Computing and Informatics, Vol. 35, 2016, 1386–1415

EFFECTIVE COMPUTATION RESILIENCE IN HIGH PERFORMANCE AND DISTRIBUTED ENVIRONMENTS

Ladislav Hluchý, Giang Nguyen, Ján Astaloš
Viet Tran, Viera Šipková

Department of Parallel and Distributed Information Processing
Institute of Informatics, Slovak Academy of Sciences
Dúbravská cesta 9, 845 07 Bratislava, Slovakia
e-mail: {hluchy, giang, astalos, viet, sipkova}.ui@savba.sk

Binh Minh Nguyen

School of Information and Communication Technology
Hanoi University of Science and Technology
e-mail: minhnb@soict.hust.edu.vn

Abstract. The work described in this paper aims at effective computation resilience for complex simulations in high performance and distributed environments. Computation resilience is a complicated and delicate area; it deals with many types of simulation cores, many types of data on various input levels and also with many types of end-users, which have different requirements and expectations. Predictions about system and computation behaviors must be done based on deep knowledge about underlying infrastructures, and simulations' mathematical and realization backgrounds. Our conceptual framework is intended to allow independent collaborations between domain experts as end-users and providers of the computational power by taking on all of the deployment troubles arising within a given computing environment. The goal of our work is to provide a generalized approach for effective scalable usage of the computing power and to help domain-experts, so that they could concentrate more intensive on their domain solutions without the need of investing efforts in learning and adapting to the new IT backbone technologies.

Keywords: Effective resilience, computational intensive strategy, computation management, mathematical modeling, complex simulations

Mathematics Subject Classification 2010: 68U35

1 INTRODUCTION

Computer simulations represent a very important and useful adjunct by understanding and solving problems. Thanks to the new computation and information technologies the field of mathematical modeling and computer simulation has enjoyed rapid advances not only in various engineering disciplines, but in social, economics and political sciences as well. However, the efficient exploitation of the computational power is still not easy and straightforward, neither from the technological background viewpoint for common end-users, nor from the computing power utility or energy consumption viewpoint for computing resources providers. Though the available services offered by computing infrastructures deliver most of the functionalities necessary for the development, mapping, scheduling and running of complex simulations, in many cases they are rather complicated and require non-trivial knowledge and skills to be used perfectly.

In the following text the term large-scale simulation (or applications) is being used for denoting computationally and/or data intensive simulation problems that can not be executed on convenient sequential computing machines, because of the complexity of input specifications, and/or requirements of diverse types, e.g. difficult real-life scenarios, massive input data, fast responses, high accuracy and reliability of calculations, and others.

To simplify the process of the job submission and to improve the execution efficiency of sophisticated, long runtime simulations, our conceptual framework for computation resilience has been designed. The realization of the framework has the capability to isolate end-users from all of the middleware difficulties, it gives them the impression of comfortable working in a coherent virtual computer center. Additionally, the framework provides the possibility to allocate a (sub)optimum hardware and software configuration needed for running large-scale simulations, which proves to be beneficial for both parties: providers of computing resources and end-users as well.

Considering all the mentioned facts, the central goal of our work was to attain the resilient and effective distribution of the computational load under the generalized framework applied to complex simulations running on High Performance Computing (HPC) infrastructures. Moreover, it simplifies the end-users life and thereby they can concentrate themselves more on their own domain problems with less effort adapting to the new IT infrastructure technologies.

The content of this paper is organized as follows:

- Section 2 presents backgrounds and the conceptual framework of the computation resilience for the workload distribution in large-scale simulations, including the principle of the computational intensive strategy (Section 2.2), and the technological concept of high performance and distributed computing systems (Section 2.1).
- Section 3 describes realizations of the conceptual framework for several real and complex applications as case studies, and herewith it illustrates the potential of

the employment of high performance and distributed infrastructures in carrying out complex and large-scale simulations.

- The conclusion of the paper is given in Section 4 introducing the main advantages and drawbacks of the work.

2 BACKGROUNDS TOWARDS COMPUTATION RESILIENCE

Computation resilience for complex and large-scale simulations in high performance and distributed environments is built based:

1. on deep knowledge about
   (a) high performance and distributed environments, such as infrastructures, interconnection networks, middleware involvements, process states, programming environment settings, and alternatives for data creations and reallocations,
   (b) computational intensive strategies;
2. on the uniform *conceptual framework* involving at least the first three of the following issues:
   (a) understanding mathematical background of the simulation models,
   (b) simulation preparation,
   (c) computation management and data distribution,
   (d) know-how for the upper layer of the computation management to obtain a more comfortable access to the high performance computational power, e.g. portal/gateway access.

Taking into account the viewpoint of computational power providers and the benefit viewpoint of end-users, the point 2 (c) above (*computation management and data distribution*) lies in the center of our interests. It determines the effective usage of the computing power, which still needs a generalized adaptive approach to produce the actionable knowledge from different domain sources.

Regarding the user-friendliness point of view, here we introduce two template-based approaches to control the created instruments that a user can choose according to his preferred comfortable level. Both approaches enable to make scenario modifications and system requirements:

1. through a separate tool using an input text file containing all the rules to vary the object quantities,
2. as a part of the functionality of the portal/gateway access, where input values specified within the web form are processed using the support for statistical calculations (if it is available in the given specific domain).

End-users can utilize the domain realizations of the conceptual framework:

1. easily to perform the complex simulation process running primarily on the IISAS HPC cluster, but with the option to exploit also the computational power of the European Grid infrastructure,
2. to find out the optimal setting of hardware and/or software configuration which seems the most suitable for making simulations with the given input scenario.

Particulars of the subjects introduced above are presented in the following subsections. The practical usage and implementation details are described in Section 3 as case studies of real complex and large-scale simulations running on IISAS high performance and distributed environments.

2.1 High Performance and Distributed Environments

The architecture of modern computing systems is different from the traditional sequential one. Most compute clusters are composed of well-elaborated nodes which can consist of multi-core processors, floating-point accelerators, graphics processing units, complex memory hierarchies and infinity bandwidth network connections. Grid systems are able to join thousands of various computing resources which may be geographically spread. A compute cluster can be exploited as local, stand-alone computer, or it can be integrated in a Grid system as a computing element, or it can be involved in the shared pool of configurable computing resources in Cloud. The local access to the cluster is provided generally through a batch system, and the remote access is enabled using the Grid or Cloud services.

Grid computing [8, 9, 10] is defined as a form of distributed system that provides a seamless access to the computational power and data storage capacity distributed over the world. In Europe, the motivation which calls Grid computing into existence is the need to analyze the huge volume of experimental data produced by the particle accelerator LHC (Large Hadron Collider) at CERN (Conseil Européen pour la Recherche Nucléaire) [56], i.e. large-scale simulations. The research and development of Grid technologies started within the project European DataGrid [39] and proceeded in projects EGEE [47], EGI-InSPIRE [46] and EGI-Engage [45]. In the last stage of EGI-InSPIRE the Grid infrastructure integrates about 350 centres distributed across 56 countries, with about 43 million jobs running per month.

Slovakia participates in the European Grid Infrastructure EGI [40] including the SlovakGrid [41]. Its role is to bind and coordinate Slovak computing centres supplying all works to be done at a national level. At present, six computing centres are integrated in the production Grid infrastructure making available the high performance computing systems. In order to meet the demands of researchers working in various scientific domains, the SlovakGrid was extended by the *Slovak Infrastructure for High Performance Computing* (SIVVP, Figure 1) [42], which can be used within the virtual organization *sivvp.slovakgrid.sk*.

The IISAS production cluster [43], which is a part of SIVVP and Grid site, has the following hardware configuration: 52×IBM dx360 M3 (2×Intel E5645 @2.4 GHz, 48 GB RAM, 2×500 GB scratch disk), 2×IBM dx360 M3 (2×Intel E5645 @2.4 GHz,

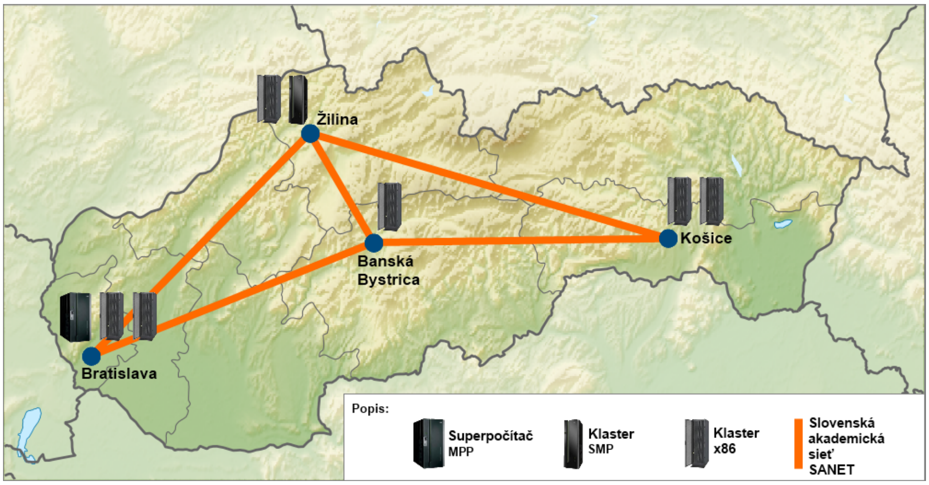


Figure 1. SIVVP – Slovak infrastructure for High Performance Computing [42]

48 GB RAM, 2 × 500 GB scratch disk, NVIDIA Tesla M2070: 6 GB RAM + 448 CUDA cores), 2 × x3650 M3 managing servers (2 × Intel E5645 @2.4 GHz, 48 GB RAM, 6 × 500 GB disks), 4 × x3650 M3 data-managing servers (2 × Intel E5645 @2.4 GHz, 48 GB RAM, 2 × 500 GB disks, 2 × 8 Gbps FC), 1 × x3550 M4 server (1 × Intel E5-2640 @2.5 GHz, 8 GB RAM, 2 × 500 GB disks), InfiniBand 2 × 40 Gbps (in 52 + 2 + 2 + 4 nodes), 2 × DS3512 with 72 TB disks.

