# D4.3 Final report on enabling technologies


| | |
|---|---|
| CONTRACT NO | EESI2 312478 |
| INSTRUMENT | CSA (Support and Collaborative Action) |
| THEMATIC | INFRASTRUCTURE |

Due date of deliverable: 31/11/2014

Actual submission date: 31/07/2015

Publication date: 31/07/2015

Start date of project: 1 September 2013 Duration: 30 months

Name of lead contractor for this deliverable: PRACE-BSC, Rosa M. Badia

Name of reviewers for this deliverable:


Abstract: This is the first intermediate report of EESI2 WP4 Enabling Technologies WGs. The document reports on the initial findings of each of the WGs and present initial recommendations.

Revision: 1.0

| Project co-funded by the European Commission within the Seventh Framework Programme (FP7/2007-2013) | | |
|---|---|---|
| **Dissemination Level to be filled out** | | |
| **PU** | Public | X |
| **PP** | Restricted to other programme participants (including the Commission Services) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission Services) | |
| **CO** | Confidential, only for members of the consortium (including the Commission Services) | |

# Table of Contents

**1. EXECUTIVE SUMMARY ..... 3**

**2. WP4 COVERAGE ..... 4**

**3. ACTIVITIES IN THE BIG DATA AND EXTREME-SCALE COMPUTING (BDEC) MEETINGS ..... 5**

**4. WG 4.1 NUMERICAL ALGORITHMS ..... 6**

4.1 SCIENTIFIC CONTEXT AND TASKS OF THE WORKING GROUP ..... 6
4.2 ORIGINS OF EXPERTISE ..... 6
4.3 WG4.1 FINAL REPORT ..... 7
4.4 UPDATE ON KEY CHALLENGES AND GAP ANALYSIS BY AREA ..... 8

**5. WG 4.2: SCIENTIFIC SOFTWARE ENGINEERING, SOFTWARE ECO-SYSTEM AND PROGRAMMABILITY ..... 11**

5.1 MEMBERS AND THEIR EXPERTISE ..... 11
5.2 KEY CHALLENGES ..... 12
5.3 DEVELOPMENTS DURING THE LIFETIME OF EESI2 ..... 13
5.4 GAP ANALYSIS ..... 13
5.5 PROPOSITION OF R&D PROGRAM FOR 2015 AND BEYOND ..... 14
5.6 FURTHER RECOMMENDATIONS ..... 14

**6. WORKING GROUP 4.3 "DISRUPTIVE TECHNOLOGIES" ..... 17**

6.1 MEMBERS AND THEIR EXPERTISE ..... 17
6.2 ANALYSIS ..... 17
6.3 IDENTIFIED DISRUPTIONS AND TECHNOLOGIES THAT ENABLE TO COPE WITH THEM ..... 18
6.4 REACTING TECHNOLOGIES ..... 18

**7. WORKING GROUP 4.4 "HARDWARE AND SOFTWARE VENDORS" ..... 21**

7.1 KEY CHALLENGES ..... 22
*7.1.1 Hardware: Energy efficiency, Power Wall, Power Density ..... 22*
*7.1.2 Hardware: Memory/Storage Capacity, Packaging, Bandwidth ..... 22*
*7.1.3 Hardware/Software: Reliability/Resilience/Fault Tolerance ..... 22*
*7.1.4 Software: Inter-/Intra-Node Scalability to 1M Tasks, also Tools & Debuggers ..... 22*
*7.1.5 Software: Programmability & Programming Environments ..... 22*
*7.1.6 Software: Characteristic Mini-Apps / Benchmarks ..... 22*
7.2 MARKET AND TECHNOLOGY WATCH ..... 23
*7.2.1 Hardware: Energy efficiency, Power Wall, Power Density ..... 23*
*7.2.2 Hardware: CPUs, GPUs, and Accelerators ..... 23*
*7.2.3 Hardware: Memory/Storage capacity, packaging, bandwidth ..... 23*
*7.2.4 Hardware: Reliability/Resilience ..... 24*
7.3 GAP ANALYSIS ..... 24
*7.3.1 Hardware ..... 24*

**8. CONCLUSIONS ..... 25**

# Glossary

| Abbreviation / acronym | Description |
|---|---|
| DAG | Direct Acyclic Graph |
| EMWG | Exascale Mathematics Working Group |
| FMM | Fast Multipole Method |
| | |
| | |

# 1. Executive Summary

This document is the final EESI2 report on enabling technologies, corresponding to WP4. The WP4 is composed of four Working Groups (WG).

With regard WG4.1, Numerical Analysis is an enabling technology that underlies all numerical computation in all application areas. The efficient and reliable implementation of these core numerical algorithms is crucial and essential if we want to realise the potential of future Exascale systems.

With regard WG 4.2 Scientific software engineering, software eco-system and programmability, tackles the development, operation and maintenance of software. The challenges in this area come from difference sources, between them the long life of codes or the lack of high-level programming environments.

WG4.3 focuses on disruptive technologies in enabling technologies. The activity in the group has focused on identifying how the disruptions can be identified, and afterwards identifying technologies that will help handling the disruptions.

WG4.4 focuses on establishing and maintaining a global network of contacts with vendors in the HPC industry and to leverage this network to investigate the state of the art and trends related to the Exascale roadmap in the HPC hardware and software industry.

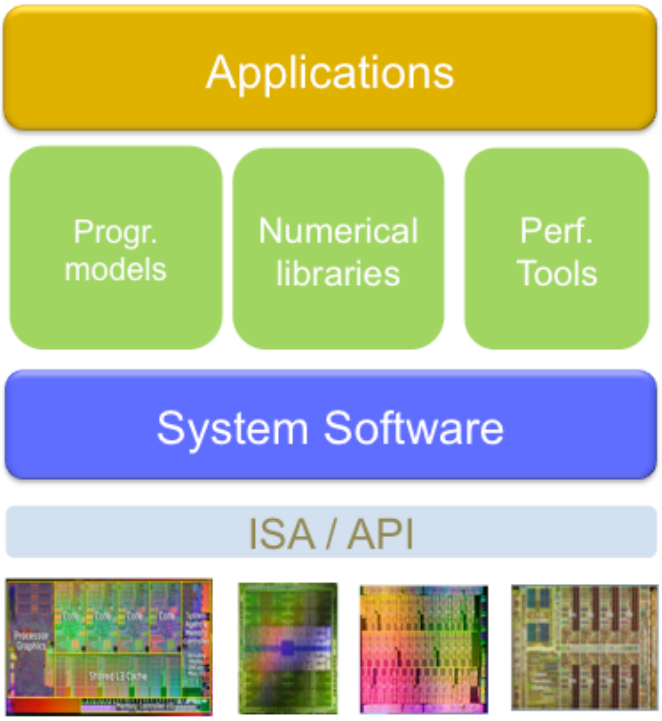
This deliverable presents the final report of the WP, summarizing the findings of the different WGs along the project lifetime. Additionally to what it is presented in this document, the WGs have been actively participating in the project recommendations, especially in the following ones:

- Ultra scalable algorithms
- Resilience
- High productivity programming models
- Software Engineering Methods for High-Performance Computing
- Verification, Validation and Uncertainty Quantification
- Algorithms for Communication and Data-Movement Avoidance
- High productivity programming models for Extreme Computing
- Software Engineering Methods for High-Performance Computing

# 2. WP4 Coverage

This is the second deliverable of WP4 "Enabling Technologies". This WP covers the different levels offered to the applications: programming models and performance analysis tools, numerical libraries, system software, and also the hardware level. Although organized in different WGs, a comprehensive view is necessary, since in order to achieve the exascale challenges a collaboration between all levels is necessary.



