DOI 10.12694/scpe.v16i4.1134
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 4, pp. 467–489. http://www.scpe.org

ON PROCESSING EXTREME DATA [1]

DANA PETCU, GABRIEL IUHASZ, DANIEL POP[2], DOMENICO TALIA[3], JESUS CARRETERO[4], RADU PRODAN, THOMAS FAHRINGER, IVAN GRASSO[5], RAMÓN DOALLO, MARíA J. MARTíN, BASILIO B. FRAGUELA[6], ROMAN TROBEC, MATJAŽ DEPOLLI[7], FRANCISCO ALMEIDA RODRIGUEZ, FRANCISCO DE SANDE[8], GEORGES DA COSTA, JEAN-MARC PIERSON[9], STERGIOS ANASTASIADIS, ARISTIDES BARTZOKAS, CHRISTOS LOLIS[10], PEDRO GONÇALVES, FABRICE BRITO[11], AND NICK BROWN[12]

Abstract. Extreme Data is an incarnation of Big Data concept distinguished by the massive amounts of data that must be queried, communicated and analyzed in near real-time by using a very large number of memory or storage elements and exascale computing systems. Immediate examples are the scientific data produced at a rate of hundreds of gigabits-per-second that must be stored, filtered and analyzed, the millions of images per day that must be analyzed in parallel, the one billion of social data posts queried in real-time on an in-memory components database. Traditional disks or commercial storage nowadays cannot handle the extreme scale of such application data.

Following the need of improvement of current concepts and technologies, we focus in this paper on the needs of data intensive applications running on systems composed of up to millions of computing elements (exascale systems). We propose in this paper a methodology to advance the state-of-the-art. The starting point is the definition of new programming paradigms, APIs, runtime tools and methodologies for expressing data-intensive tasks on exascale systems. This will pave the way for the exploitation of massive parallelism over a simplified model of the system architecture, thus promoting high performance and efficiency, offering powerful operations and mechanisms for processing extreme data sources at high speed and/or real time.

Key words: Extreme Data, HPC, Exascale Systems, Extreme Computing, Parallel Programming Models, Scalable Data Analysis

AMS subject classifications. 65Y05, 68W10

1. Introduction.

Inheriting the characteristics of Big Data, Extreme Data has at least three main dimensions, Volume, Variety and Velocity. However, the Extreme attribute is associated with a virulence of the Vs and their strong inter-relationship. The Volume dimension is well explained by the Big Data concept describing the data size or weight. However, in the context of scientific computing, the extreme data processing has a high degree of complexity due to the tighter interactions among different parts of the dataset during processing. As a result, the volume that can be processed is currently limited and existing abstractions such as the MapReduce model applicable to Big Data are insufficient. New programming paradigms are needed to be specially tailored to increase the volume of the scientific data processed by HPC systems.

The processing is complicated by data Variety: e.g. the simple requirement of adding and processing data from various streaming sources is not yet supported by the current HPC tools in the context of new types appearing every day. Furthermore, the challenge is not only storing the data, but also getting data in and out of store from the data centers and offering support for reproducibility and data provenance assurance.

The Velocity express how rapidly one can move data from a data volume to a user. Extreme velocity is crucial for real-time advanced data visualization and analytics (embedded analytics into the databases is the only current solution).

[1]This work is partially supported by EU under the COST Program Action IC1305: Network for Sustainable Ultrascale Computing (NESUS).
[2]West University of Timişoara, Romania
[3]University of Calabria, Italy
[4]University Carlos III of Madrid, Spain
[5]University of Innsbruck, Austria
[6]University of Coruña, Spain
[7]Jozef Stefan Institute, Slovenia
[8]University of La Laguna, Spain
[9]IRIT, University of Toulouse, France
[10]University of Ioannina, Greece
[11]Terradue, Italy
[12]EPCC, University of Edinburgh, Scotland, UK

In addition to the above three dimensions inherited from Big Data, we consider relevant for extreme data a fourth dimension, Value. The Value can be obtained by analyzing datasets and so extracting knowledge from them. Part of the research work is dedicated to the design and development of methods and tools for programming extreme data mining applications and extracting value from extreme data. The Value is mentioned also as a fourth dimension in High Performance Data Analysis (HPDA, [18]) concept. HPDA needs HPC resources and smart algorithms to offer solutions in near-real time, implies search and pattern discovery, simulation and analytics, and deals with regular and irregular data patterns, as well as partitionable and non-partitionable problems.

In this paper we propose a roadmap to developing new programming models for extreme data applications and ensure their take-up. More precisely, in order to offer support to the extreme data processing through HPC, we propose to change the current programming paradigms by following a data-driven model (to increase the Volume), the heterogeneous data access mechanisms (to increase the Variety), and the methods for data analytics/mining (to increase the Value). The scale of the changes on the three dimensions (Volume, Value, Variety) should be finally estimated through the redesign and reimplementation of existing scientific applications, dealing with large volumes of data. We are not discussing in this paper how the velocity can be increased, as being currently subject of co-design techniques.

The proposed roadmap is aligned with the challenges identified by PRACE [75], HiPEAC [39], PlanetHPC [74] and ETP4HPC [28]. As underlined by the PRACE report "The Scientific Case for High-performance Computing in Europe 2012–2020", handling large data volumes generated by research is both a major challenge and opportunity for future HPC systems, and integrated environments for compute and data are expected to emerge. Another identified challenge is the end-to-end management of, and fast access to, large and diverse data sets through the infrastructure hierarchy; most application areas foresee the need to run several long-time jobs at sustained high performance to generate core datasets and many shorter-time jobs at lower performance for pre- and post-processing. The HiPEAC initiative recommends to address the issues of efficiency (maximizing power efficiency and performance through heterogeneous computer system design and catering for data locality), dependability and applications (bridging the gap between the growth of data and processing power; safety, predictability and ubiquitous availability of systems). The main technological challenges identified by PlanetHPC are data locality (i.e., the proximity of data to the processing location), new programming models and tools (for massively parallel and heterogeneous systems), technologies supporting new and emerging applications that require robust HPC with real-time capability, data-intensive HPC and low-energy computing from both an architectural and application perspective. ETP4HPC SRA (Strategic Research Agenda) underlines that important trends or needs are efficient communication and data movements at all levels and on different flavors and levels of architectures.

The Japanese JST-CREST on-going programme Extreme Big Data (EBD) aims to achieve the convergence of Supercomputing and Big Data [63]. EBD is now focusing on the hierarchical use of new generation non-volatile memories and processor-in-memory technologies, hierarchical management of memory objects and their high resiliency, and resource management to accommodate complex workflows being able to minimize data movement. EBD advocates the need for a convergent SW/HW architecture and their software stack will be implemented in the next generation of Japanese supercomputers (including also new strategies for scaling on the Velocity dimension using new interconnection network topologies). The co-design EBD applications include climate simulation with real-time data assimilation to predict extreme weather phenomena.

In this paper we advocate for an extended level of support for the development, deployment and execution of extreme data application using the current HPC facilities, while we do not exclude the fact that co-design techniques can follow the proposed approach. The four pillars of the proposed roadmap are the following:

1. Design and develop new HPC programming and energy models for extreme data applications. Designing and developing programming paradigms and mechanisms is needed to better support the implementation of scalable algorithms and applications on exascale computing systems. Models for energy consumption estimation are needed at an application level, as well as efficient runtime mechanisms for supporting the workload executions expected to scale on very large computing systems, including support to cope with the dynamic variability in the required or available resources.
2. Build new tools for monitoring and data-analysis for extreme data applications. Designing and de-

veloping scalable monitoring and data analysis tools should include low-level instrumentation, data collection, mining and data-centric online performance analysis with the ability to perform at the exascale level. Beyond the use of performance metrics, the efficiency needs to be targeted through advanced modelling and correlation heuristics aiming to understand their inter-relationships, essential for online optimisation and tuning.

3. Adapt the data management and storage techniques to the extreme scale. Providing high-performance and reliable support for I/O, storage and data-throughput, as well as resilience capabilities are needed. The strategies to enhance an application's data locality need to be coupled with dynamic I/O configurations by restructuring the I/O stack functionalities, and by providing predictive and adaptive data allocation strategies. The resilience problem in the I/O stack needs to be treated by sharing cross-cutting issues and highly scalable and energy-efficient cross-layer checkpointing strategies, using the I/O hierarchy and memory storage greedily. Object-based file systems and scalable data stores need to be promoted for efficient and durable storage structures for exascale applications.
4. Validate the concepts and tools usefulness through extreme data applications. Two main directions are considered in this paper: environmental science (particularly Earth observation, weather forecasting and urban computing fields) and dynamic systems simulations.

Section 2 discusses how the state of the art in processing large datasets should be advanced in order to reach the exascale dimension. Section 3 introduces an approach to the proposed advances. We draw the conclusions in the closing section.

2. Advancing the state of the art. We follow in this section the topics related to exascale systems from the ETP4HPC Strategic Research Agenda [28], selecting those connected to the extreme data processing field:

1. Programming Environments: Non-conventional parallel programming; APIs for auto-tuning performance or energy; Inspection of data locality at Exascale level; New metrics, analysis techniques and models;
2. System Software and Management: On-the-fly data analysis, data mining; Fault-tolerant MPI and checkpointing;
3. Balancing Compute Subsystem, I/O and storage performance: Hardware resilient Exascale storage demo; Big Data + HPC integrated software stack;
4. Big Data and HPC Usage Models: Problem-solving environments for large data.

More precisely, in this section we try to identify the limitations of the current approaches for data processing using HPC in the conditions of Extreme Data, as well as potential paths to overpass these limitations.

2.1. Advancing the Programming Environments.

2.1.1. Non-conventional parallel programming. A parallel programming model provides a set of abstractions that simplifies and structures the way a programmer thinks about and expresses a parallel algorithm. High-level programming models help programmers access and use resources abstracted from physical entities (cores or memory). This is done to facilitate programming and to improve the software portability and to increase the productivity. The use of high-level programming abstractions also helps by using a reduced set of programming patterns that limits errors, reduces programming time and facilitates resource exploitation.

Programming paradigms traditionally used in parallel and distributed systems, like MPI, Map-Reduce, OpenMP, and OpenCL are not sufficient for developing codes designed to run on systems composed of a massive amount of processing nodes. To reach exascale requires the definition of new programming paradigms combining abstraction with scalability and performance. Hybrid approaches (shared/distributed memory) and subsets of communication mechanisms based on locality and grouping are currently investigated. The approaches proposed until now are based on the adaptation of traditional parallel programming languages and hybrid solutions. This incremental approach is too conservative, often resulting in very complicated code with the risk of limiting the scalability of programs on millions of cores.

Approaches based on a partitioned global address space (PGAS) memory model appear to be more suited in meeting the exascale challenge. Today, they are investigated as a potential solution that may limit the cost of shared memory access, however load balancing is still an open question. Many current parallel programming solutions contain an implicit trade-off between simplicity and performance; those which abstract the programmer

from the lower level parallel details often sacrifice performance and scalability in the name of simplicity, and vice versa. Many current implementations of the PGAS model are no exception to this and unfortunately, limited progress has been done in last few years toward the implementation of simple models that can be used to program at any scale and performance. The need for scalable programming models continues and the emergence of new hardware architectures makes this need more urgent.

