INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Runtime

Efficient Runtime Systems for Parallel Architectures

Futurs

THEME NUM

Table of contents

1. Team

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

Administrative assistant
Corinne Brisset [Project Assistant, until Sep. 30st 2006]
Sylvie Embolla [Project Assistant, from Sep. 1st 2006]
Brigitte Larue-Bourdon [Project Assistant]

Staff members
Olivier Aumage [Research Associate (CR) Inria]
Alexandre Denis [Research Associate (CR) Inria]
Brice Goglin [Research Associate (CR) Inria, from Oct. 1st 2006]
Guillaume Mercier [Assistant Professor, ENSEIRB, LaBRI, from Sep. 1st 2006]
Pierre-André Wacrenier [Assistant Professor, Université Bordeaux 1, LaBRI]

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

Research engineers
Christophe Frézier [Associate Engineer, INRIA]
Nathalie Furmento [Research Engineer, CNRS]

Ph.D. students
Elisabeth Brunet [Regional Grant, LaBRI]
Marc Pérache [CEA Grant]
Samuel Thibault [Ministry of Research and Technology Grant, LaBRI]
François Trahay [Ministry of Research and Technology Grant, LaBRI]

2. Overall Objectives

2.1. Designing Efficient Runtime Systems

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

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

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

The research project centers on three main challenges:

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

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

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: *NUMA*, *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 Mescal project).

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 project and 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.

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 [40], 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 [54] for multi-threading management. Environments such as Athapascan [41], Chant [52] or PM^2 [53] 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 [46], Panda [57], PM^2 [53]) 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 [56], GM [32], MX [35], FM [55] or LFC [39] 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 [43], [51] for the Athapascan-1 [41] environment, Panda [57] for the Orca [37] compiler, PM [58] for the SCore environment, PM^2 [53] 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 [49], a high-performance communication framework for grids (PadicoTM [45]) and even a multi-network version of the MPICH [7], [6] library.

3.2. Current Trends

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

Even though several problems still remain unresolved so far (communication schemes optimization, reactivity to I/O events), we now have at our disposal efficient runtime systems that *do* efficiently exploit small-scale homogeneous clusters. However, the problem of mastering large-scale, hierarchical and potentially heterogeneous configurations (that is, clusters of clusters) still has to be tackled. Such configurations bring in many new problems, such as high-performance message routing in a heterogeneous context, dynamic configuration management (fault-tolerance). There are two interesting proposals in the particular case of heterogeneous clusters of clusters, namely MPICH-G2 [34] and PACX-MPI [38]. 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 [31] follows an approach similar to ours through trying to provide an efficient runtime system for such architectures. A preliminary contact has already been established in order to set up a collaboration about this topic.

Besides, even if the few aforementioned *success stories* demonstrate that current runtime systems actually improve both portability and performance of parallel environments, a lot of progress still has to be made with regards to the optimal use of runtime systems features by the higher level software layers. Those upper layers still tend to use them as mere "black-boxes". More precisely, we think that the expertise accumulated by a runtime system designer should be formalized and then transferred to the upper layers in a systematic fashion (code analysis, specialization). To our knowledge, no such work exists in the field of parallel runtime systems to date.

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

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

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

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

Although it was originally designed to support programming environments dedicated to parallel computing (PM^2, MPI, etc.), our software is currently successfully used in the implementation of middleware such as object brokers (OmniORB, INRIA Paris project) or Java Virtual Machines (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 (GridExplorer cluster at IDRIS, Grid5000 project, clusters at CEA/DAM, etc.), in Europe (cluster DAS-3 in the Netherlands, etc.) or in the rest of the world (the US TeraGrid Project, etc.). As a general rule, these clusters are built out of a homogeneous set of PCs interconnected with a fast system area network (SAN). Such SAN solutions (Myrinet, Quadrics, Infiniband, etc.) typically provide 10Gb/s throughput and a couple microseconds latency. Commonly found computing node characteristics range from off-the-shelf PCs to high-end symmetrical multiprocessor (SMP) or non-uniform memory access (NUMA) machines with a large amount of memory accessed through high-performance chipsets with multiple I/O buses or switches.