# 3. Activities in the Big Data and Extreme-Scale Computing (BDEC) meetings

The WP participants, as the whole project, have participated in the recent BDEC meetings, especially in the one organized in January 2015 in Barcelona and in the ISC workshop in Frankfurt in June 2015.

The meeting was organized in session presentations and in breakouts of the following topics:

- Algorithms and Applied Math
- Applications and Science
- Architecture and Operations
- Software Stack

The final report of these meetings is under elaboration, but some of the conclusions can be found in the meeting website: http://www.exascale.org/bdec/documents/barcelona

Between the conclusions, we outline:

- There is a growing overlap between both worlds (Big Data and Exascale). Many Exascale applications produce Big Data and Big Data is a growing consumer of HPC capabilities.
- There is a growing opportunity for common approaches, common software frameworks, and even reuse of data structures.
- There are differences in the data and how it is structured in both worlds, and to promote convergence there is a need for new data structures.
- In terms of HW, Big Data has traditionally used more memory and distributed storage, but vendors are somehow forcing similar HW configuration for both worlds.
- There is a need for joint testbeds that enable evaluation of both types of applications.
- There is a need to build open data repositories and develop data challenges that highlight the unification of the techniques.
- There should be friendlier allocation processes for major facilities for data intensive workflows.
- There is a need for more meetings to enable both communities to meet and share their views.

# 4. WG 4.1 Numerical Algorithms

## 4.1 Scientific context and tasks of the working group

Numerical Analysis is an enabling technology that underlies all numerical computation in all application areas. The efficient and reliable implementation of these core numerical algorithms is crucial and essential if we want to realise the potential of future Exascale systems.

Our basic building blocks involve dense matrix kernels, the most ubiquitous being the multiplication of two dense matrices, a kernel that should be designed to attain the peak performance of the machine. This software may be used directly on extremely large problems or may itself be a building block for the factorisation of large sparse matrices and the solution of the corresponding set of equations which may come from the discretisation of a continuous problem, for example the solution of a three-dimensional PDE. An alternative for solving large sparse systems is to use iterative methods where the main kernel is usually a sparse matrix-vector computation.

Very similar iterative methods can be used in the solution of large eigensystem problems where only a subset of eigenvalues and vectors are required. More recently there have been advances in combining direct and iterative methods in so-called hybrid methods that can again be designed to exploit the hierarchical structure of the evolving hardware. We also discuss software that sits further up the stack including problems in control and the major area of optimization, both linear and nonlinear. A major tool for decomposing large problems for all these approaches is graph and hypergraph partitioning which we also discuss, including the parallel implementation of software in this area. We then comment on aspects of structured and unstructured grid calculations and parallel random number generation, particularly in the context of Monte Carlo methods.

As for EESI-1 we have found it useful to break down our area into the following subtopics: Dense linear algebra, Graph partitioning, Sparse direct methods, Iterative methods, Eigenvalue problems, Optimization & control, Structured & unstructured grids and Monte Carlo. These are listed roughly in the order they appear on the software stack, i.e. Dense Linear Algebra is the building block on which most other areas depend, and so on. Due to their increase importance we have decided to add the topics of Tensors and Fast Multipole Methods as separate items.

Topics such as Dense and Sparse Linear Algebra that are used in all other areas have always had the highest pressure to develop efficient implementation. As a result when it comes to exascale progress is more advanced and algorithms are more mature in these areas. On the other hand addressing the remaining gaps is also a prime priority.

This section presents an update with regard deliverable D4.1 for this WG.

## 4.2 Origins of Expertise

The working group consists of a chair and vice-chair and eleven experts chosen to cover the domains of interest. Their names and area of expertise are listed below:

| Name | Organization | Area of expertise |
|---|---|---|
| Andreas Grothey | University of Edinburgh | Continuous & Stochastic Optimization |
| Iain Duff | STFC | Sparse Linear Algebra |
| Jack Dongarra | University of Manchester | HPC, Numerical Linear Algebra |

| | | |
|---|---|---|
| Mike Giles | University of Oxford | GPU, CFD/Finance, Grids, Monte Carlo |
| Thorsten Koch | Koch Zuse-Institut Berlin | Combinatorial Optimization |
| Peter Arbenz | ETH Zürich | Eigenvalues, Iterative Methods |
| Bo Kågström | Umeå University | Dense Linear Algebra |
| Julius Žilinskas | Vilnius University | Global Optimization, Meta-heuristics |
| Salvatore Filippone | Università di Roma "Tor Vergata" | Numerical Software |
| Luc Giraud | INRIA Bordeaux | Iterative Methods, Multipole Methods |
| Patrick Amestoy | ENSEEIHT-IRIT, Université de Toulouse | Direct Methods, Solvers |
| Karl Meerbergen | K.U. Leuven, ExaScience Lab | Eigenvalues, Tensors, Model reduction |
| Francois Pellegrini | LaBRI Bordeaux | Partitioning |

## 4.3 WG4.1 Final report

The final report of the working group 4.3 "Numerical Libraries, Solvers & Algorithm" of EESI-1, provides a detailed discussion of the state-of-the-art and challenges to achieve exascale performance within the remit of the current working group. Although now three years old, most of the discussion are still valid. The current report should be read as an addendum to the previous document.

Many of the challenges facing numerical algorithms are common across our remit. As memory accesses are increasingly the bottleneck in computations algorithms need to maximise the number of useful calculations per memory access. This is frequently achieved by blocking/tiling and communication hiding. Load balancing issues mean that synchronisation points are expensive and asynchronous versions of existing algorithms need to be investigated. Often such algorithms have been suggested in the past but have been unfavoured due to stability issues that are difficult to understand and control. These algorithms ought to be given a fresh look but better understanding of their theoretical properties is required. Dynamic scheduling based on DAG representations of algorithms is better suited to exploit all possible concurrency in computations. To make full use of its potential and to avoid unnecessary synchronisation points the traditional hierarchy of numerical library calls (matrix-vector multiplication called by an iterative solver called after a graph partitioning routine – which by design lead to a fork-join execution model) should be broken down. This has fundamental implications on the design of numerical libraries since correspondence of library routines with mathematical operators would not necessarily be maintained. Care is required however to ensure maintainability and stability of the resulting algorithms. A complete rethink of the current approach to writing numerical libraries is needed to provide scalable software framework.

Other common issues that will be increasingly important are to address the trade off between speed, accuracy and reproducibility, the impact of fault tolerance on algorithm design and uncertainty quantification. Naturally progress towards dealing with these challenges is more advanced in some areas than in others.

