D4.1 First intermediate report on enabling Technologies

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

Due date of deliverable: 30 July 2013

Actual submission date: 30 July 2013

Publication date:

Start date of project: 1 September 2012 Duration: 30 months

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

Name of reviewers for this deliverable: Rosa Maria Badia, Herbert Huber, Andreas Grothey, Mathias Muller, Felix Wolf,

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: draft

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

Glossary

Abbreviation / acronym	Description
DAG	Direct Acyclic Graph
EMWG	Exascale Mathematics Working Group
FMM	Fast Multipole Method
Table text	Table text
….	….

1. Executive Summary

This document is the first EESI2 intermediate report on enabling technologies, corresponding to WP4

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 this core numerical algorithms is crucial and essential if we want to realise the potential of future Exascale systems.

The area is broken down in the report into the following subtopics: Dense linear algebra, Graph partitioning, Sparse direct methods, Iterative methods, Eigenvalue problems, Optimization & control, Structured & unstructured grids and Monte Carlo. Due to their increase importance the topics of Tensors and Fast Multipole Methods as separate items they have been added to the list as well.

As memory accesses are increasingly the bottleneck in computations, algorithms need to maximise the number of useful calculations per memory access. A way of tackling this is by blocking/tiling and communication hiding. 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 Also, to reduce synchronization and load-balancing overheads, computations need to be expressed at multiple levels of abstraction as task graphs and make use of data-driven schedulers.

Future challenges include the development of new scalable algorithms based on modular frameworks that support alternative scheduling methods, memory affinity schemes, load balancing methods, etc. Communication avoidance and fault tolerance will be increasingly important as we approach the exascale level. The last couple of years have seen incremental improvements in both of these areas but the real challenges to reach scalable exascale computing remain.

Some categories of methods face the challenge of adapting to matrix structures arising from new applications such as big data processing but also the increasing exploitation of low rank approximations and compression methods. Others are tackling their challenges with techniques based on irregular octrees, for example, although these data structures face irregular computation patterns which have load-balancing issues.

Asynchronous/chaotic relaxation methods offer much scope for parallelisation but give rise to stability issues that are poorly understood. Finally, auto-tuning it is becoming essential for almost all applications.

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.

During the last year, most popular programming interfaces (MPI and OpenMP) have presented substancial extensions. For MPI, new features such as non-blocking collectives that enable overlap of communication and computation or neighbourhood collectives, which contribute towards better performance on exascale platforms. With regard programming environments, the WG concludes that there is a lack of methods that support the high-level design and quality management of exascale applications that match their expected complexity. Tools for error and performance analysis need to keep pace with new developments, expand their functionality to allow better insight, and extend their coverage to the (re-)design phase. Also, 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.

The WG concludes that there is a 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.

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.

Relevant disruptions identified so far are the variability in the resources (resource performance, resource availability), a change in the level of abstraction in the way that applications are written (far away from the HW details) or a change in the execution model, from synchronous to asynchronous.

Technologies that can help on surviving this disruptions are, for example, system level mechanisms that support auto-management, dynamic adaptation or malleable programming models with runtimes and resource management that support this variability, as dynamic load balancing; programming models that enable to separate in the applications the algorithmic part from the specificities of the resources (OmpSs, OpenACC, …) or rapid prototyping programming environments (eDSLs, Perl, Python, …); and runtimes that support asynchrony (as task-based, data-flow schedulers or non-blocking MPI communications).

The WG has also identified technologies that themselves can cause disruptions, such as new memory or packaging technologies from the hardware side, or virtualization from the software side.

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 WG identifies hardware challenges in the energy efficiency area for all components of the system, in the bandwidth requirements of the memory system, increased requirements in resilience, and in the routing of interconnection networks, betweem others.

With regard software, challenges are identified in areas such as scalability to large number of tasks, programming models and tools, techniques for checkpoint and restart and fault tolerance in general, maintenance of legacy codes, in the development of mini-apps and benchmarks and in virtualization.

Additionally, the WG is aligned with ETP4HPC conclusions and plan to work together in order to optimize resources.

Recommendations

1. To establish a single worldwide initiative for the study of numerical algorithms for exascale computing

2. To follow up on new paradigms for parallel programming that are able to support the requirements of exascale applications

3. To follow up on the evolution of memory technologies

4. To align and merge efforts with ETP4HPC, organizing a common meeting for example

2. WG 4.1 Numerical Algorithms

2.1 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 this 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.