This increasing worldwide expansion of parallel architectures is actually driven by the ever growing need for computing power needed by numerous real-life applications. These demanding applications need to handle large amounts of data (e.g. 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 [2] that the problem is actually much deeper than a mere matter of implementation optimization. It is mandatory to rethink the existing *interfaces* from a higher, semantic point of view. The general idea is that the interface should be designed to let the application "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 [44]).

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 [50]). Several studies have shown since then that complex execution models such as those integrating multi-threading and communication (e.g. Nexus [47], [46], Athapascan [41], PM2 [53], MPI+OpenMP [42]), 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 544 16-cores 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 or a remote DMA based network such as Infiniband. Similar observations can be made about I/O (the Infiniband technology's interoperability with Fiberchannel and iSCSI is bringing in new issues), processors and other peripheral 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: Quadrics, Myrinet, SCI, Ethernet or VIA and runs on many architectures: Linux/IA32, Linux/IA64, Linux/x86-64, Linux/Alpha, Linux/Sparc, Linux/PowerPC, Solaris/Sparc, Solaris/IA32, AIX/PowerPC, WindowsNT/IA32. A version enabled to work *inside* the Linux kernel is also available. The current production version of Madeleine is version 3. Madeleine and its external software roughly consists in 71 000 lines of code and 154 files. This library, distributed as part of the PM^2 software is developed and maintained by Olivier AUMAGE, Elisabeth BRUNET, Nathalie FURMENTO and Raymond NAMYST. The software is freely available under the terms of the GNU General Public License version 2 at the following URL: http://runtime.futurs.inria.fr/madeleine/.

5.2. NewMadeleine

The design and development of the NEWMADELEINE communication library started during Q1, 2006. NEWMADELEINE is complete redesign and rewrite of Madeleine. The new architecture aims at enabling the use of a much wider range of communication flow optimization techniques. It is entirely modular: the request scheduler itself is interchangeable, allowing experimentations with multiple approaches or on multiple issues with regard to processing communication flows. In particular we implemented an optimizing scheduler called SchedOpt. SchedOpt targets applications with irregular, multi-flow communication schemes such as found in the increasingly common application conglomerates made of multiple programming environments and coupled pieces of code, for instance. ScheOpt itself is easily extensible through the concepts of optimization *strategies* (*what* to optimize for, what the optimization goal is) expressed in terms of *tactics* (*how* to optimize to reach the optimization goal). Tactics themselves are made of basic communication flows operations such as packet merging or reordering. The NEWMADELEINE software consists in 35 000 new lines of code. NEWMADELEINE is available on various networking technologies: Quadrics, Myrinet, SCI and Ethernet. This library, distributed as part of the PM^2 software is developed and maintained by Olivier AUMAGE, Elisabeth BRUNET, Nathalie FURMENTO and Raymond NAMYST. The software is freely available under the terms of the GNU General Public License version 2 at the following URL: http://runtime.futurs.inria.fr/newmadeleine/.

5.3. Marcel

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

The most important feature of Marcel is its scheduler, named *BubbleSched*. BubbleSched is a framework that allows scheduling experts to implement and experiment with powerful user-level thread schedulers (http://runtime.futurs.inria.fr/marcel/bubblesched.php). It is based on high-level abstractions called *bubbles*. The application describes affinities between the threads it launches by encapsulating them into nested bubbles (those which work on the same data for instance). BulleSched then allows to implement various advanced

bubble schedulers that distribute bubbles (and hence threads) over the hierarchy of the computer so as to beneficiate from cache effects and avoid NUMA factor penalties as much as possible. A trace of the scheduling events can be recorded and used after execution for generating an animated movie showing a replay of the execution: how bubbles and threads were created, how they got distributed over the machine, how they eventually got scheduled on processors, etc. End users may hence easily try and tune various bubble schedulers for their applications, and select the most suited one.

Marcel provides a POSIX-compliant interface and a set of original extensions. It can also be compiled to provide ABI-compabiblity with NTPL threads under Linux, so that multithreaded applications can use Marcel without being recompiled. This permits for instance to run Java applications with Marcel. All these *flavors* are based on the same thread management core kernel and are specialized at compilation time.

While keeping the possibility to be run autonomously, Marcel combines perfectly with 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.

This library is developed and maintained by Samuel THIBAULT. The software is freely available under the terms of the GNU General Public License version 2 at the following URL: http://runtime.futurs.inria.fr/marcel/.

