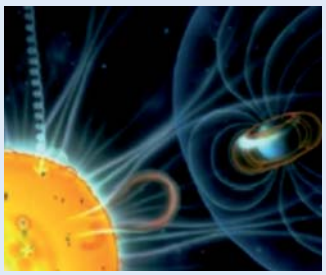
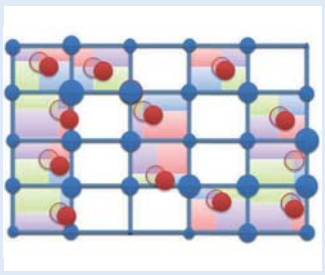
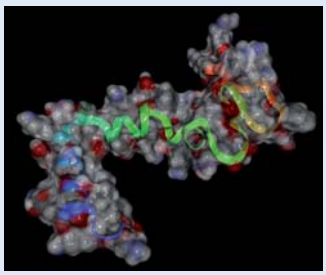
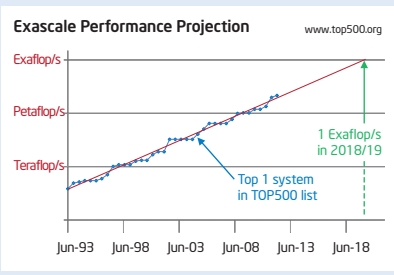
Intel European Exascale Labs

Report 2011









INDEX

FOREWORD

As a major R&D investment in Europe, Intel has established Exascale Labs in Jülich, Leuven, and Paris during 2010. The labs are collaborations with leading European HPC research organizations, and their goal is to address the challenges of Exascale computing. In the last year the three labs have made good progress toward their R&D agenda, and the first demonstrable results were presented throughout the year. In this 2011 report we will summarize the activities of the Intel European Exascale Labs and describe some of our results in short research papers.

In June 2011 Kirk Skaugen (Intel Vice President and General Manager of the Datacenter and Connected Systems Group) announced that Intel will build an Exascale HPC system by 2018 with a power consumption of less than 20 megawatts. There is widespread agreement that Exascale HPC systems will be very different from today's HPC systems in the petascale class, and that building, operating and using Exascale systems will face serious technological challenges. There is also wide-spread consensus in the HPC community that these challenges cannot be dealt with on the hardware level only. The complete software stack (tools, middleware, and applications) has to address the Exascale challenges:

- **Scalability:** An Exascale system will contain in the order of 10^8 cores. This is 1000 times more than today's most powerful petascale class HPC systems. Not only the applications and the algorithms, but the whole software ecosystem has to support this unprecedented level of parallelism.
- **Resilience:** Statistics predict that the meantime between critical failures will become shorter than those of today. The general expectation is that resiliency cannot rely on only hardware features and global checkpoint-restart mechanisms. The whole software ecosystem has to be aware of the resilience issue; even applications will potentially have to adopt new algorithms (e.g., stochastic algorithms) which can tolerate a certain amount of failures.
- **Energy:** Exascale systems will be based on low-power components (cores, memory, interconnect, and so on). Energy consumption will be crucial and needs to be dynamically managed through software control. In particular, developer tools have to adopt the notion of "energy optimization" in addition to the standard "performance optimization".

The main role of the Intel European Exascale Labs is to investigate these challenges, identify potential solutions, and influence the feedback and co-design process between hardware architecture, HPC middleware, and HPC applications.

Each lab focuses on different aspects and technologies: At the Intel Exascale Computing Research Lab in Paris important applications are characterized using advanced tools, and Exascale performance bottlenecks are identified. The Intel ExaScience Lab in Leuven develops optimized numerical kernels and evaluates these using a new simulator for many-core architectures. Finally, the Intel ExaCluster Lab in Jülich is building an innovative hardware and software cluster architecture to surpass the limits of Amdahl's law.

After their start in 2010 as separate labs, the three labs are now working together in certain areas to make best use of Intel's and the partners' investment. The results of the labs are generally very accessible, most of the software developments will be released under open source licenses. We are interested in the discussion and exchange with the European HPC community, and we will present our results in international conferences, workshops, round tables, and throughout the Intel Labs Europe organization.

In 2011 Intel European Exascale Labs have become a relevant player in the European HPC and Exascale R&D community. Besides our engagement in the EU-FP7-funded DEEP ("Dynamic Exascale Entry Platform") project, we support the European Exascale Software Initiative (EESI) project, which recently finalized its work on a European software roadmap for Exascale. We participate in two FET flagship pilots, and we will actively contribute to the research agenda of the new European Technology Platform (ETP). We are part of Intel Labs Europe (ILE), a network of Research & Development, Product, and Innovation Labs spanning the European region, as well as a variety of Intel business units.

Finally we are proud to announce a further European Exascale lab in cooperation with the Barcelona Supercomputing Center (BSC). The new Intel and BSC Exascale Laboratory will focus on scalable run-time systems, advanced performance optimization and prediction tools, and innovative new algorithms.

Enjoy reading our 2011 report.

Karl Solchenbach
Director Intel European Exascale Labs

KEY EVENTS 2011

INTEL EXASCALE LEADERSHIP CONFERENCE
MAY 18, 2011, GENEVA, SWITZERLAND

Intel invited the European thought leaders in Exascale to the first "Intel Exascale Leadership Conference" to bring together experts from HPC centers, European R&D centers, European industry, and European policy makers to share their view on the challenges ahead for Europe and on strategies to address them. In the prestigious GLOBE, CERN's ecological conference center, keynote speakers from the European Commission, the PRACE consortium, and European HPC users shared their views on the chances and challenges of Exascale computing, and their strategies and roadmaps. Intel leaders Steve Pawlowski, Shekhar Borkar and Antonio Gonzalez presented an outlook of Intel's technology roadmap toward Exascale.

INTERNATIONAL SUPERCOMPUTER CONFERENCE (ISC'11)
JUNE 19-23, 2011, HAMBURG, GERMANY

The Exascale labs contributed to ISC'11 through various activities, including a speaker role at the Exascale panel session. At the Intel booth the Jülich ExaCluster lab demonstrated a full application (SMMP) running on a software development system of the Intel® MIC architecture (code name: Knights Ferry). The SMMP work touches on many interesting algorithmic aspects, such as vector operations, sorting, hashing, and irregular kernels.

EUROPEAN EXASCALE SOFTWARE INITIATIVE (EESI) FINAL CONFERENFCE
OCTOBER 10-11, 2011, BARCELONA, SPAIN

The European Exascale Software Initiative (EESI) has built a European vision and roadmap to address the challenges of the new generation of massively parallel systems composed of millions of heterogeneous cores which will provide Petaflop performances in 2010 and Exaflop performances in 2020. EESI coordinates the European contribution to the International Exascale Software Project (IESP). The EESI work, and in particular the recommendations and vision, were presented during the final conference in Barcelona, opened by Mario Campolargo (European Commission DG INFSO), who recalled the importance of this event for the future and the necessity of a joint effort. Intel European Exascale Labs support the EESI agenda and participated in the conference and the social program.

INTEL EUROPEAN EXASCALE LABS MEETING WITH INTEL HPC LEADERS
NOVEMBER 10-11, 2011, HILLSBORO, OREGON

A delegation of European Exascale Labs met with Intel HPC experts, researchers and product architects to present and discuss the results achieved in the Exascale labs. The agenda of these intensive two days covered the application portfolio of the labs, the work on tools, programming models and runtime systems, and the development on the system architecture level, such as the Sniper simulator and the DEEP architecture.

SUPERCOMPUTING 2011 (SC11)
NOVEMBER 13-17, 2011, SEATTLE, WASHINGTON

The European Exascale Labs teams played a very active role at SC11:

- Organizing and contributing to various BOF sessions (e.g., on single-node optimization, Exascale software resiliency, and collective mining rich information for Exascale computing).
- Demonstrating an improved version of the protein folding code SMMP running on Intel® Xeon® E5 processors and the "Knights Ferry" coprocessor at the Intel booth.
- Announcing the new Intel-BSC Exascale Lab in Barcelona, as well as cooperation with STFC Daresbury.

Participants of the Intel Exascale Leadership Conference at CERN

"DEEP" KICKOFF MEETING
DECEMBER 5-6, JÜLICH, GERMANY

In the European Framework 7 project DEEP, a consortium of European leaders in HPC systems, software, and applications is building a prototype of the Dynamical Exascale Entry Platform architecture. This system will couple a top-of-the-line HPC Cluster using Intel® Xeon® processors to a Cluster Booster made out of Intel® MIC coprocessors and the Extoll* extreme performance interconnect. Extensions to ParTec's ParaStation® Cluster middleware and to the OmpSS programming model of the Barcelona Supercomputing Center will provide fully dynamic booster resource allocation and enable applications to transparently execute highly scalable kernels on the Booster Cluster with unparalleled throughput performance, while less scalable modules with highly complex control flow will profit from the proven performance of the Intel® Xeon® processor-based cluster. In effect, the system will alleviate the effects of Amdahl's law. Six important HPC applications from life sciences, engineering, climate research and astrophysics will highlight the performance potential of the DEEP architecture. The novel architectural concept together with the use of advanced cooling and thermal management technology will yield a significant improvement in power efficiency.

The DEEP project started on December 1st, 2011, and a two-day kickoff was held at the Jülich Supercomputing Center. The ExaCluster Lab plays a leading role in the project.

OTHER EVENTS

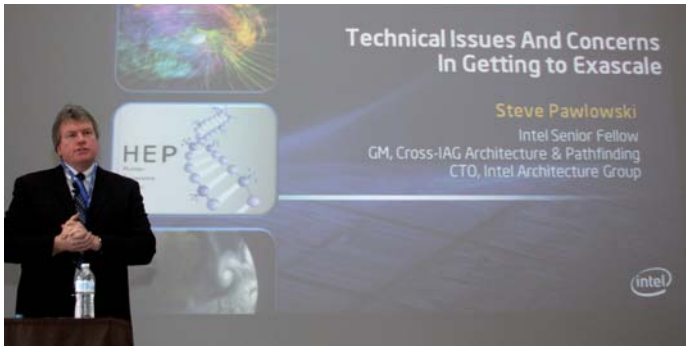
In 2011 the Exascale labs contributed to several other conferences:

- Intel HPC Round Table EMEA,
  April 7, Paris, France
- HPC Workshop,
  May 16–17, Ghent, Belgium
- Teratec Forum,
  June 28–29, Paris, France
- IDC HPC User Forum,
  October 3–4, Bruyères-le-Châtel, France
- European Health Forum Gastein,
  October 5–7, Bad Gastein, Austria
- ORAP Forum,
  October 14, Paris, France
- HIEPACS Seminar,
  November 2, Bordeaux, France
- Barcelona Multicore Workshop 2011,
  November 2–3, Barcelona, Spain
- FET Flagship Pilots Midterm Conference,
  November 24–25, Warsaw, Poland

Intel European Exascale Labs team at the Intel Jones Farm campus

ExaCluster Lab demo on Knights Ferry

Kick-Off of the new Intel-BSC Exascale Lab



Steve Pawlowski giving the Keynote presentation at the HPC Workshop in Gent

The DEEP project team at the Kick-Off meeting

INTEL EXACLUSTER LAB – JÜLICH

"The close collaboration between Intel, Europe's largest scientific computer center in Jülich, and the leading Cluster ISV ParTec presents an exciting opportunity to accelerate the evolution of clustered HPC platforms. The work on the novel DEEP architecture will be a key component to communicating the development of future Exascale systems, middleware, and applications."

Steve Pawlowski,
Senior Fellow, Intel

"Working closely with Intel is instrumental for Jülich Supercomputing Center to accelerate the research into cluster architectures that are ready for Exascale, and to address the hardware and software challenges in building, programming, and operating such systems."

Prof. Thomas Lippert,
Director of JSC

JÜLICH FORSCHUNGSZENTRUM

PARTEC CLUSTER COMPETENCE CENTER

High Performance Computing (HPC) plays a pivotal role in science and engineering. Increasing the performance of supercomputers to Exascale (10^{18} operations per second) will enable breakthrough discoveries that are not possible today. Current HPC systems follow the cluster approach; many thousands of identical computational nodes, each one equipped with a moderate number of CPU cores, are connected by a high-performance network and interfaced with a storage system. HPC applications are highly parallel, consisting of up to hundreds of thousands execution threads. The ExaCluster Lab focuses on two critical issues on the way to Exascale: effectively scaling up cluster systems, middleware and applications, and providing a sufficient level of reliability.

The ExaCluster Lab develops architectural concepts and prototype hardware and software for the next generations of cluster systems, leading up to Exascale.

EXACLUSTER RESEARCH

In the ExaCluster Lab, Intel has partnered with two leading centers of excellence in scalable HPC systems, applications, and middleware. The Jülich Supercomputing Center (JSC) is one of the largest HPC centers in Europe, with world-class research of software tools, applications and unsurpassed insight into operating Top 10 HPC systems. JSC is a centerpiece of the Forschungszentrum Jülich* (FZJ), the largest German government lab which focuses on materials science, energy, health and environmental research. ParTec GmbH* is one of the leading ISVs in scalable communication libraries and cluster management software.

ExaCluster research performed in the lab touches all hardware and software layers – from processor and network architecture, scalability and power efficiency to scalable and robust communications and management middleware, and top-tier applications from science and engineering that will turn new levels of HPC cluster performance into exciting discoveries and create improvements by leaps and bounds in products and services.

As a centerpiece of the lab's research, the DEEP project will prototype a new class of HPC clusters based on the Intel® MIC architecture which will push back the scalability limitations expressed by Amdahl's law and enable application sections to be run at exactly the right level of parallelism.

Hans Christian Hoppe, Intel

Nikolaus Lange, Intel

Thomas Lippert, FZJ

Wolfgang Gürich, FZJ

Hugo Falter, ParTec

SCALABILITY

Today's largest supercomputers deliver petascale performance with hundreds of thousands of compute cores and power consumption in the tens of megawatts. To reach Exascale, performance has to grow by a factor of close to 1000, while power consumption must stay within the range of 20 megawatts. Moore's law alone will not take us there by 2018, or even 2020. Significant improvements in power efficiency are needed, and Exascale systems will have tens if not hundreds of millions of compute cores.

Building, programming and, operating systems of this enormous scale pose extremely difficult challenges. Networks have to scale up and connect the cores with very low latency and high bandwidth; the memory subsystem has to match ever-increasing processor speed, while power consumption of all hardware components has to be pared down as far as possible. Communication and resource management systems must be significantly improved to support scalable applications and ensure efficient system usage. Finally, HPC applications have to be re-architected for extreme scalability, and extensions for today's proven programming models have to be found.

Michael Kauschke, Intel

Ilona Illgen, Intel

Bart Stukken, Intel

RESILIENCY

Projections strongly indicate that Exascale systems will have to operate in the presence of regular, if not frequent component failures. Since most HPC applications are tightly coupled across the cores running in parallel, a non-recoverable component failure puts the whole application at risk. To address this, the ExaCluster lab is developing new resiliency concepts that involve hardware, system and application software with the objective of allowing applications to recover from such failures. This goes far beyond conventional checkpointing, which does not scale up. The work leverages JSC's insight into failure modes of today's largest systems, Intel's competency on the hardware, and OS side and ParTec's extensive experience in cluster management and communication software.

Norbert Eicker, FZJ

Jochen Kreutz, FZJ

Estela Suarez, FZJ

"DEEP" EXASCALE EXPERIMENTS

Intel® Many Integrated Core (Intel® MIC) architecture promises a breakthrough in performance per watt with unparalleled throughput performance in a very compact form factor, making it an interesting platform for scaling experiments. The Dynamic Exascale Entry Platform (DEEP) architecture is based on combing a "booster" cluster of MIC nodes with a second cluster made of Intel® Xeon® processors. HPC applications do consist of sections with different levels of parallelism, with the least scalable part putting a hard limit on the achievable total parallelism and therefore performance. The heterogeneous DEEP architecture pushes these limits by running each application section at a "sweet spot" with regard to parallelism and per core performance. Sections with less parallelism run on the by core faster Intel® Xeon® processor part of the system, while the highly parallel computational kernels can use the far wider parallelism of the booster cluster. To make such a system work, DEEP relies on an extremely highly performing custer network for the booster and for coupling the two clusters.

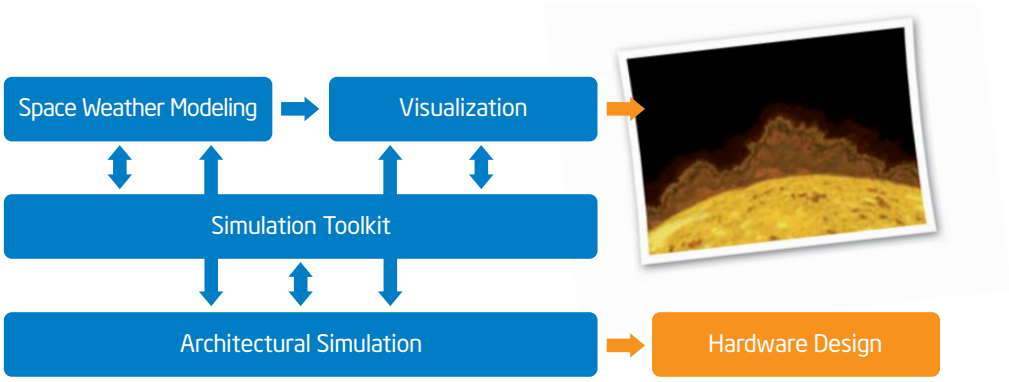
Early work in the ExaCluster lab on the first, not publicly released implementation of MIC, has clearly shown MIC's potential to execute extremely complex applications with high efficiency and excellent throughput performance. Within the DEEP project, a selection of highly relevant and complex HPC applications will be ported to MIC and then to the DEEP prototype systems.