2.2 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

2.3 Management of the Working Group

Information presented in this report has been gathered through interaction with experts at a working group meeting in Edinburgh on 8 April and by email. Experts have been asked to:

1. Update the important developments in their respective area(s) of expertise since the final EESI-1 report
2. Provide a brief (max 1/2 page) gap analysis of the key challenges in their area, where we are now, where we have been a year ago and what still needs to be done

In producing this report, the members of the working group consulted widely with colleagues and would like to record their thanks to: Peter Richtarik (Edinburgh).

2.4 Key challenges and gap analysis

The final report of the working group 4.3 “Numerical Libraries, Solvers & Algorithm” of EESI-1, provides a fairly detailed discussion of the state-of-the-art and challenges to achieve exascale performance within the remit of the current working group. Although now two years old, most of the discussion is 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. It is time to give these algorithms 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 (matrixvector 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 3D and higher dimensional 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.

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. 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.

2.4.1 Key challenges

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. Communication avoidance and fault tolerance will be increasingly important as we approach the exascale level. The last couple of years have seen incremental improvements in both of these areas but the real challenges to reach scalable exascale computing remain.

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 rely 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. Currently, most tensor applications arising from data driven modelling are written in Matlab. 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 hidde the communication is also a very active field. There is a starting FP7 STREP-Exascale project that will address this issue. Otherwise effort should concentrate on matrix-vector multiplications since they are now limiting speed-up. 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. 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 Krylovmethods will increasingly be employed for multiple right hand sides. This offers scope for parallelism that has not been exploited yet.

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. The methods will probably be mature in five years. 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 Nbody 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 new chip designs such as ARM are making increasing inroads even for high performance computing (for example in the EU-funded Mont Blanc project at Barcelona). Active progress is being made on the hardware and operating system software, but if ARM is to be successful in HPC, it is vital that there are high quality, high performance numerical libraries available for its processors. Europe has various centres (like the PRACE centres in Barcelona, J¨ulich, CINECA, Edinburgh, or a not-for-profit supplier such as NAG) that would be well suited for this activity.

2.4.2 Recommendation

The Department of Energy (DOE) Office of Science Program on Advanced Scientific Computing Research has formed an Exascale Mathematics Working Group (EMWG) for the purpose of identifying mathematics and algorithms research opportunities that will enable scientific applications to harness the potential of exascale computing[1].

The objectives of this working group are aligned with the objectives of the EESI2 WG4.1 and for this reason we recommend the EC to support a unique initiative on mathematics at worldwide level that enable these and other similar initiatives to work together.

[1] https://collab.mcs.anl.gov/display/examath/Exascale+Mathematics+Home

3. 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.

3.1 Members and their expertise

Name	Organization	Area of expertise
Felix Wolf (chair)	German Research School for Simulation Sciences, 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
Vincent Bergeaud	CEA, France	Software engineering in scientific simulation, parallel programming and uncertainty analysis
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

3.2 Management of the group

The group interacts via phone conferences and physical meetings. So far, the group has had three phone calls and two physical meetings, including the one in Le Tremblay. A wiki is used to collect ideas and relevant literature. An email list ensures efficient group communication.

3.3 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.

3.4 Developments of the past year

During the past year, 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. Moreover, two release candidates of OpenMP 4.0 were published and the final version 4.0 is expected to be 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.

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.

3.5 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.

3.6 Proposition of R&D program for 2014 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.

3.7 Further recommendations

In addition to pure R&D measures, the EU must also foster the inclusion of software engineering in HPC-related training, for example, in the form of PRACE training schools. Given the breadth and depth of the subject, the integration into HPC-related university curricula would be highly desirable.

Since advances in software engineering for exascale will require a change of culture, away from the picture of a domain split into the fields domain science, numerics, programming, and systems towards a more integrated view that also encompasses people and processes, community-building measures similar in spirit to ESPRIT working groups or networks of excellence will be a key to success. To leverage lessons learned in other domains, it will be of particular importance to connect the HPC community with the general software engineering community.

An ideal instrument to make progress towards the technical milestones listed above would be a center of excellence where software engineers would closely interact with application groups. To ensure close collaboration, it might be worth considering the physical integration of application experts into software engineering groups and vice versa. The participation of vendors will ensure harmony between hard- and software. If critical mass is reached, the center should also plan and carry out educational measures.

Finally, interdisciplinary support structures such as HECToR Distributed CSE Support (UK), Platform for Advanced Scientific Computing (Switzerland), and Simulation Laboratories (Germany) need to be expanded and further developed based on an evaluation of existing models and experiences from the past.

As next steps, the WG should follow up with an special focus on programming models and runtime systems for the second period of the project.

4. 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.

4.1 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

4.2 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.