5.4. Mad-MPI

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

Mad-MPI also implements some optimizations mechanisms for derived datatypes [48]. MPI derived datatypes deal with noncontiguous memory locations. The advanced optimizations of NEWMADELEINE allowing to reorder packets lead to a significant gain when sending and receiving data based on derived datatypes.

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

5.5. MPICH2-Nemesis

Nemesis is a new generic communication subsystem which goal is to address the communication needs of a wide range of programming tools and environments for clusters and parallel architectures. It has been designed to yield very low latency and high bandwidth, especially for intranode communication. Nemesis has successfully been integrated within the next-generation MPI implementation MPICH2 as a communication channel [9].

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

High-performance networks are also supported within MPICH2-Nemesis as modules and all major protocols and technologies are currently supported such as Myrinet, Quadrics, Infiniband, SCTP and TCP. The development of such modules has lead to some contacts with the companies developing those various networks in order to improve the current code and the overall performance of MPICH2-Nemesis. A NEWMADELEINE module is also available.

This work has been initiated by Darius BUNTINAS and Guillaume MERCIER during his postdoctoral stay at the ARGONNE NATIONAL LABORATORY (ANL). Guillaume being still an active member of the MPICH2's development and support team we have regular contacts with ARGONNE NATIONAL LABORATORY regarding the device version of Nemesis. Since our NEWMADELEINE communication library and the MPICH2 Nemesis channel possess common goals and design features, we believe that this is a very good opportunity to share experience about these two pieces of software and to influence altogether the design of MPICH2. For instance, the NEWMADELEINE network module developed for Nemesis has already shown some limitations in the MPICH2-Nemesis's design that will be addressed consequently.

MPICH2-Nemesis, a joint development between the ANL and the Runtime Project, is downloadble on the MPICH2 ANL website and is developed and maintained by Darius BUNTINAS and Guillaume MERCIER.

5.6. MPICH-Madeleine

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

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

MPICH-Madeleine is based on the popular MPICH implementation (currently MPICH1 1.2.7) but the communication engine concept is implementation-independent and could be adapted to other free implementations, such as YAMPII or Open MPI. MPICH-Madeleine supports Fortran and C++ MPI applications, as well as a wide variety of architectures and compilers.

A collaboration has been started with Olivier GLÜCK and Ludovic HABLOT at the LIP to improve performances of MPICH-Madeleine for long distance network communications.

Actually, since the new generation of MPICH (MPICH2) is now available, and given the fact that the team member Guillaume MERCIER is involved in the development of MPICH2, we plan to introduce a port of MPICH2 on top of Marcel and Madeleine in the future that would be the successor of MPICH-Madeleine.

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

5.7. PadicoTM

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

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

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

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

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

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

5.8. MPC

The MPC library proposes a programming environment allowing to design efficient parallel programs on top of clusters of multiprocessors. It features a message passing programming model centered around collective communications and synchronizations, and provides load balancing facilities. The programming interface is close to the MPI standard to allow easy migration from MPI applications to MPC.

MPC executes tasks using threads, allowing to overload CPUs if needed (i.e. using more thread than CPUs). This technique enhances communication overlaping and maximizes caches effects (using smaller data domains per thread). The thread scheduler used by MPC is optimized for "overloaded" situations. Another original aspect is that collective communication operations are integrated within the scheduler. Thus, these operations are aware of thread scheduling decisions and can be optimized accordingly when the number of threads is high.

Regarding inter-processes communication, MPC currently lies on top of the MPI standard but will soon be able to use MADELEINE generic communication library.

This library has been developed in collaboration with CEA-DAM. Its main contributor is Marc PÉRACHE.

5.9. XPaulette

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

XPAULETTE is able to isolate blocking syscalls on dedicated threads so that the whole process isn't suspended. It is actually a portable alternative to the Scheduler Activations model proposed by Anderson [36] and implemented in the LinuxActivations library [10]. By isolating blocking syscalls, it becomes possible to suspend only the thread responsible for the blocking call, while the other threads continue their execution.

MADELEINE and NEWMADELEINE have been ported over XPAULETTE. We also plan to use XPAULETTE inside MARCEL to detect quickly some events such as signals.

François TRAHAY is the main contributor to this piece of software.