Ina Schmitz, ParTec

Thomas Moshny, ParTec

INTEL EXASCIENCE LAB – LEUVEN

Breakthroughs in Exascale computing could mean the ability to simulate very complex systems, which are impossible to replicate today, such as the simulation of human body systems or the prediction of Earth's climate. Exascale computing will make it possible to find new cures for diseases or to more easily natural catastrophes. The ExaScience Lab is focused on enabling scientific applications for highly parallel systems, more specifically improving the simulation and prediction accuracy of space weather.



The ExaScience Lab develops software for high performance computing (HPC). This software will run on future Exascale computer systems, delivering 1,000 times the performance of today's fastest supercomputers. The lab is a unique collaboration between Intel, Imec* and the five Flemish universities, and is staffed by over 25 scientists.

EXASCIENCE RESEARCH

The ExaScience Lab focuses on enabling scientific applications, beginning with the simulation and prediction of space weather influenced by the solar electromagnetic activity in the region nearby to the Earth's atmosphere.

This project advances the current state of the art in four scientific areas: (1) accurate space weather modeling and simulation, (2) scalable numerical kernels, programming models and resilient runtime layer, (3) scalable, in situ visualization through virtual telescopes and (4) architectural simulation of large-scale systems and workloads.

*(**Intel:** Luc Provoost*
***Imec:** Wilfried Verachtert)*

SPACE WEATHER

The ExaScience Lab is organized around a linchpin application: space weather simulation and prediction. Space weather was chosen as the driver for ExaScience Lab research for a number of reasons:

- to better understand space weather physics and to enable better predictions of its effects on Earth
- because the algorithms and parallelization strategies needed for such simulations are similar in nature to the ones that are expected to drive a host of applications on future Exascale systems

*(**KU Leuven*:** Arnaud Beck ▪ Pierre Henri ▪ Giovanni Lapenta ▪ Stefano Markidis ▪ Stefaan Poedts ▪ Alexander Vapirev)*

SIMULATION TOOLKIT

The toolkit shields the application developer from hardware architectural details, including hierarchy, variability, and failures.

It provides numerical kernels optimized for power/energy efficiency and scalability, including Krylov methods and AMR. This will lead to Exascale workloads typical for space weather and similar simulations.

The simulation toolkit also offers a hierarchical programming model and a matching runtime layer built on a resilient work stealing scheduler with a reactive work rebalancer. It is this runtime layer that shields the application program from the runtime effects of variable and failing hardware components.

*(**Universiteit Antwerpen*:** Pieter Ghysels ▪ Przemyslaw Klosiewicz ▪ Wim Vanroose*
__KU Leuven:__ Pawan Kumar ▪ Karl Meerbergen ▪ Dirk Roose ▪ Bart Verleye ▪ Albert-Jan Yzelman
__Vrije Universiteit Brussel:__ Bruno De Fraine ▪ Wolfgang De Meuter ▪ Charlotte Herzeel ▪ Tom Van Cutsem
__Imec:__ Tom Ashby ▪ Roel Wuyts
__Intel:__ Pascal Costanza)

"We are excited about this unique collaboration with Intel and five Flemish universities here at imec. By sharing the expertise, I'm convinced that the Flanders ExaScience Lab brings valuable software solutions for Intel's future Exascale computers."

Luc Van Den Hove,
president and CEO of imec

ARCHITECTURAL SIMULATION FRAMEWORK

The architectural simulation framework allows studying the behavior of this Exascale workload on future Exascale systems.

This framework enables the study of the detailed behavior of representative workloads on an accurate software model of Exascale hardware.

These experiments will lead to recommendations for the actual hardware design of future Exascale systems and feedback on performance, power, and reliability.

*(**Universiteit Gent*:** Trevor Carlson ▪ Koen De Bosschere ▪ Jeroen De Wachter ▪ Lieven Eeckhout ▪ Wim Heirman ▪ Souradip Sarkar*
__Imec:__ Ma Zhe
__Intel:__ Ibrahim Hur, Alexander Isaev)

VISUALIZATION SOFTWARE

Todays visualization approaches are not appropriate for Exascale computing. A novel, extremely scalable, approach is being developed which will operate in situ: on data in memory. It will allow presentation (animation), interpretation and in-depth space-time analysis of the simulation results.

The approach is generic and applicable to other problems beyond space weather, but at first, will allow space weather experts to make better forecasts, taking into account the space weather impact on Earth by utilizing the virtual observation points in space, called "virtual telescopes."

*(**Vrije Universiteit Brussel*:** Jan Lemeire ▪ Petar Marendic ▪ Dean Vucinic*
__Universiteit Hasselt:__ Cosmin Ancuti ▪ Philippe Bekaert ▪ Tom Haber)*

INTEL EXASCALE COMPUTING RESEARCH LAB – PARIS

"The Exascale momentum created around the lab, with partners from industry and academia, makes the French lab a unique player for HPC research in Europe.

Extreme parallelism, performance, and power consumption are the cornerstones of our work, in close collaboration with leadership class applications."

William Jalby, UVSQ, CTO of Exascale Computing Research Lab

PRESENTATION

The French Exascale Computing Research (ECR) Lab has been the first Exascale lab established by Intel and partners in Europe. ECR is the fruit of the collaboration between CEA*, Genci*, the University of Versailles-Saint-Quentin-en-Yvelines*, and Intel. The mission of ECR is to conduct R&D studies in co-operation with European researchers on applications that are critical for industries and HPC users from European universities.

It focuses on software for Exascale in two main research areas:

- Application enabling co-design: collaborative research with industry and academia to prepare existing applications for Exascale through close partnership between application developers and the lab. The research addresses specific features, and concentrates on programming models for high scalability, data flows, and numerical performance.
- Software tools and middleware to characterize applications and optimize their performance on future Exascale machines: The work will allow developers to improve scalability, performance, resource saturation and power consumption of their parallel applications. It will also help hardware designers and compiler builders optimize their products.

William Jalby, Chief Technologist, UVSQ Professor

Bettina Krammer, software Tools and run-time systems, UVSQ Lab Director

Marie-Christine Sawley, enabling application co-design, Intel Lab Director

ENABLING APPLICATION CO-DESIGN

Leveraging from the capacity of new architectures for progress in science

The purpose of this research area is to analyze selected applications with their full complexity, assess their behaviour on prototypes of future architectures, and work in close collaboration with the developer to increase scalability and efficiency. The methodology for performance evaluation developed at the lab is central to these activities. Comprehending the full complexity of a scientific application can be done only in close collaboration with scientists using such codes.

An example of a molecular simulation is described here on page 32. In order to ensure synergy across the different cases, the team concentrates on three main features:

- how to best map the "natural parallelism", as inherited by the model, onto the programming models and through to hybrid programming
- focus on the numerical algorithms, their stability, precision and scalability on different architectures
- understand the main data flows in the application, and anticipate possible bottlenecks, through to building skeletons apps, if required

Othman Bouizi, Intel

Thomas Guillet, Intel

"Thanks to the expertise of the lab, our scientific users are assisted in becoming prepared for the latest technologies in HPC and supported in being world-class competitors in their field of expertise."

Catherine Rivière, CEO Genci

SOFTWARE TOOLS FOR APPLICATION CHARACTERIZATION AND PERFORMANCE OPTIMIZATION

Helping hardware, middleware, and application software developers

The efficiency of an application running on a specific system is a direct function of the key characteristics of the application, the underlying software stack, and the hardware. Being able to detect, understand, take advantage of these characteristics, and possibly fine-tune the code, is critical in an Exascale environment. To achieve this, the lab is working on several projects, which are all being validated with real-world applications:

- Programming models and runtime system extensions based on the MPC framework (see the article on page 34). MPC is a unified runtime environment for parallel applications on very large clusters of multicore/multiprocessor NUMA nodes. MPC is designed to be highly scalable and to provide maximum performance to the application, while keeping the memory footprint of the runtime as low as possible.
- A set of performance tools and a methodology to help users identify performance problems quickly and evaluate potential gains through optimizations, with special focus on the analysis of memory behaviour and single-node optimization, and in collaboration with renowned performance tool development teams world-wide. In addition, tools are being developed for balancing out an application's performance with its power consumption.
- An application characterization framework, which extracts the hot code segments and necessary parameters from an application, performs systematic analysis of these segments, and derives optimization recommendations and performance predictions. This framework is designed to deliver critical and fine-grain information for tuning the overall performance of an application.

"Combining CEA's high level of expertise in HPC with the lab skills and tools has proved extremely fruitful for us and the HPC community as demonstrated by the last developments on MPC."

Jean Gonnord, Computer Simulation and IT Project Director at the CEA

Jean-Thomas Acquaviva, UVSQ

Jean-Christophe Beyler, Intel

Patrick Carribault, CEA

Souad Koliai, UVSQ

Pablo Oliveira, UVSQ

Emmanuel Oseret, UVSQ

Marc Pérache, CEA

Franck Talbart, UVSQ

Marc Tchiboukdjian, UVSQ

DEEP – An Accelerated Cluster Architecture for Exascale Computing

Norbert Eicker[1,2]
Thomas Lippert[2]
[1] Intel ExaCluster Lab Jülich
[2] Jülich Supercomputing Center, Forschungszentrum Jülich

Introduction

Today clusters of general-purpose CPUs are the dominant system architecture for HPC. They have replaced specialized architectures due to the ability to leverage advances in mainstream computing. Each new CPU generation brings more compute power and can replace their predecessors without the need to change the high-speed interconnect fabric or disrupt the complex system and application software stack at the same time.

With petascale (10^{15} floating point operations per second) systems in production today, the next challenge is to reach Exascale by the end of the decade. In recent years, specialized accelerator architectures have emerged as strong competitors to mainline CPUs, in particular regarding energy efficiency and price. General-purpose GPUs (GPGPUs) attached to cluster nodes boost the performance of several top HPC systems as shown by the current Top500 list [1]. Unfortunately, this way of using accelerators will not deliver truly scalable performance for real world applications, since Amdahl's Law imposes strict limits on application scalability, and node attached accelerators lack the flexibility to run highly dynamic workloads efficiently. Therefore it is necessary to develop the concept of the cluster architecture in HPC and keep it viable on the path to Exascale.

For guidance in which direction clusters should evolve it is prudent to reinvestigate the requirements of relevant Exascale applications. We will argue that such applications will have more than one level of scalability; highly scalable parallel kernels with simple communication patterns make up the majority of the computation, but less scalable kernels with complex communication patterns will strictly limit the overall scalability of the application according to Amdahl's Law [2].

Based on this analysis we will propose a novel HPC architecture that complements a classical HPC cluster with a booster system consisting of accelerator nodes and a high-speed interconnect with an extremely scalable torus topology. The basic idea is to use the booster for the highly scalable kernels of Exascale applications, and run the less scalable complex parts on the cluster CPUs. This uses both systems at their optimal operating point — the booster leverages extreme parallelism to deliver unsurpassed throughput performance, and the cluster can rely on its high per-thread performance for complex workloads.

This paper is organized as follows: In the next section we explore the space of HPC architectures by analyzing results taken from the TOP500 list. With that we identify the challenges arising from the goal to reach Exascale by the end of the decade. In the fourth section general ideas concerning scalability in HPC are revised. The fifth section details our proposed next-generation system architecture, christened DEEP for *Dynamic Exascale Entry Platform*. After discussing the DEEP software stack we conclude our thoughts and give an outlook on future activities in the DEEP project funded by the European Community's Seventh Framework Programme under grant agreement no. 287530.

Significant parts of the analysis and architecture definition were performed within the ExaCluster Lab, and system prototyping activities in the DEEP project will be the lab's focus for the next three years.

HPC Architectures in the TOP500 List

With its detailed statistics of the world's fastest HPC systems spanning 19 years, the TOP500 list [1] is a powerful tool in used to identify trends in computer architectures. It shows the emergence of clusters is the dominant HPC architecture, which slowly started in the late 1990s and rapidly gained momentum in the early 2000s. Their stage was set by commodity processors with ever increasing floating point performance (e.g., Compaq's Alpha, IBM's Power, AMD's Opteron or Intel® Xeon® processors), by powerful networks that enabled the connection of such commodity nodes (at the time, MyriNet and Quadrics, and later, InfiniBand and GigaBit Ethernet variants), and last but not least the availability of a complete Open Source software stack.

Today the prevailing operating system in HPC is Linux. In fact the big success of clusters would have been unlikely without the availability of an Open Source OS that vendors and users can freely adapt to meet specific HPC requirements (support for low-latency high-throughput networks, efficient multi-threading, highly tuned communication stacks, scalable I/O) and at the same time provides a uniform SW development platform, significantly cutting the costs of developing, porting tools, and applications.

Distributed memory systems like clusters do require a convenient and efficient programming paradigm. The HPC community standardized the Message Passing Interface (MPI) [3], and several free, high-quality MPI implementations of the MPI standard quickly became available. Access to source code enabled the community to adapt to every hardware innovation in this field, ensuring both usability of clusters for real applications and sufficient performance.

Since then cluster architectures have taken over more than 80% of the market leaving the remainder for Massively Parallel Processing (MPP) systems. This incredible success is caused by the ability to benefit from the enhancements of computer technology in general. A relatively small market segment like HPC cannot afford to drive development of specialized CPUs and systems. Instead, HPC profits from the "trickle up" of technology from mainstream segments such as enterprise servers, personal computing, and gaming [4]. An excellent example are GPGPUs — the steady revenue stream from sales to gaming and professional graphics customers did enable GPU vendors to compete in the HPC market with extended versions of their graphics cards.

A crucial feature of HPC architectures is the balance of all building-blocks in a system. Most HPC systems don't rely on the highest performing CPU available at a given time, but instead use a compromise that is matched to the available memory and network bandwidths, and one that uses energy efficiently or optimizes the ratio of price vs. delivered performance.

Another important result derived from the TOP500 list is the growth of available compute power. Combined with results of historical HPC systems it shows that HPC systems gain a factor of 1000 in performance per decade. This by far outperforms "Moore's Law" [5], i.e., the observation that semiconductor technology doubles the number of transistors per unit area every 18 months. The performance gain derives from an ever increasing level of parallelism, up to the order of millions of CPUs. All current architecture are massively parallel.

The Exascale (Cluster) Challenge

Having systems at petascale today, the community is considering about the next step, i.e., introducing Exascale systems (10^{18} floating-point operations per second) by the end of the decade. An analysis done by a group of experts two years ago showed [6] that simply extrapolating today's technology to produce Exascale systems will create extremely serious problems:

- **Power consumption:** projecting today's architectures and concepts onto the technology of 2018 shows that an Exascale system would require several 100 MW of power. Of course, it is not acceptable to use a full sized nuclear power plant exclusively for running a supercomputer.
- **Resiliency:** projecting the trend of an ever increasing number of components in HPC systems by the end of the decade will lead to many millions of components. Combined with today's components' mean-time to failure (MTTF) this will make Exascale systems unusable since the whole system's MTTF will drop into the range of hours or even minutes.
- **Memory and I/O:** the increasing gap between the development of compute performance and bandwidth of both memory and storage will require additional layers in the memory hierarchy of Exascale systems like higher level CPU caches or flash-memory as disk caches.
- **Concurrency:** the rapid growth in degree of parallelism makes it harder for the users to exploit this systems. Additionally, it introduces new requirements on the scalability of Exascale applications as we will discuss in the next section.

Thus, in order to achieve the ambitious goal of Exascale by the end of the decade one has to look for new, innovative and potentially disruptive ideas.

With these challenges in mind, the ability of HPC clusters to reach Exascale must be reviewed. In particular, the question has to be raised, whether the central concept of clusters — the utilization of commodity, general-purpose CPUs — competes with specialized accelerators or HPC CPU architectures.

For the HPC architectures, the bar over the next years is set by IBM's efforts leading to the BlueGene/Q system. An analysis of preliminary results [6] creates reasonable doubt that commodity CPUs will be sufficient in the future. This is due to apparent limitations on energy efficiency and the superior price/performance ratio of the BlueGene/P technology. Both are caused by the need for commodity CPUs to carry "dead weight" that is required for running general workloads with good performance, but does not come into play for comparatively simple floating point workloads common in HPC. This extra complexity does cost die space and adds additional stages to instruction execution, limiting achievable clock frequency and power efficiency.

The rapid development of GPGPUs does show a second potential path to Exascale. The rendering algorithms used by DirectX and OpenGL are similar to many HPC kernels in their need for absolute floating-point performance and high degree of single instruction/multiple data (SIMD) parallelism. Extending their established GPU architectures to support for instance, double precision and single program/multiple data parallelism (SPMD), NVIDIA and AMD have produced accelerators that by far surpass the best general-purpose CPUs in throughput performance, still at the cost of reduced functionality and increased programming effort.