MPI offers low programmability due to the fragmented view of data it offers. This resulted in the proposal of the PGAS model, whose current implementations (e.g. UPC [95], CAF [73], Global Arrays [71]) inherit the SPMD model due to their multi-threaded view and they have no support for hierarchical decomposition, which is key for exascale computing. Additionally, these tools only focus on communication, leaving to the user all the issues related to the management of large volumes of data that appear in exascale problems.

Novel, complex abstract structures are needed for extreme data application support. The MapReduce model is often oversold; although frequently used on clusters and Cloud-based applications, additional research is needed to develop scalable higher-level models and tools. Defining new programming paradigms and APIs to represent data-intensive tasks on exascale systems will allow the exploitation of massive parallelism in a way that provides an abstraction of the system architecture, promotes the high performance and efficiency, and offers powerful operations and mechanisms for processing extreme data in reduced and/or real time. Such strategy can open the way to implement many data-intensive applications in several application domains both in science and business.

Heterogeneity is crucial to achieve exascale computing and it is in fact being increasingly adopted in high-performance petascale systems, as the first entries of the TOP500 show. For instance, the June 2014 list includes 62 systems with accelerators, a 17% raise comparing to 53 in the previous list. These platforms require the cooperation of thousands of nodes, with MPI being the state-of-the-art, sometimes complemented by other tools such as OpenMP for intra-node communication, completed by a variety of APIs for managing the accelerators, OpenCL being the most portable solution.

There are previous attempts for novel data processing languages such as DISPEL [20], which is an imperative workflow-based scripting language aimed at supporting analysis of streamed data. The programmer describes workflows for data intensive applications, a workflow being a knowledge discovery activity, where data is streamed through and transformed into higher level knowledge. Apart from requiring the data scientist to learn a variety of new concepts, this language is heavily integrated with the ADMIRE platform and requires the features and workflow provided by the platform. The use of DISPEL is supported in the seismology e-science data analysis environment of the VERCE [96] project. DISPEL was ported from the ADMIRE platform to Python and dispel4py [21] was created, which shares the same concepts as DISPEL but it is integrated with a platform familiar to seismologists. This approach of building abstractions on top of an already familiar programming language has many advantages, such as an easy integration with existing tools, or a wider acceptance by community. The R [78] language is designed for statistical computing and graphics, one of its strengths being the integrated support for data manipulation and calculation; for instance, effective data storage and calculations on matrices. As a functional language, R might require, to some extent, the data community to learn new concepts, which may impact its acceptability. The SPRINT [85] project has developed an easy to use parallel version of R, aimed at the bioinformatics community. This technology allows the addition of parallel functions, without requiring in depth parallel programming knowledge. SPRINT has been very successful in this domain, but it is not designed for exascale and, although it works well on a thousand of cores, it is not designed to scale up to hundreds of thousands or millions of cores.

We argue that libraries are needed to support abstract data types (ADTs) that represent global explicitly-distributed objects in a widely spread object-oriented language (the main focus can be on arrays because of their ubiquitousness). Since objects contain both data and the key semantics of their management, the methods of ADT can provide users with optimized implementation of common operations and underlying support for requirements for exascale applications. The ADT methods can provide data-parallel semantics, and can encapsulate abstractions that allow for hierarchical decomposition of the data and the representation of irregular problems and operations. The meta-information within ADTs, together with the high-level semantics of its operation, can allow one to optimize its operation in exascale environments while strongly reducing the code complexity. Finally, the usage of a widely-adopted language can favor code reuse and the gradual migration

of legacy code to exascale systems on top of these ADTs.

The availability of many accelerators (GPUs, Phi, FPGAs etc.) with different APIs to manage them and no clear winner at this point leads to possibility of many more appearing in the future. The usage of OpenCL in exascale applications, which is the only portable approach that avoids lock-ins and code rewrites, seems to be a promising choice. Unfortunately, the OpenCL API is too low level [72], resulting in code verbosity and low productivity of programmers, who need to manually perform buffer allocations, data transfers, synchronizations etc. High level annotations [81, 2] or embedded languages [97] built on top of OpenCL that automate these tasks and hide their complexity are a better approach. Several authors have considered the programmability of accelerators on top of OpenCL across a cluster [7, 35]; however, existing solutions only abstract away some complexities of OpenCL but still expose other low-level details or restrict the non-accelerator part of an application to a single node.

In order to exploit all the available hardware in exascale systems, we need to extend and evolve the high-level programming of accelerators (e.g., [81, 97, 2]) into high-productivity seamless approaches, for instance, through the natural integration with the ADT previously described.

2.1.2. APIs for auto-tuning performance or energy. At the moment, most monitoring tools of HPC systems directly measure only metrics of performance. Classical monitoring tools, such as Ganglia or Nagios, track the processor and memory load, the network bandwidth and the communication pattern. But this approach can only improve one key metric, the raw performance. In a more realistic world, multiple conflicting objectives need to be achieved. Indeed, in some cases the improvement of performance or energy consumption is a difficult choice because the optimization of the one may have a negative impact on the other. In order to reach an efficient state, any runtime for optimizing energy and performance requires in-depth information about the system state along with the ability to predict the impact of possible choices. Several tools already exist [19, 9] to evaluate the power consumption of servers or applications without specialized hardware, only using information of resource consumption at the system level. However, the current solutions either are imprecise or consume resources intensively. In an HPC environment, we need to optimize both the performance and energy through the improvement of current approaches. One way is to design and implement adaptive prediction tools. Indeed, a decision system for auto-tuning the performance and energy needs to predict the impact of its decision in order to provide efficiency. At the moment, most models [62] are only considering instantaneous power consumption, while not trying to predict its evolution.

2.1.3. Inspection of data locality at Exascale level. Data locality is a major concern for I/O in current Petaflop machines. Nowdays, data locality is managed independently at various levels such as the application, middleware, or file system. At the lowest level of the I/O stack, parallel file systems such as GPFS [42], Orange/PVFS [77] or Lustre [60] manage block-based or object-based storage through parallel servers running on storage nodes. These parallel file systems strive to comply with the POSIX requirements. However, one main critique for POSIX is that it does not expose data locality, even though researchers agree that locality-awareness is a key factor for building exascale systems [79].

Other scalable file systems such as GFS [34] have dropped the POSIX interface and been co-designed with processing frameworks such as MapReduce [22]. Although these frameworks simplify programming for a class of embarrassingly parallel data-intensive computations based on best-effort strategies to co-locate computation and data, they are not generic enough to efficiently address the needs of scientific applications and have not been widely adopted on the High End Computing (HEC) infrastructure. The main take-outs from their experience are co-design and locality-aware computing [24].

The most widely used middleware in HEC systems is the MPI-IO [68]. The standard MPI-IO data operations are not locality-aware. Additionally, there are no mechanisms allowing data accesses to take advantage of data layout in the file system (e.g. I/O caching, data distribution over servers, block alignment). Some applications are using high-level I/O libraries, such as HDF5 [37] and ParallelNetCDF [55] that map application data models to storage data models through MPI-IO. However, these libraries often require experience to achieve reasonable performance and scalability.

2.1.4. New metrics, analysis techniques and models. Over the last years, a large number of performance measurement and analysis tools (like the ones described in [82, 67, 65]) have been created for the new

heterogeneous, many-core, parallel computing architectures and systems. These tools support low-level instrumentation, performance measurement and analysis; they are based on hardware counters, time measurements, call graphs, and tools for tracing the operating system or measuring the performance of specific programming models; and they cover, for instance, multi-threaded, OpenMP, and MPI programs. A major drawback of these tools is that they operate at the process or thread level which will clearly do not scale to the exascale level.

In order to avoid such major bottlenecks in exascale systems, we argue that a new tool design is needed shifting the focus from single thread and regions to the distribution and layout of data.

The first issue is whether performance analysis should be done in an online or offline manner. The online approach implies that the application is still executing while performance data is processed to support the on-the-fly optimization of an application for given objectives. This is usually considered more difficult in comparison to offline analysis, since the analysis consumes resources in parallel to the investigated application (possibly causing perturbation). A further difficulty is the processing of data streams rather than full data sets, i.e. the analysis must work on a subset with missing —potentially important— information that might appear later. Offline analysis on the other hand does not cause any performance perturbation for the application and has the entire data set available. However, it usually requires more storage space and inherently lacks support for on-the-fly optimizations. Existing tools mostly focus on offline analysis, while online analysis is only addressed in isolated efforts [67].

An important issue involved in performance analysis at the exascale level is the reduction of the amount of data involved. Ideally, we should minimize the required storage space —and the amount of input data for analysis— at maximum retained information. There are several methods and techniques that try to balance this trade-off such as clustering of data [32, 57] or pattern extraction [31]. Thus, they reduce the amount of performance data but provide a higher level of information in comparison to other techniques of performance instrumentation and measurement, such as sampling or profiling, that only give summary information. Nevertheless, few tools employ such techniques by default and, despite their applicability, they have never been used for online performance analysis.

The lossless compression methods that are widely available may have unsuitable compression ratio or excessive overhead, and do not affect the amount of data that needs to be analyzed. In addition, some tools like TAU [82] or Paradyn [67] support self-limitation of instrumentation perturbation, e.g., by estimating the overhead caused by instrumentation function calls and stopping instrumentation if that overhead reaches a certain threshold. By decreasing the instrumentation and analysis overhead, this also increases the scalability of the tools. Although this approach is successful in reducing the amount of produced performance data, it also limits its usefulness because there might be program phases that are not represented by any performance data. Once the data is analyzed, the issue of sophisticated visualization remains. In particular, subgraph folding [80] addresses the visualization of large-scale data by merging multiple subgraphs of similar information into a single, composite subgraph.

Measurement and analysis tools for non-functional parameters (NFPs, including execution time, memory usage and energy consumption) should be analyzed. Holistic views of NFPs (including a novel data-centric view to replace the traditional region-centric ones) characterizing applications, components, runtime system and computing system behaviour are needed. The tools need to correlate NFPs across these views. Measurement techniques need to ensure correct and accurate mappings of NFPs from the computing system level to the application level. In particular, measurement techniques should be selectively used for different parts of applications and computing systems to control the measurement overhead and accuracy.

In order to support dynamic performance optimization at exascale, scalable measurement and analysis techniques should target the on-the-fly classification and frequent pattern algorithms, NFP estimation techniques, and comprehensive analysis patterns for detecting and understanding performance problems. Tools that enable domain experts to control the granularity and accuracy of performance analysis are needed.

There are also approaches that compute the Pareto frontier to represent the correlation between different conflicting objectives. Examples targeting two objectives refer to time and operational cost [104] or to execution time and energy consumption [76] but without considering costs and utilisation. For instance, [12] employs the hyper-volume metric for pruning the trade-off solutions generated in each step, at cost that is exponential with the number of objectives. On the other hand, genetic algorithms can provide better solutions [100, 89]; however,

it is well-known that their performance unacceptably degrades beyond three objectives [51]. Few approaches target three objectives, like the ones from [49, 83]. All of them are nature-inspired algorithms that require a long time for convergence to high-quality solutions. All these approaches do not target data-intensive applications and do not apply energy modelling and prediction techniques.

