FEBRUARY 2014

# HighPerformanceComputing

The reference media in HPC and Big Data

Volume I | Number 2

Americas

In this issue 2 AMD FirePro™ S10000 (OpenCL 1.2) TO WIN Again!

## 0x02

/verbatim

## JACK WELLS, ORNL

A insider vision of the future of HPC

/hpc_labs

## OPENACC 2.0 - PART II

The new compute management features

/discover

## TURBULENCE MODELING

New solutions for (almost) every industry

/hpc_labs

## PERFORMANCE PROGRAMMING

A scientific, practical methodology for measurable results

# ON THE ROAD TO EXASCALE: A 360° PROGRESS REPORT

The questions... and part of the answers

WWW.HPCMAGAZINE.COM

Subscribe
Access our archives
Discover exclusive contents

An HPC Media publication


Volume I, number 2



---

**Submissions**
We welcome submissions. Articles must be original and are subject to editing for style and clarity.

---

**Contacts**
editorial@hpcmagazine.com
subscriptions@hpcmagazine.com
advertising@hpcmagazine.com

---

**HighPerformanceComputing**
**HPC MEDIA**
**11, rue du Mont Valérien**
**92210 St-Cloud - France**

## From the Publisher

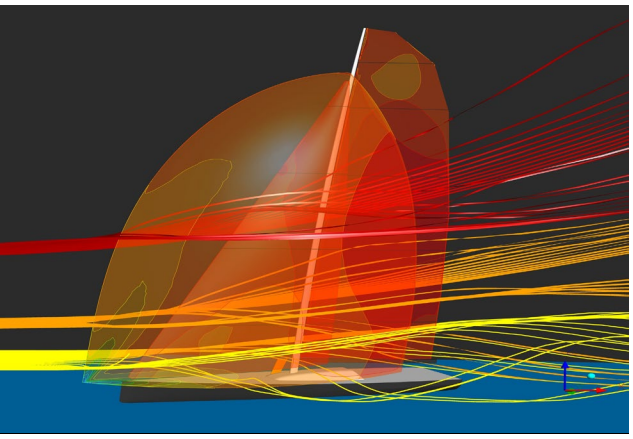
# Dear new readers,

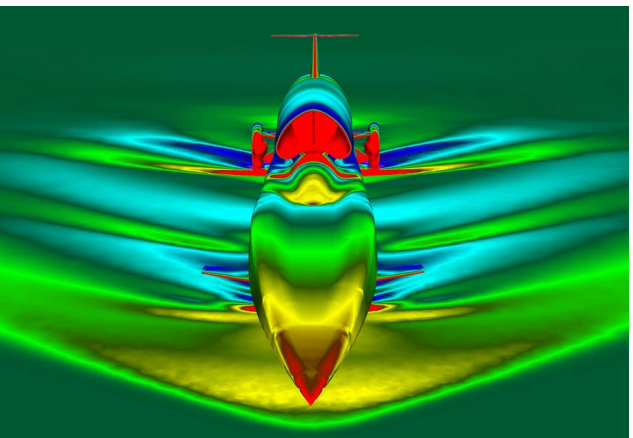
welcome to the second issue of **HighPerformanceComputing**. If supportive feedback is any indication of the success of our debut issue, we might be on the right track. From what we read, many of you seem to have appreciated the variety of its contents, its technical level and its general tone. We were also particularly happy to receive special mentions for our development papers - always "source code included!" Which is why we challenged ourselves to prepare, together with world-class experts in their fields, a series of ambitious articles, like this month's "Introduction to Performance Programming" by Romain Dolbeau. Hope you like it!

Now, there's more to high-performance computing than code writing. In keeping with a forward-looking spirit, this #2 issue focuses on long term innovation. In addition to our Exascale Progress Report, written from a technological and architectural perspective, we have been fortunate enough to interview Jack Wells, Oak Ridge's Director of Science, whose vision about the future of HPC is always enlightening. Last, because our computations are generally simulation-oriented, we wanted to put a spotlight on the newest turbulence models - a domain in which great progress has been made recently. On these articles, as well as on the rest of the magazine, your comments and critiques will be much appreciated. Happy reading !

frederic@hpcmagazine.com

---





# CONTENTS

## THIS MONTH



Visit www.hpcmagazine.com for multimedia content related to the features in this issue.

CRAY®

# Cray® CS300™ Cluster Supercomputers for Big Computing and Big Data Challenges

# /NEWS

## Energy materials: quantum effects finally demonstrated

Researchers at the University of South Carolina and Oak Ridge's Nanophase Materials Sciences center have just developed a new method for improving the simulation of the dynamics of energy materials, the number one property of which is to violently release energy through chemical transformation.

These materials are undergoing fundamental research in every respect with regard to their structure and behavior. And to this day, they remain a scientific challenge in every dimension of the physical space. This is especially true at the nano scale, where the introduction of one substance into another must theoretically lead to an increase in energy.

The scientific contribution of this new method is that quantum effects are now taken into account in the dynamic analysis of atomic systems. Due to the computational difficulty of simulating them, these effects have thus far been neglected in the hope that their impact would be insignificant. Come to find out, that's just not true, as the researchers demonstrated incidentally. "*The heavier the atomic nucleus, the more valid this hypothesis,*" points out Jacek Jakowski from University of Tennessee's NICS. "*But in the case of hydrogen and deuterium, which have very light nuclei, we were able to demonstrate that neglecting the quantum nature of atomic nuclei led to qualitatively different results. Hydrogen and deuterium are so light that the quantum effects are especially strong.*"

Researchers have used the Kraken petaflops machine at ORNL to simulate the absorption of hydrogen by carbon. They noticed that, on graphene, nuclear properties are responsible for greater deuterium absorption selectivity compared to hydrogen. More in ACS' Journal of Chemical Theory and Computation.

### Now is a good time to test-drive the K40

**As part of its now usual advanced products launch strategy, NVIDIA is offering free test drives of its latest accelerator, the Tesla K40, via remote access. Scientists and engineers have the possibility of evaluating their own codes or testing already installed applications, among which AMBER, NAMD, GROMACS, and ACEMD. To do so, an on-line application has to be filled out, after which all reasonable requests should be eligible. A quick reminder: the K40 theoretically performs 40% better than the K20X thanks to a larger internal memory of 12 GB and the GPU Boost technology, which basically "overclocks" graphics processors on demand.**


# $1,000,000,000 for Watson

Everyone knows Watson, the "intelligent" computer designed by IBM to understand unstructured data, which showed off its prowess by winning Jeopardy, the famous American game show. But IBM has not yet been able to convert this critical acclaim into hard currency.

This explains why they created the Watson Group, a new division whose mission is to accelerate the development and marketing of new solutions dedicated to cognitive computing. The program includes $1 billion in investment, $100 million of which is earmarked as venture capital to support startups affiliated with the Watson Developers Cloud launched last November.

For its designers, Watson is a revolutionary platform, which ushers in a new generation of machines capable of filtering, structuring and "understanding" massive quantities of data written in human language. And according to IBM CEO Virginia Rometty, Watson "*has the potential to transform industry and enterprises to new levels of knowledge for citizens and the masses.*"

From this point of view, which we share, a broad range of analytical applications with a very high added value should logically ensue. We already mentioned the case of medical diagnostics with a higher accuracy rate than that of specialized practitioners, but Watson can also be used to better define customer expectations and meet them by providing the right discount or the right offering, to provide an immediate response to the numerous questions asked of public service agencies, to help travelers find their itinerary according to a multitude of criteria, etc.

Not only do these examples give meaning to the term "smart machine", they also clear the path for a seemingly infinite sequence of smart-machine-driven disruptions, first in the corporate world, then among consumers. Gartner is spot on, listing Watson in its Top 10 Strategic Technology Trends for 2014. The knowledgeable analysts predict that by 2017, 10% of all computers will have cognitive capabilities derived from IBM's work. That for sure would mean progress for real!

# Intel leaves its mark on the Cloud...

Like fog, the Cloud has so far been conceived of as a kind of black box concept with no visibility into the technologies – essentially servers – used to power it. This relative opacity is about to change. At least that's Intel's objective with its Cloud Technology Program, which will start to be deployed this year in partnership with major CSPs (Cloud Services Providers) such as Amazon Web Services.

This initiative is primarily aimed at promoting the "Powered by Intel Cloud Technology" seal, thus allowing customers to make an informed choice of the type of leased hardware. The benefit claimed by Intel is typical from a marketing standpoint, which is that of a hardware environment presented as optimal. "*Just like choosing a car, the engine behind the Cloud service has an impact on performance and efficiency*" says Jason Waxman, VP of the Data Center Group and General Manager of the Cloud Platform Group at Intel.

But since the manufacturer's technology portfolio is no longer limited to processors, CSPs participating in this program will also have to provide detailed information on the type of CPU (with or without an accelerator), storage, software, and networking capabilities. This will probably bring a certain number of performance- and security-related solutions out of the shadows, such as Intel Turbo Boost Technology, Intel Advanced Vector Extensions (AVX), Intel Data Protection Technology with Advanced Encryption Standard New Instructions (AES-NI), and Intel Virtualization Technology (VT), to name a few.


# ...but experiences production overcapacity

It's the story of a $5 billion investment. It takes place in Chandler, Arizona and it features one of the biggest, most sophisticated plants in the world. Completed last year and touted in a speech by President Obama as a symbol of the future and the appeal of the American manufacturing sector, Intel's Fab 42 will remain empty, without machines and without staff, for some time to come. How is this possible? Designed to manufacture the next 14-nm components, the plant has become the collateral damage of Intel's recent production overcapacity, as the company preferred to use the existing capability of its other plant, also located in Chandler, Arizona, for efficiency and profitability reasons. Intel's local business manager wants to be reassuring, maintaining that this facility will indeed be used, "*probably in the near future*." The alternative would be unfortunate indeed...

# Spatial programming: a truly disruptive model!

The principle of locality is well known in programming: there is temporal locality, which makes it possible to reuse previously used data and instructions, and there is spatial locality, which characterizes the likelihood of accessing data neighboring an item that has just been used. The new OpenSPL (Open Spatial Programming Language) programming model attempts to further exploit the latter principle. Born of a consortium mixing industry (Chevron, CME Group, Juniper Networks, Maxeler Technologies, etc.) with universities (Imperial College London, Stanford, Tokyo, and Tsinghua), this revolutionary approach achieves very significant performance and energy consumption gains compared to the conventional model of processor instruction execution.

OpenSPL

Originally, OpenSPL comes from the idea that limits in terms of scalability, efficiency, and complexity of the temporal computation model of multicore processors require radical innovations in the design of systems destined for scale-up. For its conceptors, the proof lies in this simple factual observation: while the number of transistors has risen for example from 400 million (Itanium 2) to 3 billion (Xeon Phi), memory latency has only improved by a factor of 3.

Based on these considerations, OpenSPL's general principle lies in decoupling the control and data flows. The model consists of a spatial hardware representation (Spatial Computing Substrate or SPS) and three types of memory (scalar, slow, and fast memory) - the SPS being capable of executing static computation kernels linked together by data flows in a single instance and in parallel. Sounds unclear? Stay tuned: we are preparing, in collaboration with OpenSPL's designers, a detailed overview of the language and its potentialities.

## HydrOcean receives an HPC Innovation Excellence Award

GENCI should be proud. Thanks to its support through the HPC for SMB project (in partnership with INRIA and BPI France), the startup HydrOcean has received the prestigious IDC HPC Innovation Excellence Award for the scientific quality of its works in simulation.

Created seven years ago to develop and sell solutions derived from research conducted at the fluid dynamics laboratory of Ecole Centrale de Nantes, HydrOcean specializes in fluid flow simulation for the marine and industrial sector. The SPH-flow software that is the flagship of this startup (now with more than 20 employees) is a solver based on a particle method for modeling complex fluid flows and fluid/structure interactions. SPH-flow is also in itself a model of scale-up performance: its MPI parallelism achieves an operating ratio of 85% to 100%. What's more, a GPU version is available which, according to HydrOcean, improves the solver's performance by a factor of 5.

"We are especially happy with the success of HydrOcean, which we have been assisting for more than 18 months," delights Catherine Rivière, GENCI CEO. "Not only does this demonstrate that supercomputing is a source of innovation, and hence, competitiveness for small and medium companies everywhere in the world, it also shows the need to help them achieve its full potential."

Today, HydrOcean is developing its own digital simulation tools, which it uses to offer specialized design assistance services in the four sectors of shipbuilding, offshore operations, boating, and marine energy. Good luck and Godspeed to them for the future!

# UK bets on quantum computing

Hear ye: the British Crown is investing in quantum computing! Such is one of the items in the budget revision of Her Majesty's government aimed at thwarting the recession undermining the economy of the United Kingdom of Great Britain and Northern Ireland. The budget allocation of approximately £270 million over five years will be shared by five research centers dedicated to these new technologies ("Quantum Technology Centers") and backed by the country's major universities. Note that a specialized institute bearing the name Peter Higgs will also be created in Edinburgh.

Following NASA's and Google's investment in the D-Wave 2 machine, it's now Britain's turn to secure a position in the qubits approach and attempt to make it a national specialty. Although quantum computing is advancing slowly, it is still one of the main disruptive technologies on the horizon. Knowing that a quantum machine can simultaneously represent a theoretically unlimited number of states, the application potential is immense, particularly in the field of cryptography. Hence the quip by many analysts that "*Quantum is our big brother*."

# Final stretch for PRACE's "Energy" PCP

As part of its third implementation project (PRACE-3IP), PRACE launched a Pre-Commercial Procurement (PCP) procedure on November 20, 2013, intended to define innovative technological solutions for energy efficiency, a major challenge for the next generation of computers. This PCP is a new tool of the European Commission for opening R&D phases to competition - the general idea being to co-finance innovative R&D work with selected companies. Logically, it will be implemented by a consortium consisting of GENCI for France, CINECA for Italy, CSC for Finland, EPCC for the United Kingdom, and JSC (Jülich) for Germany, among others.

With this first PCP, the HPC leadership in Europe is recognizing the paramount importance of the energy dimension in scale-up feasibility. For this reason, the initiative is aimed at promoting a "whole system design". PRACE stresses the fact that energy efficiency cannot be conceived of as a simple architectural feature, but instead must be approached as a multi-scale problem, so to speak, running the gamut from transistor material research to thermo-hydraulic engineering for overall cooling.

Registration for entities wishing to apply will end this month, on February 24 to be precise. PRACE is planning to choose up to five projects for phase I of the plan ("Solution design", six months duration) and has a budget of 5 x €180k to do so. Phases II ("Prototype development") and III ("Pre-commercial small scale product/service development") should last 10 and 16 months with allocated budgets of €2.7 and €5.4 million, respectively, to be shared by three and then two of the projects chosen along the way. The official announcement, including application details, is available in English at **CINECA's website**.

# In-memory computing: a strategic shift for SGI

By aggregating flash and DRAM memories between disks and processors, In-Memory Computing (IMC) platforms greatly reduce data access latency. As such, they are particularly well suited to computer centers facing the exponential growth of databases, be they structured or not. Everyone knows the primary determinant of this trend, at least from experience. But not until it is quantified precisely do we realize the full extent of this technological advancement. No less than 85% of HPC computing cycles are lost simply because the servers are waiting for data. The transition to IMC could therefore theoretically bring an energy efficiency improvement on the order of 80% – nothing to sneeze at – to which a theoretical improvement in storage density by a factor of 40 can be added as a positive side effect.


Well-aware of the strategic importance of this technology, SGI just announced a strengthening of its partnership with SAP, another in-memory pioneer with SAP-HANA (High Performance Analytic Appliance). The result is that the next generation of SGI UV systems based on SAP-HANA will be designed to take advantage of high-performance DRAM modules providing roughly 200 times the transfer rate of current flash memory for workloads of up to 64 TB. Combined with the company's shared memory technology, it should lead to an unprecedented level of performance per node, with an estimated 50% reduction in management costs compared to multi-node solutions. The first production units are expected to be demonstrated in June, with commercial availability slated for 3rd quarter 2014.

## One more patents record for IBM

With more than 8,000 inventors in some one hundred countries worldwide, IBM is again number one among patent holders in the United States for the 21st consecutive year, with no less than 6,809 titles, a historic record for Big Blue.

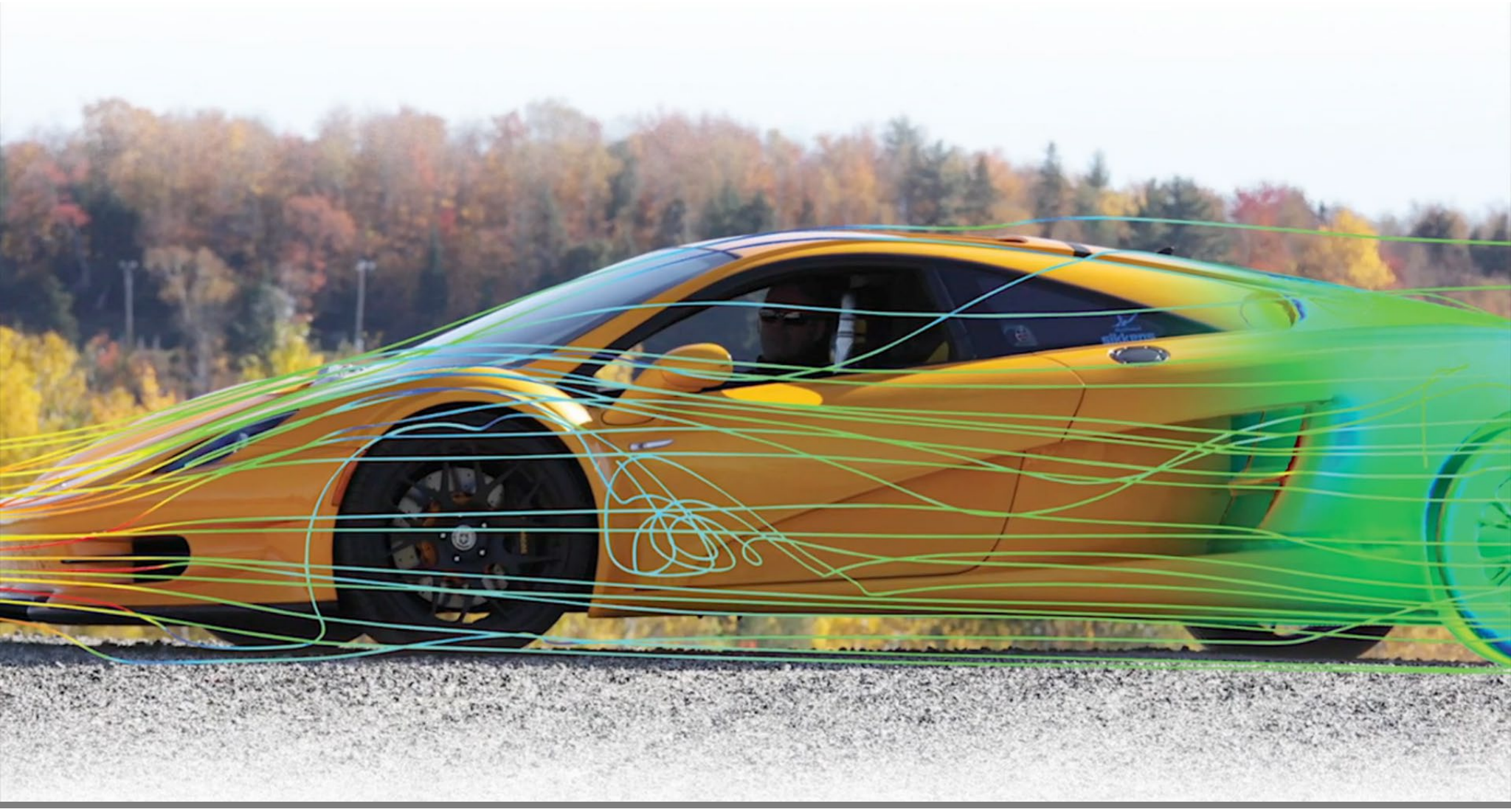
The figure representing patents held in the US alone, it is no surprise that, in this ranking provided by American consulting firm IFI CLAIMS Patent Services, Europe pales in comparison. Of all foreign countries, only Germany with Siemens and Bosch appears among the top 50.

By comparison, the number of IBM's patents exceeds the total number of patents granted to Amazon, Google, EMC, HP, Intel, Oracle/SUN, and Symantec combined. For the first time, Google and Apple, the superstars of Silicon Valley, rank in the top 20, at 11th and 13th place, respectively.

### Kalray's MPPA256 now on board!

**Until recently, Kalray's MPPA256 many-core processor was only available in the form of a card to be connected to a host system via a PCIe gen3 bus. It is now integrated into a compact 15x12 cm form factor with an x86-compatible host CPU.**

**This new card, called EMB01, is a real speed demon: with its 230 Gflops for a power draw of 5 to 10 W, it's ideal for on-board systems requiring substantial computing power. As with all MMPA implementations, it comes with a C/C++ programming environment and an SDK for application development, optimization, and uploading to the chip. Pre-orders are now being taken. First scheduled deliveries: March 2014.**



# Altair is on the move

The new year is off to a strong start for the American publisher, a major player in the solver market, as it furthers its growth objective by acquiring EM Software & Systems. In addition to providing Altair with additional distribution offices in the United States, Germany, and China, this strategic move integrates HyperWorks with EMSS's suite of FEKO electromagnetic solutions. The HyperWorks suite, with its wealth of star applications for modeling, rendering, analysis, and optimization (including RADIOSS, OptiStruct, HyperMesh, HyperView and HyperGraph, etc.), will now benefit from recognized functions in the field of electromagnetism, and more specifically in multiphysics simulations and design optimization. In particular, the FEKO solver can resolve electromagnetic/thermal or electromagnetic/mechanical coupling problems.

**Automotive wind tunnel simulation**

In addition to this acquisition, Altair has also announced HyperWorks Virtual Wind Tunnel, a new application dedicated to wind tunnel simulation for the automotive industry. Based on AcuSolveMD, the in-house CFD solver, Virtual Wind Tunnel can predict flow fields and flow separations, among other phenomena. As many case studies have demonstrated, external aerodynamic simulation plays an important role in modern vehicle design, as it affects fuel consumption, stability, engine cooling, interior noise - even windshield wiper performance.

**HyperWorks Unlimited:**
**a private cloud offering with SGI**

But wait, there's more. In addition to the classic HyperWorks licensing offering, Altair is joining forces with SGI to provide a private cloud solution for its software portfolio. HyperWorks Unlimited, as it is called, is based on an HPC cluster configured according to the type of code, and it allows for unlimited use of all Altair software, including the PBS Works computational load management tools. We're curious to see what will be the reaction of the pue players in on-demand high-performance computing...

/news

# Bull innovates in data center management

With the help of CA Technologies, Bull is now assisting businesses with the operation of their data centers. In an era of cloud computing and green IT, and with space and energy constraints ever more stringent, the progressive DCIM (Data Center Infrastructure Management) approach offered by Bull aims to be global, from consulting on the upstream end to real-time usage management. The goal is to improve energy efficiency, keep critical systems up and running, and enhance user productivity.

At a time when digital systems already consume 10% of the global electrical power production (which is expected to double over the next 10 years), data center management is becoming a challenge both for the economy and for society as a whole; hence the need for an approach that is as rigorous as it is pragmatic. Concretely, the proposed solution consists of CA ecoMeter, which issues detailed energy consumption information (and alerts) in real time direct from the monitored systems, and CA Visual Infrastructure, which displays comprehensive resources dashboards with the possibility of managing and optimizing these resources on the fly. For more information, we recommend reading the white paper titled "Operating data centers in light of the new energy paradigm."

# Who wants to win an AMD FirePro S10000 6GB accelerator?

## (Get lucky, there's 2 to be given away this month - again)

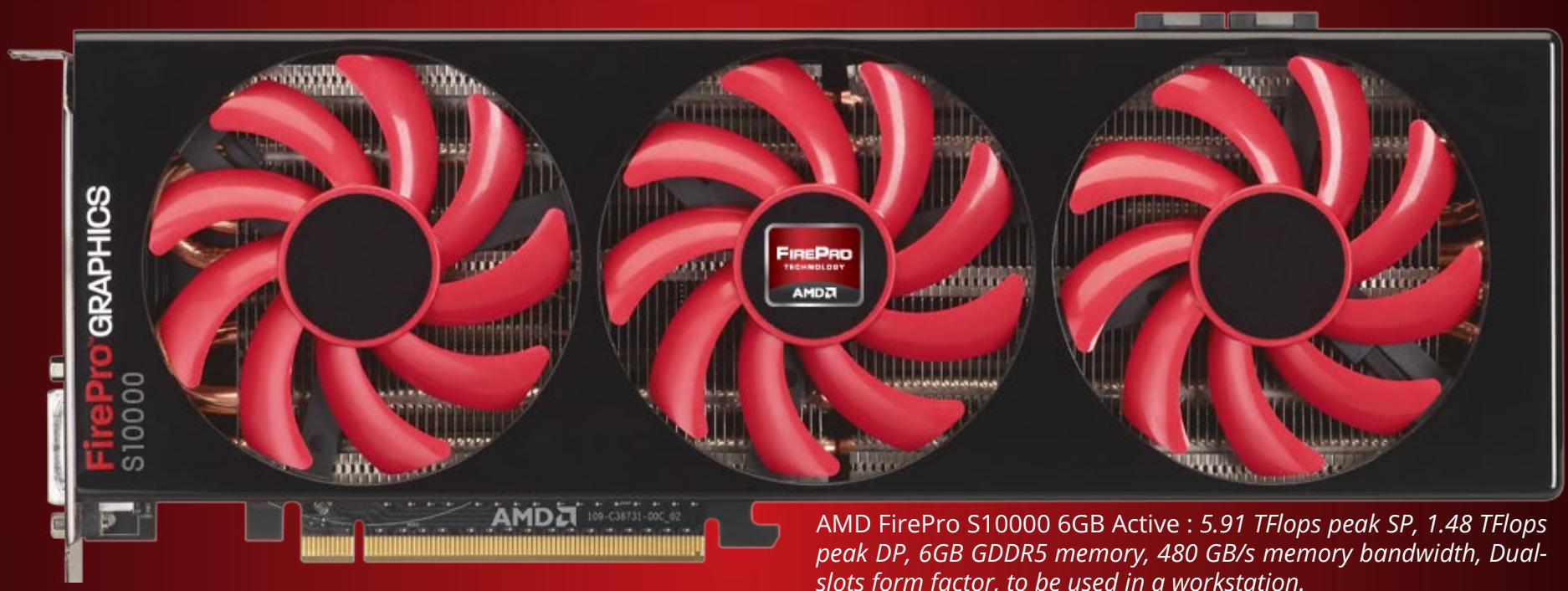
**Fits comfortably in OpenCL developers workstations.**

**Two high-end AMD GPUs directly on-board for powerful dual GPUs programming.**

**Free AMD Accelerated Parallel Processing SDK, OpenCL 1.2 compliant with BLAS and FFT libraries.**

**OpenCL 2.0 ready!**

**New! Free AMD CodeXL 1.3 with CPU-GPU debugger, profiler and static OpenCL code analyzer.**




AMD FirePro S10000 6GB Active : *5.91 TFlops peak SP, 1.48 TFlops peak DP, 6GB GDDR5 memory, 480 GB/s memory bandwidth, Dual-slots form factor, to be used in a workstation.*

**Future-minded developers want a parallel computing architecture that does not restrict their ability to use open tools and APIs to develop cross-platform.**

**Managed by The Khronos Group, like the widely used OpenGL framework, OpenCL is the solution to this legitimate demand. OpenCL is all about easy code portability, which makes it the only future-proof path to address all HPC coding requirements.**

## Last month's winners:

**Adnan Ozsoy, University of Indiana**
**Bradley Hakeman, Raven Industries**

## Win yours @ www.hpcmagazine.com

# /AGENDA

## FEBRUARY 2014

### Stanford Conference and Exascale Workshop 2014

**Where**: Stanford, CA, USA
**When**: February 3-5
The conference will focus on High-Performance Computing (HPC) usage models and benefits, the future of supercomputing, latest technology developments, best practices and advanced HPC topics. The conference is open to the public and will bring together system managers, researchers, developers, computational scientists and industry affiliates.