Nowadays, IISAS also offers the Cloud infrastructure, which is a part of the EGI Federated Cloud Infrastructure designed in cooperation with the EU projects EGI-Engage H2020 [45], EGI-InSPIRE [46], Helix Nebula [84] and Cloud Plugfests [83]. On IISAS Cloud infrastructure the middleware OpenStack [85] is available together with various management and optimization tools [24, 5].

Cluster computing. Typically, to submit and start a job on a computing cluster (Figure 2 left) the workload management system (PBS Portable Batch System) [59] is used. PBS fulfills three primary roles: *Queuing* (jobs submitted to the resource management system are held until the system is ready to run them), *Scheduling* (selecting which jobs to run, when and where, according to a predetermined policy), and *Monitoring* (tracking and reserving resources, enforcing usage policy, and monitoring and tracking user jobs). PBS provides a set of commands that the user can apply for the job management operations. The job submission process returns a job identifier used in any next actions, such as job status checking, modifying, tracking, or deleting the job. After submitting, the job can get the following states: *Held*, *Queued*, *Waiting*, *Running*, *Completed*, *Exiting*, *Suspend* and *Moved*.

Grid computing. In a computational Grid, a large number of computational tasks can be distributed among geographically distant individual machines. The Grid

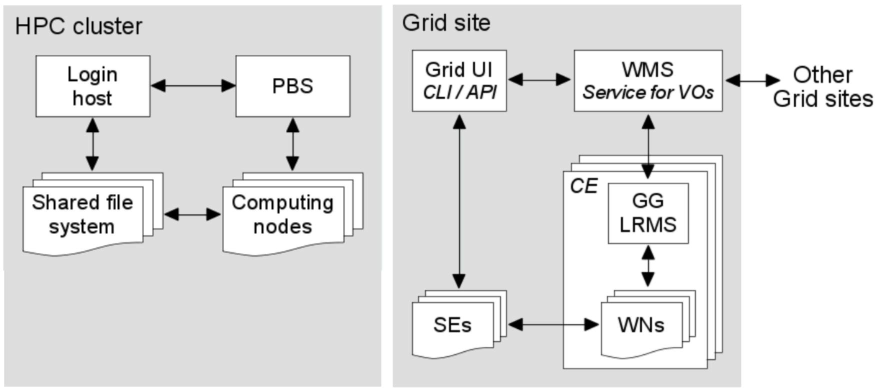


Figure 2. HPC cluster and Grid site anatomy

architecture (Figure 2 right) is based on the Open Grid Services Architecture (OGSA) and the Web Services Resource Framework (WSRF) [60], which support the coordinated resource sharing and collaboration in problem solving within the Virtual Organization (VO). The access to the globally distributed Grid resources is enabled by interacting with a Grid middleware, e.g. EMI (European Middleware Initiative) [54], which delivers high-level services [54, 55] for scheduling and running jobs, accessing and moving data, and obtaining information on the Grid infrastructure, as well as Grid applications, all embedded in a consistent Grid security framework. The Workload Management System (WMS) [61] takes care for the distribution and management of jobs across computing and storage resources available in the EGI infrastructure. After submitting, the job is pushed along the following states: *Submitted*, *Waiting*, *Ready*, *Scheduled*, *Running*, *Aborted*, *Done* and *Cleared*. If the job status is *Done*, then the user can retrieve the job output to the Grid User Interface (UI), which represents his entry point to the Grid infrastructure.

2.2 Computational Intensive Strategy

The behavior of a distributed system can be presented based on [14] as follows:

Definition 1. The system topology ST is defined as a quintuple

$$ST = (CE, SW, CL, A_{ST}, s_{ST}(t))$$

where

- CE is a set of available and assigned computing elements,
- SW is a set of switch elements,
- CL is a set of communication links,

- A_{ST} is the incident matrix of the system topology ST, which determines connections among computing elements CE and switching elements SW by means of communication links CL,
- $s_{ST}(t)$ is the state of the system topology ST in the time t.

Definition 2. The state $s_{ST}(t)$ of the system topology ST in the time t represents the vector

$$s_{ST}(t) = (s_{ce_1}(t), \dots, s_{ce_k}(t), s_{cl_1}(t), \dots, s_{cl_m}(t), s_{sw_1}(t), \dots, s_{sw_n}(t))$$

where

- $s_{ce_i}(t)$ is the state in the time t of the computing element ce_i, for $i = 1, \dots, k$,
- $s_{cl_j}(t)$ is the state in the time t of the communication link cl_j, for $j = 1, \dots, m$,
- $s_{sw_h}(t)$ is the state in the time t of the switch element sw_h, for $h = 1, \dots, n$,
- k, m, n are the numbers of computing elements in CE, communication links in CL, and switching elements in SW, respectively.

Definition 3. The computational model CM is defined as a triple

$$CM = (P, CC, V_{CM}(t))$$

where

- P is a set of processes of the computational model CM,
- CC is a set of communication channels among processes in P,
- $V_{CM}(t)$ is an incident vector of the computational model CM in the time t, it identifies linking among processes and communication channels.

Definition 4. Let the system topology be $ST = (CE, SW, CL, A_{ST}, s_{ST}(t))$, and the computational model be $CM = (P, CC, V_{CM}(t))$. Then the mapping

$$m : CM \to ST \text{ is } m(V_{CM}(t), A_{ST}) = (A_P, A_{CC})$$

where

- A_{CC} is the allocation matrix of communication channels of the computational model CM to the system topology ST,
- A_P is the allocation matrix of processes of the computational model CM to the system topology ST.

The mapping m can change over the time, it is a function of the state of the system topology $s_{ST}(t)$ at time t: $m(S_{ST}(t)) : CM \to ST$. The communication channel presents the data exchange among processes using the mapping m and is projected in computing elements in CE, communication links in CL and switching elements in SW. The realization of the mapping computational model within a complex simulation on a system topology is delicate and complicated.

In practice, the solution of this problem is moved to identification of the application programming model which should reflect the architecture of potential target computing platform. The structure of the application is strongly influenced by the problem domain and its computational nature (computation-centric, data-centric, or community-centric problems, etc.) as well. In our work we are concerned mainly with applications which can be divided according to their structural granularity into two basic classes:

- a set of *loosely-coupled tasks* which are relatively independent and can be executed separately or in a predefined order (workflow) in networks of distributed Computing Elements (CEs),
- a set of *tightly-coupled tasks* which are highly dependent on each other and run altogether on CEs equipped with a fast interconnection network; tasks communicate usually with each other over the communication channels.

2.3 Conceptual Framework for Computation Resilience

As stated above, each instrument for the computation resilience is designed and developed based on the same concept following at least the first three from the next steps and based on deep knowledge about the behavior of the underlying high performance environments.

Understanding the mathematical background of simulation models. A *mathematical model* is defined as a representation of the essential aspects about an existing system, or a system to be constructed, in a usable form. It uses a mathematical language to define the characteristics and to describe the behavior of the system or process. The term *simulation* refers to the process of applying a mathematical model of a real system in order to predict the behavior of the system from a given set of parameters and initial conditions in order to find out solutions before some time, money, and materials are invested.

A critical part of the modeling process is the *test of reliability* to ensure that the simulation using the proposed mathematical model produces authentic true results for the given input scenario. This verification task requires to investigate and resolve several issues, among others, the calibration and adjustment of the simulation model which consists generally of the analysis and comparison of simulation results and data gathered from experimental measurements.

Simulation preparations. The first step in the simulation process is the preparation of input scenarios, i.e. the creation of the mathematical representation of the simulated object. Typically, this task can be realized separately from the simulation, either manually or through various software tools which may be combined together and linked to different databases. Another possibility is to integrate the preparation of the input scenario into the simulation process as its preprocessing phase. Commonly, this problem cannot be solved fully automatically, in most cases domain-experts must determine how to define objects attributes in order to be interesting for the research and meaningful in the practice. However, a semi-automatic method working on the basis of fixed template specifying rules for the modification of the object characteristics presents a good alternative.