A third relevant development is the new Intel® Many Integrated Core (MIC) architecture [7]. Originally designed as a graphics architecture [8], MIC relies on x86 cores that are highly optimized for functioning workloads and provide multiple vector units for extreme SIMD performance. Early results on the Knights Corner MIC implementation disclosed at SC'11 show floating point performance and energy efficiency more than matching the best GPGPUs on the market. The inherent advantage of the MIC approach is that x86 codes can be moved to such a system with limited effort, and that common parallel programming paradigms are supported.

Approximately 40% of the top 10 systems in the current TOP500 list are equipped with GPGPU accelerators attached to their cluster nodes. This "accelerated cluster" concept suffers from severe limitations of balance and scalability. The accelerators are connected to their cluster node by a comparatively slow bus (PCI Express), and can neither access the CPU memory with full speed nor directly communicate with each other. To move data between the local memory of two accelerators, it has to be copied back and forth between main memory and the memory residing on the accelerator card, using up the scarce resource of bandwidth on the node's system bus. This limits available bandwidth and drives up latency for communication between the actual compute elements.

A good solution to this dilemma would be to have a cluster of accelerators. In this concept the node consists of an accelerator only — accompanied by some memory and a high-speed interconnect — without a general purpose CPU. Programs running on the accelerator cores are capable of initiating communication operations directly. A good example is the QPACE system [9] that was ranked #1 on the GREEN500 list [10] of the most energy-efficient HPC systems in the world in 2009.

Nevertheless, this concept has limitations, too. Besides the problem of finding accelerators that are capable of running autonomously they might be not flexible enough to drive a high-speed interconnect efficiently. Furthermore, the gain introduced by the direct connection between the compute element and the interconnect fabric might be wasted by the fact that accelerators — as highly dedicated devices — suffer when running general purpose codes.

Scalability Considerations

Talking about Exascale it is crucial in discussing the effects of parallelism on scalability. A first result in this field is known as Amdahl's Law [2]. It states that the scalability of a parallel program is inherently limited by the sequential part of this application.

Assume that the run-time of the program's part that is parallelizable is given by $p * t$ on a single processor. Thus, the sequential part will take $(1-p)t$. If this program is executed on a computer providing N processors, the amount of time taken by the parallelizable part reduces to pt/n. By definition, the run-time of the sequential part will not be reduced at all by additional processors. The speedup S of the program on a parallel machine is given by the ratio of the execution time on this machine T_N compared to the run-time on a serial machine T_S, i.e.:

$$S = \frac{T_S}{T_N} = \frac{(1-p)+p}{(1-p)+\frac{p}{N}} = \frac{1}{(1-p)+\frac{p}{N}}$$

In practice this means that the parallel speedup is limited by the sequential part of a given program. For example, in a program with a sequential part of just 1% the speedup will be limited to 100, even if we add $N = \infty$ processors. The fact that some applications are able to scale on BlueGene systems with $O(100000)$ processors does indicate that these do have a vanishing sequential part.

Amdahl assumes that the problem size is the same for all systems. Gustafson's Law [11] investigates a case where the problem complexity scales with the machine. Here, the speedup is still dominated by N, assuming perfect scalability of the parallel part. However, this is hardly realistic; any collective communication operation in a parallel system like a global sum or a broadcast will inherently introduce costs beyond $O(1)$ and is, thus, not scalable. Building highly scalable interconnect fabrics with full flexibility of communication pattern is very expensive. This will, in most cases, significantly limit the scalability of the parallel portion of an application.

Thus, a different viewpoint to the question on how to reach Exascale might be taken by looking at applications and their inherent scalability. The latter will play a crucial role in exploiting the performance of such systems. Analyzing JSC's application portfolios one finds basically two classes of applications:

1. Highly scalable codes using regular communication patterns. These are the codes that are able to fully exploit JSC's BlueGene/P system.
2. Less scalable but significantly more complex codes. Most often these codes require complicated communication patterns. Their requirements constrain these applications to clusters.

A more detailed analysis of the second class unveils that among them are several applications that have highly scalable kernels, too. In principle they should be able to also exploit BlueGene-type of machines. Nevertheless, analogous to the serial work in Amdahl's Law, their scalability is limited by the least scalable kernel.

The second class includes many very important applications. It would be highly desirable to be able to use the performance of Exascale systems for them, too. Also, we do expect that even applications of the first class will be pushed "past the brink" by the extraordinary scale of parallelism on Exascale machines. Furthermore, the trend to cover more and more

aspects of a given scientific question in the simulation codes will increase complexity and is therefore likely to limit the scalability of the resulting codes. Last but not least, problems we do not face today due to their complexity might occur when Exascale systems become available. Most probably they are expected to fall into the second class too.

DEEP – A Highly Scalable Cluster Architecture

Tying it all together, cluster architectures viable for Exascale will have to provide sufficient flexibility to run complex applications with Exascale performance. To achieve this, we propose the Dynamic Exascale Entry Platform (DEEP) architecture as shown in Figure 1. The key idea is to provide two separate parts, each one optimized for one of the application classes described above. A general purpose cluster element consists of Intel Xeon nodes (CN) connected by a highly flexible switched network such as InfiniBand. An attached booster element comprises booster nodes (BN) connected by a highly scalable torus network optimized for extreme performance. Each booster node hosts a number of accelerators and enables them to boot autonomously and communicate with each other using the booster network. Booster Interface (BI) nodes connect the booster element directly with the cluster element's fabric. We have chosen the Intel® MIC architecture as the accelerator because of its flexibility, good performance, and power efficiency, and because it already uses a full OS kernel. The first prototypes of the DEEP architecture will use the 22 nm Knights Corner (KNC) implementation expected for a release in 2012.

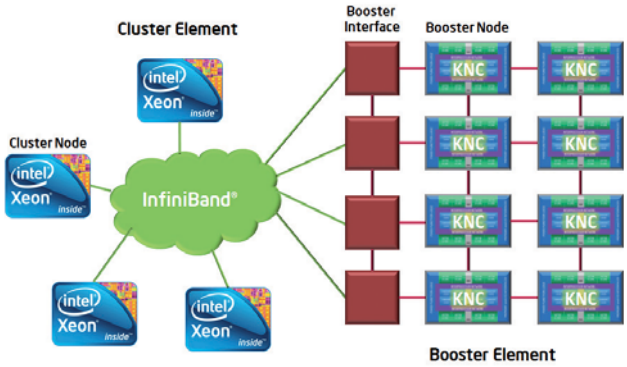


Figure 1: Sketch of the DEEP Architecture.

The proposed architecture provides several advantages:

- The cluster element can run complex application kernels with high per thread performance. It avoids restrictions on communication-patterns or highly irregular code sections common on today's dedicated MPP platforms.
- The booster element is able to run highly scalable kernels in the most energy efficient way. The limitations of today's accelerated clusters are avoided by enabling the compute elements, i.e., the accelerators in the BNs, to directly communicate with each other and leverage the full performance potential of the booster network.
- It cannot be expected that the ratio between the amount of work to be executed on the commodity CPU and the accelerator is fixed between different applications or even between different kernels of the same application. The DEEP architecture virtualizes the accelerator and enables dynamic allocation of CNs and BNs over the runtime of an application.
- At the same time the dynamic assignment of CNs and BNs improves resiliency, since failing nodes can be mapped out of the system without loss of functionality. Moreover, faulty BNs do not affect the CNs and vice versa.

In addition, the architecture supports the user in leveraging the high degree of parallelism in future machines. Today the type of kernels to be offloaded to accelerators is very limited, since accelerators typically do not support cross-node communication within a kernel efficiently. The DEEP Booster on the other hand supports more complex kernels including inter-node communication, as long as the patterns are regular enough and don't swamp the torus topology.

In this context the booster might be seen as highly scalable system on its own. When considering the highly scalable codes that are able to exploit BlueGene or QPACE today, it should be possible to run them on the booster alone with only minimal support from the Cluster element for I/O and similar tasks.

At the other extreme, BNs might be assigned to single CNs and used in the same fashion as in today's accelerated clusters. Both use cases — and anything in between — are possible without modification of the hardware concept, but are simply implemented by means of configuration on the level of system and application software.

Coping with Amdahl's Law

The DEEP architecture offloads kernels with high scalability (*O(N)* concurrency) onto the booster while leaving kernels with limited scalability (*O(k)* concurrency) on the cluster element. Let's assume a highly idealized situation where the code fraction with *O(N)* concurrency is *H* and the fraction of the *O(k)* concurrency is *(1−H)*. Given a number *h* of cores on the booster and a number of *f* cores on the cluster, and assuming the cluster's multi-core units to be a factor *c* times faster than the booster's many-core units, Amdahl's Law predicts the speedup *S* of the DEEP architecture as

$$S = \frac{1}{\frac{1-H}{fc} + \frac{H}{h}}$$

For a 10-PFlop/s booster with $h = 500000$, a cluster with $f = 10000$, an effective performance difference of a factor of 4 between CN and BN cores and a favorable concurrent *O(N)* code fraction of $H = 0.95$, the speedup *S* would reach a value of 320000 compared to a single core on the booster. This corresponds to an efficiency of 64%. It is evident that the efficiency strongly depends on the value of *H*. Since H is approaching 1 in a weak scaling scenario as advocated by Gustafson one can be optimistic to evade severe performance degradation that would be expected for *O(k)* concurrent kernels on the booster. Note that the communication between cluster and booster was not taken into account in this consideration. The optimal balance between *h* and *f* is specific for a given application. The DEEP architecture allows for adjusting this balance dynamically. The recipe to achieving scalability of applications on DEEP is of course to maximize *H* in the breakdown of the code into kernels while limiting data traffic as far as possible or even hiding communication between cluster and booster behind computation.

In summary, we expect that the proposed DEEP Architecture will demonstrate that it can not only reach the Exaflop/s level but more importantly, that applications can achieve unprecedented speed-ups and increases in performance.

The DEEP Software Stack

To benefit from the booster element of the DEEP architecture the programmer has to identify highly scalable kernels that can be offloaded to the booster. In a way this is similar to the identification of kernels to be offloaded into the GPGPUs of accelerated clusters today. DEEP does simplify this task since it provides much more flexibility — kernels can be more complex, can use cross accelerator communication, can adapt the number of BNs to the actual load, and can overlap communication between cluster and booster elements.

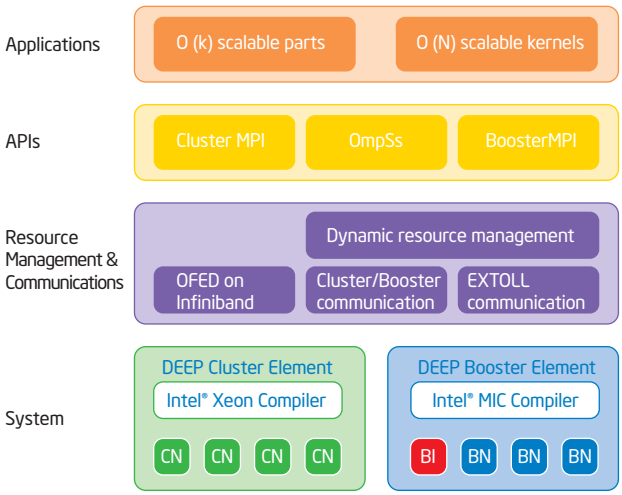


Figure 2: High Level View of the DEEP Software Stack

A DEEP software stack like the one shown in Figure 2 will support the user in expressing the different levels of parallelism in the application. Kernels of limited scalability remain on the cluster element and make use of a software stack of standard compilers, math libraries, thread packages and MPI. Highly scalable kernels will be offloaded to the booster element by a dynamic resource management component. The booster element is in effect a cluster of accelerators and will use its own software stack, again consisting of compilers, math libraries, thread packages, and a communication library. For the sake of simplicity, we assume that MPI will be used here. Finally, a glue library will enable transparent communication between the two elements.

The OmpSs API as implemented by the Barcelona Supercomputing Center [12] provides abstractions for defining and annotating tasks. It will be used to program both kinds of kernels at the thread level, and to annotate the scalability of the kernels and their data dependency. The DEEP runtime system will then use this information to allocate CNs and BNs as needed, offload tasks according to their characteristics, and manage the migration of data elements.

Conclusions and Outlook

We presented the innovative DEEP architecture that extends the classical clusters concept, leverages the performance of modern accelerators and coprocessors, and scales way beyond today's accelerated node clusters.

Within the DEEP project, co-funded by the European Community's Seventh Framework Programme under grant agreement no. 287530, the ExaCluster Lab will create a fully operational prototype of the DEEP architecture and its software stack based on Intel's KNC coprocessor.

References

[1] Top 500 Supercomputer Sites. November 11, 2011. www.top500.org

[2] G. Amdahl: Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. AFIPS Conference Proceedings. pp. 483-485, 1967

[3] The MPI Forum. MPI: A Message-Passing Interface Standard. 2009. 2009. www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf

[4] J. Rattner: SC'09 Conference Keynote. Portland, 2009

[5] G.E. Moore: Cramming more components onto integrated circuits. Electronics, Vol. 19, pp. 114-117

[6] Peter Kogge, et al. ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems. 2008. www.cse.nd.edu/Reports/2008/TR-2008-13.pdf

[7] Intel Many Integrated Core Architecture. www.many-core.group.cam.ac.uk/ukgpucc2/talks/Elgar.pdf

[8] L. Seiler, et al.: Larrabee: A Many-Core x86 Architecture for Visual Computing. ACM Transactions on Graphics, Vol. 27(3), pp. 18:3 - 18:15, 2008

[9] H. Baier, et al.: QPACE: power-efficient parallel architecture based on IBM PowerXCell 8i. Computer Science - R&D, Vol. 25, pp. 149-154, 2010

[10] Ranking the World's Most Energy-Efficient Supercomputers. www.green500.org

[11] J.L. Gustafson,.: Re-evaluating Amdahl's Law. Communications of the ACM, Vol. 31(5), pp. 532-533, 1988

[12] A. Duran, et al.: OmpSs: A Proposal for Programming Heterogeneous Multi-Core Architectures. Parallel Processing Letters, Vol. 21(2), pp. 173-193, 2011

Protein Folding with SMMP on Intel® MIC Architecture

Michael Kauschke[1,3]
Jochen Kreutz[1,2]
Jan H. Meinke[2]
Bart Stukken[1,3]
[1] Intel ExaCluster Lab Jülich
[2] Jülich Supercomputing Center, Forschungszentrum Jülich
[3] Intel GmbH

Proteins are long chains of amino acids. The sequence of amino acids determines a protein's native shape. The sequence is encoded in the genome and assembled by the ribosome (itself a complex of RNA and proteins) amino acid by amino acid.

Proteins need anywhere from a few micro seconds to several minutes to obtain their native structure. This process is called protein folding. It occurs reliably in our body many times each second, yet it is still poorly understood.

For some globular proteins it has been shown that they can unfold and refold in a test tube. At least for these proteins folding is driven purely by classical physical interactions. This is the basis for folding simulations using classical force fields.

The program package SMMP, first released in 2001 (see [1], [2], and [3]), implements Monte Carlo algorithms that can be used to study protein folding. It uses a simplified model of a protein that keeps the bond angles and lengths fixed and only allows for changes of the dihedral angles. To calculate the energy of a given conformation of a protein, SMMP also implements several energy functions, the so called force fields. A force field is defined by a functional form of the energy function and its parameterization. Here we'll consider the ECEPP/3 force field [4]. The effect of water on the protein is modeled implicitly with an energy term E_{sol} that is proportional to the solvent-accessible surface of each atom. The total energy is given by

$$E = E_C + E_{vdW} + E_{hb} + E_{tor} + E_{sol},$$

where E_C is the Coulomb energy between two charges, E_{vdw} defines the size and preferred distance between two atoms, E_{hb} models hydrogen bonding, and E_{tor} is a many-body term that ensures the right distribution of dihedral angles.

To explore the rough energy landscape and determine thermodynamic properties SMMP provides a number of advanced Monte Carlo methods, e.g., multicanonical Monte Carlo, simulated annealing, and parallel tempering also known as replica exchange. Each of these methods allows a more efficient sampling than a canonical Monte Carlo simulation.

During a canonical Monte Carlo simulation the energy of each new trial conformation is evaluated and compared with the energy of the last accepted conformation. The trial conformation is accepted if its energy is lower than the previous conformation. If the energy of the trial conformation is higher it can still be accepted. The probability P depends on the temperature T at which the simulation is performed.

$$P (E_0, E_1) = min (exp (- ß (E_0 - E_1)), 1.0)$$

One simple but effective way to improve sampling is parallel tempering. During a parallel tempering simulation the system is being simulated at several different temperatures simultaneously. After a number of Monte Carlo sweeps, where each sweep consists of n update attempts and n is the number of degrees of freedom, an exchange between different temperatures is attempted.