### PRACE Winter School 2014

**Where**: Tel Aviv, Israel
**When**: February 10-13
The PRACE 2014 Winter School programme offers a unique opportunity to bring users and developers together to learn more about the technologies that power HPC research infrastructures. The program is free of charge.

## MARCH 2014

### GPU Technology Conference

**Where**: San Jose, CA, USA
**When**: March 24-27
The GPU Technology Conference is the world's biggest and most important GPU developer conference. GTC offers unmatched opportunities to learn how to harness the latest GPU technology, along with face-to-face interaction with industry luminaries and NVIDIA experts. Stay tuned for announcements on who'll be bringing the 'wow factor'...

## APRIL 2014

### EASC2014 - 2nd Exascale Applications and Software Conference

**Where**: Stockholm, Sweden
**When**: April 2-4
This conference brings together developers and researchers involved in solving the software challenges of the exascale era. The conference focuses on applications for exascale and the associated tools, software programming models and libraries.

### 4th Cloud & Big Data Summit

**Where**: London, UK
**When**: April 10-11
The 4th Cloud And Big Data Summit 2014 will focus on the emerging area of cloud computing enhanced by latest developments related to infrastructure, operations, security, governance and services available through the global network. Panels, sessions and workshops are designed to cover a range of latest topics, trends and innovations related to cloud computing and data management. This Conference is a European platform to identify your company as a key player in the dynamic game changing market priorities in the field of cloud computing, data and information security.

## MAY 2014

### International Conference on Big Data Science and Computing

**Where**: San Jose, CA, USA
**When**: May 27-31, 2014
The Second IEEE/ASE International Conference on Big Data Science and Computing aims to bring together academic scientists, researchers, scholars and industry partners to exchange and share their experiences and research results in Advancing Big Data Science & Engineering.

## JUNE 2014

### ISC'14

**Where**: Leipzig, Germany
**When**: June 22-26, 2014
Join the global supercomputing community, ISC'14 again brings together 2,500 system managers, researchers from academia and industry as well as developers, computational scientists and industry affiliates from over 50 countries.

**To our readers:**

**HPC MAGAZINE's INTERACTIVE AGENDA debuts next month!**

**www.hpcmagazine.com**
**www.hpcagenda.com**

# Improving efficiency is the smarter way to operate.

On a smarter planet, businesses must accelerate growth and improve profitability. Through a unique combination of industry experience and expertise, IBM is helping manufacturing companies address critical issues across the value chain. With a strong global presence, advanced research and development capabilities and comprehensive hardware, software and services, we are equipped to help you improve operational efficiency as well as health, safety and environmental practices, manage costs and become more customer centric. IBM has the tools, technology and people to help you meet today's manufacturing challenges.

A smarter business needs smarter thinking.
Let's build a Smarter Planet.

View this executive summary of the results of a new survey conducted by Desktop Engineering to gauge audience familiarity with high performance cluster computing and its benefits. **http://bit.ly/14zq83c**

**ibm.com**/platformcomputing

/viewpoints

# INTELLIGENT NETWORKS TRANSFORM BIG DATA INTO INFORMATION

**Over the past few years, public and private organizations have experienced an awakening to the knowledge hidden in Big Data. While a concrete definition of the term has yet to emerge, the explosion of dedicated tools, among which intelligent networks, is strongly indicative that Big Data holds the key to limitless kingdoms of new understanding and discoveries.**

According to a widely quoted **statistic** from IBM, 90 percent of the world's data has been created in the last two years alone. The world is now generating data at a rate of 2.5 quintillion bytes each day, the storage equivalent of 57.5 billion 32 GB iPads. According to UK analyst firm The Big Data Insight Group, every minute of every day YouTube users upload 48 hours of new video, 571 new websites are created and Google receives over two million search queries. And Big Data is not just derived from search, social, and web – other sources include sensors that monitor climate information, purchase transaction records, and cell phone GPS signals.

Compounding this challenge is that much of what constitutes Big Data is actually unstructured data. While structured data fits neatly into traditional database schemas, unstructured data is much harder to wrangle. Much of the value obtained from Big Data analytics now comes from the ability to search and query unstructured data, for example, the ability to pick out an individual from a video clip with thousands of faces using facial recognition algorithms.

Technology companies are feverishly working to develop technologies such as Hadoop Map/Reduce, Dryad, Spark, HBase to turn all of this data quickly and efficiently into information capital for businesses and research organizations. At the heart of it, these technologies achieve speed and efficiency by parallelizing analytic computations across clusters of hundreds of thousands of servers connected via high-speed Ethernet networks. Hence, the process of mining intelligence from Big Data fundamentally involves three steps that could be summarized as follows: splitting the data into multiple server nodes; analyzing each data block in parallel; merging the results. These three operations are repeated through successive stages until the entire dataset has been analyzed.

***What we need is an intelligent network that, through each stage of the computation, adaptively scales to suit the bandwidth requirements of the data transfer in the Split and Merge phases.***

Owing to the Split-Merge nature of these parallel computations, Big Data analytics can place a significant burden on the underlying network, regardless of its architecture. Even with the fastest servers in the world, the data processing speeds can only be as fast as the network's capability to transfer data between servers in both the Split and Merge phases. What we need is an intelligent network that, through each stage of the computation, adaptively scales to suit the high bandwidth requirements of the data transfer in the Split and Merge phases, thereby not only improving speed-up but also improving utilization and experience.

***It is clear that a successful implementation will leverage two key elements: the existence of patterns in Big Data applications and the programmability that SDN offers.***

**The Role of Software-Defined Networking in the Big Data Equation**

The emergence of Software-Defined Networking (SDN) is a development with huge potential towards building this intelligent adaptive network for Big Data Analytics. Due to the separation of the control and data plane, SDN provides a well-defined programmatic interface for software intelligence to program networks that are highly customizable, scalable and agile, to meet the requirements of Big Data on-demand.

This software intelligence, which is fundamentally an understanding of what the application needs from the network, can be derived with much precision and efficiency for Big Data applications. The reason is twofold: first, the existence of well-defined computation and communication patterns e.g. Hadoop's Split-Merge or Map-Reduce paradigm; second, the existence of centralized management structures that make it possible to leverage application-level information e.g. Hadoop Scheduler or HBase Master. With the aid of the SDN Controller, that has a global view of the underlying network – its state, its utilization, etc. – the software intelligence can accurately translate the application needs by programming the network on-demand.

SDN also offers other features that assist with management, integration, and analysis of Big Data. New SDN oriented network protocols, including OpenFlow and OpenStack, promise to make network management easier, more intelligent and highly automated. OpenStack enables the set-up and configuration of network elements using a lot less manpower, and OpenFlow assists in network automation for greater flexibility to support new pressures such as data center automation, BYOD, security and application acceleration.

From a size standpoint, Software-Defined Networking also plays a critical role in developing network infrastructure for Big Data, facilitating streamlined management of thousands of switches, as well as the interoperability between vendors that lays the groundwork for accelerated network build out and application development. OpenFlow, a vendor-agnostic protocol that works with any vendor's OpenFlow-enabled devices, enables this interoperability, unshackling organizations from the proprietary solutions that could hinder them as they work to transform Big Data into information capital.

As the powerful implications and potential of Big Data become increasingly clear, ensuring that the network is prepared to scale to these emerging demands will be a critical step in guaranteeing long-term success. It is clear that a successful solution will leverage two key elements – the existence of patterns in Big Data applications and the programmability of the network that SDN offers. From that vantage point, SDN is indeed poised to play an important role in enabling the network to adapt further and faster, driving the pace of knowledge and innovation.

**Bithika Khargharia, Ph.D.**
**Senior Engineer, Vertical Solutions and Architecture, Extreme Networks**

CARRI Systems

Technology architect since 1992

intel Technology Provider Platinum 2013

NVIDIA PARTNERFORCE TESLA PREFERRED PROVIDER

# Conception Analysis Realization for Research and Industry


*Founded in 1992 by enthusiasts, CARRI Systems is a French company recognized for its technological expertise in high performance computing systems. Based in Noisy-le-Sec near Paris, CARRI Systems is a member of the European Pole of Competence in high performance simulation (Ter@tec) since 2011.»*

**Franck Darmon**

CEO, CARRI Systems

FABRIQUÉ EN FRANCE

intel NVIDIA AMD

**For more information about XLR solutions**

**www.carri.com**

/cover_story

# ON THE ROAD TO EXASCALE:

# A 360° PROGRESS REPORT

**Why the exascale? To this question, three answers are commonly accepted: first, to do things faster than today; second, to expand the scope of today's research and bring it to new horizons; third, to solve problems that can only be considered at this scale. Let's now see how - and when...**

While reaching the exascale is certainly what keeps most of the HPC and Big Data community busy - for the term, let's not forget it, means exapflops but also exabytes - two schools of ideas meet face to face on how we will get there. On one side, the defenders of the scalable approach, who believe all the necessary technologies are more or less available right now. According to them, most of the work that remains to be done to reach the $10^{18}$ units is going to consist in refining the hardware and software tools required to build clusters and make use of the data they will produce. On the other, the partisans of the disruptive approach, to whom exascale can only be conceived with new hardware, new software, new means to produce, exchange and store data.

As to the question of timing, forecasts vary, precisely because of this software dimension. A recent consensus seems to appear to consider that the hardware will be ready anytime

between 2018 and 2020 – inclusive of new energy efficiency and resilience management techniques, according to the most optimistic. The major difficulty, then, would concern software: programmability at this level of parallelism poses problems that we are far from being able to solve yet. When this dimension is taken into consideration, completion date becomes 2022…

This is the context, summarized. In this story, we have chosen to detail the many dimensions that are its main features. Starting with today's questions, we invite you to find out what will probably be the HPC architectures of tomorrow. A tour of the labs will allow us to review the most promising technologies currently under development. We'll then attempt to demonstrate why the Promised Land will be data-intensive by necessity. Some of the issues raised being rather polemical, your reactions are most welcome!

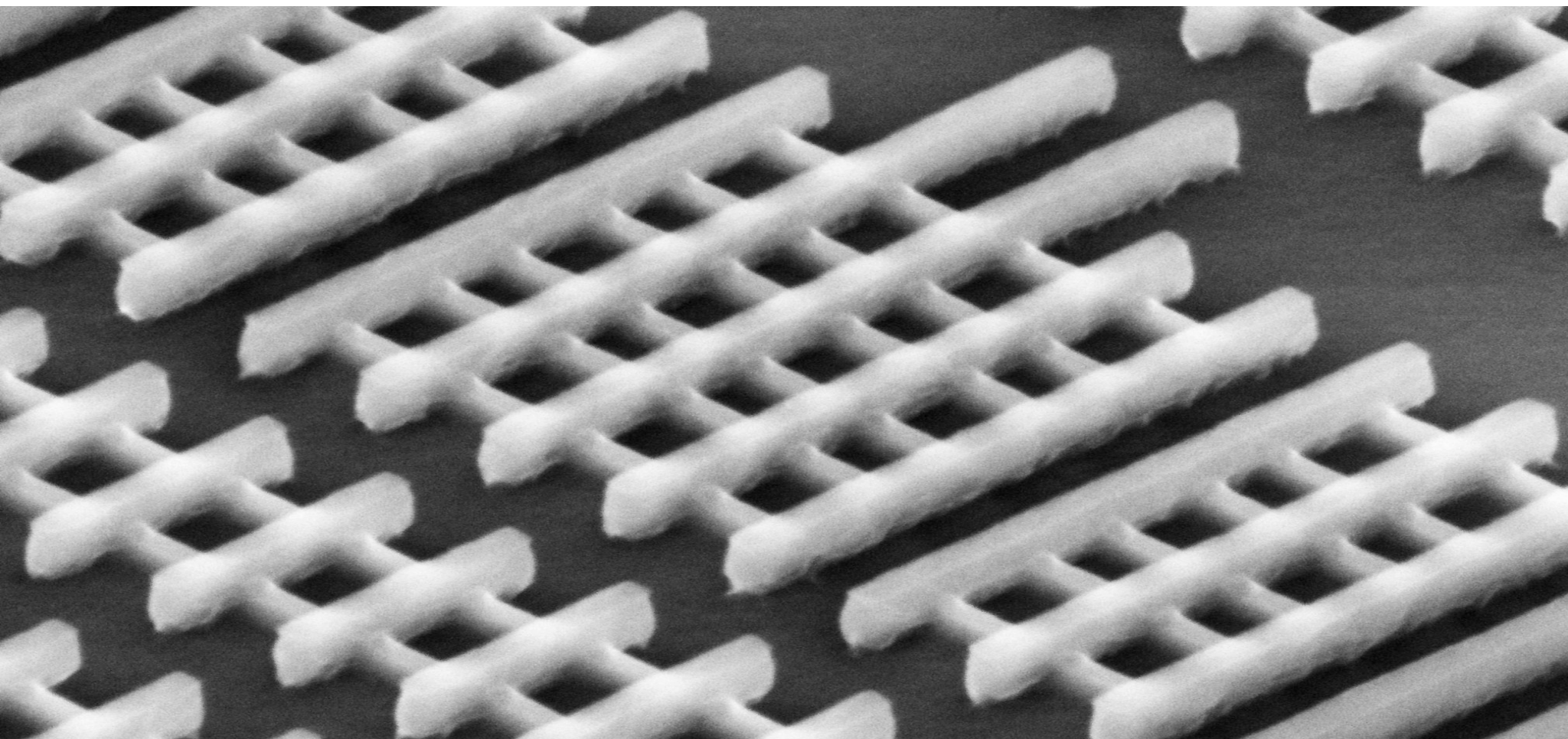
# 1 – An equation with X unknowns

**The community already knows that the exascale will not be reached by just multiplying the number and density of the components available today. Before thinking of the future, let's understand why.**

It is widely agreed that we are at the end of a cycle. The performance and energy-efficiency components of today's HPC platforms are already the result of a long optimization process, meaning that no decisive progress is to be expected in the near future. The same applies at the more global architecture level. It would indeed be possible to drive current designs a bit further, but in no way to bring them up to the next scale as such. Top this up with the increasing obsolescence of applications, which are sometimes several decades old, the deadlocks of programmability for a number of cores multiplied a hundredfold (at least), the explosion in volume of the data to be managed in pre- or post-processing… and you have an idea of the difficulty of the challenges ahead of us.

## Moore's law *vs* Dennard's scaling

At the hardware level, three major issues. The first is concurrency: Moore's law will probably continue to apply until the end of the decade, with VLSI geometries that could go down to 5 nanometers. But this is already challenged by another law, Dennard's scaling, which has it that the levels of voltage and current must remain proportional to the linear dimensions of a transistor. The potential result of this deadlock may

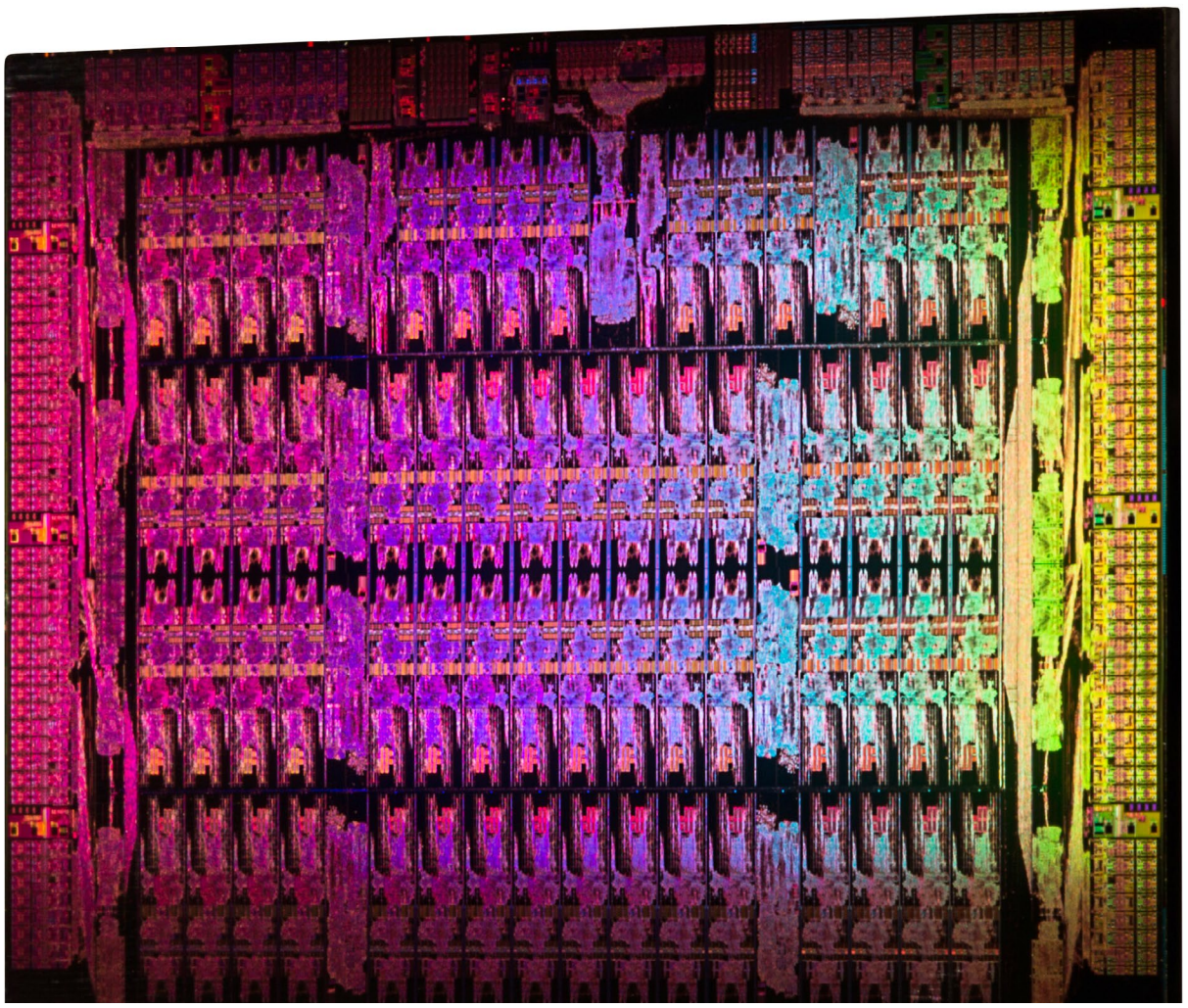


*Dennard's scaling means a reduction of the frequencies of processors as the size of their transistors (here in 3D version) decreases. For density and energy efficiency, it is a plus… to the detriment of performance.*

lie in flat and 3D integrated circuits operating at progressively reduced clock speeds, if only to contain their electric consumption. In this case, it will become necessary to multiply them. Nobody is surprised anymore by the mention of several hundreds of millions ALUs (Arithmetic Logic Unit) in an exascale platform; but if one considers that several threads for each ALU are necessary to hide the memory and network latencies, the number of simultaneous operations reaches several more orders of magnitude than what is considered programmable today.

The reliability of such systems adds to the difficulty of designing exascale architectures. Bearing in mind the MTTF (Mean Time to Failure) of the components available today, a typical exa platform managed according to current best practices would have a life span of between one minute and one hour. Improving industrial processes in this area will not be good enough. The multiplying nature of reliability implies global mechanisms of material resilience that are not yet mastered, mechanisms that also require new methods of control at software level.

If we do know that applications are what needs to be adapted, and that this will have to be achieved on the basis of more sophisticated mechanisms than just simple checkpoints, which imply processing intermediary computations again, then digital logic needs to be adapted. And finally, let's not forget that the behavior of transistors is bound to become more and more probabilistic. The noise margin is indeed going to be reduced considerably due to lower voltages and thinner internal channels, which is not without consequences. This will have to be put up with as well.

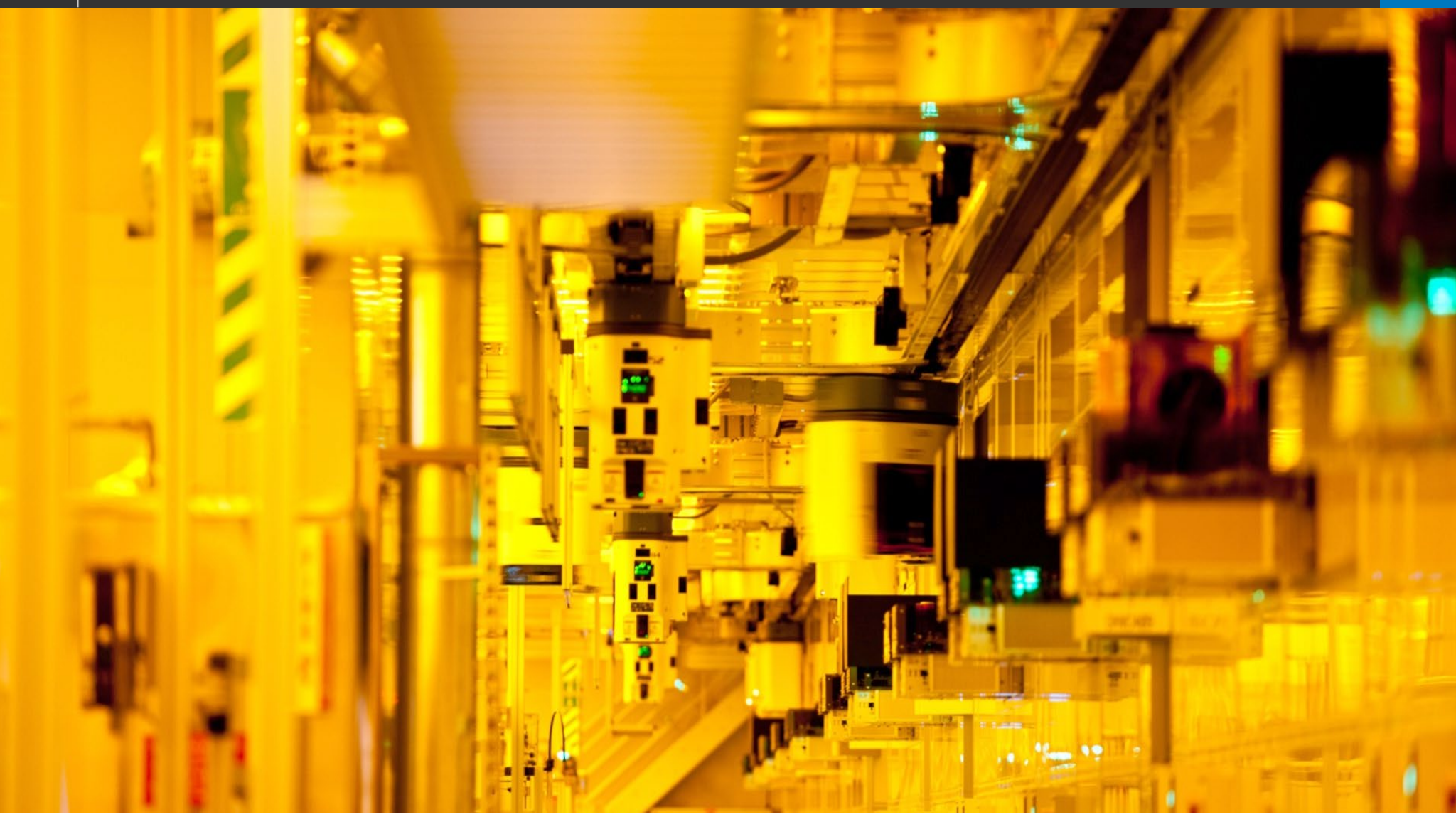


*Today's best processors (here the Xeon Phi die) do not exceed 2 Gflops / Watt. For 2020, the most optimistic forecasts predict a ratio between 6 and 8 Gflops / Watt, a figure clearly insufficient - by itself - for exascale energy-efficiency requirements.*

**50 Gflops / Watt**

The third major problem in scaling up is energy efficiency. Based on current standards, an exascale system could use as much as 100 MW, which is obviously not worth considering. What politician would be naïve enough to even suggest allocating a nuclear plant to a computer? Today's mainstream production efficiency reaches 2 Gflops/Watt, or 500pJ/flop. To make exascale tenable from the point of view of electric supply and running costs, a twenty-five-fold increase is required, which would bring "round-of-the-mill" HPC systems' efficiency to 50 Gflops/Watt (equivalent to 20 pJ/flop).

Clearly, such figures will not be reached by a simple evolution in semi-conductors design, whose increase in 2020 is only expected to be somewhere between a factor of 2 and 4. Other routes have therefore to be found – the plural is needed here – to save on energy. As we shall see in more details further below, among the ones most generally considered is the strong hybridization of clusters with very low consumption components, the reduction of data transfer wherever possible, and the automated switch off of processors dedicated to certain tasks (CPUs, GPUs, ASICs, FPGAs...) depending on the level of completion of the application.



*Without a MTTF 100 times higher than it is today, exascale machines, especially MPI systems, will have a production lifetime of less than one hour. By 2020, the improvement of industrial processes will not achieve that high a rate. Other solutions have therefore to be found...*

**The software cul-de-sac**

Last, but not least, of the problems that need be addressed is programmability. Logical obstacles to reaching $10^{18}$ flops are considerable. If one considers that 10 billion threads are needed to use 100 million cores to their maximum, new programming methods must be invented. For immediate practicability reasons, users will most certainly be offered downward compatibility solutions to run their MPI codes. But this will be the ultimate step, a quasi dead-end track, which already demonstrates that the MPI + C/Fortran duet is just not suitable at this kind of scale.

The problem concerning just about everyone, a lot of effort is currently being produced, generally focusing on the implementation of parallelism at the highest level. To free developers and scientists from the constraints of designing and coding it, one of the ideas most widely considered consists in allowing them to specify their algorithms in a descriptive way - i.e. to merely describe computing parallelism and data affinity - and to let robots finely map the whole description to the available computing resources.

**Case by case solutions**

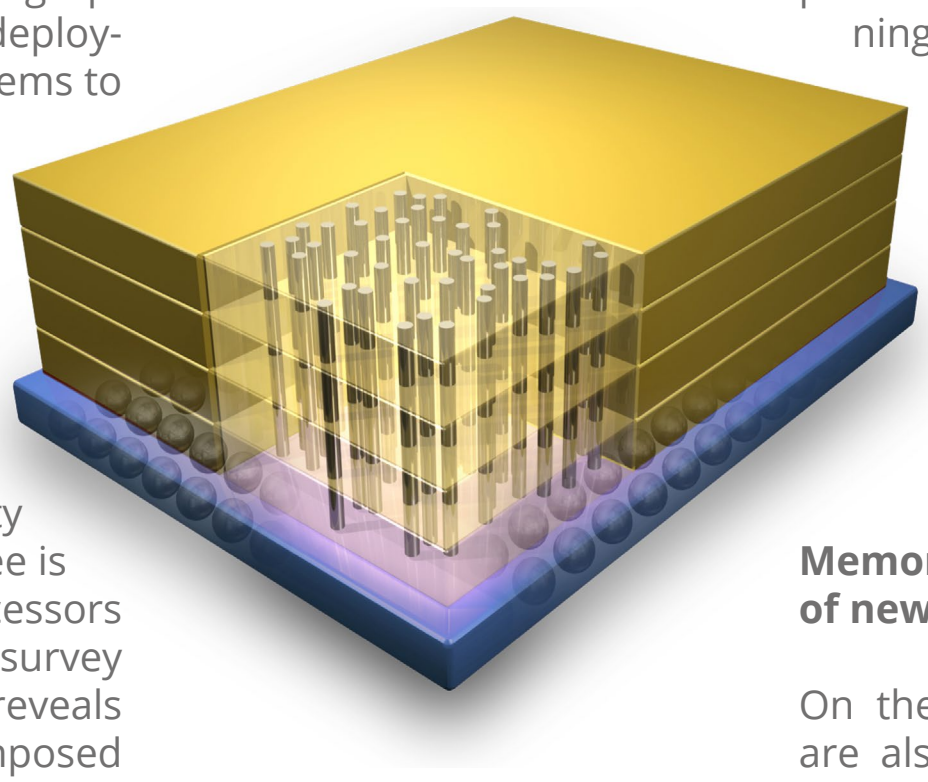
Another interesting approach is to parallel... parallelism. In this respect, the only way to go is to proceed application after application, which limits the frequency of publications, but results are potentially interesting. A Chinese team from the University of Defense in Changsha, for instance, is currently working on re-architecturing a portfolio of molecular simulation codes with the triple parallelization of tasks (adapting distribution to CPUs and accelerators), of threads (laying out multi-threading mechanisms dynamically) and of data (using dedicated SIMD instruction sets).