Another important aspect to be considered is the use of GPGPU (CUDA, OpenCL) for many multi-objective optimization problems. GPGPUs are attractive for their processing power and cost effectiveness [64]. Their main drawback is that not all methods are suitable for this type of architecture and most algorithms need to be redesigned and reimplemented. Different optimization techniques have been implemented for a variety of problem types [58, 59] and also for real world applications with stringent time and computational constraints [93]. The advantages and disadvantages of this approach need to be analyzed, when applied to exascale problems and data.

2.2. Advancing the System Software and Management.

2.2.1. On-the-fly data analysis and data mining. Data analysis tasks usually involve a large amount of variables that are recorded or sampled. The problems caused by anomalous values in exascale data can be disastrous from a data-analytic point of view. In order to obtain a coherent analysis, the detection of outlying observations is paramount. Outliers can in some instances carry important information (e.g. a century flooding event with occurring probability of 1%). However outliers depend heavily on the nature of the data and the expected outcome of the analysis. These outliers are candidates for potentially aberrant data that ultimately can lead to poor sample-based performance or model degradation/misspecification. Outliers detection should happen before modeling and data analysis. However, not only pre-processing (in particular outliers detection) but also stream data analysis encounters difficulties in on-line extraction of models from data streams.

There are several types of anomaly detection mechanisms. Classification techniques require labeled data (distance-based kNN). These types of methods make the assumption that 'normal' data occurs in dense neighborhoods while anomalies happen far from these. There are of course other methods which can be of some value, such as artificial neural networks [87, 69], SVM [103], Bayesian Networks [45] or Decision Trees [53]. In the case of unsupervised learning methods, outliers are considered part of no cluster, while normal instances of data are part of a cluster. Clustering methods include DBSCAN [17], SNN [3], k-Means, SOM, HMM, WaveCluster, FindOUT or FSM [88]. Anomaly detection can be used to in filtering significant features out from data sets. These features can represent a wide variety of events ranging from unwanted behavior to identifying new data. Thus, it can be used to process data as well as the behavior of complex systems.

The ADMIRE [4] platform is a software stack that implements an architecture where a wide range of data analytics tools are connected together through specific interaction points known as gateways. Although this approach has delivered some success, three years after project completion it has not gained significant traction in the wider area of data analysis. One of the reasons might be the fact that ADMIRE uses technologies such as REST and WSDL. This makes the architecture seem more suited towards cloud/cluster analytics, rather than HPC data analytics at exascale. Weka [99] is a collection of machine learning algorithms for data mining tasks. There are a variety of tools such as data pre-processing, classification and visualization which can be used directly or called from Java code. The fact that this is a Java library limits the suitability to HPC. Moreover, we must mention that distributed versions of the Weka framework have been developed (e.g., Weka4WS [94]), but they are suitable for small/medium size Grids or Cloud platforms, not for extreme computing systems where a massive degree of parallelism must be exploited.

Multivariate exascale data poses a significant problem for data mining and visualization methodologies. This type of data requires many degrees of freedom which can lead to poor data comprehension, and visualization amplifies this problem [26]. The ability to generate data will by far outweigh the ability to store this data. Also, the movement of the data in large quantities through the memory hierarchy to more permanent storage is extremely costly in terms of power [26]. The development of on-the-fly analysis methodologies can provide important advantages, such as data comprehension and improved prediction/classification performance.

Dimensionality reduction and feature selection can happen as early as possible in the data processing workflow. Identifying the most important features in a particular data set can help prevent potentially large execution times of data mining algorithms as well as costly data storage and transfer operations [15]. Dimensionality reduction methodologies that are good candidates for further investigations and potentially further

improvements are Principal Component Analysis [5, 103], Wrapper Methods [90, 40, 47] and others [101, 43, 102]. The idiosyncrasies of each method have to be taken into consideration in order to effectively work on exascale systems.

2.2.2. Fault-tolerant MPI and checkpointing. Pure hardware solutions are not enough to deal with the number of failures of large systems. For this reason, hardware failures must be tolerated by the applications to ensure that not all computation done is lost on machine failures. Checkpointing and rollback recovery is one of the most popular techniques to provide fault tolerance support to MPI parallel applications. It periodically saves the computation state to stable storage, so that the application execution can be resumed by restoring such state.

In case of failure, most of the current checkpointing and rollback solutions restart all the processes from their last checkpoint. However, a complete restart is unnecessary, since most of the nodes will still be alive. Moreover, it has important drawbacks. Firstly, full restart implies a job re-queuing, with the consequent loss of time. Secondly, the assigned set of execution nodes is generally different from the original one. As a result, checkpoint data must be moved across the interconnection network in order to restart the computation. This usually causes significant network contention and therefore high overheads. All these limitations should be overcome through the support for malleable jobs, that is, parallel programs that are able to modify the number of required processors at run-time. This feature allows the automatic adaptation of the parallel execution to the available resources, coping with hardware errors and avoiding the restart of the whole application.

The literature includes several proposals for transforming MPI applications into malleable jobs. However, most of them are very restrictive with respect to the kind of supported applications. Indeed, only iterative or master-slave applications are considered (e.g., [61, 86, 14]), because in these cases the modification of the number of processes is much easier than in a general application. Furthermore, in these approaches reconfiguration can only take place in very specific points within the applications. Thus, new solutions are needed to implement MPI malleable applications in a general and scalable way.

Malleability not only offers advantages from the fault-tolerance point of view [33]. It also provides a higher productivity and a better response time [11, 41]. These characteristics allow to improve the usage of resources, which will have a direct effect on the energy consumption of application execution, resulting in both cost savings and a greener computing.

2.3. Advancing the Balance of Compute, I/O and Storage Performance.

2.3.1. Hardware resilient Exascale storage demo. Traditionally I/O is an activity that is performed before or after the main simulation, analysis computation, or periodically for activities such as checkpointing. Instead, the I/O should be regarded as an integral activity which must be optimized while architecting the underlying software [25]. As HPC systems are becoming larger, the overall system resilience decreases. The resilience of an application in an exascale system depends significantly on the I/O system because saving the state of the application through checkpoint/restart remains an essential component of the global system resilience [23]. The overhead for performing checkpoints and the consumed energy can both be optimized by leveraging new non-volatile memory [29].

We believe that new memory and storage hierarchies can drastically impact performance and resilience. Therefore, we consider worth investigating the integration of non-volatile memory within nodes, interconnects and networks or within I/O storage nodes. Seamless data migration between the multiple tiers of memory and storage is key to achieving the exascale. New technologies, such as flash memory (SSD), phase-change RAM and accelerators, will provide new opportunities to achieve both the above properties (stable local storage, faster checkpointing, faster checkpoint compression etc).

The traditional homogeneous I/O interfaces do not explicitly consider the type of an I/O operation. However, a checkpointing I/O activity is different from an I/O activity. Therefore, we need purpose-driven I/O software layers and cross-layer data management that is specialized for resilience [44], such as checkpointing in memory, hierarchical checkpointing, and checkpointing that leverages the data layout. To achieve resilience and integrity, objects should be replicated in physically separate zones [6]. An example in this direction is VIDAS, an object-based virtualized data sharing for High Performance Storage I/O [56]. Using fast memory (RAM, NVRAM, SSD, etc.) at intermediate storage levels in the hierarchy is becoming more and more popular. Thus,

some cached parallel file systems are now available as powerful I/O tools such as memcachedFS and Panache [27]. Nonetheless, resilience requires a system-wide approach. A cross-layer mechanism with clear failure semantics can support the design of resilient techniques and optimizations at different stack layers. However, it is also important to provide isolation between global/local storage, so that there can be a timely notification and avoidance of failures to ensure data integrity.

2.3.2. Big Data + HPC integrated SW stack. A recent study characterized the storage I/O of computational scientific applications handling large amounts of data, including jobs in Earth science, nuclear physics, energy, and climate [13]. It has been found that the majority of applications prefer to use POSIX interfaces rather than the MPI-IO interface or high-level libraries. Additionally, processes accessed either unique files, shared files, partially shared files or zero files. A surprising amount of data is stored on small files in the range 64KiB-128KiB, and 86% of the files have size below 1MiB. The read and write size was 100KiB to 1MiB across most jobs, although two specific jobs biased the most popular write size between 100B and 1KiB. Checkpointing is recognized as the primary storage driver. Current requirements are specifying that a checkpoint of up to 64PB should complete in 300s.

Another recent study introduced methods to automatically identify I/O patterns at the block level and utilized them to manage the metadata indexing and support data prefetching in physics and geoscience applications [38]. Statistical analysis of NetCDF scientific datasets was successfully explored over the Hadoop platform developed for big data applications [10]. Another study identified several common characteristics between the PVFS distributed file-system used in high-performance computing, and HDFS used in big data analytics [91]. Quality of service in shared storage servers has been previously investigated in the context of transactional and analytical workloads. Promising techniques include automated configuration of prefetch and writeback I/O size along with quanta-based device scheduling [98]. The latest approaches proactively manage quality objectives across multiple resources (e.g., network and storage) but are far from handling exascale data in the complex scientific processing [106].

Based on prior research, we recognize the challenge of managing exascale data in scientific applications and their peculiarities with respect to big data processing for analytical purposes. However, the characterization of I/O requirements of an application can include system services such as monitoring. One way to address current bottlenecks emerging from the handling of numerous files or small I/O requests is through support for storage and network I/O aggregation at the client side. Quality of service in exascale I/O performance is important in this context. The guaranteed performance obtained from shared storage servers can be improved through the exploitation of caching at the client side and efficient storage management at the server. The durability restrictions can be overcome with persistent caching over high-performance storage devices. The main advancement from existing methods can be the native incorporation of the proposed improvements within the distributed filesystem or datastore rather than adding the aggregation support in middleware layers between the client and the storage backend.

2.4. Advancing the Big Data and HPC Usage Models: Problem-solving environments for large data. While there is much debate about the programming models used to program exascale computing systems, the main requirements of the applications targeting this extreme scale are clear: a very large number of threads, limited communication (in space and time) and synchronization, tolerance for execution time variance, scalable data access and low energy consumption [36]. New applications must be written to satisfy these requirements. Scaling up applications to exascale will require programming efforts even if new programming models prove to be adequate for instance handling limited data exchange asynchronously will require new algorithms. Therefore, programmer productivity also matters. Writing efficient programs for a large scale computing platform is a difficult engineering endeavor that requires a lot of knowledge and experience also because complex engineering tasks are not easy to automate [84]. High-level and efficient programming environments will help in achieving this objective.

Exascale-ready problem-solving environments (PSE) should consider computing, data flows, and energy efficiency when determining how to execute pipelined tasks; for exascale computing, measuring and acting upon the computing performance alone is insufficient [30]. Since the constituents of PSE can act as data producers or data consumers, the optimization of data-flow between them is a task that should be handled by the PSE.