Assume that we have two temperatures T_i and T_{i+1}, where $T_i < T_{i+1}$ and two corresponding energies E_i and E_{i+1} then the two conformations are exchanged with probability

$$P_{PT} (i, i+1) = min (exp ((ß_i - ß_{i+1}) (E_i - E_{i+1})), 1.0)$$

This means that if $E_i > E_{i+1}$ the two conformations are exchanged and if $E_i < E_{i+1}$ an exchange becomes less and less likely the larger the difference in energy and temperature. At a given temperature this exchange is just another Monte Carlo move and the resulting chain of conformations can be analyzed as if it were a canonical simulation without any exchange of replica.

Algorithmic properties

The simulations at each temperature are nearly independent. Even for small proteins, the time needed for the parallel tempering move and the respective communications can be neglected compared to the effort for the individual tempering sweep. This makes parallel tempering a natural candidate for a parallel implementation.

Further, the energy function can be decomposed into independent but structurally equivalent partial sums (e.g., the interactions between specific atoms) making the effective use of wide SIMD units feasible.

Evaluation on the Intel® MIC architecture prototype

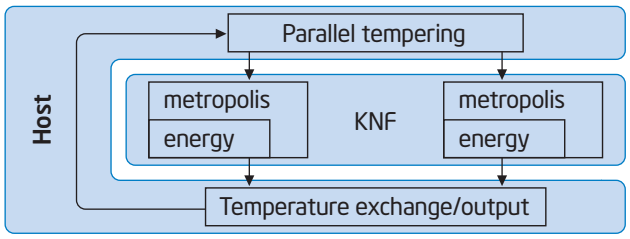
Based on the properties of the algorithm, implementing SMMP on a computing architecture with many rather simple cores that operate in parallel and contain wide SIMD units looks

promising. Knights Ferry (KNF), a software development platform for the upcoming Intel® Many Integrated Core (MIC) architecture is such a system, and this paper presents first results for an adapted version of SMMP. The work described here was performed within the ExaCluster Lab Jülich.

KNF implements 32 fully IA-compliant cores extended by a 512-bit wide SIMD unit that supports operations on up to 16 single precision numbers [7]. Most SIMD instructions are three-operand based, leading to improved throughput. Operands may be selected from a dedicated SIMD register file or a memory reference. A set of versatile gather/scatter operations in combination with masking write operations significantly improves applicability of the SIMD concept. MIC implements the standard Intel symmetric multi-processing (SMP) making direct execution of conventional multi-threaded programs (e.g., based on OpenMP, pthreads) written in C(++) or Fortran feasible. Physically KNF is implemented as a plugin card interfacing to the host via a 16x PCIe Gen2 port.

Partitioning SMMP for MIC

As said above, SMMP's performance is determined by rather regular functions that compute the energy for a given configuration. Thus in a naïve accelerator inspired approach, one would offload just these computations to KNF, while running the complex and partly serial sections of the Monte-Carlo sweep and the configuration exchanges on the host processor. However, the time necessary to transfer the actual protein configuration required by each call to the energy function through the PCIe port imposes a serious bottleneck. Instead the general purpose computing capabilities of KNF were exploited by moving the complete parallelizable sweep operation written in Fortran to KNF, while the sequential code required for the configuration exchanges is still executed on the host CPU:



Efficient Implementation of the Energy Functions

Efficient use of the KNF SIMD units is most critical for SMMP's performance. The following describes how the Fortran-based serial implementation of the energy functions can be mapped efficiently to KNF. All computations are performed in single precision (32-bit), with KNF in effect offering a 16 element wide SIMD unit.

The energy of a certain configuration is computed by two independent components. First the energy resulting from the interaction between the single atoms of the protein is computed while the interaction of the protein with its environment (E_{sol}) is modeled in a second step.

The contribution of a pair of atoms, *i* and *j*, is determined as a sum of the Coulomb ($E_{c,i,j}$), van der Waals ($E_{vdw,i,j}$) and hydrogen bond energy ($E_{hb,i,j}$) which are all a function of the Euclidean distance r_{ij} between the atoms. However, while the Coulomb component only depends on the distance and the respective charges, the other components are modeled by coefficients (a_{ij}, b_{ij},c_{ij},d_{ij}) depending on the respective types of the atoms:

$$E_{vdw,i,j} = a_{ij} / r_{ij}^{12} + b_{ij} / r_{ij}^{6}$$

$$E_{hb,i,j} = c_{ij} / r_{ij}^{12} + d_{ij} / r_{ij}^{10}$$

As only 18 different types of atoms are ever modeled, a single row of the two-dimensional coefficient array fits in two adjacent cache lines for a given i using single precision floats. Thus, using the **gather** operation of KNF enables efficient vectorization of the energy computation. **Gather** allows to load each way *k* of the 16 ways from a memory location M_k given by

$$M_k = Base+Offset_k$$

with *Offset* being a 16 wide integer vector. The respective pseudo code to compute the Van der Waals energy is shown below:

```
VdwVec=[ 0,...0] ;
for(i=0;i<x,i++){
 AtomTypeIVec=[ AtomType[ i]*RowSize, ...,
 AtomType[ i]*RowSize] ; for(j=i+1;j<x;j+=16){
  DVec=Dist(Point[ i] ,Point[ j:j+15] );
  IndexVec=AtomTypeIVec+VecLoad(AtomType+j);
  AijVec=gather(Aij,IndexVec);
  CijVec=gather(Cij,IndexVec);
  VdWVec+=Term(DVec,AijVec,CijVec);
}
VwdEnergy=AddReduce(VdwVec);
```

As for smaller-sized proteins, all required data resides in the L1 cache, a near optimum performance can be achieved with this approach.

Computation of the interaction with the solvent is more complex as it requires computation of the hull of the protein to determine the interacting surface. The relevant code section has the following structure:

```
n=0;
for(i=0,i<x;i++){
 j=Lut[ i] ;
 if(j!=k){
  D=Dist(Atom[ k] ,Atom[ j] );
  if(D<f(A[ j] )){
   n++;
   DStore[ n] =D;
}}
```

It contains two conditionals and requires storage in `DStore` in a "dense" fashion. However, efficient vectorization on KNF is still possible by careful use of masks and the **compress** operation:

```
for(i=0,i<x;i+=16){
 JVec=[ Lut+i,...,Lut+i+15] ;
 WMask=compare_not_equal(JVec,[ k,...,k] );
 AtomJVec=gather(WMask,Atoms,JVec);
 DVec=Dist(AtomKVec,AtomJVec);
 AVec=gather(WMask,A,JVec);
 FVec=f(AVec);
 LeMask=v_compare_less(WMask,DVec,FVec);
 compress(Dstore+n,LeMask,Dvec);
 n+=countbits(LeMask)
}
```

Conditionals are implemented by vectorized compare operations, which set mask registers controlling subsequent memory operations. Each way of the SIMD unit is associated to a respective bit in the controlling 16-bit wide mask register. If set, the respective operation is executed for the bespoke way, otherwise the operation is skipped. To store results to `DStore` a **scatter** operation, which is the inverse to **gather**, would not be efficient, as it requires a complex sequential computation of the respective mask and offset vectors beforehand. Instead a **compress** operation is used, which incrementally writes data of the "active" ways to locations starting at `DStore+n`. Still using a vectorized formulation gives the processor all options to perform operations in parallel and out of order, which significantly improves performance, especially when stores primarily hit the L1 and L2 caches.

Simulating GS-α_3W

As a test, we performed a parallel tempering simulation with 30 temperatures of the designed protein GS-α_3W [5], which one of us (JHM) had previously studied [6].

GS-α_3W is a designed three-helix bundle. It is 67 amino acids long and has about eleven hundred atoms (see Figure 1).

All replicas start with a random conformation. These conformations have very high energies since a random set of dihedral angles usually results in some atomic distances that are too small. A brief equilibration phase takes care of such overlaps and the different replicas start to explore the conformational space.

For replicas at temperatures below 550 K helices start to form. The fraction of residues that remains in a helical conformation quickly approaches that of the native state as temperature decreases. Once the three helices have formed they remain helices and only their relative orientation changes until they find the native conformation (cf. Figure 1).

During a parallel tempering simulation, each replica walks up and down the temperature ladder in a random fashion. This random walk enables the replica to explore low-energy conformations without getting stuck. Figure 2 shows the total energy vs. time for replica 26. The energy increases as it moves up the temperature ladder and decreases as it moves down again.

We created a graphical user interface (see Figure 3). It enables us to monitor the progress of the simulation by showing the history of conformations at a temperature and the energy. It also shows the status of all replicas.

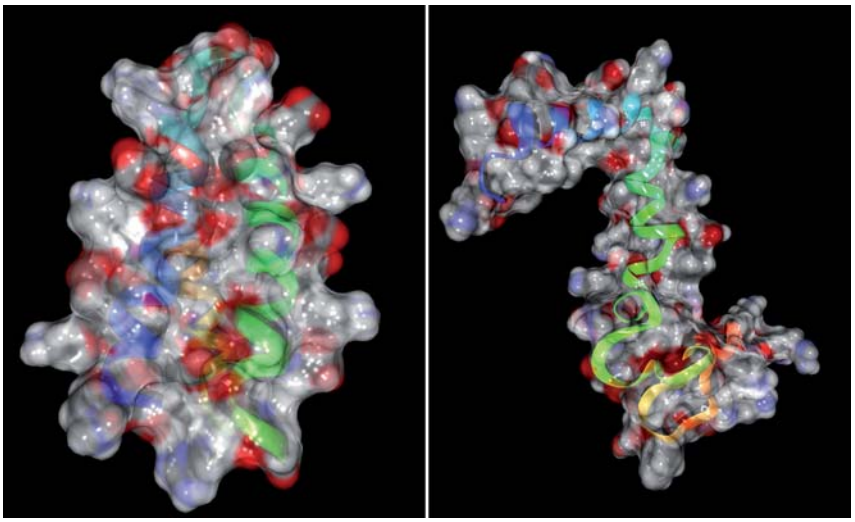


Figure 1: Three-helix bundle GS-a3W (PDB: 1LQ7). The left panel shows the experimentally determined structure. Note how compact the native structure is. The right panel shows a snapshot from our simulation. The helices are mostly formed but haven't found their native arrangement, yet.

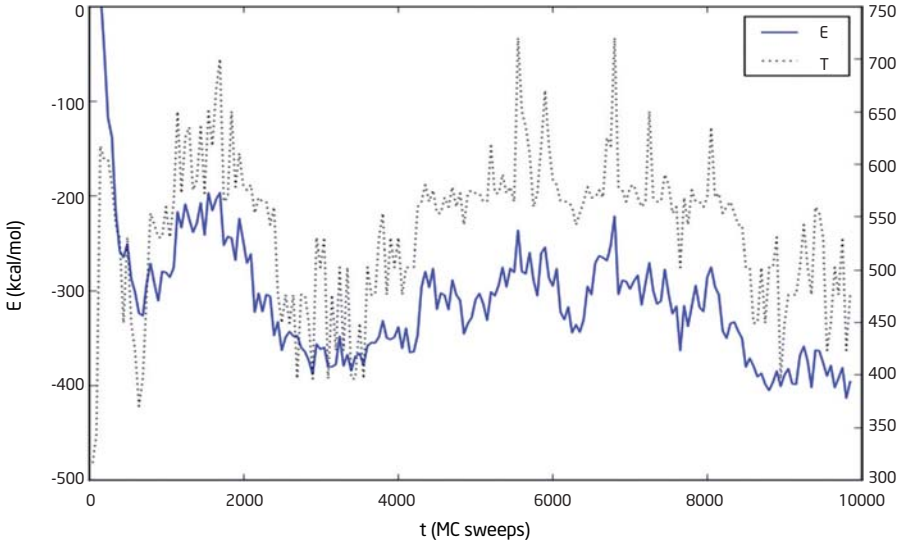


Figure 2: Energy of replica 26. The energy of each replica (solid blue line) increases and decreases as the replica moves up and down the temperature ladder (dotted black line).

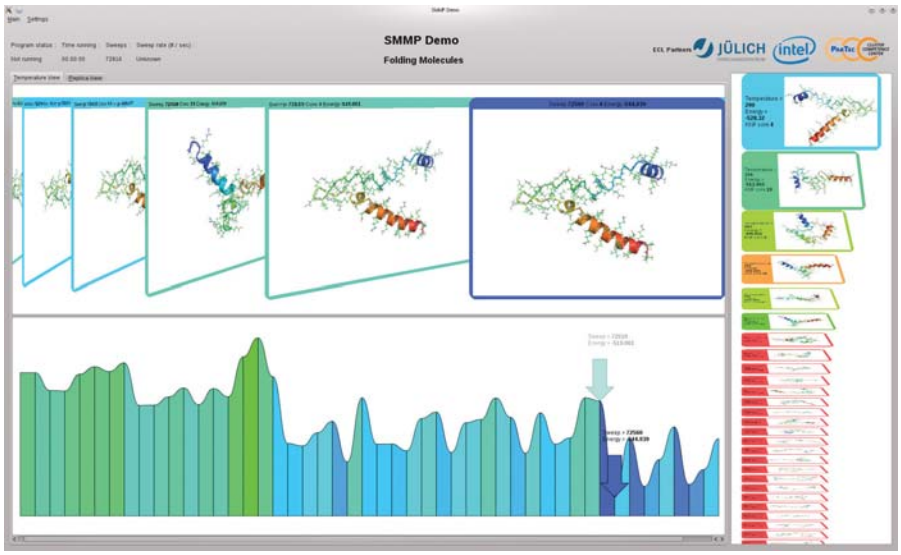


Figure 3: Graphical user interface for running SMMP on KNF. The screenshot shows the simulation of GS-α_3W with 30 replicas. The frames in the top panel show the history of conformation. The bottom panel shows the energy and the right panel shows the current set of conformations.

Conclusions

We implemented parallel tempering Monte Carlo simulation of proteins with the ECEPP/3 force field efficiently on Intel's KNF software development vehicle, which is the first implementation of the Intel® MIC architecture. Our implementation allows us to perform simulations on a single workstation or cluster node equipped with KNF that previously required a whole HPC cluster. The gather and scatter operations available on MIC enable the efficient usage of the wide vector units despite the irregular data access pattern of the energy calculation.

References

[1] F. Eisenmenger, et al.: A Modern Package for Simulation of Proteins. Computational Physics Communication, Vol. 138, pp. 192-212, 2001

[2] F. Eisenmenger, et al.: An Enhanced Version of SMMP-Open Source Software Package for Simulation of Proteins. Computational Physics Communication, Vol. 174, p. 422, 2006

[3] J.H. Meinke, et al.: SMMP v. 3.0—Simulating proteins and protein interactions in Python and Fortran. Computer Physics Communications, Vol. 178, pp. 459-470, 2008

[4] G. Nemethy, et al.: Energy parameters in polypeptides. 10. Improved geometrical parameters and nonbonded interactions for use in the ECEPP/3 algorithm, with application to proline-containing peptides. 15, 1992, Journal of Physical Chemistry, Vol. 96(15), pp. 6472-6484, 1992

[5] Q.-H. Dai, et al.: Structure of a de Novo Designed Protein Model of Radical Enzymes. Journal of the American Chemical Society, Vol. 124(37), pp. 10952-10953, 2002

[6] J.H Meinke and U.H.E. Hansmann: Free-energy-driven folding and thermodynamics of the 67-residue protein GS-α_3W - A large-scale Monte Carlo study. Journal of Computational Chemistry 30(11), pp. 1642-1648, 2009

[7] L. Seiler, et al.: Larrabee: A Many-Core x86 Architecture for Visual Computing. ACM Transactions on Graphics, Vol. 27(3), pp. 18:3 - 18:15, 2008

Sniper: Scalable and Accurate Parallel Multi-Core Simulation

Trevor E. Carlson[1,2]
Wim Heirman[1,2]
Lieven Eeckhout[2]
[1] Intel ExaScience Lab, Leuven
[2] ELIS Department, Ghent University

ABSTRACT

The Intel ExaScience Lab in Leuven, Belgium focuses on accurate space weather modeling, extremely scalable fault-tolerant simulation toolkits, in situ visualization through virtual telescopes, and architectural simulation of large-scale systems and workloads. This article describes recent advances achieved in the lab in architectural simulation of individual nodes in the Exascale era.

Two major trends in high-performance computing, namely, larger numbers of cores and the growing size of on-chip cache memory, are creating significant challenges for evaluating the design space of future processor node architectures. Fast and scalable simulations are therefore needed to sufficiently explore large multi-core systems within a limited simulation time and budget. By bringing together accurate high abstraction analytical models with fast parallel simulation, architects can trade off accuracy with simulation speed to run applications longer, covering a larger portion of the hardware design space. Interval simulation provides this balance, while still providing the detail necessary to observe core/uncore interactions across the entire system. Validations against real hardware show average absolute errors within 25% for a variety of multi-threaded workloads; more than twice as accurate on average than one-IPC simulation. Further, we demonstrate scalable simulation speed of up to 2.0 MIPS (million instructions per second) when simulating a 16-core system on an 8-core SMP machine.