Finally exascale computing opens up new application areas, such as multiscale/multiphysics modelling and big data processing. Matrices arising from these applications frequently possess different structural properties to the traditional application areas of numerical algorithms. A certain amount of redesign, often on a quite fundamental level, is needed to deal with these structures efficiently. Especially for big data processing low-rank approximations and numerical methods to assist these such as Tensor computations and Fast Multipole Methods are increasingly important.

It needs to be mentioned that very few areas within the scope of this working group have easily identifiable gaps that can be closed by concentrated effort alone. Mainly these are where established algorithm and paradigms have to be adapted to new emerging application areas. One of them is dealing with uncertainty. Often progress is slow but steady. Solutions may not appear where they were initially suspected. It is therefore paramount that many possible avenues are explored and not only the (initially) most promising ones. In many areas fundamental breakthroughs are needed and these require sustained funding over many years. Challenges have been known for years and fast progress is not to be expected. Therefore little concrete progress can be reported over the previous three years.

On issue that has been hampering progress is lack of funding avenues for long term maintenance of key libraries and development of standards. The concentration of research on key highlights in select areas leads to fragmentation and divergence of efforts.

One notable development in the last few years has been an increased focus on Big Data and Data Science both in terms of research undertaken and available funding. Recent progress in this area includes

- First order methods for optimization,
- Compact representations/low-rank approximations/clustering
- Hierarchical algorithms
- Randomised algorithms

We welcome this development. Although “Big Data” is not the only research challenge, many of the challenges involved are the same as for exaflop computing.

We welcome the Horizon2020 call on “New Mathematics for Exascale”. This call has received strong interest from the scientific community including Numerical Algorithms and more than 80 applications have been received.

## 4.4 Update on key challenges and Gap Analysis by area

**Dense Linear Algebra:** Scalable and reliable parallel algorithms and the library functionality available for distributed-memory systems are lagging behind that offered for shared-memory systems and accelerators. Algorithms based on Level 3 BLAS have the potential to approach sustained (practical) peak performance. Research is focusing on exploiting this performance through increased use of blocked algorithms and data structures such as hierarchical blocking, recursive blocking, tiling, and efficient matrix storage format conversions. The current parallel BLAS imposes a fork-join model of parallelism and implicitly synchronises the processors at the beginning and end of each operation. Computations need to be expressed at multiple levels of abstraction as task graphs and make use of a data-driven scheduler.

The last couple of years have seen several new developments with respect to multicore and multi-GPU heterogeneous algorithms and frameworks with auto-tuning, but only modest improvements with respect to distributed memory. Moreover, the current efforts are diverging due, in no small part, to the lack of standards for expressing fine-grained parallel algorithms in a framework independent language. The existing frameworks are typically monolithic and specialised to handle certain types of algorithms. Future challenges include the development of new scalable dense linear algebra algorithms based on modular frameworks that support alternative scheduling methods, memory affinity schemes, load balancing methods, etc.

**Tensors** arise frequently in applications to express principal components of high dimensional measurements. Scientific computing for uncertainty quantification and parametric matrix computations is expected to make more use of tensors. Traditionally tensor calculation relies on finding decompositions and basing calculations on them. This approach requires efficient exploitation of tensor structures in SVD, dense GEMM based operations and least squares which is still lacking.

Few computations are done on tensors directly, although this may change with exascale capabilities. There is little progress so far on efficient implementations of direct tensor calculations and most work has to be started up. Basic efficient tensor operations (i.e. BLAS4) are lacking and need to be developed. The target is far from being in sight. Since tensors will be used in preconditioners, model reduction etc., the time line of tensor software development is connected to the timeline of those themes.

Parallel **graph partitioning** software such as PT-Scotch and ParMetis have been around for a number of years, and work well for many problems up to fairly large scale. To achieve scalability up to the levels required for Exascale a new generation of software for multi-set partitioning and partition refinement is needed. This is of high importance since it is used by many other algorithms.

The main challenges for **Sparse direct** methods are the adaptation to matrix structures arising from new applications such as big data processing but also the increasing exploitation of low rank approximations and compression methods. There is very active development in memory scalability and hybrid methods but more work is needed. Energy aware algorithms are probably unavoidable but it is currently far from obvious how they would be integrated into libraries.

For **iterative methods** the main emphasis remains on the development of efficient (parallel) preconditioners. The design of variants of Krylov solvers that enable to hide the communication is also a very active field. Additive rather than multiplicative preconditioners have advantages for parallelisation and should be explored. In view of the increasing importance of uncertainty quantification and sensitivity calculations Krylov methods will increasingly be employed for multiple right hand sides. This offers scope for parallelism that has not been exploited yet. Otherwise effort should concentrate on matrix-vector multiplications since they are now limiting speedup. Asynchronous/chaotic relaxation methods offer much scope for parallelisation but give rise to stability issues that are poorly understood. Better theoretical understanding is needed here.

**Eigenvalue solvers** depend largely on iterative or direct linear solvers and their parallel performance is correspondingly mainly determined by those components. Nonlinear eigenvalue problems will be increasingly important as mathematical models in, e.g., mechanical engineering and physics, become more complicated. Theory and implementations are currently in progress. Europe takes the lead in this area (groups in Leuven, Manchester, Berlin, Lausanne). Contour integral methods are an example of a previously explored idea that has been rejected in serial but should be evaluated due to its parallelisation potential. Japan is taking the lead in this work.

The **Fast Multipole Method** (FMM) developed for astrophysics and molecular simulations solves the N-body problem for any given precision with O(N) runtime complexity against O(N2) for the direct computation. The idea is to decompose the potential field in a near field part, which is directly computed, and a far field part that is approximated (either by expansions or by interpolation). The FMM has application in many other fields including elastic materials, fluid mechanics, Helmholtz and Maxwells equations, the Laplace equation, etc. To reduce the computational cost, adaptative techniques based on irregular octrees are of interest, where the depth of the octree varies. The resulting irregular computation implies load balancing issues that must be addressed. There are active groups, although mainly in the US (Courant Institute, Georgia Tech, Maryland, Stanford, Boston). European partners are involved as collaborators. Currently computations scale up to a few thousand cores.

For **Optimization** the key challenge remains the lack of scalable and efficiently warmstartable algorithms for LP, which is the main building block for more complex problem classes such as nonlinear or mixed integer programming. Barring a new disruptive technology progress is likely to be very gradual and has been for years.

Efficient preconditioners for larger classes of optimization problems remain a priority. The field is starting to mature, but building preconditioners still needs expert intervention. Hierarchical preconditioners are promising. There is significant expertise in Europe with several active research groups. Optimization under uncertainty has the potential to become a prime driver for exascale issues in optimization. Uncertainty structure can be efficiently exploited for massive parallelisation for two-stage linear problems. Still much work needed for nonlinear and mixed integer problems.

Progress has been made on the development of first order methods (such as randomised coordinate descent or, for stochastic problems, stochastic gradients) which are able to solve very large problems (such as arising from processing of large data sets) to at least reasonable accuracy. For extremely sparse problems these methods have been reported to parallelise well.