Whether generic or on a case by case basis, the fact that these researches have already started is not a coincidence: once successfully completed, the update of applications with an exascale potential will take a lot of time. Meanwhile, a number of new developments in hardware architectures and topologies, in components engineering and in otherwise disruptive technologies are also being carried out. This is what we invite you to discover in details now.

# 2 - Looking for new architectures...

**Hybrid processors, HPC / HTC convergence, memory-centrism... the exascale perspective excites the technical creativity of the community. Here, in our opinion, are the propositions on architecture with the greatest potential.**

It is now established that exascale platforms will bear only a very distant resemblance to the most powerful machines of today. Even those holding to the scalable approach recognize that architectural modifications of greater or lesser scope are essential to such a scaling up. From the optimized redeployment of existing subsystems to the integration of new, forthcoming technologies, there is no lack of innovative structural propositions...



## The hybrid approach

One on which a majority of experts seems to agree is the association of processors of various types. A quick survey of big HPC applications reveals that they are often composed of two types of segments: those that are limited in performance by their monothread nature and those that are limited in their throughput by the underlying equipment. Considering this heterogeneity, the logical solution consists in having a small number of components optimized for latency, like the cores of current CPUs, and a more important set of components optimized for throughput (cores of MIC / GPU accelerators) work together, as close as possible to each other. That is what the DEEP platform project proposes at the global system architecture level (more below). But nothing prevents us from contemplating that this mix may be implemented directly at the socket level. How? By hybrid "processors" which, thanks to this integration, would definitely better support multiplication in very large numbers than conventional CPUs. For these processors to happen, the industry will have to find them a use other than HPC, if only to finance their development. Perhaps in the area of mobile phones?

This concept of convergence or integration is also to be found in the idea of combining once and for all our existing HPC designs and HTC (High Throughput Computing) architectures like that of CERN for example. For the supporters of this approach, we would obtain the best of both worlds: a huge HPC capacity on a virtually unlimited volume of data. The concept is tempting, and that is probably why it seems to be making its way. Without revealing secrets, announcements in this direction should be made during this year's prominent HPC conferences. They will probably confirm the beginning of an actual reconciliation between HPC and HTC, with roadmaps encompassing great ambitions. For them to lead to new integrated architectures available commercially, it will be necessary, however, to wait a little longer.

## Memory at the center of new architectures

On the memory front, things are also moving fast, starting with the imminent industrialization of hypercubes of piled up components placed in the direct vicinity of cores, or even directly above them. This progress should meet our needs in nominal bandwidth, at least for some time, but it probably won't be enough to homogenize the ambitious platforms that the community envisions.

Based on this analysis, another architectural proposal is starting to be talked about. If exascale is considered synonymous with very large volumes of data, why not simply give up the conventional compute-centrism of


*According to the people in charge of the EHA (Extremely Heterogeneous Architecture) project for exascale, new uses of ASICs and FPGAs could play a major role in solving the performance vs energy efficiency equation.*

current platforms and switch to more memory-centric designs? Concretely, the idea is to place the memory subsystem at the functional core of our machines and to consider processors as peripherals - which would remove, in several possible ways, the problems of global bandwidth for data transfer. This Copernican revolution, provided it gains support among the community, could take some time. But after all, so is reaching 2020...

Another innovative concept is the one developed by the EHA project (Extremely Heterogeneous Structures). The target, this time, is the structural energy efficiency of computation cores. EHA takes up on its account the idea of hybridizing the nodes, but this time by associating general purpose cores with other hyperspecialized ones (GPUs, ASICs and FPGAs). The generic cores would be responsible for the control and the general processing of the application, while the specialized cores would be dedicated to accelerating designated and/or specialized computation segments. Thus, when such segments are inactive, or when there is no need for acceleration, the components in charge of their execution can be shut down. The idea in itself is not new; what is, is that the diversity in nature of the components involved allows such an architecture to be finely tuned to the needs of specific applications. And from a purely electronic standpoint, the approach makes sense: acceleration is not constrained, but no Watt is wasted. The concept is still under development but EHA should soon publish efficiency results quantified from reproducible simulations.

**The exascale from the existing**

If these approaches have not really gone past the early experimental stage yet, there is another, more conservative one whose first practical developments are already in progress. DEEP (Dynamical Exascale Entry Platform), such is its name, has this in particular that it largely rests on existing technologies, namely x86 CPUs and accelerators. To ensure their scaling up, DEEP implements a dual architecture known as “Cluster Booster” which brings together the CPUs in a conventional Cluster and the accelerators in a second one called the “Booster”. Because the accelerators are located outside the nodes, applications made up of kernels of different nature can be executed part on the Booster side (for the kernels which support scalability), and part on the Cluster side.

Developed by Jülich Supercomputing Center in Germany and strongly supported by Intel, DEEP is motorized by Xeon processors and opens on the outside via InfiniBand Mellanox interfaces. The Booster contains Xeon Phi accelerators interconnected through a terabit network in 3D toric topology called EXTOLL (and developed at the

University of Heidelberg). The level of latency of the exchanges, currently less than a microsecond, allows applications to get the most out of the MIC subset.

According to its initiators, this architecture is the most likely to serve as a basis for the first production exaflops systems, taking into account the current state of the art and the market. If the experiments are conclusive, it should also benefit from liquid cooling at room temperature, which will optimize its energy efficiency and facilitate the reuse of collected heat. Once validated, the technical realization of the whole system (cabinets, racks and blades) will be entrusted to Eurotech.

**Don't touch the applications**

On the software front, DEEP plays the card of reasonable innovation. The software stack should allow the execution of existing codes without modification - both at the Cluster and Booster levels. For that, the system will integrate a special version of ParaStationMPI, from ParaTech, which supports the network Booster and its toric topology. In order to get the most out of the specificities of this architecture, an additional layer above MPI is envisioned. It is this layer that will ensure the integration of high and low scalability code segments, and allow them to be executed on the Booster and the Cluster, respectively.

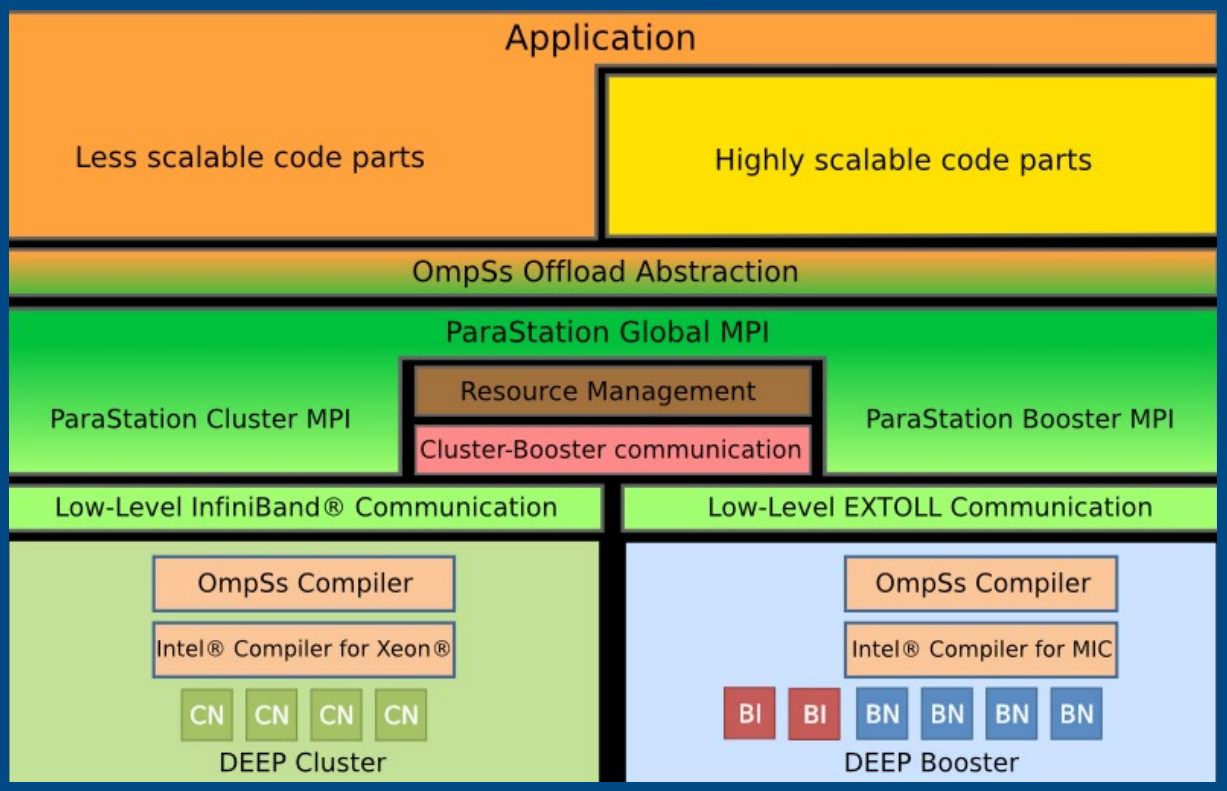
To facilitate the programmability of the system as a whole, a new programming model called OmpSs has been designed (by Intel's Exascale Laboratory at the Barcelona Supercomputing Center) with the purpose of allowing applications to be split into tasks. The idea, of course, is to use the silicon available on the Booster and the Cluster as much and as well as possible, but also to simplify the distribution of operations and exchanges between both sides of the architecture. An additional middleware, still under development, will be responsible for managing the underlying hardware resources and will allow, in particular, accelerated inputs-outputs at the Booster level.

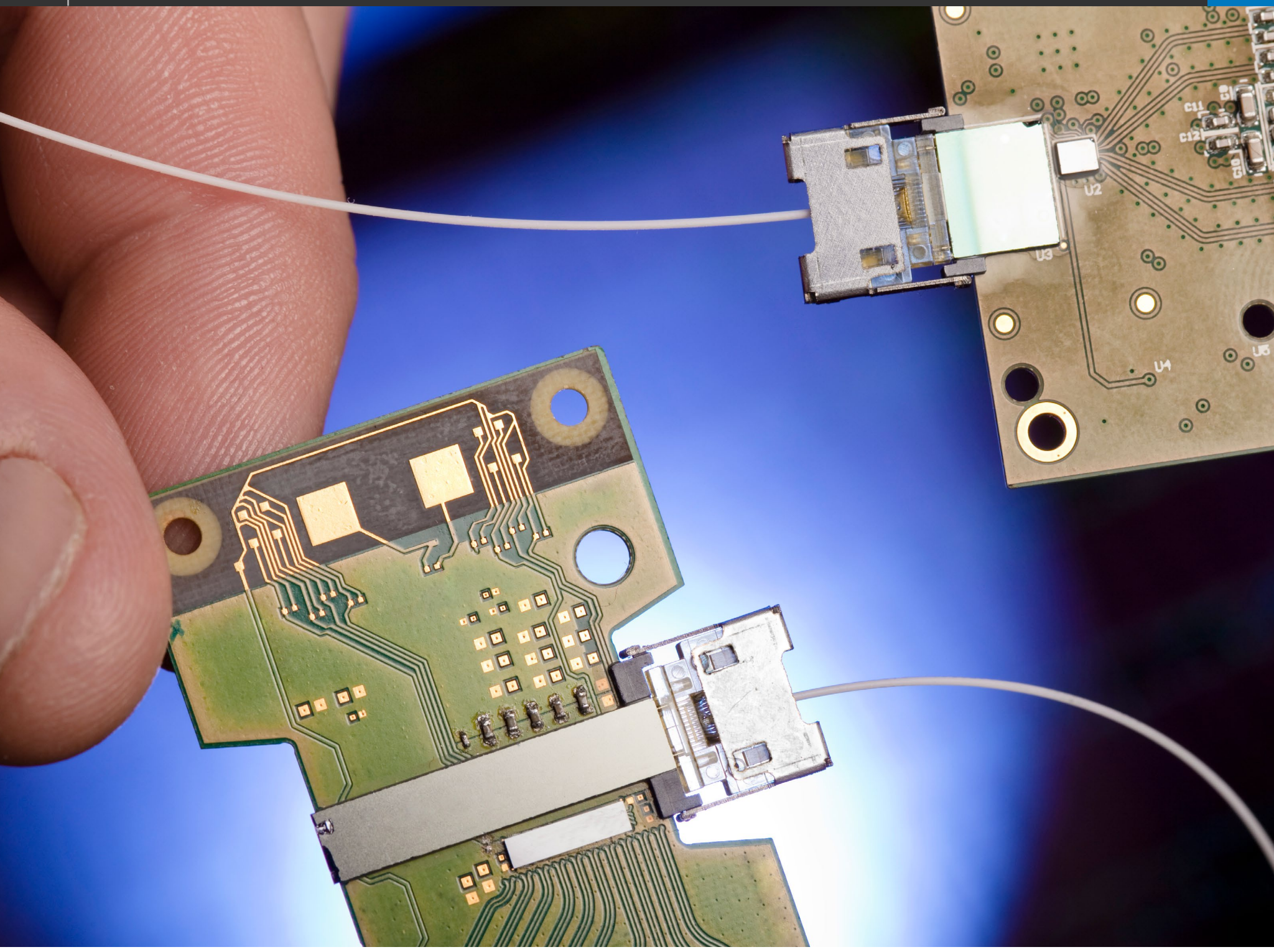
## DEEP, the conservative exascale

Mixing the concepts of cluster of accelerators and cluster with accelerators, the DEEP architecture project, strongly supported by Intel, is characterized by the use of current technologies and a lot of adaptation work for scaling up.

The idea is to couple a cluster of CPUs with a second cluster of x86 accelerators (the "Booster") equipped with its own network of ultra high performance interconnects.

The figure describes its software stack, which includes, in particular, a middleware dedicated to managing the Booster, a specific compiler allowing the splitting of applications into Cluster or Booster tasks, and an API for the execution of existing applications without modification.






# 3 – Innovations to watch

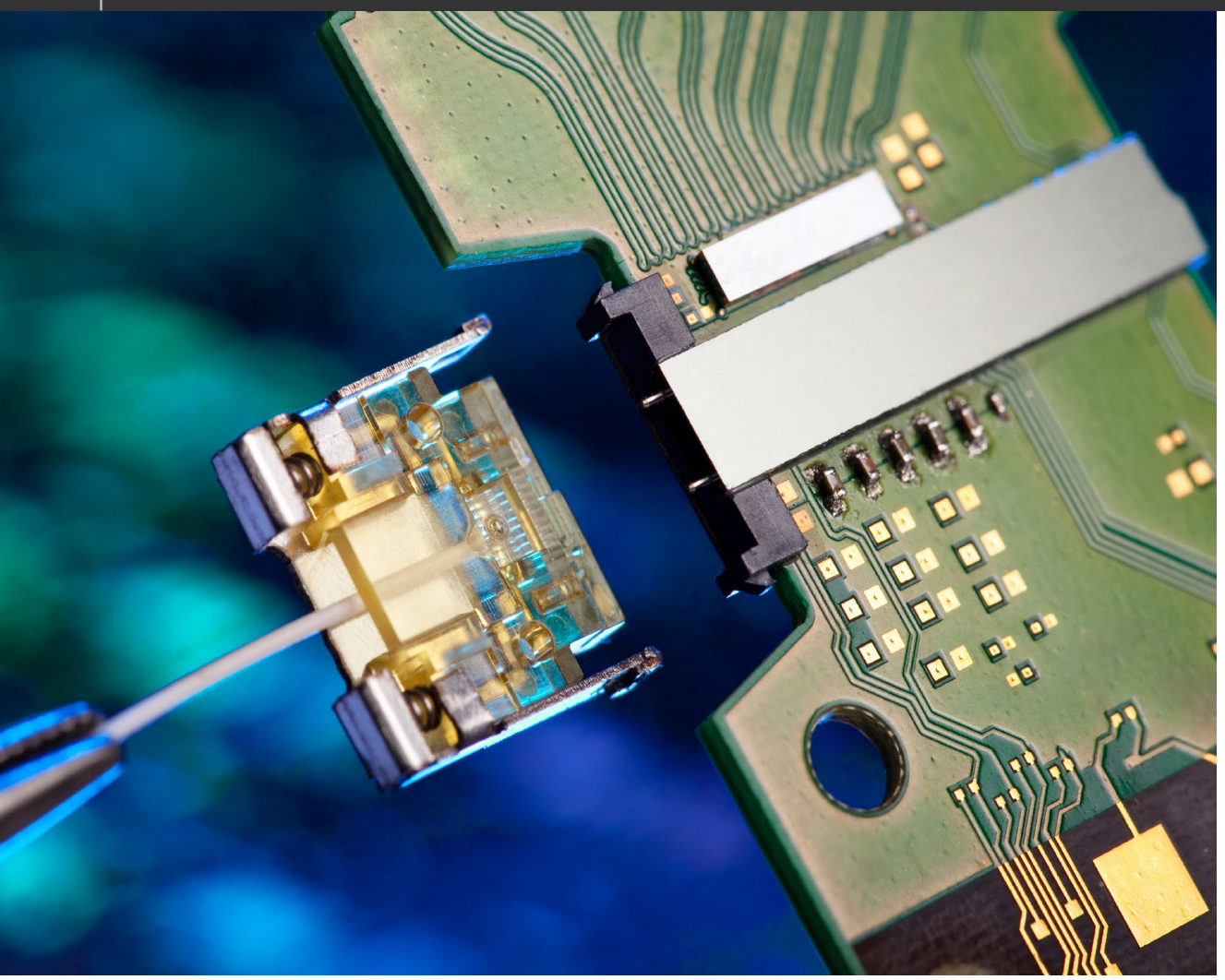
**Whatever the supporters of the evolutionary approach might say, exaflopic platforms will feature technologies that don't exist today. Ah, but which ones? That's the question....Here are three, of different scope, that can reasonably be bet on.**

The spectrum of technological research in the IT world is virtually infinite. In order to identify the most promising candidates for the exascale, let's start from the weaknesses generally identified in current architectures. The first is, of course, interconnects. Since they play a crucial role in data transfer at every level, and since increasing component integration won't completely eliminate them, these passive links could continue to figure among the major performance bottlenecks. That is, unless photonics technology becomes widespread – an increasingly likely prospect between now and 2020.

**Light inside the tunnels**

The growing number of silicon photonics research projects does in itself hold great promise. The general idea is to integrate optical links into electronic components so as to make the most of their intrinsic characteristics of density, compact size and energy efficiency. In addition to these benefits, these same electronic components can be simplified, leading to lower production and operating costs.

The good news is that research is currently at the prototype stage. At the leading edge in this field, Intel already presented a photonic rack (manufactured by Quanta Computer) with an initial 100 Gbps implementa-



*The 50 Gbps photonic transmission device (on the left) handles the connection between two silicon processors that modulate and demodulate laser pulses. Downstream, the signal, which has become electric, is processed in the conventional way. In order to guarantee maximum performance, the optical link connection design uses a passive alignment mechanism: the electronic component pins serve as guides for positioning the laser beam.*

tion of the technology. Beyond demonstrating that optics could actually be integrated at the system level, the company wanted to highlight its advantages in the simplification of installations (fewer wires = fewer failure points), a very low energy consumption and above all the ability to disaggregate HPC platforms by separating the classic subsystems (computing blades, storage units, network interfaces, and power supply) at comfortable distances from one another.

Of course, this is only the first stage. The goal is to implement photonics at every possible level of the current architectures – PCIe, Ethernet, etc. - to enhance their sustainability and thus to enable the rest of the industry to capitalize on the investments already made. If we are to believe the leading specialists in the field, this objective should not be too difficult to achieve. From a technical standpoint, adding optics-only parts (laser diodes, modulators, etc.) to motherboards is similar to surface-mounting processors. As a result, in terms of industrial design, we are moving towards discrete functional modules that are interchangeable and stackable, and that include both types of technology. And when it comes to energy consumption levels, the first prototypes display remarkable efficiency, on the order of 1 mW per Gbps (1 pj/bit).

**Memory in 3 dimensions**

Let's now take a trip down Memory Lane. The usual problems (bandwidth, latency, capacity) should find a comprehensive solution when modules are stacked on the same package as the processors. This direct integration seeks to achieve three interrelated objectives: eliminating poorly-performing copper links, avoiding having to amplify signals, and scaling memory performance at the same level as processor performance in order to maintain a constant byte/flop ratio over several generations. Another advantage is that it is less expensive to assemble stacks than to manufacture RAM modules. All that circuit manufacturers have to do, for stacked memory to become a reality, is perfect intra-silicon connections.

Research projects underway at Intel, Micron and Samsung would lead us to believe that we will get there in a relatively short time, with a stated performance level of 1 TB/s. Electronics engineers optimism is consistent with the announcement made last year by the Hybrid Memory Cube Consortium of a global standard, supported by the entire industry, that defines the practical outlines of these very high density memory modules. Additional confirmations are to be found in several roadmaps for the 2016-

2018 horizon, which indicate that products manufactured for mass distribution have already anticipated integrating this technology.

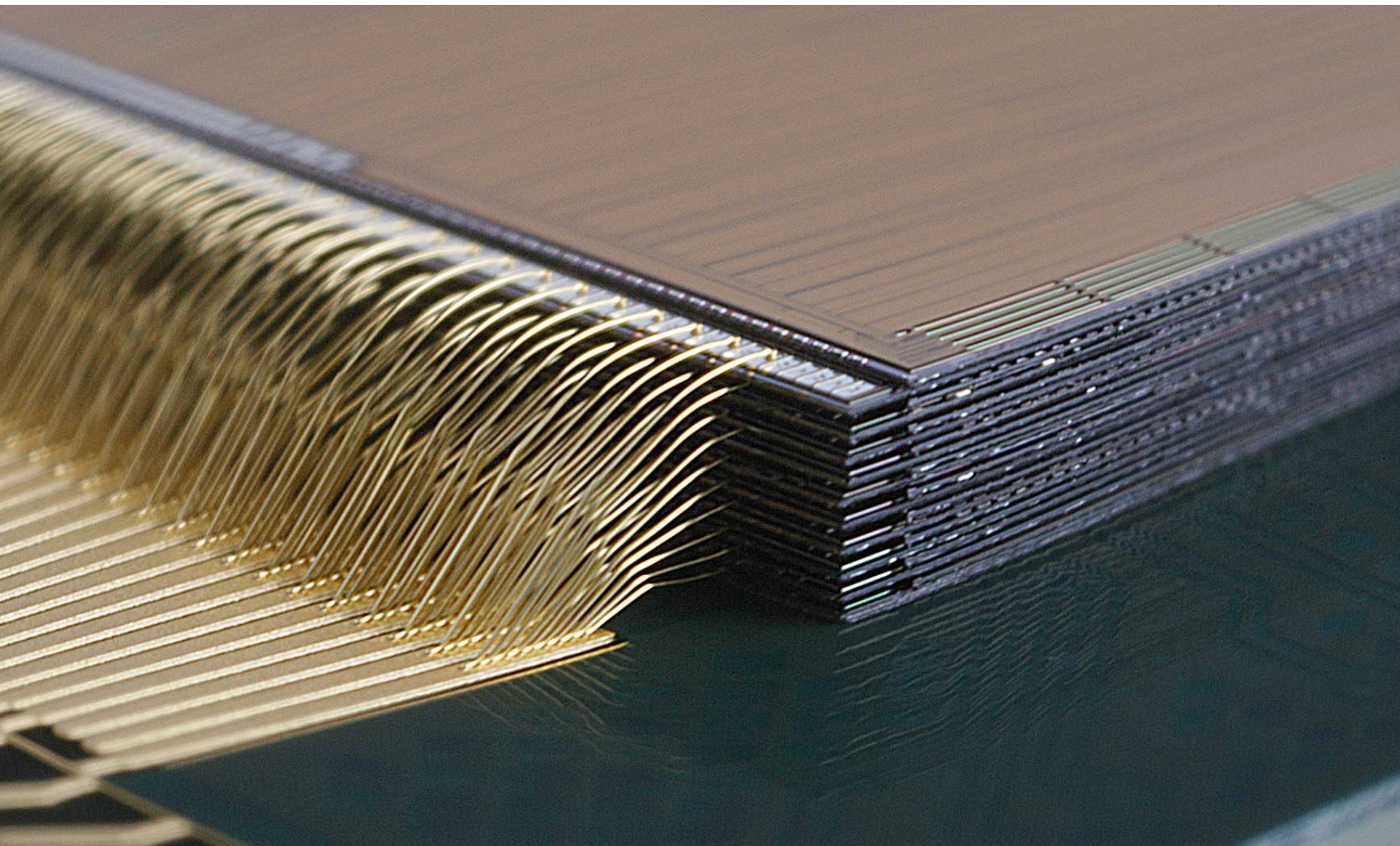
**Optimizing the network exchanges' energy efficiency**

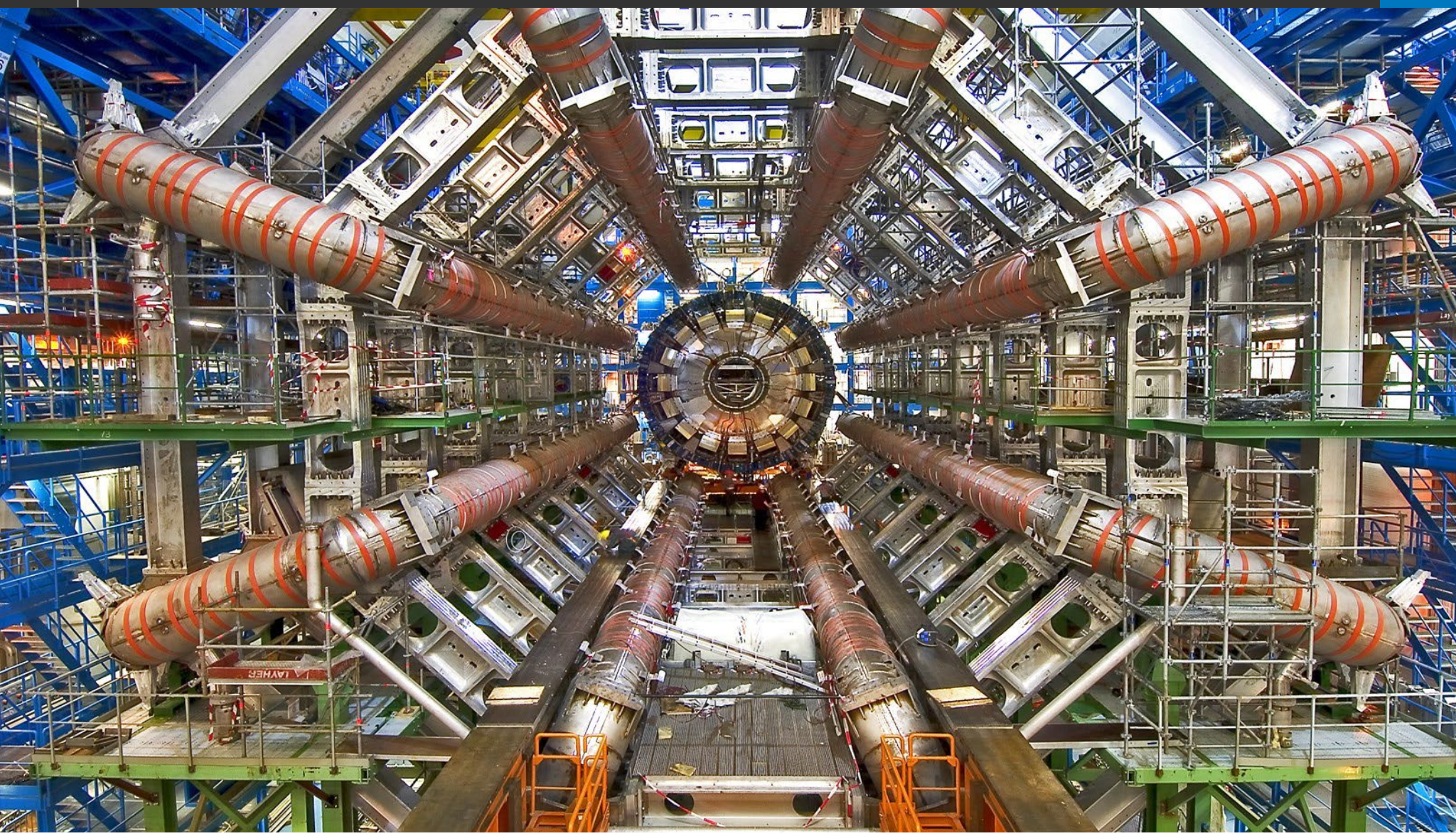
Let's wrap up with a proposal from researchers at the AMES Laboratory and Old Dominion University in the US. In the race to improve energy efficiency, several mechanisms to reduce energy consumption have already been implemented at the very core of CPUs and ancillary processors. Most hardware drivers are capable of reducing voltage dynamically and modulating clock frequency in a more or less optimized fashion. The idea developed in the article in question, which dates back to August 2013, consists of applying this frequency scaling to communications processes and to network connections in particular.