1. INTRODUCTION

We observe two major trends in contemporary high-performance processors as a result of the continuous progress in chip technology through Moore's Law. First, processor manufacturers integrate multiple processor cores on a single chip — multi-core processors. Eight to twelve cores per chip are commercially available today (in, for example, Intel® Xeon® Processor E7-8800 Series), and projections forecast tens to hundreds of cores per chip in the near future, often referred to as many-core processors. In fact, the Intel® Knights Corner Many Integrated Core Architecture contains more than 50 cores on a single chip. Second, we observe increasingly larger on-chip caches. Multi-megabyte caches are becoming commonplace, exemplified by the 30MB L3 cache in the Intel® Xeon® Processor E7-8870.

These two trends pose significant challenges for the tools in the computer architect's toolbox. Current practice employs detailed cycle-accurate simulation during the design cycle [1, 5], which typically yields a simulation speed around tens to hundreds of KIPS (kilo instructions per second) for academic simulators; industry simulators are even more detailed and hence slower, typically in the 1 to 10 KHz range. While this has been (and still is) a successful approach for designing individual processor cores as well as multi-core processors with a limited number of cores, cycle-accurate simulation is not a scalable approach for simulating large multi-cores with tens or hundreds of cores, for two key reasons. First, current cycle-accurate simulation infrastructures are typically single-threaded. Given that clock frequency and single-core performance are plateauing while the number of cores increases, the simulation gap between the performance of the target system being simulated versus simulation speed is rapidly increasing. Second, the increasingly larger caches observed in today's processors imply that increasingly larger instruction counts need to be simulated in order to stress the target system in a meaningful way.

These observations impose at least two requirements for architectural simulation in the multi-core and many-core era. First, the simulation infrastructure needs to be parallel: the simulator itself needs to be a parallel application so that it can take advantage of the increasing core counts observed in current and future processor chips. A key problem in parallel simulation is to accurately model timing at high speed [7]. Advancing all the simulated cores in lock-step yields high accuracy; however, it also limits simulation speed. Relaxing timing synchronization among the simulated cores improves simulation speed at the cost of introducing modeling inaccuracies. Second, we need to raise the level of abstraction in architectural simulation. Detailed cycle-accurate simulation is too slow for multi-core systems with large core counts and large caches. Moreover, many practical design studies and research questions do not need cycle accuracy because these studies deal with system-level design issues for which cycle accuracy only gets in the way (i.e., cycle accuracy adds too much detail and is too slow, especially during the early stages of the design cycle).

Our work in the Intel ExaScience Lab deals with exactly this problem. Some of the fundamental questions we want to address are: What is a good level of abstraction for simulating future multi-core systems with large core counts and large caches? Can we determine a level of abstraction that offers both good accuracy and high simulation speed? Clearly, cycle-accurate simulation yields very high accuracy, but unfortunately, it is too slow. At the other end of the spectrum lies the one-IPC model, which assumes that a core's performance equals one Instruction Per Cycle (IPC) apart from memory accesses.

While both approaches are popular today, they are inadequate for many research and development projects because they are either too slow or have too little accuracy.

Figure 1 clearly illustrates that a one-IPC core model is not accurate enough. This graph shows CPI (Cycles Per Instruction) stacks that illustrate where time is spent for the SPLASH-2 benchmarks. We observe a wide diversity in the performance of these multi-threaded workloads. For example, the compute CPI component of radix is above 2 cycles per instruction, while radiosity and cholesky perform near the 0.5 CPI mark. Not taking these performance differences into account changes the timing behavior of the application and can result in widely varying accuracy. Additionally, as can be seen in Figure 2, simulated input sizes need to be large enough to effectively stress the memory hierarchy. Studies performed using short simulation runs (using the small input set) will reach different conclusions concerning the scalability of applications, and the effect on scaling of proposed hardware modifications, than studies using the more realistic large input sets.

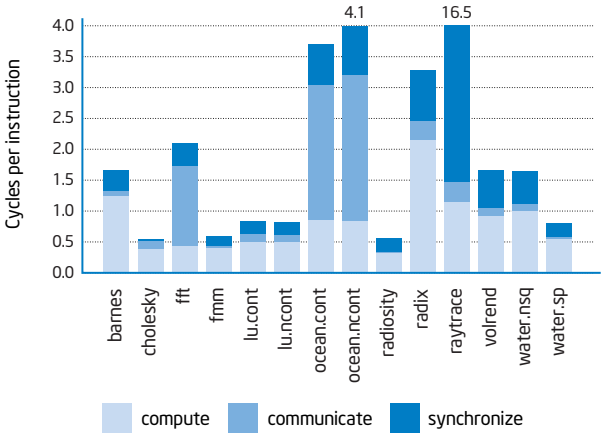


Figure 1: Measured per-thread CPI (average clock ticks per instruction) for a range of SPLASH-2 benchmarks, when running on 16 cores. (Given the homogeneity of these workloads, all threads achieve comparable performance.)

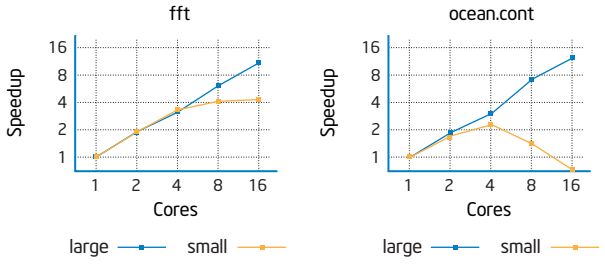


Figure 2: Measured performance of SPLASH-2 on the Intel® Xeon® Processor X7460 using large and small input sets.

The goal of our work is to explore the middle ground between the two extremes of detailed cycle-accurate simulation versus one-IPC simulation, and to determine a good level of abstraction for simulating future multi-core systems. To this end, we consider the Graphite parallel simulation infrastructure [6], and we implement and evaluate various high-abstraction processor performance models, ranging from a variety of one-IPC models to interval simulation [4], which is a recently proposed high-abstraction simulation approach based on mechanistic analytical modeling. In this process, we validate against real hardware using a set of scientific parallel workloads, and have named this fast and accurate simulator Sniper. We conclude that interval simulation is far more accurate than one-IPC simulation when it comes to predicting overall chip performance. For predicting relative performance differences across processor design points, we find that one-IPC simulation may be fairly accurate for specific design studies with specific workloads under specific conditions. In particular, we find that one-IPC simulation may be accurate for understanding scaling behavior for homogeneous multi-cores running homogeneous workloads. The reason is that all the threads execute the same code and make equal progress, hence, one-IPC simulation accurately models the relative progress among the threads, and more accurate performance models may not be needed. However, for some homogeneous workloads, we find that one-IPC simulation is too simplistic and does not yield accurate performance scaling estimates. Further, for simulating heterogeneous multi-core systems and/or heterogeneous workloads, one-IPC simulation falls short because it does not capture relative performance differences among the threads and cores.

This article presents a condensed description of our work. We refer the interested reader to the full paper published at Supercomputing'11 [2]. Further, we make the simulator publicly available at http://snipersim.org/ for free academic use.

2. PROCESSOR CORE MODELING

As indicated in the introduction, raising the level of abstraction is crucial for architectural simulation to be scalable enough to be able to model multi-core architectures with a large number of processor cores. The key question that arises though is: What is the right level of abstraction for simulating large multi-core systems? And when are these high-abstraction models appropriate to use?

There are currently a number of different levels of abstraction that exist to allow one to simulate the core of a microprocessor. The most detailed model, or cycle-accurate model, simulates the core in complete detail. Each hardware structure is modeled, allowing designers to accurately predict of the performance of a processor. Unfortunately, this detailed model also requires significant investment to develop as well as to simulate.

This section discusses higher abstraction processor core models, namely, the one-IPC model as well as interval simulation, that are more appropriate for simulating multi-core systems with large core counts.

2.1 One-IPC model

A widely used and simple-to-implement level of abstraction is the so-called "one-IPC" model. Many research studies assume a one-IPC model when studying for example memory hierarchy optimizations, the interconnection network and cache coherency protocols in large-scale multiprocessor and multi-core systems. A one-IPC model assumes in-order single-issue at a rate of one instruction per cycle, hence the name one-IPC or "one instruction per cycle". However, it simulates the memory hierarchy: in particular, an L1 instruction cache miss incurs a penalty equal to the L2 cache data access latency; an L2 cache miss incurs a penalty equal to the L3 cache data access latency, or main memory access time in the absence of an L3 cache.

2.2 Sniper: Interval simulation

Interval simulation is a recently proposed simulation approach for simulating multi-core and multiprocessor systems at a higher level of abstraction compared to current practice of detailed cycle-accurate simulation [4]. Interval simulation leverages a mechanistic analytical model to abstract core performance by driving the timing simulation of an individual core without the detailed tracking of individual instructions through the core's pipeline stages. The model is mechanistic in the sense that it was built from understanding the mechanisms in contemporary processors. The foundation of the model is that miss events (branch mispredictions, cache, and TLB misses) divide the smooth streaming of instructions through the pipeline into so called intervals [3]; see also Figure 3. Branch predictor, memory hierarchy, cache coherence and interconnection network simulators determine the miss events; the analytical model derives the timing for each interval. The cooperation between the mechanistic analytical model and the miss event simulators enables the modeling of the tight performance entanglement between co-executing threads on multi-core processors.

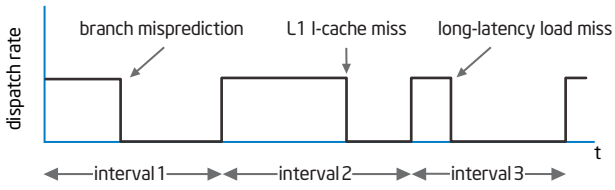


Figure 3: Interval modeling: miss events divide the smooth streaming of instructions into intervals.

The multi-core interval simulator models the timing for the individual cores. The simulator maintains a "window" of instructions for each simulated core. This window of instructions corresponds to the reorder buffer of a superscalar out-of-order processor, and is used to determine miss events that are overlapped by long-latency load misses. The functional simulator feeds instructions into this window at the window tail. Core-level progress (i.e., timing simulation) is derived by considering the instruction at the window head. In case of an I-cache miss, the core simulated time is increased by the miss latency. In case of a branch misprediction, the branch resolution time plus the front-end pipeline depth is added to the core simulated time, i.e., this is to model the penalty for executing the chain of dependent instructions leading to the mispredicted branch plus the number of cycles needed to refill the front-end pipeline. In case of a long-latency load (i.e., a last-level cache miss or cache coherence miss), we add the miss latency to the core simulated time, and we scan the window for independent miss events (cache misses and branch mispredictions) that are overlapped by the long-latency load — second-order effects. For a serializing instruction, we add the window drain time to the simulated core time. If none of the above cases applies, we dispatch instructions at the effective dispatch rate, which takes into account inter-instruction dependencies as well as their execution latencies. We refer to [4] for a more elaborate description of the interval simulation paradigm.

We added interval simulation into Graphite and named our version, with the interval model implementation, Sniper[1], a fast and accurate multi-core simulator. During the course of this work, we extended Sniper substantially over the original Graphite simulator. Not only did we integrate the interval simulation approach, we also made a number of extensions that improved the overall functionality of the simulator. In particular, we added a shared multi-level cache hierarchy supporting write-back first-level caches and an MSI snooping cache coherency protocol. In addition to the cache hierarchy improvements, we also added branch predictor models. Further, we implemented a kernel lock contention model, introducing the concept of a rescheduling cost. This cost advances simulated time each time a thread enters the kernel to go into a wait state, and later needs to be rescheduled once it is woken up.

2.3 Interval simulation versus one-IPC

There are a number of fundamental differences between interval simulation and one-IPC modeling.

- Interval simulation models superscalar out-of-order execution, whereas one-IPC modeling assumes in-order issue, scalar instruction execution. More specifically, this implies that interval simulation models how non-unit instruction execution latencies due to long-latency instructions such as multiplies, divides and floating-point operations as well as L1 data cache misses, are (partially) hidden by out-of-order execution.
- Interval simulation includes the notion of instruction-level parallelism (ILP) in a program, i.e., it models inter-instruction

[1] *The simulator is named after a type of bird called a snipe. This bird moves quickly and hunts accurately.*

dependencies and how chains of dependent instructions affect performance. This is reflected in the effective dispatch rate in the absence of miss events, and the branch resolution time, or the number of cycles it takes to execute a chain of dependent instructions leading to the mispredicted branch.

- Interval simulation models overlap effects due to memory accesses, which a one-IPC model does not. In particular, interval simulation models overlapping long-latency load misses, i.e., it models memory-level parallelism (MLP), or independent long-latency load misses going off to memory simultaneously, thereby hiding memory access time.
- Interval simulation also models other second-order effects, or miss events hidden under other miss events. For example, a branch misprediction that is independent of a prior long-latency load miss is completely hidden. A one-IPC model serializes miss events and therefore overestimates their performance impact.

Because interval simulation adds a number of complexities compared to one-IPC modeling, it is slightly more complex to implement, hence, development time takes longer. However, we found the added complexity to be limited: the interval model contains only about 1000 lines of code.

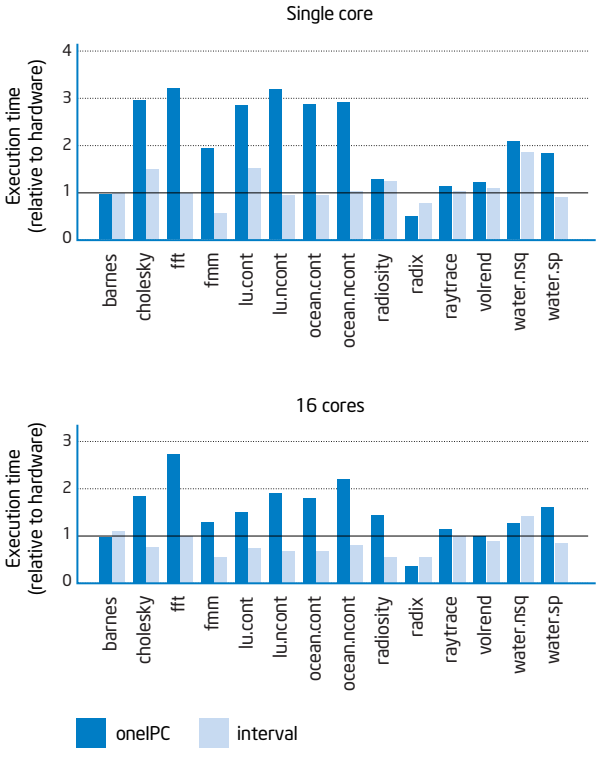


Figure 4: Relative accuracy for the one-IPC and interval models for a single core (top graph) and 16 cores (bottom graph).

3. SNIPER ACCURACY AND SPEED

We now evaluate Sniper interval simulation and compare its accuracy and speed against one-IPC model. We compare the absolute accuracy of the simulation models and then compare scaling of the benchmarks as predicted by the models and hardware. Additionally, we show a collection of CPI stacks as provided by interval simulation. Finally, we report Sniper's simulation speed.

3.1 Core model accuracy

The hardware that we validate against is a 4-socket Intel® Xeon® processor X7460 based shared-memory machine. Each Intel® Xeon® X7460 processor chip integrates six cores, hence, we effectively have a 24-core SMP machine to validate against. Each core is based on the 45 nm Penryn microarchitecture, and has private L1 instruction and data caches. Two cores share the L2 cache, hence, there are three L2 caches per chip. The L3 cache is shared among the six cores on the chip.

Figure 4 shows accuracy results of interval simulation compared with the one-IPC model given the same memory hierarchy modeled after the Intel Xeon processor X7460 based machine. We find that the average absolute error is substantially lower for interval simulation than for the one-IPC model in a significant majority of the cases. The average absolute error for the one-IPC model using the large input size of the SPLASH-2 benchmark suite is 114% and 59.3% for single and 16-threaded workloads, respectively. In contrast, the interval model compared to the Intel Xeon processor X7460 based machine has an average absolute error of 19.8% for one core, and 23.8% for 16 cores. Clearly, interval simulation is substantially more accurate for predicting overall chip performance than the one-IPC model; in fact, it is more than twice as accurate.

3.2 Application scalability

So far, we focused on absolute accuracy, i.e., we evaluated accuracy for predicting chip performance or how long it takes for a given application to execute on the target hardware. How-

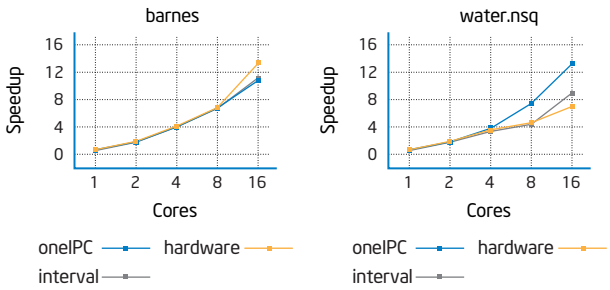


Figure 5: Application scalability for the one-IPC and interval models when scaling the number of cores.

ever, in many practical research and development studies, a computer architect is more interested in relative performance trends in order to make design decisions, i.e., a computer architect is interested in whether and by how much one design point outperforms another design point. Similarly, a software developer may be interested in understanding an application's performance scalability rather than its absolute performance. Figure 5 shows such scalability results for a select number of benchmarks. A general observation is that both interval and one-IPC modeling is accurate for most benchmarks (not shown here for space reasons), as exemplified by barnes (left graph). However, for a number of benchmarks, see the right graph for water.nsq, the interval model accurately predicts the scalability trend, which the one-IPC model is unable to capture. It is particularly encouraging to note that, in spite of the limited absolute accuracy for water.nsq, interval simulation is able to accurately predict performance scalability.

