INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Runtime

Efficient Runtime Systems for Parallel Architectures

Bordeaux - Sud-Ouest

THEME NUM

Table of contents

RUNTIME is an INRIA Project-Team joint with University of Bordeaux (UB1) and CNRS (LaBRI, UMR 5800)

1. Team

Research Scientist

Olivier Aumage [INRIA, Research Associate (CR)]
Alexandre Denis [INRIA, Research Associate (CR)]
Brice Goglin [INRIA, Research Associate (CR)]

Faculty Member

Raymond Namyst [University Bordeaux 1, Team Leader, Professor (Pr), HdR]
Marie-Christine Counilh [University Bordeaux 1, Assistant Professor (MCF)]
Guillaume Mercier [ENSEIRB, Assistant Professor (MdC)]
Samuel Thibault [University Bordeaux 1, Assistant Professor (MdC), since Sep. 2008]
Pierre-André Wacrenier [Université Bordeaux 1, Assistant Professor (MdC)]

Technical Staff

Ludovic Courtès [INRIA, Research Engineer (IR), since Apr. 2008]
Nathalie Furmento [CNRS, Research Engineer (IR)]
Cécile Romo-Glinos [INRIA, Associate Engineer]

PhD Student

Cédric Augonnet [University Bordeaux 1, École Normale Supérieure de Lyon grant, since Sep. 2008]
Elisabeth Brunet [University Bordeaux 1, Conseil Régional d'Aquitaine grant, till Dec. 2008]
François Broquedis [University Bordeaux 1, MESR grant, since Sep. 2007]
Jérôme Clet-Ortega [University Bordeaux 1, MESR grant, since Sep. 2007]
François Diakhaté [CEA, since Sep. 2007]
Mathieu Faverge [INRIA, ANR grant, since Nov. 2006]
Stéphanie Moreaud [INRIA, ANR grant, since Sep. 2007]
François Trahay [University Bordeaux 1, MENRT grant, since Apr. 2006]

Visiting Scientist

Darius Buntinas [Argonne National Laboratory, Illinois, USA, June 2008]
David Goodell [Argonne National Laboratory, Illinois, USA, Oct.-Nov. 2008]
Éric Maisonnave [European Centre for Research and Advanced Training in Scientific Computation, France, Apr. 2008]
Vincent Pichon [École Normale Supérieure de Lyon, Apr. 2008]

Administrative Assistant

Sylvie Embolla [INRIA]

2. Overall Objectives

2.1. Designing Efficient Runtime Systems

Keywords: *HPC*, *MPI*, *NUMA*, *OpenMP* , *SMP*, *environment*, *heterogeneity*, *high-speed networks*, *multicore*, *parallel*, *protocols*, *runtime*, *scheduling*, *thread*.

The RUNTIME research project takes place within the context of high-performance computing. It seeks to explore the design, the implementation and the evaluation of novel mechanisms needed by **runtime systems** for parallel computers. *Runtime systems* are intermediate software layers providing parallel programming environments with specific functionalities left unaddressed by the underlying operating system. Runtime systems can thus be seen as functional extensions of operating systems, but the boundary between them is rather fuzzy since runtime systems may actually contain specific extensions/enhancements to the underlying operating system (e.g. extensions to the OS thread scheduler). The increasing complexity of modern parallel hardware, making it more and more necessary to postpone essential decisions and actions (scheduling, optimizations) at run time, emphasizes the role of runtime systems.

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

The originality of the runtime group lies in the fact that we address all these issues following a global approach, so as to propose complementary solutions to problems which may not seem to be linked at first sight. We actually realized, for instance, that we could greatly improve our communication optimization techniques by increasing the functionalities of the underlying core thread scheduler. This illustrates why most of our research efforts have consisted in cross-studying differents topics, and have led to co-designing many software.

Our research project centers on three main directions:

Mastering large, hierarchical multiprocessor machines With the beginning of the new century, computer makers have initiated a long term move of integrating more and more processing units, as an answer to the frequency wall hit by the technology. This integration cannot be made in a basic, planar scheme beyond a couple of processing units for scalability reasons. Instead, vendors have to resort to organize those processing units following some hierarchical structuration scheme. A level in the hierarchy is then materialized by small groups of units sharing some common local cache or memory bank. Memory accesses outside the locality of the group are still possible thanks to bus-level consistency mechanisms but are significantly more expensive than local accesses, which, by definition, characterizes NUMA architectures.

Thus, the task scheduler must feed an increasing number of processing units with work to execute and data to process while keeping the rate of penalized memory accesses as low as possible. False sharing, ping-pong effects, data vs task locality mismatches, and even task vs task locality mismatches between tightly synchronizing activities are examples of the numerous sources of overhead that may arise if threads and data are not distributed properly by the scheduler. To avoid these pitfalls, the scheduler therefore needs accurate information both about the computing platform layout it is running on and about the structure and activities relationships of the application it is scheduling.

As quoted by Gao *et al.* [63], we believe it is important to expose domain-specific knowledge semantics to the various software components in order to organize computation according to the application and architecture. Indeed, the whole software stack, from the application to the scheduler, should be involved in the parallelizing, scheduling and locality adaptation decisions by providing useful information to the other components. Unfortunately, most operating systems only provide a poor scheduling API that does not allow applications to transmit valuable *hints* to the system.

This is why we investigate new approaches in the design of thread schedulers, focusing on high-level abstractions to both model hierarchical architectures and describe the structure of applications' parallelism. In particular, we have introduced the *bubble* scheduling concept [20],[15], [76] that helps to structure relations between threads in a way that can be efficiently exploited by the underlying

thread scheduler. *Bubbles* express the inherent parallel structure of multithreaded applications: they are abstractions for grouping threads which "work together" in a recursive way.

Aside from greedily invading all these new cores, demanding HPC applications now throw excited glances at the appealling computing power left unharvested inside the graphical processing units. A strong demand is arising from the application programmers to be given means to access this power without bearing an unaffordable burden on the portability side. Efforts have already be made by the community in this respect but the tools provided still are rather close to the hardware, if not to the metal. Hence, we decided to launch some investigations on addressing this issue. In particular, we have designed a programming environment named STARPU that enables the programmer to offload tasks onto such heterogeneous processing units and gives that programmer tools to fit tasks to processing units capability, tools to efficiently manage data moves to and from the offloading hardware and handles the scheduling of such tasks all in an abstracted, portable manner.

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

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

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

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

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

Finally, as most of our work proposed is intended to be used as a foundation for environments and programming tools exploiting large scale, high performance computing platforms, we definitely need to address the numerous scalability issues related to the huge number of cores and the deep hierarchy of memory, I/O and communication links.

2.2. Highlights of the year

- OPEN-MX is under experimentation at ANL as the networking layer for the PVFS2 storage stack in the BlueGene/P system (ranked #3 in June 2008 Top500 with 450 Tflop/s).

3. Scientific Foundations

3.1. Runtime Systems Evolution

Keywords: *cluster*, *communication*, *distributed*, *environment*, *library*, *multicore*, *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 [53], 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 [45] for communication handling and POSIX-threads [70] for multithreading management. Environments such as Athapascan [54], Chant [67] or PM^2 [69] well illustrate this trend. Even though such approaches are still widespread, they do suffer from numerous limitations related to *functional* incompatibilities between the various software components (decreased performance) and even to *implementation* incompatibilities (e.g. thread-unsafe libraries).

In the past, several proposals (Nexus [60], Panda [73], PM^2 [69]) have shown that a better approach lies in the design of runtime systems that provide a tight integration of communication handling, I/O and multithreading management. In order to get closer to an optimal solution, those runtime systems often exploit very low-level libraries (e.g. BIP [72], GM [44], MX [47], FM [71] or LFC [52] 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 [56], [66] for the Athapascan-1 [54] environment, Panda [73] for the Orca [49] compiler, PM [74] for the SCore environment, PM^2 [69] 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 [13]. μPM^2 integrates communication handling and multithreading 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 [64], a high-performance communication framework for grids (PadicoTM [59]) and even a multinetwork version of the MPICH [8], [7] library.