For example, global monitoring from space requires new platforms that will ensure the correct exploitation

of data. This unprecedented large volume of data brings new constraints in terms of discovery, access, exploitation, and visualization. How these new volumes of Earth Observation data are integrated in data-intensive applications frameworks define the next generation of Earth Science services. Likewise, even if user and application communities share the original data, we will observe an even more diversification and specialization with new requirements, methods and protocols giving origin to innovative services.

2.5. Overview of the limitations and the paths to overcome them. Table 2.1 summarises the discussions from the previous subsections. They list the main limitations on large data processing and report on possible solution paths.

3. Proposed approach and methodology. In this section we follow the paths proposed in the previous section to overcome the current limitations in extreme data processing by proposing concrete actions.

3.1. Design and develop new HPC programming and energy models for extreme data applications. Exascale computing faces many challenges as parallel codes will need to control millions of threads running on many cores. Such programs will need to avoid synchronization, use less communication and memory, failures could be more frequent, and power management is needed. Whilst these considerations are critically important to the scalability of future codes, the programmers themselves typically want to concentrate on their own application and not have to deal with these lower level, tricky, details. Today no available programming models and languages provide solutions to these issues. Therefore, new programming models are required to handle these challenges.

3.1.1. Programming paradigms and models. High-level programming can help application developers to access and use resources without the need to manage low-level architectural entities, as a parallel programming model defines a set of programming abstractions that simplify the way by which the programmer structures and expresses his or her algorithm.

To effectively support the development of data intensive algorithms and applications on exascale systems, we consider that is urgent to design a simple but scalable programming model (e.g. a SPMD programming model) based on basic operations for data intensive/data-driven applications to deal with the use of a massive amount of processing elements and develop an API based on that model which includes operations for data access, data communication and data processing on groups of cores. The model must be able to manage a very large amount of parallelism and to implement reduced communication and synchronization.

At the extreme scale, the cost of accessing, moving, and processing data across a parallel system is enormous. This requires mechanisms, techniques and operations for efficient data access, placement and querying. In addition, scalable operations must be designed in such a way to avoid global synchronizations, centralized control and global communications. Many data scientists want to be abstracted away from these tricky, lower level, aspects of HPC until at least they have their code working and then potentially to tweak communication and distribution choices in a high level manner in order to further tune their code.

Interoperability and integration with the MapReduce model and MPI must be investigated with the main goal of achieving scalability on large-scale data processing.

3.1.2. Productive exascale heterogeneity exploitation. Exascale data-extreme applications require large clusters of distributed memory systems, including often heterogeneous devices of very different natures. The programming of these systems is mostly based on relatively low level standards, such as CUDA and OpenCL for the exploitation of heterogeneity, or MPI for managing communication among computing nodes, which hugely reduces the productivity of programmers.

Different data-centric abstractions can be integrated in order to provide a unified programming model and API that allows the productive programming of both heterogeneous and distributed memory systems. A software implementation can take the form of a library. Its application to reference benchmarks (like Graph500 - the data-intensive benchmark - or hpcchallenge.org) allows to gather feedback that will lead to improvements in the prototype.

In order to simplify the development of applications in heterogeneous distributed memory environments, large scale data-parallelism can be exploited on top of the abstraction of n-dimensional arrays subdivided in tiles, so that different tiles are placed on different computing nodes that process in parallel the tiles they own.

Table 2.1: Limitations on large data processing and paths to overcome them

Category	Limitations	Paths to overcome the limitations
Non-conventional parallel programming	Lack of abstract representation of data-intensive tasks; exascale computing cannot be reach without considering the heterogeneity of the systems; MPI limitations in what concern data types; PGAS do not support hierarchical decomposition; MPI and PGAS implementations focus on communications rather on management of large volumes of data.	Build abstractions on top of an already familiar programming language; new libraries to support abstract data types for explicitly distributed objects; high level annotations build on top of APIs for accelerators and evolve the low-level programming of accelerators into high productivity approaches through integration with the abstract data types.
APIs for auto-tuning performance or energy	Current monitoring tools are measuring raw performance; current tools for evaluation of power consumption use system level information, leading to an intensive consumption; instantaneous power consumption is considered.	Implement adaptive prediction tools for decision systems for auto-tuning the performance and energy.
Inspection of data locality at Exascale level	Data locality is managed independently at application, middleware and file system levels; scalable file systems are not widely adopted by HEC; MPI-IO data operations are not locality-aware.	Design high-level I/O libraries to map hierarchical data models to storage data models with scalability and performance in mind.
New metrics, analysis techniques and models	Current performance and analysis tools operate at process or thread level (not scalable); existing tools focus on offline analysis; approaches for representing correlation between conflicting optimization objectives are not targeting data-intensive applications and do not apply energy modeling or prediction techniques.	Build new tools to operate at the level of the distribution and layout of data; target the on-the-fly classification and frequent pattern algorithms, non-functional parameters estimation techniques; use of GPGPU for many multi-objective optimization problems.
On-the-fly data analysis, data mining	Current data analysis tools are designed for distributed systems and not for HPC systems; preprocessing (in particular outliers detection) and stream data analysis encounters difficulties in on-line extraction of models from data streams; multivariate exascale data have too many degrees of freedom which can lead to poor data comprehension.	Outliers detection before modeling and data analysis by using of anomaly detection mechanisms; development of on-the-fly analysis methodologies to improve data comprehension; dimensionality reduction and feature selection to reduce execution and transfer times.
Fault-tolerant MPI and checkpointing	Full restart in case of faults creates job requeuing and huge checkpoint data transfers.	Implement malleable job mechanisms to be able to modify the number of required processors at run-time.
Hardware resilient exascale storage demo	The resilience of an application in an exascale system depends on the I/O system; the current I/O interfaces do not consider the type of an I/O operation (e.g. checkpointing vs. other I/O acivities)	Overhead for performing checkpoints can be optimized by leveraging non-volatile memory techniques; design I/O software layers and cross-layer data management specialized for resilience, such as checkpointing in memory, hierarchical checkpointing.
Big Data + HPC integrated software stack	Bootleneck in handling numerous files or small I/O requests.	Storage and network I/O aggregation at client side; caching at client side, persistent caching over high-performance storage devices.
Problem-solving environments for large data	PSE are usually not considering data flows or energy efficiency when determining how to execute pipelined tasks.	Optimize the data-flow between the data producers or data consumers inside the PSE.

Such an approach allows one to easily process the tiles at each node in either regular CPUs or in any of the heterogeneous systems available (GPUs, Xeon Phi coprocessors, etc.) using a unified API and runtime that hides the complexity of the underlying process. This abstraction should be supported by abstract data types provided by libraries, so that they can be easily integrated in existing applications.

Another issue is the gap between users with HPC needs and experts with the skills to make the most of these technologies. An appropriate directive-based approach can be to design, implement and evaluate a compiler framework that allows generic translations from high-level languages to exascale heterogeneous platforms. A programming model should be designed at a level that is higher than that of standards, such as OpenCL. The model should enable the rapid development with reduced effort for different heterogeneous platforms. These heterogeneous platforms need to include low energy architectures and mobile devices. The new model should allow a preliminary evaluation of results on the target architectures.

3.1.3. Energy awareness. On exascale systems, it becomes impossible to assume that each component of applications will behave exactly the same. To manipulate tasks and components efficiently, different types of information are required. Most of the time, only resource-related information is manipulated.

In exascale systems, energy-related information is of utmost importance. Such information cannot be directly measured at the application level, therefore it must be obtained by using higher-level models of applications. The resource consumption of applications must be tracked during their execution in order to evaluate bottlenecks – when manipulating data-intensive applications these bottlenecks become even more important. Thus resource and energy consumption models need to be temporal. Either during design phase or by runtime-detection, we can model and then predict the resource and energy consumption of each particular component. Reference benchmarks (like the ones from Green Graph 500) allows one to gather feedback leading to improvements in the models.

Using this high level information, a runtime will be able to take informed decisions and thus will reduce energy consumption and prevent bottlenecks.

3.1.4. Runtime support. The programming models previously described need to be supported by a runtime system. The runtime system is responsible for managing, coordinating and scheduling the execution of an application by deciding when, where and how its constituent components should be executed.

The execution can be based on data-driven, dynamic DAG scheduling mechanisms, where a component in the DAG will be ready for execution once all of its required input parameters have become available. In this case the runtime system is expected to control and optimize the execution of the applications in an optimized internal representation as a dynamic DAG: the nodes represent computational components and the edges track the data dependencies between components. Once a component is ready to execute, the runtime will choose the potentially best implementation variant, depending on the available hardware resources, the estimated performance, energy and efficiency aspects, the dynamic feedback of the program and system behavior, and the user-specified optimization goals.

Such a runtime can be a rule-based system with event-condition actions. The event specifies the rule that is triggered after which the condition part is queried to see the best suites on which the application should run. When particular conditions are met, the rule fires and the appropriate actions are executed. There are several advantages to this method compared to encoding such rules into an application code. Firstly the rules are organized into a knowledge base which makes them more modular and expendable. It also improves maintenance and a knowledge base is well suited for analysis and optimization of the rules it contains. Lastly an embedded rule-base is much more specific to a certain type of reactive behavior and in our scenario the rule-base is a much more generic mechanism.

In addition, extreme data applications often involve dynamic workflows that demand a changing amount of resources. Moreover, exascale systems aggregate a very large number of computational resources with a high rate of variation in their availability and load state. Thus, applications should be able to dynamically adapt to these changing conditions. Nowadays, many parallel applications are written using the traditional MPI programming paradigm. However, MPI does not provide explicit support for malleability. Thus, MPI should be extended with libraries and run-time support to implement malleable applications. The solution can be based on checkpointing and be implemented at the application-level to improve portability and efficiency. Two different solutions can be explored. First, a virtual malleable approach, where the number of processes

is preserved and the application adapts to changes in the amount of resources by oversubscribing processors. As the number of processes is preserved, this solution has not restrictions on the type of applications allowed. Second, a scalable data redistribution approach to achieve an efficient reconfiguration of the number of processes. An incremental procedure can be followed in this case having as starting point regular data distributions and iterative applications, and extending the application's field in every new step.

3.2. Build new tools for monitoring and data-analysis for extreme data applications.

3.2.1. Extreme data mining techniques. A novel set of mechanisms and resources needs to be integrated into high-level operations and tools. These should be provided to programmers or data mining researchers for developing, configuring, and running data mining applications targeting at reaching the exascale. The novel set of operations and resources needed to address the areas of accessing, exchanging, transforming, and mining big data sets. The high-level operations should allow a user to compose the novel exascale programming mechanisms in order to access, filter, pre-process, and mine data in parallel using data analysis graphs that correspond to scalable data mining applications. These scalable tools based on novel programming paradigms should be used to design scalable codes that support the implementation of large-scale data mining applications.

Dimensionality reduction of exascale data is an important consideration which can help make near real-time analysis possible by reducing the size of the data necessary for analysis tasks thus reducing computational and ultimately energy requirements.

