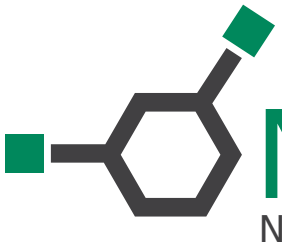

NESUS cost IC1305

Network for Sustainable Ultrascale Computing

Report on the requirements for ultrascale systems from the applications' perspective

Working Group 6

December 17, 2014

Contents

1 Introduction

1.1 Overview

Ultrascale computing systems will be available in the near future in the form of large-scale complex systems comprising both parallel and distributed components. There will also be an additional complexity at the micro level in the form of heterogeneous (GPU-enhanced) processors. These systems will provide a huge amount of potential computing power, which has to be made accessible to the applications. However, due to the complexity of such ultrascale systems, their usage is not as straightforward as for HPC systems available now. Consequently, adequate and timely software support is needed to make ultrascale computing possible and to make the hardware performance of these ultrascale systems available at the application level.
This report investigates ultrascale computing from the application programmer's point of view. The goal is to identify the requirements that need to be fulfilled in order to make ultrascale computing possible for real-world applications areas. Accordingly, this report is split into three sections. Section 1 contains a short introduction. Section 2 discusses specific requirements and their significance for ultrascale computing. Section 3 presents specific applications and discusses their specific requirements with respect to ultrascale computing. The appendix contains the results of a questionnaire that have been collected for different applications by the members of Working Group 6 (WG 6) of the Cost Action IC1305 NESUS (Network for Sustainable Ultrascale Computing). This report contains contributions from Lars Ailo Bongo, Raimondas Ciegis, Neki Frasheri, Jing Gong, Dragi Kimovski, Peter Kropf, Svetozar Margenov, Anatoliy Melnyk, Viktor Melnyk, Milan Mihajlović, Ricardo Morla, Maya Neytcheva, Thomas Rauber, Gudula Rünger, Roman Trobec, Roel Wuyts and Roman Wyrzykowski.

1.2 Objectives and topics

An important starting point for investigating the potential of ultrascale computing is to identify algorithms, applications, and services amenable to ultrascale systems. In this report, we will give a first analysis of the requirements which need to be fulfilled to port applications to ultrascale systems.
Applications amenable to ultrascale systems are a key aspect in the development of sustainable ultrascale computing. Candidate areas include earth sciences, astrophysics, chemistry such as molecular dynamics, material sciences, biology and life sciences, such as genomics and HPC sequencing, health science, high energy physics, such as QCD, fluid dynamics, scalable robust multiscale and multi-physics methods, and diverse applications for analysing large and heterogeneous data sets related to social, financial, and industrial contexts.
In applied science, there is a persistent requirement for the increase in the problem sizes that are tackled and for the optimisation of increasing parameter sets. Ultrascale computing might be a way to provide sufficient compute resources needed to process these increasingly larger problems in a reasonable amount of time. However, it is generally agreed that applications will have to be re-designed or re-programmed substantially in order to

reach ultrascale computing dimensions. The application redesign and reprogramming efforts can be in terms of new data structures, new algorithms or even new mathematics. In addition, each redesign of an application must take into consideration the cross-cutting issues, which are identified to be critical for sustainable ultrascale computing. Most important are resilience and fault tolerance mechanisms, handling of data I/O, especially for a growing amount of data on geographically distributed systems (big data), power management and energy efficiency, programmability and portability with respect to the underlying ultrascale hardware system. New programming models for flexible coding and performance adaptation as well as more abstract and advanced programming interfaces are key components for delivering highly sustainable and scalable applications. In this context, the following key objectives for analyzing applications have been identified as important research topics in studying ultrascale computers from the applications' perspective:

- Selection of a representative set of key applications that have a need for ultrascale computing with respect to computational power and data storage.
- Analysis of the data requirement and data access behaviour of selected applications.
- Evaluation of the needs of the selected applications in terms of scalability, programmability, portability, resilience.
- Identification of computational patterns and kernels that can be combined to build an application at a higher level of abstraction for leveraging programming for ultrascale systems.
- Categorization of applications for ultrascale systems with respect to important characteristics such as data requirements and distribution, computational structure and data access patterns.
- Identification of a set of key characteristics to determine a priori whether applications are amenable for ultrascale computing.

This report is a first step towards these objectives. Its emphasis is on the identification of requirements of applications towards sustainable ultrascale computing. Especially, the following requirements have been identified:

- power consumption and energy efficiency,
- sustainable data storage and data management,
- scalability,
- portability,
- programmability,
- resilience,
- productivity,
- security, integrity and privacy of data,
- continuous and data stream execution
- integration of self-configurability

- aspects of interconnection networks

These requirements are discussed with respect to sustainable ultrascale computing in more detail in Section 2. Specific applications are described in Section 3 and their specific requirements are gathered, distinguishing between (i) current state of the application, (ii) requirements concerning future ultrascale systems and (iii) necessary redesign/reprogramming aspects of the application for future ultrascale systems.

2 Cross-cutting requirements

Besides the functional correctness, applications always have other non-functional requirements to be fulfilled by the software and hardware system they are running on. Examples are time constraints for the parallel runtime or storage size to be provided. In the case of future ultrascale computing, such requirements are expected to be more dominant. An example is the energy consumption. In addition, new features may be necessary, e.g., productivity concerning the application development time of human resources. Other aspects might become more important, such as resilience or the impact of the interconnection network because of the size of the system. In this section, we reflect on a collection of such requirements with respect to ultrascale computing. This collection can also be seen as a collection of challenges to be solved for ultrascale computing.

2.1 Power consumption and energy efficiency

Energy consumption is an important aspect of ultrascale infrastructures. At the hardware level, energy-efficient processors with features such as power gating or DVFS (dynamic voltage frequency scaling) are designed to enable a reduction of the energy consumption at hardware level [31, 32].

These energy-saving hardware features are supported at different software levels. At the system software level, appropriate runtime systems can be developed to map computational parts of an application to hardware resources such that the overall energy consumption is reduced. The runtime systems are based on suitable load balancing and scheduling methods with the minimization of the resulting energy consumption as objective function. Here, an interaction with WG 2 is suitable. >From the perspective of the applications, the availability of energy-oriented runtime systems is an important requirement.

As a basis for an energy-efficient mapping of computations to hardware resources, suitable energy models are required that capture the power and energy behavior of application programs. These models have to take the computational characteristics of the application into consideration to provide good estimates for its power consumption on a target architecture. To explore such models, an interaction with WG 5 is useful.

At the application level, it can be observed that different applications may lead to a different power consumption, depending on the computational behavior of the application and the resulting usage of hardware resources. For ultrascale systems, the memory access and communication behavior definitely play an important role, but there may be other influences as well that need to be explored. This is particularly relevant for the case studies from Sections 3.1.1, 3.1.2, 3.1.6. The influencing factors should be included

into the energy models to capture the application characteristics. The identification of the influencing factors is an important research issue that needs to be addressed in the framework of ultrascale computing. The availability of suitable energy models is another important requirement to address the energy behavior of ultrascale systems from the application perspective. The redesign and reimplementation of the algorithms/codes for ultrascale applications need to address the problem of extensive communication patterns. In addition, the use of specialised architectures (such as FPGAs) which are known to have favourable flops/Joule ratios can be considered for reimplementing some frequently used kernels from scientific codes (for example, parts of linear algebra routines).

2.2 Sustainable data storage and data management

All applications have data that must be stored and managed, but the requirements for infrastructure and services have a wide variability. However, scientific applications can be divided into computationally-intensive and data-intensive. For the latter, application performance is dominated by the performance of data access operations. Traditionally, high performance computing applications have been computationally-intensive, but data-intensive applications have become essential in fields such as genomics, astronomy, and high-energy physics.

High performance computing systems typically use a centralized network-attached high capacity storage system. Applications therefore transfer data from the storage system to the compute nodes, run the computation, and write the results back to the storage system (this may be transparent to the applications). This works well for computationally-intensive applications. But for data-intensive applications there is a need for a higher I/O bandwidth. This can be achieved by distributing the storage on the compute nodes. The Hadoop Distributed File System (an implementation of the Google File System design) provides such a storage system.

Additional systems are built on a distributed file system. These provide services required by different applications such as fault-tolerance, random I/O, low-latency operations, iterative computations, incremental computations, transaction support, secure data storage, and so on. In addition there are many frameworks that makes developing applications for a particular domain easier, including high-level languages, declarative queries, and interactive data analysis frameworks.

Some applications will be both computationally-intensive and data-intensive. A possible example from the life sciences domain are machine learning techniques that take, among other inputs, high-resolution images as input. Further information can be found in [14, 36, 35].

These topics are addressed in the NESUS project in cooperation with working groups WG4 (Sustainable data management). Relevant large scale projects include the ELIXIR project, see `http://www.elixir-europe.org/`. There are also intertwined issues between this section and Section 2.8.