3.2. Current Trends

Keywords: *cluster*, *communication*, *distributed*, *environment*, *library*, *multicore*, *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 [46] and PACX-MPI [51]. 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 [43] follows an approach similar to ours through trying to provide an efficient runtime system for such architectures.

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

The members of the RUNTIME project-team 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 focused on addressing the complexity of efficiently scheduling irregular parallel applications on hierarchical machines like SMPs of multicore chips and NUMA machines. Our high-level *bubble* scheduling concept enables parallel programming environments to express the irregularity of the application's parallelism, and we are able to to automatically detect the structure of the target machine. We thus explored how to dynamically schedule the irregular nested sets of threads on hierarchical machines, the key challenge being to schedule related threads as closely as possible in order to benefit from cache effects and avoid NUMA penalties. We also explored how to improve the transfer of scheduling hints from the programming environment to the runtime system, and we showed that these permit to achieve better computation efficiency.

Concerning the use of *heterogeneous* multicore architectures (e.g. specialized coprocessors and accelerators), one of our concerns has been to design a portable API to offload tasks onto various accerelator technologies. Additionaly, we paid attention to fully exploit each of the different computational resources at the same time by properly mapping tasks in a dynamic manner by the means of scheduling policies. Along with portability, we provide expressive abstractions, such as a distributed shared-memory library that makes it possible to manipulate data accross heterogeneous multicore architectures in a high-level fashion.

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

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

4. Application Domains

4.1. Panorama

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

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

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

This quest for cost-effective solutions gave rise to a much wider diffusion of parallel computing architectures, illustrated by the large and steadily growing number of academic and industrial laboratories now equipped with clusters, in France (GridExplorer cluster at IDRIS, Grid'5000 project, clusters at CEA/DAM, etc.), in Europe (cluster DAS-3 in the Netherlands, etc.) or in the rest of the world (the US TeraGrid Project, etc.). As a general rule, these clusters are built out of a homogeneous set of PCs interconnected with a fast system area network (SAN). Such SAN solutions (Myrinet, Quadrics, Infiniband, etc.) typically provide 10Gb/s throughput and a couple of microseconds latency. Commonly found computing node characteristics range from off-the-shelf PCs to high-end symmetrical multiprocessor (SMP) or non-uniform memory access (NUMA) machines with a large amount of memory accessed through high-performance chipsets with multiple I/O buses or switches.

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

First of all, each cluster is almost unique in the world regarding its processor/network combination. This simple fact makes it very difficult to design a runtime system that achieves both portability and efficiency on a wide range of clusters. Moreover, few softwares are actually able to keep up with the technological evolution; while others involve a huge amount of work to adapt the code due to an unsuitable internal design. We showed in [2] that the problem is actually much deeper than a mere matter of implementation optimization. It is mandatory to rethink the existing *interfaces* from a higher, semantic point of view. The general idea is that the interface should be designed to let the application "expresses its requirements". This set of requirements can then be mapped efficiently by the runtime system onto the underlying hardware according to its characteristics. This way the runtime system can guarantee *performance portability*. The design of such a runtime system interface should therefore begin with a thorough analysis of target applications' *specific* requirements.

Moreover, and beside semantic constraints, runtime systems should also address an increasing number of functional needs, such as fault tolerant behavior or dynamically resizable computing sessions. In addition, more specific needs should also be taken into account, for example the need for multiple independent logical communication channels in modular applications or multiparadigm environments (e.g. PadicoTM [58]).

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 [65]). Several studies have shown since then that complex execution models such as those integrating multithreading and communication (e.g. Nexus [61], [60], Athapascan [54], PM2 [69], MPI+OpenMP [55]), 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. multiprocessors featuring the *Simultaneous Multi-Threading* technology). The challenge here is to express those characteristics as part of the execution model provided by the runtime system without compromising applications portability and efficiency on "regular" clusters.
- The widespread availability of clusters in laboratories combined with the general need for processing power usually leads to interconnect two or more clusters by a fast link to build a *cluster of clusters*. Obviously, it is likely that these interconnected clusters will be different with respect to their processor/network pair. Consequently, the interconnected clusters should *not* be considered as *merged* into one big cluster. Therefore, and beside a larger aggregated computing potential, this operation results in the addition of another level in the cluster hierarchy.
- A current approach tends to increase the number of nodes within the clusters (the CEA/DAM, for instance, owns a cluster of 544 16-cores nodes linked with a Quadrics network). These large clusters lead to a set of new issues to be addressed by runtime systems. For instance, a lot of low-level communication libraries do not allow users to establish point-to-point connections between the whole set of nodes of a given configuration when the number of nodes grows beyond several dozens. It should be emphasized that this limitation is often due to physical factors of network interconnection cards (NICs), such as on-board memory amount, etc. Therefore, communication systems bypassing the constraint of a node being able to perform efficient communications only within a small neighbourhood have to be designed and implemented.
- Finally, each new communication technology brings its own new programming model. Typically, programming over a memory-mapped network such as SCI is completely different from programming over a message passing oriented network such as Myrinet or a remote DMA based network such as Infiniband. Similar observations can be made about I/O (the Infiniband technology's interoperability with Fiberchannel and iSCSI is bringing in new issues), processors and other peripheral technologies. Runtime systems should consequently be openly designed from the very beginning not only to deal with such a constantly evolving set of technologies but also to be able to integrate easily and to exploit thoroughly existing as well as forthcoming *idioms*.

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

5. Software

5.1. Marcel

The Marcel library. MARCEL is the thread library of the PM^2 software suite. MARCEL features a two-level thread scheduler (also called N:M scheduler) that schedules user-level threads on top of a set of kernel threads (usually one kernel thread per logical processor). Such a model 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 large number of threads and to efficiently exploit hierarchical architectures (e.g. multicore chips, NUMA machines).

BubbleSched. At the core of the MARCEL architecture, we find the BUBBLESCHED scheduling framework (http://runtime.bordeaux.inria.fr/bubblesched/). Computing platforms are becoming increasingly hierarchical. As an answer to this trend, BUBBLESCHED provides the *application* programmer with high level constructs, called *bubbles*, to let him express the affinity between the various activities of his application: the application describes affinities between the threads it launches by encapsulating them into nested bubbles (threads which work on the same data for instance), which thus form a tree of the hierarchical activity structure of the application. BUBBLESCHED then provides the *scheduler* programmer with a *hierarchical runqueues tree* that represents the detected hierarchical platform and a *toolkit* of basic operations to dynamically map the hierarchical activity structure (that is, the threads) of the application (as modeled by the bubbles) onto the hierachical platform in a suitably tailored scheduling. That permits to benefit from cache effects as much as possible, or favor bandwidth, or favor load balancing, or whatever fits the application best.

Scheduling algorithms. Multiple ad-hoc scheduling policies have already been implemented and are available as part of the MARCEL software distribution. The *Spread scheduler* [14], for instance, is designed to deal with multilevel applications. It consists in a recursive function that greedily distributes a hierarchy of bubbles over the hierarchy of runqueues according to their computation load (either explicitly specified by the programmer, or inferred from the number of threads). The *Affinity scheduler* [22], as another example, deals with very irregular applications by enforcing a distribution of threads that maximizes the locality of threads belonging to the same *bubble*, and uses a NUMA-aware work stealing strategy when load balancing is needed [68]. Other affinities may also be taken into account, such as between communication threads and physical I/O devices.