Another important consideration is that high data throughput is prone to errors. These errors are detrimental to any data mining application or algorithm, thus a method to detect anomalies or outliers in the incoming data needs to be investigated.

3.2.2. Extreme data collection and monitoring. Instrumentation, monitoring and measurement techniques should be tuned for data-intensive applications in order to provide fine-granular performance and energy information with low-overhead at the exascale level. Scalable, low overhead measurement and analysis techniques should enable trade-offs between measurement perturbation, measurement data accuracy, analysis response time and overall benefit. An analysis of such techniques should be based on lightweight messages between measurement/analysis threads and the run-time system that selectively applies tracing, profiling or sampling to measure individual non-functional parameters (NFP) for different parts of the application or computing system to control measurement accuracy and program perturbation. To achieve this, an investigation can be done on upgrading options based on techniques such as the Erlang component isolation [1] or subgraph folding algorithms [80]. For collecting measurements, statically and dynamically instrumentation is needed for the dynamic code representation in the run-time system.

To deal with missing data (e.g., energy measurements not available on a code region basis) or for compressing it, analysis techniques based on Hoeffding Tree, adaptive Bayes, classification, and frequent pattern algorithms (i.e., to find memory usage patterns that require less storage compared to simple, time-sampled data) should be investigated. The estimation of NFPs can be exploited for specific architecture parts and application components or regions with incomplete measurements based on sophisticated event processing and analysis patterns (e.g. analyse correlation between execution time and energy consumption) applied to different parallel patterns.

Scalable event processing analysis techniques should allow performance experts to spend their time on developing reusable patterns for NFPs and detecting performance problems, rather than on specific performance analysis features that are hardly reusable for new parallel applications and architectures. The necessary storage infrastructure need to be investigated to address the concurrency needs of monitoring and event processing at large scale. According to the requirements of the implemented operators, memory and disk resources need to support real-time processing and historical data access at low cost.

Existing energy monitoring software is based on specific acquisition devices or performance counters and no standard exists on how to energy profile a HPC application. Standard metrics and profile grain has to be defined, which depends on the capabilities of each power metering device. The different types of energy metering devices available should be identified and a common interface for energy metering needs of exascale HPC should be proposed. The automatic discovery of the necessary libraries and available devices at runtime should be a mandatory feature in this context.

3.2.3. Scalable data analysis. Existing performance analysis tools are instruction-centric and offer a code region-based profile of the application. Thus they may hide important sources of bottlenecks related, for example, to an inefficient data distribution or to some misaligned memory addresses causing false sharing. This problem is caused by the focus on processing rather than locality which is a central reason for performance problems in contemporary parallel architectures. A data-centric analysis tool should be therefore designed (in addition to the classical region-based view) exposing the complete global application address space and associating the collected metrics with program variables rather than instructions and code regions. The key insight behind such an approach is that a source of a bottleneck in a HPC application is often not the affected code fragment where it is detected (i.e. where the data is processed with a high communication or thrashing overhead), but where it is allocated. The tool can drive and filter the required instrumentation and data collection described before required for the analysis. The analysis and visualization requirements will be supported by the necessary storage facility with schema support for the operators of data insertion and retrieval involved.

3.2.4. Automatized correlation analysis. Understanding possible correlations between conflicting parameters such as execution time, energy consumption, and efficiency at the exascale level may significantly reduce the search space for extreme data mining or runtime optimization purposes at the exascale level. Therefore, search strategies should be investigated to approximate the set of Pareto-optimal solutions. For this kind of search, techniques such as ranked Pareto dominance, crowding, and other techniques with a proven track record in challenging multicriterial search problems like NSGA-II and PAES can be used. These techniques can be enhanced with theoretical results, for example from elementary landscape theory to reduce the search time.

In order to guide dynamic optimization and reduce the search overhead, performance and energy models associated with components that are automatically generated by the runtime system or by the user are needed. To enable dynamic optimization with minimal overhead, the optimization logic can become an HPC problem, distributed among the otherwise inefficiently used resources in a highly malleable fashion.

3.3. Adapt the data management and storage techniques to the extreme scale.

3.3.1. Predictive and adaptive data layout strategies. In the software storage I/O stack of current petaflop machines, data locality is managed independently at various levels such as application, middleware, or file systems, thus, data locality is not exposed or propagated. Understanding the relationship between the different layers of the I/O stack is critical to providing locality-awareness in every layer, which is a key factor for building exascale systems. Therefore strategies and mechanisms required to provide predictive and adaptive data layout strategies need to be investigated. It is needed to advance the best-effort strategies that co-locate computation and data towards predictive techniques implemented in cooperation with the system scheduler. However, in the current state-of-the-art software stack data locality cannot be dynamically controlled on the data path from application to storage. This may cause inefficient accesses due to various factors such as data redundancies, lost optimization opportunities, and unnecessary data reorganization. One way to overcome these limitations is to adopt adaptive data-layout strategies that dynamically adjust their behavior to fit the application and storage data layout.

One of the major problems in exascale I/O is the lack of data distribution strategies that can increase data locality by taking into account application behavior. Current I/O systems are mostly statically configured and cannot provide dynamic features to applications and a methodology for profiling and analyzing data-intensive applications for identifying opportunities for exploiting data locality is needed. Such a methodology should allow the study of the dynamics of data movement and layout throughout the whole data storage I/O path from the back-end storage up to the application in order to understand the requirements for building mechanisms for exposing and exploiting data locality. Then, techniques for providing dynamic configurations of the I/O system can be designed and developed to enhance the whole data life-cycle by reflecting applications I/O patterns, restructuring the I/O stack functionalities, and providing predictive and adaptive data allocation strategies.

3.3.2. Resilience through cross-layer data management. A cross-layer data management and strategies for resilience should allow checkpointing in memory, hierarchical checkpointing, and should leverage the data layout checkpointing. To achieve resilience and integrity objects can be replicated in physically separate memory zones using an object-based virtualized data sharing for high performance storage I/O. Using fast memory (RAM, NVRAM, SSD, etc.) at the intermediate storage levels of the hierarchy is instrumental to

providing resilience through replication and checkpointing, while maintaining scalability and energy efficiency. As resilience requires a system-wide approach, a cross-layer mechanism with failure semantics to support the design of resilient techniques and optimizations at different stack layers is needed. This mechanism should provide isolation between global and local storage so that there can be a timely notification and avoidance of failures to ensure data integrity.

Supporting the development of I/O software through cross-layer mechanisms will contribute to: enhancing energy efficiency by reducing the communication through a better exploitation of data locality and layout; improving resilience by providing isolation between global/local storage so that there can be a timely notification and avoidance of failures to ensure data integrity; exploiting data locality and concurrency to hide the latencies, avoid congestion, and identify and bypass failures in an early stage. This development of the I/O software also allows the users to express some key aspect of what they want to do with their data, and lets the users know what happened with the management of their data. This bi-directional path of meta-information will provide better information for optimization and the goal is to express this meta-information in a clear way for developers.

3.3.3. Storage structures for exascale I/O. For several years, I/O-intensive HPC has been primarily based on distributed object-based filesystems that separate data from metadata management and allow each client to communicate in parallel directly with multiple storage servers. Exascale I/O raises the throughput and storage capacity requirements by several orders of magnitude, therefore methods that can manage the network and storage resources accordingly are needed. The systems already developed for Big Data analytics are not directly applicable to HPC due to the fine-granularity I/O involved in scientific applications. Another weakness of existing systems is the semantic gap between the application requests and the way they are managed at the block level by the storage backend.

The storage bottlenecks of the well known data-intensive applications should be analyzed when scaling the size of the processed dataset and the generated output, or the frequency of the obtained checkpoints. Approaches to aggregate the I/O requirements at the client need to be identified, as well as methodologies for providing the necessary storage support at the backend. The supported concurrency should be increased through a combination of a simple I/O interface, algorithms for both data and metadata indexing, and methods of heterogeneous device management.

Exascale data processing stresses the storage infrastructure in terms of resource utilization, reliability and performance. Therefore appropriate storage structures that rely on object-based filesystems and scalable datastores are needed. Issues that have to be examined include the degree of replication according to fault-tolerance guarantees, caching at different system components for performance at low cost, and concurrency level for parallel data processing. The proposed solutions can take advantage of the latest advances in storage and hardware technology to provide through extreme parallelism improved data throughput, low latency, and durable storage in HPC application execution – experimental software based on production-grade open-source file/storage/data management systems can be built.

An important aspect in those new exascale I/O management techniques will be avoiding metadata management, because metadata generates almost 70% of the I/O operations in most systems – small control I/O operations – usually on shared data structures, which limits global scalability. The requirements of an efficient and fault tolerant metadata service for exascale I/O need to be analyzed, and generic scalable metadata service to use in the storage structure of exascale I/O systems need to be designed and implemented.

3.4. Validate the concepts and tools usefulness through extreme data applications. Extreme data is encountered as input, output or intermediate data in multiple data-intensive applications, in fields such as environmental sciences or dynamic system simulations. Applications in these areas are able to measure the advances in processing data on the three dimensions introduced in Section 1, Volume, Value and Variety. We discuss in what follows some candidates for such validations.

3.4.1. Earth Observation. Earth Observation (EO) from satellites produces vast amounts of data and is playing an increasingly important role as a regular and reliable high-quality data source for scientific, public sector and commercial activities. The Landsat archive alone corresponds to more than five million images of the Earth's land surface (roughly 1 petabyte of data). From 2014 onwards, new satellites from Europe, USA,

China, Brazil, and India will each produce in a year as much new data as one Landsat satellite has acquired in ten years. This unprecedented amount of data made available for research and operational use will open new challenges.

For more than two decades, the European space industry has developed remote sensing sensors and launched them on robust platforms to assure regular and near-real-time provision of key parameters for the monitoring of land, ocean, ice and atmosphere. The recent data hiatus from ENVISAT's abrupt end of mission will be overcome with the new Sentinel and Earth Explorer programs operated by ESA and the European Global Monitoring for Environment and Security/Copernicus Program Contributing Missions. The above and third-party missions will be combined with data from the long-term EO archive (e.g. ERS, Envisat), in-situ networks and models. Thus, they will provide routine monitoring capabilities of the environment at a global scale and unprecedented insight into how oceans, atmosphere, land and ice operate and interact as part of an interconnected Earth System.

The expanding operational capability of global monitoring from space opens a unique opportunity to build sustainable platforms that support services exploiting archived or new rich EO data sets. In order to be successful, such platforms must exploit the latest advances in scientific data management. Thus they will offer easy and seamless access to all relevant data repositories, as well as efficient operations (search, retrieval, processing/re-processing, projection, visualization and analysis) to extract and distribute single parameters or combined products on user demand.

The broad range of existing constraints for using satellite Earth observations is related to the ability of EO missions to (i) access and monitor the target areas at the right time, and (ii) extract the needed information from the collected sensor data. When not considering timeliness constraints, the sheer volume of data needed for change analysis presents a huge challenge for the current techniques with respect to accessing and processing very large collections of satellite data. Existing classification algorithms using HPC and extracting features from data gathered by satellites or airplanes scanning the Earth cannot be applied in real time and therefore the data collected from satellites is processed only at request (event based). Recently the scalability of classification algorithms has been improved, while the problem of feeding the HPC systems with the remote sensing data and exploiting the data locality remains open.