On the other hand, revolutions can be very slow to be adopted, maybe not at macroscopic scale, but it can take more than 10 years to be adopted. An example of this is the adoption of virtual memory against overlay programming; it took around 10 years, since people wanted to control what was in memory and load every region under demand. We are in a similar situation with local memories, GPUs, …

From our point of view it is easier to identify the concepts (disruptions). In this sense, a disruption is a change in philosophy, practice, culture…

Once the disruptions have been identified, we can find out the technologies that allow to bear with the existing disruptions. Alternatively, there are new technologies that can originate disruptions.

4.3 Identified disruptions and technologies that enable to cope with them

Following the previous analysis, the group has identified the following disruptions so far:

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
7. …

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

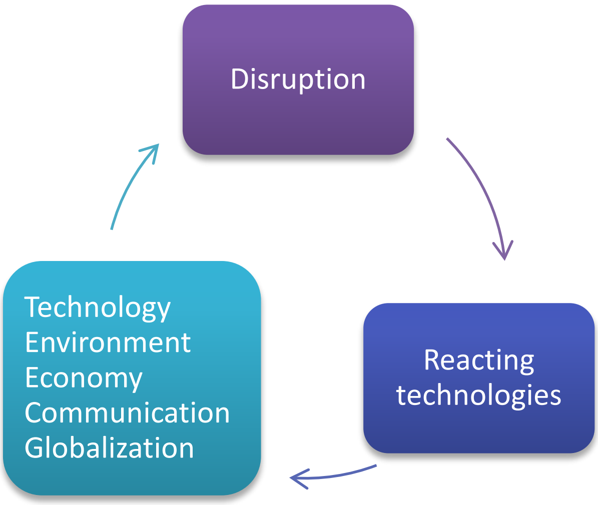
4.4 New technologies that originate disruptions

The group has been able to identify the following technologies that originate disruptions:

- Hardware techonogies
  - 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

4.5 Conclusions

The results of this WG are very preliminary, but had proposed a methodology for structuring disruptive technologies. A disruption is a radical change and as a results of such a change a set of reacting technologies appear to alleviate the situation and enable to cope with the change. At the same time, these new technologies may imply a change in aspects not only of technology, but also of environment, economy, etc, that may result in a new disruption.



5. Working Group 4.4 “Hardware and Software Vendors”

One of the main objectives of this working group is to establish and maintain 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. A group of 13 experts, mostly from the HPC hardware industry, have agreed to contribute to this working group and to share their insights into the industries R&D roadmaps:

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 need to be tackled on the way to Exascale – from a hardware point of view as well as from a software point of view:

5.1 Hardware challenges

5.1.1 Energy efficiency, Power Wall, Power Density

The energy requirements of an Exascale system will be prohibitive (economically as well as ecologically) if they do not improve significantly over currently available technology. This applies to all system components, including CPUs, memory, storage, interconnects, power supplies, and cooling infrastructure.

5.1.2 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 performance and hence limit the sustained performance of a future Exaflops system to well below 100 Petaflops. 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.

5.1.3 Reliability/Resilience

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 level.

5.1.4 Data-processing closer to data

Data movement is very costly: it uses memory and storage bandwidth which is limited (see above) and also consumes energy. Even in today's systems, the actual computation performed on data often requires less energy than moving it from the main memory to the processor and back. This is particularly true if the data originates from background storage or network. Hence, more data processing needs to be performed closer to the data, i.e., in memory, storage, or network.

5.1.5 Efficient use of additional transistors (dark silicon)

As Moore's law still holds, the number of transistors available in any chip will continue to double every 18 month. Using these additional transistors wisely will be key to increase each components performance and energy efficiency. Just increasing the number of functional units, cores, or caches on an CPU will not be sufficient.

5.1.6 Multi-level (energy-proportional) interconnection networks

Due to the expected increase in the number of components of future Exascale systems, there will be multiple networks at different levels: within each chip, package, and node, and between nodes. These complex networks will require smart routing and will have to be energy-proportional, i.e., consume energy only when actually transferring data.

5.2 Software challenges

5.2.1 Inter-/Intra-Node scalability to 1M tasks, also Tools & Debuggers

Given that future Exascale systems are predicted to 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. Hence, research into more scalable algorithms and tools is urgently needed to put future Exascale systems to use.

5.2.2 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. New programming paradigms will however have to guarantee a long-term perspective for the developers in order to be adopted.

5.2.3 Data locality, Data movement avoidance

As data movement is very costly in terms of energy and since memory bandwidth already today is a bottleneck that limits application performance in many cases, data locality and the avoidance of data movement will be key to obtain good application performance in the future. This will require research in algorithms that exploit data locality and avoid data movement by, e.g., re-computing interim values instead of moving them to/from memory.