Computation management and data distribution. The wide availability of HPC systems and information technologies enables scientists to develop more and more complex and exact simulation models. In our work we concentrate on large-scale simulation models arising in many contexts:

- applications which come under the class of *loosely-coupled tasks*,
- applications which can be structured as a *scientific workflow* represented by a Directed Acyclic Graph (DAG).

Loosely-coupled tasks, due to their inherent parallelism, can be performed in a distributed way thus significantly reducing the execution time. In some cases, the input represents a collection including a great number of autonomous data files. Then, the set of input files can be divided into a number of independent subsets of arbitrary sizes which may be processed separately in any order on different hardware resources. To achieve a good performance, the computation management must ensure that the input data are distributed appropriately, and the strategy for scheduling of independent tasks should be chosen regarding predictions about the computation, communication and data distribution workload, and balance.

A *scientific workflow* provides a methodology of composing different predefined programs, data, services, and other software modules running in collaborative environments. Compared with a business workflow it has special features such as the computation, data and transaction intensity, less human interaction, and a large number of activities. The complexity of a workflow depends on the application, it can range from the simplest linear form up to a very compound hierarchical graph. In general, the computation management has the ability to provide definition, creation, and execution of application workflows in parallel and distributed computing environments. A scientific workflow can also contain *tightly-coupled tasks.*

Portal access as upper layer of the computation management. A portal provides a comfortable bridge to the computational power for performing complex simulations. Besides the simulation execution itself, the portal can include various functionalities which are beneficial in the phase of input preprocessing, output post-processing or in various interactivity actions. It can also serve as an upper layer in collaboration with the *computation management.*

3 EFFECTIVE RESILIENCE FOR DISTRIBUTED COMPUTATIONS

This section includes several large-scale applications for which the realization of our conceptual framework for effective computation resilience has been developed. The implementation is based on the *Manager-Worker* (or *Master-Slave*) parallel programming paradigm and management of DAG scientific workflows applied for complex and large-scale simulations. The following parts are focused mainly on the central issue "*Computation management and data distribution*" introduced in the previous section.

3.1 Training of Predictive Analytic Models with Big Data

The simplest case study of the resilient *computation management and data distribution* for independent tasks is *grid-search* realization in the field of data mining (DM) using machine learning (ML) technique [29, 7, 26] where input data for ML is large-scale or Big Data. Today, raw data comes from many processes such as information retrieval, web monitoring, ubiquitous mobile devices, IoT (Internet of Things) and IoT systems [76]. IoT refers to the world of devices connected to the Internet, which is the way the extensive large-scale data is continuously collected, concentrated and managed [76]. Mining in such data means analysing in order to obtain usable results and/or knowledge. The research in this direction is currently a hot topic in both academic and commercial spheres due to technological resources and accessible larger amount of data for the DM/ML process.

One of the most used data mining concept and methodology is CRISP-DM (Cross-Industry Process for Data Mining, Figure 3), which consists of six steps:

1. Business Understanding,
2. Data Understanding,
3. Data Preparation,
4. Modeling,
5. Evaluation,
6. Deployment.

The group of the first five steps is also called the development phase. The Data Preparation step consists of sub-steps, namely:

1. Data Transformation,
2. Exploratory Data Analysis (EDA),
3. Feature Engineering.

Each sub-step can be further divided into many smaller steps, e.g. feature extraction, feature selection/dimension reduction, etc.

Usually, a large number of available ML methods, each with a large number of specifications in combinations with variability of data and time frames, time series

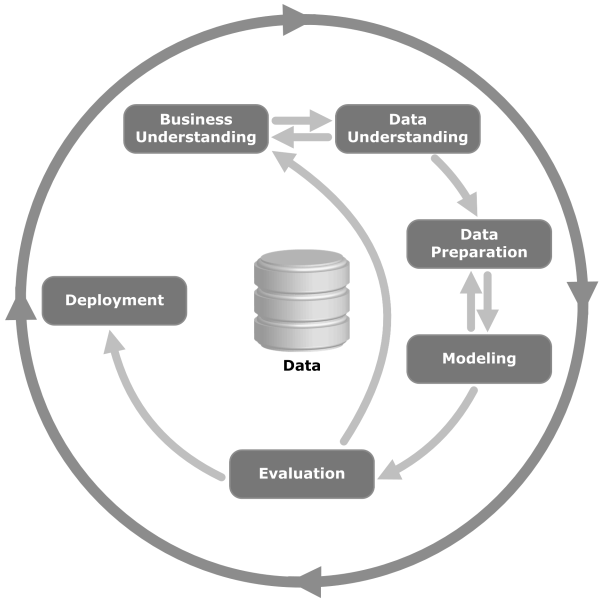


Figure 3. CRISP-DM methodology – Cross-Industry Process for Data Mining [82]

can lead to the examination of the large number of ML cycles *train-test-evaluate* in the development phase [22, 21, 28]. Other problem-solving techniques, e.g. forward selection, backward eliminations (feature selection), can also lead to computation intensive tasks, especially, when input data for ML is large-scale or Big Data. The whole process is therefore time-consuming and computation intensive due to the size of input data and the above-mentioned problem-solving techniques.

Besides the primary focus on the DM process, which is not presented in this paper, we are also interested in the performance of the development phase. The concurrent approach provides significantly lower running time in comparison with the sequential way if the training, testing and evaluation of individual predictive models can be distributed simultaneously among the cluster Computing Nodes, called also Worker Nodes (WNs, Figure 2). Theoretically, the maximum parallelism is equal to the number of necessary trained models and it is limited by the available computational power, e.g. a number of available WNs. There is also additional cost of the resource management, communications, data distribution and result aggregation.

The demonstration of the Algorithms 1 and 2 realization is shown in Table 1 as the running time of ML cycles on the HPC cluster. The shortcut NFS in algorithms stands for the Network/Shared File System (Figure 2).

- The *Manager* process (Algorithm 1) runs through three primary steps: configuration creation (lines 5 to 10) for workers' processes, work distribution (lines 11

Algorithm 1 Concurrent training of predictive models: Manager

```
Require: $location_{data}, location_{result}, timeseries, specifications, criterions$
 1: procedure MANAGER
 2:     $L \leftarrow$ generated set of data locations from $location_{data}$ and $timeseries$
 3:     $P \leftarrow$ generated set of ML specifications from $specifications$
 4:     $CF \leftarrow \emptyset$
                                                        ▷ Configuration creation
 5:     for all $location \in L$ do
 6:         for all $param \in P$ do
 7:             $cf \leftarrow \{location, param, criterions\}$
 8:             $CF \leftarrow CF \cup \{cf\}$
 9:         end for
10:     end for
                                                        ▷ Work distribution
11:     for all $cf \in CF$ do
12:         create ML train-test-evaluate cycle as $workercode(cf)$
13:         submit a request to run $Worker(cf)$ to PBS
14:     end for
                                                        ▷ Result aggregation
15:     wait for all $Worker(cf)$s to finish their works
16:     $R \leftarrow \emptyset$
17:     for all $cf \in CF$ do
18:         $result \leftarrow NFS_{location_{result}(cf)}$
19:         $R \leftarrow R \cup \{result\}$
20:     end for
21:     return $R$
22: end procedure
```

to 14), where the requests to run a worker code with a specific configuration enter the system, result aggregation (lines 15 to 22) from all concurrent running distributed workers.

- The *Worker* processes (Algorithm 2) are running parallel on WNs. Each worker process fetches assigned input data and requirements based on the specific configuration from the shared space (NFS) and returns results after training a predictive model.

The number of available WNs while testing is quite high in comparison with the number of ML cycles. The current hardware configuration (Section 2.1) allows running more than one cycle per node in the same time without significant performance degradation (concretely three simultaneous cycles per WN). One ML cycle running in the sequential mode has the following characteristics as reported by the system:

user time (seconds):	2 853.15
system time (seconds):	242.33
elapsed (wall clock) time (h:mm:ss or m:ss):	49:04.88
maximum resident set size (kbytes):	29 202 032 (approx. 29 GB)
file system inputs:	8 297 816
file system outputs:	11 475 400

Each input file in the compressed format (bzip2 with 10× compression rate) has approximately 0.5 GB size (5 GB in uncompressed text format) and contains nearly 2 million records. One ML cycle uses many input files in predefined time frames based on the setting. The ML approach applied for such big input data is *incremental learning* to overcome the machine memory limitation. In practice, the number of generated ML cycles is relatively high in repeated hundred ranges.