2.3 Scalability

Ultrascale systems are envisioned as large-scale complex systems joining parallel and distributed computing systems. Scalability is an important requirement to ensure that adding more resources to a system results in solving a given problem in the limits of potential achievable speedup, in allowing to increase the problem size in terms of increased output under increased load, and in maintaining system properties when the system size increases. A scalable system should allow to upscale from few variable smaller data sets on current parallel systems to many variable large data sets on future ultrascale systems maintaining the efficiency.
One may distinguish two principal types of scalability : scale-out, where more nodes are added to a system; and scale-up, where more resources are added to a node (where previously mentioned heterogeneity at micro-level can play a role). Scale-out often means adding CPU's to a node which offers opportunities for virtualization. Scalability raises difficult challenges as well in the case of substantial increase of the size of databases where strong consistency might be needed to be replaced by the weaker eventual consistency.
Another aspect is software scalability, capturing the behavior of an application on a hardware system with a larger number of resources [25]. On a larger hardware system, the application should exhibit satisfactory efficiency. For HPC systems, software scalability is typically related to the execution time and the resulting speedup when running the application. It is often distinguished between strong scaling, capturing the scaling behavior on a larger system for the same input size, and weak scaling where the input size is increased corresponding to the number of resources used for the execution. Weak scaling is much more relevant for ultrascale execution. Many applications mentioned later in this document would struggle to achieve very good strong scaling beyond hundreds of processors, but with a suitable load per processor may weakly scale way beyond that. For ultrascale systems, software scalability is especially important, since the resources of the ultrascale systems should be used efficiently without a significant re-writing of the application code when using a different hardware system. Because different heterogeneous architectures can be expected, for ultrascale systems software adaptivity also plays an important role here. Ideally the application software should be able to automatically adapt to a new execution situation on a new architecture.
Scalability for ultrascale systems requires new approaches such as task-based approaches as they are discussed in the next subsections.

2.4 Programmability

Programmability of large scale computing systems is a major concern for potential applications. It expresses the ability to implement an application in such a way that ultrascale computer systems can be exploited and the application's execution leads to high performance. The term programmability includes the requirement portability, since parts of or the entire application have to be ported to newer and larger hardware platforms and well-established programming models, such as MPI or OpenMP, as well as portable libraries or simulation tools are important. However, porting is usually not enough to

achieve high performance and a redesign or sometimes even a reimplementation of an application might be required. The complex task of redesigning an application has to be supported by a programming environment and a concise programming model for ultrascale applications. Such a programming model should provide an abstract view of the coarse (top-down) structure of an application. The specific way to support programmability is still to be investigated and proposed solutions may be application-specific. The implementations based on the abstract view may include many well-known subsolutions and the inclusion of standards, such as MPI, seems reasonable [30]. The requirement for the support of programmability will be investigated for specific applications in Section 3. Also, programmability is strongly related to productivity in code design for sustainable ultrascale computing.

Task-based programming approaches in combination with suitable runtime systems can support the software adaptivity of applications, since the task-based approach allows a hardware independent formulation of the application and the mapping of tasks to hardware resources can be performed by the runtime system such that the resources are efficiently used. This decouples the specification of the application's computations from the actual mapping to the computing resources. The runtime system can dynamically map tasks that are ready for execution to the computing resources, thus providing a dynamic load balancing that can adapt to the current execution situation of the hardware platform. This enables an efficient use of the computing resources and a good overall scalability of the application, provided that enough tasks are available for execution at each point in time during the execution of the application. Task-based approaches can be used with single-processor tasks [16], where each task is executed by a single execution unit, or multiprocessor tasks, where each task can be executed by multiple execution units in parallel [29, 28]. In the latter case, the actual number of execution units can be adapted to the execution situation at the time of task execution. Task-based approaches can also be used for balancing load between CPU and GPU by providing tasks in different versions for different platforms such as CPU or GPU and assign the right version to the platforms with free resources [18].

The task-based programming paradigm is also a promising approach in order to develop scalable solvers for ultra-scale computers. But here we should take into account that most real world large applications are dealing with parallel algorithms for PDE based models. Parallelization of such algorithms is based on another paradigm – data parallel algorithms. It is important to investigate if the existing parallel algorithms can be redesigned by using the task-based templates (e.g. Monte-Carlo methods).

2.5 Portability

The portability of parallel codes and algorithms is a very important issue for any specialist who is involved in parallel computing and applications. There are two aspects of portability: portability of functionality and portability of efficiency.

Clearly, these questions are mainly connected with selection, definition and constant improvement of programming languages and standards such as MPI, OpenMP and hybrid programming MPI+OpenMP. These have served very efficiently for the last 20 years. Now

the situation is changing and the new parallel architecture (many cores GPU) requires new ideas and tools (e.g. CUDA). Interesting approaches in this direction are captured by the HPCS (High Productivity Computing Systems) languages (X10, Chapel, Fortress) as well as other new programming languages such as CAF or UPC. Many of these new languages are based on the PGAS (Partitioned Global Address Space) concept which uses a global address space which is logically partitioned between the resources such that each resource has a portion of the address space attached to it to support the locality of memory reference. Languages, such as Cilk, that are based on the concept of tasks or task-based libraries also provide a useful abstraction that can support the portability to new hardware systems, see the discussion in Section 2.3. To explore these issues, an interaction with WG 2 is useful.
The second way is to use special libraries and templates (or macros), well adapted to some specialized types of problems (mostly for stencil driven algorithms) – examples are given by PETSc, LAPACK, ML, Hypre, CVODE libraries and toolkits. These libraries can be highly scalable and adaptable for a usage in ultrascale systems of different architecture. Using the libraries can increase the portability of application codes due to the portability of the libraries. There is an interaction between the development of scientific libraries and the usage of new programming approaches for the development of these libraries in the sense that the usage of specific programming approaches influences the characteristics and the efficiency of the libraries.
The third way is to use simulation tools as OpenFOAM, COMSOL, ANSYS, where the portability of solvers must be guaranteed by developers of this software.

2.6 Resilience

Efficient utilisation of ultrascale computer architectures in scientific computing is restricted by possible availability deficiencies. To handle hardware failures, the software needs some special features such as checkpointing and rollback facilities. In particular, the possibility of availability deficiencies requires that an application has frequent checkpoints for synchronisation and correctness verification. Runtime system configuration changes caused by failing processing elements are usually difficult to handle by numerical algorithms and require a complete restart of the application (this can be alleviated to some extent by checkpointing). Efficient mapping of numerical algorithms to high-end architectures should allow for robust execution in the presence of hardware failures, either by the ability to take preemptive actions before a failure affects a running application, or, by creating a (hardware/software) fault-tolerant version of the algorithm, capable of recovering the solution within a timescale that is much shorter than that of re-running the entire application. In addition, the computational error introduced by this process should be mathematically bounded (easy to measure and control). The resilience issue of applications is an important point for ultrascale systems and will therefore be addressed in more detail in the future in collaboration with WG 3.

2.7 Productivity

Productivity refers to the efficiency of implementing applications on specific architectures. The resulting application should be functional and reasonably efficient. Productivity is especially important for ultrascale systems and also includes the effort that must be invested to extend existing (parallel) applications to the new ultrascale systems such that the available resources of the systems are sufficiently well exploited. It is clear that a complete re-implementation of the application with a new programming model should be avoided in most situations. However, it would be unreasonable to expect that no change in the software is required in order to use it efficiently on an ultrascale system. A preferred scenario would be if the application can be adapted with minor changes to the new platforms. Another aspect of productivity is the extensibility of the application code to include new features and functionalities without negative effect on its scalability and efficiency.

The reimplementation effort needs to be sustainable in the sense of making an application reasonably efficient on different ultrascale architectures. The use of new task-based runtime systems which decouple the concerns of the computation specification and its mapping to computational resources can be an important step towards an increase in software development productivity for ultrascale systems. In this context, an interaction with WG 2 is useful.

Productivity is inherently intertwined with other requirements discussed in other subsections. In particular, a good portability as well as a good scalability of an application code leads to a higher productivity.

2.8 Security, integrity and privacy of data

The security, integrity and privacy of data is of paramount importance in the development of the state-of-the-art ultrascale computing systems, in particular if these are cloud-based. The intrinsic heterogeneity of such systems induces flexible and ever-changing structure, in the sense of the geographical and logical distribution of the hardware resources, thus allowing scalability of the system to a very large size. Unfortunately, this architectural concept requires numerous remote data transfers. This problem is aggravated when we take into account that nodes distributed on different geographical points, could be instructed to execute a parallel code on a given data set simultaneously.

The transfer and storage of data poses security threats, which could be perceived as risks involved in the transfer itself and security risks connected with the enormous scaling of the system. Each time a new node is connected to the system it could increase the security risks, as the new node could be infected by ill-intended code or it could be a Trojan node. Moreover, the frequent data transfer could be intercepted and the data could be stolen and used for unwanted purposes. The security of the data, especially in today's "Big Data" world, still remains an unsolved problem. The researchers from the COST IC1305 action should direct their skills towards finding novel ways of solving these problems.

On the other hand, it is also important to address the integrity of the data and to find

new ways to provide high data redundancy with a minimal number of involved hardware resources. The HADOOP framework addresses this problem, but it is unknown if it could be scaled up to an ultrascale size. Furthermore, HADOOP lacks some security protocols and it requires a huge amount of hardware resources.
The COST IC1305 researchers, in collaboration with the social sciences, should also find a way to optimally classify the "Big Data" information without any intrusion of the privacy of the users.