5.2.4 Checkpoint/Restart, Fault tolerance

As a higher failure rate of individual hardware components is expected (see above), system software and applications will be required to accommodate such failures that cannot be contained at the hardware level. Therefore, additional research into concepts for tolerating failures at the application level or for quick checkpoint and restart will be needed.

5.2.5 Standardization of APIs, libraries

In the Exascale regime, highly optimized algorithms will be even more important to achieve good application performance than today. Hence, application developers should be able rely on properly defined APIs and efficiently implemented libraries and only develop code by themselves that does not exist already.

5.2.6 Legacy codes in C/C++/Fortran

Many scientific applications that are currently used on supercomputers have been developed over decades, are mostly written in C/C++/Fortran, and utilize MPI and/or OpenMP to exploit parallelism. While new programming paradigms will be key for the efficient exploitation of Exascale systems (see above), concepts will also have to be devised to bring such applications to the Exascale era.

5.2.7 Resource scheduling

The enormous number of components in future Exascale systems will also require additional research and development in resource scheduling mechanisms to ensure high utilization of all components at all times.

5.2.8 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.

5.2.9 Virtualization

Virtualization is already being used intensively in server environments and could be similarly beneficial to high performance computing, e.g., for better resource utilization, checkpoint/restart, performance optimization, etc. However, this will require additional effort to reduce the overhead induced by currently available virtualization techniques.

5.3 ETP4HPC

The challenges listed above are also in line with the findings of the European Technology Platform for High Performance Computing (ETP4HPC). Their Strategic Research Agenda lists four fields they consider to be crucial for the further development of HPC. Two of them, “HPC Stack Elements” and “Extreme Scale Requirements”, overlap with the scope of this working group. Within these two fields, they identified “HPC System Architecture”, “System Software and Management”, “Programming Environments”, “Energy Efficiency and System Resiliency”, and “Balanced Compute Subsystem, I/O and Storage Performance” as urgent fields for R&D efforts in HPC. Although named differently, the content of these topics largely matches the challenges identified by this working group. Additionally, the ETP4HPC lists “New HPC Deployments” and “HPC Usage Expansion”. These fields mainly cover the usability of and new use cases for HPC systems and were not within the scope of this working group.

5.4 Recommendations

The WG will follow up on the evolution of new memory technologies that can largely increase the available bandwidth due to its potential large impact.

Also, the WG will strength its relation with ETP4HPC, for example by setting up a common meeting, in order to align positions and optimize resources.

6. Conclusions

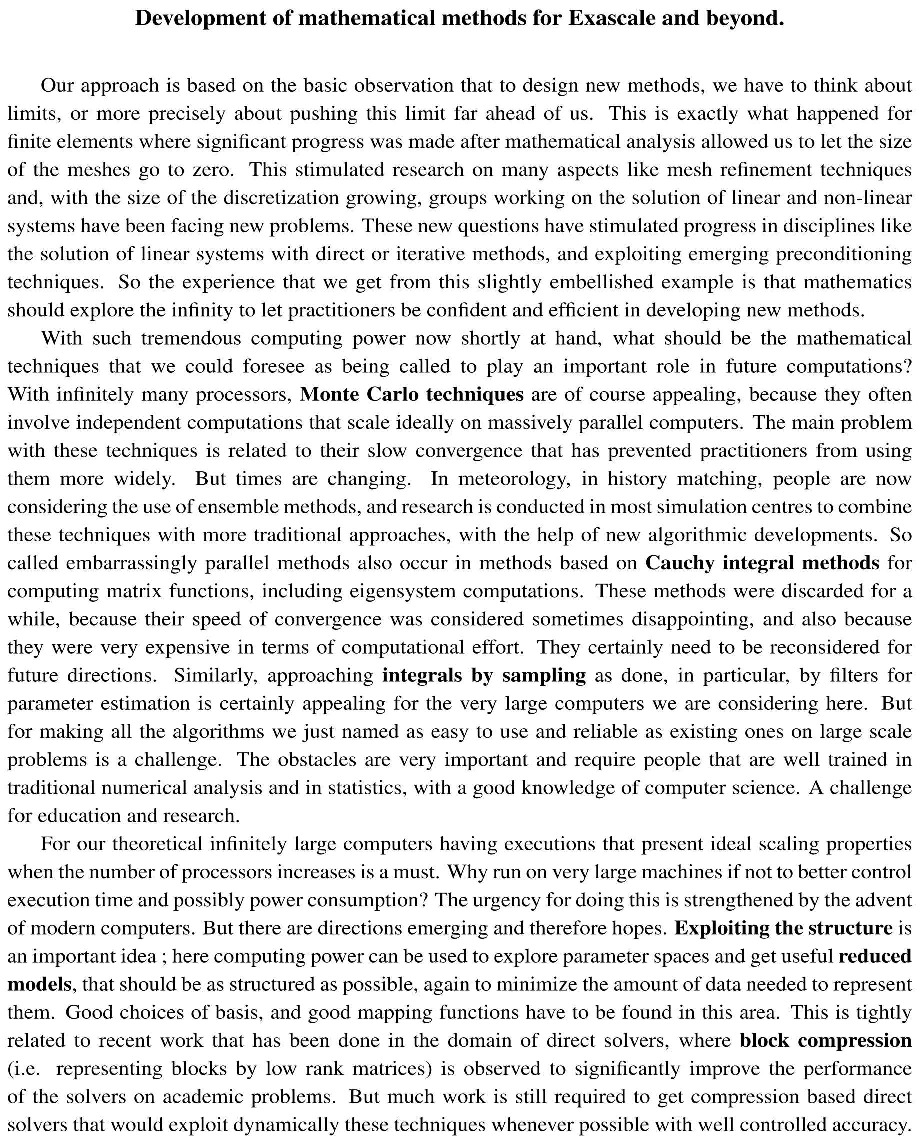
This document is the first intermediate report of the EESI2 WP4 Enabling Technologies. The WP is organized in four WGs: Numerical analysis; Scientific software engineering, software eco-system and programmability; Disruptive technologies; and Hardware and operating software vendors.