6. New Results

6.1. Communication Optimization over High Speed Networks

We have undertaken a complete redesign of the MADELEINE communication engine, now called NEWMADELEINE [21]. NEWMADELEINE introduces fundamental changes in communication request handling and optimizations. Traditionally, communication libraries, being synchronous, tightly link the communication requests to the application workflow, and therefore transmit incoming packets immediately to the lower network layer without any accumulation. On the contrary, NEWMADELEINE keeps accumulating packets in its optimization window while the NICs are busy. As soon as a NIC becomes idle, the optimization window is analyzed so as to create a new ready-to-send packet to be transfered through the card: NICs are exploited at their maximum (they are not overloaded when there is a high demand of transfers and under exploited when there is not) and the communication optimizations are made *just-in-time* so they closely fit the ongoing communication scheme at any given time.

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

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

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

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

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

6.3. Message reordering within the Mad-MPI lighweight implementation of MPI

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

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

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

6.4. Thread Scheduling over Hierarchical Architectures

Exploiting full computational power of current more and more hierarchical multiprocessor machines requires a very careful distribution of threads and data among the underlying non-uniform architecture, so as to minimize the number of remote memory accesses, to favor cache affinities, or to guarantee fast completion of synchronization steps. Unfortunately, most operating systems only provide a poor thread scheduling API that does not allow applications to transmit valuable scheduling hints to the system.

We have proposed to extend classical thread schedulers with high-level abstractions called *Bubbles* [60], [59], which are used to dynamically describe relations betweens threads in order to improve applications' performance in a portable way. Programmers can model the relationships between the threads of their applications using (nested) bubbles, with no particular dependency upon the underlying architecture. The concept of bubbles can be understood as a coset with respect to a specific affinity relation, and bubble nesting expresses refinement of a relation by another one. This lets express relations like data sharing, collective operations, good behavior with regards to co-scheduling on a SMT processor or on a NUMA machine, or more generally a particular scheduling policy need (serialization, preemption, gang scheduling, etc.). The scheduler can use this information to maximize the locality of threads belonging to the same bubble while still trying to keep all the processors busy. This mechanism is generic enough for developing a wide range of schedulers, hence letting programmers try various approaches for distributing bubbles on the machine: simple top-bottom distribution, gang scheduling, work stealing, etc.

Furthermore, we have designed a framework name *BubbleSched* that allows scheduling experts to prototype, experiment and implement user-level bubble-based thread schedulers. It provides a powerful API for dynamically distributing bubbles among the machine in a high-level, portable, and efficient way. Programmers can hence focus on algorithmic issues rather than on nasty technical details. Examples of implementing scheduling strategies have shown how easy this is and how powerful it can be. Non-expert programmers may even try different combinations of existing strategies to schedule the threads of their applications. Of course, such combination may still be difficult from an algorithmic point of view, but with the additional help of a debugger, programmers can really focus on algorithmic issues rather than on gory details.

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

Actually, part of this work was done in collaboration with researchers at the CEA (french Atomic Energy Commission) who have been developing huge HPC applications for a few decades and who are looking for a tool allowing them to transfer their expertise to the underlying runtime system.

This work opens numerous future prospects. In the short term, a generic facility for attributes and statistics on bubbles will be developed as a help for decision: application programmers could even provide their own measuring tools, which the BubbleSched platform would gather according to the bubble hierarchy. Several algorithmic approaches will then be implemented, tested, and tuned for scheduling real applications.

6.5. High-Performance In-Kernel Communication

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

Our Kmadeleine prototype is a version of the MADELEINE communication library enabled to run inside the Linux kernel. It provides the generic API and optimizing layer of Madeleine on top of cluster interconnects such as Myrinet/MX and Quadrics/QsNET. It has already been experimented with the iSCSI stack written by the team of R. Russell at the InterOperability Laboratory (IOL) from the University of New Hampshire, which is an implementation of the Internet SCSI protocol. R. Russell is involved in the standardisation process of iSCSI at IETF. Kmadeleine is also currently being experimented as a potential high performance communication layer for the Lustre distributed file system.