2.9 Continuous and data stream execution

Performing operations on continuous or streaming data sets on ultrascale platforms poses additional challenges. The usual setting is that a set of tasks is performed periodically on large and continuously changing data sets. Examples of such changing data set are social media data, communication networks logs and financial modelling. The goal is for the execution time of the set of tasks to be smaller than the execution period. We can have hard or soft deadlines. With hard deadlines results are only valid if they are obtained before the next execution period. With soft deadlines results may still be valid even after the next execution period, depending on the chosen soft constraints. If this happens persistently the application lag can become unbounded. The second part of this requirement is the data stream processing of the changing data set. Data stream processing means that the processing of each data set is not independent of the previous data set and online learning from the data is possible. One implication for sustainable ultrascale computing from the first part is on scheduling and resource allocation to make sure the hard or soft deadlines are met; one implication from the second part is that temporary data structures must be maintained between executions.

2.10 Self-configurability

Today's computer systems performance is steadily improving, based on two major architecture approaches: multicore and multiprocessor architecture, and heterogeneous architecture with use of hardware acceleration. The last approach is often based on the application of FPGA-based hardware accelerators, also called reconfigurable, and is characterized by a better performance / power consumption ratio and lower cost as compared to general-purpose computers of equivalent performance. The combination of a general-purpose processor and application-specific processors synthesized in reconfigurable logic, the structure of which considers executed algorithms features, allows to increase its overall performance by orders of magnitude.
However, FPGA-based accelerators and reconfigurable computer systems (that use FPGAs as a processing unit) have typical problems: 1) the process of writing applications requires a special program to perform computing tasks balancing between the general-purpose computer and the FPGAs; 2) require designing the application-specific processor soft-cores; and 3) are effective for certain classes of problems only, for which application-specific processor soft-cores were originally developed. The problems related to designing the heterogeneous computer systems with the use of the hardware accelerators, primarily the

reconfigurable ones, are considered in the book [22].
A solution of the mentioned problems for reconfigurable computer systems is proposed in [23], using the concept of the self-configurable FPGA-based computer systems design, the method of information processing in them, and their structure.
A self-configurable computer system is a computer system with reconfigurable logic, where a program compilation includes automatically performed actions to create the particular configuration and which acquires that configuration automatically during the program loading for execution.
In a self-configurable computer system, 1) the execution of the computational load balancing between the general-purpose computer and the reconfigurable logic and 2) the creation of the ASP's programming model are automated, and loading the configuration files obtained after the logical synthesis into reconfigurable logic is carried out not by the user but by the operating system in parallel with loading the computer subprogram executable file into its main memory after the program initialization. This ensures an effective use of the reconfigurable logic to perform arbitrary tasks, shortens information processing time, and reduces information processing complexity, since requirements to the user experience are simply reduced to knowing the high level programming language. This can be relevant for some of the application scenarios outlined in Section 3.

2.11 Interconnection networks

Interconnection networks (ICNs) have an important role in cooperating computing systems, because they directly influence the speed-up, scalability and consequently the run-time and power consumption. Today the ICN's performance has the same importance as the performance of the CPU because the execution time depends on both - communication time and calculation time. The ICN determines the efficiency of a high-performance parallel computer on most real-world parallel applications. It can shorten the overall execution time and increase the number of processors that can be efficiently exploited, which both leads to a higher ultimate speedup.
The performance of an ICN depends on many factors with the three most important: topology, routing, and flow-control algorithms. The routing and flow-control algorithms have advanced to a state where efficient techniques are known and used [7]. Many topologies have already been present since the dawn of parallel computing and are still widely used. With contemporary standards like Infiniband, vendors and end-users do not have the capability to alter the routing and flow-control. Recent initiatives in the Network Functions Virtualization (NFV) assume the software-based virtual implementations of networking devices [6] such as switches, routers, firewalls, traffic analyzers, load balancers, etc. The NFV can be easily combined with the concept of Software Defined Networking (SDN), which improves the performance and manageability of network functions. The end users can apply the SDN to define a number of different topologies, based on an anticipated usage.
A further step in performance increase is made possible by an improved ICN topology or by innovative technological approaches in optical networking, which could solve current ICN bottlenecks, i.e., message latency and non-efficient collective communication. New

approaches in Networks on Chips (NoC) with high level node radices will be considered as an option for further improvement of performances on the chip level. It is expected that ICNs will be able to adapt dynamically to the current application in some optimal way in the near future. It is important to analyze applications and their requirements concerning the ICN topologies that are currently used in high-performance parallel computers [37], concentrating on the ICNs from the present top-level systems. Based on past and present technology trends it is also relevant to establish several proposals for future ICNs that are expected to better fit the needs of high-performance parallel computer applications or to be tailored to exascale applications.
There is a well known problem of sparse matrix computations losing performance on multicore systems due to the ICN problems. It would be interesting to develop models of data traffic and try to optimise it for frequently used sparse matrix kernels.

3 Requirements of specific applications

3.1 Applications from Scientific Computing

3.1.1 FEM supercomputer applications

An important class of supercomputing applications involve efficient implementation of advanced methods and parallel algorithms for numerical solution of partial differential equations (PDEs). Nowadays, the finite element method (FEM) can be considered as a leading computational technology for continuum (macroscopic) modelling in science and engineering. The recent and future advanced simulations are inter-disciplinary and have multiple spatio-temporal scales, leading to practically unlimited requirements for supercomputing resources. The FEM supercomputing applications are inherently computationally intensive. At the same time, the related parallel algorithms are strongly coupled, causing specific requirements related to the balance between computations and communications. In this consideration we will focus on topics related to single process problems (scalar or vector) which could be stationary (e.g. elliptic PDEs) or time dependent (e.g. parabolic PDEs).
More than 70% of the entire computing time of FEM based engineering simulations is spent solving linear algebra problems (this can be verified by using popular libraries, such as Trilinos and HIPRE). The included efficient fast parallel preconditioned conjugate gradients type solvers are often of (aproximately) optimal complexity, see e.g., the available implementations of algebraic multigrid (AMG) methods. However, the robustness in the case of strongly heterogeneous media and/or strong anisotropy is still a challenging problem. The same applies to singular perturbations of the elliptic PDEs, such as the convection-diffusion problem, which is a major issue in fluid mechanics and transport phenomena. The efficient implementation of FEM models includes also the important steps of mesh generation and partitioning of the graph representing the sparsity pattern of the FEM linear system. The available mesh generators are based on a sequentially constructed coarser (unstructured) mesh using, e.g., Netgen, which is then refined uniformly in parallel. This is not necessarily a computationally optimal scenario – adaptive refinement may be

a better option, but requires frequent grid repartitioning to preserve the load balance. Perhaps some interesting link can be established between this problem and the low cost mechanism for particle distribution suggested in Subsection 3.1.7. The complete parallel generation of conforming unstructured meshes is still a challenge. The next related problem concerns the mesh partitioning. Here we could refer to the commonly used software packages ParMETIS and SCOTCH. They both are based on recursive partitioning strategies balancing the measure of the sub graphs and minimizing at the same time the measure of the interfaces. In this respect, the quality of the results could be considered as acceptable. The problem is that the number of the neighbors is not properly controlled which leads to serious problems mapping the graph of the algorithm onto the graph of the parallel architecture. Our last related comments concern the more general problem of balancing the local and global communications. For parallel distributed systems with hundreds of thousands of processors/cores, the global communications (related, e.g., to dot product, FFT, etc.) have become one of the fundamental bottlenecks. Some of the user communities will have to change their way of thinking. As a consequence, most probably some of the FFT based codes will have to be modified to AMG solvers as a way to avoid the transposition step reducing also the logarithmic factor in the almost optimal order of computational complexity. Similar to other approaches, AMG may have its own problems when mapped to an architecture with large number of processors – recent works aimed at reducing the operator complexity, improving the quality of interpolation, reducing the communication patterns at coarse levels, exploring fine-grained parallelism, while retaining numerical robustness might be of interest here.
The energy efficiency of FEM applications on ultrascale systems is one of the key challenges. New proper metrics are to be developed and tuned to get a complex assessment of the related simulators. The fault-tolerance issues are also to be addressed in a proper way at the levels of methods, algorithms and software implementations. Some advantage of the iterative solvers as well the time-stepping algorithms is that they have inherently self-correcting mechanisms which are to be further developed in the context of ultrascale computing.
The experience of the IICT-BAS team includes FEM supercomputing simulations of bio-medical, environmental and engineering problems. High parallel scalability (both, strong and weak) is obtained on heterogeneous Linux platforms, including IBM Blue Gene/P, HPC CPU clusters, and more recently hybrid CPU/GPU/MIC clusters. The used programing methods and environments include C, MPI, OpenMP, CUDA. Among more recently released libraries for platforms with accelerators, we could mention PARALUTION. More information can be found in [27, 25, 26].
This application is provided by Svetozar Margenov.