In order to do this, the authors worked out several strategies to break up data transfers into consecutive phases. Since these phases are characterized by different time lapses separating calls, the authors suggest adapting the component power supply to the functional nature of the transactions. Such strategies obviously involve real-time analytical calculations that could slow down throughput (these calculations are performed at the application runtime level). But depending on the strategy implemented, experiment results show that a delicate balance between the complexity of the analyses and the amplitude of the changes in frequency results in a mere 2% decrease in performance, whereas the theory's energy consumption reduction goals are achieved.

This is only a first attempt which will have to be validated over a more extensive range of cases than the NAS benchmark and the GAMESS application used in this instance. However, the technique has been tested, and extending it to a dimension as essential to the exascale as network communications would seem to be almost essential.

*In its current temporary implementation, stacked memory continues to use copper links to connect to motherboards. Optimal performance will only be reached when the connections to the associated processor are made via silicon.*





# 4 - Why the exascale will be data-intensive

**The numbers are in: the generation of scientific data is experiencing hyper-exponential growth. This anticipated flood of information will require not only that systems and applications adapt in profound ways, but that research practices change as well.**

For a long time, data-intensive applications were considered to be trivial. Uncomplicated from a technical perspective, they didn't make excessive demands on computing resources, used classic data structures (whatever their volume), and they didn't excite much interest in the information systems research community, which as a result relegated them to a kind of second-class status, well below the set of high-profile major problems on which the scientific world spent its time. Those days are gone.

Born at the same time that social networks and digital marketing were expanding on a global scale, the concept of Big Data now applies to a broad range of applications, including HPC codes, based on the generally accepted idea that answers to some of our questions are likely to be found in the data that we already have but that we don't know how to fully process or correlate yet.

**> 4,300% per year**

A simple look at the numbers is enough to gauge the scale of the famous data deluge. The annual growth rate of digital information worldwide through the year 2020 (i.e. the theoretical horizon of the exascale), is estimated at 4300%. By then, the global volume of analyzable data should reach about 35 zettabytes, or 35 billion terabytes. So much for the IT world in general. As for the HPC domain in particular, it is interesting to take a look at the amount of data generated today by the major application areas. An interesting **study** published by researchers at Oxford University, Indiana University, and Microsoft, allows us to assess the complexity of the challenges that we must deal with from now on. Having classified applications into 3 categories (observation, experimentation, simulation), the authors note that, for example, each year medical imaging generates 1 Exabyte of new data, global monitoring 4 Petabytes, LHC 15 PB, climate modeling 400 PB, and genome

*Human genome sequencing now generates 700 PB each year. By 2020, the volume of data to be analyzed should reach 10 EB...*

sequencing 700 PB – but estimated to be 10 EB by 2020. Until now, the main challenge was to generate these data. From now on, it will be to analyze them.

**The end of affinity**

This situation becomes even more complicated when one takes into consideration the functional evolution of HPC applications. Until recently, most codes were characterized by common features that made them executable on classic architecture platforms: among them, we could cite a relative spatial and temporal affinity, the ability to store the problems in random access memory, and mostly static data sets. Today, the gains resulting from these applications are changing the givens: the data are becoming dynamic in such a way that they are losing their affinity, and beyond a certain level of complexity, the problems can no longer be contained in random access memory.

As a result, in addition to the programming constraints inherent to scaling, we have to add application functions specifically dedicated to managing data input and output. Depending on the type of application and on the state of the art, the list is long: quantification of uncertainty in the data themselves, the models, and the methods; modeling, simulation and analysis of extended networks; automation of analysis and if possible of post-execution discovery, i.e. in the warehousing phase...

**New science, new skills**

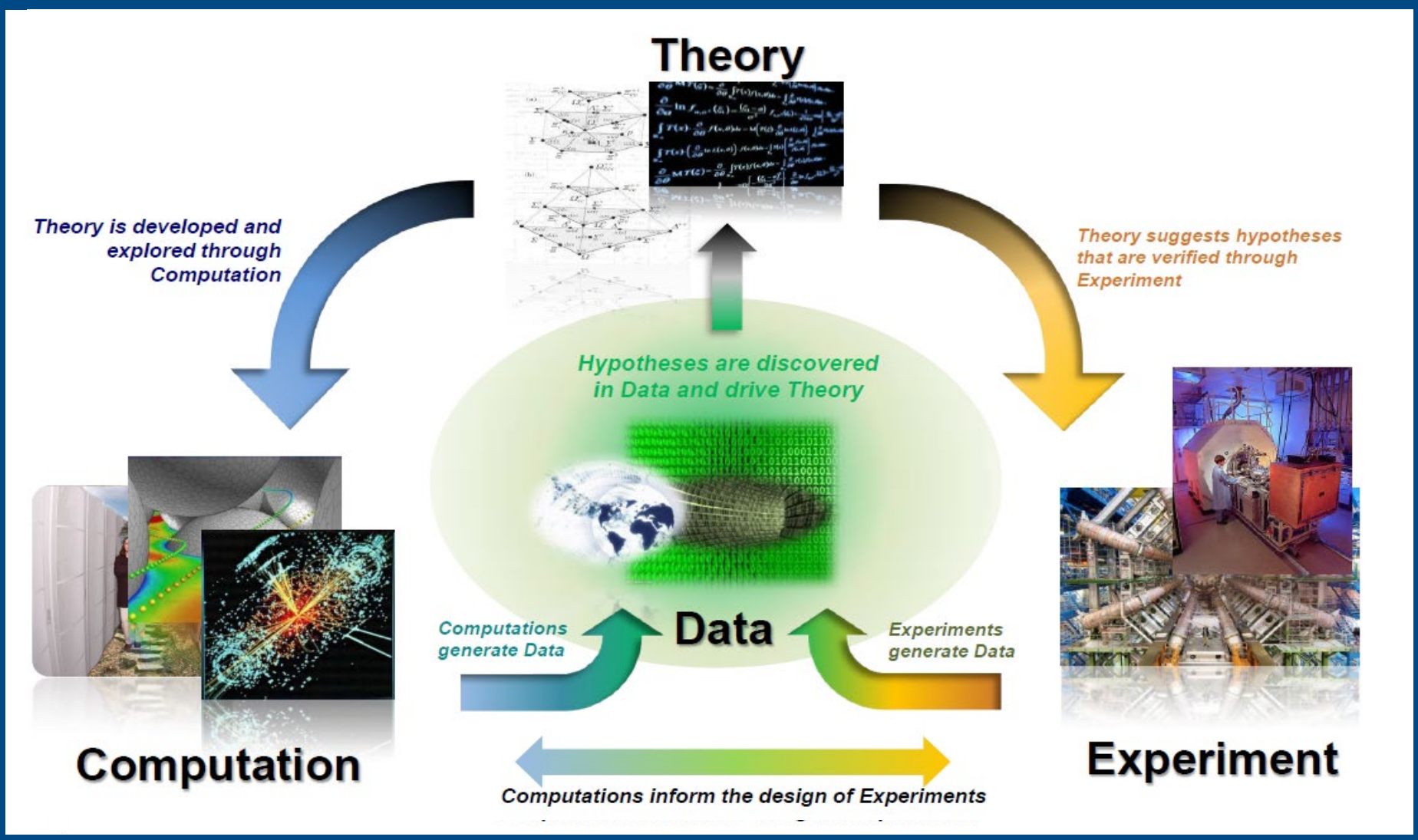
This proactive approach, which is already determining a certain number of works in progress throughout the HPC community, is obviously already present in the business world. Big Data has received a big push from marketing, as evidenced by heavy investments in R&D by major general purpose (HP, Fujitsu, Dell...) and specialized (DataDirect, NetApp...) vendors alike. It is also interesting to note that this private research corresponds fairly closely to public research - both taking place on three different axes. First of all is the acceleration of input and output, aimed at democratizing the TB/s. In this area, everyone is focusing on overlays compatible with distributed file systems - overlays

that will allow warehousing, including remote warehousing, to be prioritized more efficiently. More interesting from a technical perspective, infrastructure convergence aims at combining computing and networking technologies to allow pre- and post-processing of native analysis routines within the warehousing infrastructure itself. Finally, in almost every laboratory, researchers are actively working on non-volatile components, based on the reasonable assumption that warehousing memory, in one form or another, will soon replace hard disks due to the expectable savings in energy efficiency, deployment costs, and physical space.

That the exascale will be data-intensive is now certain. Until then, the sequence of hardware and software developments in scientific data management and processing will probably increase research productivity. However, it will also make it necessary for scientists to acquire new skills. We must resign ourselves to that. ■

## When research reinvents itself

**The reign of data has begun and, as this diagram proposed by John Johnson from PNNL shows, the entire scientific method will be affected. The explosion of the volume of scientific data should lead to a sort of cycling of research transactions in which calculations and experimentation become two transitory phases that will in turns refresh, almost automatically, the information available and the theory derived from it. Granted, the community is not there yet. But a certain number of scientists and engineers are already feeling the change. And while some are still asking whether Linpack is appropriate for guiding architectural choices concerning hardware development, many others already know that the only valid measurement – which is still to be invented – will be one that, taking into account the entire spectrum of data-oriented problems, will allow us to compare systems on the basis of their ability to produce exploitable knowledge.**



INTERNATIONAL
SUPERCOMPUTING CONFERENCE

# ISC'14 THE HPC EVENT

## June 22 – 26, 2014, Leipzig, Germany

# Join the Global Supercomputing Community

www.isc-events.com/isc14

/verbatim

# JACK WELLS

## DIRECTOR OF SCIENCE

## OAK RIDGE LEADERSHIP COMPUTING FACILITY

INTERVIEW BY STEPHANE BIHAN

**On the agenda:**

- ORNL, behind the scene
- The implications of adopting GPU acceleration
- The roadmap for the successors of Titan
- The benefits of the US-China competion for supercomputing leadership

*Mr Wells, can you briefly describe what Oak Ridge National Lab is and tell us about your role in this organization?*

ORNL is the US DoE's largest science laboratory with full class capabilities in material science and engineering, nuclear science and technology, neutron scattering technologies and computing and computational sciences. Our Computing and Computational Sciences directorate is one of the five science directorates at the Laboratory and each directorate has approximately 600 employees. Globally, ORNL counts for about 3,300 staff.

The lab inherited from the Manhattan project during World War II. The vision that emerged post World War II was at the intersection of material science and technology, and nuclear science and technology. Within the last fifteen years, the lab has really grown capabilities in neutron science and technology, and computation science and technology. The mission of the Oak Ridge Leadership Computing Facility (OLCF) is to procure and operate the most capable production supercomputing facility as we can and make these capabilities available for high impact research by US industry, universities and other federal agencies. In addition to our user facility, a second one, funded by the National Science Foundation and led by our partner, the University of Tennessee, is located here. There are also two research divisions: one is the Computer Science and Mathematics Division and the second one is the Computer Science and Engineering Division.

Today, in the OLCF, we operate Titan, the number 2 supercomputer in the world on the Top500 list. We operate it for users and access is granted through open and competitive peer-reviewed allocation programs. As Director of Science, my role is to engage the user community, to ensure high-quality and cost-effective science outcome, to manage the allocation program and user policies, to collect user requirements for future procurement and to ensure the effective communication of research results.

*Titan has been in operation since early 2013. How would you describe the process of building such a giant machine? How long did it take from initial sketching to end users delivery?*

The process of building a supercomputer is actually a big part of what we do. In January 2009, Ray Orbach, then Director of the DoE's Office of Science in Washington (that is, the lead of all the science investments in DoE), signed a mission need statement for a 10 to 20 Pflops upgrade of the OLCF system. The January 2009 date is really when the OLCF-3 project, i.e. the Titan project, started. In December 2009, the next step was the approval of the Acquisition Strategy and Cost Range. We proposed a set of alternatives to accomplish the mission need and we then began to make progress on meeting it. This all fits within a very structured project management process that had to undergo peer-review and management approval. The next step came in August 2011 when the OLCF received the approval from DoE to order the system that was selected, a Cray XK7.

*Did you discuss with other vendors?*

The path that was chosen for the OLCF-3 procurement was actually an upgrade of Jaguar rather than a brand new machine. That is why it became a sole source procurement.

***"In order to reach exascale, the nodes will have to be more heterogeneous. We think this is what the future will likely be.***

*What was the overall budget for building this flagship? Can you also give us an estimate (and breakdown) of its operation costs?*

This is in the order of a hundred millions of dollars. There are three main elements in the operating budget: one is the lease for the supercomputer. We don't purchase the machine, we amortize the cost over a period of 4 or 5 years through a lease. Therefore, the cost of the machine becomes part of the operating budget. The other two elements are the cost of the electricity and the cost for our staff.

*What is the power consumption of Titan?*

If you look at the amount of power Titan consumes when it's running compute intensive applications like the Linpack benchmark, it is about 8.2 MW as indicated in the top500 list. In terms of cooling you have about an additional 20%, which makes something close to 10 MW for Titan proper. But our computer facility has a total power supply of 25 MW as we also run other computers and petascale systems. There is the Kraken machine we operate for the National Science Foundation, another petascale machine that we operate for NOAA, and other laboratory scale servers and workstations and clusters.

*Why did you choose a hybrid, GPU-based architecture? Was NVIDIA influential in ORNL's choice?*

This is actually the result of our application requirement process. From interviews and surveys with our user community back in 2009, we determined that the top requirement for a new system or a system upgrade was to deliver more raw flops in order to perform the needed science simulations. More raw flops on the node ended up being the number one requirement, although some codes were starting to reach limits in their ability just to scale out in a flat MPI space. Therefore, rather than just adding more nodes or more cores as we grew the size of Jaguar, we decided to make the nodes more powerful and flop rich. Then, from our early discussions with NVIDIA, we came convinced that GPUs would be able to satisfy that requirement for many of our applications.

The community thinking at the time, and I think it's even stronger today four years on, was that in order to reach exascale, the nodes would have to be more heterogeneous. That seems to be a strong trend. We thought that this is what the future would likely be and we decided that it would be a good role for a leadership computing facility to step out and take that on.

*Did you see any risk in adopting GPUs?*

Yes, the main risk we identified and worked to manage was that our codes would not be able to run on that machine and effectively use the GPU accelerators.

*So was porting some codes part of the evaluation process?*

Actually we had the application readiness review where our strategy for getting codes ready to run was peer reviewed. If we had not passed that peer review, we probably wouldn't have been be able to proceed with the project. But we did. We were able to convince the peer review panel that we had a good strategy for getting the code ready.


*Which technical strategies have been adopted by ORNL to migrate applications from CPU parallelism to GPU acceleration?*

This is an important point. As I said, earlier in the project, we had to present our strategy. A huge part of that strategy on the application itself concerned forming a Center for Accelerated Application Readiness (CAAR). And that was formed as part of the Titan project as a way to manage this risk, to manage the task of having some user codes ready to go on Titan. Through CAAR, we focused on a number of application codes from our user programs. We tried to select them for their diversity, so that their requirements in algorithm would represent a good variety of computation science problems. The codes selected included LAMMPS, a molecular dynamic code outside the National Lab, S3D, a combustion code, LSMS, a material science code from Oak Ridge, Denovo DM, a neutron radiation transport code for nuclear reactors which is being developed here, and also CAM-SE which is a community atmosphere model with spectral elements code that is part of a big climate code.

*How was CAAR structured?*

There was a team assigned to each application that included an application lead from OLCF, a CRAY engineer, an NVIDIA development engineer and other application developers. If we needed someone from another university or from another laboratory, we would bring them on contract. We formed little teams to port the code for each application. On average, the cost was one to two person-year of work to take a code that was running on Jaguar and get it ready to go on Titan.

Now, we did not take on the task of porting the whole code because so many of these codes would be community codes that have a lot of different components used for a lot of different problems. What we did was trying to select any science problem that would be both manageable and attractive. And that's an important concept when you are doing something like this, just trying to bring focus so that you can be successful within a finite amount of time and a finite budget. Of course, along with benchmarks, these codes were also part of the acceptance suite for the machine.

Throughout that process, there were lessons to learn. One is that in order to get the code ready for a specific accelerator architecture, there are major code restructuring required. This code restructuring took 70 to 80% of the effort. It has nothing to do

***Code restructuring is really important and it is going to be essential for the next generation machines. You need to do it whether you use CUDA, OpenCL or OpenACC...***

with any specific programming model for offloading to the GPU: if that were an Intel MIC processor, we believe the same 80% of work would have been necessary. It has to do with preparing the code to work well in the context of this hierarchical parallelism.

*In other words, you modified the algorithms rather than the science?*

That's right, and this was related to how the memory was structured in the machine. For instance, some arrays in the code had to be reordered, arrays of lists had to become lists of arrays, different kind of things like this, so that when you get to the bottom of this memory hierarchy, you will have a chunk of well-organized work to head on to the GPU accelerator.

This is really a combination of various expertise: science, numerical engineering, processor architecture, etc. We started to do that work on Jaguar, the old CRAY XT5. And almost every one of the codes went two times faster. You see, working on code is good. Thinking hard about the way codes will run on this big machine is good. Sometimes, codes can be run in a production mode for a while and no one takes a hard look at them. That was therefore a good outcome. We also believe that this restructuring actually envisions things well for the future. And this is why you need to do it whether you use CUDA, OpenCL or OpenACC. As I said earlier, even if you want to move this to a MIC accelerator based machine or something else like this in the future, this kind of restructuring is really important. And it is going to be essential for the next generation machines.

*How is the support to Titan's users organized in terms of expertise and tools?*

The first thing coming on is the Scientific Computing Group, which consists of computational domain scientists. They bring a combination of domain science expertise and extensive experience with Titan and its architecture into partnerships with science teams that are awarded big resources on a machine. This domain science expertise covers a wide range of computational approaches and disciplines, including astrophysics, biophysics, nuclear physics, chemistry, climate science, material science, engineering, mathematics, visualization, etc. The models were created at the founding of our facility, about ten years ago in 2004, to provide an advanced support and collaboration with projects so that our resources can be used effectively. This kind of scientific computing group is becoming a best practice in supercomputing centers in the US. In order to have the users use these complex machines, you need this kind of advanced support.

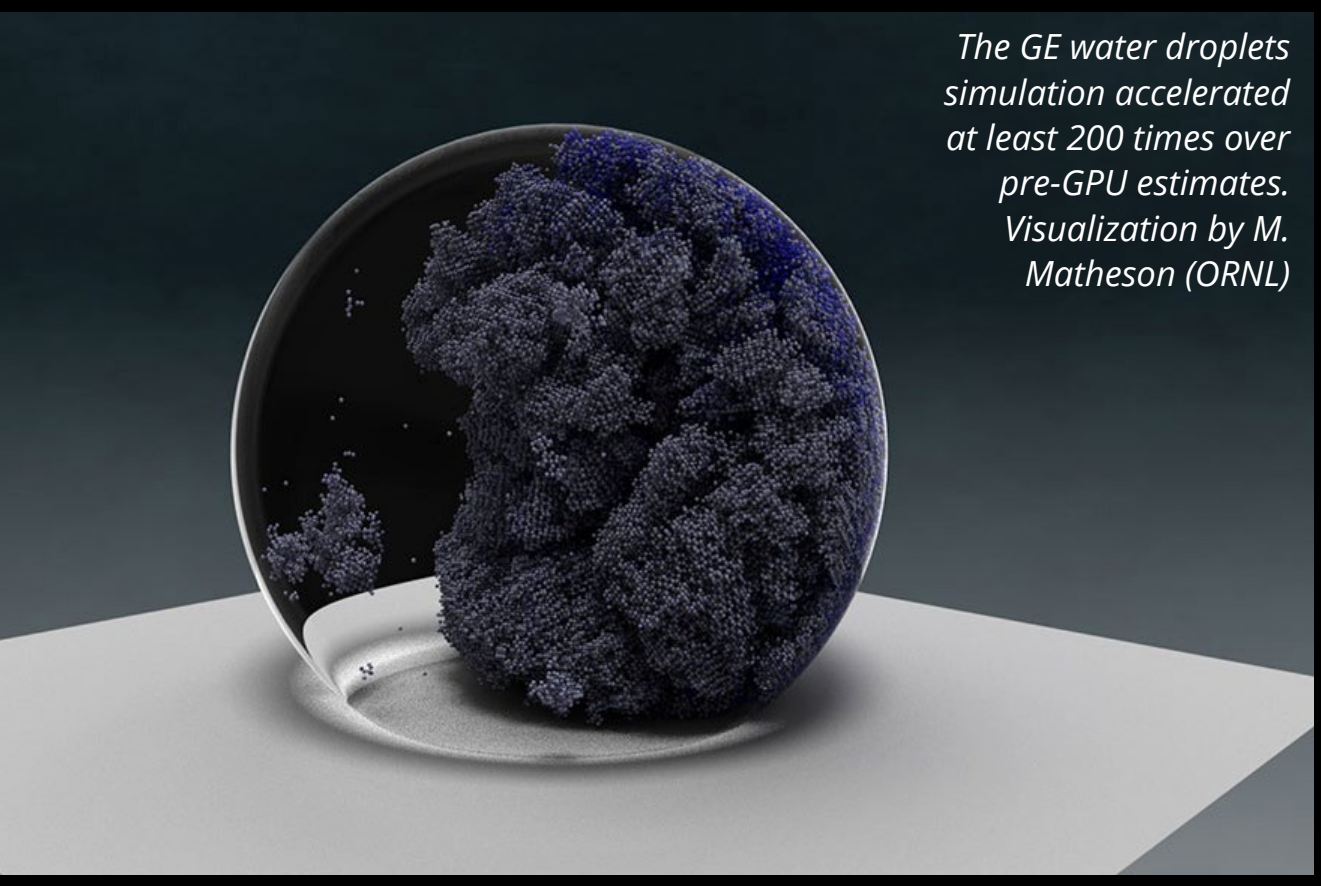
We also have another group called User Assistance and Outreach that handles a wide variety of tasks: set up user accounts, manage the help desk to provide the front line technical support and triage of our users' needs, create user portals and web pages, create and execute a training curriculum that spans a range

of topics like introduction to supercomputing, accelerators and management techniques, etc. These two groups work hand in hand: one has a more traditional role for computing centers and the second one, the scientific and computing group, is less traditional but we think it is essential for the work that we do.

OLCF provides a wide spectrum of tools to its users with the aim of offering an environment including the widely used and familiar tools, many open source tools and a certain number of commercial tools. In this case, we work closely with the vendors to ensure support for jobs that are in scale with Titan.

From the organization point of view, we have a Programming Environment and Tools group at the OLCF that really spans our Oak Ridge Leadership Computing Facility. We also have the Computer Science Research Group within the Computer Science and Mathematics division which carries out cutting edge research in computer science. We have this major arrangement here to make sure that the tools for cutting edge computation research and operational support are combined. For example, ORNL people in this group are active in the OpenACC, OpenMP, and MPI standards organizations to push in these standards and their implementation, in order to better meet the needs of our users. We are working on better algorithms for collected communications, better MPI fault-tolerance and various techniques to facilitate porting codes to new platforms like Titan and beyond. These are some of the research activities that are paid for by the Computer Science Mathematics program within DoE. These people helped us manage the development of these tools by working with our vendors. The vendors are an important part of the story here.

As part of the Titan budget, we also have contracted with several key tool vendors to ensure that these new tools really work on the scale and heterogeneous nature of Titan. In particular, we have relationships with Allinea for the DDT debugger and the Technical University in Dresden for the Vampir profiler suite. More recently, we have added the CAPS Entreprise team who focuses on compilers for the Titan GPU accelerators and on the OpenACC language. This is in addition to the tools that are provided by CRAY.



*The GE water droplets simulation accelerated at least 200 times over pre-GPU estimates. Visualization by M. Matheson (ORNL)*

***Four of the six SC'13 Gordon Bell Prize finalists used Titan to run their simulations. Can you elaborate on one or two science challenges that Titan has been be able to solve?***

This is what it's all about, the outcomes. There is a paper in the Gordon Bell competition from a Swiss American team: Peter Staar from ETH at Zurich, Thomas Maier and Thomas Schulthess at ORNL. They were performing superconductivity calculations on Titan at 15 PFlops, effectively using the architecture. The superconducting materials

conduct electricity without resistance and therefore without loss, showing a promise for important applications in power transmission and magnets. From the point of view of science, this team solved the two-dimensional single-band hypermodel in a completely converged fashion. They showed that this model correctly reproduces a phase diagram for high temperature superconductivity in the cuprate based superconductors, those high temperature superconductors that include a copper. These models were proposed for high-Tc superconductors in 1987, just a year after they were discovered. Literally thousands of papers were written on this subject over the last 26 years. This team has for the first time successfully converged the solution using a new extremely effective numerical algorithm that scales very efficiently on Titan. Compared to a CPU only machine - the XE6 that instead of one CPU and one GPU has two CPUs - Titan achieved the simulation 6 times faster and 7 times more energy-efficiently. We take that as a tremendous outcome.