Due to the location of optimization on top of the software stack, pressure for efficient massive parallelisation has in the past not been as high as in other fields. Techniques such as hierarchical/multilevel methods, synchronisation avoidance, communication hiding and dynamic scheduling are only just starting to make an impact in optimization. This gap needs to be filled for exascale.

**Structured/unstructured grids:** Multi-block structured grid applications are often bandwidth-limited. Tiling overlaps execution of blocks resulting in much greater re-use of data within caches, reducing the amount of data transfer. However, tiling can be very tricky to implement, which is why it is almost never used except for fairly trivial applications. High level abstractions could help implement tiling in a way which is transparent to the application programmer.

The main activity in the area is in US; for example Berkeley and collaboration of MIT with Intel. Europe is in a good position to contribute with high level abstraction where there is a strong history.

**Uncertainty Quantification** is of growing importance in science and engineering. Because of this, there is increasing interest in **Monte Carlo** and Markov Chain Monte Carlo (MCMC). The latter is able to take Bayesian inference into account: each new ”instance” possibly involving the parallel solution of a PDE approximation. Significant progress on Multilevel Monte Carlo which leads to natural massively parallel computation (many trivially-parallel processes each of which is itself often a heavily-parallel computation) and is thus well-suited to Exascale computing. It is now becoming the method of choice in SPDE modelling areas such as sub-surface flow modelling for nuclear waste repositories.

**Auto-tuning** has been used in the past for specific libraries such as BLAS and FFTW, but with the increasing complexity of HPC hardware and software it is becoming essential for almost all applications.

On a very basic level, as new chip designs such as ARM are making increasing inroads even for high performance computing, it is vital that there are high quality, **high performance numerical libraries** available for these processors. Europe has various centres (like the PRACE centres in Barcelona, Jülich, CINECA, Edinburgh, or a not-for-profit supplier such as NAG) that would be well suited for this activity.

# 5. WG 4.2: Scientific software engineering, software eco-system and programmability

This working group focuses on methods, processes, tools, and support structures required to create robust, correct, efficient, and maintainable code under economic constraints. In an adaptation of ISO/IEC/IEEE 24765:2010, we define "[scientific] software engineering as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to [scientific] software". Our target software includes mostly highly scalable simulation codes but also other data-intensive applications such as graph analysis and is developed in both academic and industrial settings.

## 5.1 Members and their expertise

| Name | Organization | Area of expertise |
|---|---|---|
| Felix Wolf (chair) | TU Darmstadt, Germany | Parallel programming tools (performance analysis & modeling, parallelism discovery), parallel programming models |
| Matthias Mueller (co-chair) | RWTH Aachen University, Germany | Parallel programming models, correctness checking, runtime error detection, performance analysis, energy efficiency |
| Mike Ashworth | Science and Technology Facilities Council, UK | High-performance applications development, numerical algorithms, benchmarking, languages, software tools and environments |
| Achim Basermann | DLR (national aeronautics and space research centre), Germany | Parallel numerical algorithms and data structures, development of parallel applications, parallelization technology for modern computer architectures, Python for HPC |
| Vincent Bergeaud | CEA, France | Software engineering in scientific simulation, parallel programming and uncertainty analysis |
| David Brayford | Leibniz Computing Centre, Germany | Software design, software architecture, software development, low level (CPU, embedded, SIMD, drivers), high level (applications) |
| Jim Cownie | Intel, UK | Parallel programming, Occam, MPI, OpenMP, Cilk, Fortran, UPC, parallel debugging, message passing hardware, computer architecture from the SW viewpoint |

| | | |
|---|---|---|
| Alessandro Curioni | IBM Zurich, Switzerland | Scientific computing, parallel programming, computational sciences, algorithm re-engineering for massive scaleout, HPC applications design and maintenance |
| Torsten Hoefler,: | ETH Zurich, Switzerland | Parallel programming models, parallel library development and stacking, performance modeling, performance portability, message passing, RMA, scalable algorithms and runtime systems |
| Horst Lichter | RWTH Aachen University, Germany | Software architectures, metrics and measurement, requirements engineering, model-based development, test and validation, software development processes |
| Mariana Vertenstein | NCAR, USA | Software development, performance analysis for community climate models, testing and validation, software development processes |
| Andrea Walther | University of Paderborn, Germany | Algorithmic differentiation (AD) including the development of an open-source AD tool, nonlinear optimization, high performance computing for the simulation of complex systems and their optimization |

## 5.2 Key challenges

At exascale, applications have to address numerous technical hurdles simultaneously, including (i) scalable and energy efficient algorithm design, (ii) hardware faults which may compromise scalability, (iii) soft errors which may compromise correctness, and (iv) the pre- and post-processing of vast amounts of input and output data. As a consequence, the development costs of such applications will rise significantly, creating a need to reassess software processes from an economic perspective. While algorithm development and parallel programming models have received considerable attention in the past, the software engineering aspects of HPC have been broadly neglected. The main challenges arise from:

- The long lifetime of codes, which makes developers reluctant to adopt new and potentially unstable technologies
- The difficulty of verifying and validating the correctness of results that cannot be precisely reproduced in experiments
- The lack of stable high-level parallel programming environments with a long-term perspective, especially in view of increasingly heterogeneous target hardware
- Incrementally specified requirements and – at least in academic environments – high staff fluctuation, which both lead to organic growth of the software
- Multi-physics problems, which frequently require the coupling of methods across many length and time scales
- The desire to maintain portability across a range of modern and emerging parallel hardware platforms

Under such constraints, performance and maintainability are often conflicting goals. Because these challenges are already present today, they may drastically slow the evolution of applications and the underlying software ecosystem as we move towards exascale.

## 5.3 Developments during the lifetime of EESI2

We have seen substantial extensions of the most popular programming interfaces MPI and OpenMP. In September 2012, MPI 3.0 was approved, which brought numerous new features, including non-blocking collectives for the overlap of computation and communication, neighborhood collectives for the simplification and optimization of stencil exchanges, additions to the RMA interface to make it more suitable as the underlying substrate of PGAS languages, and a tool information interface for the inspection and manipulation of MPI control and performance variables. MPI-3's extensions were quickly deployed and adopted. Especially the new RMA programming model, known from UPC and Fortran 2008, has been gaining attention and fast implementations are available. Moreover, OpenMP 4.0 was ratified by the ARB in July 2013. Among other features, the new version covers accelerator support, an enhanced tasking model with dependencies, thread affinity, and support for SIMD instruction-level parallelism. Finally, new task-based programming models such as OmpSs and Charm++ are quickly evolving and may be adopted at large scale. All these affect software development methodologies in that we require new tools and development environments for these new programming models.