With the establishment and maintenance of long-term EO programs (Sentinels, Earth Explorer, Landsat or past ENVISAT mission) it is finally possible to obtain a long and homogeneous time series of satellite imagery. Time-series analysis involves capturing the change of one or more variables over time and providing a rich source of information on the dynamic nature of Earth surface processes for monitoring land-cover change and vegetation-climate dynamics. Using a large set of satellite images that cover the same spatial domain but originate from different satellites at different acquisition times, makes it possible to obtain an extended data series with shorter time intervals. This capability will offer new opportunities for monitoring landscape changes and advance our understanding of the reasons behind those changes. Different sensors will support different spatial resolution and registration constraints. When the generated data is integrated in a common model will result in a valuable information source revealing complex patterns for environmental monitoring and analysis of landscape dynamics.

Analyzing big temporal datasets of satellite imagery and derived products qualifies as processing of extreme data. An example is the mining of temporal patterns on the geospatial domain through a composition of services that discover, access and analyze geospatial data cubes in an exascale environment (storages, metadata services, data, analysis tools, etc.) that connects to existing data infrastructures. In particular such EO extreme-scale application can be applied to measure land-use change over time.

3.4.2. Weather forecast. The current weather forecasting systems are reliant on numerical simulations. The achieved accuracy of prediction is sensitive to the accuracy of initial conditions and estimates of the current state of the atmosphere —the accuracy depends on an optimal combination of numerical simulations and observational data. Severe weather events can be very localized (within a few km) and over a short lifetime (a few minutes.) To forecast such phenomena the sensors must be able to send data at a rate of half a minute intervals and simulations should run at a resolution of few hundred meters. However, 2Pflops are estimated to be needed to run 100 simulations for 30s at a 100m resolution, producing 200GB of data at each 30s which must be merged with observational data, producing another 200GB of data in the next 30s [63].

The WRF (Weather Research and Forecasting) Model is a recent mesoscale numerical weather prediction system designed to serve both atmospheric research and operational forecasting needs [66]. It features two dynamical cores, a data assimilation system, and a software architecture facilitating parallel computation and system extensibility. WRF is in operational use at US National Centers for Environmental Prediction, Air Force Weather Agency and other centers. Currently the WRF implementations suffer from lack of accuracy and real-time response even though they run on a cluster level. To improve the quality of output and build effective warning system, data collected in real time from various sources should feed into the WRF simulator and the resolution should be increased (resulting in both computation and data-intensive challenges.) The forecasting application needs to be reprogrammed for efficiently handling the increased data requirements based on project developments in exascale programming, monitoring and storage.

Extreme data processing arises in advanced weather forecasting that is based on the effects of heat fluxes and wave height in the sea surface under consideration of hydrological models, and the sensitivity to various convective parameterization schemes of extreme precipitation events over a complex terrain. The future plans of weather forecasting and a related early-warning systems can include an extension to existing processes in space and time to cover parts of a continent at an increased vertical resolution and multiple runs per day [8, 16, 54]. Such a redesign of the forecast process can increase the volume of the output results by approximately 700 times, which will approach 1 Terabyte per day.

3.4.3. Urban computing. Urban computing refers to the process of acquisition, integration, and analysis of big and heterogeneous data. The data is generated by a diversity of sources in urban spaces, such as sensors, devices, vehicles, buildings, and human. The goal is to tackle the major issues that cities face, such as air pollution, increased energy consumption and traffic congestion [105]. It involves sensing technologies, advanced data management and analytics models, as well as visualization methods in order to generate solutions that improve urban environment, human life quality, city operation systems. HPC capabilities were recently introduced to develop solutions for sustainable urban operations [92, 46]. The discovery of mobility models is one of the most challenging issues which requires HPC [70], but it can improve resource furnishing and management of cities.

The category of extreme-scale applications includes the smart-cities planning, designed as a composition of different services allowing to gather and collect environmental data, and subsequently process and analyse it in order to mine social and urban behaviors. The involved components in an exascale environment (e.g. storages, metadata services and analysis tools) should interoperate and cooperate seamlessly. The access to underlying infrastructure should be secure, pervasive and ubiquitous. A trajectory pattern-extraction methodology will be applied to a real-world dataset concerning mobility of citizens within an urban area. The goal is to discover user behavior and provide useful information about mobility-related phenomena so as predict future movements of citizens, in order to support decisions in various ways.

We plan to analyze the mobility of citizens in order to discover people and community behaviour, i.e., patterns, rules and regularities in moving trajectories. A novel methodology is needed to extract and analyze the time- and geo-references associated with social data so as to mine interesting locations and frequent travel sequences among these locations in a given geo-spatial region. It will allow to infer human dynamics and behavior within urban context. The extreme amount of geo-tagged data generated also by social-media users together with the data-intensive computations needed for their analysis requires an extreme-scale distributed implementation in which memory size and I/O capabilities scale with compute power.

3.4.4. Numerical simulations. Large-scale numerical simulations of complex domains, modeled by partial differential equations (PDEs), comprise processing of data from billions of discretization nodes. For each node, several hundred bytes of data are stored for several thousand time steps of the simulation, creating simulation results in order of Petabytes. This data is then post-processed, analyzed and visualized, creating a new bottleneck in both data-movement and processing phases. Consequently, the scalability preservation in exascale systems requires new algorithms and data structures supported by approaches of efficient communication, data manipulation, storage principles and tailored high-performance processing.

Many of the established numerical algorithms and approaches have been designed at a time when computer platforms were based on a moderate number of very fast computing nodes. In the exascale era, new computing architectures are foreseen, built from millions of cores and equipped with accelerators able to implement data flow

approaches. The existing numerical methodologies and algorithms must follow these trends. For example, an implicit methodology is often preferred for the solution of PDEs because of longer possible time steps, however, because of the additional price for the solution of a large global linear system. An alternative approach, based on explicit methods, requires only a matrix-vector multiplication in each time step. Even though the number of time steps in explicit methods could be higher, such an approach could better fit the new architecture with many smaller processing units resulting in a more efficient methodology.

One of the most complex fields of scientific computing is the computational fluid dynamics (CFD), which is of great interest among researchers and engineers in many areas of science and technology. The core problem is the solution of the Navier-Stokes PDE equation or its variants, e.g., Darcy or Brinkman equation for flow in porous media. The CFD plays a crucial role in the modeling of many important industrial processes, e.g., heat transport-energy studies, solidification of advanced materials, microfluidics and nanofluidics, vehicles design, etc. The solution approaches that are based on local information from neighboring discretization nodes will be preferred in exascale CFD solvers, because such methodologies are closer to natural behavior. Moreover, such approaches are tailored to the execution on exascale computer architectures that incorporate multi-level processor and memory hierarchies, processing accelerators, and high-radix interconnection networks.

Solving extensive problems via local numerical methods requires efficient subdivision of the domain into several million to several billion discretization nodes. Each simulation works on Terabytes of data, but even more data is stored for each simulation in form of checkpointing and simulation results. Each node produces its own simulation data that can create substantial overhead if it has to be communicated to a centralized storage. Distributed storage in which nodes store data locally, or at least at a nearby storage location is preferable to centralized storage at a single location that becomes an interconnection and storage bottleneck. Furthermore, distributed storage of results promotes distributed analysis and data visualization, which also proves highly problematic when working with Terabytes and Petabytes of data. Therefore, the data storage of the simulation procedure should be planned with consideration of the optimal storage locations required.

3.5. Overview of the proposed methodology. Tables 3.1 and 3.2 summarize the proposed actions which were discussed in details in the previous subsections. In a concise way, they show the key actions to be carried out and the expected validations coming from some applications domains.

4. Conclusions. We provided in this paper a roadmap to reach the extreme-scale level in data processing. It is based on four pillars: new programming and energy models for extreme data applications, new tools for monitoring and data-analysis for extreme data applications, adaptation of data management and storage techniques to extreme scale, and validation of the concepts and tools through specific extreme data applications. For each topic some limitations of the existing concepts and technologies have been identified and potential solutions to overcome these limitations were sketched without claiming that they guarantee the success or that they are the only solution. We recognize that the efforts to implement these solutions can be enormous and will require the joint efforts of multiple teams. Therefore we considered it useful to expose our ideas and proposals in this paper and to seek for contributions in the proposed directions in the years to come.

REFERENCES

[1] J. Armstrong, *Erlang*, Communications of ACM, 53(9), September 2010.
[2] A. Acosta, F. Almeida, *Towards a Unified Heterogeneous Development Model in AndroidTM*, Euro-Par 2013, 238-248
[3] I. Ahmad, A. Abdulah, A. Alghamdi, *Towards the Designing of a Robust Intrusion Detection System through an Optimized Advancement of Neural Networks*, Advances in Comp. Science and IT, Springer 2010.
[4] M. Galea, M. Atkinson, C. S. Liew, P. Martin, *Final Report on the ADMIRE Architecture*. 2011
[5] E. Alpaydin, *Introduction to machine learning*. The MIT Press, 2004.
[6] J. Arnold, *Software Defined Storage with OpenStack Swift*, Amazon, 2013.
[7] A. Barak, T. Ben-Nun, E. Levy A. Shiloh, *A package for OpenCL based heterogeneous computing on clusters with many GPU devices*, 2010 IEEE Intl. Conf. on Cluster Computing Workshops and Posters, 17, 2010
[8] A. Bartzokas, V. Kotroni, K. Lagouvardos, C.J. Lolis, A. Gkikas, M.I. Tsirogianni, *Weather forecast in north-western Greece: RISKMED warnings and verification of MM5 model*, Natural Hazards and Earth System Sciences, 10, 383-394, 2010.
[9] A. Bourdon, A. Noureddine, R. Rouvoy, L. Seinturier, *Powerapi: A software library to monitor the energy consumed at the process level*, ERCIM News 2013, no. 92.