3.1.2 Parallel preconditioning of multi-physics problems

Numerical solution of partial differential equations (PDEs) requires the solution of systems of linear equations. Discretisation methods with local basis sets (such as finite elements) result in linear systems that are large, have sparse coefficient matrices, and are ill-conditioned. Optimal solution strategies for such systems are based on preconditioned

Krylov solvers.
Recent advances in computational modelling techniques and the increasing computing power allow us to tackle complex physics and engineering problems referred to as the multi-physics problems. These problems involve several unknown quantities and the underlying mathematical model is expressed as a system of PDEs (the examples include thermal convection, fluid-structure interaction, magnetohydrodynamics). Grouping the discrete unknowns of the same type leads to a natural blocking of the coefficient matrix. In this context, block preconditioners are commonly deployed to accelerate the convergence of Krylov solvers.
We have developed the block preconditioning framework (BPF) within OOMPH-LIB (see http://oomph-lib.org/), an object-oriented multi-physics finite element library [9, 34, 24]. The BPF facilitates rapid development of new preconditioners, while hiding the low-level implementation details (including parallelisation). However, the overall parallel performance of any multi-physics solver crucially depends on scalar solvers (usually library codes produced by third parties), such as algebraic multigrid (AMG). Standard AMG codes, such as BoomerAMG (developed at LLNL) perform robustly and show good scalability over hundreds of processors. Novel coarsening techniques (PIMS, HIMS, non-Galerkin) and thechniques for enhanced (long-distance) interpolation are designed to reduce communication and enhance scalability, while retaining robustness, especially for complex diffusion and convection-diffusion problems. By contrast, there is comparatively little work on resilience and energy efficiency of AMG solvers. The aim of this part is to make some progress in this direction.
This application is provided by Milan Mihajlovic.

3.1.3 Meshless numerical solutions of multi-physics problems

During the last few decades computational mathematics and numerical simulations have been steadily drawing much attention, enabling the development of advanced technologies and contributing to better understanding of numerous natural phenomena that are not tractable via classical theoretical research or fields or lab experiments. The modelling and simulation of more and more complex physical systems helps the society to address important issues such as identifying environmental problems, improving technological processes, developing biomedical applications, etc. The models describing such problems are commonly described with coupled systems of Partial Differential Equations (PDE) that require numerical treatment. Traditionally, the mathematical models are converted into discrete models using numerical methods such as Finite Differences (FDM), Finite Elements (FEM), Finite Volume (FVM), Boundary Integral Methods (BEM) and some combinations of those. A common feature of all these methods is that they rely on some discretization mesh and that sometimes poses additional difficulties as resolving complex geometries, adaptive refinements that require local refinements and de-refinements, large stencils, moving meshes resolving dynamical interfaces etc. The latter requirements affect the parallelization and implementation of the methods on HPC platforms as they affect the load balancing, the amount of local communications etc. A promising alternative to the mesh-based methods is the class of the so-called Local Meshless Methods (LMM) based on

scattered discretization points, in particular some variants that result in algebraic systems of equations with better conditioned matrices. LMM allow for easy implementation of local refinements and de-refinements, basis augmentation, increasing approximation quality, treating special features in the problem, such as sharp discontinues or other intricate situations, which might occur in complex simulations.
A parallel computational framework for solving multi-physics problems based on LMM has been developed, (see http://www-e6.ijs.si/ParallelAndDistributedSystems/) providing the possibility to model and perform numerical simulations of problems originating from molecular dynamics, graph algorithms (clique) and discrete simulations (ECG simulator based on AP in cells), multiple transport equations (heat, bio-heat, solute, radiation, etc.), multi-phase fluid flow (free fluid, porous media), phase change dynamics on two levels (micro-macro), drift-diffusion equations (semiconductor simulations), all on non-uniform domains r-adaptive dynamic nodal distributions. The code is parallelized and can be efficiently executed on multi-core computers and distributed systems. The communication issues have been studied and tested on an in-house computing cluster. The solvers are coupled with an evolutionary multiobjective optimization package for automatic optimization of parameters. The code could be extended to exascale range and could be used for performance benchmarking of novel hardware platforms.
This application is provided by Roman Trobec.

3.1.4 Earth Sciences Applications

Earth sciences include a number of important scientific disciplines such as geology, geophysics, ecology, hydrology, oceanography, climatology etc, important for living conditions of human society, raw materials and circulation of wastes. The complexity of physical, chemical and biological processes, as well as the volume of data structures makes the HPC obligatory for the analysis, modeling and simulation of relevant processes. For such problems where experiments are impossible the extreme-scale computing approach may enable the solution of high resolution models and the analysis of massively large data sets, for example: regional climate changes (sea level rise, drought and flooding, and severe weather patterns). Climate models developed through decades have over one milion lines of code. At the same time architectural changes of computing platforms need more sophisticated algorithms and computer techniques [33, 15, 12]. Oceanographic, atmospheric and climate simulations are a typical example of applications that require HCP to process a huge volume of data [19, 1, 38]. Analysis of remote sensing data in space-time domains to evaluate environmental changes and predict their future evolution requires complicated processing of a huge volume of data. Moreover, such an analysis must be coupled with the simulation of related environmental processes and interfaced with human activities. Earth System Science mixes together physics, geology, geophysics, engineering, chemistry, mathematics and computating to study interconnected systems operating on extreme time and spacial scales where small-scale heterogeneities affect large-scale phenomena. The scientific research accommodating these scales pushes and challenges the frontiers of numerical and computational methods [3, 20, 5].
A particular direction of earth sciences applications is geophysical modelling and inversion.

The work for the application "Geophysical Modeling and Inversion" started during the FP7 project HP-SEE in the Center for Research and Development in IT of Polytechnic University of Tirana in collaboration with specialists from the Faculty of Geology and Mining and of the Academy of Sciences in Albania. It offers an example for the scalability of geoscience applications in HPC systems and the need for ultra-scale computing in order to cope with high resolution models required for regional scientific and local engineering studies. The application is focused in the inversion of gravity anomalies using approximation based on relaxation methods and is the result of an interdisciplinary cooperation - computer sciences, applied mathematics and geophysics. The runtime depends on the model size in the best case at the range of $O(N^8)$ where N is the spatial resolution in one dimension. The application has been developed in C and MPI, OpenMP was used as well while testing in NIIFI platforms. The application has been executed in the HPCG Cluster of the Institute of Information and Communication Technologies, Academy of Sciences of Bulgaria, and the SGE system of the NIIFI Supercomputing Center at University of Pécs, Hungary in parallel with a maximum of 1000 cores. It was possible to achieve in a reasonable runtime a resolution with $N = 80$ nodes and a spatial step of 50 meters, which may be sufficient for gravity prospecting but not for other geophysical methods looking for two dimensional structures with thickness of the order of one meter in complicated heterogeneous and partially anisotrop medium. In small parallel systems is not possible to run high resolution models necessary for engineering works; also in the current version of the application only gravity is considered that does not require high resolution models, while for applications in other areas as magnetism and electricity the high resolution is mandatory.

One example of an application problem that is of significant importance and a potential impact for our society is the so-called glacial isostatic adjustment that comprises the response of the solid Earth to redistribution of mass due to alternating glaciation and deglaciation periods. The processes that cause subsidence or uplift of the Earth surface are active today. To fully understand the interplay between the different processes, and, for example, be able to predict how coast lines will be affected and how glaciers and ice sheets will retreat, these have to be coupled also to recent global warming trend and melting of the current ice sheets and glaciers world wide.

Due to the extreme space and time scales involved (the simulations should be performed on the Earth globe for time periods of about 100000 years) the GIA processes can only be studied via computer simulations. A detailed model of the phenomena includes three-dimensional geometry, viscoelastic and inhomogeneous material behavior, self-gravitation effects, modelled via a coupled system of partial differential equations. Presently, at the Division of Scientific Computing, Uppsala University, a two-dimensional benchmark, often used by the geophysisists, has been studied from a point of view of accuracy as well as numerical and computational efficiency. The problem is discretized using the Finite Element method and performance studies have been done on CPU and GPU platforms using OpenMP and MPI paradigms (cf. [8]). The long-term aim is to couple GIA modeling with other large scale models, such as Climate and Sea-level changes, Ice modeling etc.

This application area has been provided by Neki Frasheri and Maya Neytcheva.

For environmental systems that require the simulation of surface water and groundwater, HydroGeoSphere [4] (HGS) presents an advanced solution, allowing for the physics-based simulation of interactions and feed- back mechanisms between the two compartments. HydroGeoSphere is a numerically demanding code implementing a 3D control-volume finite element hydrologic model describing fully integrated surface-subsurface water flow and solute and thermal energy transport.
The model parameters employed in HGS need to be calibrated in order to adequately represent a given environmental system. So-called data assimilation systems provide an alternative to conventional model calibration systems: they allow the sequential update of system states and model parameters whenever new data becomes available, thus guaranteeing a continuous improvement of predictions. The Ensemble Kalman Filter (EnKF) [11] provides an optimal data assimilation mechanism for conjunctive use with HydroGeoSphere and environmental data, because it allows quantifying the uncertainties of predictions. The prediction provided by an EnKF-based simulation is then represented by the statistical moments of the ensemble of realizations. As a high number of simulations is required, and as data assimilation techniques such as the EnKF allow continuous re-adjustment of system states and re-calibration of model parameters whenever new data becomes available, such modeling systems are ideally suited for parallelization. At University of Neuchatel, a cloud-based environment (OpenStack, AWS S3 compliant object store) has been developed to provide *near-real-time* re-adjustement of system states and re-calibration of model parameters whenever new monitoring data becomes available [2, 17].
When simulating larger fine grained HGS models, parallelization is essential because the model solves tightly-coupled highly-nonlinear partial differential equations. The target parallelization includes the composition of the Jacobian matrix for the iterative linearization method and the sparse-matrix solver, preconditioned BiCGSTAB [13]. Performance studies are currently undertaken on CPU and GPU using OpenMP and MPI paradigms concentrating on the forward- and backwards solvers on the sparse matrices.
This application has been provided by Peter Kropf.