Topology detection. MARCEL integrates a topology detection subsystem to generate a consistent, uniform representation of the platform topology it is running on, out of pieces of information gathered from the operating system through various sources (`/proc` and `/sys` on Linux for instance). This internal representation is then used to build the hierarchy of runqueues used by the BUBBLESCHED scheduler framework. It also provides the schedulers with quantitative numbers on memory caches and banks that are attached to these runqueues.

Traces. 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.

Beyond fine grain multithreading. The evolution of computing resources towards an increasingly dense integration of cores implies a similarly, increasingly tough quest for bits of parallelism in user applications. However, at such a fine grain of parallelism, the inherent cost of parallelism management per grain is not neglectible anymore when compared to the grains themselves. Even a user-level thread, which is a rather light object by the standards of the threading bestiary, is too expensive: the cost of allocating a stack for a new user-level thread plus the cost of context switching to this new thread become prohibitive in this context. MARCEL thus provides a *seed* construct which can be seen as a precursor of thread. Creating a thread seed merely instructs MARCEL to book the request to run a given task at some point in the future. This operation does not reserve any resource except from the information about the task to be run. It is only when the time comes to actually run the seed that MARCEL attempts to reuse the resources and the context of another, dying thread, significantly saving management costs when succeeding while keeping the penalty low otherwise.

POSIX compliance. In addition to a set of original extensions, MARCEL provides a POSIX-compliant interface which thus permits to take advantage of it by just recompiling unmodified applications or parallel programming environments. MARCEL can also be compiled to provide ABI-compatibility with NTPL threads under Linux, so that multithreaded applications and environments can use MARCEL without even being recompiled. This permits for instance to run Java applications with MARCEL and allowed us to build the FORESTGOMP OPENMP environment described in Section 5.2. All these *flavors* are based on the same thread management core kernel and are specialized at compilation time.

Relationships with RUNTIME Team's other softwares. While keeping the possibility to be run autonomously, MARCEL combines perfectly with NEWMADELEINE and PIOMAN, improving reactivity to

communications. Specific softwares matching the needs of PM^2 are also included, allowing thread migration between homogeneous machines.

Distribution. The reference software development branch of the MARCEL software consists in 87 000 lines of code. This library is developed and maintained by Samuel THIBAULT and Olivier AUMAGE. The software is freely available under the terms of the GNU General Public License version 2 at the following URL: http://runtime.bordeaux.inria.fr/marcel/.

5.2. ForestGOMP

FORESTGOMP is an OPENMP environment based on both the GNU OPENMP run-time[1] support and the MARCEL thread library (described in Section 5.1). It is designed to schedule efficiently nested sets of threads (derived from nested parallel sections) over hierarchical architectures. Indeed, approaching the theoretical performance of hierarchical multicore machines requires a very careful distribution of threads and data over the underlying architecture so as to minimize cache misses and NUMA penalties.

Language extensions such as OPENMP provide opportunities to improve thread scheduling over hierarchical architectures in a portable way. High-level parallel constructs such as OPENMP's provide useful hints about a program's parallel structure. These hints can be leveraged by the underlying scheduler and run-time system to improve thread scheduling, e.g., by taking into account cache affinity among threads, or affinity between threads and data.

The FORESTGOMP runtime generates nested MARCEL bubbles each time an OPENMP parallel section is encountered, thereby grouping threads sharing common data. Topology-aware scheduling policies implemented by BUBBLESCHED can then be used. Such policies dynamically map bubbles onto the various levels of the underlying hierarchical architecture, such as the *Spread* and *Affinity* policies described in Section 5.2. FORESTGOMP allowed us to validate the BUBBLESCHED approach with highly irregular, fine grain, divide-and-conquer parallel applications. This approach has proved to yield noticeably better performance for a variety of parallel applications.

The experiments we conducted validate this approach in terms of ease of development for the programmer, flexibility, portability and performance. It is a way for experts to build and combine complex scheduling strategies that take characteristics of the application into account. Application programmers get a greater control on scheduling of their OPENMP programs while completely avoiding intrusive and non-portable changes to their applications.

François BROQUEDIS and Ludovic COURTÈS are the main contributors to this piece of software. It is freely available under the terms of the GNU General Public License version 2 at the following URL: http://runtime.bordeaux.inria.fr/forestgomp/

5.3. NewMadeleine

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

The NEWMADELEINE SchedOpt optimizing scheduler aims at enabling the use of a much wider range of communication flow optimization techniques such as packet reordering or cross-flow packet aggregation [18]. SchedOpt targets applications with irregular, multiflow communication schemes such as found in the increasingly common application conglomerates made of multiple programming environments and coupled pieces of code, for instance. It is designed to be programmable through the concepts of optimization *strategies* (*what*

[1] GNU OPENMP (GOMP) is distributed as part of the GNU Compiler Collection (GCC). See http://gcc.gnu.org/projects/gomp/ for more information.

to optimize for, what the optimization goal is) expressed in terms of *tactics* (*how* to optimize to reach the optimization goal), allowing experimentations with multiple approaches or on multiple issues with regard to processing communication flows. Tactics themselves are made of basic communication flows operations such as packet merging or reordering.

Special purpose strategies have also been developped. A strategy is dedicated to heterogeneous multirail support [24]. A QoS strategy [39] is responsible for differentiated service support: it allows to use distinct optimizations and priorities for distinct communication flows.

NEWMADELEINE has strong relationships established with other software projects in the RUNTIME project-team, each of whose having been the subject of dedicated work during the last year. Indeed, NEWMADELEINE is the direct core communication library of the Mad-MPI [5] and MPICH-Madeleine modules and is being ported as a communication subsystem target for the MPICH2-Nemesis software from Argonne National Laboratory [33]. It is built upon the PadicoTM software component model, and is now the default communication stack for clusters in PadicoTM. It fundamentally relies on the new PIOMAN [16] module and the MARCEL module for asynchronous communication request processing and progression. It now works together with the Fast User Trace module to provide post-mortem communication schemes analysis. And finally it directly depends on the recent work on *Non-Uniform Input-Ouput Access* (NUIOA) [40],[12] when run on non-uniform hierarchical architectures.

The reference software development branch of the NEWMADELEINE software consists in 62 500 lines of code. NEWMADELEINE is available on various networking technologies: Myrinet, Infiniband, Quadrics, SCI and Ethernet. This library, distributed as part of the PM^2 software is developed and maintained by Alexandre DENIS, Elisabeth BRUNET, François TRAHAY 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.bordeaux.inria.fr/newmadeleine/.

5.4. PIOMan

PIOMAN [41] is the event detector server used by the PM^2 software suite. It aims at providing the other software components with a service that can guarantee a predefined level of "reactivity" to I/O events. It is typically used by NEWMADELEINE and PadicoTM to quickly react to network events, such as the arrival of a new packet. PIOMAN is derived from the former MARCEL event server developed by Vincent DANJEAN and thus works closely with MARCEL so as to be triggered upon context switches, processor idleness, etc.

PIOMAN is able to isolate blocking system calls on dedicated threads so that the whole process is not suspended. It is actually a portable alternative to the Scheduler Activations model proposed by Anderson [48] and implemented in the LinuxActivations library [11]. By isolating blocking system calls, it becomes possible to suspend only the thread responsible for the blocking call, while the other threads continue their execution. This mechanism makes PIOMAN reactive even during heavy computing phases.

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

The level of reactivity provided by PIOMAN allows NEWMADELEINE to make communications progress in the background (by making the *rendez-vous* handshake progress for instance) and thus to fully overlap computation and communication [36].

As PIOMAN is able to balance the processing of communication requests accross the whole machine, message submission to the networks can be offloaded on idle core in order to overlap communication and computation even for eager message that require CPU-intensive memory copies [24].

MADELEINE, NEWMADELEINE and PadicoTM have been ported over PIOMAN. Within our collaboration with the Argonne National Laboratory, MPICH2-Nemesis is being ported over PIOMAN [33]. We also plan to use PIOMAN in MPI implementations such as YAMPII (within the collaboration with the University of Tokyo).