The different WGs have performed

For each of the topics, the WP have identified the corresponding experts and organized the corresponding meetings. In each working group, the current practice in the areas have been reviewed and evaluated. The corresponding gap analysis has been performed and first recommendations derived.

7. Annex I: Contribution to DOE EMWG workshop

A contribution to a DOE EMWG workshop can be found attached to this document.



Development of mathematical methods for Exascale and beyond.

Our approach is based on the basic observation that to design new methods, we have to think about limits, or more precisely about pushing this limit far ahead of us. This is exactly what happened for finite elements where significant progress was made after mathematical analysis allowed us to let the size of the meshes go to zero. This stimulated research on many aspects like mesh refinement techniques and, with the size of the discretization growing, groups working on the solution of linear and non-linear systems have been facing new problems. These new questions have stimulated progress in disciplines like the solution of linear systems with direct or iterative methods, and exploiting emerging preconditioning techniques. So the experience that we get from this slightly embellished example is that mathematics should explore the infinity to let practitioners be confident and efficient in developing new methods.

With such tremendous computing power now shortly at hand, what should be the mathematical techniques that we could foresee as being called to play an important role in future computations? With infinitely many processors, **Monte Carlo techniques** are of course appealing, because they often involve independent computations that scale ideally on massively parallel computers. The main problem with these techniques is related to their slow convergence that has prevented practitioners from using them more widely. But times are changing. In meteorology, in history matching, people are now considering the use of ensemble methods, and research is conducted in most simulation centres to combine these techniques with more traditional approaches, with the help of new algorithmic developments. So called embarrassingly parallel methods also occur in methods based on **Cauchy integral methods** for computing matrix functions, including eigensystem computations. These methods were discarded for a while, because their speed of convergence was considered sometimes disappointing, and also because they were very expensive in terms of computational effort. They certainly need to be reconsidered for future directions. Similarly, approaching **integrals by sampling** as done, in particular, by filters for parameter estimation is certainly appealing for the very large computers we are considering here. But for making all the algorithms we just named as easy to use and reliable as existing ones on large scale problems is a challenge. The obstacles are very important and require people that are well trained in traditional numerical analysis and in statistics, with a good knowledge of computer science. A challenge for education and research.

For our theoretical infinitely large computers having executions that present ideal scaling properties when the number of processors increases is a must. Why run on very large machines if not to better control execution time and possibly power consumption? The urgency for doing this is strengthened by the advent of modern computers. But there are directions emerging and therefore hopes. **Exploiting the structure** is an important idea ; here computing power can be used to explore parameter spaces and get useful **reduced models**, that should be as structured as possible, again to minimize the amount of data needed to represent them. Good choices of basis, and good mapping functions have to be found in this area. This is tightly related to recent work that has been done in the domain of direct solvers, where **block compression** (i.e. representing blocks by low rank matrices) is observed to significantly improve the performance of the solvers on academic problems. But much work is still required to get compression based direct solvers that would exploit dynamically these techniques whenever possible with well controlled accuracy.