3.1.5 OpenACC acceleration for Nek5000: spectral element CFD code

Nek5000 is an open-source code for simulating incompressible flows and its discretization scheme is based on the spectral-element method [21, 10]. The code is widely used in a broad range of applications and more than 200 users are using Nek5000 in the world. In the EU project CRESTA (Collaborative Research into Exascale Systemware, Tools and Applications), PDC-HPC at KTH Royal Institute of Technology mainly focuses on software challenges using hybrid computer architectures with accelerators for ultrascale simulations in collaboration with KTH Mechanics, EPCC, Cray UK and Argonne National laboratory. We have ported the CFD code Nek5000 on massive parallel hybrid CPU/GPU systems and presented a case study of porting simplified version, NekBone, to a parallel GPU-accelerated system. We reached a parallel efficiency of 68.7% on 16,384 GPUs of the Titan XK7 supercomputer at the Oak Ridge National Laboratory. Currently we are working on porting and optimizing the full Nek5000 code to multi-GPU systems and

maximum can run on 1,024 GPUs.
Nek5000 employs the multigrid preconditioner that combines the Schwarz overlapping method with subdomain solvers based on the fast diagonalization method. However the rather complicit multigrid preconditioner would reduce the global solution time when port the Nek5000 code to multi-GPU systems. Within the action COST IT1305, we may investigate the efficient preconditioner developed in 3.1.2 in collaboration with Milan Mihajlovic. The application is written in mixed C/Fortran and requires a system with multi-GPUs.
This application is provided by Jing Gong.

3.1.6 EULAG numerical model

The EULAG model is an ideal tool to perform numerical experiments in a virtual laboratory with time-dependent 3D adaptive meshes and within complex, and even time-dependent model geometries. These abilities are due to the unique model design that combines the nonoscillatory forward-in-time (NFT) numerical algorithms and a robust elliptic solver with generalized coordinates. The code is written as a research tool with numerous options controlling the numerical accuracy, and to allow for a wide range of numerical sensitivity tests. The computational core of EULAG contains two main parts: MPDATA advective transport algorithm and GCR elliptic solver with the Thomas preconditioner. A more sophisticated preconditioner could be used in this context - see Section 3.1.1).
The application is a result of a cooperation of the Czestochowa University of Technology, Poznan Supercomputing and the Networking Center, Institute of Meteorology and Water Management in Warsaw. We port this code on multi GPUs and multi Intel Xeon Phis platforms. The application is written in C++/Fortran using CUDA, OpenMP and MPI standards. It requires a cluster with nodes containing NVIDIA GPUs or Intel Xeon Phi co-processors. The application is optimized for Fermi and Kepler GPU architectures, as well as Intel MIC architecture. Memory requirements include about 20 GB of HDD (for input and output data), and about 16 GB of RAM per node and about 8 GB of inter co-processor memory.
Performance of the EULAG application within a single node of cluster is mostly limited by the low flop-per-byte ratio of computation - less than 1.7 for MPDATA, and even less than 0.2 for GCR solver, while the minimum flop-per-byte ratio required e.q. by NVIDIA Tesla K40m to achieve the maximum performance is 5.2. The main constraint for providing scalability of the application across cluster nodes is the presence of global communications in the GCR elliptic solver. This is the crucial bottleneck for all sparse matrix calculations – a low computation to fetch ratios, made even worse on multicore architectures where multiple cores have common fetch buses.
This application is provided by Roman Wyrzykowski.

3.1.7 Particle–In-Cell method for particle distributions – Helsim

Helsim implements an explicit three-dimensional electro-magnetic Particle-in-Cell simulation. It was developed in the ExaScience Lab in Leuven, with contributions from many

partners in the project. Helsim takes particular care of balancing the computational load for handling the particles and trading this off against computation. This allows Helsim to simulate experimental configuration with highly imbalanced particle distributions with ease. Moreover Helsim includes in-situ visualization, where the visualization happens during the simulation.
Helsim uses the Shark Library, developed in the lab, to store all of its distributed data structures, including the particles and the various grids. Shark is a high-level library for programming distributed n-dimensional grids. It allows building performant implementations of grid computations in a highly productive manner. Broadly speaking, Shark manages the bookkeeping and distribution of grid data structures, and offers specific computation and communication operations to work with the grid data. It extensively uses C++11 constructs like lambda expressions to make it easy to work with n-dimensional grids. The Shark runtime system manages parallelism on three levels, which are common in today's multicore cluster architectures: distributed memory parallelism using one-sided communication from MPI 2/3; shared memory parallelism using a thread scheduler such as OpenMP, direct Pthreads, Intel Threading Building Blocks (TBB), and others; and SIMD vector instructions using compiler auto-vectorization assisted with pragmas.
Specific for Helsim, when compared to other PIC simulators, is that particles are evenly distributed over the cluster such that each core holds the same amount of particles, stored according to the cell they belong to. As particles move throughout space during the simulation a low-cost, lightweight mechanism is used to adjust the particle distributions. The 3D fields (electric field, magnetic field, etc.) are block-distributed over the cluster, completely decoupled from the particle data structures. When particle information is propagated to the fields (charge density and current interpolation), and vice versa (interpolating the electric and magnetic fields to particle positions) each core uses a local representation of the grid in order to be able to work locally and overlap computation with communication (updating the actual distributed grid).
Helsim also features in-situ visualisation that runs in parallel with the simulation, directly using the data from the simulation, using a custom distributed raycasting engine. Helsim was run on up to 32 thousands cores on the Curie T0 System in France, as well as on various smaller clusters. Helsim was developed at the ExaScience Lab, a collaboration between Imec, Intel, and all five Flemish universities.
This application is provided by Roel Wuyts.

3.2 Non-numerical applications

3.2.1 Scalable network traffic analysis

Computer networks move a huge amount of data. Each link can easily reach several Gbit/s and a network can have hundreds of links. Analyzing the data that goes through the network is important for traffic engineering and network security purposes. This means that the raw packets must be analyzed at line speed and processing of the overall information flows must be done in a coordinated way, for example to extract graph topology metrics relevant for classifying traffic and outliers. Several work exists on using

Hadoop and map-reduce to process traffic data, however this is still earlier research and has focused mostly on characterizing the performance and the limits of the approach on different network traffic type and bandwidths. Challenges remain on understanding the performance of more demanding graph-like processing of network traffic and on characterizing the continuous execution performance of this application and of schedulers that may be used for this.
Besides the real-time analysis of large datasets referring to network traffic, there are many open challenges concerning the intelligent storage and management for further use of these datasets on secondary storage. The increasing capacity and lower costs of these devices do not counterbalance the present and future requirements where terabytes of data generated every day is the norm in ordianry scenarios. Hence, novel solutions should be pursued in terms of distributed and parallel I/O architectures, data structures, indexes, aggregation and search algorithms.
This application is provided by Ricardo Morla.

3.2.2 LPreP – Preparing files for variant calling in sequencing pipelines

elPrep is a high-performance tool for preparing .sam/.bam files for variant calling in sequencing pipelines. It can be used as a drop-in replacement for SAMtools/Picard, and was extensively tested with the GATK best practices pipeline for variant analysis.
elPrep is designed as an in-memory and multi-threaded application to fully take advantage of the processing power available with modern servers. Its software architecture is based on functional programming techniques, which allow to easily compose multiple alignment filters and perform optimizations such as loop merging.
elPrep is released as an open-source project under a BSD 3-Clause License (BSD 2.0). It lives on github, where you can also find an explanation on how to use it and demo files to get started: https://github.com/ExaScience/elprep. elPrep is being developed at the ExaScience Life Lab, a collaboration between Imec, Intel, Janssen Pharmaceutica, and all five Flemish universities.
This application is provided by Roel Wuyts.

References

[1] Modeling atmospheric and oceanic flows: Insights from laboratory experiments and numerical simulations. American Geophysical Union, Wiley, 2014.

[2] Lapin A., Schiller E., P. Kropf, O. Schilling, Ph. Brunner, A. Jamakovic-Kapic, Braun. T., and S. Maffioletti. Real-time envoronmental monitoring for cloud-based hydrogeological modelling with hydrogeosphere. In *High Performance Computing and Communications conference HPCC*, Paris, 2014. IEEE.

[3] A.Fichtner, D.Giardini, A.Jackson, T.Nissen-Meyer, D.Peter, J.Robertsson, P.Tackley, L.Dalguer, D.Roten, O.Schenk, and M.Grote. Hpc roadmap. white paper, Solid Earth Dynamics, 2013.

[4] Philip Brunner and Craig T. Simmons. Hydrogeosphere: A fully integrated, physically based hydrological model. *Ground Water*, 50(2):170–176, 2012.

[5] C.Yang, M.Goodchild, Q.Huang, D.Nebert, R.Raskin, Y.Xu, M.Bambacus, and D.Fay. Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing? *International Journal of Digital Earth*, 4(4):305–329, July 2011.

[6] E. Keller D. Drutskoy and J. Rexford. Scalable network virtualization in software-defined networks. *IEEE Internet Computing*, 17:20–27, 2013.

[7] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

[8] Ali Dorostkar, Dimitar Lukarski, Björn Lund, Maya Neytcheva, Yvan Notay, and Peter Schmidt. Parallel performance study of block-preconditioned iterative methods on multicore computer systems. In *Proceedings of the Europar 2014 conference*. Springer LNCS, 2014.