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

5.5. Mad-MPI

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

Mad-MPI also implements some optimizations mechanisms for derived datatypes [62]. 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.bordeaux.inria.fr/MadMPI/.

5.6. MPICH2-NewMadeleine

MPICH2-NEWMADELEINE is the successor to the old MPICH-Madeleine implementation that was derived from MPICH 1.2.7. MPICH2-NEWMADELEINE is based on the most recent MPICH2 software (1.0.6pX) and utilizes the NEWMADELEINE communication library for all network communication. As far as intranode communication are concerned, the Nemesis communication subsystem is employed [10]. Nemesis is a generic communication subsystem which goal is to address the communication needs of a wide range of programming tools and environments for clusters and parallel architectures. It has been designed to yield very low latency and high bandwidth, especially for intranode communication.

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

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

This work takes advantage of the Associate Team program in order to make frequent visits in order to coordinate the work.

MPICH2-NEWMADELEINE , a joint development between the ANL and the RUNTIME project-team will soon be available on the RUNTIME website and is developed and maintained by Darius BUNTINAS, Guillaume MERCIER, François TRAHAY and David GOODELL.

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 resources to reach transparent portability and flexibility: the various middleware systems use PadicoTM through a seamless virtualization of networking resources; only PadicoTM itself uses directly the networks.

PadicoTM architecture is based on software components. Puk (the PadicoTM micro-kernel) implements a light-weight high-performance component model that is used to build communication stacks. Typical communications stacks built in PadicoTM follow a three-layer approach. The lowest layer, called the *arbitration layer*, aims at making the access to the resources cooperative rather than competitive. It enables the use of multiple middleware systems atop a single network, as needed by code coupling programming models such as parallel objects or parallel components. This layer is based on PIOMAN to ensure high performance and good interactions between threads and networking. The middle layer, called the *abstraction layer*, decouples the paradigm of the programming interface from the paradigm of the network; for example, it can do dynamic client/server connections over static SPMD networks. The highest level layer, called the *personality layer*, gives several API called "personalities" over the abstractions. It aims at providing the middleware systems with the API they expect. It enables PadicoTM to seamlessly integrate *unmodified* middleware systems.

PadicoTM currently supports most high performance networks (Infiniband, Myrinet, SCI, Quadrics, etc.), communication methods for grids (plain TCP, splicing to cross firewalls, routing, tunneling). Various middleware systems are supported over PadicoTM: various CORBA implementations (omniORB, Mico), popular MPI implementations (MPICH from Argonne – actually, MPICH/PadicoTM is derived from MPICH-Madeleine —, YAMPII from the University of Tokyo, our own Mad-MPI), 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 and François TRAHAY (RUNTIME) with some occasional contributions from Christian PÉREZ (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.bordeaux.inria.fr/PadicoTM/ . It is has been hosted on InriaGForge since mid-2005. PadicoTM is registered at the APP under number IDDN.FR.001.260013.000.S.P.2002.000.10000.

PadicoTM component model is now used in NEWMADELEINE. It is the cornerstone for networking integration in the ARA "LEGO" from the ANR. The EPSN computational steering framework from the SCALAPPLIX project-team uses PadicoTM for its communications. Previously, it has been used by several projects: ACI GRID "RMI", ACI GRID HydroGrid, ACI GRID EPSN, Inria ARC RedGrid, and the European FET project POP.

5.8. Open-MX

The OPEN-MX software stack is a high-performance message passing implementation for any generic Ethernet interface. It was developped within our collaboration with Myricom, Inc. as a part of the move towards the convergence between high-speed interconnects and generic networks. We are thus developing OPEN-MX [28] in collaboration with Myricom, Inc. to expose the raw Ethernet performance at the application level through a pure message passing protocol.

While the goal is similar to the old GAMMA stack [57] or the recent iWarp [50] implementations, OPEN-MX relies on generic hardware and drivers and has been designed for message passing. OPEN-MX is also wire-compatible with Myricom's MX protocol and interface so that any application built for MX may run on any machine without Myricom's hardware and talk other nodes running with or without the native MX stack. OPEN-MX is under experimentation at the Argonne National Laboratory as a networking layer for the PVFS2 parallel file-system on the upcoming BlueGene/P machine in the Argonne laboratory [2]. It will connect BlueGene specific nodes with generic 10 gigabit/s Ethernet boards to generic I/O nodes with Myri-10G running in native MX mode.

[2] ANL's BlueGene/P system is ranked #3 in the June 2008 Top500, with 450 Tflop/s

This work enables the study of high-speed network innovations in the context of generic hardware that has not been designed for such a usage. By using the I/OAT hardware of modern Intel hardware to offload expensive memory copies, OPEN-MX is able to work around the inability of the Ethernet hardware to perform zero-copy receiving [29]. OPEN-MX also enables multiple optimizations of the management of memory buffers in high-speed networks since its requirements on memory pinning are dramatically more simple. OPEN-MX thus proposes an innovating intra-node communication stack that overlaps the expensive memory pinning with offloaded memory copies [31].

OPEN-MX is also an interesting framework for studying next-generation hardware features that could help Ethernet hardware become legacy in the context of high-performance computing. By adding OPEN-MX-aware packet filtering capabilities in the firmware of Myri-10G boards, OPEN-MX is able to benefit from a cache-friendly receive stack that processes all packets on the core that runs their destination process [32].

Brice GOGLIN is the main contributor to OPEN-MX. The software is already composed of more than 31 000 lines of code in the Linux kernel and in user-space. It is freely available under the terms of the GNU General Public License version 2 at the following URL: http://open-mx.org/.

5.9. StarPU

STARPU is a unified runtime system that offers a support for heterogeneous multicore architectures (multicore, GPGPUs, Cell ...). Its goal is to help higher level softwares such as compiler environment or specific high performance libraries to harness the power of those emerging architectures in a high level and portable way.

STARPU differs from most similar projects as it considers all computing resources simultaneously, instead of merely offloading computation onto one or several accelerators. It is organized around three main components : a distributed shared memory that offers a high level interface to manipulate data in a transparent and efficient manner, a unified execution model called “codelets” which models tasks that can be executed on different architectures, and a scheduling engine that makes possible to design scheduling policies which assign tasks to the different computing resources as efficiently as possible.

STARPU has not only demonstrated that it is possible to distribute computations efficiently over a machine’s heterogeneous computing resources, but it has also shown that it is possible to take advantage of the actual heterogeneity of the machine, since parts of applications are better suited to running on multicores while other parts benefit from running on a GPGPU.

Cédric AUGONNET is the main contributor to STARPU which is currently composed of more than 14 000 lines of code. It is freely available under the terms of the GNU General Public License version 2 at the following URL: http://runtime.bordeaux.inria.fr/StarPU/.

6. New Results

6.1. Fine-Grained Multithreading over Modern Computing Architectures

Year 2008 confirmed and strengthend in a spectacular fashion the tendencies initiated over the last few years towards densifying and deepening computing platforms, and resulted in an unprecedented need for multithreading inside applications. We therefore dedicated our effort on the threading front in three major directions: improving our support for NUMA platforms as an answer to platform hierachies *deepening*, improving our support for extra-fine-grained parallelism with the concept of threads *seeds* (see Section 5.1) as an answer to platform computing cores *densifying*, improving our support of the POSIX API to widen the range of applications to benefit from our multithreading software.

On the NUMA side we jointly worked on improving our BUBBLESCHED scheduling framework and the accuracy of our topology detection code. Indeed, 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. Unfortunately, the limited thread scheduling API that most operating systems provide does not allow applications to hint the system scheduler, let alone in a portable way.