In general, a trend can be observed away from explicit accelerator programming to directive-based approaches that leave the accelerator-specific part to the compiler, improving portability, maintainability, and programmer productivity. In June 2013, version 2.0 of OpenACC was released with support for dynamic parallelism, explicit function calls, and separate compilation being among the novelties. OpenACC was designed to be interoperable with the host-level parallelism offered by OpenMP. The support for heterogeneous architectures and accelerators in OpenMP 4.0 roughly corresponds to the functionality provided by the initial release of OpenACC 1.0. In comparison to the previous version, OpenACC 2.0 offers enhanced functionality to write efficient code for state-of-the-art GPU architectures. However, OpenMP 4.0 explicitly strives to support a range of devices broader than only GPU-style architectures. For example, the Intel compiler already supports Intel Xeon Phi in a beta version and other implementations are expected that target SoCs consisting of ARM cores with DSP engines as well as FPGAs. Finally, the release of the Intel Xeon Phi in 2012 has popularized the idea of programming accelerators using host-style programming models such as Cilk, TBB, or native OpenMP and MPI.

Intensive research was also done in the area of domain-specific languages (DSLs), e.g. for stencil codes and graph algorithms. It was shown that higher-level source does not necessarily contradict performance and in fact can improve performance and portability – even in real applications.

Another critical development was the growing realization that Moore's law will likely come to an end between 2020 and 2025, as feature sizes beyond 5 nm seem neither physically nor economically feasible. This will have a profound impact on the entire HPC landscape, whose precise implications are hard to predict. With increased integration density no longer available to create added value, design may emerge as the primary vehicle for generating profit. The resulting diversification would cause a huge challenge for the HPC software industry. It is unclear, however, how many new designs the HPC market can absorb. At some more distant point in the future, hardware may stabilize and we may see a shift of focus and resources away from hardware to software. With hardware becoming more standardized, investments in HPC software would become more attractive. Either way, further fostering Europe's strong position in HPC software seems to be an advisable proposition.

## 5.4 Gap Analysis

While IDEs are being increasingly integrated into HPC environments, the adoption of other state-of-the-art software-engineering approaches such as object-oriented design, development frameworks, domain-specific languages, and standardized test suites have so far enjoyed only limited success in the HPC arena. In spite of the advances that have made accelerator programming easier and more sustainable, parallel programming still occurs at a low level of abstraction. Although still an active area of research (e.g., CAF 2.0), the adoption of PGAS languages is slow and their advantages have not yet been shown in a large population of codes. In this context, PGAS libraries such as OpenSHMEM can be seen as a more lightweight alternative to full-blown PGAS languages as a means to try PGAS concepts in selected portions of the code. Furthermore, while contemporary research focuses mostly

on the creation of new code, many grown commercial and community applications face the task of restructuring existing code (e.g., to improve load balancing), often a costly endeavor where decision rather than programming support would be useful. In general, the programming-centric roadmap of the past has diverted attention from other important aspects such as data structures as a means to raise the level of abstraction or load balancing methods. In general, we lack methods to support the high-level design and quality management of exascale applications that match their expected complexity. While powerful tools for error and performance analysis exist, they need to keep pace with new developments, expand their functionality to allow better insight, and extend their coverage to the (re-)design phase. Finally, the domain scientists who shape development practices rarely receive formal software engineering training – not to mention that software engineering curricula tailored to the specific needs of HPC barely exist.

In conclusion, we need to expand the currently mostly algorithm- and programming-centric view of HPC software development and achieve a better understanding of the (re-)design and quality management processes with the goal of providing appropriate methods and tools to support them.

## 5.5 Proposition of R&D program for 2015 and beyond

The proposed R&D program includes the following items:

- Methods and tools to support design, re-design, and co-design decisions and processes (e.g., performance engineering, re-factoring), especially in view of scalability, efficiency, and fault tolerance
- Higher-level programming environments including domain-specific languages and concepts to raise the degree of abstraction and improve performance at the level of data structures
- Efficient compiler and language support for domain-specific languages and optimized compilation of novel programming schemes (e.g., OpenACC, OpenMP 4.0)
- Scalable tools for correctness analysis and performance optimization
- Improved methods and tools for (automatic) program verification and validation
- Coupling and workflow technologies
- Continued integration of IDE technology into HPC environments
- Further studies on HPC development practices to reliably assess the state of the art

The anticipated funding level is €20M per year, ideally as a collection of STREPS, loosely coordinated, e.g., through joint workshops. Technologies such as programming languages that require a major commitment of application developers must be developed with strong community interaction, ideally standardization – at least as the ultimate goal – to reduce the risk and fear of lock-in. A coordination of the respective R&D research program with other international funding agencies and incentives to collaborate internationally beyond the EU borders will improve the prospects for success.

The overall impact will be much higher quality and lower complexity of codes and, as a consequence, significantly lower development costs. The latter should be seen as a prerequisite for the development of true exascale-enabled applications, a goal that may not be reached without either substantially increased funding and/or lower costs.

The WG recognizes the support that will be provided through the recent FET HPC call in H2020, but it is still to early to assess the degree to which the resulting projects will close the gaps outlined in this analysis.

## 5.6 Further recommendations

In addition to the above recommendation summary, the WG created a number of more detailed recommendations outlined in the section below.

**Co-designing** applications and the system is a powerful technique to ensure early and sustained productivity as well as good system design. In their early phases, such co-designs often rest on back-of-the-envelope (BOE) calculations. In general, such calculations allow problems in applications to be detected early on and their severity to be determined years before the machine is installed or the first prototype becomes available. On the system side, BOE calculations allow designers to adjust system

parameters to target applications, for example, they can be used to determine the required bytes-to-flop ratio of memory, network, or even the file system. However, even such BOE calculations are very time consuming and also error prone. Especially projecting applications requirements for large workloads still poses a challenge. Therefore, tools are needed to automate such projections, making the co-design process more reliable and efficient.

**Accelerators** are widely used in large-scale systems, i.e., the top 10 systems of TOP500. At smaller scale, systems without accelerators dominate. Most vendors have an accelerator-based roadmap towards exascale. Independent of the specific vendor implementations, accelerators share properties (sometimes known under vendor specific names) that are also envisioned for future general purpose CPUs (many cores, many threads with SMT, longer vector registers). Currently a broad range of programming models exists to program accelerators (CUDA, OpenCL, OpenACC, OpenMP 4.0). They all evolve at different speeds and are supported by different groups. While this is a normal situation for an evaluation phase of new hardware architectures, there clearly is a need for better support for programming accelerators in the long run. Otherwise the impact made at exascale level will not trickle down to smaller scale. We therefore recommend to strengthen the EU participation in relevant vendor-neutral standardization efforts (e.g., OpenCL, OpenMP). In addition research to provide abstraction levels independent of the specific programming method and hardware implementation should be supported.

**Domain Specific Languages** (DSL) targeting specific application areas (e.g., climate, engineering, materials) offer a high-level abstraction enabling application development to be protected and insulated from architectural issues at exascale. Development of applications using DSLs should take place using co-design principles with teams comprising science domain experts and computational technologists. With multi-disciplinary teams there is a great advantage in separation of concerns so that the science code can be developed separately from the computer science aspects. DSLs can do this, encapsulating the growing complexity of code required for multiple levels of parallelism and targeting a range of multi-petascale and exascale architectures.