The second topic I want to emphasize is actually a project led by Masako Yamada, a General Electric (GE) Global Research computational scientist. At GE Global Research, they have a science project focusing on understanding how water droplets freeze upon surfaces of a wind turbine placed in a cold climate. Her goal is to understand the molecular mechanism behind the freezing of water droplets on these non-icing surfaces. To do so, in collaboration with Mike Brown at ORNL. Yamada performed a series of molecular dynamic simulations over a range of the problem parameters. These simulations are the largest and longest of this kind ever attempted, and they were able to replicate GE's experimental result. GE knew from experiments that the creation of ice is delayed on these non-icing surfaces, and that this effect becomes less prominent as the operating temperature decreases. Yamada was able to do this successfully with reproducible results and now GE is in position to use Titan to study this phenomenon and try to engineer solutions with the computational science approach in addition to their existing experimental capabilities. For that, she received an IDC HPC Innovation Excellence award at SC'13.

*ORNL has strong industry partnership programs (with Boeing or Ford, for instance). What proportion of Titan's computing resources and computation time do these programs represent?*

Our industrial partnership program does not assign a specific amount of time on Titan for these industrial projects. Instead, we work with our partners to ensure that they are effective in accessing time on Titan through the same three pathways as any university-based or laboratory-based research project. The first of these pathways is the INCITE (Innovative and Novel Computational Impact on Theory and Experiment) program. We manage that program jointly with our sister facility at Argonne. That represents 60% of the time allocated on Titan. The second one, called ALCC (ASCR Leadership Computing Challenge), is also a program that our sponsor manages at Washington. That's 30% of the time. The remaining 10% is the Director's Discretionary allocation program that we run here in our center. I chair the internal committee that manages that time. INCITE and ALCC are annual calls for proposals for projects beginning either in January for INCITE and in July for ALCC. The Director's Discretionary program accepts proposals at any time during the year.

The Discretionary program exists for three main purposes. One is to allow people to have some preliminary results for the INCITE and ALCC programs because these are

***Competition between China and the US can be a very constructive and healthy thing: it enables the gauging of our progress towards our goals.***

very competitive programs, oversubscribed three times, so people need preliminary results to have successful applications. It is also used to outreach to new communities that do not correctly use leadership computing and that includes industry. The third case is that we can use some of our discretionary time for ORNL priorities. Otherwise, through all the other programs, ORNL scientists are on a level field with researchers in the US and around the world.

*In your opinion, how does Titan contribute to the innovation and competitiveness of the US?*

Our partnership program does contribute to the innovation and competitiveness in a certain way. First, our Leadership Computing Facility is a brain magnet. It attracts very smart scientists and engineers from around the world. We think adding brain power to our laboratory helps competitiveness since the basis for any new innovation and any new idea is attracting top-notch talent. This strengthens our nation's infrastructure. In addition to the worldwide expertise combined with our leadership computing systems, this motivates research from around the world and brings the most challenging science and engineering problems to our center. Having first class science and engineering research occur here is another way for our country to benefit.

*China's Tianhe-2 has retained its pole position in the last Top500 listing. How do you see China and the US compete for the world's leadership in supercomputing for, say, the next five years? Is emulation is a good thing in this respect?*

It's the mission for our Leadership Computing program at DoE and the strategic intent for ORNL to compete for world leadership in computing and computational science and engineering. This has been our focus for more than twenty years and our team is fully committed to this going forward into the future. We aspire to be world leader in this area. Let me be clear: high impact scientific achievement at a larger scale is our goal. Being high on the Top500 list is one measure that can tell if we have been successful at procuring production supercomputers for our users at the appropriate scale. It is one measure of the mission success, but not the ultimate one. Also, the broad common interest created by the regular communication that the Top500 list gives supercomputing centers such as ours is a valuable platform for publicizing the impact of our projects. People are interested in what we're doing on this complex and fascinating machine and the Top500 list provides a good opportunity to give our message out.

To be more specific, supercomputing centers across China made tremendous achievements, including the great leap to the number one spot on the Top500 list. It's clear to me that they have the vision for a productive leadership role in supercomputing within their scientific and engineering enterprise. And I expect to hear about many research achievements from their supercomputing centers. Competition can be a very constructive and healthy thing as it enables the gauging of our progress towards our goals. We also expect to see increased cooperation especially in application science areas. For example China, the US and many other countries are partners in the ITER project, a large international effort to build a fusion reactor

in France. This is a huge challenge. Computational plasma physicists and engineers from around the world are working on a variety of challenging problems associated with the eventual operations at ITER and the prediction and understanding of phenomena observed there.

International teams in fusion are running on Titan and on the Chinese supercomputer as well. This is very healthy. Running the same kind of problems running on different supercomputers is good because we will learn a lot from this type of applications on cutting-edge real sized problems. We have a vision for the importance of computational science and engineering. We believe that being first class in science means being first class in computing. And it seems to me that China shares the same vision, so that's good, that's a confirmation that computing and computational science is important for the broader science and engineering enterprise.

*Tianhe-2 is about twice as powerful as Titan. Who do you believe will reach the exascale first - in a sustainable way? And, incidentally, do you see that happen in 2020?*

To be objective, China's big leap in the Top500 list is evidence to me that they are in the best position to reach exascale first. And practically, yes, this can happen by 2020. There are many problems that have to be solved but it's a function of investment.

*The next, pre-exascale OLCF-4 project is targeting 100-200 Pflops. What hardware and software challenges are you facing for this project?*

There are two big hardware and software challenges here. First is the scale of the system in terms of the number of processing cores to be managed and the extreme parallelism to be identified and effectively expressed in the scientific codes. This is typically a hardware and software challenge that refers back to our previous discussion about what the biggest challenge on Titan was. A second challenge is the resilience of the OLCF-4 system. How do we design a hardware and software to be adaptable in the face of faults? With a large number of components, how do we do that?

*Is it going to be a GPU-based, hybrid architecture as well?*

This is a big question. We have been working on this since over the past year. The architecture of the OLCF-4 system will not be decided until this year.

*Is it going to be an upgrade of Titan or is it going to be a completely new machine?*

This is going to be a completely new machine. We have reached the end of our road with the Jaguar and Titan infrastructure. We have a brand new room for preparing OLCF-4 and we will be building it while we operate Titan. And, I must add, it is an open competitive procurement.

*Will it be a long term project like OLCF-3? Are you going to have the same vendor for the OLCF-4 and OLCF-5 projects?*

That's our hope. Our intent is to build another long term partnership for the future. That has the advantage of providing some measure of consistency for users in the face of tremendous changes in the technology. There are aspects of the environment now on Titan that are consistent with what we had in 2004, 2005 and 2006. Right now, everything is in place to change. So much has happened in the HPC industry. It's kind of a brand new day...

*In a 4-year time frame, we don't really see how processor architectures will evolve, or which of the current architectural trends may take the lead. What is your vision on this hot topic and how do you anticipate this evolution with regard to application portability?*

This is a true statement. There are uncertainties in the details of future architectures. We believe certain trends will hold based on current hardware development. Perhaps the most important observation is that the complexity in heterogeneity of the nodes is increasing faster than the size of the machine in terms of nodes. For software development, we first need to explicitly identify all available levels of parallelism that exist in our algorithms and we then need to appropriately map those levels of parallelism onto the available hardware architecture.

*Do you see opportunities to further increase the parallelism of applications?*

This is the question we asked in our most recent application requirement document. In the context of a 100 to 200 Pflops machine, where is the parallelism, how can you express it for the problem you hope to solve on that machine? The majority, not all, of our users responded that they had additional parallelism that could be expressed. Once we have identified the hardware that we will have, we will really be able to focus on how to map the parallelisms.

*We all know energy efficiency is one of the main barriers to the next scale. Do you consider the programmability of millions of cores - whatever their nature - an equally difficult challenge to overcome?*

Energy efficiency is primarily a hardware design challenge but it also has significant consequences for the programmability of the machine. One of the key software design issues is that a data move is expensive in terms of energy use. Sometimes you might say that computing is free. It is once you have data in the registers. Computing on the data once you have it on the processor is very much an energy efficient process. While in the past development focused on limiting data movement because of the cost in terms of time, the same is now extended to the cost in terms of energy. Going to millions of cores is in large part a data movement challenge which has to be done fast and in an energy-efficient manner. In the future, I think we shall talk both in terms of time to solution and energy to solution.

*Do you think supercomputing will scale up based on today's technologies or do you expect decisive technology breakthroughs? If so, which ones?*

This actually depends on the meaning of the term. What is "a technology breakthrough"? It is hard to have a common understanding of that. No breakthrough technologies are required to build a pre-exascale system in 2017 and the exascale systems in 2022. There is a lot of research and engineering required over the next eight years to deal with the scaled power consumption, resilience and programmability challenges. Whatever is your understanding of breakthrough, whatever that means, the challenges become more severe as you try to pull that date in. Now take the OLCF-4 project. In that program, we are already two major steps down the road. No world-changing breakthrough has to be achieved in order for us to deliver a machine in 2017... ■

# Intelligent Rack PDUs

## High Power and Intelligence precisely designed for HPC racks.

**Choose from a broad portfolio of high power Intelligent PDUs:**

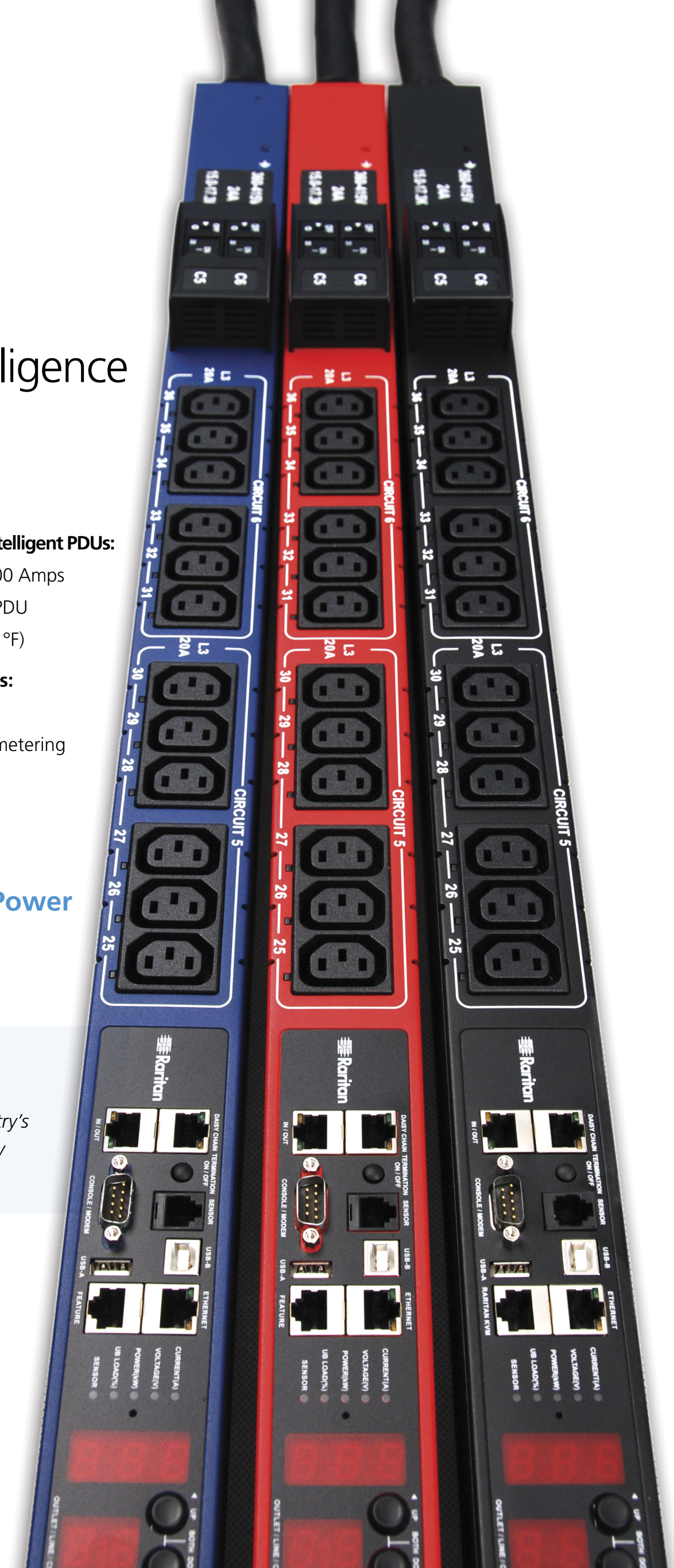
- High Power, capacities up to 55 kW and 100 Amps
- High Density, up to 54 outlets in a single PDU
- Highest ambient temperature (60 °C, 140 °F)

**Leverage the industry's smartest capabilities:**

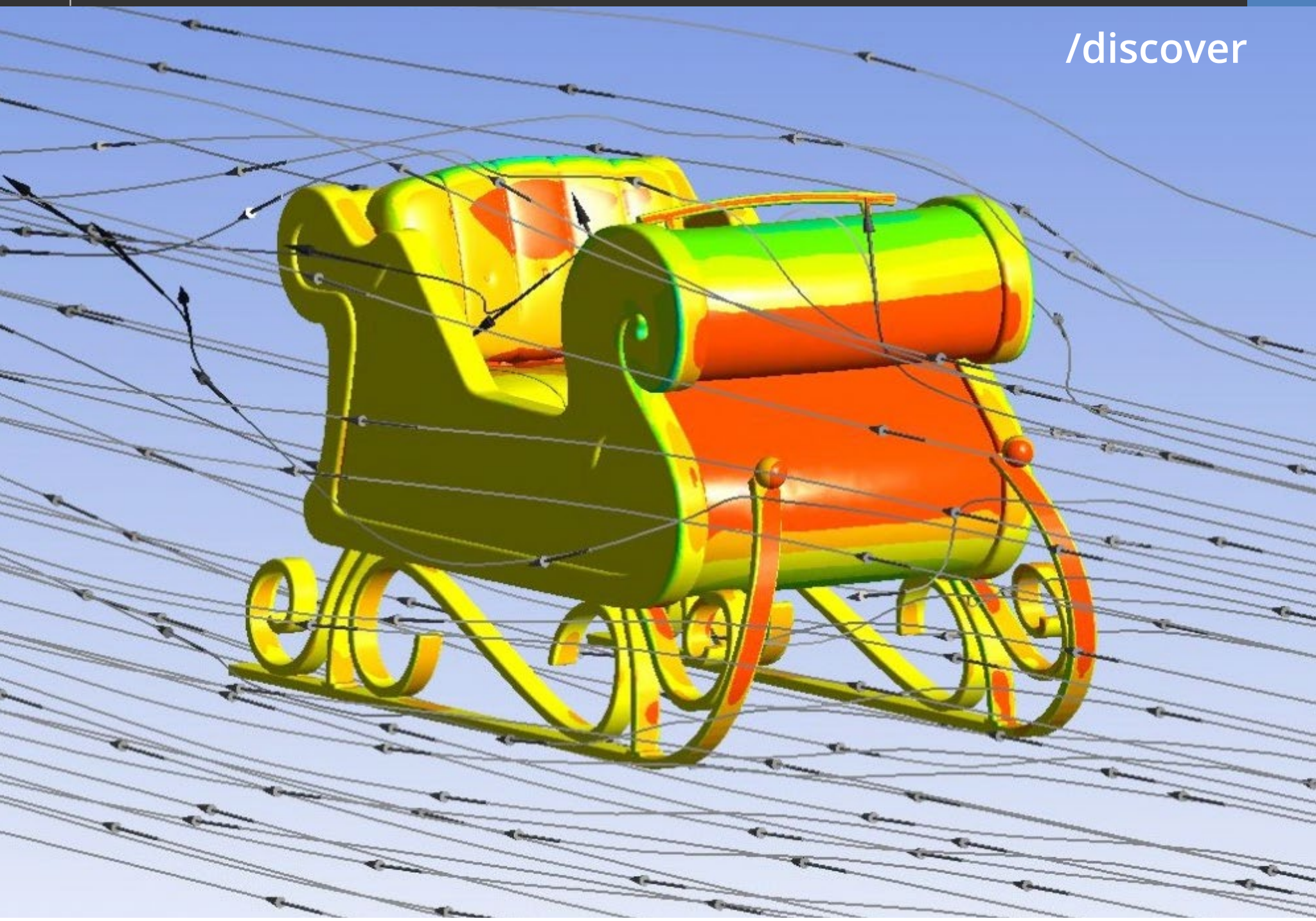
- Plug-and-play environmental sensors
- Accurate unit-level and outlet-level kWH metering
- Wi-Fi or wired networking
- Circuit breaker metering and monitoring
- Customizable to fit your HPC racks

**Visit www.raritan.com/SmartPower to learn more and explore all your PDU options.**

***Blue, red, green, yellow, orange…***
***and a whole lot more.***
*Smart managers can choose from the industry's most extensive PDU color palettes to simplify visual identification in their data centers.*




Raritan
Know more. Manage smarter.™




# TURBULENCE MODELING:

## NEW SOLUTIONS FOR (ALMOST) EVERY INDUSTRY

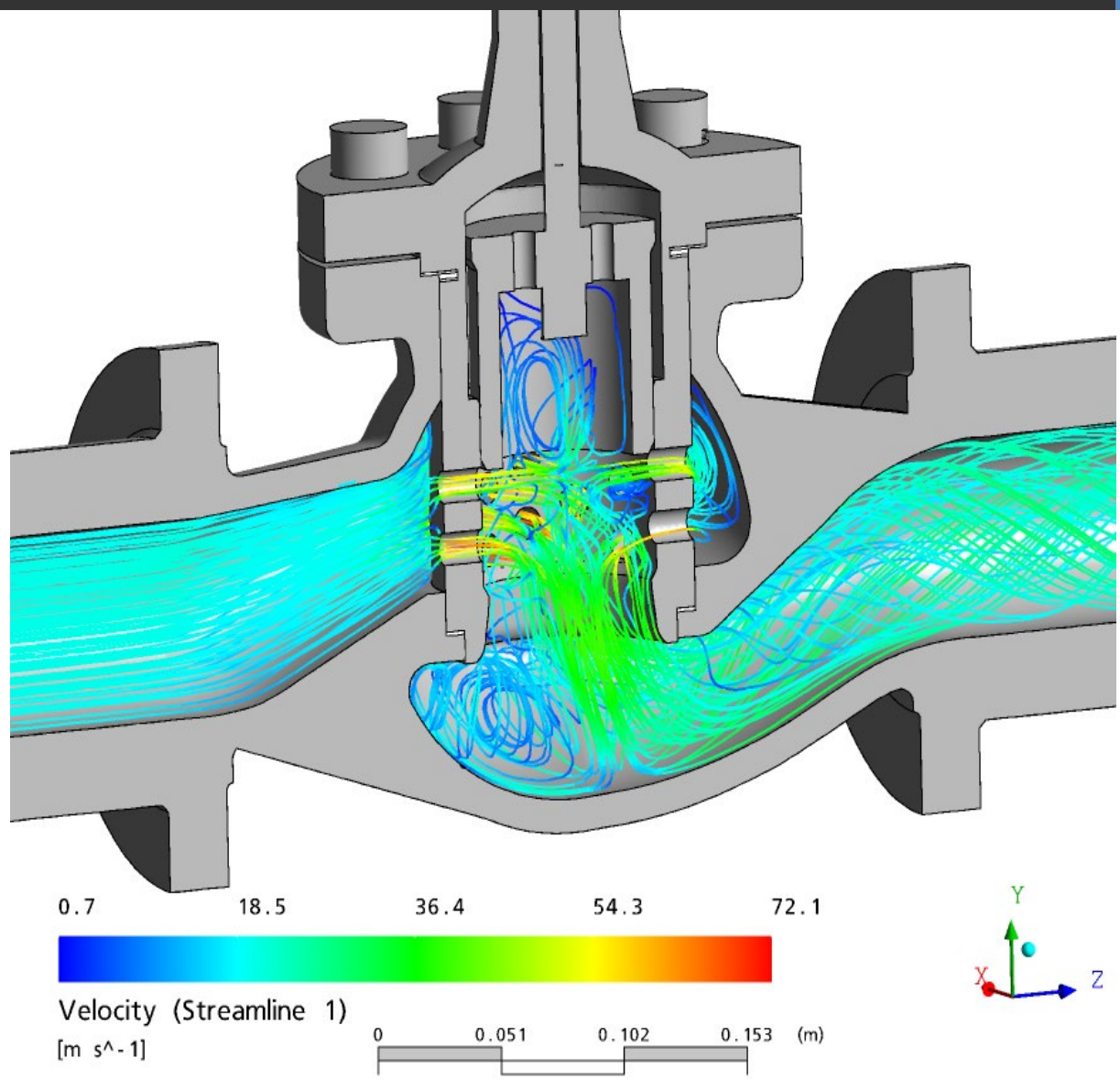
**Traditionally synonymous with inaccuracy in the simulation of fluid flows, turbulence phenomena are now globally harnessed by major CFD solutions. The question that remains is which, among the new models they propose, is best suited to your application requirements and available computing resources...**

GILLES EGGENSPIELER, PH.D.

Turbulence plays an important role in the vast majority of industrial fluid flow applications. It constitutes a classic multi-scale problem in which turbulent flow structures of many different scales interact with each other. Accurate prediction of a system's aerodynamics, heat transfer characteristics, mixing performance and other factors is key to determining performance with high precision - to the point that accurate and robust turbulent modeling capabilities are critical in computational fluid dynamics (CFD). Resolving all turbulent flow scales present in industrial applications via simulation is generally impossible with today's computational resources, but formulations can now be used that reduce a problem's complexity while still delivering accurate information on flow turbulence and its effect on product performance.

* Sr. Product Manager, ANSYS.

Because turbulence is a very complex phenomenon, no single do-it-all formulation has been found to date. There are so many turbulence models in existence that the challenge for CFD software developers is to incorporate the right subset of models, resulting in a package that is robust, accurate and validated - and that covers applications that users need. Leaders in the field don't stop there; they offer best practices so a user knows which model to use for a specific turbulence problem. They also advance physics by innovating, testing and validating new hybrid and transition models that deliver the best mix of accuracy and computational intensity.

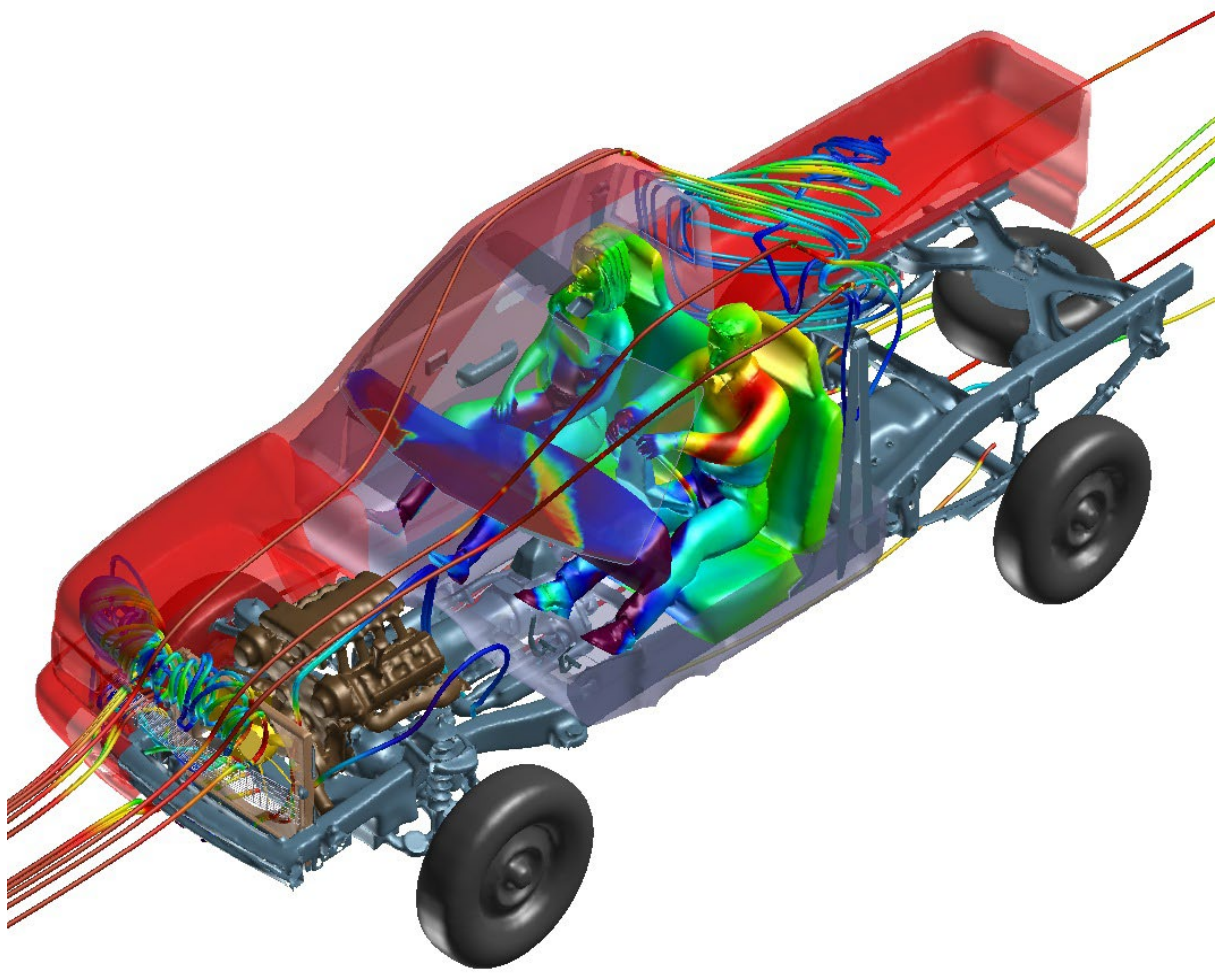



*Accurately predicting the performance of a complex valve system requires turbulence models that can capture flow characteristics in both the large main pipe and the small openings. Wall effects, which generate a large amount of turbulence, need to be captured as well.*

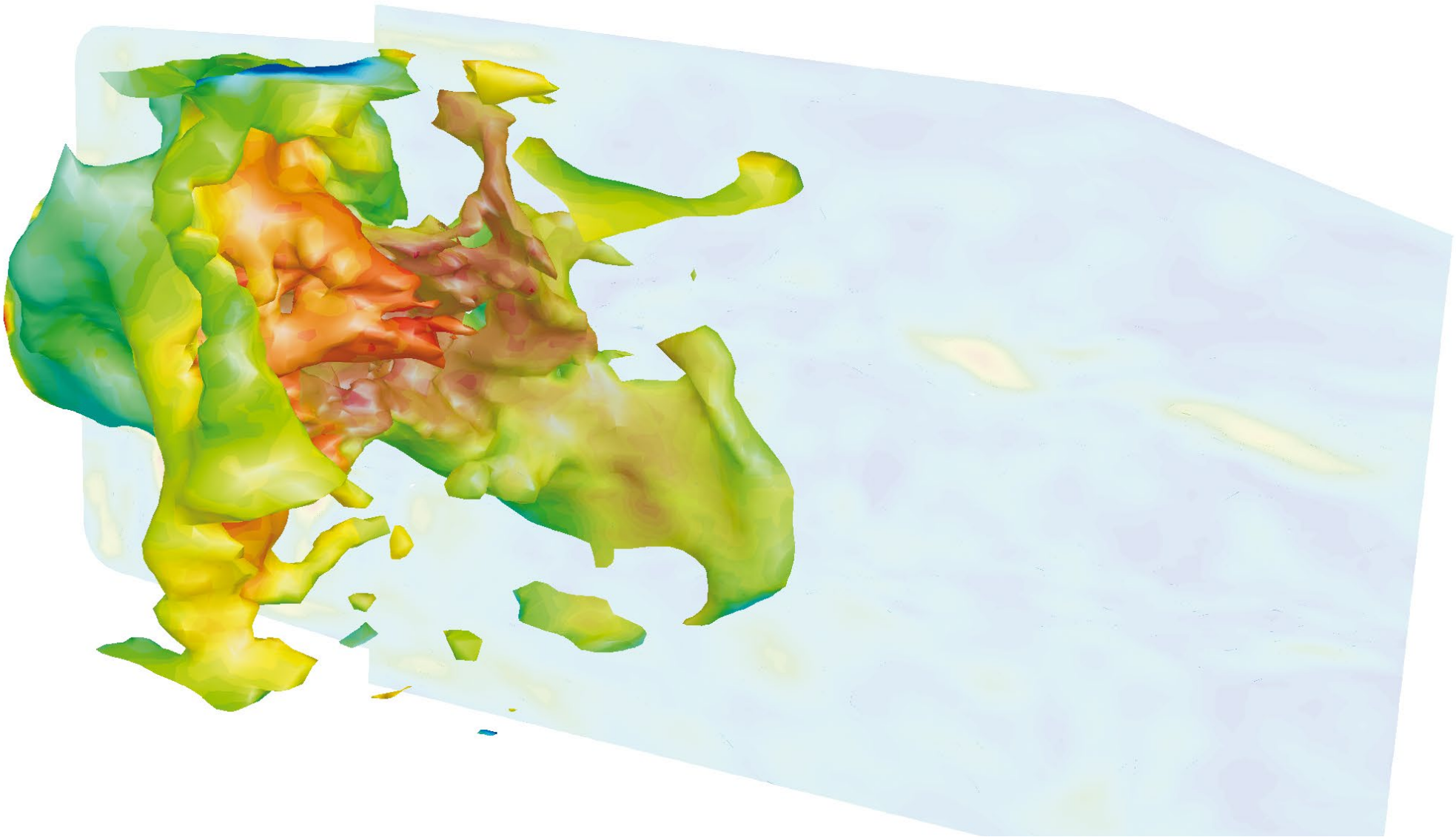
## Turbulence Simulation Challenges

Most engineering flows are turbulent, and these flows are inherently multiscale, three-dimensional and unsteady. Fluid flow applications involving complicated geometries and complex physics present the greatest challenges from a turbulence-modeling standpoint, particularly in the industries listed below.