In order to address this performance portability issue, we thus proposed to extend the PThread API with high-level abstractions called *Bubbles* [20], [75]. The bubble scheduling concept allows to express the inherent parallel structure of the application: bubbles are abstractions for grouping threads that "work together" (because they share data for instance) in a recursive way as thread trees. On the other side, we model hierarchical machines with a hierarchy of runqueues : each component of each hierarchical level of the machine is represented by one runqueue. Therefore, a tree-based representation of the hardware topology is computed in a portable way out of available system informations and benchmarks. From this point of view, scheduling comes down to dynamically maintaining the mapping of thread trees onto the topology tree.

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. Thus, we both worked on improving the BUBBLESCHED core itself and experimenting with new scheduling algorithms. Part of this work was done in cooperation with the OPENMP work and with the memory migration work in the team (see below). We also added a topology simulation code to help in experimenting with BUBBLESCHED algorithms and in the overall understanding of the NUMA issues.

6.2. Efficient scheduling of OpenMP threads on NUMA machine

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

To solve this problem, we designed "FORESTGOMP", an extension to the GNU OPENMP (GOMP) runtime support that relies on the MARCEL/BUBBLESCHED thread scheduling package already described in Section 5.2. This approach allows programmers to define complex scheduling policies to deal with multilevel, hierarchical and fine-grain parallelism: thanks to this framework, OPENMP is now a good tool to tackle irregular applications on NUMA architectures [14]

In particular, we wrote the AFFINITY bubble-scheduler specifically designed to deal with irregular applications based on a divide and conquer scheme. The FORESTGOMP/*Affinity* approach is running the MPU application about 3 times faster than the GOMP/PTHREAD environment on a 16 cores computer [22], [38]. On the same way, tests performed with the BT-MZ application on the same computer results in a 10.2 speedup for the FORESTGOMP/*spread* approach with nested parallelism, while GOMP/PTHREAD reaches a speedup of 7 and does not get any better result with nested parallelism.

FORESTGOMP is also now able to take thread/memory affinities into account while distributing the load on hierarchical architectures. It relies on the MAMI memory manager to allocate, bind or migrate memory buffers [23].

6.3. High-performance memory migration in Linux

We diagnosed a dramatic performance problem in the migration subsystem of the Linux kernel caused by the quadratic complexity of the model. We then reworked this implementation to restore a linear behavior and thus achieve a constant asymptotic throughput, as well as dramatically reduce its temporary memory consumption. This work enables high-performance memory migration within multithreaded applications on Linux for the first time [27]. These changes[3] will be integrated in the upcoming Linux kernel 2.6.29.

We added a memory manager, MAMI, to our thread library MARCEL. It enables memory management with regard to NUMA nodes. It allows users to allocate memory by specifying a NUMA node, or based on the *first touch* policy, or the *next touch* policy. MAMI also support memory migration between nodes, migration cost evaluation, as well as remote read/write access cost evaluation.

[3] http://lkml.org/lkml/2008/10/13/410

MAMI is used by FORESTGOMP to place threads and memory in a combined manner so as to take memory affinity into account [23].

6.4. Communication Optimization over High Speed Networks

The NEWMADELEINE communication subsystem introduces fundamental changes in communication request handling and optimizations [18], [17]. Traditionally, communication libraries, being synchronous, tightly link the communication requests to the application workflow, and therefore transmit incoming packets immediately to the lower network layer without any accumulation. On the contrary, NEWMADELEINE keeps accumulating packets in its optimization window while the NICs are busy. As soon as a NIC becomes idle, the optimization window is analyzed so as to generate a new ready-to-send request to be transfered through the card: NICs are exploited at their maximum (they are not overloaded when there is a high demand of transfers and under exploited when there is not) and the communication optimizations are made *just-in-time* so they closely fit the ongoing communication scheme at any given time.

When at least one of the multiplexing units becomes idle, an *optimization function* is called to elect the next request to be submitted to each idle unit. In doing so, it may select a packet to be sent from the optimization window, or for instance, synthesize a request out of several packets from that window. A wide panel of arguments may be used as an input to the optimizing function. The optimization function is to be selected among an extensible and programmable set of *strategies*. Each strategy aims at some particular optimizing goal. A strategy is itself made of one or more tactics that apply some elementary optimizing operations selected from the panel of usual operations. In particular we have experimented with multifragment messages with multiple policies of multirail packet balancing on heterogeneous high performance networks [24] and also with providing differentiated quality of service to distinct communication flows [39] through the use of corresponding strategies. NEWMADELEINE showed both its usefulness in conducting such experiments and very good results in terms of latency and bandwidth, while incurring only negligible overhead on basic single packets micro-benchmarks.

NEWMADELEINE is able to efficiently exploit heterogeneous network technologies. We have shown that, by sampling each network during the initialization, it is possible to predict the behavior of each multiplexing unit [24]. As a result, messages can be sent in parallel accross multiple networks. Messages are split with an adaptative split ratio depending on networks capabilities and current load in order to balance the load over the networks.

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.

6.5. Multithreaded Communication Engine

The increase of the number of cores per node in clusters requires changes in the exploitation of high performance networks. The classical MPI approach that consists in running one MPI process per core do not scale because of memory limitations. The use of an hybrid approach mixing MPI and threads seems to be an efficient way to exploit nowadays clusters. Thus, communication libraries have to take into account that applications may be multithreaded.

We designed an efficient thread-safe communication library that allows threads to communicate concurrently. The use of PIOMAN as well as fine-grain locking permits to achieve good performance even on multithreaded environment. We analyzed [37] the impact of various way of ensuring thread-safety on performance.

We showed [41] that, by using a centralized I/O event manager, a high level of "reactivity" can be guaranteed in a portable way even during heavy computation phases. We have designed a scalable architecture for this I/O server named PIOMAN. By interacting with the thread scheduler, PIOMAN detects the completion of the communication queries (either by polling the network or waiting for interrupts) and triggers the appropriate callback as soon as possible. The progression of asynchronous communications in the background can thus be performed efficiently, allowing a full overlap of communication and computation [36].

PadicoTM, MADELEINE and NEWMADELEINE are already using PIOMAN, making them reactive even when many computing threads are running. We are currently developping an enhanced version of NEWMADELEINE that optimizes the communications more efficiently thanks to the multithreading support provided by PIOMAN. By delegating message submission to idle cores, NEWMADELEINE is able to send messages in parallel over several (potentially different) networks interfaces, reducing the transmission duration [24].

6.6. High-performance message passing over generic Ethernet hardware

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

The API compatibility with the native Myrinet Express stack already enables existing parallel application to use OPEN-MX. Indeed, several legacy high-performance layers such MPICH2 or Open MPI run works transparently on top of OPEN-MX with satisfying performance [28]. Also, the PVFS2 storage system has been successfully tested and is under experimentation in the Argonne national laboratory in the context of the BlueGene/P installation.

While being compatible with any legacy Ethernet hardware, OPEN-MX suffers from missing hardware features that help high-speed networks reaching high-performance in clusters. We worked around the inability to perform zero-copy on the receive side by implementing an offloaded copy backend that relies on Intel I/OAT hardware features. This model frees the host from having to perform expensive memory copies and enables linerate performance for OPEN-MX on 10G Ethernet networks [29]. Since I/OAT hardware is now widely available and seems to bring high-performance networking to legacy Ethernet hardware, this result raises the question of whether it is still worth using advanced NIC.

We also showed that OPEN-MX opens a large room for innovative memory management optimizations. Indeed, thanks to OPEN-MX not requiring complex synchronization between the application, the driver and the NIC, we were able to implement an overlapped memory pinning model, causing the expensive pinning overhead to be hidden behind the actual communication time [30]. Moreover, by combining this idea with I/OAT copy offload, we implemented in OPEN-MX a dramatically improved intra-node communication stack. As soon as large messages are involved or processes are not sharing a cache, OPEN-MX now outperforms most existing MPI layers as soon [31].