**Software Development Processes (SDP)** heavily impact the quality of the developed software. As there are several challenges (e.g., developers are domain experts not software engineers, high developer fluctuation in scientific projects) the SDP needs to be customized in order to effectively support exascale development projects. Furthermore it is not known which important technical characteristics of exascale development projects should be supported by a dedicated exascale SDP. Hence, existing best process practices in developing exascale applications have to be collected and analysed to define a lightweight SDP framework. The process framework should be extensible and customizable to the specific needs of exascale development projects.

**Software Product Lines (SPL)** are sets of software systems for a particular market or domain developed from a common set of core assets in a prescribed way. In short, a SPL potentially decreases development time and cost, improves productivity, and increases the quality of software. Although SPLs have been successfully implemented in several domains, it is an open question whether existing product line techniques can be reused or adopted in the context of HPC and exascale computing. Hence, intensive research is needed regarding e.g., which technologies support the development of highly reusable components or which kind of software architectures are needed to build exascale applications on top of a reusable platform.

**Static program analysis** is a means to prove whether the behavior of a program matches its specification. **Software testing** is a dynamic analysis process to detect errors or to identify the difference between actual and expected result or behaviour. Behaviour may cover non-functional properties like performance and scalability, which are especially relevant for HPC. The overall goal is measuring software quality. There are challenges both for the static analysis and the dynamic test of highly scalable codes. While the low cost of static analysis make it extremely powerful, code complexity and the lack of runtime information make the static verification of code properties very hard. On the other hand, the infrastructure needed for dynamic debugging of high numbers of processes is very expensive. Effective software testing for highly parallel software can make software development more efficient, productive, and stable. Another impact is cost reduction through cheaper

regression tests, an advantage especially important for industry. Topics of future research should therefore include innovative verification strategies, advanced tool support for quality tests, and new test strategies for extremely parallel codes.

# 6. Working Group 4.3 “Disruptive technologies”

The objective of this working group is to tackle specific disruptive technologies in the field of software eco systems and numerical analysis. Finding disruptive technologies is not an easy task, since it is only known that a given technology is disruptive when it is already main stream. However, the WG has been able to devise some conclusions.

## 6.1 Members and their expertise

Due to the nature of the group, its formation has been slightly slower than others. Right now is composed by the following members:

| Name | Organization | Area of expertise |
|---|---|---|
| Iain Duff | STFC | Sparse Linear Algebra |
| Serge Gratton | ENSEEIHT | Optimization, data assimilation |
| Jesus Labarta | BSC | Programming models, Performance analysis, System software |
| Mike Giles | Oxford | CFD, MC, Financial Maths, GPUs |
| Hatem Ltaief | KAUST | Sparse Linear Algebra, GPU and heterogeneous |
| Rosa M Badia | BSC | Programming models, Heterogeneous programming, distributed computing |

## 6.2 Analysis

Disruptives technologies can be defined as:

“a new technology that unexpectedly displaces an established technology e. g, the digital camera, the telephone, CMOS technology, RISC instruction set, smart phones, …”

Harvard Prof. Clayton M. Christensen

A disruptive technology it is a revolution, and a revolution it is difficult to identify. A revolution runs over you, it is not planned and you even cannot notice that you are in a revolution. In this sense, it is very difficult to predict what will be a disruptive technology, since it is identified when it is already established.

## 6.3 Identified disruptions and technologies that enable to cope with them

In deliverable 4.1 a set of disruptions and technologies that originate disruptions were identified. The updated list of identified disruptions throughout the project is:

1. Variability in a dynamic world: in resource performance, resource availability
2. Abstraction: from low level device features to high level specification
3. Asynchrony
4. Bottleneck shift from computing to data transfer
5. Imprecise computations
6. Power constraints

With regard technologies that originate disruptions, the following were identified:

- Hardware technologies
  - Memories: PCM, memristor, NVRAMs
  - Packaging
    - 2.5D
    - 3D stacking
    - Optic communication between devices
  - Multicores, manycores, accelerators & SoC
  - Storage
    - SSDS
- Software technologies
  - Virtualization
  - No SQL: Key-value storage
- New Paradigms
  - Quantum Computing
  - Bio-inspired

## 6.4 Reacting technologies

To cope with these disruptions and new technologies, the WG has identified some reacting technologies, which appear as a result to them. The list of reacting technologies is the following:

- Load balancing
- Asynchronous programming models and system software
- Communication reducing, communication hiding, synchronization reducing algorithms
- Power aware schedulers
- Mixed precision computation, low rank compression
- Hybrid algorithms and solvers
- Stochastic PDEs
- New techniques: tensor calculus, novel algebras, stochastic programming
- New algorithms: Chaotic relaxation, contour integration, Monte-Carlo techniques, vectorization
- Hierarchy (i.e., nesting) – it is a need for new algorithms: Fast multipole methods, dense eigenvalue, H-matrices
- Big data

**Variability:**

In this case the disruption is the change from a situation where computers have stable and fixed resources to a situation with variable, unstable, non guaranteed, dynamic resources. This situation is given for multiple aspects, such as turboboost technologies, manufacturing variability, interactive dynamic load in environments, etc. Since this situation is not going to change, it requires a change of mind with regard keeping the control on the execution in the hardware by the application developers.

Technologies that can help coping with variability are, to mention a few:

- System level mechanisms that support auto-management, dynamic adaptation
- Malleable programming models
- Runtimes and resource management that support this variability, as dynamic load balancing (Real-time techniques)
- Autotuning
- Malleable job schedulers

**Abstraction:**

Abstration refers to a change from a low level of abstraction, close to the HW to high levels of abstraction, far from the HW. As with the previous disruptions, it also requires a change of mind with regard how applications are written, focus in the algorithm, not in the specific hardware, and forget about trying to control how the application behaves at the low level.

Technologies that can help to coping with variability are:

- Programming models that enable to separate in the applications the algorithmic part from the specificities of the resources (OmpSs, OpenACC, …)
- Rapid prototyping (eDSLs, Perl, Python, …)

**Asynchrony:**

The disruption identified here is the change from a synchronous execution model (fork – join) to a asynchronous execution model (data-flow, task-based + dependencies).

Technologies that can help to cope with asynchrony are:

- Runtimes that support this asynchrony
- Accept lack of direct control by programmer
- Programming models such as OmpSs
- APGAS
- non-blocking communications in MPI

**Bottleneck shift from computing to data movement:**

The disruptive change in this case is the change from limitations in computation to limitations in data movement.

Technologies that can help to cope with this are:

- Communication avoiding algorithms
- Overlap communication with computation
- Move computation instead of data
- 3D stacking (this maybe even can avoid the problem?)

**Imprecise computations**

The disruption here is the change from exact computations to approximate computations. This can be due to new schemes in numerical methods or also a way of surviving failures.

Technologies that can be applied to cope with this are:

- Uncertainty quantification
- Ensemble prediction
- Fuzzy computation
- Mixed precision

**Power constraints:**

A big disruption in this case is the change from the absence of power constraints in HPC computing to strong power/power-density and power cooling constraints in order to be able to build future exascale supercomputers.