**Aerospace** - Accurate turbulence modeling is critical to major aerospace challenges such as optimizing lift/drag ratio of a wing to ensure that the plane will fly safely while consuming as little fuel as possible. Turbulence modeling is involved in many other aerospace design problems, such as designing the engine to minimize both external and cabin noise.



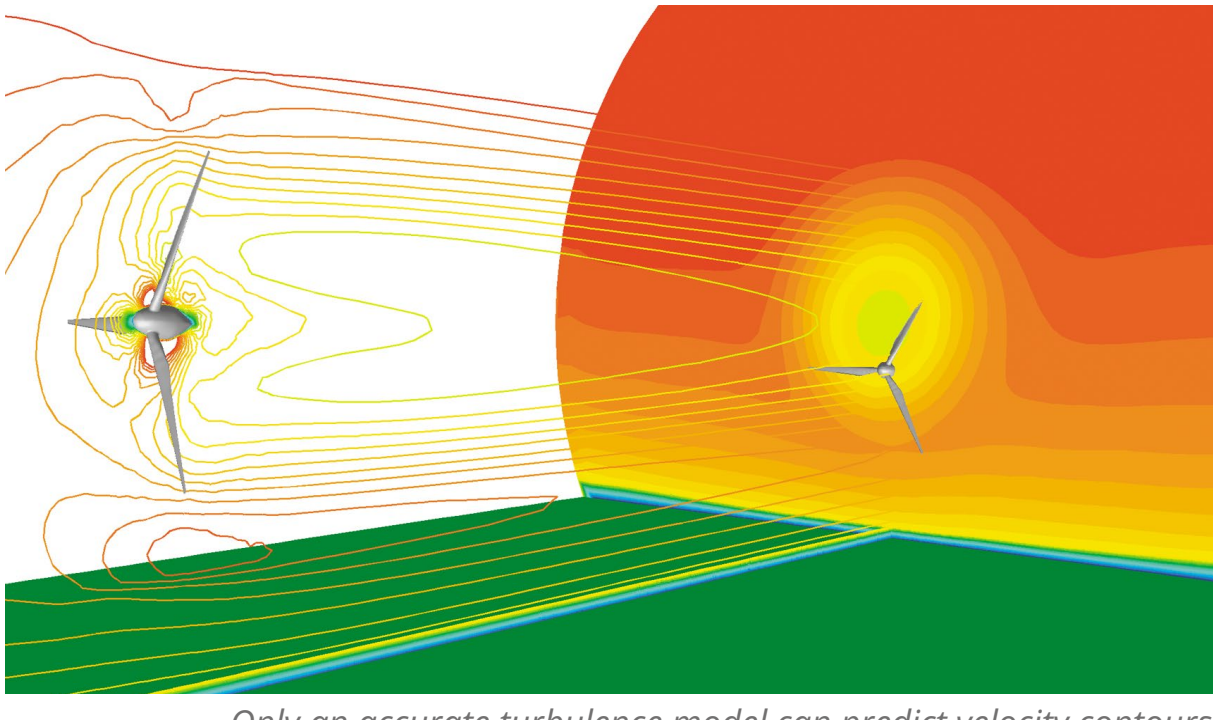
*Car manufacturers rely on accurate computational fluid dynamics of turbulent flows to compute the cooling and heating of a passenger car.*



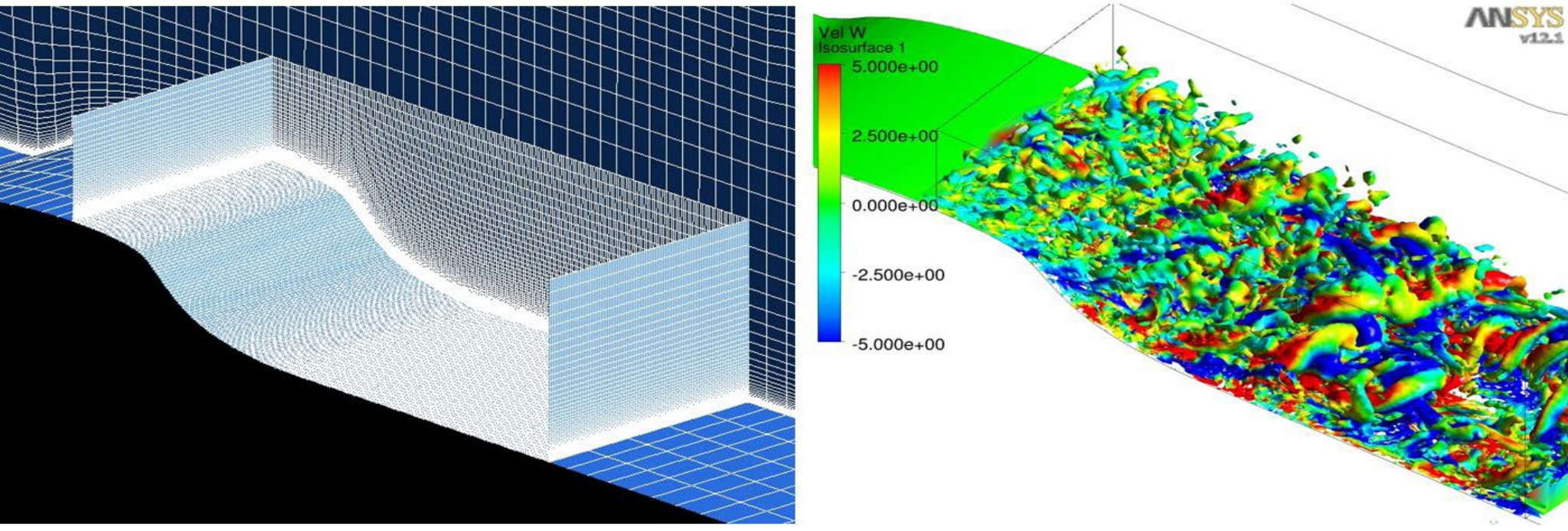
*In a number of turbulence applications, LES models are needed for accurate results. In this combustion chamber simulation, accurate simulation of the flame location, heat release and pollutant emissions requires resolution of flame-front wrinkling by the larger turbulent scales. This is possible using the LES turbulence model.*

**Oil and Gas** - Turbulence dissipates flow energy, so turbulence modeling is critical for companies involved in moving fluids over long distances. For example, it is important to accurately predict energy loss to determine the optimal distance between pumping stations. If the distance is too large, the flow moves too slowly or even stops flowing. If the distance is too small, the pumping system is overdesigned and underoptimized, leading to higher-than-necessary pump installation, operation and maintenance costs. The ability to accurately predict flow behavior and pressure drop induced by turbulence helps companies to design the most efficient and cost-effective oil and gas transport systems.

**Automotive** - Automotive aerodynamics is all about trade-offs, particularly striking the right balance between body style needs and aerodynamic concerns. Predicting the drag of alternative designs is the key to delivering an appealing design that is as fuel efficient as possible. Turbulence dictates a design's aerodynamic performance, so accurate prediction



*Only an accurate turbulence model can predict velocity contours behind a wind turbine. This is critical when studying wake-effect impact on a second down-wind turbine.*




*Software development experts at ANSYS recently progressed the state of the art with two advanced hybrid models: wall-modeled LES (WMLES) and embedded LES (ELES) turbulence models. WMLES allows the simulation of wall-bounded flows without the excessive computational costs associated with classical LES modeling. The ELES model offers even more flexibility by allowing an embedded LES zone within a larger RANS-simulated steady-state domain. This results in accurate simulation without excessive computational expense. This image shows turbulence structures created in the wake of a Volvo bluff body flame holder. Results were obtained using the detached delayed eddy simulation.*

of a design's drag requires accurate, tested and validated turbulence models.

**Chemical and Pharmaceutical** - Mixing is a crucial step in most chemical and pharmaceutical processes. The throughput of many process operations depends on achieving homogeneous mixing in as short a time as possible. Designers of mixing equipment utilize CFD to evaluate alternative agitator and tank designs and operating conditions to determine optimal configuration. Turbulence plays a critical role in mixing, so accurate turbulence modeling is essential to getting the design right the first time.

**Green Energies** - For power companies that operate wind turbines, the goal is to generate the maximum amount of power at the lowest cost from a given wind farm site. The challenge is that the wake of a turbine has significant effects — including reduced power output and shorter turbine life — for units located downwind. Wind farm developers are demanding more accurate methods for calculating these wakes as well as any terrain effects that impact a turbine's performance. By accurately predicting these phenomena, power companies can optimally place each wind turbine to produce maximum energy from a given land parcel.

## Turbulence Models for Complex Flows

Turbulence modeling makes it possible to account for turbulence effects using CFD at a reasonable computational cost. No single model or modeling approach can cover all types of turbulent flow, so different types of turbulence models have been developed in the past decades. For combustion, acoustics and other similar applications, some of the turbulent structure needs to be resolved to ensure accurate results.

**Steady-State Models** - Steady-state or Reynolds-averaged Navier–Stokes (RANS) models reduce the complexity of modeling turbulent flows by averaging the velocity field, pressure, density and temperature over time. The steady-state approach calculates mean flow quantities, and no attempt is made to resolve turbulent structures in time and space. Therefore, RANS methods are computationally frugal, which makes them eligible to reasonable computational resources. If selected and applied properly, these models offer engineers a highly attractive solution to predict the effect of turbulence without having to explicitly capture all scales involved in turbulent flows. Experience shows that RANS techniques are adequately accurate for the vast majority of industrial applications. However, for some of them, more advanced models capable of resolving a certain number of turbulence scales are required.

## Speedo: complex models for competitive hydrodynamics

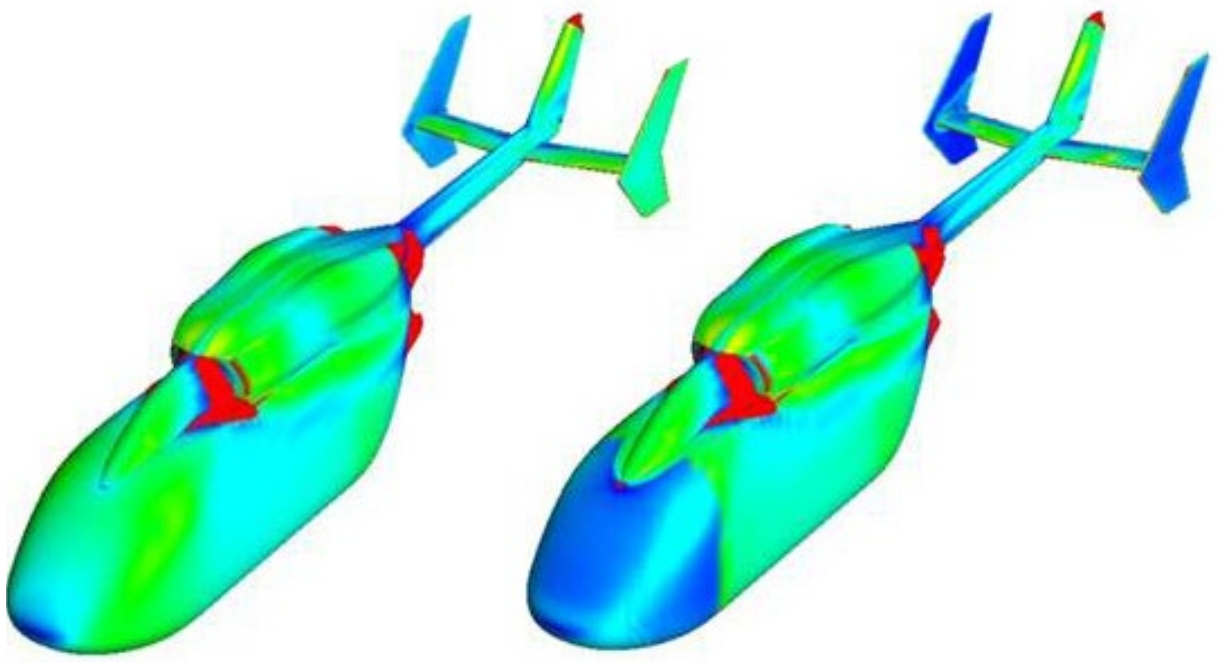
In Olympic swimming competitions, the difference between gold and silver can be mere milliseconds, so tiny improvements to multiple details can be critical. Legendary swimsuit manufacturer Speedo is using CFD to remove swimwear imperfections that increase drag while the athlete is swimming. Speedo Aqualab engineers studied three key elements of the gear: swimsuit, goggles and cap. The combination of flow around these elements and the athlete's body features creates an extremely complex and turbulent environment. Hence, very accurate turbulence models are needed to predict which design will create the least resistance in the water and give a competitive edge to the swimmer, even before the race starts.

For the most recent world-class races, the engineering team focused on improving the hydrodynamics of goggles, cap and suit. The innovative suit precisely fits the athlete's body and reduces drag. The cap reduces significant flow-field disruptions leading to improved performance. The goggle design blends with the athlete's facial features and results in minimum flow disruptions. Performance gains were real: for example, the goggle's CFD simulation- led design resulted in a performance improvement of 2 percent. When added to improvements that come from the suit and cap as well, this innovative swimwear system can enable an athlete to reduce race time by milliseconds, enough to turn a silver medal into gold. According to Dr. Tom Waller from Speedo Aqualab, "Engineering simulation has been absolutely critical in launching this world-first concept."

**Large-Eddy Simulation and Hybrid Models** - Large-eddy simulation (LES) turbulence models resolve the large turbulent structure in both time and space and simulate only the influence of the smallest, non-resolved turbulence structures. These models are usually one or two orders of magnitude more computationally expensive than the RANS approach, especially for complex industrial applications. As turbulent structures become very small in the near-wall region, the associated computational resources needed would make it too expensive and time consuming to use in the product development process. This poses severe challenges for LES simulations of applications in which wall effects impact product/design

performance. The solution to this problem is hybrid models, such as detached-eddy simulation (DES) and scale-adaptive simulation (SAS) models that combine steady-state and LES treatments for the model's wall boundary layer and free shear portions, respectively.

**Transition models** - The flow around turbine blades, wings and many other applications often features upstream laminar boundary layers that transition into a turbulent flow further downstream. Capturing this phenomenon is key to predicting wing lift or compressor performance, for example. To aid in analyzing this widely observed physical behavior, a new class of models predict behavior of a flow over a surface that starts in the laminar regime and transitions to the turbulent regime. As such, turbulence transition models significantly expand the range of applications for CFD software that include them.



*Drag coefficient prediction on a helicopter. Results (left) using a fully turbulent RANS model overpredict actual drag of the helicopter body; (right) using a transition turbulent model better predicts actual drag.*

*Accurate turbulence modeling is critical to designing high-performance racing sailboats.*

CARRI Systems

Technology architect since 1992

intel®

# Discover our **XLR** solutions **with PCI Express acceleration cards.**

The extreme computing servers from CARRI Systems are now compatible with Intel® Xeon® Phi™ 5110P coprocessor.

# HighServer XLR4i

**Power your breakthrough innovations with the highly parallel processing of the Intel® Xeon Phi™ coprocessor. We've packed over a teraFLOPS of double-precision peak performance into every chip—the highest parallel performance per watt of any Intel® Xeon® processor.**

intel® inside™ XEON PHI™

Intel® Xeon® Phi™ Coprocessor


## HighServer XLR4i
**133548**

Intel® Xeon® E5-2620 2 Ghz
32 Go DDR3 1600 MHz ECC REG
2 x 1 TB RAID Edition (7 200 rpm, 64 MB cache)
1 x Intel® Xeon® Phi™ 5110P with 8 GB GDDR5 @ 5.0 GT/s
Integrated video chipset
2 x Gigabit Ethernet
2U rack format
2 x redundant power supply - 1620 W
Linux Centos®
3 years on-site warranty (excluding accessories)
Metropolitan France only

## Starting at 5 990 euros HT

**For more information about XLR solutions**

## www.carri.com

intel Technology Provider Platinum 2013

intel inside™ XEON®

intel inside™ XEON PHI™

intel inside™ SOLID-STATE DRIVE


/success-stories

# THE BLOODHOUND SSC PROJECT: A 100% CFD DESIGNED ROCKET CAR

**An interview with Dr Ben Evans, Lecturer in Aerospace Engineering at Swansea University, about the HPC challenges undertaken in the increasingly famous Bloodhound SSC (SuperSonic Car) project and how CFD has been applied throughout in the vehicle design process.**

***HighPerformanceComputing: Can you tell us about the origin of the project?***

Dr Ben Evans: The Bloodhound project is not only making a car capable of going at 1,000 mph, it was also designed as an educational project. We are using the project as a hook to inspire young people across the UK to consider engineering and science careers. We hope to raise the profile of engineering in the UK and promote British engineering overseas.

***How did you get involved in the Bloodhound Project?***

Towards the end of my PhD in Computational Fluid Dynamics at Swansea University, Richard Noble, Bloodhound's project Director, approached Swansea University to see if we would be interested in taking on the challenge of the CFD modelling for the aerodynamic design of the vehicle. The University had previously worked with him on the Thrust SSC project in the 1990s, which was a great success. Professor Ken Morgan recommended me and I have been working on the Bloodhound aerodynamics as part of the design team, based at Swansea University, ever since.

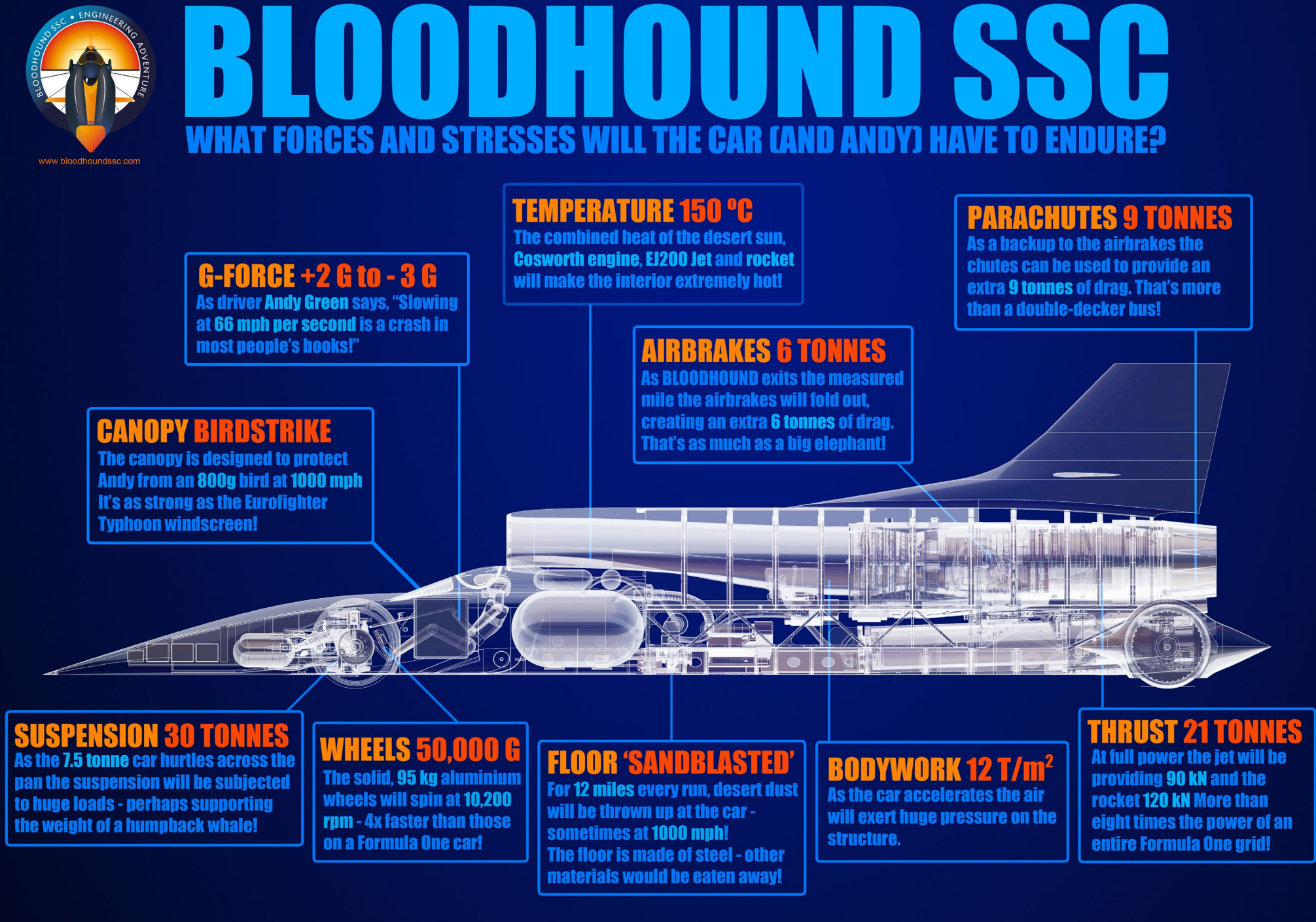
***What kind of challenges did you face during the design phase?***

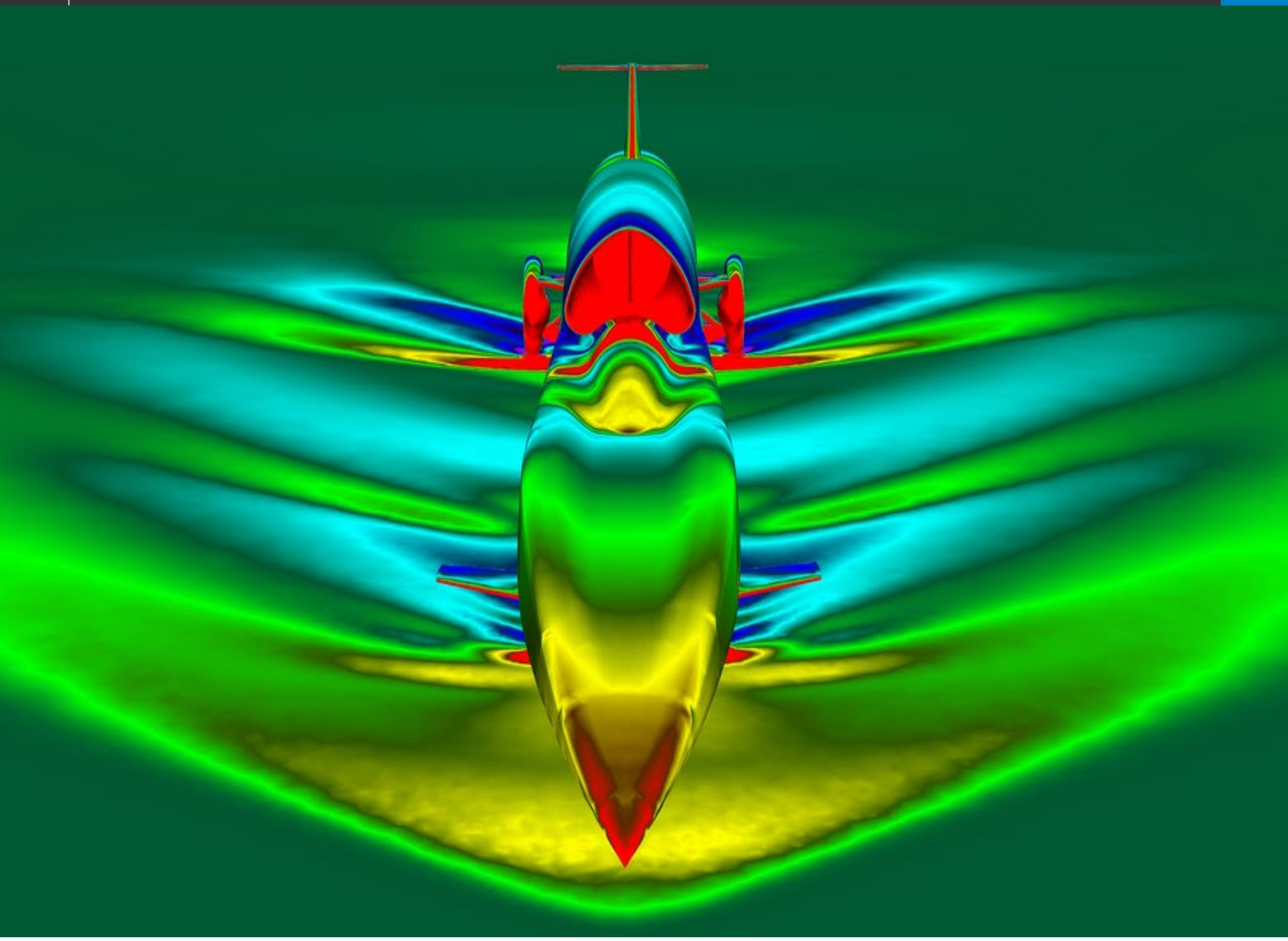
From an engineering perspective, the challenge is quite simple. We want to break the current Land Speed Record (LSR), which stands at 763 mph, by de-


signing the fastest car on Earth, and take it to 1,000 mph. For me, from a design perspective as an aerodynamicist, there are two simple issues to address: Can we keep the car on the ground, and can we minimize its drag sufficiently to allow it to achieve 1,000 mph?

***How many design iterations have been produced?***

We have run through approximately 12 full vehicle design iterations, with many sub-assembly iterations in between as you can imagine. The initial concept for the vehicle is actually fairly different from what Bloodhound is today. We have now reached the end of the design evolution phase and have frozen the "config 12" car. It's the one that is currently being built and will begin testing later this year at the Hakskeen Pan in South Africa.