[9] H.C. Elman, M.D. Mihajlović, and Silvester D.J. Fast iterative solvers for buoy-ancydriven flow problems. *Journal of Computational Physics*, 230(10):3900–3914, 2011.

[10] J. Gong, S. Markidis, M. Schliephake, E. Laure, D.S. Henningson, P. Schlatter, A. Peplinski, A. Hart, J. Doleschal, D. Henty, and P. F. Fischer. *Nek5000 with OpenACC*. Springer (accepted), 2014.

[11] H. J. Hendricks Franssen and W. Kinzelbach. Ensemble kalman filtering versus sequential self-calibration for inverse modelling of dynamic groundwater flow systems. *Journal of Hydrology*, 365(3-4):261–274, 2.

[12] H.Johansen, L.C.McInnes, D.E.Bernholdt, J.Carver, M.Heroux, R.Hornung, P.Jones, B.Lucas, and A.Siegel. Software productivity for extreme-scale science workshop report. white paper, Workshop on Software Productivity for Extreme-scale Science, January 13-14, 2014, Rockville, MD, 2014.

[13] Hyon-Tae Hwang. *Development of a parallel computational framework to solve flow and transport in integrated surface-subsurface hydrologic systems*. PhD thesis, University of Waterloo, Canada, 2012.

[14] H. V. Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7):86–94, July 2014.

[15] H. Johansen, D. Bernholdt, B. Collins, M. Heroux, R. Jacob, P. Jones, L.C. McInnes, J.D. Moulton, T. Ndousse-Fetter, D. Post, and W. Tang. Extreme-scale scientific application software productivity: Harnessing the full capability of extreme-scale computing. white paper, ASCR, 2013.

[16] M. Korch and T. Rauber. A Comparison of Task Pools for Dynamic Load Balancing of Irregular Algorithms. *Concurrency and Computation: Practice and Experience*, 16:1–47, 2004.

[17] Peter Kropf, Eryk Schiller, Philip Brunner, Oliver Schilling, Daniel Hunkeler, and Andrei Lapin. Wireless mesh networks and cloud computing for real time environmental simulations. In *Recent Advances in Information and Communication Technology - Proceedings of the 10th International Conference on Computing and Information Technology, IC2IT 2014, Angsana Laguna, Phuket, Thailand, 8-9 May, 2014*, number 265 in Advances in Intelligent Systems and Computing, pages 1–11. Springer, 2014.

[18] J. Lang and G. Rünger. An execution time and energy model for an energy-aware execution of a conjugate gradient method with CPU/GPU collaboration. *Journal of Parallel and Distributed Computing*, 74(9):2884–2897, 2014.

[19] M. Guest (lead author). The scientific case for high performance computing in europe 2012-2020. Technical report, FP7 Project PRACE RI-261557, 2014.

[20] L.Hwang, T.Jordan, L.Kellogg, J.Tromp, and R.Willemann. Advancing solid earth system science through high-performance computing. Technical report, Lawrence Livermore National laboratory, University of California, 2014.

[21] S. Markidis, J. Gong, M. Schliephake, E. Laure, A. Hart, D. Henty, K. Heisey, and P. F. Fischer. Openacc acceleration of nek5000, spectral element code. *Advances in Engineering Software Journal (to appear)*, 2013.

[22] A. Melnyk and V. Melnyk. *Personal Supercomputers: Architecture, Design, Application*. Lviv Polytechnic National University Publishing, 2013.

[23] A. Melnyk and V. Melnyk. Self-configurable FPGA-based computer systems. *Advances in Electrical and Computer Engineering*, 13(2):33–38, 2013.

[24] R.L. Muddle, Mihajlović M.D., and M. Heil. An efficient Preconditioner for Monolithically- Coupled Large-Displacement Fluid-Structure Interaction Problems with Pseudo-Solid Mesh Updates. *Journal of Computational Physics*, 231(21):7315–7334, 2012.

[25] S. Margenov N. Kosturski and Y. Vutov. Balancing the communications and computations in parallel fem simulations on unstructured grids. In K. Karczewski R. Wyrzykowski, J. Dongara and J. Wasniewski, editors, *Parallel Processing and Applied Mathematics*, pages 211–220, Heidelberg Dordrecht London New York, 2012. Springer LNCS 7204.

[26] S. Margenov N. Kosturski and Y. Vutov. Computer simulation of rf liver ablation on an mri scan data. In M.D. Todorov, editor, *Application of Mathematics in Technical and Natural Sciences*, pages 120–126, Melville, NY, USA, 2012. AIP Conf. Proc. 1487.

[27] S. Margenov P. Arbenz and Y. Vutov. Parallel mic(0) preconditioning of 3d elliptic problems discretized by rannacher-turek finite elements. *Computers and Mathematics with Applications*, 55(10):2197–2211, 2008.

[28] T. Rauber and G. Rünger. A Transformation Approach to Derive Efficient Parallel Implementations. *IEEE Transactions on Software Engineering*, 26(4):315–339, 2000.

[29] T. Rauber and G. Rünger. Tlib - A Library to Support Programming with Hierarchical Multi-Processor Tasks. *Journal of Parallel and Distributed Computing*, 65(3):347–360, 2005.

[30] T. Rauber and G. Rünger. *Parallel Programming for Multicore and Cluster Systems, Second edition*. Springer, 2013.

[31] T. Rauber and G. Rünger. Modeling and Analyzing the Energy Consumption of Fork-Join-based Task Parallel Programs. *Concurrency and Computation: Practice and Experience*, 2014.

[32] T. Rauber, G. Rünger, M. Schwind, H. Xu, and Melzner S. Energy Measurement, Modeling, and Prediction for Processors with Frequency Scaling. *The Journal of Supercomputing*, 2014.

[33] S.Ashby, P.Beckman, J.Chen, P.Colella, B.Collins, D.Crawford, J.Dongarra, D.Kothe, R.Lusk, P.Messina, T.Mezzacappa, P.Moin, M.Norman, R.Rosner, V.Sarkar, A.Siegel, F.Streitz, A.White, and M.Wright. Report on exascale computing. Technical report, ASCAC, 2010.

[34] C.A. Smethurst, D.J. Silvester, and Mihajlović M.D. Unstructured finite element method for the solution of the Boussinesq problem in three dimensions. *International Journal for Numerical Methods in Fluids*, 73(9):791–812, 2013.

[35] Science Staff. Special Collection: Dealing with Data. *Science*, 331(6018), 2011.

[36] Stewart Tansley and Kristin Michele Tolle. *The fourth paradigm: data-intensive scientific discovery*. Microsoft Research, 2009.

[37] Top500. *Top500 supercomputers site*. Available on: http://www.top500.org, (accessed August 2014).

[38] Parkinson C.L. Washington W.M. *Introduction To Three-dimensional Climate Modeling*. University Science Books, California, 2005.

4 Appendix

Requirements for applications amenable for ultrascale computing.

As a first step of the action, the members of WG 6 have collected some important information about applications that are amenable for ultrascale computing and that could play a role for the action. The information has been collected in the form of a questionnaire. This information has been considered to be important for all working groups to pursue the issue of the action. In the following, the results of the questionnaire are given.
Name and Institution

Svetozar Margenov
Institute of Information and Communication Technologies, BAS

Application Name	Finite element supercomputer simulation of strongly heterogeneous media
Application website	http://parallel.bas.bg/SuperCA++/index_en.html
Important characteristics of your application.	Scalable implementation of FEM on advanced supercomputing architectures including hybrid platforms with GPU/MIC accelerators
Is this application the result of an interdisciplinary cooperation?	yes National Centre for Supercomputing Applications
Has there be a project funding?	yes : COST IC805 action (funding for Short missions)
On which platforms have these applications been executed?	Linux
What would be an ideal platform to execute your applications?	Linux
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	yes
Can you specify a typical data size for the application?	GB
Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	C, MPI, OpenMP, CUDA
What are critical issues for your applications?	Programmability Performance
Could you make the source code of your application available?	yes In case of collaborations in testing new solvers

Would you be interested to provide your applications for a repository developed in the NESUS Action?	no
Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	yes We would be interested in further development of efficient solvers for hybrid platforms
What is the status of your application?	Research working code + publications with Demonstration cases
Which other applications that you have not worked with would you be interested in?	Hybrid stochastic/deterministic models and related applications for uncertain data and sensitivity analysis

M.Serdar ÇELEBi, Istanbul Technical University

Which (larger) applications have you worked with or developed in your research group?
We worked with OpenFOAM and its direct solver kernels and graph partitioning algorithms.

Please, include a form for each application.