3.3 CPI stacks

A unique asset of interval simulation is that it enables building CPI stacks which summarize where time is spent. A CPI stack is a stacked bar showing the different components contributing to overall performance. The base CPI component typically appears at the bottom and represents useful work being done. The other CPI components represent "lost" cycle opportunities due to inter-instruction dependencies, and miss events such as branch mispredictions, cache misses, etc., as well as waiting time for contended locks. A CPI stack is particularly useful for gaining insight in application performance. It enables a computer architect and software developer to focus on where to optimize in order to improve overall application performance. Figure 6 shows CPI stacks for all of our benchmarks.

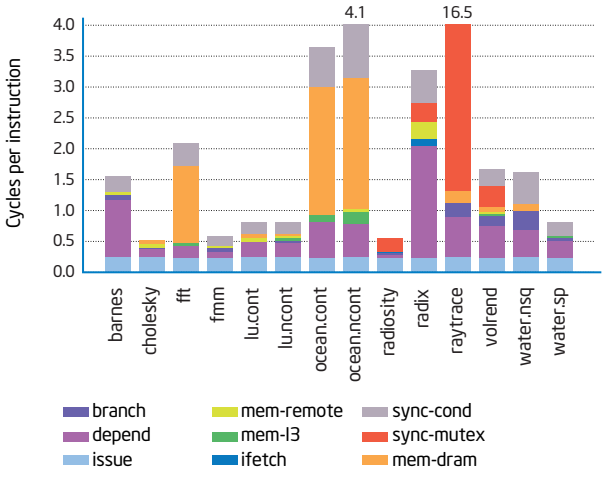


Figure 6: Detailed CPI stacks generated through interval simulation.

As one example of how a CPI stack can be used, we analyzed the one for raytrace and noted that this application spends a huge fraction of its time in synchronization. This prompted us to look at this application's source code to try and optimize it. It turned out that a pthread_mutex lock was being used to protect a shared integer value (a counter keeping track of global ray identifiers). In a 16-thread run, each thread increments this value over 20,000 times in under one second of run time. This results in a huge contention of the lock and its associated kernel structures. By replacing the heavy-weight pthread locking with an atomic lock inc instruction, we were able to avoid this overhead. Figure 7 shows the parallel speedup (left) and CPI stacks (right) for raytrace before and after applying this optimization.

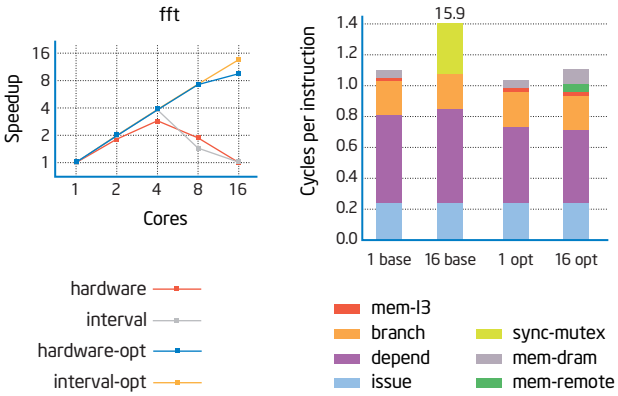


Figure 7: Speedup (left) and CPI stacks (right) for raytrace, before and after optimizing its locking implementation.

3.4 Simulation speed and complexity

As mentioned earlier, the interval simulation code base is quite small; consisting of about 1000 lines of code. Compared to a cycle-accurate simulator core, the amount of code necessary to implement this core model is several orders of magnitude less, a significant development savings. A comparison of the simulation speed between interval simulation and one-IPC modeling can be found in Figure 8. These runs were performed on dual-socket Intel® Xeon® Processor L5520 machines with a total of eight cores per machine. When simulating a single thread, the interval model is about 2–3 times slower than the one-IPC model. But when scaling up the number of simulated cores, parallelism can be exploited and aggregate simulation speed goes up significantly — until we have saturated the eight cores of our simulation host machine. Also, the relative computational cost of the core model quickly decreases, as the memory hierarchy becomes more stressed and requires more simulation time. From eight simulated cores onwards, on an eight-core host

machine, the simulation becomes communication-bound and the interval core model does not require any more simulation time than the one-IPC model, while still being about twice as accurate.

Note that the simulation speed and scaling reported for the original version of Graphite [6] are higher than the numbers we show for Sniper in Figure 8. The main difference lies in the fact that Graphite, by default, only simulates private cache hierarchies. This greatly reduces the need for communication between simulator threads, which enables good scaling. For this study, however, we model a realistic, modern CMP with large shared caches. This requires additional synchronization, which affects simulation speed.

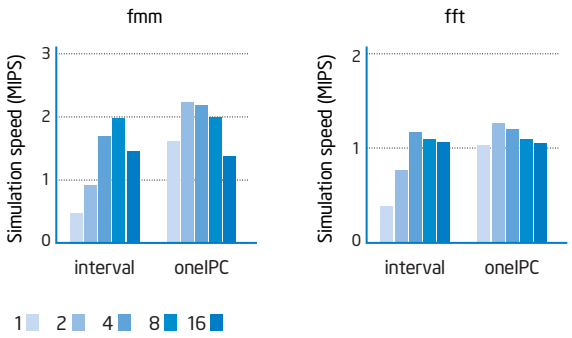


Figure 8: Simulation speed of 1–16 simulated cores on an eight-core host machine.

4. SUMMARY

Exploration of a variety of system parameters in a short amount of time is critical to determining successful future architecture designs. With the ever growing number of processors per system and cores per socket, there are challenges when trying to simulate these growing system sizes in reasonable amounts of time. Compound the growing number of cores with larger cache sizes, and one can see that longer, accurate simulations are needed to effectively evaluate next-generation system designs. But, because of complex core/uncore interactions and multi-core effects due to heterogeneous workloads, realistic models that represent modern processor architectures become even more important. In this work, we present the combination of a highly accurate, yet easy-to-develop core model, the interval model, with a fast, parallel simulation infrastructure. This combination provides accurate simulation of modern computer systems with high performance, up to 2.0 MIPS.

Even when comparing a one-IPC model that is able to take into account attributes of many superscalar, out-of-order processors, the benefits of the interval model provide a key simulation trade-off point for architects. We have shown a 23.8% average absolute accuracy when simulating a 16-core Intel® Xeon® Processor X7460-based system; more than twice that of our one-IPC model's 59.3% accuracy. By providing a detailed understanding of both the hardware and software, and allowing for a number of accuracy and simulation performance trade-offs, we conclude that interval simulation and Sniper is a useful complement in the architect's toolbox for simulating high-performance multi-core and many-core systems.

5. REFERENCES

[1] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The M5 simulator: Modeling networked systems. IEEE Micro, 26:52–60, 2006

[2] T. E. Carlson, W. Heirman, and L. Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Nov. 2011. © 2011 ACM, Inc

[3] S. Eyerman, L. Eeckhout, T. Karkhanis, and J. E. Smith. A mechanistic performance model for superscalar out-of-order processors. ACM Transactions on Computer Systems (TOCS), 27(2):42–53, May 2009

[4] D. Genbrugge, S. Eyerman, and L. Eeckhout. Interval simulation: Raising the level of abstraction in architectural simulation. In Proceedings of the 16th IEEE International Symposium on High-Performance Computer Architecture (HPCA), pages 307–318, Feb. 2010

[5] M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. ACM SIGARCH Computer Architecture News, 33(4):92–99, Nov. 2005

[6] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald III, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal. Graphite: A distributed parallel simulator for multicores. In Proceedings of the 16th IEEE International Symposium on High-Performance Computer Architecture (HPCA), pages 1–12, Jan. 2010

[7] S. K. Reinhardt, M. D. Hill, J. R. Larus, A. R. Lebeck, J. C. Lewis, and D. A. Wood. The Wisconsin Wind Tunnel: Virtual prototyping of parallel computers. In Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pages 48–60, May 1993

This work is based on an earlier work: "Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-Core Simulation," in SC '11 Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis © ACM, 2011.
http://doi.acm.org/10.1145/2063384.2063454

Space Weather Prediction with Exascale Computing

Giovanni Lapenta[1,2]
[1] Intel ExaScience Lab, Leuven
[2] Centrum voor Plasma-Astrofysica, Departement Wiskunde, KU Leuven.

The Intel ExaScience Lab in Leuven, Belgium focuses on accurate space weather modeling, extremely scalable fault-tolerant simulation toolkits, in situ visualization through virtual telescopes, and architectural simulation of large-scale systems and workloads. This article describes recent advances achieved in the lab in the application of high performance computing applied to space weather modeling.

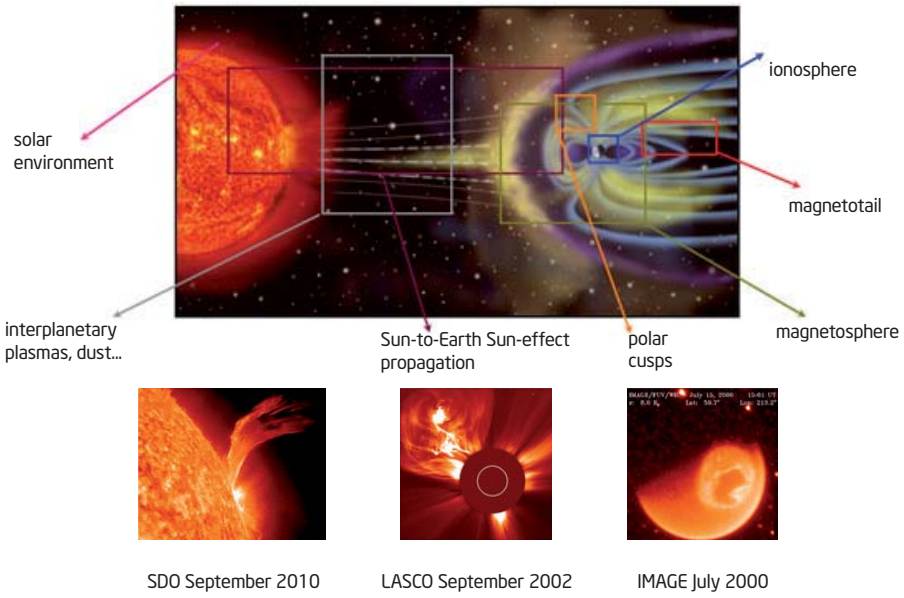
Space weather refers to conditions on the Sun, in the interplanetary space and in the Earth space environment that can influence the performance and reliability of space-borne and ground-based technological systems and can endanger human life or health. Adverse conditions in the space environment can cause disruption of satellite operations, communications, navigation, and electric power distribution grids, leading to a variety of socioeconomic losses. The conditions in space are also linked to Earth's climate. The activity of the Sun affects the total amount of heat and light reaching the Earth and the amount of cosmic rays arriving in the atmosphere, a phenomenon linked with the amount of cloud cover and precipitation. Given these great impacts on society, space weather is attracting a growing attention and is the subject of international efforts worldwide.

We focus here on the steps necessary for achieving a true physics-based ability to predict the arrival and consequences of major space weather storms. Great disturbances in the space environment are common but their precise arrival and impact on human activities varies greatly. Simulating such a system is a grand challenge, requiring computing resources at the limit of what is possible not only with current technology but also with the foreseeable future generations of super computers.

Space Weather and the Exascience Lab

In the ExaScience Lab we use the space weather simulation as a representative example for a large class of applications (CFD, MHD, and so on) and we analyze the requests on Exascale systems. The actors of space weather are the Sun, the Earth and the vast space in between. Like any star, the sun is made of a highly energetic and highly conductive gas, called plasma. In plasmas, the atoms have been broken into their nuclei and their electrons that become free to move. The hot plasma of the Sun is confined by gravity and moves in complex patterns that produce large currents and large magnetic fields. The gravitational confinement is not perfect and a highly varying outflow of plasma, called solar wind, is emitted from the Sun to permeate the whole solar system, reaching the Earth. The Earth has itself a magnetic field. The Earth magnetic field makes compasses point towards the north pole and allows many species of animals to find their way during migrations. This very same field protects the Earth from the incoming solar wind and its disturbances. Only a small part of the particles can reach the Earth surface, the so-called cosmic rays. Most of the incoming plasma is stopped and deflected, reaching only high strata of the atmosphere at the polar regions and causing the aurora seen by the peoples of the northern latitudes as the northern lights.

Space weather simulations must follow the plasma emitted



The Sun emits a constant solar wind in the form of a hot ionized gas and a magnetic field. The Solar wind interacts with the Earth magnetic field forming an elongated magnetosphere that protects the Earth environment from many of the most devastating consequences. Strong perturbations and high energy particles still penetrate creating serious trouble in space and on Earth.

Figure 1: Different phases of a space weather event.
Top: Artist view from NASA
Bottom: Images from different past space missions.

from the Sun and track its evolution in the interplanetary space, focusing on its impact on the Earth. The different phases of the processes are described in Fig.1.

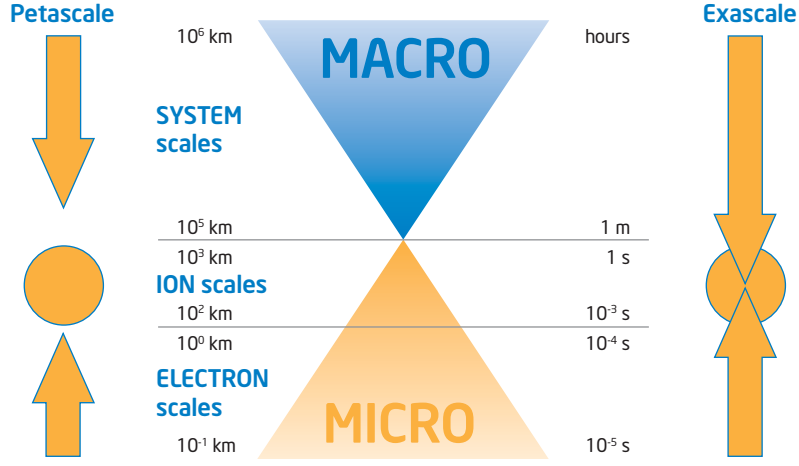
To model such processes, both electromagnetic fields and plasma particles need to be described. The nuclei (mostly protons, the nuclei of hydrogen) and the electrons of the plasma are coupled together loosely and each species has its own typical scales. The electromagnetic fields keep the species coupled forming a very non-linear and multi-scale system.

Modeling space weather is a daunting task: daunting because the system is enormous and because it includes a wide variety of physical processes and of time and space scales. Fig. 2 shows the typical scales observed from space exploration missions in the Earth environment. The scales are presented in the form of a hourglass with the top part presenting the macroscopic scales of evolution and at the bottom the smallest scales of importance.

A computer model of space weather must face this challenge by using state of the art mathematical techniques to deal with non-linear multiscale systems. The ExaScience Lab relies on the implicit moment method, intuitively described in Fig. 3. The fields and the particles are studied together in a coupled manner. The word "implicit" refers to the ability of the method to advance both fields and particles together without any lag between the two (the time lag is a typical aspect of the explicit methods, instead). The word "moment" refers to the use of moments of the particle statistical distribution. The moments are local statistical averages that characterize the particle properties.

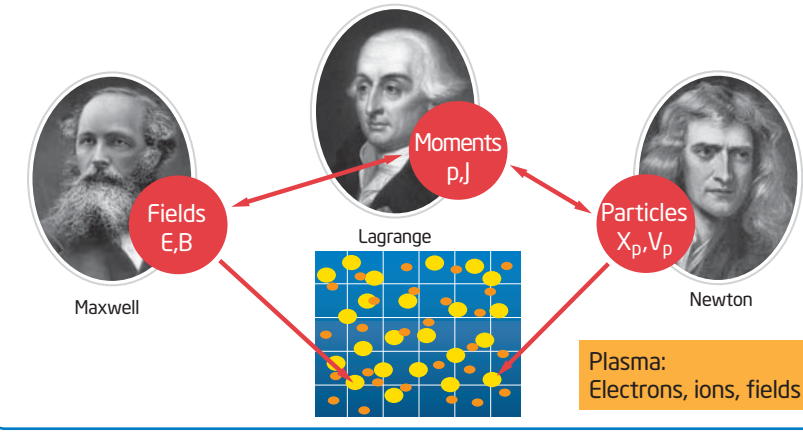
The implicit moment method has been developed at Los Alamos by Jerry Brackbill in the 80's and the author has been the leader of the development effort at Los Alamos during the last decade, before moving to the present effort. At Leuven, the new code iPic3D has been developed by Stefano Markidis and the author and is in use and under further development by the ExaScience Lab.

The implicit moment method allows us to select the local level of resolution according to the scales of the local processes. This feature allows to model space weather events with the minimum effort, increasing the resolution only where absolutely needed. Even with this method, however, the current petascale computers can only model subsections of a typical space weather event. With existing resources, reduced models need to be used. Some of the processes are neglected and approximated with adjustable parameters and heuristic assumptions, reducing the predictive value of the approach. To achieve a truly predictive model of space weather events Exascale computing will be needed.