***How important has CFD been in the design process?***

CFD is critical in the design process. The aerodynamics of an LSR vehicle like this is arguably the most difficult thing to get right. The external shape of the vehicle drives its aerodynamic performance and CFD is the only realistic tool available to help us predict the car's aerodynamic characteristics. Wind tunnels simply couldn't model the complex supersonic flow with rotating wheels and rolling ground that are a key feature of Bloodhound's aerodynamics, not to mention the fact that supersonic tunnels are incredibly expensive.

***What kind of HPC architecture do you use to run your simulations?***

We run our simulations on large PC clusters. A typical simulation takes up about 128 cores and 24 hours. The computational resource for this work is being provided by HPC Wales.

***Is there anything that CFD has highlighted in the design process that no one could think about?***

Yes, the shock wave structure at the rear of the vehicle has been a real challenge that we have had to overcome. Without the level of understanding that CFD has brought, we certainly would not have foreseen that problem. A big part of the CFD study has been to minimize the lift generated by this rear wheel shock system.

***Did you develop your own CFD software?***

Yes, we made developments to our in-house FLITE CFD system to stabilize it for the high Mach number flow in combination with the moving surfaces (wheels and rolling ground). We also developed a novel high speed particle entrainment model which we hope to validate during the testing of Bloodhound.


***What about the uses of CFD in the other components and sub-systems of the car (e.g. the rocket motor)?***

CFD has been used to design the intake duct for the EJ200 jet engine, the auxiliary flow that feeds into the body to feed the auxiliary power unit and cool the systems, and also in modelling the jet and rocket exhausts.

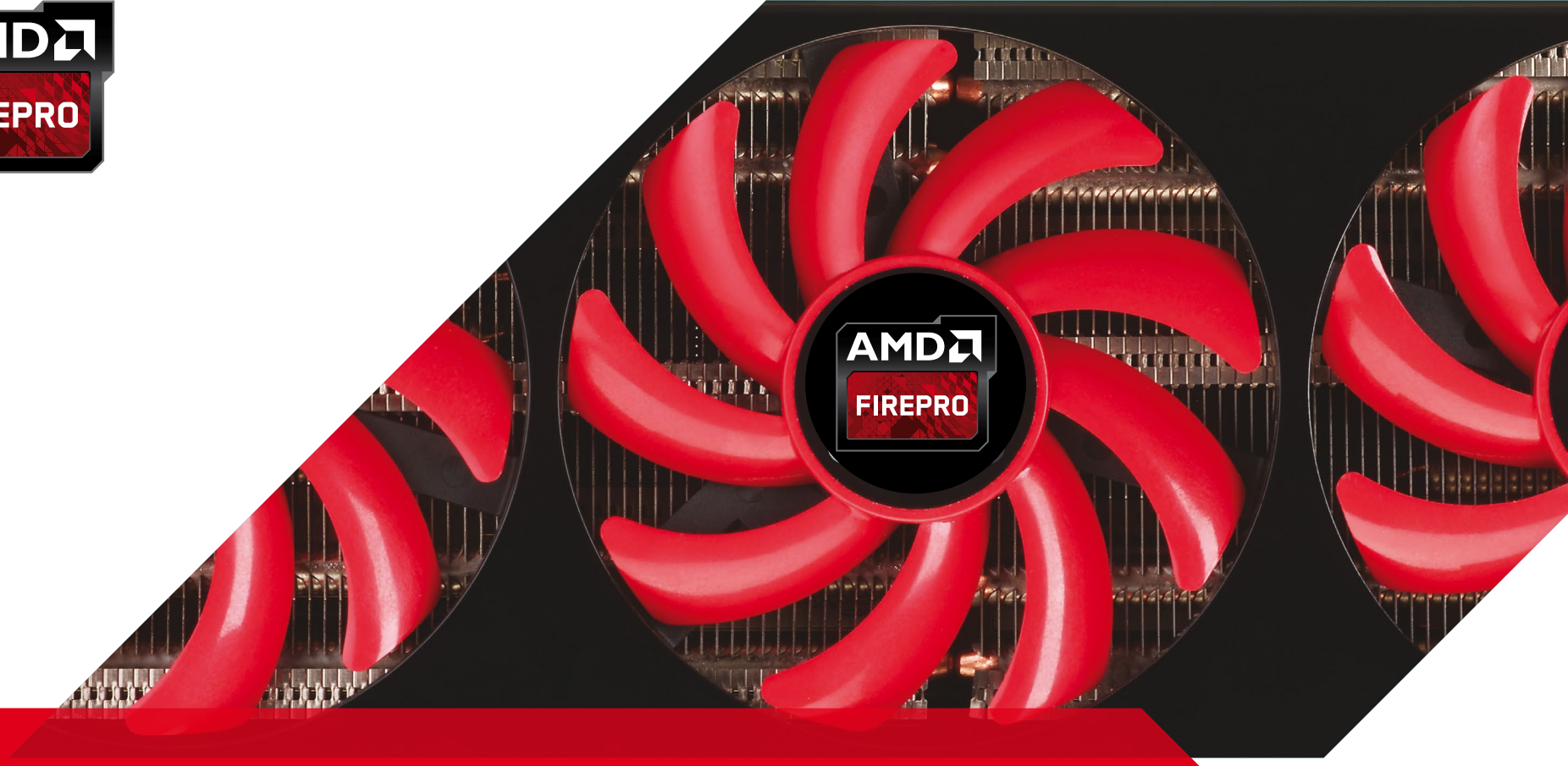
***How did you collaborate and interact with other calculation tools (e.g. structural or vehicle dynamics software)?***

Communicating with other engineering disciplines has been a challenge. For example, when working with the structural engineers on flutter predictions, we had to agree on file formats that allowed us to pass information between us and them to do the multidisciplinary work required to predict flutter.

***When will the World Land Speed Record of 1,000 mph take place and where? Will you be there?***

We will start low speed testing (up to 250 mph) in early 2014 on a runway in the UK. This will mainly be to check the vehicle's systems are all operating properly. To be honest, the aerodynamics doesn't really start getting interesting until higher speeds. We hope to be at the Hakskeen Pan in South Africa by the second half of 2014. My job there will be to compare the actual vehicle's aerodynamic performance with the CFD predictions. This analysis will feed in to the decision about whether it is safe to continue testing at higher speeds as we push towards a new LSR. ■

AMD FIREPRO




# Be locked or be free

# OpenCL

AMD FirePro™ S-Series solutions are the very best in open-standard GPU acceleration with OpenCL™ 1.2. Programmers can maximize their investment when developing code by targeting multi-core CPUs, the lastest APUs and discrete GPUs, freeing themselves from proprietary technologies.

## Find out more @ www.amd.com

/hpc_labs

# DISCOVERING OpenACC 2.0 – PART II

# THE NEW COMPUTE OPTIMIZATION FEATURES

**After a first release that proved essential to move parallel computations to accelerators in a standardized way, OpenACC 2.0 was long due. Now that it's here, this series of articles proposes to help you make the most of its new capabilities. Last month, we focused on the new data management features. This month, let's take a practical look at new ways to improve the management of compute regions.**

STEPHANE CHAUVEAU, PH.D.

The OpenACC 2.0 specification features a new section about atomic directives. Developers familiar with OpenMP may have a sense of déjà-vu while reading it: OpenACC atomics management is almost an exact copy of the OpenMP specification with only of a few minor changes such as replacing the OMP directive sentinel by the ACC sentinel. This is of course intentional. There was no practical reason to come up with a new syntax for a feature with almost identical requirements. As a consequence, developers will be able to easily migrate codes using atomics from one standard to the other.

## Going atomic

The problem of atomics is simple to understand but can lead to very annoying bugs. Imagine an OpenMP or OpenACC application with at least 2 threads (or gangs or workers) that may both attempt to increment a single global counter **X**. In prac-

tice, this task will be broken down into three distinct actions: read **X** from memory into a register, increment the register and write the register back to **X**. Within a given thread, the order of these actions is perfectly defined, but with multiple threads, it's a completely different story. The pseudo-code in **Listing 1** illustrates a possible behavior where the incrementation of **X** by two threads produces a final **X** that is incremented only once.

The situation is even more chaotic in real life as the memory caches found on all modern GPUs and CPUs are adding another level of complexity. Fortunately, all modern accelerators also provide mechanisms to implement these actions in a way that appears to be atomic regarding the memory accesses. In OpenACC 2.0, incrementing a scalar variable atomically is as simple as inserting an **atomic update** pragma before the increment statement, as illustrated in **Listing 2**. Most of the basic arithmetic operations on the native scalar types of each language are officially supported but some of them may not be available on all targets since implementing proper atomic operations usually require some kind of hardware support.

**Listing 1 - Example of 2-thread non-atomic increment with race condition.**

```
[thread1]  reg1 = X
[thread2]  reg2 = X
[thread1]  reg1 = reg1+1
[thread2]  reg2 = reg2+1
[thread1]  X = reg1
[thread2]  X = reg2
```

**Listing 2 - Simple atomic increment with OpenACC 2.0.**

```
#pragma acc atomic update
x = x + 1;
```

**Listing 3 - Poor implementation of a sparse vector packing function.**

```
#define NB 1000000
int pack( const float X[NB] , float Y[1000] )
{
  int i ;
  n = 0 ;
  #pragma acc kernels copy(n) copyin(X[0:NB]) copyout(Y[0:1000])
  {
    #pragma acc loop independent
    for (i=0;i<NB;++) {
      if ( X[i] != 0.0f ) {
        if ( n < 1000 ) {
          Y[n] = X[i] ;
        }
        /*  bug: The n values above may actually be different
         *  than the ones below.
         */
        #pragma acc atomic update
        n = n + 1 ;
      }
    }
  }
  return n ;
}
```

The **atomic update** directive is perfectly suited to count for instance the number of errors or matches in the entire execution of an OpenACC compute construct. However, it should not be used when the thread performing the update has to know the updated value. To understand why, let's just consider a large sparse vector and a function to compact its non-zero values into a vector of at most 1,000 elements. The naive approach shown in **Listing 3** is incorrect because all occurrences of the **n** variable found outside the atomic operation are subject to race conditions with other threads. In other words, any of these **n** may have a different value within the same loop iteration in this implementation.

This issue is solved by the **atomic capture** variant which provides an **atomic update** of a variable but also a copy of the original or of the resulting value into another variable. The proper version of our sparse vector packing function using an **atomic update** is given in **Listing 4**.

**Listing 4 - Corrected packing function using an atomic capture.**

```
#define NB 1000000
int pack( const float X[NB] , float Y[1000] )
{
 int i ;
 n = 0 ;
 #pragma acc kernels copy(n) copyin(X[0:NB]) copyout(Y[0:1000])
 {
  #pragma acc loop independent
  for (i=0;i<NB;++) {
   if ( X[i] != 0.0f ) {
    int oldn ;
    #pragma acc atomic capture
    {
    oldn = n ;
    n = n + 1 ;
    }
    if ( oldn < 1000 ) {
     Y[oldn] = X[i] ;
    }
   }
  }
 }
 return n ;
}
```

## Efficiently targeting device types

A few years ago, when the OpenACC working group started to develop the specification, the promise was to write once and execute efficiently everywhere. If the portability issue is more or less solved today, efficient execution on multiple platforms from the same sources is often just not possible because of the architectural differences between the accelerators currently found in the market. In practice, the number of gangs, the number of workers and the vector length of a **parallel** construct usually need to be manually fine-tuned for each target accelerator in order to get consistent performance. The impact of using an improper value if often dramatic, and that is especially true in environments using both Xeon Phi's and GPUs from NVIDIA or AMD. Compilers will of course attempt to implement heuristics so as to choose the best values according to the target but the result will rarely be optimal.

A partial solution to this problem is the new **device_type** or **dtype** clause, which provides a mechanism to restrict some clauses to a specific target. The **device_type** clause needs a keyword to specify the target; its role is to make further clauses specific to that target. The effect **device_type** stops when another **device_type** clause is found or at the end of the directive. Note also that the special target name ***** acts as a default that is used when no other **device_type** clause is matched. Simply speaking, **device_type** is like a C **switch** that would be applied to clauses within a directive. **Listing 5** shows how to use **device_type** to customize the number of gangs and workers in a loop.

**Listing 5 - Typical use of the device_type clause.**

```
#pragma acc loop gangs workers \
  dtype(NVIDIA)  num_gangs(250) num_workers(128) \
  dtype(NVIDIA2) num_gangs(130) num_workers(256) \
  dtype(RADEON)  num_gangs(200) num_workers(64) \
  dtype(XEONPHI) num_gangs(53)   num_workers(8) \
  dtype(*) num_gangs(100) num_workers(32)

for (i=0;i<n;i++) {
  do_some_work(i) ;
}
```

The OpenACC specification does not enforce specific names

for the different target architectures available today. The **NVIDIA**, **RADEON** and **XEON-PHI** "keywords" used in **Listing 5** are the recommended names for the three major kinds of accelerators but compilers are supposed to provide mechanisms to create other keywords (that would typically be a command line option). In **Listing 5**, it is also assumed that **NVIDIA2** is a user defined keyword for a specific NVIDIA GPU. Unknown target keywords are simply ignored by the OpenACC compiler.

The **device_type** clause is only allowed on a few directives, namely **routine**, **kernels**, **parallel**, **loop**, **update** and all their combinations. Be also aware that a certain number of clauses are not allowed after a **device_type**. A quick look at your directives' documentation will give you a detailed list of the possibilities you have.

**Listing 6 - Typical OpenACC external routine definition in C.**

```
// ---- in file sum.c

#pragma acc routine worker
void sum(int n , float *A)
{
  int i ;
  float s = 0.0f ;
  #pragma acc worker reduction(+:s)
  for (i=0;i<n;i++) {
    s = s + A[i] ;
  }
  return s ;
}

//--- in file sum.h

#pragma acc routine worker
void sum( int n , float *A ) ;

//--- in file main.c

float X[100][200] , Y[100] ;
//...
#pragma acc gang
for( j=0; j<100; j++ ) {
  Y[j] = sum( 200 , X[j] ) ;
}
//...
```

**Listing 7 - Prototype declaration of an external native function.**

```
//--- in file sum.h

#pragma acc routine worker \
  dtype(NVIDIA) bind("my_cuda_sum") \
  dtype(RADEON) bind("my_opencl_sum")
void sum(int n , float *A) ;
```

## User-defined calls

In version 1.1, OpenACC was not very explicit about function calls. Calling intrinsic mathematical routines was inherently possible but calling user-defined functions was not really part of the specification. This led compiler vendors to accept calls to user-defined functions as an extension to the standard. OpenACC 2.0's **routine** directive is an attempt at standardizing calls to user-defined functions on the device. At first glance, the problem may seem simple, but considering the three levels of parallelism in the OpenACC execution model, it is not. For instance, a function that contains a gang loop should never be called from inside another gang loop, and similar issues exist at the worker and vector levels.

Accordingly, the new **routine** directive must be specified for each user-defined function called from within an accelerated region. In C and C++, the directive will typically appear before the function; in Fortran, it will be placed inside the specification part of the function or subroutine.

The **routine** directive must also contain a **gang**, **worker**, **vector** or **seq** clause to specify the context in which it may be called

and the maximal worksharing level allowed in its body. For example, a routine with a **worker** clause can contain **worker**, **vector** and **seq** loops but no **gang** loops. Similarly, this routine may not be called from within the body of a **worker** or **vector** loop. The proper syntax is illustrated in **Listing 6**. Routines that do not contain any kind of loop worksharing must have a **seq** clause indicating that they can be called from anywhere.

## Calling CUDA or OpenCL functions

Native functions written in CUDA or OpenCL haven't been forgotten. To call them, a **bind** clause must be specified on the **routine** directive of the function declaration in order to specify the name of the corresponding native function. The **device_type** mechanism can then be used to select the right native function name for each target as illustrated in **Listing 7** with pseudo CUDA and RADEON versions of our sum function. Note that OpenACC 2.0 does not specify an exact mechanism to integrate external native code, so the mechanism may vary from one compiler to another.

An alternative form of the **bind** clause can be used to generate customized versions of a routine for different accelerator targets. The **bind** argument is not a string anymore; instead, you can use the name of an alternative C, C++ or Fortran routine that will be compiled in OpenACC mode. Any call to the original function inside an accelerated region is then replaced by a call to the specified alternative function. As for native routines, the proper **bind** clause can be selected using the **device_type** mechanism. All these mechanisms are illustrated in **Listing 8**.

**Listing 8 - Routine customization for multiple hardware targets.**

```
//--- File sum_nvidia.c will only be used when targeting NVIDIA
accelerators

#pragma acc routine worker
void sum_for_nvidia(int n , float *A)
{
  // here comes the ACC code tuned for NVIDIA targets
}

//--- File sum_radeon.c will only be used when targeting Radeon
accelerators

#pragma acc routine worker
void sum_for_radeon(int n , float *A)
{
  // here comes the ACC code tuned for Radeon targets
}

//--- in file sum.h

#pragma acc routine worker \
  dtype(NVIDIA) bind(sum_for_nvidia) \
  dtype(RADEON) bind(sum_for_radeon)
void sum(int n , float *A) ;
```

## When tiling becomes standard

The OpenACC committee has also been working on some high level loop tuning features that may improve code efficiency in some circumstances. In OpenACC 2.0, that work resulted in the **tile** clause of the **loop** directive, which can be used to decompose a loop nest into tiles of a fixed size. This is a classical code transformation whose purpose is to improve the locality of memory accesses inside the tiles and make the use of memory caches more efficient.

**Listing 9** shows a typical use case of the **tile** clause: the bi-dimensional convolution. The computation performed at each location is accessing several of the neighbor location in both directions. The effect of the **tile** clause is further illustrated in **Listing 10** where the code from **Listing 9** is translated into a "no-**tile**" version. Each of the loops affected by the tiling is broken down into two loops: an outer loop that iterates over the tiles and an inner loop that iterates inside each tile. The outer and inner loops are then collapsed using a **collapse** clause before apply-

ing the **gang** clause to the outer loops and the **worker** clause to the inner loop. As a result, cache re-use between workers of the same gang should be improved.

Needless to say, the code in **Listing 10** is over-simplified. Real codes generated by the compiler would have to account for loops with a non-constant number of iterations that may not even be exact multiples of the tile sizes. This is typically not something a normal user would want to write manually. Also, the **tile** clause is applicable to more than two loops and may use any of the gang, worker and vector levels of parallelization provided by OpenACC.

**Listing 9 - Typical example of the tile clause applied to a 2D loop nest.**

```
#pragma acc loop gang worker tile(8,16) num_workers(64)
for (i=0 ; i<1024; i++)
  for (j=0 ; j<512; j++) {
     X[i][j] =
       c11 * Y [i-1] [j-1] + c12 * Y [i-1] [j-1] + c13 * Y [i-1] [j-1] +
       c21 * Y [i] [j-1] + c22 * Y [i] [j] + c23 * Y [i] [j-1] +
       c31 * Y [i+1] [j-1] + c32 * Y [i-1] [j-1]  + c33 * Y [i+1] [j-1] ;
   }
```

**Listing 10 - A possible "no-tile" translation of the code in Listing 9.**

```
#pragma acc loop gang collapse(2)
for (ii=0 ; ii<1024; ii+=16)
  for (jj=0 ; ji<512; jj+=8)
   #pragma acc loop worker num_workers(64) collapse(2)
    for (i=ii ; i<ii+16; i++)
      for (j=jj ; j<jj+8; j++) {
        X[i][j] =
          c11 * Y [i-1] [j-1] + c12 * Y [i-1] [j-1]  + c13 * Y [i-1] [j-1] +
          c21 * Y [i] [j-1] + c22 * Y [i] [j]  + c23 * Y [i] [j-1] +
          c31 * Y [i+1] [j-1] + c32 * Y [i-1] [j-1]  + c33 * Y [i+1] [j-1] ;
    }
```

As we have seen, the OpenACC 2.0 specification provides several new features meant to improve the efficiency and portability of your codes. But with the recent release of OpenMP 4, which also brings accelerator support, we are often asked whether both standards will ultimately join and, if not, which one will ultimately "win". This is a difficult question. Whereas OpenACC started as an attempt to speedup OpenMP's development process, both specifications are now significantly different and can hardly be re-united. OpenACC has the advantage of being more mature while OpenMP is clearly recognized and supported by more players in the HPC market. The OpenMP model for accelerators is also more flexible but that is not necessarily an advantage because flexibility usually comes with a significant cost in term of raw performance. That is why the CUDA model with all its constraints was so successful over the last few years. Eventually, the success of both models will probably depend on the ability of accelerator vendors to find the right balance between performance and flexibility.

Happy programming! ■

/hpc_labs

# AN INTRODUCTION TO PERFORMANCE PROGRAMMING [PART I]

**While it is always difficult to optimize existing codes, an appropriate methodology can provide provable and replicable results. In this first article in a series of two, we'll begin by evaluating and validating sources before addressing the critical issue of the efficient use of the compiler. Next month, we'll focus on the organization of data, the algorithmic implementation and the optimization of parallelization.**

ROMAIN DOLBEAU*

The subject of performance programming is a complex one, about which many scholarly papers have been published. Deriving from classic introductions such as Chellappa et al. [1], this article aims at providing an experience-based approach to the newcomers. Its target audience includes all those interested in writing high-performing applications, whether their background includes computer science or not. On the basis of existing codes, we will detail a path leading to performance improvements with minimal effort and in a minimal amount of time. We'll also cover good practices to ensure the correctness and efficiency of the work done.

Performance is a difficult thing to obtain in programming. Not just because writing fast code is complex, but also because understanding why the code isn't fast in the first place is often far from obvious. When tasked with improving the performance of a code, develop-

* HPC Fellow, CAPS Entreprise

ers find themselves with a well-known multi-step agenda that has to be iterated upon:

1) Identify the sections of the code that take up most of the time;

2) Analyze why these sections are so costly;

3) Improve their performance by targeting bottlenecks.

But this conventional methodology, based on Amdalh's law [2], is only a very succinct explanation - or a high-level view - of the actual processes involved. It does not express the low-level concerns of the everyday programmer. For performance isn't just about the code; it is about the entire sequence of decisions and elements that leads from the code to the actual computations, i.e.

1) The chosen algorithms;

2) The specific implementation of these algorithms in the chosen programming language;

3) The organization of the data on which the algorithms will work;

4) The choice of an optimizing compiler to convert the code from source to binary;

5) The third-party libraries on which the code relies;

6) The actual hardware on which the code will run.

Each of these decisions and elements must be taken into account to achieve the goal of optimized performance; ignoring any one can lead to a significant waste of efficiency. And while the process of creating and running codes is sequential through or within each elementary step, it is the entire sequence that has to be targeted at once. Optimizing one step for a given set of states of the other steps does not mean it will remain optimal once these other steps have changed.

This paper suggests a very pragmatic approach built on years of experience in optimizing code - an approach that tries to maximize the efficiency of the programmer's time. Rather than rewriting the code straight away, a better idea is to start by leveraging all the tools available to maximize its efficiency, and then only to consider modifying it. In this respect, the important points this paper will try to make are the following:

## References

[1] S. Chellappa, F. Franchetti, and M. Püschel, "How to write fast numerical code: A small introduction," in *Summer School on Generative and Transformational Techniques in Software Engineering (GTTSE)*, ser. Lecture Notes in Computer Science, vol. 5235. Springer, 2008, pp. 196–259.

[2] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*, ser. AFIPS '67 (Spring). New York, NY, USA: ACM, 1967, pp. 483–485. [Online]. http://doi.acm.org/10.1145/1465482.1465560

[3] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002.

[4] J. L. Gustafson, "Reevaluating amdahl's law," *Commun. ACM*, vol. 31, no. 5, pp. 532–533, May 1988. [Online]. Available: http://doi.acm.org/10.1145/42411.42415

[5] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. San Mateo, CA: Morgan Kaufmann, 1990.

[6] D. Barthou, G. Grosdidier, M. Kruse, O. Pène, and C. Tadonki, "QIRAL: A High Level Language for Lattice QCD Code generation," in *CoRR*, vol. abs/1208.4035, 2012.

[7] A. J. C. Bik, *Software Vectorization Handbook, The: Applying Intel Multimedia Extensions for Maximum Performance*. Intel Press, 2004.

[8] Intel Corporation, *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2* (2A, 2B & 2C): Instruction Set Reference, A-Z, March 2013, no. 325383-046.

**Reproducibility -** We want to be able to reproduce both the results of the execution of the code (fast but wrong is not a good thing), and the measured performance. This is an issue for programs whose performance is data-sensitive, such as convergence algorithms, as several data sets will have to be validated.

**Maintainability -** Even if ultimate performance is the objective, the code will have to be maintained in one way or another. As obvious as it sounds, from two implementations with similar performance, the easier to explain and maintain is usually the better choice.

**Prioritization -** Time should be invested where the best results are likely to be obtained. Manually fine-tuning a kernel is fun but very often it is also an inefficient use of time. Once a code section is fast enough for its purpose, there is no point in making it faster.

**Performance -** With everything else above in mind, performance becomes a much more manageable goal.

## I - EVALUATION AND VALIDATION

### I.A - Test cases

The single most important thing when working on a code is to maintain the validity of the results. The second most important thing is to reliably quantify the gain (or loss) of performance of the modified, validated version.

Validating results is always a difficult task. While some codes will have the good property of producing output files whose content depends solely on the input parameters, it is not always the case. Any code involving IEEE754-2008 floating-point operations will suffer from rounding approximations, unless compiled with extremely strict conformance options that prevent any performance gain. Any code involving random numbers (such as Monte Carlo methods) will produce unreproducible results. And not all algorithms are equal with regards to numerical accuracy, a subject covered in details by Higham [3].

Accordingly, the first step is to establish a test case (or a set of test cases) that will have several good properties for validating the results:

1) An execution time short enough that it can be tested on a regular basis;

2) An execution time long enough that small mistakes are likely to become noticeable in the output, rather than being masked by numerical approximations;

3) A good coverage of the significant portions of the computations, both in terms of amount of code and type of input data (for instance, when propagating energy in a discretized space, the first few steps are usually full of non-significant zeroes, while boundary conditions are not reached and thus not tested until many iterations have been run);

4) A clear validating procedure to ensure the output is correct.

These aspects should be discussed with the people interested in running the code for production. The second and third point, in particular, are not always easy to achieve. For codes involving random numbers, test cases should be derandomized, for instance by fixing a seed or by precomputing one or more set of values, as it is usually the easiest way to ensure reproducibility.

The second step is to establish another test case (or another set of test cases), whose purpose will be to check performance. This test case will usually be larger (and longer running) than the validating case. It should also come with a clear validating procedure, in order to trap any residual mistake that might not be noticeable with the smaller test case. And it should be reasonably representative of typical production inputs. This can be very hard to achieve for extremely long and/or extremely big codes. Unfortunately, there is no shortcut here, as representativeness is highly dependent on the type of code.

Finally, never forget that it does not make much sense to try and improve a code with remaining issues. Verifying that the code doesn't misbehave is therefore an important preliminary step. Compilers can optionally do runtime bounds checking. Tools like valgrind will check for accesses that haven't been initialized or that reach beyond properly allocated memory.

Any such issue in the code must be dealt with prior to any kind of optimization as it is bound, sooner or later, to result in unreliable behaviors.

### I.B - Measuring performance

While measuring performance seems at first glance quite simple, it is actually a fairly complex subject as well. What should you measure? And how should you measure it? When you've come up with reasonable answers to these two questions, there remains the difficulty of ensuring the consistency of measurement from one run to another.