Algorithm 2 Training of one predictive model: Worker

```
1: procedure WORKER(CF)
2:     $data \xleftarrow{fetch} NFS_{location} : location \in cf$
3:     $result \leftarrow (model, performance) \leftarrow workercode(cf)$
4:     $result \xrightarrow{store} NFS_{location_{result}(cf)}$
5:     remove $data$
6: end procedure
```

In theory, the speedup of concurrent ML training is expected to be nearly linear, as the input dataset is divided into fairly equal groups and processed independently. In practice, the speedup is lower due to the data distribution from NFS to WNs local spaces. The use of bzip2 (free open-source software that uses Burrows-Wheeler algorithm) makes the data transfer more effective.

number of cycles	1 cycle/node	2 cycles/node	3 cycles/node
1	00:49:04	00:56:23	1:04:52
10		00:56:45	1:05:52

Table 1. Running time of ML cycles on HPC environment

The next issue that slows down simulations, is the green computing management which requires a certain time interval to start up sleeping WNs on demands. The whole runtime of all ML cycles in concurrence can be in a simplified form estimated like in the following formula:

$$t_{all} \approx \frac{n_{cycles}}{n_{nodes} \times n_{cycles\_per\_node}} (t_{fetch\_data} + t_{cycle}) + t_{management}. \qquad (1)$$

This concurrent ML approach was realized for EDA and for faster model selection (Figure 3) [82]. The realization of our conceptual framework enables

- a near-linear performance scalability for concurrent training of predictive models achieved by optimal data and computation distributions,
- shortening of time required for ML development phase, i.e. by carrying out the complex simulation process and finding out the suitable configuration setting for applications.

The conceptual framework realization was applied for two commercial applications:

1. click-through-rate advertising and
2. malicious behavior detection based on available logs from mobile devices.

The novelty is the technological meet of the HPC environment and DM using ML techniques for large-scale ML input data, which accelerates the ML development phase.

Remarks: When raw input data is really Big Data, the use of a Hadoop/Apache Spark cluster [80, 81] is often mentioned in the context of data processing, data integration and data management in data-centric distributed computing. IISAS also provides Hadoop cluster [44] with 1 × server and 11 × clients with the following node's specification: 2 × Intel® Xeon® Processor E5-2620 (15 MB cache, 2.00 GHz, 7.20 GT/s Intel® QPI, 6 × cores, 12 × threads), HyperX threading (24 simultaneous tasks per client), 32 GB RAM, 1 TB HDD per node.

In the recent years, GPU accelerators have been successfully used in the context of ML, neural networks and deep learning applications [27]. The capacity of the IISAS production cluster [43] is also extended by GPU capacity of 8 × IBM dx360 M4 (2 × Intel E5-2670 @ 2.6 GHz, 64 GB RAM, 2 × 500 GB scratch disk, 2 × NVIDIA Tesla K20 with 5 GB RAM and 2 496 CUDA cores) and 10 Gbps Ethernet.

3.2 Adaptable and Fault-Tolerant Porting Simulations to Grid

The more complex *computation management and data distribution* case study of our conceptual framework realization is the adaptable and fault-tolerant porting a large number of long running and continuously incoming simulations to the Grid infrastructure. The main problem with such simulations in the Grid is varying availability and reliability of Grid resources. If the run-time spans over multiple months and number of jobs reaches multiple thousands, end-users have to deal with erroneous Computing Elements (CEs). Grid sites became unavailable because of various reasons from network or power failures due to system maintenance or from unknown reasons.

However, the users expect that they just put their application code and data into the Grid and after a certain time they get the output data for further analysis. Therefore, the main motivation for development of adaptable and fault-tolerant job

Algorithm 3 Adaptable and fault-tolerant porting simulations to Grid: Manager

```
Require: $SE_{input}, SE_{working}, SE_{output}$, $application_{code}$, $workers_{limit}$, $time_{interval}$
 1: procedure MANAGER
 2:     $collection_{list} \leftarrow \emptyset$
 3:     $collection_{archive} \leftarrow \emptyset$
 4:     $workers_{list} \leftarrow \emptyset$
 5:     $blacklist \leftarrow \emptyset$
 6:     while True do
 7:         if $lifetime_{voms\_proxy} > 0$ then
 8:             $n \leftarrow count(SE_{input})$
 9:             $m \leftarrow workers_{limit} - count(workers_{list})$
                                        ▷ Start new workers in a new collection
10:             if $(n > 0)$ AND $(m > 0)$ then
11:                 $request \leftarrow \min(n, m) \times Worker()$
12:                 $collection_{id} \leftarrow$ submit the $request$ to WMS
13:                 $collection_{list} \xleftarrow{add} collection_{id}$
14:                 for all $worker_{id} \in collection_{id}$ do
15:                     $worker_{list} \xleftarrow{add} worker_{id}$
16:                 end for
17:             end if
                              ▷ Monitor states of collections and active workers
18:             for all $collection_{id} \in collections_{list}$ do
19:                 $state_{collection_{id}} \leftarrow$ get the state of $collection_{id}$ from WMS
20:             end for
21:             for all $worker_{id} \in workers_{list}$ do
22:                 if $state_{worker_{id}} = Done$ then
23:                     remove $worker_{id}$ from $workers_{list}$
24:                 else if $state_{worker_{id}} = Aborted$ then
25:                     $blacklist \xleftarrow{add}$ CE of $worker_{id}$
26:                 end if
27:             end for
                                              ▷ Remove finished collections
28:             for all $collection_{id} \in collections_{list}$ do
29:                 if $state_{collection_{id}} \in \{Done, Aborted\}$ then
30:                     $collections_{archive} \xleftarrow{move} collection_{id}$
31:                 end if
32:             end for
33:         else
34:             return
35:         end if
36:         sleep $time_{interval}$
37:     end while
38: end procedure
```

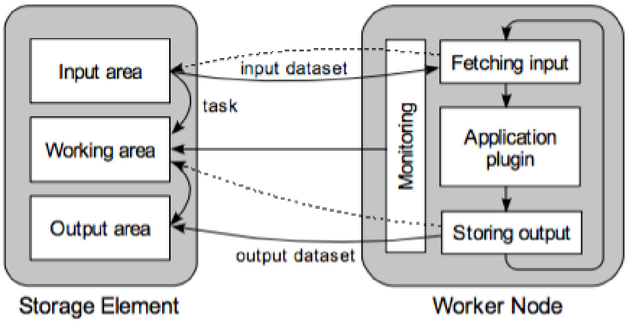


Figure 4. Adaptable and fault-tolerant porting simulations to Grid

Algorithm 4 Adaptable and fault-tolerant porting simulations to Grid: Worker

```
Require: $SE_{input}, SE_{working}, SE_{output}, application_{code}, time_{limit}$
1: procedure WORKER(ID)
2:     $timer_{id} \overset{start}{\longleftarrow} 0$
3:     start worker's background heart-beat monitoring
4:     while $SE_{input} \neq \emptyset$ do
5:         if $(lifetime_{voms\_proxy} > 0)$ AND $(time_{limit} - timer_{id} > 0)$ then
6:             $SE_{working} \overset{move}{\longleftarrow} random(dataset_{input} \in SE_{input})$
7:             $local_{dir} \overset{fetch}{\longleftarrow} dataset_{input} \in SE_{working}$
8:             $dataset_{output} \leftarrow application_{code}(dataset_{input} \in local_{dir})$
9:             $SE_{working} \overset{upload}{\longleftarrow} dataset_{output}$
10:            $SE_{output} \overset{move}{\longleftarrow} dataset_{output} \in SE_{working}$
11:        else
12:            return
13:        end if
14:    end while
15: end procedure
```

management framework is to minimize the effort needed for managing long running simulations based on reusable job wrappers.

The long running application in this case is from the *astrophysical area*. Its aim was to work out a theory of the formation of Jovian planets, Kuiper belt, Scattered Disc and Oort cloud based on the dynamical evolution of a large number planetesimals treated as test particles in the proto-planetary disc. It consists of a sequence of simulations with many independent tasks within each sub-simulation. The necessary requirement is to finish all the tasks of a given sub-simulation before starting the next sub-simulation.

The adaptable and fault-tolerant porting simulations to Grid works as follows.

- *Workers* (Algorithm 4) are executing the application code in a cycle with randomly selected input datasets downloaded from the input area on Storage Element (SE). When the processing finishes, the worker uploads output dataset back to SE.
- To identify hanging jobs or jobs that perform too slowly, the workers start a background process which is capturing monitoring information (heart beat) and sending it to SE. To avoid termination of workers by the queuing system, workers are running only for a limited time.
- To check the progress, the user just occasionally checks the contents of the working area and the output folder.
- The *Manager* main goal (Algorithm 3) is to maintain the requested number of active workers. Some of the workers do not start and some reach their time limit, so, the manager needs to detect the failed submissions and finished or waiting workers. To speed-up the start-up of workers it submits workers in so called *collections* while automatically detecting and excluding full or erroneous CE.

single CPU machine	**ideal case in Grid**	**real case in Grid**
21 years	3 months	5 months

Table 2. Runtime estimation and runtime with automatic Grid job management

Even with the automatic management some of the computing tasks have failed and needed to be re-processed which caused a delay in the overall computation. However, the effort for re-processing was limited to only moving the datasets that need to be reprocessed back to the input area and starting the manager. The runtime estimation of the simulation execution on a single CPU machine is around 21 years. In ideal case without fault facilities, the run with Grid computational power would take nearly 3 months. In reality, the simulation was completed within 5 months, which is certainly more human-acceptable response than 21 years [20].