6.6. Flexible network communications on computational grids

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

PadicoTM is a communication framework that decouples middleware systems from the actual networking resource. The applications are thus able to transparently and efficiently utilize any kind of middleware (either parallel or distributed) on any network. Since year 2005, PadicoTM is built with true *software components*. Communications methods, network access, and paradigm adaptation are implemented as components. Thus, using components as building blocks, the user may assemble communication stacks following the needs of the application and the requirements imposed by the network infrastructures. We were able to add various communication methods as new components, mainly communication methods for wide area networks: various compression filters (ZIP, LZO, BZIP) and flexible socket factories to cross firewalls without compromising security (SSH tunnel, TCP splicing, relaying with dynamic routing).

To control the component assembly process, and to allow advanced communication methods that require negotiation or synchronization between nodes, we have shown that a *control channel* used for bootstrap and out-of-band communication is required. We proposed [25] a novel approach for the management of this control channel as an overlay network that combine security, connectivity, and high performance. The Salomé project (CEA/EDF, http://www.salome-platform.org/) is very interested in these features from PadicoTM and actually employed an intern in collaboration with RUNTIME to finalize the implementation of

the PadicoTM control channel. Salomé investigate the use of PadicoTM to solve network-related problems (mostly connectivity and performance) encountered by Salomé when deploying on a network topology made of supercomputers with high performance internal network and restrictive security policy towards the outside, and standalone visualization workstations. Moreover, we are currently investigating the use of the PadicoTM software component model and control channel infrastructure in NEWMADELEINE to get the same flexibility in configuration and dynamicity at the cluster level as we have at the grid level.

For the first time in 2005, PadicoTM, MADELEINE and MARCEL were developed in the same INRIA project. This led to a better integration that began to be effective in year 2006. Regarding multi-threading, PadicoTM is now able to take benefit from all MARCEL flavors (including NUMA) and from MARCEL network polling service. Moreover, PadicoTM legacy binary support and MARCEL `pthread` have been combined for legacy multi-thread binary support in PadicoTM. On the networking side, PadicoTM is able to utilize directly low-level MADELEINE drivers and generic layer of MADELEINE has been embedded in a PadicoTM component known as "Madico". This enables MADELEINE applications to seamlessly take benefit from PadicoTM communication methods for grids.

Finally, we worked on widening the availability of middleware systems over PadicoTM. The Apache Portable Runtime (APR) and JXTA-C (Sun Microsystems) were ported by members of the PARIS project to enable the JuxMem environment to run over PadicoTM; this will be used in the "LEGO" ANR project. We are currently investigating the use of PadicoTM in the EPSN (http://epsn.gforge.inria.fr) steering software from the Scalapplix project to couple visualisation on a dedicated cluster and simulation running on a computational grid. Moreover, we ported the MPI implementation YAMPII/GridMPI (http://www.gridmpi.org/) from the University of Tokyo over PadicoTM with good results in the context of the NEGST grant (CNRS-JST) with Japan.

6.7. Reactivity to I/O events

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

We showed that, by using a centralized I/O event server, a high level of "reactivity" can be guaranteed even during heavy computing phases. We have designed a scalable architecture for this I/O server named XPAULETTE. By interacting with the thread scheduler, XPAULETTE detects the completion of the communication queries (either by polling the network or waiting for interrupts) and triggers the appropriate callback as soon as possible [30].

PadicoTM and MADELEINE are already using XPAULETTE, making them reactive even when running many computing threads. We are currently porting NEWMADELEINE over XPAULETTE in order to fully benefit from the NEWMADELEINE communication optimizations in multithreaded contexts.

6.8. Efficient collective communications on NUMA machine

Efficient collective communications, such as barriers or reduction operations, are essential in achieving good scalability of parallel computer programs, most notably on large-scale, hierarchical multiprocessor computers. Most collective communication algorithms designed to run with one single process per processor suffer serious performance degradation when used in a multithreaded context. Such performance penalties result from the poor coordination of thread synchronizations and thread scheduling.

As discussed previously, scheduler-aware algorithms are needed to perform efficient collective communications. The algorithm used in MPC is designed to take care of data location. Data location is a key idea to design efficient algorithm on NUMA architectures. Each request to a remote data is subject to the NUMA factor penalty which multiplies the access time to the data by a specific factor. On our test architecture, this factor is equal to 3. This is why, any data structure will be placed close to the processor which needs a regular access to it.