Typical scales observed during missions of exploration in the Earth space environment. The spatial scales are on the left and the temporal scales are on the right. Space plasmas are made of electrons and ions (mostly protons, the nuclei of hydrogen). Electrons are much lighter and their scales are orders of magnitude smaller. A tremendous spread is present requiring advanced modelling techniques and the largest computing resources conceivable.

Figure 2: the physical scales of the Earth space environment



The plasma particles (electrons in orange and ions in yellow) evolve in a grid for the electric and magnetic fields. The Maxwell equations controls the fields and the Newton equations the particle The Implicit Moment Method introduces the moments in Lagrangian form to handle the multiple scales and to allow the user the desired accuracy.

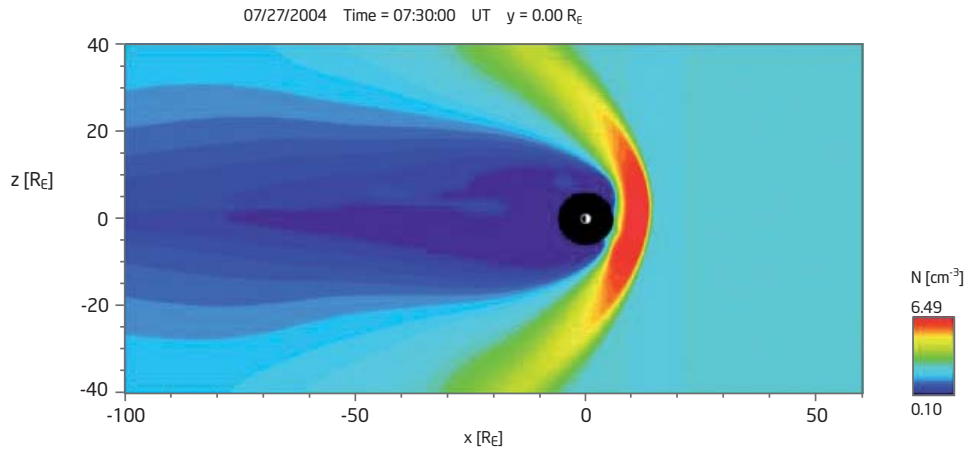
Figure 3: Agents in the implicit moment method

Space Weather Modelling on Exascale

Figure 4 illustrates the current situation of the state of the art in space weather modeling. The full description based on physically sound first principles requires one to resolve the electron scales in very localized regions of the system where energy conversion processes develop. In the figure, we report a state of the art simulation based on existing limited models that do not capture the full physics. To reach truly predictive capability Exascale resources will be needed.

A simple calculation provides the reason. Missions of exploration and models provide a clear indication of the scales involved (see Figs. 1–2). The needed simulation box for the modeling the Earth environment must be of the order of 100 Earth radii in each of the 3 spatial direction ($100R_E$ x $100R_E$ x $100R_E$) where the Earth radius is 6353Km. As observed in Fig.2, the smallest spatial scales are the electrostatic responses (called Debye length) at about 100m. Clearly simulating the whole box of size of more than 600 thousand kilometers with a resolution of 100 meters will require impossible resources (for the foreseeable future). Yet that is exactly what the most common methods in use would need to do. The so-called explicit methods need to resolve the smallest electrostatic responses and the table in Fig. 4 reports the impractical requirements of such a calculation. Our experience leads us to assume that a single core of current design can fit at most 16 x 16 x 16 cells each with a hundreds of particles for each species (electrons and ions). With that assumption a current state of the art explicit code would require 63 millions of billions of cores. Not even Exascale can provide that.

At the ExaScience Lab, however, the focus is on implicit moment methods that allow to select the local accuracy. Two steps can then be followed. First, just the use of the implicit method allows the user to avoid the need to resolve the electrostatic scales. Instead, the method averages over them and can allow the resolution to be increased to the electron electromagnetic response (electron inertial range), at about 10Km. This mathematical modeling approach saves then 2 orders of magnitude in each direction and similarly allows a corresponding reduction of time steps needed. The user can rise the bar on the hourglass of Fig. 2 by two notches in space and in time. Keeping the bar horizontal, the implicit moment method allows to rise the resolution in space and in time concurrently. A uniform grid with the implicit moment method still requires 63 billions of cores. This is closer to being within reach but it is still too large a number. This resolution remains still outside the reach of even future Exascale computers.



METHOD	CELLS PER DIMENSION	CORES
Explicit Approach	6,353,000	6.2600e+16
Implicit Moment Method	63,530	6.2600e+10
Implicit Moment Method with AMR	Coarse: 635 Adaptivity on localized regions	Coarse level: 6.2600e+04 Refinement: 1.9812e+06

On petascale (order 100,000 cores) supercomputers, it is not possible to simulate space weather events with full resolution even with the most advanced modelling method. Exascale will instead allow a fully resolved model.

Figure 4: Physics-based model of space weather.
Cut of a 3D simulation with OpenGGCM done by the Author at the facilities of the CCMC at NASA Goddard. The regions requiring additional resolution are identified with arrows. Only a very thin layer in ach place needs to be resolved.

A second and final step is taken in our approach: adaptive grids. The regions of dissipation where the simulations truly need to resolve the electron dynamics happen on very limited regions (see Fig. 4) of reduced dimensionality: thin crusts over surfaces dividing different types of plasma (for example the solar wind from the Earth magnetosphere). A reasonable assumption guided by practical experience is to use a layer of 10 cores away from the surfaces of interest. Additionally, we assume the impacted surfaces to be about 10% of a sphere encircling the Earth at the distances of interaction with the solar wind, an estimate can be computed of the total number of cores needed. For the large scale modeling of the wholes system, resolving just the ion scales, a very manageable number of cores is needed: 62 thousand. This is well within reach of petascale computing and indeed currently this is the state of the art. But to resolve the electron physics and avoid ad hoc assumptions that limit the predictive value of the simulations, the AMR approach will need an additional 1.98 million cores. All together the first ever predictive simulation based on physics principles will require about 2 million cores and will be feasible on the first Exascale computer.

References

[1] D.N. Baker: How to cope with Space Weather, Science, 297, 1486, 10.1126/science.1074956, 2002

[2] V. Bothmer and I.A. Daglis: Space Weather – Physics and Effects, ISBN 978-3-540-23907-9. Published by Praxis Publishing, Chichester, UK, and Springer Science+Business Media, LLC, New York, NY USA, 2007

[3] J.U. Brackbill and D.W. Forslund: An implicit method for electromagnetic plasma simulation in two dimensions, Journal of Computational Physics, 46, 271, ISSN 0021-9991, 10.1016/0021-9991(82)90016-X, 1982

[4] G. Lapenta, J. Brackbill, and P. Ricci, 2006, Kinetic approach to microscopic-macroscopic coupling in space and laboratory plasmas. Phys. Plasmas 13, 055904, 10.1063/1.2173623, 2006

[5] G. Lapenta: Particle simulations of space weather, Journal of Computational Physics, ISSN 0021-9991, 10.1016/j.jcp.2011.03.035, 2011

[6] S. Markidis, G. Lapenta, and Rizwan-uddin: Multi-scale simulations of plasma with iPIC3D, Mathematics and Computers in Simulation, 80, 1509, ISSN 0378-4754, 10.1016/j.matcom. 2009.08.038, 2010

Multi-scale Polarizable Microscopic Molecular Simulations: Towards Massive Free Energy Computations

Michel Masella
Philippe Cuniasse
Commissariat à l'Énergie Atomique et aux Énergies Alternatives
Life Science Division – Service of Molecular Protein Engineering.

Two main families of theoretical approaches are commonly used in today's research to simulate a molecular system at the atomic scale: quantum methods, based on solving the Schrödinger equation, and molecular modeling techniques, based on a classical formalism. The latter approaches are more than 1000x efficient more efficient compared to the quantum ones, which explains their intensive use to simulate microscopic systems on "long" time intervals (from the nano to the millisecond, this is achieved by solving the Newtonian equations of motion, the core of molecular dynamics methods).

However, molecular modeling approaches are based on empirical sets of parameters (the so-called molecular modeling force-field), whose quality is pivotal for computing reliable "trajectories" of a microscopic system (a trajectory is the result of a molecular dynamics simulation, i.e. a set of structures of the simulated microscopic system obtained by solving the Newtonian equations of motion). Since the first attempts in 1970 of simulating microscopic systems at the atomic scale, huge efforts involving hundreds of research teams world wide have been devoted to improve the "force field" accuracy, to develop accurate and efficient simulation algorithms (such as algorithms allowing to model a system at constant temperature and pressure) and to implement them efficiently on modern computational architectures. For instance, among all the advances made in the recent years, we may cite the achievement of the first generation of polarisable force fields (which allow us to account in part for the fluctuations of the electronic cloud corresponding to the simulated microscopic system, a pure quantum phenomenon), the development of multi-scale simulation protocols allowing us to model efficiently some particular degrees of freedom of the simulated system (usually, those corresponding to the solvent, which are the main bottleneck of molecular dynamics simulations in the biochemistry field), as well as new methods to compute thermodynamics quantities (such as the free energy, a quantity allowing a direct comparison with experiment).

From a purely computational point of view, implementing molecular modeling/molecular dynamics protocols efficiently and in a parallel way is a major challenge, because of the inherent efficiency of the physical models they consider. In particular, the simulation times, which can be achieved on modern super computer systems, do not depend on the size of the simulated systems. Obviously, the larger is a molecular system the larger is the number of processors, which can be used efficiently to simulate it. However, whatever the molecular system size, there is a limiting number of processors beyond which no more speed up is observed for the computations. This means that even if we can access "infinite" computational resources, we are not presently able to simulate at the day/week scale, a molecular system at a larger time scale than the µs one, whatever the size of the molecular system.

The latter limitation however, does not represent a noticeable drawback for a particular application field of molecular modeling/molecular dynamics, i.e. the computation of differences in free energy, the so-called ΔΔG's. In Figure 1, the interactions of two close potential drugs (ligands) with the protein serine protease factor Xa (fXa) are represented [1]. The protein fXa is a key enzyme in the activation cascade of the blood coagulation, as such it is inferred to be a target for the therapy of thrombosis-related diseases, a major cause of mortality in western countries.

The two ligands differ only by a reduced set of atoms, however, their relative affinities (which can be measured in terms of ΔΔG) for the protein fXa differ by a factor 4. This difference has a practical significance for drug development but corresponds to a very small difference in ΔΔG. Hence, being able to compute accurately and efficiently the ΔΔG's for ligand sets containing hundreds or thousands of compounds is expected to have a strong impact in the pharmaceutical field, in particular by reducing the cost of drug development.

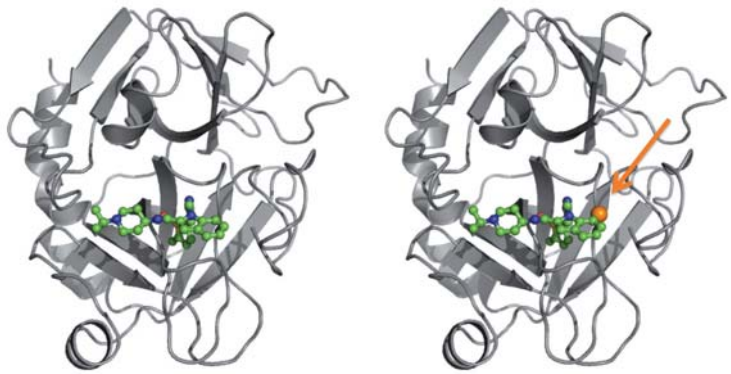


Figure 1. The protein fXa (shown in gray) and two potent ligand molecules, differing by only fours atoms (a methyl group CH_3), whose is represented by an orange sphere on the right (the ligands have been developed by the pharmaceutical group Sanofi-Aventis). The latter ligand is four times more potent compared to the one represented on the left.

Computing ΔΔG's based on molecular modeling techniques requires the simulation of a set of virtual molecular systems (typically 10 and above), intermediate between the two real systems under investigation. Our team is presently developing accurate polarisable force fields, which can be used in conjunction with a mesoscopic solvent approach (cf. Figure 2a) [2]. Such a multi-scale molecular modeling approach is very efficient, and it is particularly well suited for computing ΔΔG's. For instance, computing the ΔΔG's corresponding to the two ligands shown in Figure 1 can be achieved with a high statistical accuracy by using 160 2.93 GHz Intel® Xeon® 5500 processors processors within less than 24h. With molecular modeling methods treating all the atoms at the same level of accuracy, about 20 times more computational resources are needed. Moreover, our approach is also accurate, as shown in Figure 2b: a linear relation is obtained between the results predicted by our multi-scale approach and the experimental ΔΔG's concerning 12 different ligands related to those shown in Figure 1.

These encouraging first results illustrate the reliability of our approach. In particular, its accuracy in computing ΔΔG's is about 0.5 kcal mol-1 here, a value considered to be the uncertainty of high level ab initio quantum methods. Moreover, this series of ΔΔG's have been computed in 32h by using 2240 cores on the petaflopic TERA 100, the super computing system of CEA/DAM (TERA 100 is made of 140 000 computing 2.23 GHz Intel® Xeon® 5500 processors cores) [3]. Hence, on an exaflopic computational system, our approach may be used to evaluate 875 000 ΔΔG's per 32h. To go beyond this level of performance, the present version of the code POLARIS(MD) will have to be adapted to extreme parallelism as it develops on single node. Indeed, the trend of the next generation super-computers will have significantly more cores per node without increasing memory per node in the same proportion. In the context of a pure MPI approach, the number of communications will dramatically increase with the number of cores, despite the shortening of the messages. A hybrid MPI/Threads approach is probably the best approach to tackle the challenge of the Exascale. Intel Exascale Computing Research Lab [4] is a collaboration between CEA, GENCI, the University of Versailles and Intel. The ECR teams are testing and evaluating different MPI/Threads hybrid parallel schemes a priori well suited to be used on different types of architecture, in particular, heterogeneous architectures based on the Intel® MIC, with particular attention to defining the optimal data structure for the code POLARIS(MD). This will result in an improved and optimized version of POLARIS(MD), a promising step towards the computation of millions of ΔΔG's per day on such computational systems, which appears to be a reasonable and feasible objective.

Our methodological and algorithmic developments, together with all the progress made recently in the molecular modeling/molecular dynamics field, paved the road towards high throughput docking techniques, whose aim is to generate "in silico" new families of drugs by knowing only their possible in vivo biological target (hence, this corresponds to a true ab initio approach). This will have a profound influence in the near future on the way pharmaceutical research is handled, in particular, by exploring new therapeutic approaches, by reducing the cost of the development of the future drugs, by limiting testing on animals, and, at least, by favoring the emergence of personalized therapeutics.

References

[1] M. Nazare, et al: Probing the subpockets of factor Xa reveals two binding modes for inhibitors based on a 2-carboxyindole scaffold. J. Med. Chem., 48 (2005) 4511

[2] M. Masella, D. Borgis, and Ph. Cuniasse: Combining a Polarizable force-field and a coarse grained solvent model. J. Comput. Chem., 29 (2008) 1707; ibid , 32 (2011) 2664

[3] www-hpc.cea.fr/fr/complexe/tera10.htm

[4] www.exascale-computing.eu

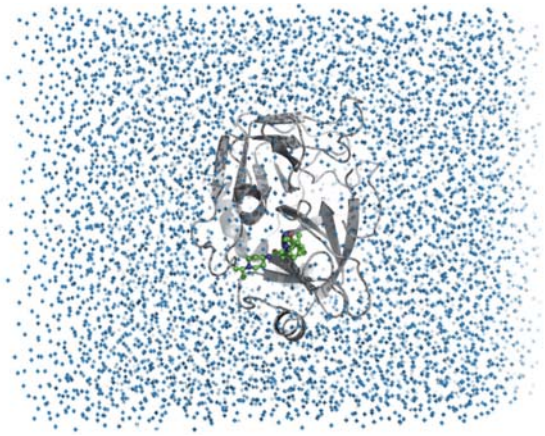


Figure 2a: A protein fXa/ligand complex embedded in a water box. In our mesocopic solvent approach, the three atoms of a water molecule are modeled by a single particle. The interactions among these particles are modeled using simple mathematical functions, explaining the model efficiency. With such a model, simulating a system in water or in gas phase requires almost the same computational time.

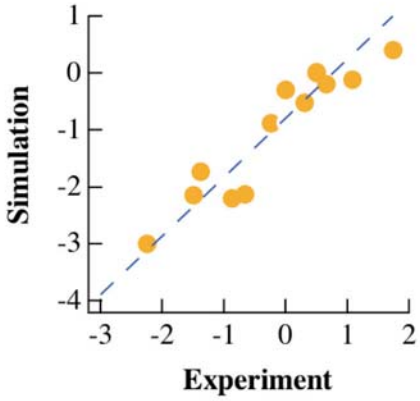


Figure 2b: The theoretical ΔΔG's corresponding to 12 ligands interacting with the protein fXa are plotted vs the experimental values. The result of the linear regression is shown in dashed line (the regression factor is here 0.94).