3.3 Workflow Scheduling and Data Management

The mathematical modeling and *computation management* was put into practice to implement the workflow scheduling and data management for many sophisticated applications coming from various scientific areas. We focused especially on the *environmental modeling and forecasting* included in the *Earth science.* Within the environmental domain many simulation models have been developed and utilized. Using the HPC power with workflow scheduling and data management gives the opportunity to explore the real life situations, such as natural disasters, before they actually occurred. Simulating systems, which have been used as the heart of large-scale simulations running in our HPC environment, are listed in the following:

- Hydraulic models: DaveF [36], FESWMS (Finite Element Surface Water Modeling System) [66], SMS (Surface-water Modeling System) [77], MIKE-2D [70], TELEMAC [78], CFD (Computational Fluid Dynamics [64], which is also a core of other simulation systems, e.g. FDS [67, 17]), EPANET [65].
- Hydrological models: HSPF (Hydrological Simulation Program–Fortran) [69], NLC (National Land Cover) [72], HEC (Hydrologic Engineering Center) [68], MIKE-1D [70].
- Meteorological models: ALADIN (High Resolution Numerical Weather Prediction Project) [62], MM5 (Fifth-Generation Penn State/NCAR Mesoscale Model) [71], WRF (Weather Research and Forecasting) [79].
- Nanoscale frameworks: OOMMF (Object Oriented MicroMagnetic Framework) [73], Quantum Monte Carlo for electrons in real materials [74].

Our research and development of high-performance technologies, especially in the scheduling field [11, 12] applied for long running and complex applications in the domain of environmental modeling and forecasting, started within the project EU IST RTD ANFAS [53] and proceeded in EGEE [47], CrossGRID [52], Medi-GRID [50], K-Wf Grid [51], DEGREE [49], ADMIRE [48, 2], several international and Slovak national projects, in most cases under collaborations with domain expert institutes and commercial sphere [13, 36, 4, 19].

In most cases simulation processes have been designed as cascading workflows (e.g. Figure 5) represented by the DAG structure composed of a set of software modules, where the input, output, or execution of one or more modules is dependent on one or more other modules. Software components in the DAG may be of different types:

- sequential,
- parallel MPI (Message Passing Interface) [58],
- OpenMP (Open Multi-Processing) [57],
- hybrid parallel MPI+OpenMP,

and may collaborate with large-scale data distribution and data mining systems.

The workflow management for complex grid-based simulations controls and executes jobs with data dependences, co-operates with the *resource broker* to search out a suitable computing element for running simulations, and monitors the status of submitted jobs. It has the capabilities to work according to predefined workflow templates, to spawn the running workflow, and to modify parameters of jobs.

Before the real simulation itself, it is often necessary to carry out a number of experiments (i.e. a *test of reliability*) using some test data in order to find out the most suitable modeling configuration e.g. the number of compute nodes, cores, MPI processes and/or OpenMP threads, data and simulation time ranges, suitable software configurations and user's requirement settings etc. for given input scenarios. Due to the time and computing power consuming nature of large-scale real

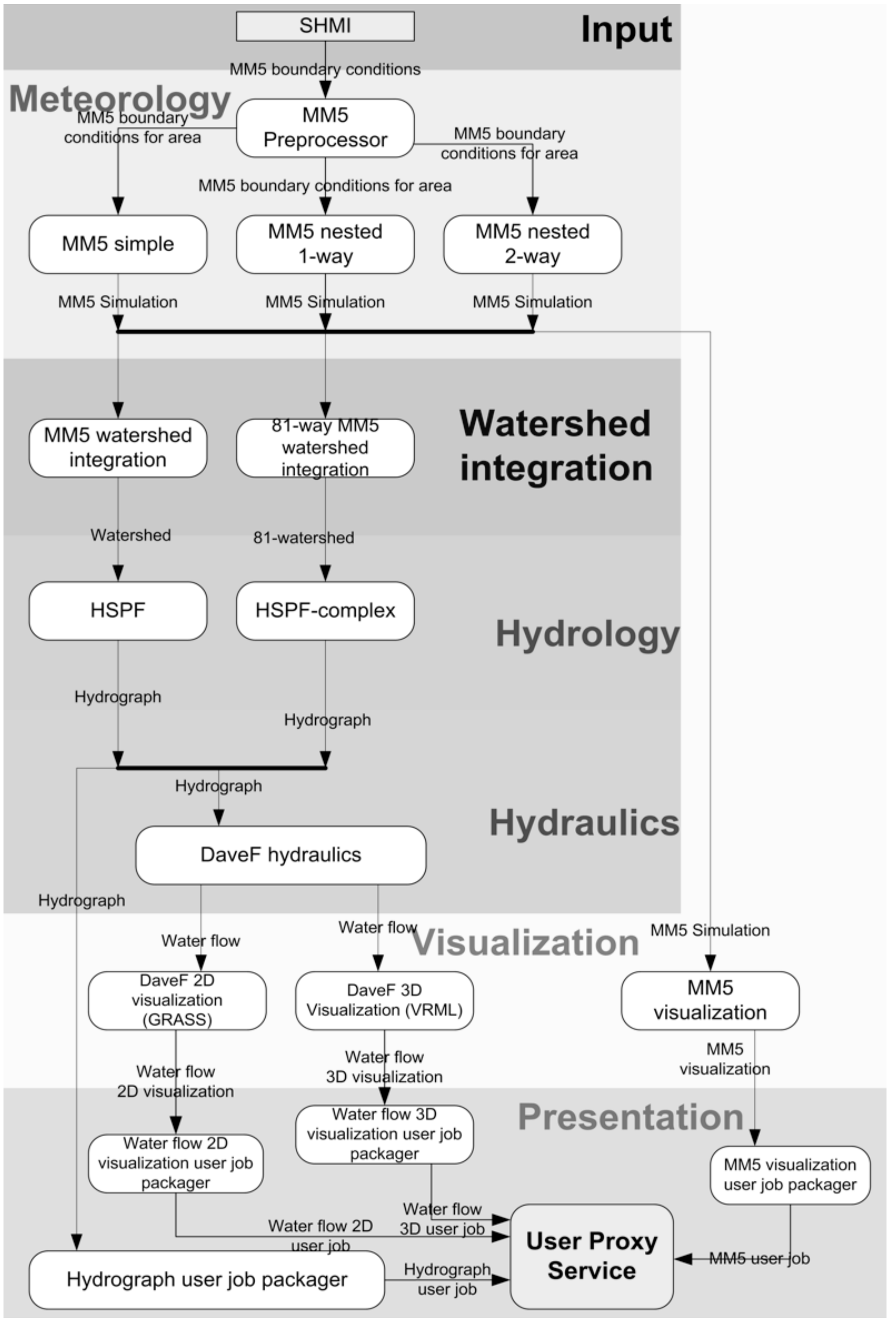


Figure 5. An example of complex grid-based flood application with DAG workflow [48] (scheme by the courtesy of Ondrej Habala)

simulations, the testing ones are usually smaller samples. At the end, the final, (sub)optimal configuration is delivered to domain experts (end-users) for repeatedly running of time and computing power consuming simulations making use of real and more extensive input data [38, 33].

Many publications can be found in the research field of workflow scheduling and management [30, 31, 15, 35, 23] within the high performance and distributed environments, including Grid and Cloud computing. These works present also many advantages and disadvantages of the variety of hardware infrastructures, difficulties arisen in the environment setup, and options for flexible handling. Many of them are narrowed to domain-oriented problem-solving solutions and optimizations. The works [18, 3] describe the ongoing research on the theme "*e-Infrastructure conceptualization and implementation as ecosystem*", that points out the need and the actuality of our conceptual framework for both parties: providers of the computing power and end-users.

3.4 Portal Access as an Upper Layer of Computation Management

The next case studies of our *computation management* are portals realizations, which came into existence as a team-work with domain experts from several scientific areas [25, 26, 19]. A *portal* represents a layer of our framework which can incorporate applications, data and tools to enable running applications on HPC infrastructures. It enables user communities associated with a common discipline to use compute and data resources in an easy way through a graphical user interface. As a result, users can focus on their applications instead of learning and managing the complex underlying infrastructure. They do not feel any difference while running their applications, no matter which HPC infrastructure is employed. If the computational capacity is available in the local computing cluster, users can exploit it directly. If not, then the Grid computational capacity accessible within our VO is provided.

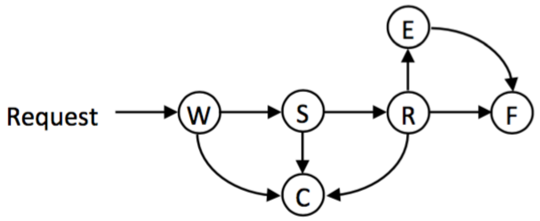


Figure 6. Simulation states through the portal

The simulation execution through the portal begins by sending a user request with input data placed in the request pool of the *Web Server* together with simulation requirements and specifications. The user has the possibility to keep track of the execution process through the service of the *Monitoring tool* (Algorithm 5) in the login host or Grid UI (see Figure 2), which are from here denoted under the same name *appServer*. The Web Server is built separately from the rest of the computing site due to security reasons.