Technologies that can be applied in this case:

- Use of mobile devices to build supercomputers
- Dynamic scheduling of non-priority tasks into low-power processors

**Hierarchy:** In this case, it is mean how to express new algorithms (Fast multipole methods, dense eigenvalue, H-matrices) with ***hierarchy***. The concept of hierarchy enables the algorithm to be organized in different levels, or even recursively. For example, this can be done with some programming models with task nesting (OpenMP tasks or OmpSs). The hierarchical implementations will improve aspects such as synchronization and communication (vertical and horizontal), data reuse, resiliency (local failures), power, etc. One good and concrete example in this area is the H-matrices, which are a compressed format of dense matrices using low rank representations. Not all matrices can be H-matricized but many from engineering applications can.

**Big Data**

Big Data has emerged in the recent years as the term that describes the processes required to do analysis and analytics of large and complex data sets, but also the different processes to aquire, curate, store, ... the data. HPC and BD have been seen as different topics in IT, but convergence between them is foreseen. BD technologies can largely contribute to improve post-processing of data typically produced by HPC simulations, for example. The convergence of HPC and BD has been the topic of a series of workshops co-organized by EESI2.

# 7. Working Group 4.4 “Hardware and Software Vendors”

The main objective of this working group was to investigate the state of the art and trends in the HPC hardware and software industry. This task was accomplished mainly from two different angles: by leveraging a network of distinguished HPC experts who agreed to share their insights into the HPC industry and by performing an extensive market and technology watch.

A group of 13 experts, mostly from the HPC hardware industry, have agreed to contribute to this working group:

| Expert | Email | Organization | Position |
|---|---|---|---|
| David Lecomber | david@allinea.com | Allinea | CTO and founder |
| Chris Adeniyi-Jones | Chris.Adeniyi-Jones@arm.com | ARM | R&D engineer |
| Jean-Pierre Panziera | Jean-Pierre.Panziera@bull.net | Bull | Director of Performance Engineering |
| Alex Ramirez | aramirez@ac.upc.edu | BSC | Computer Architecture Research Manager |
| Francois Bodin | Francois.Bodin@caps-entreprise.com | Caps Entreprise | CTO |
| Giampetro Tecchiolli | giampietro.tecchiolli@eurotech.com | Eurotech | CEO/CTO |
| Ulrich Brüning | ulrich.bruening@ziti.uni-heidelberg.de | EXToLL | Director |
| Luigi Brochard | luigi.brochard@fr.ibm.com | IBM | Distinguished Engineer |
| Karl Solchenbach | karl.solchenbach@intel.com | Intel | Director Exascale Labs Europe |
| Axel Koehler | akoehler@nvidia.com | nVIDIA | Senior Solution Architect HPC |
| Matthias Müller | mueller@rz.rwth-aachen.de | RWTH Aachen | Head of Compute Center |
| Kai Dupke | kdupke@suse.com | SUSE | Senior Product Manager Server |
| Malcom Muggeridge | Malcolm_Muggeridge@xyratex.com | Xyratex | VP of Emerging Technologies |

During a meeting at the Leibniz Supercomputing Centre on April 17th 2013 and follow-up discussions via email and teleconference meetings, the experts identified a list of challenges that they felt had to

be tackled on the way to Exascale. The following section only gives a brief overview of the most important challenges, a more detailed description can be found in Deliverable D4.1.

## 7.1 Key challenges

### 7.1.1 Hardware: Energy efficiency, Power Wall, Power Density

The energy requirements of an Exascale system will be prohibitive if they don't improve significantly over currently available technology. This applies to all system components, including CPUs, memory, storage, interconnects, power supplies, and cooling infrastructure.

### 7.1.2 Hardware: Memory/Storage Capacity, Packaging, Bandwidth

Exascale computing also means Exascale data – in main memory as well as on background storage. Although main memory capacities will continue to grow reasonably thanks to Moore's law, the bandwidth to the processor will not. Without additional R&D effort, insufficient memory bandwidth will yield humble application. Similarly, the performance of the storage system will become a bottleneck and limit the overall system performance if it does not keep pace with compute nodes.

### 7.1.3 Hardware/Software: Reliability/Resilience/Fault Tolerance

Compared to today's systems, the number of components in an Exascale system will grow by several orders of magnitude which will increase the failure rate of the overall system. Hence failures of individual components will have to be anticipated and accommodated at the hardware, system software, and application level.

### 7.1.4 Software: Inter-/Intra-Node Scalability to 1M Tasks, also Tools & Debuggers

As future Exascale systems will be comprised of ten thousands of nodes, each with hundreds of (heterogeneous) cores, applications on such systems will be required to scale to over 1 million tasks in order to exploit the potential performance of the machine. Currently, there are not many algorithms (and applications) available that are able to scale to such high levels of parallelism. Additionally, currently available development tools and debuggers cannot scale to such high levels of parallelism neither but will be essential for the development of highly scalable applications.

### 7.1.5 Software: Programmability & Programming Environments

The high level of parallelism and likely heterogeneity of future Exascale systems will not only require scalable algorithms, but also programming paradigms and environments that support the efficient implementation of these on the actual hardware. For many users, the currently prevailing paradigms like MPI, OpenMP, and OpenCL are (and will be) inapt for mapping their algorithms to hardware and hence will not allow to yield satisfactory performance on Exascale machines.

### 7.1.6 Software: Characteristic Mini-Apps / Benchmarks

Any development related to high performance computing, whether at the hardware or the software level, will ultimately need to be applied to or used by actual applications. A set of characteristic mini-apps that only implement integral components of actual applications and are easy to use, could help to drive and improve such developments.

# 7.2 Market and Technology Watch

## 7.2.1 Hardware: Energy efficiency, Power Wall, Power Density

The power consumption of HPC systems is largely dominated by CPUs and GPUs, and to a lesser extent by memory, network, and storage. As Moore's law still holds thanks to ever finer semiconductor manufacturing processes (14nm being the state-of-the-art in 2014 and 10nm being predicted for 2016), future microprocessors automatically become more energy efficient. As leakage currents fall with smaller transistor sizes, they deliver more compute performance within the same power envelope of their predecessors. However, as it becomes harder and harder to put these additional transistors to good use, the degree of parallelism within a single CPU or GPU increases steadily. They feature more cores and employ ever wider instruction sets that require parallel code even at the instruction level. Consequently, it becomes more and more difficult for programmers to exploit these systems efficiently.

More and more system integrators offer not only air-cooled systems but also direct-liquid cooled or immersion cooled solutions. As these approaches allow for much more efficient waste heat removal, they facility high density form factors and therefore increase the power density of HPC systems.

## 7.2.2 Hardware: CPUs, GPUs, and Accelerators

For HPC systems, the most established providers for CPUs are Intel, IBM, and AMD. As IBM sold its x86 business to Lenovo, they concentrate on its Power architecture in terms of HPC and also founded the OpenPOWER foundation that will develop the Power architecture in the future. ARM started moving from the mobile market to the server market and recently introduced its 64bit architecture ARMv8 that seems to be suitable for HPC as well: first OEMs like AMD, AppliedMicro, and Cavium announced ARMv8-based server SOCs that feature wide memory buses, high-bandwidth interconnects, and 3$^{rd}$ generation PCIExpress for extension cards like Infiniband. AMD also offers HPC-grade GPUs and is the main contender of NVIDIA in this market. Besides its Xeon line of server CPUs, Intel also offers and further develops its accelerator product line Xeon Phi.