The algorithm uses the concept of "participating processors". Each collective communication involves a set of participating processors. This set is composed of all the virtual processors which include in their running queue, some of the threads that are involved in the collective communication operation. This set can not be known *a priori* due to dynamic thread migration among processors, as a migration may remove or insert virtual processors into the set. Hence the set is created at the initialization of the collective operation and updated accordingly to thread migrations.

In order to maximize data locality, data in our algorithm are distributed among processors allowing virtual processors to decide whether a thread should block or not without interfering with other virtual processors. Each participating processor owns a data structure that contains the number of its threads that are involved in the collective communication, a queue to store the threads which are blocked and some structures used to build the hierarchical evaluation tree between "participating processors".

Our algorithm relies on two extensions to the underlying thread scheduler: the ability to register threads that are blocked due to a collective communication call in a specific queue, and the ability to transfer a specific queue back into the ready queue in a constant time. Of course, if special scheduler extensions featuring special polling features may increase performances if multiple collective communications occur .

Thread registration in the collective communication blocked queue is performed by modifying the function that is used to pick up the current thread from the ready queue to place it in the blocked queue by a function that is used to pick up the current thread from the ready queue to place it in a user-specified queue. This means that a thread may specify the queue used to store the blocked threads.

Thread wake-up needs the ability to merge the thread queue into the ready queue. This operation is quite simple as all the threads in this queue are in a running state (no verification is needed). We also know that all these threads were running on the virtual processor performing the wake-up call when the blocking call was performed (as no other virtual processor can wake up these threads). This merge operation does not need to be performed into a critical section (if multiple running queues are used) and does not need any thread status verification.

We did implement all the mechanisms described within the two-level thread scheduler of MPC. The results obtained with micro-benchmarks and with real applications show that our approach achieves excellent performance, even when the number of threads exceeds the number of processors.

7. Contracts and Grants with Industry

7.1. PhD thesis co-supervised with CEA/DAM

3 years, 2004-2006

We did set up a collaboration with the CEA/DAM (French Atomic Energy Commission, Pierre LECA and Hervé JOURDREN, Bruyère le Chatel) on the support of nuclear simulation programs (adaptive mesh) on large clusters of SMP (thousands of processors) and on Itanium2-based NUMA machines. In September 2003, Marc PÉRACHE has started a PhD thesis granted by the CEA under the co-supervising of Hervé JOURDREN and Raymond NAMYST. He worked on thread scheduling over clusters of SMP and defended his PhD in October 2006.

7.2. Contract between INRIA and Myricom

We are setting up a collaboration with Myricom, Inc. (US Company building high-speed interconnect hardware and software) regarding the design and implementation of a message passing protocol on top of generic Ethernet interfaces. We initially focus on keeping a design close to Myricom's Myrinet Express software suite which is known to provide high-performance MPI for parallel applications, and try to adapt this design to the restricted features that generic Ethernet interfaces provide. This contract is expected to begin at the end of year 2006 or the beginning of 2007.

8. Other Grants and Activities

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

3 years, 2005-2007

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

LEGO Grid infrastructure and middleware is now a mature technology; however, grid programming and use is still a very complex task, because each middleware only take into account one paradigm: MPI, RPC, workflow, master-slave, shared data, ... Thus a new model must be learnt for each kind of application. Current high performance computing application are becoming multi-paradigm. The aim of LEGO is to propose and to implement a multi-paradigm programming model (component, shared data, master-slave, workflow) comprising state of the art grid programming. It will use efficient scheduling, deployment, and an adequate communication layer. The model will be designed to cope with three kinds of classical high performance computing applications: climate modeling, astronomy simulation, and matrix computation.

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

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

8.2. Grid’5000 Ministry Grant

3 years, 2003-2006

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

8.3. "Masse de données" Ministry Grant

3 years, 2003-2006.

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

8.4. NEGST (NExt Grid Systems and Techniques)

3 years, 2006-2009.

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

It aims at promoting collaborations between Japan and France on grid computing technology. Following successful France-Japan workshops hosted by CNRS in Paris and NEREGI/NII in Tokyo, three important novel research issues have been identified: 1) Instant Grid and virtualization of grid computing resources, 2) Grid Metrics and 3) Grid Interoperability and Applications. The objective is to accelerate the intensive works of several research teams in these subjects in both countries. An international testbed including the French Grid5000 project and its Japanese counterpart NEREGI will be use to demonstrate and validate systems, software and applications.