Algorithm 5 Monitoring tool (described in collaboration with Miroslav Dobrucký)

```
Require: $webServer, appServer, datapool, interval_{MT}$
 1: procedure MONITORING
 2:     select PBS or EMI based on settings with restricted rights
 3:     while True do
 4:         synchronize $webServer_{datapool}$ and $appServer_{datapool}$
 5:         for all $sim_{id} \in appServer_{datapool}$ do
 6:             if $state_{id} \in \{Waiting, Submitted, Running\}$ then
 7:                 $state_{id} \xleftarrow{get}$ PBS/EMI
 8:                 if $state_{id} = Waiting$ then
 9:                     create $on\text{-}fly\text{-}code(sim_{id})$
10:                     submit a $request$ to run $on\text{-}fly\text{-}code(sim_{id})$ to PBS/EMI
11:                     $state_{id} \xleftarrow{get}$ PBS/EMI
12:                 else if $state_{id} = Finished$ then
13:                     $appServer_{datapool_{id}} \xleftarrow{move} output_{id}$
14:                 end if
15:             end if
16:         end for
17:         sleep $interval_{MT}$
18:     end while
19: end procedure
```

Simulation states displayed in the portal are a little different from states achieved by PBS and EMI, they are: *Waiting* (W), *Submitted* (S), *Running* (R), *Finished* (F), *Cancelled* (C), *Failed* (E). Transitions among states are described in Figure 6.

Nanoscale framework. The first presented case study for which the portal approach was realized comes from the area of nanotechnology research that is focused on the temperature effect on magnetic structures [25]. Nanoscale simulations are performed by means of the system OOMMF (Object Oriented Micro-Magnetic Framework) [73]. The original total time of simulations takes about 4.5 days on the four-core PC for small physical model. If simulations run on HPC cluster using 100 cores, the total simulation time was reduced to 8 hours for larger physical model, which additionally produces larger and more detailed output data (approx. 3.5 GB per each simulation).

The temperature is introduced to simulations via stochastic noise added to the magnetization dynamics. To get reliable results for making a reliable statistics one has to repeat simulations with various realizations of stochastic noise. This task can be partitioned naturally by running multiple instances of the simulation with various random noises. The simulation performance results demonstrate the needful of the HPC power, they cannot be achieved on the original machine configuration with a human-acceptable feedback. In the nanoscale (micromagnetism) experiments, the input is created automatically using experimental data of the fabrication device in order of thousands devices at once. This is the reason to make the statistics over fabricated samples. The portal approach is one of the ways to automatize experimental process providing feedbacks in a short time (i.e. overnight) when experiment devices can work in continuation.

Hydraulic simulations. The next case study of the portal approach realization presents a gateway to the HPC power to perform risk analysis of the water-supply in big cities [26]. The portal engages also other modern information technologies such as GIS and data mining. The integrated system for hydraulic simulations works as follows: input data are prepared in the GIS environment and ported to the HPC environment through the *hydraulic portal* (Algorithm 5 and Figure 6). After obtaining simulation results from the HPC environment, output data are transformed back into the GIS format [63] for further use by the cross-platform python script combined with ArcGIS geographical python tools. The core of whole integrated simulation system is EPANET, which calculates the friction headloss using one of the Hazen-Williams, Darcy-Weisbach, or Chezy-Manning methods, and its solver applies the Todini's gradient – a variant of the Newton-Raphson method [65]). Modeling the water quality is based on the principle of conservation of mass coupled with the reaction kinetics.

Complex grid-based applications. The portal approach was also created for complex grid-based applications as a pioneering upper layer of computation management within projects [32, 19], with the aim to provide a comfortable access to the HPC power including the collaborating features for domain-experts. There exist several grid-based implementations [16, 37] for portal/gateway access to HPC power. In general, they fulfill their expected objectives but in most cases they are too general and need a complicated customization to be adapted for the specific domain.

Though our approach to portal access is user-centric and domain-oriented, it is commonly usable and easily customizable for another domain. It is applicable not only for computing clusters and Grid, but also for cloud computing [25, 26].

4 CONCLUSIONS

Our work aims at computation management for complex and large-scale simulations intended for a domain expert user category. The main emphasis was put on the effective workload balance and data distribution along with the achievement of the possible shortest execution time. The central role of the computation resilience operating, based on the installed middleware on the target high performance machine, is to simplify the computation management process and to improve the execution efficiency of sophisticated, long runtime simulations. It brings out the following positive points:

1. It enables to carry out in a easy way the complex simulation process running primarily on IISAS production cluster (SIVVP) with the optional possibility to exploit also the computational power of the European Grid infrastructure. It is also designed towards cloud computing.
2. It enables to find out the most suitable combination of hardware and/or software configuration setting for the simulation made with the given input scenario and/or based on input data. The aim is the efficient exploitation of the computational power an its effective usage.
3. The framework realizations are in many cases implemented in collaborations with domain-experts, following and satisfying their requirements by taking all of the deployment troubles arising within a given computing environment. The final result is that users are less exposed to a burden of the underlying infrastructure complexity which enables them to concentrate more on the particular domain solutions.
4. It offers a fragile balance between the domain-oriented and user-centric vs. the generalized realization and tailoring the portal access as an upper layer of computational management.

In the near future, our work will be improved in the following:

1. Although the conceptual framework is designed with a partial tailoring portal implementation, a lot of work need to be done concerning divergences of hardware infrastructures, difficulties in the environment setup and handling that require special cases customizations.
2. (Sub)optimal configuration is still open issue with various optimization strategies towards the cloud computing and green computing.

This paper presents the work which has been a part of a long-time precise and strenuous scientific research in the Department of Parallel and Distributed Information Processing, Institute of Informatics, Slovak Academy of Sciences.

Acknowledgements

This work is supported by projects VEGA 2/0167/16, EGI-Engage EU H2020-654142, and ERDF – ITMS 26240220060. Simulations and technical realization were made on the hardware equipment obtained within the European Regional Development Fund ITMS 26230120002 SIVVP project "Slovak infrastructure for High Performance Computing". We would like to thank to all colleagues, partners and domain experts for collaborations and consultations.

REFERENCES

[1] Astaloš, J.—Babík, M. et al.: Slovak Participation in the World LHC Computing Grid. Proceedings of the 6th International Workshop on Grid Computing for Complex Problems (GCCP 2010), Bratislava, 2010, pp. 21–27, ISBN 978-80-970145-3-7.

[2] Atkinson, M. et al.: The DATA Bonanza: Improving Knowledge Discovery in Science, Engineering, and Business. Wiley Series on Parallel and Distributed Computing. John Wiley & Sons, Inc., New Yersey, 2013, pp. 301–326, ISBN 978-1-118-39864-7.

[3] Bachniak, D.—Liput, J.—Rauch, L.—Słota, R.—Kitowski, J.: Massively Parallel Approach to Sensitivity Analysis on HPC Architectures by Using Scalarm Platform. Parallel Processing and Applied Mathematics (PPAM 2015), Part I. Springer International Publishing Switzerland, Lecture Notes in Computer Science, Vol. 9573, 2016, pp. 172–181, DOI: 10.1007/978-3-319-32149-3_26.

[4] Bartok, J.—Habala, O.—Bednár, P.—Gažák, M.—Hluchý, L.: Data Mining and Integration for Predicting Significant Meteorological Phenomena. In: Sloot, P. M. A., van Albada, G. D., Dongarra, J. (Eds.): Proceedings of the 10th International Conference on Computational Science (ICCS 2010). Elsevier, Procedia Computer Science, Vol. 1, 2010, No. 1, pp. 37–46, ISSN 1877-0509.

[5] Bobák, M.—Hluchý, L.—Tran, V.: Methodology for Intercloud Multicriteria Optimization. 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2015). IEEE, 2015, pp. 1820–1825, ISBN 978-1-4673-7681-5.

[6] Cambel, V.—Tóbik, J.—Šoltýs, J.—Fedor, J.—Precner, M.—Gaži, Š.—Karapetrov, G.: The Influence of Shape Anisotropy on Vortex Nucleation in Pacman-Like Nanomagnets. Journal of Magnetism and Magnetic Materials, Vol. 336, 2013, pp. 29–36.

[7] Dlugolinský, Š.—Krammer, P.—Ciglan, M.—Laclavík, M.: MSM2013 IE Challenge: Annotowatch. Concept Extraction Challenge at the Workshop on Making Sense of Microposts Co-Located with the 22nd International World Wide Web Conference (WWW '13). May 13, 2013, Vol. 1019, pp. 21–26, ISSN 1613-0073.

[8] Foster, I.—Kesselman, C.—Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal on Supercomputer Applications, Vol. 15, 2001, No. 3, pp. 200–222.

[9] Foster, I.—Kesselman, C.—Nick, J. M.—Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure. Work Group, Global Grid Forum, June 22, 2002.

[10] Foster, I.—Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003, ISBN 1558609334.

[11] Hluchý, L.—Dobrucký, M.—Tran, V.—Astaloš, J.: The Mapping, Scheduling and Load Balancing Tools of GRADE. Parallel Program Development for Cluster Computing. Chapter 12. Nova Science Publishers, Inc., Huntington, New York, Advances in the Theory of Computational Mathematics, Vol. 5, 2001, pp. 265–278, ISBN 1-56072-865-5.