Finally, OPEN-MX is also an interesting framework for studying next-generation hardware features that could help Ethernet hardware becoming legacy in the context of high-performance computing. We exhibited some cache-inefficiency problems in the OPEN-MX receive stack that are inherited from the Ethernet model. By adding OPEN-MX-aware packet filtering capabilities in the *Multiqueue* firmware of Myri-10G boards, we are able to control the location of the processing of the incoming OPEN-MX traffic. We extended this model by providing an automatic binding facility for user-space applications. This model enables the whole processing of each incoming OPEN-MX packet on the core that runs its target application, causing the overall cache efficiency to improve dramatically [32].

6.7. Scheduling over Heterogeneous Multicore Architectures

In almost every computer nowadays lies a Graphical Processing Unit (GPU) and in that GPU lies a nominal computing power that makes the power of even the most recent multicore central processing units looks anemic in comparison. Innovative processing architectures such as the Cell Broadband Engine from IBM found in the Sony's Playstation 3 also come full of promises. Thus, it did not take long in these days of ever growing computing needs for people to explore these new lands.

The power of these new heterogeneous computing architectures does not come for free, however. Cell's multiple synergistic processing units (SPUs) are equipped with a very small amount of memory. GPUs put drastic constraints on the data access pattern and require highly regular computations to actually deliver their full power. While scheduling over those architectures, the problem of mapping tasks onto available units is not the only one anymore. One also has to provide constructs and mechanisms to tailor those tasks to the characteristics of a given processing unit — refinement/filtering mechanisms — as well as to make sure that such tasks have the suitable data at hand when needed — memory/caching management and consistency mechanisms.

We thus designed a unified scheduling engine that makes it possible to easily implement task scheduling policies on top of heterogeneous multicore architectures. Combined with the use of performance models, we have shown that substantial performance improvements result from the use of such scheduling policies. Not only does StarPU allow to make use of all computing ressources at the same time, but its scheduling engine even enables to *benefit* from the actual heterogeneity of the machines.

The memory mangement library of the StarPU runtime system (described in Section 5.9 has been designed in order to leverage the inner complexity of accelerator programming by automating data coherency management [42]. Our approach makes it possible to handle arbitrarily large datasets instead of being limited by the size of accelerators' embedded memory [21]. Therefore, we used StarPU to implement dense linear algebra parallel kernels that run simultaneously on multicore processors and GPGPUs. We showed that in addition to a unified execution model, it is important to exhibit an expressive interface to let the programmer (or the upper library) express his/her knowledge at the algorithmic level in order to give hint to the runtime system.

6.8. Optimization of the Sparse Direct Linear Solver PaStiX

In the context of distributed NUMA architectures, a work has recently begun, in collaboration with the INRIA ScalApplix team-project, on studying optimization strategies and improving the scheduling of communications, threads and I/O on the sparse direct linear solver PaStiX. The solver provides the NewMadeleine and Marcel libraries with an advanced application to validate those strategies. Mathieu Faverge studies these aspects in the context of the NUMASIS ANR project. It has been proved that NUMA allocation can significantly improve the efficiency. In the *out-of-core* context, new problems related to the scheduling and the management of the computational tasks may arise (processors may be slowed down by I/O operations). Thus, specific algorithms for this particular context have to be designed and implemented. First results on NUMA aspects and dynamic scheduling have been published in [26], [25] while the use of the NewMadeleine communication library is described in [17].

7. Contracts and Grants with Industry

7.1. PhD thesis co-supervised with CEA/DAM

3 years, 2007-2010

We did set up a collaboration with the CEA/DAM (French Atomic Energy Commission, Marc Pérache, Bruyère le Chatel) on the support of nuclear simulation programs (adaptive mesh) on large clusters of SMP (thousands of processors) and on Itanium2-based NUMA machines. In October 2007, François Diakhaté started a PhD thesis granted by the CEA under the co-supervision of Marc Pérache and Raymond Namyst about the use of virtualization within parallel applications over clusters of multiprocessor machines.

7.2. Contract between INRIA and Myricom

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

Brice Goglin visited Myricom in Arcadia in May 2008 to present the OPEN-MX stack. We are now involved in the design of the future specifications of Myricom's native MX (*Myrinet Express*) wire protocol and application interface, as well as hardware features that should be supported next-generation Ethernet hardware.

8. Other Grants and Activities

8.1. ANR projects

The National Agency for Research (ANR) has launched a program called CIGC (*Calcul Intensif et Grilles de Calcul*) about the development of High Performance computing and Grids. In 2005, twelve research proposals have been selected by the national committee. We participate to three of these projects (granted each a three-years funding, from 2006 to 2008):

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 multiparadigm. The aim of LEGO is to propose and to implement a multiparadigm 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. The LEGO project has been extended for 6 more months until June 2009.

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. The NUMASIS project has been extended for 1 more year until December 2009.

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, Cell processor, 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).

The National Agency for Research (ANR) launched in 2008 a program called COSINUS about design and numerical simulation for scientific research, industry and services. We wrote a research proposal called ProHMPT (*Programming Heterogeneous Multicore Processor Technologues*) which has been selected by the national committee. It was granted a three-years funding starting in 2009.

8.2. NEGST (NExt Grid Systems and Techniques)

3 years, january 2006 – march 2009.

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

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

8.3. PHC Pessoa MAE Grant

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

8.4. Associate Team between Runtime and MPICH2 team

Our proposal for an associate team between our group and the MPICH2 development team was accepted in December 2007. Thanks to this programm, numerous visits from both sides have been scheduled throughout this year. NEWMADELEINE has been integrated as a network module in the Nemesis communication channel (which is now the defaut communication channel in MPICH2). A paper ([33]) describing this work has been submitted the to IPDPS conference. Also some work regarding the integration of some mechanisms of the Open-MX software into the Nemesis channel has started during this year and the results are very promising.

8.5. PHC Sakura MAE Grant (submission)

During his stay at University of Tokyo (Japan), Raymond NAMYST has worked with Prof. Yutaka ISHIKAWA on the design of a multicore-enabled framework for communication and I/O in next generation clusters.

Today's high-end clusters usually use 16-core nodes and feature more than 10,000 cores. It is expected that the amount of cores per node will dramatically increase in the upcoming years, which will deeply modify the structure of clusters: they will move from large clusters of PCs to smaller clusters of powerful multicore machines. However, the other hardware components did not evolve so swiftly. In particular, each cluster node is only equipped with a small shared pool of network interface cards. As a consequence, remote IO operations and communication operations have to face a major bottleneck problem. Since this issue is inherent to the hardware, it actually affects most existing programming environments such as MPI libraries. As a result, the gap between theoretical peek and sustained performance of clusters is still widening.

Significant progress has recently been achieved about improving communication within clusters of multicore nodes. One approach, designed by the Runtime project-team, is based on a low-level flexible communication scheduler that can apply various packet-level optimizations. This work has demonstrated that careful optimizations at the runtime-level can dramatically increase network utilization and minimize the bottleneck problem that typically arises when multiple processes simultaneously access a small pool of shared network interface cards. Moreover, new results regarding efficient IO mechanisms for clusters of multicores have recently been achieved by Prof. Ishikawa's group. In a cluster of 10,000 multicores, for example, 10,000 remote file IO requests may be independently sent to the single file server via network. Such a large amount of independent IO requests causes low utilization of both the network and file server due to congestion. In order to avoid such low utilization, Prof. Ishikawa's group has developed a new mechanism to coordinate IO requests in order to reduce the network trafic and the file server load.

By comparing our respective approaches, we have realized that most of our optimization techniques are very similar in the sense that requests are gathered and arranged based on some application-specific optimization policy. The challenge is now to design a unified multicore-enabled framework to integrate and optimize both communication and remote file IO. Such a framework will form a great research vehicle to develop new optimization policies for the future multicore technologies. This research proposal has two main goals: providing a unified framework for dealing with IO and communication together with a common database of optimization strategies, and addressing a large spectrum of programming paradigms, such as multithreading, pure message-passing or hybrid programming.