Application Name	OpenFOAM
Application website	http://www.openfoam.com
Important characteristics of your application.	Finite Volume and Finite Element Software and Library written in C++. Several advanced techniques for CFD, FSI, Turbomachinery, Aerospace etc. simulations are implemented and new techniques can be implemented in relatively shorter time
Is this application the result of an interdisciplinary cooperation?	Yes. Mechanical Engineering, Computer Science & Engineering, Mathematics and knowledge of other engineering disciplines usch as Chemistry and Physics. Computational Science and Engineering.
Has there be a project funding?	No.
On which platforms have these applications been executed?	OS: UNIX, Linux, OS X Machines: x86, amd64. But mostly Linux based large scale clusters with Intel processors.
What would be an ideal platform to execute your applications?	A Linux Cluster (RedHAT AS 4.0 and above, Debian, Ubuntu, CentOS) composed of fat nodes made of latest Intel Microprocessors with IB's latest network connectivity.
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	Yes, definitely. Multiphysics Simulations such as Fluid Structure Interaction simulations and next generation simulation technique called Multscale Modelling have huge computational and memory complexity. Therefore, with optimized software for exascale simulations, science and engineering problems can be modeled more realistically thus would give us more insight of the problems. We already run our models on Linux cluster with 32,000 cores and observed a promising scaleup.
Can you specify a typical data size for the application?	300 - 400 GB for mid-scale simulations. Over 1 TB for large-scale simulations
Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	C++, MPI, OpenMP, CUDA, OpenCL

What are critical issues for your applications?	For very large scale applications, MPI backend of the OpenFOAM library should be refactored in order to improive scalability of the simulations. Moreover, there is a potential bottleneck in some libraries used in OpenFOAM for exascale simulation. There is a strong need to identify these problems and solve them with better alternatives. Another critical issue is the acceleration of the solver kernels on GPU's and/or MIC. We have attempts to accelerate the simulations based on GPU clusters. We currently accelerating the simulations with a single node clustered GPU's and trying to extent to many node based GPU clusters.
Could you make the source code of your application available?	Yes. License of the OpenFOAM is GNU general public licence (GPL). Therefore, all contributions to the library is available. After potential improvements on the software we are planning to open it to community.
Would you be interested to provide your applications for a repository developed in the NESUS Action?	It may be.
Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	Yes. We believe that there are other groups who work on the same or related topics for OpenFOAM and we would like to collaborate with them within a broader context.
What is the status of your application?	A1 - Research working code + publications with results
Which other applications that you have not worked with would you be interested in?	Out other group is working on the sparse direct solver kernel tuned for many core distributed systems is called SuperLU_MCDT we developed. I may work with groups working on sparse iterative on hybrid solvers for large-scale heterogeneous systems.

Anastas Mishev, Faculty of Computer Science and Engineering, UKIM, Macedonia

Application Name	Simulation of multidimensional coupled anharmonic oscillators as model systems in physical sciences
Application website	
Important characteristics of your application.	- Possibility for exact modeling of realistic physical systems, on the basis of realistic representation of the complex multidimensional vibrational potential; - Notable acceleration of computations of the vibrational eigenenergies and eigenfunctions in the case of multidimensional quantum oscillators; - Enabling HPC platform transparency to computational chemistry community; - Efficient computation of numerous parameters related to the oscillating behavior of quantum physical systems
Is this application the result of an interdisciplinary cooperation?	Yes
Has there be a project funding?	No
On which platforms have these applications been executed?	Scientific Linux
What would be an ideal platform to execute your applications?	Scientific Linux, (but we plan to port our implementation to Windows as well)
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	yes, we would be very much interested in porting the application on Exascale system; however, there have been attempts to run the application in a distributed environment
Can you specify a typical data size for the application?	The application is not very large by itself; however, as the input consists of a multidimensional potential energy hypersurface computed on a suitably chosen grid of points, while the output could consist of both eigenenergies and eigenfunctions of the systemâĂŹs Hamiltonian, along with other parameters (e.g. average values of certain structural parameters computed by quantum mechanical averaging procedures), both the input and output could in principle easily exceed few GBs; these are, of course, simulation dependent.
Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	Fortran, C, OpenMP, MPI

What are critical issues for your applications?	The most critical issue in a fundamental sense is to get accurate eigenvalues of the high energy levels, which are close to the top of the barrier; in computational sense, the most critical issue is related to appropriate representation of multidimensional potential energy hypersurfaces.
Could you make the source code of your application available?	Yes, the implementation is aimed to be open source.
Would you be interested to provide your applications for a repository developed in the NESUS Action?	Yes
Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	Yes 1. The final goal of our research efforts is to make a robust but flexible framework code (possibly with modular structure) with could enable exact modeling of multidimensional quantum mechanical oscillators. 2. Besides providing of particular implementations, we would also like to contribute to development of fundamental algorithms relevant to the mentioned issue. 3. We would also aim to make the code as efficient as possible on a HPC distributed memory environment.
What is the status of your application?	The source codes related to the multidimensional calculations are in beta-phase; particular codes e.g. for 1D and 2D systems have already been used for productive calculations; some publications based on such operational codes are in preparation; we also plan to publish the particular implementations in specialized journals, aside from publications related to particular physical problems to which the algorithms and codes will be applied.
Which other applications that you have not worked with would you be interested in?	Network science as well as weather science computational tools, as well as codes related to signal analysis and time series analysis.

Sonja Filiposka, Faculty of Computer Science and Engineering, UKIM, Macedonia

Application Name	GPU accelerated wireless network simulation in 3D Terrain Using GPUs
Application website	http://e-tnc.com/etnc/Research/3Dpropagationextensionfor NS2/tabid/104/Default.aspx
Important characteristics of your application.	- Utilization of high-end GPU devices - Significant acceleration of the network simulation process - Introducing HPC platform transparency to the ordinary network simulation modeler - Utilization of standardized GIS terrain technology - Efficient point location in vector terrains
Is this application the result of an interdisciplinary cooperation?	No
Has there be a project funding?	Partial funding from NVidia for opening an NVidia center at the faculty for which a number of GPUs were provided by NVidia
On which platforms have these applications been executed?	Linux, Ubuntu
What would be an ideal platform to execute your applications?	Linux, Ubuntu (we are also planning to port our implementation to Windows and Mac)
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	yes, we are very interested, there are already attempts to run the application in a distributed environment
Can you specify a typical data size for the application?	The application itself is less that 1GB, however the input and output data easily exceed 1GB, and are simulation dependant.
Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	C, OpenMP, OpenCL, CUDA, MPI
What are critical issues for your applications?	Performance in the sense of running simulation scenarios and getting results fast
Could you make the source code of your application available?	Yes, the implementation is open source.
Would you be interested to provide your applications for a repository developed in the NESUS Action?	Yes

Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	Yes 1. Our goal is to further develop our implementation to support many other modules for wireless network simulation using different communication protocols for different terrain scenarios. 2. We would like to develop algorithms for maximizing field coverage. 3. We would like to continue with the migration of our implementation to multi-GPU HPC environment. 4. We would like to optimize our implementation in order to make it more energy efficient. 5. We plan to utilize and develop more efficient and smart data structures for terrain representation, and point location algorithms.
What is the status of your application?	The research source code is operational, we have published results for the implementation, and we have some publications in editing. http://e-tnc.com/etnc/Research/3DpropagationextensionforNS2/tabid/104/Default.aspx
Which other applications that you have not worked with would you be interested in?	Computational Chemistry tools and simulators that utilize new parallel and heterogeneous platforms.

Jing Gong, PDC Center for High Performance Computing, KTH Royal Institute of Technology

Application Name	OpenACC acceleration for Nek5000: spectral element method
Application website	nek500.mcs.anl.gov
Important characteristics of your application.	OpenACC, Nek5000, Spectral Element Methods, Computational Fluid Dynamics
Is this application the result of an interdisciplinary cooperation?	Yes KTH Department of Mechanics, PDC-HPC Argonne National Laboratory Cray Company and EPCC
Has there be a project funding?	EU project: CRESTA The Swedish e-Science Research Center (SeRC)
On which platforms have these applications been executed?	IBM BG/Q, Mira, 1 Million cores Cray XK7, Titan, 16,384 GPUs
What would be an ideal platform to execute your applications?	Cray, IBM
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	Cray XK7 or XC30 system with $>$ 10,000 GPUS
Can you specify a typical data size for the application?	TB
Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	Hybrid C/Fortran77 Original only MPI and Now MPI/OpenACC
What are critical issues for your applications?	Performance
Could you make the source code of your application available?	yes
Would you be interested to provide your applications for a repository developed in the NESUS Action?	yes
Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	yes 1) Autotuning and optimization of kernels 2) Profiling and debug tools
What is the status of your application?	A1 - Research working code + publications with results

Which other applications that you have not worked with would you be interested in?	OpenFoam

Neki Frasheri, Center for Research and Development in IT, Polytechnic University of Tirana

Application Name	Geophysical Modeling and Inversion
Application website	http://itc.upt.al/hp-see/
Important characteristics of your application.	Approximation based on relaxation methods
Is this application the result of an interdisciplinary cooperation?	Yes; computer sciences, applied mathematics and geophysics, with the involvement of specialists from Faculty of Geology and Mining and of Academy of Sciences in Albania
Has there be a project funding?	EU project: FP7 HP-SEE
On which platforms have these applications been executed?	HPCG Cluster of Institute of Information and Communication Technologies, Academy of Sciences of Bulgaria, and SGE system of the NIIFI Supercomputing Center at University of Pécs, Hungary (max 1000 cores)
What would be an ideal platform to execute your applications?	To be defined
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	Yes; the runtime depends on the model size in the best case at the range of $O(N^8)$ and in small parallel systems is not possible to run high resolution models
Can you specify a typical data size for the application?	TB
- Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	Standard C and MPI
What are critical issues for your applications?	Runtime
Could you make the source code of your application available?	yes
Would you be interested to provide your applications for a repository developed in the NESUS Action?	yes
Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	yes
What is the status of your application?	A1 - Research working code + publications with results

Which other applications that you have not worked with would you be interested in?	void

Maya Neytcheva, Department of Information Technology, Uppsala University

Application Name	Numerical simulation of poro-viscoelastic problems as arising in Glacial Isostatic Adjustment models
Application website	http://user.it.uu.se/∼maya/Projects/GIA/index.html
Important characteristics of your application.	Large scale time-dependent finite element model. Scalability and reliability on advanced supercomputer architectures including hybrid platforms with GPU/MIC accelerators is a major issue.
Is this application the result of an interdisciplinary cooperation?	Yes, Department of Geophysics, Uppsala University, Sweden, Université Libre de B ruxelles, Belgium
Has there be a project funding?	A PhD student is funded by the Swedish Scientific foundation.
On which platforms have these applications been executed?	Linux
What would be an ideal platform to execute your applications?	Linux
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	Yes
Can you specify a typical data size for the application?	The application is computationally intensive
Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	C, C++, Fortran, MPI, OpenMP, CUDA
What are critical issues for your applications?	Programmability, scalability, Performance
Could you make the source code of your application available?	Yes In case of collaborations in testing new solvers
Would you be interested to provide your applications for a repository developed in the NESUS Action?	No, we do not have resources for that.
Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	Yes; We already have performance results on CPU and GPU and would be interested in further development of efficient solvers for hybrid platforms.