9. Dissemination

9.1. Schools

Raymond NAMYST has been invited to give a lecture ($7 \times 1h30$) on *Efficient Programming on Parallel Architectures* at the CEA-EDF-INRIA summer school (june 2006) devoted to *Optimizing Scientific Applications on New Generation High Performance Hardware*.

9.2. Committees

Raymond NAMYST was co-chair of the EXPGRID (Experimental Grid testbeds for the assessment of large-scale distributed applications and tools) workshop held in conjunction with the 15th International Symposium on High Performance Distributed Computing (HPDC-15).

Raymond NAMYST was part of the *RenPar 2006 Conference* program committee.

9.3. Invitations

Guillaume MERCIER, as a member of the MPICH2 development team and as a NEMESIS designer, visited theRUNTIME project for a 6-months term (from March until August 2006). He began to work on the NEMESIS/Madeleine 4 convergence, and did develop the MX, Elan and NewMadeleine networks modules for Nemesis.

9.4. Reviews

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

Brice GOGLIN was involved in the paper reviewing process of the SuperComputing Conference (SC|06) and the International Conference on Cluster Computing (Cluster 2006).

Raymond NAMYST has reviewed 3 PhD Thesis during year 2006.

9.5. Seminars

Nathalie FURMENTO gave a seminar about MPICH-Madeleine at the final meeting of the “ACI Masse de données” Data Grid eXplorer (November 2006).

Nathalie FURMENTO and Raymond NAMYST gave a seminar about MPICH-Madeleine and Mad-MPI at the LaBRI (Dec. 2006).

Alexandre DENIS and Olivier AUMAGE gave a seminar about Differentiated High Performance Communication at LIP/ENS Lyon (Feb. 2006).

Olivier AUMAGE gave a seminar about the NewMadeleine communication library at IRISA/Rennes (Sep. 2006).

Alexandre DENIS gave a seminar on managing complex grid network topologies at IRISA/Rennes (Sep. 2006)

Alexandre DENIS gave a seminar about Network communication on grids with PadicoTM at the final meeting of the “ACI Masse de données” Data Grid eXplorer (November 2006).

9.6. Teaching

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

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

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

10. Bibliography

Major publications by the team in recent years

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

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

[3] O. AUMAGE, L. BOUGÉ, L. EYRAUD, R. NAMYST. *Calcul réparti à grande échelle*, F. 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.

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

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

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

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

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

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

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

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

Year Publications

Doctoral dissertations and Habilitation theses

[12] M. PÉRACHE. *Contribution à l'élaboration d'environnements de programmation dédiés au calcul scientifique hautes performances*, 141 pages, Thèse de Doctorat, spécialité informatique, CEA/DAM Île de France, Université de Bordeaux 1, Domaine Universitaire, 351 Cours de la libération, 33405 Talence Cedex, October 2006.

Articles in refereed journals and book chapters

[13] R. BOLZE, F. CAPPELLO, E. CARON, M. DAYDÉ, F. DESPREZ, E. JEANNOT, Y. JÉGOU, S. LANTERI, J. LEDUC, N. MELAB, G. MORNET, R. NAMYST, P. PRIMET, B. QUETIER, O. RICHARD, E.-G. TALBI, T. IRENA. *Grid'5000: a large scale and highly reconfigurable experimental Grid testbed.*, in "International Journal of High Performance Computing Applications", vol. 20, n^{o} 4, November 2006, p. 481-494.

[14] E. BRUNET. *NewMadeleine : ordonnancement et optimisation de schémas de communication haute performance (version étendue de Perpi'06).*, in "Technique et Science Informatiques", Submitted, 2007.

[15] D. BUNTINAS, G. MERCIER, W. GROPP. *Implementation and Evaluation of Shared-Memory Communication and Synchronization Operations in MPICH2 using the Nemesis Communication Subsystem*, in "Parallel Computing, Special Issue on EuroPVM/MPI 2006", Submitted, 2007, 15.

[16] C. MORIN, A. DENIS, R. NAMYST, O. AUMAGE, R. LOTTIAUX. *Encyclopédie de l'informatique et des systèmes d'information*, chap. Des réseaux de calculateurs aux grilles de calcul, n° ISBN : 2-7117-4846-4, Vuibert, December 2006.