[12] Hluchý, L.—Senar, M. A.—Dobrucký, M.—Tran, V.—Ripoll, A.—Cortes, A.: Mapping and Scheduling of Parallel Programs. Parallel Program Development for Cluster Computing. Chapter 3. Nova Science Publishers, Inc., Huntington, New York, Advances in the Theory of Computational Mathematics, Vol. 5, 2001, pp. 45–68, ISBN 1-56072-865-5.

[13] Hluchý, L.—Nguyen, G.—Halada, L.—Tran V.: Cluster Computation for Flood Simulations. Springer-Verlag LNCS Vol. 2110: High-Performance Computing and Networking. Berlin Heidelberg, 2001, pp. 425–434. ISBN 3-540-42293-5, ISSN 0302-9743.

[14] Hoare, C. A. R.: Comunicating Sequential Proceses. Prentice Hall Int., 1985.

[15] Jarzab, M.—Zielinski, K.: Adaptable Service Oriented Infrastructure Provisioning with Lightweight Containers Virtualization Technology. Computing and Informatics, Vol. 34, 2015, No. 6, pp. 1309–1339.

[16] Kacsuk, P.—Farkas, Z.—Kozlovszky, M.—Hermann, G.—Balasko, A.—Karoczkai, K.—Marton, I.: WS-PGRADE/gUSE Generic DCI Gateway Framework for a Large Variety of User Communities. Journal of Grid Computing, Vol. 10, 2012, No. 4, pp. 601–630.

[17] Kasanický, T.—Zelenka, J.: Optimal Pedestrian Path Planning in Evacuation Scenario. Computing and Informatics, Vol. 33, 2014, No. 6, pp. 1269–1287.

[18] Kitowski, J.—Wiatr, K.—Dutka, L.—Twardy, M.—Szepieniec, T.—Sterzel, M.—Slota, R.—Pajak, R.: Distributed Computing Infrastructure as a Tool for e-Science. Parallel Processing and Applied Mathematics (PPAM 2015), Part I. Springer International Publishing Switzerland, Lecture Notes in Computer Science, Vol. 9573, 2016, pp. 271–280, DOI: 10.1007/978-3-319-32149-3_26.

[19] Habala, O.—Šeleng, M.—Tran, V.—Šimo, B.—Hluchý, L.: Mining Environmental Data in ADMIRE Project Using New Advanced Methods and Tools. International Journal of Distributed Systems and Technologies, Vol. 1, No. 4, pp. 1–13, ISSN 1947-3532.

[20] Leto, G.—Astaloš, J.—Jakubík, M.—Neslušan, L.—Dybczynski, P. A.: The Usage of the Grid in the Simulation of the Comet Oort-Cloud Formation. Grid Computing: Towards a Global Interconnected Infrastructure. Springer-Verlag, London, 2011, pp. 293–306, ISBN 978-0-85729-675-7.

[21] Machová, K: Strojové učenie v systémoch spracovania informácií. Edícia vedeckých spisov Fakulty elektrotechniky a informatiky TU Košice, Slovakia, 2010. `http://people.tuke.sk/kristina.machova/pdf/monoSUvSSI.pdf` (in Slovak).

[22] Machová, K: Strojové učenie princípy a algoritmy. Fakulty elektrotechniky a informatiky TU Košice, Slovakia, 2002. `http://people.tuke.sk/kristina.machova/pdf/SU4.pdf` (in Slovak).

[23] Monge, D. A.—García Garino, C.: LOGOS: Enabling Local Resource Managers for the Efficient Support of Data-Intensive Workflows within Grid Sites. Computing and Informatics, Vol. 33, 2014, No. 1, pp. 109–130.

[24] Nguyen, B. M.—Tran, V.—Hluchý, L.: A Novel Approach for Developing Interoperable Services in Cloud Environment. International Conference on Information Networking (ICOIN), IEEE, 2013.

[25] Nguyen, G.—Hluchý, L.—Tóbik, J.—Šipková, V.—Dobrucký, M.—Astaloš, J.—Tran, V.—Andok, R.: Unified Nanoscale Gateway to HPC and Grid Environments. Proceedings of the Symposium on Information and Communication Technology, ACM New York, NY, USA, 2014, pp. 85–91, ISBN 978-1-4503-2930-9.

[26] Nguyen, G.—Šipková, V.—Krammer, P.—Hluchý, L.—Dobrucký, M.—Tran, V.—Habala, O.: Integrated System for Hydraulic Simulations. Computing and Informatics, Vol. 34, 2015, No. 5, pp. 1065–1089.

[27] Nguyen, G.—Astaloš, J.—Hluchý, L.: Considerations about Data Processing, Machine Learning, HPC, Apache Spark and GPU. The Conjunction Conference of the 11th Workshop on Intelligent and Knowledge Oriented Technologies, and the Conference Data and Knowledge (WIKT-DAZ 2016), Smolenice, Slovakia, 2016.

[28] Paralič, J.—Furdík, K.—Tutoky, G.—Bednár, P.—Sarnovský, M.—Butka, P.—Babič, F.: Dolovanie znalosti z textov. Technická Univerzita v Košiciach, Slovakia, 2010. ISBN 978-80-89284-62-7. `http://people.tuke.sk/jan.paralic/knihy/DolovanieZnalostizTextov.pdf` (in Slovak).

[29] Paul, S.: New Fundamental Technologies in Data Mining: Parallel and Distributed Data Mining. 2011, pp. 43, ISBN 978-953-307-547-1. `http://cdn.intechopen.com/pdfs-wm/13261.pdf`.

[30] Quarati, A.—Danovaro, E.—Galizia, A.—Clematis, A.—Agostino D.—Parodi, A.: Scheduling Strategies for Enabling Meteorological Simulation on Hybrid Clouds. Journal of Computational and Applied Mathematics, Vol. 273, 2015, pp. 438–451, ISSN 0377-0427.

[31] Schuller, F.—Ostermann, S.—Prodan, R.—Mayr, G.: Experiences with Distributed Computing for Meteorological Applications: Grid Computing and Cloud Computing. Journal of Grid Computing, Vol. 8, 2015, pp. 1171–1199, ISSN 1570-7873.

[32] Šimo, B.—Mališka, M.—Ciglan, M.—Hluchý, L.: Medigrid Application Portal. International Workshop on Environmental Applications and Dixtributed Computing (EADC 2006), 2006, pp. 197–203, ISBN 80-969202-4-3.

[33] Šipková, V.—Hluchý, L.—Dobrucký, M.—Bartok, J.—Nguyen, B. M.: Manufacturing of Weather Forecasting Simulations on High Performance Infrastructures. 2016 IEEE 12th International Conference on eScience, Environmental Computing Workshop (ECW), Baltimore, Maryland, USA, 2016.

[34] Šoltýs, J.—Gaži, Š—Fedor, J.—Tóbik, J.—Precner, M.—Cambel., V.: Magnetic Nanostructures for Non-Volatile Memories. Microelectronic Engineering, Vol. 110, 2013, pp. 474–478.

[35] Toporkov, V.—Yemelyanov, D.—Potekhin, P.—Toporkova, A.—Tselishchev, A.: Metascheduling and Heuristic Co-Allocation Strategies in Distributed Computing. Computing and Informatics, Vol. 34, 2015, No. 1, pp. 45–76.

[36] Tran, V.—Hluchý, L.—Froehlich, D.—Castaings, W.: Parallelizing Flood Model for Linux Clusters with MPI. Parallel Processing and Applied Mathematics (PPAM 2003). Springer Berlin Heidelberg, Lecture Notes in Computer Science, Vol. 3019, 2003, pp. 521–527, ISBN 3-540-21946-3, ISSN 0302-9743.

[37] Tugores, M. A.—Colet, P.: Web Interface for Generic Grid Jobs, Web4Grid. Computing and Informatics, Vol. 31, 2012, No. 1, pp. 173–187.

[38] Weisenpacher, P.—Glasa, J.—Halada, L.—Valášek, L.—Šipková, V.: Parallel Computer Simulation of Fire in Road Tunnel and People Evacuation. Computing and Informatics, Vol. 33, 2014, No. 6, pp. 1237–1268.

[39] European Data Grid (EDG). http://eu-datagrid.web.cern.ch/eu-datagrid/.

[40] European Grid Infrastructure (EGI). http://www.egi.eu/.

[41] SlovakGrid – Slovak Grid Initiative. http://www.slovakgrid.sk/.

[42] Slovak Infrastructure for High Performance Computing. http://www.sivvp.sk/.

[43] IISAS HPC Production Cluster (SIVVP). http://hpc.ui.savba.sk/.

[44] IISAS DAS Data Analytics Supercomputer. http://ikt.ui.sav.sk/index.php?n=Main.Laboratory.

[45] EGI-Engage – Engaging the Research Community Towards an Open Science Commons. EU H2020-654142. http://www.egi.eu/about/egi-engage/.

[46] EGI-InSpire – Integrated Sustainable Pan-European Infrastructure for Researchers in Europe. EU FP7-261323 RI. http://www.egi.eu/projects/egi-inspire/.