If accepted, the formal collaboration between our groups is expected to start at the beginning of 2009.

8.6. INRIA – EDF R&D

A contract ("*accord cadre*") between INRIA and EDF R&D was signed in 2007. Preliminary meetings have taken place in October and November 2008 to draw the main interesting points of collaboration between INRIA and EDF R&D. The Runtime project-team is part of this process and has proposed several research directions in the "Computer science for intensive simulation" topic, coordinated by Franck Cappello.

8.7. Expertise

Raymond NAMYST serves as an expert for the following institutions:

- CEA/DAM ("scientific advisor" for the period 2008-2010) ;
- DGRI (French Ministry of Research) ;
- GENCI (http://www.genci.fr/?lang=en) ;
- ANR (Cosinus Program).

8.8. Committees

Raymond NAMYST was a member of the following program comittees: *EuroPVMMPI 2008*, *Cluster 2008*, *CCGRID'08*, *HPCVirt 2008* and *RenPar 2008*.

Guillaume MERCIER is member of the "International Workshop on Parallel Programming Models and Systems Software for High-end Computing" *P2S2* programm commitee.

Alexandre DENIS is member of the *RenPar 2009* and *CFSE 2009* program commitees.

8.9. Invitations

Raymond NAMYST has spent two months (August & September 2008) as an "Invited Professor" at the University of Tokyo, in Japan. He has worked with professor Yutaka Ishikawa at the University of Tokyo. This collaboration was supported by the JSPS (Japanese Society for the Promotion of Science) invitation fellowship program for research.

Brice GOGLIN has been invited by Myricom to spend a week in Arcadia (California) so as to discuss the status and future of the OPEN-MX software, and participate to the elaboration of next-generation hardware and software for high-performance networking with Ethernet.

Elisabeth BRUNET, Jérôme CLET-ORTEGA, Brice GOGLIN, Guillaume MERCIER, Stéphanie MOREAUD, and François TRAHAY visited the Argonne National Laboratory (Illinois) for an agregated duration of 11 weeks in 2008. These visits took place in the framework of the "Associate Team" MPI-Runtime between RUNTIME and the Radix Lab, responsible for the development of the MPICH2 software. Several talks were given regarding the NEWMADELEINE communication library, the PIOMAN progression engine, the OPEN-MX networking stack, and affinities tasks and communication.

Olivier AUMAGE, François BROQUEDIS and Pierre-André WACRENIER were invited for a week in Ioannina, Greece, to initiate a collaboration about the integration of our MARCEL and BUBBLESCHED software as part of the OMPi compiler runtime developed at university of Ioannina.

Cédric AUGONNET, Olivier AUMAGE and Jérôme CLET-ORTEGA were invited in Lisbon and Evora Universities, Portugal, to participate to the start-up meeting of the PHC Pessoa Grant on the subject of parallelizing constraints solvers on top of our runtime systems.

9. Dissemination

9.1. Reviews

Olivier AUMAGE was involved in the paper reviewing processes of the EuroPVM/MPI and Grid'2008 conferences and of the ASSESS'08 workshop.

Brice GOGLIN was involved in the paper reviewing process of the International Symposium on Cluster Computing and the Grid (CCGrid 2008).

Guillaume MERCIER was involved in the paper reviewing process of the International Symposium on Cluster Computing and the Grid (CCGrid 2008).

Alexandre DENIS was involved in the paper reviewing process of the International Conference on Cluster Computing (Cluster 2008), the Euro-Par 2008 conference, and the Rencontres Francophones du Parallélisme (RenPar 2008).

9.2. Seminars

Samuel THIBAULT gave a seminar about BUBBLESCHED at ENS Lyon (Apr. 2008) and at the INRIA multicore meeting (Sep. 2008).

Alexandre DENIS gave a seminar about "High-performance MPI communications on multi-core architectures" at the NEGST workshop (June 2008).

Raymond NAMYST gave several seminars about multicore programming (Aristote seminar (Oct. 2008), CEA/DAM (Oct. 2008), ANR "Alpage" invited speaker (Jan. 2008), RIKEN Institure (Japan, Sept. 2008)) and about high speed networking (University of Tokyo (Aug. 2008), University of Kyushu (Aug. 2008), University of Kyoto (Sept. 2008), AIST (Tsukuba, Sept. 2008)).

Olivier AUMAGE gave three seminars about fine grain multithreading on modern architectures at the universities of Ioannina (Greece, Jul. 2008), Lisbon and Evora (Portugal, Nov. 08)

Brice GOGLIN gave a seminar about OPEN-MX at Myricom (California, May. 2008) and about message passing over Ethernet and high-throughput intra-node communication at the Argonne National Laboratory (Illinois, Dec. 2008).

François TRAHAY gave a seminar about NEWMADELEINE and PIOMAN at the Argonne National Laboratory (Illinois, June. 2008).

Cédric AUGONNET gave seminars about STARPU at the BRGM (Orléans, Sept. 08) and at the universities of Lisbon and Evora (Portugal, Nov. 08) and Perpignan (Dec. 08).

Mathieu FAVERGE gave lectures in the CNRS "autumn school on scientific computing" held in Sète (France) from september 29th to october 3rd 2008.

9.3. Teaching

Olivier AUMAGE taught a course on "System and Middleware for Parallel and Distributed Computing" in the Master of Science at the University Bordeaux 1, in cooperation with Alexandre DENIS. He gave a course about "High-Performance Communication Supports" and a course on "Programming Languages for Parallelism" at the ENSEIRB engineering school.

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

Brice GOGLIN taught courses on “Operating System” and on “Parallel and Distributed Systems” at the ENSEIRB engineering school.

10. Bibliography

Major publications by the team in recent years

[1] G. ANTONIU, L. BOUGÉ, P. HATCHER, M. MACBETH, K. MCGUIGAN, R. NAMYST. *The Hyperion system: Compiling multithreaded Java bytecode for distributed execution*, in "Parallel Computing", vol. 27, October 2001, p. 1279–1297.

[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.

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

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

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

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

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

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

[9] L. BOUGÉ, P. HATCHER, R. NAMYST, C. PÉREZ. *A multithreaded runtime environment with thread migration for a HPF data-parallel compiler*, in "The 1998 Intl Conf. on Parallel Architectures and Compilation Techniques (PACT '98), Paris, France", IFIP WG 10.3 and IEEE, October 1998, p. 418-425.

[10] 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.

[11] 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.

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

[13] 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.

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

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

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

Year Publications

Doctoral Dissertations and Habilitation Theses

[17] E. BRUNET. *Une approche dynamique pour l'optimisation des communications concurrentes sur réseaux haute performance*, Ph. D. Thesis, Université Bordeaux 1, 351 cours de la Libération — 33405 TALENCE cedex, December 2008.

Articles in National Peer-Reviewed Journal

[18] E. BRUNET. *NewMadeleine : ordonnancement et optimisation de schémas de communication haute performance*, in "Technique et Science Informatiques", vol. 27, n^{O} 3-4/2008, 2008, p. 293–316, http://hal.inria.fr/inria-00110766.

[19] B. GOGLIN, O. GLÜCK, P. V.-B. PRIMET. *Interaction efficace entre les réseaux rapides et le stockage distribué dans les grappes de calcul*, in "Technique et Science Informatiques", vol. 27, n^{O} 7/2008, September 2008, p. 911–940, http://hal.inria.fr/inria-00070218.

[20] S. THIBAULT, R. NAMYST, P.-A. WACRENIER. *BubbleSched, plate-forme de conception d'ordonnanceurs de threads sur machines hiérarchiques*, in "Technique et Science Informatiques", vol. 27, n^{O} 3-4/2008, 2008, p. 345–371, http://hal.inria.fr/inria-00329960.