Table 3.1: Actions of the proposed methodology

Category	Actions
Programming paradigms and models	Design a simple and scalable programming model based on basic operations for data intensive/data-driven applications to deal with the use of a massive amount of processing elements; develop an API based on that model which includes operations for data access, data communication and data processing on groups of cores; interoperability and integration with the MapReduce and MPI models.
Productive exascale heterogeneity exploitation	Different data-centric abstractions integrated to provide a unified programming model and API that allows the productive programming of both heterogeneous and distributed memory systems; large scale data-parallelism can be exploited on top of the abstraction of n-dimensional arrays subdivided in tiles; directive-based approach and a compiler framework that allows generic translations from high-level languages to exascale heterogeneous platforms.
Energy awareness	Model and predict the resource and energy consumption of data-intensive tasks.
Runtime support	Execution based on data-driven via dynamic DAG scheduling mechanisms; rule-based system with event-condition actions; MPI extended with libraries and run-time support to implement malleable applications.
Extreme data mining techniques	Novel set of operations needed to address the areas of accessing, exchanging, transforming, and mining big data sets; access, filter, pre-process, and mine data in parallel using data analysis graphs; dimensionality reduction techniques for exascale data; methods to detect anomalies or outliers in extreme data.
Extreme data collection and monitoring	Instrumentation, monitoring and measurement techniques tuned for data-intensive applications; support real-time processing and historical data access at low cost; common interface for energy metering needs of exascale data processing.
Scalable data analysis	Data-centric analysis tool; analysis and visualization requirements supported by the storage facility with schema support for the operators of data insertion and retrieval involved.
Automatized correlation analysis	Understanding possible correlations between conflicting parameters using parallel search strategies.
Predictive and adaptive data layout strategies	Advance the best-effort strategies that co-locate computation and data towards predictive techniques implemented in cooperation with the system scheduler; methodology for profiling and analyzing data-intensive applications for identifying opportunities for exploiting data locality.
Resilience through cross-layer management	Mechanisms for checkpointing in memory, hierarchical checkpointing, and data layout checkpointing; provide isolation between global and local storage.
Storage structures for exascale I/O	Approaches to aggregate the I/O requirements at the client; storage structures that rely on object-based filesystems and scalable datastores; efficient and fault tolerant metadata service for exascale I/O.

[10] J. B. Buck, N. Watkins, J. LeFevre, K. Ioannidou, C. Maltzahn, N. Polyzotis, S. Brandt, *SciHadoop: Array-based Query Processing in Hadoop*, ACM Intl Conf for High Performance Computing, Networking, Storage and Analysis (SC), 2011.

[11] J. Buisson, O. Sonmez, H. Mohamed, W. Lammers, D. Epema, *Scheduling malleable applications in multicluster systems*, IEEE International Conference on Cluster Computing, 372381, 2007.

[12] L. C. Canon and E. Emmanuel, *MO-Greedy: an Extended Beam-Search Approach for Solving a Multi-Criteria Scheduling Problem on Heterogeneous Machines*, International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, Shangai, 2011.

[13] P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, R. Ross, *Understanding and Improving Computational Science Storage Access through Continuous Characterization*, ACM Transactions on Storage (TOS), vol 7, no 3, 2011.

[14] M. Cera, Y. Georgiou, O. Richard, N. Maillard, P. Navaux, *Supporting Malleability in Parallel Architectures with Dynamic CPUSETs Mapping and Dynamic MPI*, Distributed Computing and Networking, 242257. Springer, 2010.

[15] G. Chandrashekar, F. Sahin, *A survey on feature selection methods*, Computers & Electrical Engineering, Volume 40, Issue 1, 16-28, 2014.

[16] F. Chen, J. Dudhia, *Coupling an advanced land surface-hydrology model with the Penn State-NCAR MM5 modeling system. Part I: Model implementation and sensitivity*, Mon. Wea. Rev., 129, 569-585, 2001

Table 3.2: Expected validations of the proposed actions by the selected applications

Category	Earth observation	Weather forecast	Urban computing	Numerical simulations
Programming paradigms and models	✓	✓	✓	✓
Productive exascale heterogeneity exploitation	✓			✓
Energy awareness		✓		
Runtime support	✓		✓	✓
Extreme data mining techniques	✓		✓	
Extreme data collection and monitoring		✓		
Scalable data analysis	✓	✓	✓	
Automatized correlation analysis		✓		
Predictive and adaptive data layout strategies	✓			✓
Resilience through cross-layer management		✓		✓
Storage structures for exascale I/O	✓	✓		

[17] Z. Chen, Y.F. Li, *Anomaly Detection Based on Enhanced DBScan Algorithm*, Procedia Engineering, 2011.

[18] S. Conway, C. Dekate, E. Joseph, *IDC HPC End-User Special Study of High-Performance Data Analysis (HPDA): Where Big Data Meets HPC*, Worldwide Study of HPC End-User Sites, 2013, www.idc.com

[19] L. F. Cupertino, G. Da Costa, J.-M. Pierson, *Towards a generic power estimator*, Computer Science - Research and Development, Springer Berlin / Heidelberg, Special Issue Ena-HPC 2014

[20] P. Martin, G. Yaikhom, *DISPEL: Users' Manual*, 2011, http://www.admire-project.eu/docs/DISPEL-manual.pdf

[21] R. F. Vicente, I. Klampanos, A. Krause, M. David, A. Moreno, M. Atkinson, *dispel4py: A Python Framework for Data-Intensive Scientific Computing*,2014 Data Intensive Scalable Computing Systems (DISCS-2014) workshop (SC14).2014

[22] J. Dean and S. Ghemawat, *Mapreduce: simplified data processing on large clusters*, Commun. ACM, 51(1):107113, 2008.

[23] Department of Energy, *Cross-cutting Technologies for Computing at the Exascale*, Washington, DC, Scientific Grand Challenges Workshop Series, pp. 99, 2009. http://extremecomputing.labworks.org/crosscut/CrosscutWSFinalRept Draft02.pdf.

[24] Department of Energy, *DOE Exascale Roadmap Highlights Big Data*, 2014, http://www.hpcwire.com/2014/04/07/doe-exascale-roadmap-highlights-big-data/

[25] J. Dongarra, P. Beckman, T. Moore, P. Aerts et al, *The International Exascale Software Project roadmap*, International J. of High Performance Computing Applications, 2011.

[26] D. Engel, L. Huttenberger, B. Hamann, *A Survey of Dimension Reduction Methods for High-Dimensional Data Analysis and Visualization*, Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering - Proceedings of IRTG Workshop, 2011

[27] M. Eshel, R. Haskin, D. Hildebrand, M. Naik, F. Schmuck, R. Tewari, *Panache: A Parallel File System Cache for Global File Access*, FAST 2010. Usenix,. 2010.

[28] European Technology Platform for High Performance Computing, *ETP4HPC Strategic Research Agenda Achieving HPC leadership in Europe*, Barcelona May 2013, www.etp4hpc.eu.

[29] European Technological Platform for High Performance Computing, *Vision Paper*, 2012

[30] M.J. Flynn, O. Mencer, V. Milutinović, G. Rakočević, P. Stenstrom, R. Trobec, M. Valero, *Moving from petaflops to petadata*, Communications of the ACM. 2013;56:39-42;

[31] M.M. Gaber, A. Zaslavsky, S. Krishnaswamy, *Mining data streams: a review*, ACM Sigmod Record, volume 34, number 2, 18-26, ACM, 2005.

[32] T. Gamblin, B.R. De Supinski, M. Schulz, R. Fowler, D.A. Reed, *Clustering performance data efficiently at massive scales*, 24th ACM International Conference on Supercomputing, 243-252, ACM, 2010.

[33] C. George, S. S. Vadhiyar, *ADFT: An adaptive framework for fault tolerance on large scale systems using application malleability*, Procedia Computer Science 9 (0) (2012) 166 175, Proceedings of the International Conference on Computational Science, ICCSg 2012. doi: 10.1016/j.procs.2012.04.018.

[34] S. Ghemawat, H. Gobioff, and S.-T. Leung, *The google file system*, Proceedings of the nineteenth ACM symposium on Operating systems principles, SOSP 03, 2943, ACM, 2003.

[35] I. Grasso, S. Pellegrini, B. Cosenza, T. Fahringer, *LibWater: heterogeneous distributed computing made easy*, ICS 2013: 161-172, 2013

[36] W. Gropp, M. Snir, *Programming for Exascale Computers*, Computing in Science & Engineering, vol.15, no. 6, 27-35, 2013.

[37] HDF group, *HDF5*, http://www.hdfgroup.org/hdf5/.

[38] J. He, J. Bent, A. Torres, G. Grider, G. Gibson, C. Maltzahn, X.-H. Sun, *I/O Acceleration with Pattern Detection*, ACM Symp on High-Performance Parallel and Distributed Computing (HPDC), 2013.

[39] High Performance and Embedded Architecture and Compilation Consortium, *HiPEAC Roadmap*, 2011, www.hipeac.net/roadmap

[40] C.-L. Huang, J.-F. Dun, *A distributed PSOSVM hybrid system with feature selection and parameter optimization*, Applied