[17] S. THIBAULT, R. NAMYST, P.-A. WACRENIER. *BubbleSched: construire son propre ordonnanceur de threads pour machines multiprocesseurs hiérarchiques*, in "Technique et Science Informatiques", Submitted, 2007.

Publications in Conferences and Workshops

[18] O. AUMAGE, E. BRUNET, N. FURMENTO, R. NAMYST. *NewMadeleine: a fast communication scheduling engine for high performance networks*, in "CAC 2007: Workshop on Communication Architecture for Clusters, Long Beach, California, USA", Submitted, March 2007.

[19] O. AUMAGE, E. BRUNET, G. MERCIER, R. NAMYST. *High-Performance Multi-Rail Support with the NewMadeleine Communication Library*, in "HCW 2007: the Sixteenth International Heterogeneity in Computing Workshop, Long Beach, California, USA", To appear, March 2007.

[20] E. BRUNET, O. AUMAGE, R. NAMYST. *Short Paper : Dynamic Optimization of Communications over High Speed Networks*, in "HPDC-15, The 15th IEEE International Symposium on High Performance Distributed Computing, Paris", June 2006, http://hal.inria.fr/inria-00110773/.

[21] E. BRUNET. *NewMadeleine : ordonnancement et optimisation de schémas de communication haute performance.*, in "Renpar'17, Rencontres Francophones du Parallélisme, Canet en Rousillon / France", October 2006, http://hal.inria.fr/inria-00110766/.

[22] D. BUNTINAS, G. MERCIER, W. GROPP. *Data Transfer in a SMP System: Study and Application to MPI*, in "Proc. 34th International Conference on Parallel Processing(ICPP 2006), Colombus, Ohio", August 2006.

[23] D. BUNTINAS, G. MERCIER, W. GROPP. *Design and Evaluation of Nemesis: a Scalable, Low-Latency, Message-Passing Communication Subsystem*, in "Proc. 6th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2006), Singapore", Held in conjunction with IEEE Computer Society and ACM, May 2006.

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

[25] A. DENIS. *Meta-communications in Component-based Communnication Frameworks for Grids*, in "HPC Grid programming Environments and COmponents (HPC-GECO), workshop held in conjunction with HPDC-15, Paris", Selected for publication in Cluster Computing, IEEE, June 2006, p. 77–84, http://hal.inria.fr/inria-00090066.

[26] A. DENIS, N. FURMENTO, R. NAMYST. *Efficient runtime systems for grids*, in "EXPGRID, Experimental Grid testbeds for the assessment of large-scale distributed applications and tools, Workshop held in conjunction

with the 15th International Symposium on High Performance Distributed Computing (HPDC-15), Paris", Poster, June 2006, http://runtime.futurs.inria.fr/Download/Publis/poster_expgrid2006.pdf.

[27] S. THIBAULT. *BubbleSched : construire son propre ordonnanceur de threads pour machines multiprocesseurs hirarchiques*, in "17ème Rencontres Francophones du Parallélisme, Canet en Roussillon / France", ACM/ASF - Université de Perpignan, 10 2006, http://hal.inria.fr/inria-00108984/.

Internal Reports

[28] É. BRUNET. *Support d'ordonnancement et d'optimisation automatisés des communications pour les réseaux hautes performances*, Research Report, n° 5641, INRIA, July 2005, http://hal.inria.fr/inria-00070366.

Miscellaneous

[29] A. DENIS, N. FURMENTO, G. MERCIER, R. NAMYST. *ACI Grid'5000, site de BordeauxPaRISTIC : Panorama des Recherches Incitatives en STIC, Nancy*, Poster, November 2006, http://paristic.loria.fr/content/grille_calcul_intensif/posters/Bordeaux.pdf.

[30] F. TRAHAY. *Gestion de la réactivité des communications réseau*, Mémoire de DEA, Université Bordeaux 1, June 2006, http://runtime.futurs.inria.fr/Download/Publis/Tra06Memoire.pdf.

References in notes

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

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

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

[36] 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.

[37] 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.

[38] 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.

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

[40] 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.

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

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

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

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

[45] 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.

[46] 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.

[47] 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.

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

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

[50] 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.

[51] 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.

[52] 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.

[53] 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.

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

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

[56] 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.

[57] 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.

[58] 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.

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

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