Exploring Parallel Programming Models and Runtime Environments: The MPC Framework

Patrick Carribault[1,2]
Marc Pérache[1,2]
Sylvain Didelot[1,3]
Bettina Krammer[1,3]
Marc Tchiboukdjian[1,3]
[1] Intel Exascale Computing Research Lab
[2] Commissariat à l'Énergie Atomique et aux Énergies Alternatives
[3] University of Versailles at Saint-Quentin

The HPC community is entering the 10 petaflop/s era, exposing the ability to execute up to 10^{16} floating-point operations per second. The next major milestone is the exaflop/s architecture with 10^{18} floating-point operations per second, predicted by the end of this decade. Yet there is a price to pay to reach such performance; while these architectures are composed of more and more cores (potentially including wider execution units), the amount of memory per core is dramatically decreasing.

Such evolutions are already impacting current large-scale applications. Using a Pure-MPI model (or MPI-Everywhere) will eventually be impossible because of the intrinsic data duplication of this parallel programming model. Therefore, applications will have to mix multiple parallel programming models to fully exploit these new-generation architectures. Hybrid MPI + OpenMP or MPI + PGAS programming models might be promising candidates, but current libraries suffer from the diversity of programming-model implementations. For example, MPI libraries and OpenMP compilers may not be comprehensive to each other, leading to large overhead or, at best, providing the possibility to tune some knobs to drive the runtime behavior.

To tackle this issue, the French Alternative Energies and Atomic Energy Commission (CEA) started in 2003 to develop their own MPC (Multiprocessor Computing) framework (freely available on sourceforge). MPC is a unified parallel framework for HPC clusters of NUMA machines. Its two main goals are to:

- unify various parallel programming models for an efficient exploitation of petaflop/s architectures, and
- interoperate with other HPC components.

Since 2010, MPC has also been partly developed within the Exascale Computing Research Center (ECR), a collaboration between CEA, GENCI, the University of Versailles and Intel.

Unification of Mainstream Parallel Programming Models

Basically, MPC is a thread library supporting multiple parallel programming models in a unified way to ease their interaction. MPC currently comprises the following programming models:

- POSIX threads: MPC implements its own non-preemptive user-level threads, fully compatible with the POSIX-thread standard. Therefore, if an application using pthreads is compiled with MPC, each new thread will be running on top of MPC instead of being managed by the underlying OS kernel. Thus, the unified user-level scheduler of MPC will take care of scheduling these threads, without paying the large overhead of going through the OS layers.
- MPI version 1.3: MPC implements an MPI-1.3 compliant runtime based on threads, meaning that every MPI task is a user-level thread instead of being a more heavy-weight OS process. This is done thanks to process virtualization. This representation leads to a more flexible management on a computational node, including some facilities for parallel memory allocation and reallocation. Intra-node communications are easily done using the shared-memory available between the threads of the unique process. Only one copy is necessary to execute point-to-point commands. On the other hand, inter-node communications are done through an implementation of low-level high-speed interconnect API, including TCP and Infiniband. Optimizing this communication layer both for intra-node and inter-node communications is part of the work done in ECR. While we aim at providing similar performance as state-of-the-art MPI implementations such as MPICH2 or Open MPI, an important aspect is also to reduce the memory consumption of the communication modules.

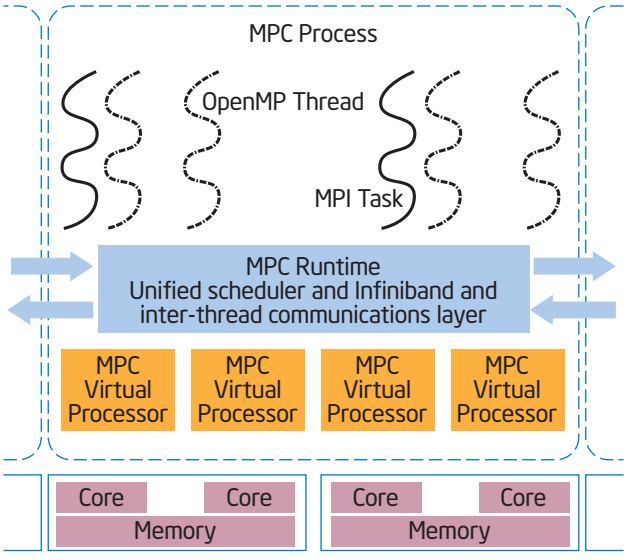
EulerMHD, 32 764 cores		
	MPC	Open MPI
Initialization time	45 s	27 min
Execution time	199,14 s	219,58 s

The table shows a comparison between MPC and Open MPI for EulerMHD, a CEA application for solving Euler and magneto-hydrodynamic (MHD) equations. The simulations with 1.6 billion particles were run on 32 768 cores of CEA's Tera 100 machine. MPC ran in multi-threaded mode, launching 1 process per node and 32 threads per process.

- OpenMP version 2.5: MPC already provides an implementation of OpenMP 2.5 while support for OpenMP 3.0 will be added

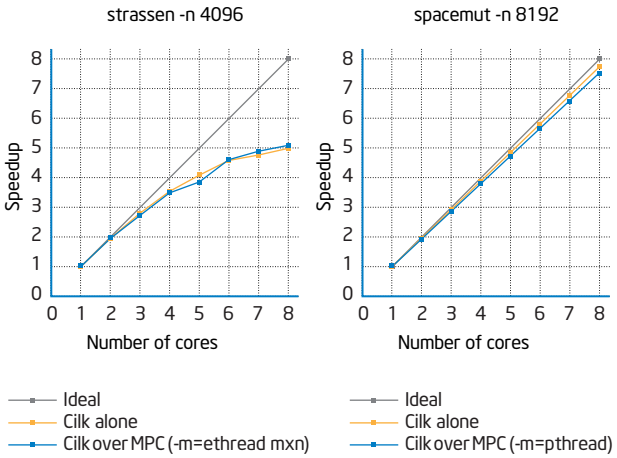
within ECR. The MPC compiler suite is based on a patched version of the GNU compiler suite for C, C++ and Fortran. It circumvents the GNU OpenMP runtime and calls MPC's own OpenMP implementation instead, which aims at being well integrated with other programming models like MPI. Therefore, OpenMP threads are modeled with stack-less and context-less threads as long as possible (typically while there is no scheduling point like an explicit barrier or a lock acquirement). This implementation leads to the design of a new scheduler-level called microVP on the top of each VP (virtual processor). With this infrastructure, the OpenMP runtime is able to handle an oversubscribing mode in an efficient way (i.e., when there are more OpenMP threads than cores allocated for the OpenMP parallel region).

- Hybrid MPI/OpenMP: With these last two programming models, MPC has been optimized to handle MPI+OpenMP applications efficiently. First of all, the main user-level scheduler sees MPI tasks and OpenMP threads the same way, to allow efficient scheduling among these different models. Then, both schedulers (main and dedicated to OpenMP) include a special polling method to avoid busy-waiting. This leads to a real fairness with respect to collective functions for both programming models. Thus, if one OpenMP thread is waiting on a barrier on the same core as an MPI task is waiting on an MPI barrier too, the scheduler will arbitrate which thread has to be scheduled.



The MPC team at CEA and UVSQ/ECR also experimented with stacking other programming models and runtimes on top of MPC, for example:

- Intel® TBB version 2.1: MPC supports the Intel® Threading Building Blocks (TBB) programming model. By re-compiling the open-source version of TBB, MPC is able to run TBB applications by handling the thread scheduling.
- Cilk: Compiling and running Cilk over MPC requires first some patches in the Cilk compilation chain but then performs almost identically as the native Cilk runtime. Below are some results obtained on a dual-socket Quad-Core Intel® Xeon® 5462 .



- UPC: While compiling and running UPC over MPC is possible, the results are less satisfying than with Cilk, and issues with scheduling, for example, remain to be resolved for efficient execution.

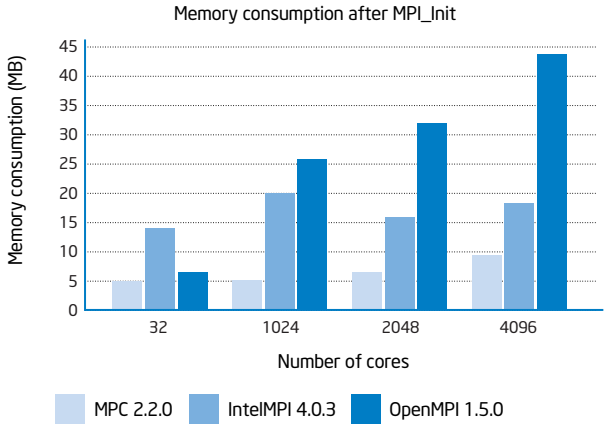
Integration with Other HPC Components

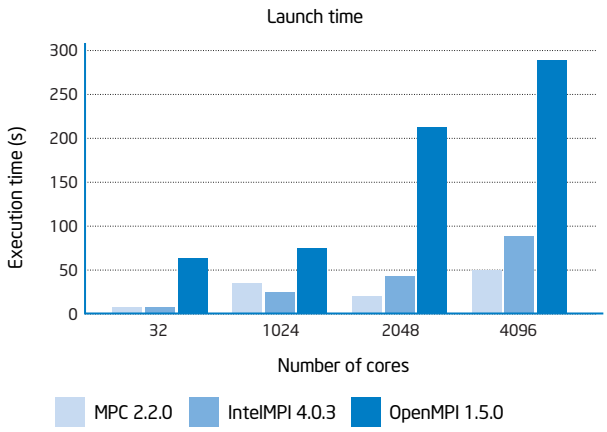
In addition to unifying parallel programming models, MPC aims at being fully integrated into the other components of the software stack and to the underlying architecture in order to reach the best possible performance. That is why the MPC distribution embeds several components, including:

- Compiler: The compiler included inside the MPC distribution is mainly used to transform C, C++ and FORTRAN programs using OpenMP. Based on GCC, MPC includes around 2500 lines of patches. Since MPC's MPI implementation is based on threads, one way to ensure the thread-safety of MPI tasks is by removing global variables. Therefore, the compiler has been patched to issue warnings about global variables at compile time. Finally, new compiler directives are currently being implemented within ECR with the goal of reducing memory consumption while ensuring best possible performance for the application.

- Parallel memory allocator: MPC is shipped with its own parallel memory allocator optimized for NUMA architecture and multi-threaded applications. With a realistic MPI hydrodynamics code from CEA, called HERA, MPC is able to gain up to 47% of memory (saving more than 1.3GB out of 2.8GB) thanks to the ability to recycle memory pages among MPI tasks on the same node.

The figure below compares the memory consumption and launch time of different MPI implementations (lower is better), running a simple benchmark (MPI_Init, MPI_Barrier, MPI_Finalize) on CEA's Tera 100 machine.





- Topology module: MPC includes a topology module to learn about the organization of the underlying architecture. This helps the runtime to allocate memory and map threads to cores more efficiently according to the cache hierarchy and the NUMA nodes.

- Debugging and performance tools: Debugger support for MPC includes e.g., GDB and DDT. The MPC team also collaborates with performance analysis tools developers to integrate support for MPC in their tools.

References:

[1] http://mpc.sourceforge.net

[2] M. Tchiboukdjian, P. Carribault, and M. Perache: Hierarchical Local Storage: Exploiting Flexible User-Data Sharing Between MPI Tasks. Accepted for IPDPS 2012

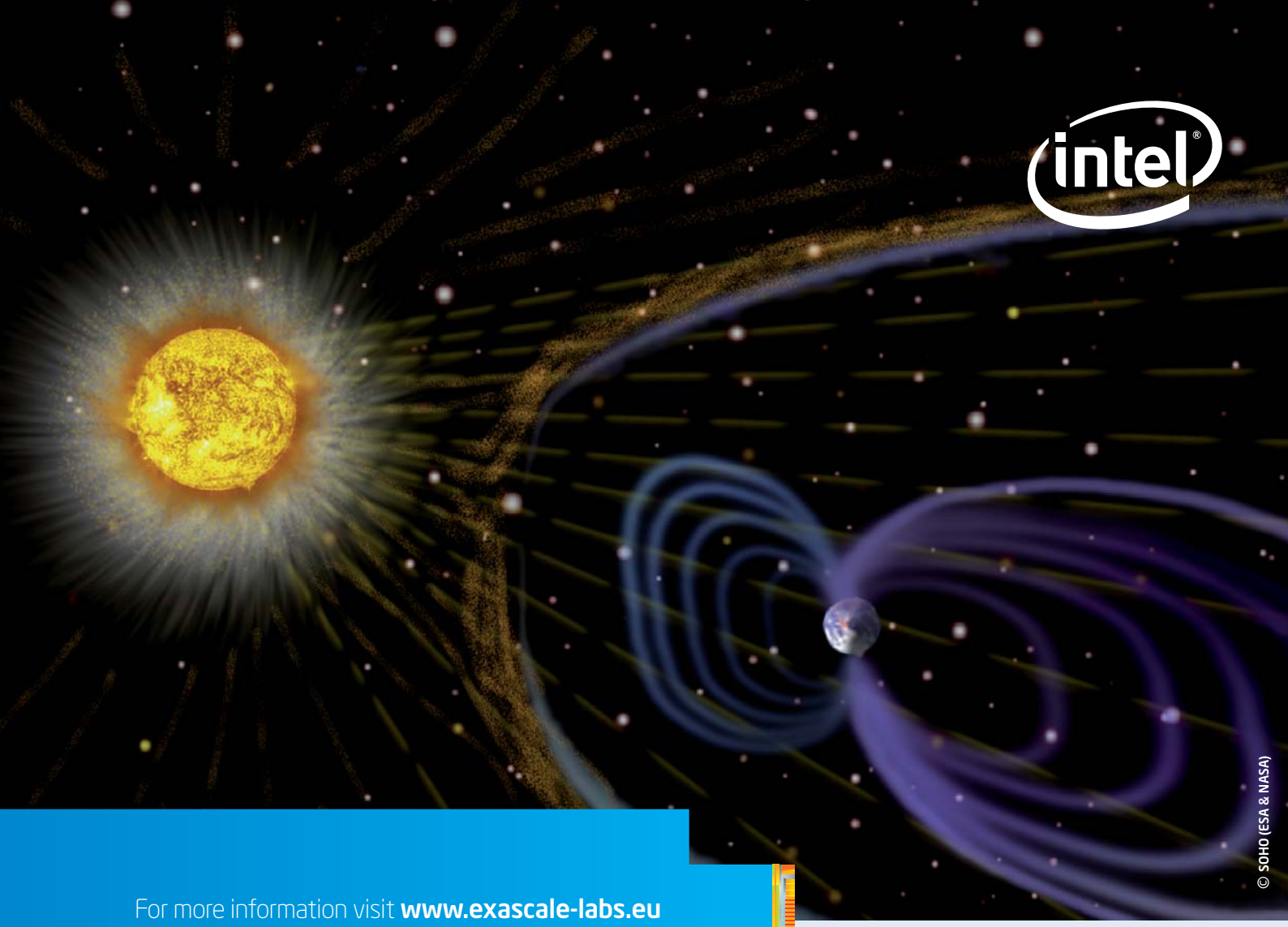
[3] P. Carribault, M. Perache, and H. Jourdren: Thread-Local Storage Extension to Support Thread-Based MPI/OpenMP Applications. IWOMP 2011: 80-93

[4] K. Pouget, M. Perache, P. Carribault, and H. Jourdren: User level DB: a debugging API for user-level thread libraries. IPDPS Workshops 2010: 1-7

[5] P. Carribault, M. Perache, and H. Jourdren: Enabling Low-Overhead Hybrid MPI/OpenMP Parallelism with MPC. IWOMP 2010: 1-14

[6] M. Perache, P. Carribault, and H. Jourdren: MPCMPI: An MPI Implementation Reducing the Overall Memory Consumption. PVM/MPI 2009: 94-103

[7] M. Perache, H. Jourdren, and R. Namyst: MPC: A Unified Parallel Runtime for Clusters of NUMA Machines. Euro-Par 2008: 78-88



For more information visit **www.exascale-labs.eu**

Intel ExaScience Lab – Leuven
IMEC
Kapeldreef 75
3001 Leuven
Belgium

Intel ExaCluster Lab – Jülich
Forschungszentrum Jülich
Jülich Supercomputing Centre
Wilhelm-Johnen-Strasse
52425 Jülich
Germany

Intel Exascale Computing Research Lab – Paris
Université de Versailles Saint-Quentin-en-Yvelines
45, Avenue des Etats-Unis
78000 Versailles
France

Intel-BSC Exascale Lab – Barcelona
Barcelona Supercomputing Center
Nexus II Building
c/ Jordi Girona, 29
08034 Barcelona
Spain

The Intel European Exascale Labs are part of Intel Labs Europe (ILE), a network of Research & Development, Product, and Innovation Labs spanning the European region as well as a variety of Intel business units. ILE was formally established in early 2009 as a central means of coordinating activities across Intel's diverse network of labs and to further strengthen Intel's commitment to and alignment with European R&D. In addition to driving key technology innovations for Intel, ILE works closely with academic, industry, and government institutions to advance innovations and strengthen Europe's technology leadership in the global community. For information visit **www.intel.com/europe/labs**.