Soft Computing, Vol. 8, Issue 4, 1381-1391, 2008
[41] J. Hungershofer, *On the combined scheduling of malleable and rigid jobs*, Proceedings of the 16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2004), 206213, 2004.
[42] IBM, *GPFS. General Parallel File System, Efficient storage management for big data applications*, http://www.ibm.com/systems/software/gpfs.
[43] H. H. Inbarani, A. T. Azar, G. Jothi, *Supervised hybrid feature selection based on PSO and rough sets for medical diagnosis*, Computer Methods and Programs in Biomedicine, Vol. 113, Issue 1, 175-185, 2014.
[44] F. Isaila, J. Garcia, J. Carretero, R.B. Ross, D. Kimpe, *Making the Case for Reforming the I/O Software Stack of Extreme-Scale Systems*,Technical Report. Argonne Labs, IL, USA, 2014.
[45] F. Jemili, M. Zaghdoud, M. Ben Ahmed, *A Framework for an Adaptive Intrusion Detection System using Bayesian Network*, Intelligence and Security Informatics, 2007.
[46] A. Jimenez-Molina and I.-Y. Ko, *Spontaneous task composition in urban computing environments based on social, spatial, and temporal aspects*, Eng. Appl. Artif. Intell. 24, 8 2011, 1446-1460.
[47] Md. M. Kabir, Md. M. Islam, K. Murase, *A new wrapper feature selection approach using neural network*, Neurocomputing, Vol. 73, Issues 1618, 3273-3283, 2010.
[48] Y. Kessaci, N. Melab and E.-G. Talbi, *A multi-start local search heuristic for an energy efficient VMs assignment on top of the OpenNebula Cloud manager*, Future Generation Computer Systems 36, 237-256, 2014.
[49] A. K. M. Khaled, A. Talukder, M. Kirley and R. Buyya, *Multiobjective differential evolution for scheduling workflow applications on global Grids*, Concurrency and Computation: Practice & Experience, vol. 21, no. 13, 1742-1756, 2009.
[50] L. Khan, M. Awad, B. Thuraisingham, *A New Intrusion Detection System Using Support Vector Machines and Hierarchical Clustering*,The VLDB Journal, 2007.
[51] V. R. Khare, X. Yao and K. Deb, *Performance Scaling of Multi-Objective Evolutionary Algorithms*,Conference on Evolutionary Multi-Criterion Optimization, 2003.
[52] Kufrin, R. Perfsuite, *An accessible, open source performance analysis environment for Linux*, 6th International Conference on Linux Clusters: The HPC Revolution, volume 151, p 5, 2005.
[53] M. Kumar, M. Hanumanthappa, T. Kumar, *Intrusion Detection System using decision tree algorithm*, IEEE 14th International Conference on Communication Technology (ICCT), 2012.
[54] G. Korres, A. Papadopoulos, P. Katsafados, D. Ballas, L. Perivoliotis, K. Nittis, *A 2-year intercomparison of the WAM-Cycle4 and the WAVEWATCH-III wave models implemented within the Mediterranean Sea*, Mediterranean Marine Science 12(1), 129-152, 2011
[55] J. Li, W.-k. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, M. Zingale, *Parallel netcdf: A high-performance scientific i/o interface*, Proceedings of the 2003 ACM/IEEE conference on Supercomputing, p. 39, ACM, 2003.
[56] P. Llopis, F.J. Garca-Blas, F. Isaila, J. Carretero, *VIDAS: Object-based Virtualized Data Sharing for High Performance Storage I/O*, Proceedings of the ACM ScienceCloud'13, 2013.
[57] G. Llort, J. Gonzalez, H. Servat, J. Gimenez, J. Labarta, *On-line detection of large-scale parallel application's structure*, 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS), p 1-10, 2010
[58] T.V. Luong, N. Melab, E.-G. Talbi, *GPU-Based multi-start local search algorithms*, Learning and Intelligent Optimization. 6683, 321-335. 2011, Springer.
[59] T.V. Luong, N. Melab, E.-G. Talbi, *GPU computing for parallel local search metaheuristic algorithms*, Transactions on Computers 62, 173-185, 2013
[60] Lustre, *High Performance and Scalability*, http://www.lustre.org.
[61] K. El Maghraoui, T.J. Desell, B.K. Szymanski, C. A. Varela, *Malleable iterative MPI applications*, Concurrency Computat.: Pract. Exper., 21(3): 393413, 2009.
[62] J. Mair, Z. Huang, D. Eyers, L. F. Cupertino, G. Da Costa, J.-M. Pierson, H. Hlavacs, *Power Modeling*, Large-Scale Distributed Systems and Energy Efficiency: A holistic view. Jean-Marc Pierson (Eds.), John Wiley and Sons, 5, March 2015
[63] S. Matsuoka, H. Sato, O. Tatebe, M. Koibuchi, I. Fujiwara, S.Suzuki, et al, *Extreme Big Data (EBD): Next Generation Big Data Infrastructure Technologies Towards Yottabyte*, Supercomputing Frontiesr and Innovations, vol. 1, no. 2, 2014, 89-107
[64] N. Melab, K. Boufaras, E.-G. Talbi, et al, *ParadisEO-MO-GPU: a framework for parallel GPU-based local search metaheuristics*, Proceeding of the 15th annual conference on Genetic and evolutionary computation conference, 1189-1196. ACM, 2013.
[65] D. Mey, S. Biersdorf, C. Bischof, K. Diethelm, D. Eschweiler, M. Gerndt, A. Knpfer, D. Lorenz, A. Malony, W.E. Nagel, et al, *Score-P: A Unified Performance Measurement System for Petascale Applications*, Competence in High Performance Computing 2010, 85-97, Springer, 2012.
[66] J. Michalakes, J. Dudhia, D. Gill, T. Henderson, J. Klemp, W. Skamarock, W. Wang, *The Weather Research and Forecast Model: Software Architecture and Performance*, 11th ECMWF Workshop on the Use of High Performance Computing In Meteorology, 2004.
[67] B.P. Miller, M.D. Callaghan, J.M. Cargille, J.K. Hollingsworth, R.B. Irvin, K.L. Karavanic, K.L., K. Kunchithapadam, T. Newhall, *The Paradyn parallel performance measurement tool*, Computer, vol. 28, no. 11, 37-46, 1995.
[68] Mpi forum, High-end-computing systems, http://www.mpi-forum.org/.
[69] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, M. Rajarajan, *A survey of intrusion detection techniques in Cloud*, Journal of Network and Computer Applications, 2013.

[70] M. Nanni, R. Trasarti, G. Rossetti, D. Pedreschi, *Efficient distributed computation of human mobility aggregates through user mobility profiles*, Proceedings of the ACM SIGKDD International Workshop on Urban Computing (UrbComp '12), 2012, 87-94.
[71] J. Nieplocha, B. Palmer, V. Tipparaju, M. Krishnan, H. Trease, E. Apra, *Advances, Applications and Performance of the Global Arrays Shared Memory Programming Toolkit*, Intl. J. of High Performance Computing Applications 20(2): 203-23, 2006.
[72] R.V. van Nieuwpoort, J.W. Romein, *Correlating Radio Astronomy Signals with Many-Core Hardware*, Intl. J. of Parallel Programming 39(1), 88114, 2011.
[73] R.W. Numrich, J. Reid, *Co-array Fortran for parallel programming*, ACM SIGPLAN FORTRAN Forum 17(2):131, 1998.
[74] PlanetHPC, *Strategy for Research and Innovation through HPC*, November 2011, http://www.planethpc.eu/images/stories/planethpc-strategy2.pdf
[75] Partnership for Advanced Computing in Europe, *PRACE Scientific Case 2012-2020*, October 2012, www.prace-ri.eu/PRACE-The-Scientific-Case-for-HPC
[76] J. E. Pecero, P. Bouvry, H. J. Fraire Huacuja, S.U. Khan, *A Multi-objective GRASP Algorithm for Joint Optimization of Energy Consumption and Schedule Length of Precedence-Constrained Applications*, 9th IEEE International Conference on Dependable, Autonomic and Secure Computing, 2011.
[77] Orange, *Orangefs/pvfs*, http://www.pvfs.org.
[78] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2014
[79] I.Raicu, I.T. Foster, P. Beckman, *Making a case for distributed file systems at exascale*, 3rd International workshop on Large-scale system and application performance, LSAP 11, 1118, ACM, 2011.
[80] P.C. Roth, B.P. Miller, *On-line automated performance diagnosis on thousands of processes*, 11th ACM SIGPLAN symposium on Principles and practice of parallel programming, 69-80, ACM, 2006.
[81] R. Reyes, I. Lpez-Rodrguez, J.J. Fumero, F. de Sande, *accULL: An OpenACC Implementation with CUDA and OpenCL Support*, Euro-Par 2012, 871-882, 2012.
[82] S.S. Shende, A.D. Malony, *The Tau Parallel Performance System*, International Journal of High Performance Computing Applications, vol. 20, no. 2, 287-311, 2006.
[83] D. Singh and R. Garg, *A robust multi-objective optimization to workflow scheduling for dynamic grid*, International Conference on Advances in Computing and Artificial Intelligence, 183-188, 2011.
[84] M. Snir, R.W. Wisniewski, J.A. Abraham, S.V. Adve, S. Bagchi et al, *Addressing Failures in Exascale Computing*, Tech. Report ANL/MCS-TM-332, Argonne Nat'l Laboratory, Mathematics and Computer Science Division, Apr. 2013.
[85] R. Hill, *SPRINT: A new parallel framework for R*, BMC Bioinformatics. 2008
[86] R. Sudarsan, C. J. Ribbens, *ReSHAPE: A Framework for Dynamic Resizing and Scheduling of Homogeneous Applications in a Parallel Environment*, International Conference on Parallel Processing, ICPP 2007, 44-44, 2007.
[87] C. Sweetlin Hemalatha, V. Vaidehi, R. Lakshmi, *Minimal infrequent pattern based approach for mining outliers in data streams*, Expert Systems with Applications, on-line October 2014
[88] I. Syarif, A. Prugel-Bennett, G. Wills, *Unsupervised Clustering Approach for Network Anomaly Detection*, Networked Digital Technologies, Springer 2012.
[89] A. K. A. Talukder, M. Kirley, R. Buyya, *Multiobjective Differential Evolution for Workflow Execution on Grids*, 5th International Workshop on Middleware for Grid Computing, California, 2007.
[90] C. J. Tan, C.P. Lim, Y.N Cheah, *A multi-objective evolutionary algorithm-based ensemble optimizer for feature selection and classification with neural network models*, Neurocomputing, Vol. 125, 217-228, 2014.
[91] W. Tantisiriroj, S. Patil, G. Gibson, S. W. Son, S. J. Lang, R. B. Ross, *On the Duality of Data-intensive File System Design: Reconsiling HDFS and PVFS*, ACM Intl Conf for High Performance Computing, Networking, Storage and Analysis (SC), 2011.
[92] A. Tenschert, M. Assel, A. Cheptsov, G. Gallizo, E.D. Valle, I. Celino, *Parallelization and Distribution Techniques for Ontology Matching in Urban Computing Environments*, Proceedings of OM 2009.
[93] C. Tsotskas, T. Kipouros, A.M. Savill, *The Design and Implementation of a GPU-enabled Multi-objective Tabu-search Intended for Real World and High-dimensional Applications*, Procedia Computer Science, Vol. 29, 2152-2161, 2014.
[94] D. Talia, P. Trunfio, O. Verta, *The Weka4WS Framework for Distributed Data Mining in Service-oriented Grids*, Concurrency and Computation: Practice and Experience. 20(16): 1933-1951, 2008
[95] UPC Consortium, *UPC language specifications v1.2*, 2005.
[96] VERCE consortium, *VERCE*, http://www.verce.eu/
[97] M.Viñas, Z. Bozkus, B.B. Fraguela, *Exploiting heterogeneous parallelism with the Heterogeneous Programming Library*, J. Parallel Distrib. Comput. 73(12): 1627-1638, 2013.
[98] M. Wachs, M. Abd-El-Malek, E. Thereska, G. R. Ganger, *Argon: performance insulation for shared storage servers*, USENIX Conf. on File and Storage Technologies (FAST), 61-76, 2007.
[99] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, *The WEKA Data Mining Software: An Update*, SIGKDD Explorations, Volume 11, Issue 1. 2009
[100] X. Wang, R. Buyya and J. Su, *Reliability-Oriented Genetic Algorithm for Workflow Applications using Max-Min Strategy*, 9th International Symposium on Cluster Computing and the Grid, 2009.
[101] H. Xia, J. Zhuang, D. Yu, *Multi-objective unsupervised feature selection algorithm utilizing redundancy measure and negative epsilon-dominance for fault diagnosis*, Neurocomputing, Vol. 146, 113-124, 2014.
[102] H. Yoon, C.-S. Park, J. S. Kim, J.-G. Baek, *Algorithm learning based neural network integrating feature selection and classification*, Expert Systems with Applications, Vol. 40, Issue 1, 231-241, 2013.

[103] H. Yu, R. Chen, G. Zhang, *A SVM Stock Selection Model within PCA*, Procedia Computer Science, Vol. 31, 406-412, 2014.
[104] J. Yu, M. Kirley and R. Buyya, *Multi-Objective Planning for Workflow Execution on Grids*, 8th International Conference on Grid Computing, 2007.
[105] Yu Zheng, Licia Capra, Ouri Wolfson, Hai Yang, *Urban Computing: concepts, methodologies, and applications*, ACM Transaction on Intelligent Systems and Technology (ACM TIST). 2014
[106] T. Zhu, A. Tumanov, M. A. Kozuch, M. Harchol-Balter, G. R. Ganger, *PriorityMeister: Tail Latency QoS for Shared Neworked Storage*, ACM Symp. Cloud Computing (SoCC), 2014.

Edited by: Viorel Negru
Received: December 16, 2015
Accepted: January 16, 2016