What to measure is logically the first decision make. Eventually, something akin to the **time** command is going to be used: when the code is run in production, it is the entire execution time from beginning to end that is going to matter to the consumers of the code. It's also going to matter to the programmer, as it is the time taken by any validation run or full-scale measurement. If this metric is obviously important, others are, too. After a profiling has been done (see section I.C below), the code will be seen as a sequence of stages: the prolog or initialization phase (usually an O(problem size)), one or more computational steps of varying complexity, and the epilog (also usually an O(problem size)). The costliest of these three steps will usually be the computational part, but it is not always the case. Prologs and epilogs generally consists of I/O and data movements, which are outside the scope of this article. If they dominate the execution time, either the code is not amenable to improvements, or the test case is too small for a significant measurement. Gustafson's law [4] helps us here, as for most codes there should be a problem size big enough that the computation part dominates the execution time. Gustafson's law is an important consideration: even if the test case used during the optimization phase doesn't offer an overall improvement (because of significant prolog and epilog times combined with Amdhal's law), larger production cases might since the fraction of time spent in the computational phase will be higher. It is therefore important to keep track not only of the overall execution time, but also of each main phases of the code. An X2 improvement in a computation phase representing 50% of the test case runtime is an overall X1.33 improvement of the whole code, but an X2 improvement in a computation phase representing 98% of the production case runtime is an overall X1.96 improvement.

How to measure it is the second step. The problem here is that time scales on computers can vary by orders of magnitudes. On a 2.5 GHz CPU, a single CPU cycle takes 0.4 ns, or $4 \times 10^{-10}$ s. Execution times that go beyond the hour take on the order of $10^{14}$ cycles, of which only the most significant digits make sense. Conversely, a small function of a few thousand cycles would take on the order of $10^{-7}$ s, a precision greater than most system calls

**Listing 1 - A trivial C version of the BLAS saxpy function.**

```
void saxpy(int n, double alpha, double *x, double *y) {
  int i;
  for (i = 0 ; i < n ; i++) {
    y[i] = alpha * x[i] + y[i];
  }
}
```

**Listing 2 - Assembly code of the saxpy function in Listing 1.**

```
..___tag_value_saxpy.1:
        xorl      %eax, %eax
        movslq    %edi, %rdi
        testq     %rdi, %rdi
        jle       ..B1.5
..B1.3:
        movsd     (%rsi,%rax,8), %xmm1 #\label{inst:load}#
        mulsd     %xmm0, %xmm1 #\label{inst:mul}#
        addsd     (%rdx,%rax,8), %xmm1 #\label{inst:add}#
        movsd     %xmm1, (%rdx,%rax,8) #\label{inst:store}#
        incq      %rax #\label{inst:incq}#
        cmpq      %rdi, %rax #\label{inst:cmpq}#
        jl        ..B1.3
..B1.5:
        ret
```

would ensure in practice. It is a good idea to keep these orders of magnitudes in mind when choosing the proper timer. For most human-scale measurements (tenths of seconds or more), the **gettimeofday** function is perfectly adequate. For hundreds of microseconds or less (millions of cycles or less) measurements, use **_rdtsc** (in x86-64 compilers) to get the current CPU cycle count. For time scales in between, things are less clear-cut: **gettimeofday** having a theoretical precision of a microsecond, it should be adequate. But since **_rdtsc** returns a 64 bits integer, measuring billions of cycles is also an option.

Consistency of measurement might be the most overlooked aspect of the problem. Codes seldom run on their own on a computer; they operate in an environment that includes the operating system, various housekeeping programs running permanently or regularly, and potentially other users. Whenever possible, reliable measurements should be made on an otherwise idle machine - one where very little is running on top of the operating system. Any other code will influence the results, by consuming memory bandwidth, inter-cpu bandwidth, cache space, I/0 bandwidth, or even by simply raising the power consumption of a CPU and affecting the thermal behavior of the others. That is not all; the operating system behavior can drastically affect the performance of the CPU. Modern machines have multiple cores, each of them with its private L1 cache (and often private L2 caches). Whether or not the operating system let the code run on the same core for the entire run will greatly affect performance, as each move from one core to another will force the code to reload the local caches. Moving from one socket to another has an even greater effect: not only the shared intra-socket cache (L3 on the Sandy Bridge architecture) will be reloaded, but NUMA effects will happen as well. To keep that under control, the Linux operating system exposes commands to pin or lock a process on a subset of cores, to specify which NUMA memory domains to use, and so forth (see for instance **numactl**). Understanding the exact topology of the machine is also advisable, using tools from e.g. the **hwloc** library. When a code has already been parallelized, many tools have the ability to automatically enforce the pinning: the **KMP_AFFINITY** environment variable in the Intel OpenMP **implementation**, the **GOMP_CPU_AFFINITY** environment variable in the GNU OpenMP **implementation**, the **-binding** option in the Intel MPI **library**, and so on. The golden rule here is that ensuring a consistent placement of processes and threads in the machine ensures reproducible measurements.

**Listing 3 - Assembly code of the saxpy function in Listing 1 modified with mulpd (simultaneous computation of two fp operations).**

```
..___tag_value_saxpy.1:
        xorl      %eax, %eax
        movslq    %edi, %rdi
        testq     %rdi, %rdi
        jle       ..B1.5
..B1.3:
        movupd    (%rsi,%rax,8), %xmm1
        mulpd     %xmm0, %xmm1
        addpd     (%rdx,%rax,8), %xmm1
        movupd    %xmm1, (%rdx,%rax,8)
        incq      %rax
        cmpq      %rdi, %rax
        jl        ..B1.3
..B1.5:
        ret        ret
```

**Listing 4 - Compiling saxpy with Intel's compiler using the -vec-report3 option.**

```
$ icc -O3 -vec-report3 -S trivial_saxpy.c
trivial_saxpy.c(4): (col. 3)
remark: LOOP WAS VECTORIZED.
```

### I.C - Profiling

Profiling is the process of evaluating the relative cost of each part of the code. Without proper profiling, there is no way to tell which part of the code should be improved. No matter how convincing is the argument that function XYZ is the most important in the code, only an objective assessment should

**Listing 5 - Compiling the original version of the "riemann" function in the Hydro application.**

```
HydroC/oaccHydroC_2DMPI/Src$ icc -std=c99 -O3 -vec-report2 -Wno-unknown-pragmas -S
riemann.c
riemann.c(51): (col. 3) remark: LOOP WAS VECTORIZED.
riemann.c(138): (col. 14) remark: loop was not vectorized: existence of vector dependence.
riemann.c(107): (col. 11) remark: loop was not vectorized: not inner loop.
riemann.c(102): (col. 7) remark: loop was not vectorized: not inner loop.
riemann.c(298): (col. 11) remark: loop was not vectorized: existence of vector dependence.
riemann.c(296): (col. 9) remark: loop was not vectorized: not inner loop.
riemann.c(291): (col. 7) remark: loop was not vectorized: not inner loop.
```

**Listing 7 - Same compilation as in Listing 5 using the #pragma ivdep directive on external loops.**

```
HydroC/oaccHydroC_2DMPI/Src$ icc -std=c99 -O3 -vec-report2 -Wno-unknown-pragmas -S
riemann.c
riemann.c(51): (col. 3) remark: LOOP WAS VECTORIZED.
riemann.c(109): (col. 11) remark: LOOP WAS VECTORIZED.
riemann.c(103): (col. 7) remark: loop was not vectorized: not inner loop.
riemann.c(300): (col. 11) remark: loop was not vectorized: existence of vector dependence.
riemann.c(298): (col. 9) remark: loop was not vectorized: not inner loop.
riemann.c(293): (col. 7) remark: loop was not vectorized: not inner loop.
```

be trusted. Intuition, past experience, alternative implementations and test on completely different hardware platforms do not represent the current status of the code on the currently available hardware.

There are two main classes of profiling: instrumentation and sampling. The first one, instrumentation, adds instructions inside the binary to dynamically measure the relative use of each function. While comparatively reliable for creating call trees (how each function calls each other) and call counts, it has the bad property of adding overhead to the code and potentially altering its behavior. Sampling takes an unmodified code, and checks its running at regular intervals to evaluate the relative importance of each part. While much less intrusive, it can sometimes miss short but important parts of the code, or have weird side-effect due to the sampling interval. Neither is strictly better than the other, so both types should be used. Tools like gprof, Intel VTune or callgrind are all useful, to the extent that all available tools should be used, not just one of them. Each has its strengths and weaknesses, and it is the combination of their results that gives a clear picture of the code's dynamic behavior.

In an ideal world, profiling should be done on nearly production-ready code, i.e. a highly optimized binary. But more often than not, optimizations obfuscate the code to the point that profiling results are not very helpful. In difficult contexts, optimizations (and inlining in particular) should be "dialed down" step by step, and the consistency of profiling results checked accordingly until meaningful numbers are obtained. For instance, if the original results indicate that the do_the_computations function does 95% of the work, then subsequent results with less inlining should show the same order of magnitude for the sum of the time periods spent in do_the_computations and its call tree (all the functions it calls, and all the functions the latter call, and so on). In case of inconsistent results, you must absolutely understand why, and identify which optimization changed the code's behavior so drastically that the profiling results change. It might become important later to understand how to improve performance.

Profiling shouldn't be viewed as a one-shot operation. Once the costlier parts have been identified and improved, profile your code all over again to get a new, up-to-date picture. Improvements in one part will make the other parts comparatively more important, but side-effect such as new cache behavior may alter their performance, for better of for worse.

## II - USING THE COMPILER

Most developers see the compiler as a necessary evil, except for those who have forsaken it completely (or so they think) for so-called interpreted languages. Obviously, these languages cannot be run directly on the hardware, and therefore a translation layer is still present. While it could be a virtual machine, for performance reason it is often a just-in-time compiler. Which, as its name implies, is still a compiler, but one that tries to be fast rather than efficient and on which the developer has very little control. For the vast majority of codes requiring all-out performance, the compiler will be offline and highly tweakable. The examples in the paragraphs below will be related to Intel's C/C++ and Fortran compilers, as they are probably the most ubiquitous in high performance computing, but most comments are applicable to other languages and other compilers as well.

**Listing 6 - Structure of the loops at lines 102, 107 and 138 of the "riemann" function in the Hydro application.**

```
for (int s = 0; s < slices; s++) { // line 102
  for (int i = 0; i < narray; i++) { // line 107
    // some code here
    for (iter = 0; iter < Hniter_riemann; iter++) { // line 138
      // some code here
    }
    // some code here
  }
}
```

### II.A - The hardware problem

To understand why compiling the programming language into machine code via an offline compiler is so important, we first have to mention a few things about hardware considerations (a good starting point on the subject is Hennessy-Patterson [5]). A computer CPU does not execute sophisticated, high-level code. It only executes very basic instructions, of which three classes are really important to us: memory operations, computations, and control flow operations. Any computational loop (or even non-loop kernels) will be essentially a set of control flow instructions surrounding a sequence of memory loads, operations, and memory stores. Anything else will be overheads that are useless to the higher-level algorithms but eat up CPU time. The list include in particular function calls, register moves, register spilling, virtual function handling, indirect references, etc.

The entire point of a good optimizing compiler is to both eliminate as much of these overheads as possible and to produce binary code that will be as efficient as possible on the target architecture for the computational parts of the code. Any spurious instructions in the code flow will degrade performance, and much more so when the code is highly efficient. Let's take a synthetic example with a fairly naive x86-64 version of the BLAS **saxpy** function, first in a basic C implementation (**Listing 1**) then in its assembly version (**Listing 2**).

One might consider that the two lines with **incq** (line 11) and **cmpq** (line 12), dealing with control flow, are overheads that should be dealt with. Actually, they're inexpensive: in the context of an otherwise idle integer unit, any modern out-of-order CPU core will execute them at the same time as the useful floating-point instructions. The real killer here is the sequence starting with the load instructions (line 7) as they will take hundreds of CPU cycles if they miss the various caches, followed by the producer-consumer dependency between the multiplication (line 8) and the addition (line 9), and between the addition and the store (line 10). Obviously, the literature is full of solutions to these problems. Loop unrolling will mask remaining overheads from the control flow instructions. It will also, along with software pipelining, try to mask the producers-consumers dependencies. These are textbook considerations, and most if not all of them can be taken care of by the compiler (or indeed the BLAS library itself).

The real point of this example is to illustrate what happens when the compiler does its job properly. While the vectorizer and the aforementioned optimizations will improve the performance of this code, any additional overhead will degrade

it. The worst-case scenario is the introduction of additional, dependent load operations in the loop. Any of those (such as accessing an object before accessing a field in it, or an indirect access through a pointer to pointer) will add extra, potentially large latency to the dependency chain. On Sandy Bridge, a **mulsd** operation has a latency of 5 cycles. A **movsd** from memory can consume from 3 cycles (best case scenario, hitting the L1 cache) to 230-250 cycles (hitting the main memory of the same NUMA node on a dual-socket Sandy Bridge) to 350-370 cycles (hitting the main memory of the other NUMA node) to over 600 cycles (hitting the most remote NUMA node in a quad-socket Sandy Bridge).

That is why analyzing the assembly code is so important, for it is the binary code that will eventually be executed. One cannot reasonably hope to optimize the performance of a program on a processor without understanding how this processor works.

**Listing 8 - Restructuring the code using temporary arrays.**

```
for (int s = 0; s < slices; s++) { // line 123
 for (int i = 0; i < narray; i++) { // line 148
  // some code here
 }
 for (iter = 0; iter < Hniter_riemann; iter++) { // line 171
  for (int i = 0; i < narray; i++) { // line 179
   // some code here
  }
 }
 for (int i = 0; i < narray; i++) { // line 206
  // some code here
 }
}
```

### II.B - General optimization

The basic rule in exploiting a compiler is to start by letting the people who created it do the job. Compilers include a lot of optimizations. Many of them are more or less mandatory because they are cheap (in terms of compilation time) and offers large gains. Some of them are additional extras that are very often used, as they are usually worth the extra compilation time. Finally, some can be very costly, and can have wildly different effects on performance depending on the specific code involved.

The first job is therefore to choose a compiler. The easiest choice is the machine vendor's, as it is likely to be the most suited to the target hardware architecture. But third-party compilers are also of interest, precisely because of their strengths and weaknesses. If multiple compilers are available, try them all. It will help you identify bugs and approximations in the source code and, eventually, lead to the selection of one compiler over the others. This, however, will not be a definitive choice: changing parts or all of the code to improve it might help one compiler more than another, to the extent that the former second choice might become the new first choice. Also, be sure to use the latest available version, barring some fatal bug in it.

The optimization options to try first are those prefixed with **-0** and suffixed with a number (the larger the number, the larger the set of optimizations applied). All codes should work at all optimization levels. If it is not the case, either there is a bug in the compiler or there is a problem with the code. While it is common to blame compilers for failure at high optimization levels, the culprit is generally the code itself. Quite often, the code exercises corner cases or unspecified behaviors of the language, leading to an apparent unreliable behavior on the part of the compiler. In this case, the first step is to ensure reproducibility at all general optimization levels, either by fixing the code or by proving there really is a bug in the compiler (and having the compiler's vendor fix it, of course). Each and every time a new set of options is used, the results of the code should be checked for conformance.

Then, compilers such as Intel's have inter-procedural optimizations that try to optimize the code across function boundaries rather than within each function independently. In recent versions, these optimizations are active by default inside each compilation units (aka files).

**Listing 9 - Compilation output for the code in Listing 8.**

```
HydroC/HydroC99_2DMpi/Src$ icc -std=c99 -O3 -vec-report2 -Wno-unknown-pragmas -S riemann.c
riemann.c(65): (col. 3) remark: LOOP WAS VECTORIZED.
riemann.c(148): (col. 5) remark: SIMD LOOP WAS VECTORIZED.
riemann.c(179): (col. 7) remark: SIMD LOOP WAS VECTORIZED.
riemann.c(171): (col. 5) remark: loop was not vectorized: not inner loop.
riemann.c(206): (col. 5) remark: SIMD LOOP WAS VECTORIZED.
riemann.c(123): (col. 3) remark: loop was not vectorized: not inner loop.
riemann.c(300): (col. 2) remark: SIMD LOOP WAS VECTORIZED.
riemann.c(296): (col. 7) remark: loop was not vectorized: not inner loop.
riemann.c(295): (col. 5) remark: loop was not vectorized: not inner loop.
```

The **-ipo** option enables these optimizations across compilation units. Instead of compiling each file independently, the compiler simply creates stubs and compiles everything in one go. The downside is that the entire code has to be recompiled each time anything changes. The upside is that all the information is available to the compiler, including hard-coded values (array bounds, constants...) and call trees. This allows the compiler to do a much better job: for instance, if it is aware of the number of iterations in a loop, it can unroll more efficiently. If a scalar parameter to a function is known to be a constant, the compiler can use that knowledge. For example, nice constants such as 0 and 1 are identity elements for the addition and multiplication respectively; using them as such prevents redundant computations.

Another must-try with recent Intel compilers is the set of inlining options. Normally, a compiler will try to inline a certain number of functions but will use heuristics to prevent an excessively large code size and/or compilation time. However, this often limits its ability to merge loops and/or functions, and to eliminate redundant computations and/or memory accesses. A particularly aggressive set of options is

**-inline-forceinline**
**-no-inline-factor**
**-no-inline-max-per-compile**
**-no-inline-max-per-routine**
**-no-inline-max-total-size**
**-no-inline-max-size**
**-no-inline-min-size**

which, combined with **-ipo**, tells the compiler to merge as much of the code as it can into computational functions, regardless of compilation time, its own memory usage and the size of the resulting binary. Although these options combined can be really aggressive in some contexts, many codes will actually benefit a lot from them. For instance, when applied to the Qiral QCD code [6], the improvement is superior than 2X with version 13.1.1.163 of the Intel compiler.

Those are general guidelines for the most common options; each compiler has its own set of advanced settings, some of which can be useful on some codes. As long as the results of the code remain valid, any set of options can be used. Again, if "the compiler broke the code", then there is a problem somewhere that must be tracked down. Otherwise, the problem may reappear with a different compiler, a different machine or, worse, a different data set.

### II.C - Vectorization

The next step is to understand whether the compiler properly vectorizes the code. The word "vector" shouldn't be taken too literally here, as it usually calls back to vector systems (from the Cray 1 onward). In the current era, "vector" refers to the short, fixed-length registers in the CPU and their associated operations (like the SSE and AVX instructions sets in the x86-64 world or the QPX on the IBM BlueGene/Q). There have already been detailed works on the subject of vectorizers, such as Intel's own [7]. These are of course interesting for advanced users, but might be too detailed for the newcomer to vectorization.

The principle of vector instruction sets is to allow a single instruction to perform multiple operations at once, thus adding potential performance at a low cost. Analyzing why this is a good trade-off would require a long explanation; suffice it to

**Listing 10 - An example of SSE's reuse of a source operand to store computation results.**

```
66 0F 59 /r                                RM  V/V SSE2 Multiply packed DP floating-point
MULPD xmm1, xmm2/m128                      values in xmm2/m128 by xmm1.
VEX.NDS.128.66.0F.WIG 59 /r RVM V/V AVX    Multiply packed double-precision floating-point
VMULPD xmm1,xmm2, xmm3/m128                values from xmm3/mem to xmm2 and stores
                                           result in xmm1.
(...)

MULPD (128-bit Legacy SSE version)
DEST[63:0]  DEST[63:0] * SRC[63:0]
DEST[127:64]  DEST[127:64] * SRC[127:64]
DEST[VLMAX-1:128] (Unmodified)
VMULPD (VEX.128 encoded version)
DEST[63:0]  SRC1[63:0] * SRC2[63:0]
DEST[127:64]  SRC1[127:64] * SRC2[127:64]
DEST[VLMAX-1:128]  0
```

say that nearly all contemporary CPU use such instruction sets. We will detail some interesting differences between them in section D below.

A good starting point to understand vectorization better is SSE2's **mulsd** mentioned in section A. In the Intel documentation [8], this instruction's full name is "Multiply Scalar Double-Precision Floating-Point Values". As this description implies, **mulsd** performs a multiplication on double-precision floating-point values. However, it has a counterpart called **mulpd**, or "Multiply Packed Double-Precision Floating-Point Values". Instead of a single multiplication, **mulpd** executes one per element in the so-called vector. The precise size of the vector depends on the instruction set (SSE or AVX). If we take the example of SSE where vector registers are 128 bits, then we have two double-precision value (64 bits each, as per IEEE754-2008) in each register. In other words, each instruction will compute 2 flops instead of one. If we go back to the **saxpy** function from **Listing 1**, we can update the assembly version to exploit this new instruction, as shown in **Listing 3**.

Obviously, this code will not work if the number of elements in the vector is not a multiple of two, as we compute two values for each iteration (note that it was hand-modified from the automatically generated code in **Listing 2** as the compiler vectorized code, including all the necessary checks, is not suitable for an introductory example). But it does illustrate the point that with the same number of instructions, we perform twice as much work. On such a simple loop the compiler has no issue with vectorization. It will even mention it in its output when using the **-vec-report3** option, as can be seen in **Listing 4**.

However, most loops are not that simple, and this is where understanding the compiler is important. A modern compiler will inform you about its successes and failures, but what these actually mean is not always obvious. Let's take a very concrete example with Hydro. Hydro is a mini-application built from N-body, hydrodynamical code RAMSES. It is used to study large-scale structures and galaxy formation. This code includes several variants, each of them designed to take advantage of a certain kind of hardware: a standard C version using OpenMP directives and MPI calls; a version using OpenACC directives to exploit accelerators; an OpenCL for the same purpose, etc. What we are going to look at is the **riemann** function, which exists in two radically different versions: the original in the OpenACC version of the code and an updated and vectorization-friendly variant in the C version.

Compiling the original, untweaked version results in what is shown in **Listing 5**. The loop nest of interest is the one referenced at line 102, 107 and 138 (see its structure in **Listing 6**). The compiler output shows that it failed to vectorize the innermost loop at line 138, because it couldn't be certain that all iterations were data-independent from each other. The compiler is absolutely

right: this innermost loop is a convergence loop, and each iteration is predicated on the results of all the preceding iterations. Therefore, it cannot be easily vectorized. As for the other two loops, the compiler doesn't even try to vectorize them since it could not process the innermost loop, a requirement for the current version of the vectorizer.

There is an easy way to force the compiler to vectorize the code as it is: if we replace the upper bound of the innermost loop (**Hniter_riemann**) by a hardwired value like 10, then the compiler can fully unroll the innermost loop, making the **narray** loop the new innermost loop. In practice, the compiler still fails as it still can't figure out that the two external loop are fully parallel, but a directive exists to help it out: **#pragma ivdep**. Using it on both the external loops allows vectorization of the code (with slightly modified line numbers), as show in **Listing 7**.

However, this does not really help: the code is now wrong, not to mention that hardwiring a value is not an acceptable practice. To be able to properly exploit the hardware in this critical function, the code needs to be modified so that the compiler can vectorize it with the original semantic. For this particular code, the author decided to switch the convergence loop (the problematic one at line 138) and the **narray** loop at line 107. Of course, this implies a complete reorganization of the loops (and the use of additional temporary arrays).

**Listing 11 - An example of logical rotation using SSE instructions.**

```
movdqa      %xmm6, %xmm8
movdqa      %xmm6, %xmm14
psllq       $1, %xmm8
psrlq       $63, %xmm14
por         %xmm14, %xmm8
```

**Listing 12 - An example of logical rotation using AVX instructions.**

```
vpsllq      $1, %xmm0, %xmm14
vpsrlq      $63, %xmm0, %xmm15
vpor        %xmm15, %xmm14, %xmm14
```

The new structure of the code is shown in **Listing 8**, while the result of compiling it is given in **Listing 9**. Note that the presence of an extra "SIMD" in the message is an artefact of the directives required to tell the compiler that the loop iterations are data-independent.

There is a lot of reasons for the vectorizer to fail, and this paper is not the place to explain them all. If some compiler output messages are self-explanatory, unfortunately some aren't. That being said, a few basic rules can help with identifying and fixing issues.

• The Intel compiler only tries to vectorize the innermost loop. Small innermost loops can and should be unrolled (automatically, by a pragma or manually) if the number of iterations is known. If it is not known, the situation is more difficult. It might be possible to invert the loop with a larger, parallel outer loop and the addition of temporary arrays.

• Except for some types of reduction (arithmetic accumulation in a scalar variable), the Intel compiler vectorizes parallel loops. If the compiler complains that there is a vector dependence, check that there really isn't, and use the appropriate directives to help with the compilation.

• Conditionals in the loop nest might prevent vectorization, and make it less efficient anyway. It's often a good idea to move iteration-independent conditions out of the loop nest.

While the compiler will tell which loops have been vectorized, it is not the only condition for successfully using vector instructions sets. Compilers will often generate multiple variants of a loop depending on various parameters (number of iterations, alignment of data arrays, etc.) but there is no guarantee the vector version will always be picked up at runtime.

Note that hardware counters inside the CPU can be used to validate the results during execution. Tools such as VTune or Likwid can also be very useful to validate that the compiler's vectorizer has been successfully exploited.

### II.D - The target machine

We mentioned it in previous sections, there are two main vector instruction sets for the x86-64 platform: SSE and AVX. SSE was originally meant for single-precision floating-point and integer data. Double-precision floating-point was added in SSE2, and subsequent extensions added new instructions for various purposes. They all have in common a 128-bit register size, which is enough to hold four single-precision values, or two double-precision values, in the XMM registers.

The AVX instructions set was introduced using a completely new instructions encoding scheme (called VEX). While this does not really concern most programmers, AVX's main selling point was the doubling of the register width to 256 bits, for eight single-precision or four double-precision values. In the original AVX, only floating-point values can be worked on in the 256 bits YMM registers. The lower half of these YMM registers is aliased with the XMM registers. As you can guess, going from SSE to AVX instructions in the same code causes a small performance penalty.

One of the most overlooked feature of the new AVX instruction set is the VEX.128 subset of instructions. These instructions only work on the lower 128-bit part of the YMM registers - the part aliased with XMM registers. While they may seem quite redundant with the SSE instructions, they actually offer a very important upgrade: whereas SSE instructions have mostly two operands, AVX instructions have three. **Listing 10** is an excerpt from the Intel documentation [8] in which the SSE version stores its result in source operand **xmm1**, while the VEX.128 version destroys neither of its source operands. Whenever both sources are reused, this saves a register-register **movapd** instruction. This may not look like a lot, in particular in numerical codes where one of the operands will be one-use only most of the time.

But the VEX.128 instructions are not limited to the floating-point instructions; many integer instructions have been re-implemented as well. They cannot access the upper 128 bits of the 256 bits YMM registers, but they do have access to the new three-operands format.

A code that benefits greatly from this is the **Keccak** algorithm by Bertoni et al., the winner of the NIST competition to create SHA-3. The "SIMD instructions and tree hashing" implementation uses integer SSE instructions to implement two 64-bit instances of the algorithm simultaneously. The implementation is done using C intrinsics rather than assembly. The instructions in use are mostly logical **xor**'s, logical **or**'s and **shifts** (used to implement the rotation of a 64 bit value).

A brief extract implementing a rotation is shown in **Listing 11** for SSE and **Listing 12** for AVX. Extra copies to preserve the input values are required by the algorithm, which reuses the input data several time. Statistics on the partially unrolled loop show that the number of data movement instructions has gone from 396 (35% of the 1132 instructions) to 111 (13% of the 837 instructions). All the remaining moves are memory accesses, none of them being register-register. Speed measurement shows an improvement of more than 20% for short hashes. As the only cost for high-level language or intrinsics-based code is to use the **-mavx** options (or similar), switching from SSE to AVX can offer a nice improvement for a very small cost.

Next month, we'll take a detailed look at how data structures can impact both code performance and algorithm complexity - with a focus on the optimization of parallel codes. In the meantime, happy (performance) programming! ■

# Subscribe now!

## [for free, forever]

Subscribe now and receive, every month,
an expert, actionable coverage of HPC and
Big Data news, technologies, uses and research...


+ get yourself access to exclusive contents and services

# www.hpcmagazine.com