What is the status of your application?	Research working code and a publication with comparison results http://www.it.uu.se/research/publications/reports/2014-007/
Which other applications that you have not worked with would you be interested in?	Inverse problems, uncertainty quantification, machine learning

Horacio Pérez-Sánchez
Bioinformatics and High Performance Computing Research Group Universidad Católica San Antonio de Murcia (UCAM), SPAIN

Application Name	BINDSURF
Application website	http://bio-hpc.eu/software/bindsurf/
Important characteristics of your application.	GPU application (CUDA) for the discovery of novel bioactive compounds.
Is this application the result of an interdisciplinary cooperation?	no
Has there be a project funding?	yes : - Fundación Séneca de la Región de Murcia (18946/JLI/13) - Nils Mobility 012-ABEL-CM-2014A - DECI-10 PRACE, "Discovery of Novel Anticoagulants"
On which platforms have these applications been executed?	Linux, Windows
What would be an ideal platform to execute your applications?	Linux, Windows
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	yes
Can you specify a typical data size for the application?	1 GB
Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	C, MPI, OpenMP, CUDA
What are critical issues for your applications?	Programmability Performance
Could you make the source code of your application available?	yes In case of collaborations in testing new solvers
Would you be interested to provide your applications for a repository developed in the NESUS Action?	no
Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	yes

What is the status of your application?	Research working code + publications with Demonstration cases
Which other applications that you have not worked with would you be interested in?	Other molecular modeling and structural bioinformatics applications

Ricardo Morla, INESC Porto and U.Porto

Application Name	Scalable network traffic analysis. This is not my application but open source code from several research groups working on network traffic analysis.
Application website	Open-source code https://github.com/ssallys/p3 https://github.com/RIPE-NCC/hadoop-pcap
Important characteristics of your application.	TCP/UDP flow information extraction from packet data, traffic classification, anomaly detection, traffic matrix computation
Is this application the result of an interdisciplinary cooperation?	no
Has there be a project funding?	no
On which platforms have these applications been executed?	Hadoop
What would be an ideal platform to execute your applications?	Hadoop
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	yes
Can you specify a typical data size for the application?	n **Gbit/s**, where n can range from 1 to 100 depending on the size and number of links in the network
Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	Hadoop
What are critical issues for your applications?	Real-time performance, variable load, efficient resource usage, data access
Could you make the source code of your application available?	Does not apply; itâĂŹs open source
Would you be interested to provide your applications for a repository developed in the NESUS Action?	Does not apply; itâĂŹs open source
Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	Yes
What is the status of your application?	Demo.

Which other applications that you have not worked with would you be interested in?	Not sure

Tomas Fryza
SIX Research Center, Department of Radio Electronics, Czech Republic

Application Name	A logical verification of combinatorial problems in security systems
Application website	–
Important characteristics of your application.	Verification of combinatorial problems, which representing the robustness of the code schema in security systems.
Is this application the result of an interdisciplinary cooperation?	Yes: CNS company.
Has there be a project funding?	Yes: *New technologies of data security by automated independent audit*, Ministry of the interior of the Czech Republic.
On which platforms have these applications been executed?	Linux, CentOS, Windows
What would be an ideal platform to execute your applications?	Linux
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	Yes
Can you specify a typical data size for the application?	16 GB
Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	C, MPI, Matlab
What are critical issues for your applications?	Programmability, memory demands.
Could you make the source code of your application available?	I am afraid not...do not have permissions from the project provider.
Would you be interested to provide your applications for a repository developed in the NESUS Action?	I am afraid not...do not have permissions from the project provider.
Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	Yes
What is the status of your application?	Research working code, demonstration cases for simpler security schemas.

Which other applications that you have not worked with would you be interested in?	Video signal processing, source code optimizing.

Lars Ailo Bongo, HPDS, Dept. of Computer Science and Center for Bioinformatics, University of Tromsø, Norway

Application Name	METAPipe: metagenomics data analysis pipeline
Application website	https://nels.bioinfo.no/
Important characteristics of your application.	BLAST, MEGAN, MetaPhlAn, Glimmer, MGA, Krona, METARAEP, custom tools, bioinformatics databases
Is this application the result of an interdisciplinary cooperation?	Yes; Dept. of Chemistry, University of Tromsø, Elixir Norway
Has there be a project funding?	NELS (Norwegian eInfrastructure project)
On which platforms have these applications been executed?	Small hadoop cluster (80 cores) Stallo, 4864 cores (Intel Xeon)
What would be an ideal platform to execute your applications?	Hadoop cluster
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	Not yet
Can you specify a typical data size for the application?	100s of GB - TB
- Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	Perl, C++, Java, HTML5, Python, RâĂę
What are critical issues for your applications?	Performance, resource usage
Could you make the source code of your application available?	yes (most tools in pipeline are already open source)
Would you be interested to provide your applications for a repository developed in the NESUS Action?	
Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	yes 1) Resource and data management 2) Profiling and debug tools
What is the status of your application?	Approaching production readiness
Which other applications that you have not worked with would you be interested in?	

Roman Wyrzykowski
Institute of Computer and Information Sciences, Czestochowa University of Technology

Application Name	EULAG numerical model
Application website	-
Important characteristics of your application.	Numerical solver for all-scale geophysical flows adapted to GPU and Intel Xeon Phi accelerators. The underlying anelastic equations are either solved in an EULerian (flux form), or a LAGrangian (advective form) framework.
Is this application the result of an interdisciplinary cooperation?	yes; Poznan Supercomputing and Networking Center; Institute of Meteorology and Water Management in Warsaw
Has there be a project funding?	yes: This application is supported by the Polish National Science Center under grant no.UMO-2011/03/B/ST6 /03500
On which platforms have these applications been executed?	Linux
What would be an ideal platform to execute your applications?	Linux
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	yes
Can you specify a typical data size for the application?	20 GB
Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	C++, Fortran, MPI, OpenMP, CUDA
What are critical issues for your applications?	Memory-bounded nature of computation within a single node, scalability on clusters (limitations of communication performance in MPDATA, global communication in GCR elliptic solver)
Could you make the source code of your application available?	yes In case of collaboration in testing and developing the code
Would you be interested to provide your applications for a repository developed in the NESUS Action?	no

Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	yes We would be interested in further development of EULAG model for GPU, CPU and Intel Xeon Phi platforms
What is the status of your application?	Research working code + publications with demonstration cases
Which other applications that you have not worked with would you be interested in?	Big data analytics

Anatoliy Melnyk
Department of Computer Engineering, Lviv Polytechnic National University

Application Name	Self-configuration in exascale heterogeneous FPGA-based computer systems
Application website	http://eom.lp.edu.ua/science-e.php
Important characteristics of your application.	Implementation of self-configuration in exascale heterogeneous FPGA-based computer systems; Implementation of automatic load balancing; Implementation of soft-cores automatic generation and synthesis.
Is this application the result of an interdisciplinary cooperation?	No
Has there be a project funding?	We applied for funding from the Ministry of Science and Education of Ukraine. The result will be known in December 2014.
On which platforms have these applications been executed?	Windows, Linux
What would be an ideal platform to execute your applications?	Windows, Linux
Would you be interested to run your applications on a really large parallel system (Exascale) or distributed (cloud)?	Yes. We see here two directions: 1. running of soft-cores automatic generation and synthesis system on exascale computer system, including using it as a cloud; 2. application of our new model of computation in exascale computer systems.
Can you specify a typical data size for the application?	Application is computationally intensive.
Which programming methods and environments have been used to develop the application (such as Fortran, C, MPI, OpenMP, ...)	C, MPI, OpenMP, Open CL, VHDL, Verilog
What are critical issues for your applications?	Efficiency, Performance
Could you make the source code of your application available?	Yes, in the part of joint research.
Would you be interested to provide your applications for a repository developed in the NESUS Action?	Yes, in the part of joint research.

Do you see potential for a collaboration in the context of the NESUS project to pursue the development of your application?	Yes. We would be interested in further development of efficient self-configurable exascale heterogeneous FPGA-based computer systems in the context of the NESUS project.
What is the status of your application?	We already have developed the theoretical part of application and have developed and tested its main software components.
Which other applications that you have not worked with would you be interested in?	Cloud computing