International Peer-Reviewed Conference/Proceedings

[21] C. AUGONNET, R. NAMYST. *A unified runtime system for heterogeneous multicore architectures*, in "Proceedings of the International Euro-Par Workshops 2008, Las Palmas de Gran Canaria, Spain", Lecture Notes in Computer Science, To appear, Springer, August 2008, http://hal.inria.fr/inria-00326917.

[22] F. BROQUEDIS, F. DIAKHATÉ, S. THIBAULT, O. AUMAGE, R. NAMYST, P.-A. WACRENIER. *Scheduling Dynamic OpenMP Applications over Multicore Architectures*, in "International Workshop on OpenMP (IWOMP), West Lafayette, IN", May 2008, http://hal.inria.fr/inria-00329934.

[23] F. BROQUEDIS, N. FURMENTO, B. GOGLIN, R. NAMYST, P.-A. WACRENIER. *Dynamic Task and Data Placement over NUMA Architectures: an OpenMP Runtime Perspective*, in "International Workshop on OpenMP (IWOMP), Dresden, Germany", Submitted, June 2009.

[24] E. BRUNET, F. TRAHAY, A. DENIS. *A Multicore-enabled Multirail Communication Engine*, in "Proceedings of the IEEE International Conference on Cluster Computing, Tsukuba, Japan", IEEE Computer Society Press, September 2008, p. 316–321, http://hal.inria.fr/inria-00327158.

[25] M. FAVERGE, X. LACOSTE, P. RAMET. *A NUMA Aware Scheduler for a Parallel Sparse Direct Solver*, in "Proceedings of PMAA'2008, Neuchâtel, Suisse", 2008, http://hal.inria.fr/inria-00344709.

[26] M. FAVERGE, P. RAMET. *Dynamic Scheduling for sparse direct Solver on NUMA architectures*, in "Proceedings of PARA'2008, Trondheim, Norway", 2008, http://hal.inria.fr/inria-00344026.

[27] B. GOGLIN, N. FURMENTO. *Enabling High-Performance Memory-Migration in Linux for Multithreaded Applications*, in "MTAAP'09: Workshop on Multithreaded Architectures and Applications, held in conjunction with IPDPS 2009, Rome, Italy", Submitted, IEEE Computer Society Press, May 2009.

[28] B. GOGLIN. *Design and Implementation of Open-MX: High-Performance Message Passing over generic Ethernet hardware*, in "CAC 2008: Workshop on Communication Architecture for Clusters, held in conjunction with IPDPS 2008, Miami, FL", IEEE Computer Society Press, April 2008, http://hal.inria.fr/inria-00210704.

[29] B. GOGLIN. *Improving Message Passing over Ethernet with I/OAT Copy Offload in Open-MX*, in "Proceedings of the IEEE International Conference on Cluster Computing, Tsukuba, Japan", IEEE Computer Society Press, September 2008, p. 223–231, http://hal.inria.fr/inria-00288757.

[30] B. GOGLIN. *Decoupling Memory Pinning from the Application with Overlapped on-Demand Pinning and MMU Notifiers*, in "CAC 2009: The 9th Workshop on Communication Architecture for Clusters, held in conjunction with IPDPS 2009, Rome, Italy", Submitted, IEEE Computer Society Press, May 2009.

[31] B. GOGLIN. *High Throughput Intra-Node MPI Communication with Open-MX*, in "Proceedings of the 17th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP2009), Weimar, Germany", To appear, IEEE Computer Society Press, February 2009, http://hal.inria.fr/inria-00331209.

[32] B. GOGLIN. *NIC-assisted Cache-Efficient Receive Stack for Message-Passing over Ethernet*, in "Proceedings of 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS'09), Rome, Italy", Submitted, IEEE Computer Society Press, May 2009.

[33] G. MERCIER, F. TRAHAY, D. BUNTINAS, E. BRUNET. *NewMadeleine: An Efficient Support for High-Performance Networks in MPICH2*, in "Proceedings of 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS'09), Rome, Italy", Submitted, IEEE Computer Society Press, May 2009.

[34] M. NIJHUIS, H. BOS, H. BAL, C. AUGONNET. *Mapping and synchronizing streaming applications on Cell processors*, in "International Conference on High Performance Embedded Architectures & Compilers, Paphos, Cyprus", To appear, January 2009.

[35] S. THIBAULT, T. DEEGAN. *Improving Performance by Embedding HPC Applications in Lightweight Xen Domains*, in "2nd Workshop on System-level Virtualization for High Performance Computing (HPCVIRT'08), Glasgow, Scotland", March 2008, http://hal.inria.fr/inria-00329969.

[36] F. TRAHAY, E. BRUNET, A. DENIS, R. NAMYST. *A multithreaded communication engine for multicore architectures*, in "CAC 2008: Workshop on Communication Architecture for Clusters, held in conjunction with IPDPS 2008, Miami, FL", IEEE Computer Society Press, April 2008, http://hal.inria.fr/inria-00224999.

[37] F. TRAHAY, E. BRUNET, A. DENIS. *Analysis of the impact of multi-threading on communication performance*, in "CAC 2009: The 9th Workshop on Communication Architecture for Clusters, held in conjunction with IPDPS 2009, Rome, Italy", Submitted, IEEE Computer Society Press, May 2009.

National Peer-Reviewed Conference/Proceedings

[38] F. BROQUEDIS. *Exécution structurée d'applications OpenMP à grain fin sur architectures multicoeurs*, in "18ème Rencontres Francophones du Parallélisme, Fribourg / Suisse", École d'ingénieurs et d'architectes de Fribourg, February 2008, http://hal.inria.fr/inria-00203188.

[39] J. CLET-ORTEGA. *Ordonnancement et services différenciés pour réseaux rapides*, in "18ème Rencontres Francophones du Parallélisme, Fribourg / Suisse", École d'ingénieurs et d'architectes de Fribourg, February 2008, http://hal.inria.fr/inria-00332260.

[40] S. MOREAUD. *Impacts des effets NUMA sur les communications haute performance dans les grappes de calcul*, in "18ème Rencontres Francophones du Parallélisme, Fribourg / Suisse", École d'ingénieurs et d'architectes de Fribourg, FEB 2008, http://hal.inria.fr/inria-00257752.

[41] F. TRAHAY. *PIOMan : un gestionnaire d'entrées-sorties générique*, in "18ème Rencontres Francophones du Parallélisme, Fribourg / Suisse", École d'ingénieurs et d'architectes de Fribourg, February 2008, http://hal.inria.fr/inria-00327177.

Other Publications

[42] C. AUGONNET. *Vers des supports d'exécution capables d'exploiter les machines multicoeurs hétérogènes*, Mémoire de DEA, Université Bordeaux 1, June 2008, http://hal.inria.fr/inria-00289361.

References in notes

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

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

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

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

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

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

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

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

[51] 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.

[52] 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.

[53] 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.

[54] 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.

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

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

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

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] G. R. GAO, T. STERLING, R. STEVENS, M. HERELD, W. ZHU. *Hierarchical multithreading: programming model and system software*, in "20th International Parallel and Distributed Processing Symposium (IPDPS)", April 2006.

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

[65] 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.

[66] 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.

[67] 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.

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

[69] 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.

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

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

[72] 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.

[73] 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.

[74] 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.

[75] 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/.

[76] S. THIBAULT. *Ordonnancement de processus légers sur architectures multiprocesseurs hiérarchiques : BubbleSched, une approche exploitant la structure du parallélisme des applications*, 128 pages, Ph. D. Thesis, Université Bordeaux 1, 351 cours de la Libération — 33405 TALENCE cedex, December 2007, http://dept-info.labri.fr/~thibault/these/.