[47] EGEE – Enabling Grid for E-SciencE. EU RTD INFSO-RI-508833. http://www.eu-egee.org/.

[48] ADMIRE: Advanced Data Mining and Integration Research for Europe. EU IST RTD FP7-215024. http://www.admire-project.eu/.

[49] DEGREE: Dissemination and Exploitation GRids in Earth SciencE. EU IST RTD FP6-034619. http://degree.ipgp.jussieu.fr:8080/DEGREE.

[50] MEDIGRID: Mediterranean Grid of Multi-Risk Data and Models. EU FP RTD GOCE-CT-2003-004044.

[51] K-Wf Grid: Knowledge-Based Workflow System for Grid Applications. EU IST RTD FP6-511385. http://www.kwfgrid.net/.

[52] CROSSGRID: Development of Grid Environment for Interactive Applications. EU FP RTD IST-2001-32243.

[53] ANFAS: datA fusioN for Flood Analysis and decision Support. EU FP RTD IST-1999-11676.

[54] European Middleware Initiative (EMI). http://www.eu-emi.eu/, https://edms.cern.ch/document/674643/.

[55] gLite, 2012. https://edms.cern.ch/file/722398/1.4/gLite-3-UserGuide.pdf.

[56] LHC – Large Hadron Collider, the European Organization for Nuclear Research. http://public.web.cern.ch/public/en/LHC/, http://lcg.web.cern.ch/lcg/.

[57] OpenMP. http://openmp.org/wp/.

[58] Open MPI. https://www.open-mpi.org/.

[59] PBS Portable Batch System. http://www.mcs.anl.gov/research/projects/openpbs/, http://www.adaptivecomputing.com/.

[60] Web Services Resource Framework (WSRF). https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf/.

[61] Workload Management System (WMS). http://web.infn.it/gLiteWMS/images/WMS/Docs/wmproxy-guide.pdf.

[62] Aladin – High Resolution Numerical Weather Prediction Project. http://www.cnrm-game-meteo.fr/aladin/.

[63] ArcGIS. http://www.arcgis.com/.

[64] CFD – Computational Fluid Dynamics. https://en.wikipedia.org/wiki/Computational_fluid_dynamics.

[65] EPANET Software (US Environmental Protection Agency). http://www.epa.gov/nrmr/wswrd/dw/epanet.html, http://en.wikipedia.org/wiki/EPANET.

[66] Flow and Sediment Transport Modeling with SMS. http://www.aquaveo.com/software/sms-feswms.

[67] FDS – Fire Dynamics Simulator. https://en.wikipedia.org/wiki/Fire_Dynamics_Simulator.

[68] HEC – The Hydrologic Engineering Center. http://www.hec.usace.army.mil/.

[69] HSPF – Water Resources of the United States. http://water.usgs.gov/software/HSPF/.

[70] MIKE Powered by DHI. https://www.mikepoweredbydhi.com/.

[71] MM5 Community Model. http://www2.mmm.ucar.edu/mm5/.

[72] NLC Groundwater Infiltration Study. http://www.rwe.com/web/cms/en/216480/rwe-technology-international/.

[73] OOMMF – The Object Oriented MicroMagnetic Framework. http://math.nist.gov/oommf/.

[74] QWalk – Quantum Monte Carlo for Electrons in Real Materials. http://qwalk.github.io/mainline/.

[75] QGIS. http://www.qgis.org/en/site/.

[76] SCADA – Supervisory Control and Data Acquisition. http://scada.com/.

[77] SMS Surface-Water Modeling System. http://www.xmswiki.com/wiki/SMS:SMS.

[78] TELEMAC – Integrated Suite of Solvers for Use in the Field of Free-Surface Flow. http://www.opentelemac.org/.

[79] The Weather Research and Forecasting Model. http://www.wrf-model.org.

[80] Apache Hadoop – Open-Source Software for Reliable, Scalable, Distributed Computing of Large Data Sets. http://hadoop.apache.org/.

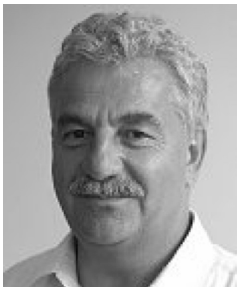
[81] Apache Spark – Fast and General Engine for Large-Scale Data Processing. http://spark.apache.org/.

[82] CRISP-DM: Cross Industry Standard Process for Data Mining. Picture Source: Kenneth Jensen – Own Work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=24930610, http://www.sv-europe.com/crisp-dm-methodology/.

[83] Cloud Plugfests. http://www.cloudplugfest.org/.

[84] Helix Nebula. http://helix-nebula.eu/.

[85] OpenStack. http://www.openstack.org/.



Ladislav Hluchý (Associated Professor, M.Sc., Ph.D.) is Head of the Parallel and Distributed Information Processing Department, and former Director of the Institute of Informatics, Slovak Academy of Sciences (IISAS) for more than 20 years. He received his M.Sc. and Ph.D. degrees, both in computer science. He is R & D Project Manager, Work-Package Leader and coordinator in a number of 4th, 5th, 6th, 7th and H2020 EU IST RTD projects as well as Slovak R & D projects (VEGA, APVV, SPVV). His research topics are focused on parallel and distributed computing, large scale applications, cluster/grid/cloud computing, service oriented computing and knowledge oriented technology. His highlighted research works are within EU IST RTD projects EGI-Engage H2020-654142 Engaging the Research Community towards an Open Science Commons, EGI-InSPIRE FP7-261323, EGEE III FP7-222667, EGEE II FP6 RI-031688, EGEE FP6 INFSO-RI-508833, REDIRNET FP7-607768, VENIS FP7-284984, SeCriCom FP7-218123, Commius FP7-213876, ADMIRE FP7-215024, DEGREE FP6-034619, INTAS FP6 06-1000024-9154, int.eu.grid FP6 RI-031857, K-Wf Grid FP6-511385, MEDIGRID FP6 GOCE-CT-2003-004044, CROSSGRID FP5 IST-2001-32243, PELLUCID FP5 IST-2001-34519, ANFAS FP5 IST-1999-11676, SEIHPC, SEPP and HPCTI as well as in international and Slovak national research projects. He is a member of IEEE, e-IRG, EGI Council, the Editor-in-Chief of the current contents (CC) journal Computing and Informatics (CAI). He is also (co-)author of scientific books and numerous scientific papers (more than 300), contributions and invited lectures at international scientific conferences and workshops. He is a supervisor and consultant for Ph.D. study at the Slovak University of Technology (STU) in Bratislava.

Giang Nguyen is a senior scientific researcher at the Institute of Informatics, Slovak Academy of Sciences (IISAS) with research topics focused on high performance and distributed computing, information processing and knowledge discovery. She received her M.Sc. and Ph.D. degrees in applied informatics from the Slovak University of Technology (STU) in Bratislava. She is (co-)author of scientific papers and has participated in EU RTD FP, international and Slovak national projects. She is a member of the IISAS scientific board, program committees and reviewer for international scientific conferences, journals and projects.

Ján Astaloš received his M.Sc. degree in informatics and information technology from the Slovak University of Technology (STU) in Bratislava. He is currently a researcher at the Department of Parallel and Distributed Information Processing, Institute of Informatics, Slovak Academy of Sciences (IISAS). His main research topics include high performance and distributed computing. He is (co-)author of scientific papers and has participated in number of EU RTD FP, international and Slovak national research projects.

Viet Tran is a senior scientific researcher at the Institute of Informatics, Slovak Academy of Sciences (IISAS) with research focused on high-performance distributed computing and cloud computing. He received his M.Sc. degree in informatics and information technology and his Ph.D. degree in applied informatics from the Slovak University of Technology (STU) in Bratislava. He has participated in a number of EU RTD FP projects as well as international and Slovak national research projects as a work-package leader, key person and scientific coordinator. He is (co-)author of scientific books and scientific papers, member of program committees, reviewer for international scientific conferences, journals and projects.

Viera Šipková is a researcher at the Department of Parallel and Distributed Information Processing, Institute of Informatics, Slovak Academy of Sciences (IISAS). She received her M.Sc. degree in mathematics and the R.N.Dr. degree in computer science from the Comenius University in Bratislava. Her research deals with parallel and distributed computing technologies focused on development and porting complex scientific applications. She has more than 10 years working experience at the Vienna University and Vienna University of Technology and has participated in many EU RTD FP projects as well as international and Slovak national research projects.

Binh Minh Nguyen received his Dipl. Eng. degree in computer-aided design from the Institute of Automation and Information Technologies, Tambov State Technical University (Russia) in 2008 and his Ph.D. degree in applied informatics from the Faculty of Informatics and Information Technology, Slovak University of Technology (STU) in Bratislava (Slovakia) in 2013. Since 2008 to 2013, he worked as researcher and project assistant at the Department of Parallel and Distributed Information Processing, Institute of Informatics, Slovak Academy of Sciences (IISAS). Currently, he is a lecturer and scientific researcher at the School of Information and Communication Technology, Hanoi University of Science and Technology (Vietnam). His research interests include cloud computing, distributed systems, Internet of Things (IoT) and data integration.