The CPU roadmaps of the established vendors look very similar in terms of technology: the number of cores keeps on increasing, DDR4 becomes the standard memory interface, the cache hierarchies become deeper, the instruction sets wider. Occasionally, there will be small on-CPU RAM for higher memory bandwidth.

The coupling between CPUs and GPUs becomes closer, with faster busses, uniform memory access, and cache-coherency: technologies like AMD's Heterogeneous System Architecture (HSA) and OpenPOWER's Coherent Accelerator Processor Interface (CAPI) may have the potential to simplify the programming models for heterogeneous architectures by removing the need for explicit memory transfers between the host CPU and accelerators. However, Intel has announced no such technology for their Xeon and Xeon Phi processors.

As the coupling of CPUs and accelerators becomes tighter, the next logical step seems to be moving the network interconnect closer to the CPU as well. All major CPU vendors also have high-performance network technology available in their portfolio: Intel has Omni-Path (derived from Cray's Aries interconnect), AMD the SeaMicro Freedom Fabric, and the OpenPower consortium has Mellanox with its Infiniband technology as a member. In the long term, the interconnect may very well move to the CPU die, allowing for higher bandwidth and lower latency.

## 7.2.3 Hardware: Memory/Storage capacity, packaging, bandwidth

The advancement due to Moore's law also applies to DRAM technology. Thanks to shrinking manufacturing processes next generation DRAM will operate at higher frequencies and lower voltages which eventually translates to higher bandwidth and lower energy consumption. New technologies like 3D-stacked memory where multiple layers of DRAM are stacked and connected using through-silicon vias, allow for denser packaging. Additionally, developments like High-Bandwidth Memory (HBM) that put stacked DRAM on the same package as the CPU will eventually also allow for significantly higher memory bandwidth and closer coupling of memory and CPU. New protocols like DDR4 and Hybrid

Memory Cube (HMC) also contribute towards higher memory bandwidths. Upcoming CPU generations will provide memory bandwidths in the order of 250 GB/s which will help to keep the balance between CPU and memory performance at least at current levels.

In terms of storage, spinning magnetic disks remain the “work horses” for data-intensive applications. However, the capacity and density growth we have seen in the previous decade seems to have come to a stop for a couple of years now. Where bandwidth and latency are important, non-volatile memory technologies like NAND flash already provide the better price/performance ratio and continue to close in on harddisks in terms of price per GB. New NVRAM technologies like Magnetoresistive RAM (MRAM) and Phase-change RAM (PRAM) have supposedly been developed by multiple vendors but no actual products have been announced yet. The same holds true for storage based on memristor technology.

### 7.2.4 Hardware: Reliability/Resilience

As the number of individual components in current and future HPC systems continues to grow, the failure rates of these systems as a whole will increase dramatically. Therefore, total breakdowns and transient errors of each single component will have to be anticipated and accommodated at the hardware level. While everybody seems to talk about reliability and resilience features, actual products that implement such on the hardware level seem still to be missing and do not appear on the roadmaps of the mainstream hardware vendors. Although being recognized as an important topic, research in this field only appears to be taking place in the academic sphere.

## 7.3 Gap Analysis

### 7.3.1 Hardware

There is no obvious gap in the roadmaps of the hardware vendors that could potentially be a showstopper on the way to Exascale. One can always build a larger system with more nodes, more cores, and more accelerators that will at least on paper will provide a peak performance of 1 ExaFlop/s. Assuming that Moore’s Law will continue to hold and extrapolating from the current Top500 HPC systems, a ExaFlop system in the 20MW power envelope seems feasible by 2020. However, this will all depend on the further refinement of semiconductor manufacturing processes and the ability of hardware developers to put the additional transistors to good use. Although the coupling of individual components in HPC systems seems to be become tighter (memory and accelerators moving closer to the CPU) the unprecedented degree of parallelism of future HPC systems will pose tremendous challenges on application developers. Additionally, it is not clear yet how potential reliability and resiliency problems of such massively parallel systems will be tackled. As none of the mainstream hardware vendors seems to be willing to provide hardware solutions to deal with these, it is likely that dealing with lacking hardware reliability will be another burden for the software developers.

# 8. Conclusions

This document is the final report of the EESI2 WP4 Enabling Technologies. The WP is organized in four WGs: Numerical analysis; Scientific software engineering, software eco-system and programmability; Disruptive technologies; and Hardware and operating software vendors.

The deliverable presents a summary of the findings of the different WGs during the project lifetime.

Between the aspects highlighted by the experts from the numerical analysis WG we find for example the fact that the number of useful calculations per memory access should be maximised since the memory accesses are increasingly becoming the bottleneck in computations algorithms; the need for synchronization avoidance, or asynchronous versions of algorithms, should be investigated to avoid load imbalance issues; and dynamic scheduling techniques based on task-based programming models and runtimes that can better exploit the concurrency in computations. The experts also highlight the increasing activity in research in Big Data and data sciences and its application to different fields of research of the WG.

The software engineering WG highlights the challenges that the Exascale architectures will expose due to the fact that millions of cores computation is relatively cheap while memory access and communication are becoming increasingly expensive. Though, higher-level programming environments including domain-specific languages that raise the degree of abstraction and improve performance are required. In this same direction, efficient compiler and language support for heterogeneous platforms and optimized compilation of novel programming schemes are needed. This group specially highlights the need for new predictive tools, debuggers and support tools that provide help and guidance to application developers when facing the challenges of the Exascale architectures. What is more, it is necessary to define a light software development process framework to standardize how scientific applications for HPC are developed.

The disruptive technologies WG also highlights the need for new programming models that offer a higher level of abstraction and runtimes that support asynchronous behaviour, eliminating global synchronizations and reducing load imbalance. This asynchronous behaviour will also provide means to better support the variability exposed by current and forthcoming architectures. New techniques at all levels than enable the avoidance or hiding of communication are required, since transferring the data is becoming more expensive, both in terms of time and power than computing it. The concept of hierarchy in the applications is considered as highly important, and matches the hierarchy offered by task nesting in task-based programming models and the hierarchical organization of Exascale systems. New data structures like the H-matrices, which are a compressed format of dense matrices using low rank representations, and the algorithms that operate on them are a good example of the type of algorithms that follow this scheme.

The hardware and software vendors WG observes that now revolution is observed in terms of hardware, and that the improvements in power and performance will be obtained thanks to new manufacturing processes. Also, no new technology that has the potential to become a game-changer is seen in the near future. The group considers that the vendors are not giving enough importance to the hardware reliability and resiliency, although the academic research highlights that these topics need to be addressed.

Additionally to the findings reported in this document, the WGs have been actively contributed to the project recommendation, especially to the ones from the tools and programming models pillar as well as to the BDEC meetings and activities.