAT, B. P. I. GINZBURG. *Athapascan Runtime : Efficiency for Irregular Problems.* in « Proceedings of the Euro-Par '97 Conference », series Lecture Notes in Computer Science, volume 1300, Springer Verlag, pages 590–599, Passau, Germany, août, 1997.

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

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

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

[23] 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.

[24] A. DENIS, C. P?REZ, T. PRIOL. *PadicoTM: An Open Integration Framework for Communication Middleware and Runtimes.* in « Future Generation Computer Systems », volume 19, 2003, pages 575–585.

[25] 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) », IEEE Computer Society, pages 144-151, Berlin, Germany, May, 2002, http://www.irisa.fr/paris/Biblio/Papers/Denis/DenPerPri02CCGRID.ps.

[26] 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 », volume 40, 1997, pages 35–48.

[27] I. FOSTER, C. KESSELMAN, S. TUECKE. *The Nexus approach to integrating multithreading and communication.* in « Journal of Parallel and Distributed Computing », volume 37, 1996, pages 70-82.

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

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

[30] 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.

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

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

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

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

[35] 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.

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

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

[38] 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) », series Lecture Notes in Computer Science, volume 1388, Held in conjunction with IPPS/SPDP 1998. IEEE, Springer-Verlag, pages 472-485, Orlando, USA, mars, 1998.

[39] 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 », pages 1477-1488, Sunnyvale, CA, August, 1996.

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

[41] 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) », series Lecture Notes in Computer Science, volume 1225, Springer Verlag, pages 708-717, Avril, 1997.