ISSN 0361-7688, Programming and Computer Software, 2020, Vol. 46, No. 8, pp. 483–502. © Pleiades Publishing, Ltd., 2020.

Mitigating Uncertainty in Developing and Applying Scientific Applications in an Integrated Computing Environment

A. Tchernykh[a,b,c,*], **I. Bychkov**[d,**], **A. Feoktistov**[d,***], **S. Gorsky**[d,****], **I. Sidorov**[d,*****], **R. Kostromin** [d,******], **A. Edelev**[e,*******], **V. Zorkalzev**[h,********], **and A. Avetisyan**[c,*********]

[a]*CICESE Research Center Carr Tijuana-Ensenada 3918, Zona Playitas, Ensenada, B.C., 22860 Mexico*

[b]*South Ural State University, Chelyabinsk, Chelyabinsk, 454080 Russia*

[c]*Ivannikov Institute for System Programming of RAS, Moscow, 109004 Russia*

[d]*Matrosov Institute for System Dynamics and Control Theory, SB of RAS, Irkutsk, 664033 Russia*

[e]*Melentiev Energy Systems Institute, SB of RAS, Irkutsk, 664033 Russia*

[h]*Limnological Institute, SB of RAS, Irkutsk, 664033 Russia*

**e-mail: chernykh@cicese.mx*

***e-mail: bychkov@icc.ru*

****e-mail: agf65@icc.ru*

*****e-mail: gorsky@icc.ru*

******e-mail: ivan.sidorov@icc.ru*

*******e-mail: kostromin@icc.ru*

********e-mail: flower@isem.irk.ru*

*********e-mail: zork@isem.irk.ru*

**********e-mail: arut@ispras.ru*

Received April 10, 2020; revised May 20, 2020; accepted June 28, 2020

Abstract—Effective solving complex mathematical modeling problems is based on the use of high-performance computing. Clouds, grids, and public access supercomputer centers are commonly used platforms. Their integration into a unified environment provides possibilities for carrying out mass large-scale scientific experiments and efficient scalable resource allocation at different stages of the application design and execution. However, end-users have to carefully select optimization criteria such as completion time, deadlines, reliability, cost, etc. It is a complicated problem due to integrated resources differ significantly in their computing capabilities, hardware and software platforms, system architectures, user interfaces, etc. The paper presents new features of the Orlando Tools framework for the development of distributed applied software packages (scalable scientific applications) that mitigates various types of uncertainties arising from the job distribution in the integrated computing environment. It provides continuous integration, delivery, and deployment of applied and system software to significantly mitigate the negative impact of uncertainty on problem-solving time, computation reliability, and resource efficiency. An experimental analysis of the sustainable design and development of the real energy sector clearly demonstrates the advantages of the tools.

DOI: 10.1134/S036176882008023X

1. INTRODUCTION

An analysis of current trends in the development of high-performance computing towards mathematical modeling of complex systems allows to draw several important conclusions. Usually, mathematical modeling of such systems related to solving NP-hard problems that requires high-performance computing systems. However, such systems can significantly differ in their computational capabilities, hardware and software platforms, system architectures, interconnects, user interfaces, and other components. Often, end-users are faced with various difficulties of their use [1], especially, when end-users switch to a new system:

• The need for considering the subject-domain specifics features, including the use of weakly structured data, presence of different uncertainties, changing of a problem sizes, etc.

• Implementing experiments in a heterogeneous systems, for example, the adaptation of problem-solving algorithms to different architectures,

• Predicting the performance and applications' makespan.

It is extremely demanded the design of universal information and computing environment that provides instrumental support of preparing and carrying out large-scale experiment in a heterogeneous environment and its specialization for a specific subject-domain [2]. Such an environment must expand the spectrum of models, algorithms, applications, and subject databases used. In addition, it has to provide the opportunity for the participation of various scientific groups in the preparation and carrying out of experiments using the necessary resources.

Clouds, grids, and public access supercomputing centers are three organizational and functional platforms of parallel and distributed computing that are highly sought-after by the scientific community [3].

Grid-systems support standard software and services for collaborative sharing and managing of federated and geographically distributed resources for computing and data processing [4].

Cloud platforms provide elasticity and scalability of computation resources, data storage, or network bandwidth with the required quality [5]. However, end-users are forced to deliver, deploy, and manage necessary software and data on the top of the allocated infrastructure. Moreover, the use of cloud resources and services is often more expensive than carrying out experiments in the grid-systems.

Public access supercomputer centers have the following distinctive advantages [6]:

• Provision of paid and free resources and services according to administrative policies,

• Providing basic system and application software,

• Providing unified and tightly regulated access to computing resources,

• Evaluation of the effectiveness of computing resources,

• Technical and methodological support for the end users.

In general, the resources of public access supercomputer centers are non-dedicated and shared by all users of the center. However, users could want to use grid and cloud resources together to solve complex problems [7]. Differences in the methods of access to above-listed resources and their use significantly limit the scope of high-performance computing. Thus, the integration of grid and cloud computing including resources of public access supercomputing centers can provide tremendous opportunities for the implementation of mass large-scale scientific experiments [8].

In the paper, we propose new features of the Orlando Tools framework (OT) [9]. We apply OT as the basis of an integrated computing environment for developing and applying distributed applied software packages. It is assumed that the environment unifies three platforms discussed above: clouds, grids, and public access supercomputing centers.

Distributed applied software packages represent a special class of scientific applications [10]. In such packages, the software has a modular structure. Package modules implement algorithmic knowledge about solving problems.

An abstract program (problem-solving scheme) describes the set of modules used to solve a problem and order of their execution. Directed Acyclic Graph (DAG) is applied to reflect relations and data flow between modules.

The concept of a problem-solving scheme corresponds to an abstract scientific workflow [11]. The specification of a problem-solving scheme (computational job) for meta-schedulers describes information about the used package modules, their input and output data, requirements on computation resources, data storages, and communication network.

In contrast with scientific workflow systems, the packages are associated with solving a class of problems in a certain subject domain considering several conditions. The important ones include the changes in algorithmic knowledge, presence of various categories of package users, and use of integrated heterogeneous resources based on various parallel and distributed computing paradigms.

New features of OT provide the processing of weakly structured data, automation of the assembly, debugging, and joint testing of package modules, prediction of the modules runtimes, and their classification on heterogeneous resources. As a result, it improves the quality of resource allocation.

The advantages of this subsystem are demonstrated and validated in practice by the development of two versions of the package for solving the problem of the sustainable development of the energy sector of Vietnam: with [12] and without this subsystem [13].

In the first case, the results show a significant reduction of the overall time for preparing and conducting experiments.

The remainder of the paper is structured as follows. Section 2 provides a brief overview of approaches related to the integration and virtualization of resources with different computational characteristics into the unified heterogeneous environment. In addition, we give the current state in the continuous integration field related to the development of scientific applications. In Section 3, we consider OT for development scientific applications. We discuss a new subsystem for the continuous integration, delivery, and deployment of applied and system software. The Section 4 demonstrates mitigating uncertainty in mapping jobs on resources based on the job classification and administrative skills. Section 5 describes an example of a job flow management for a scalable application to solve the complex practical problem. The gained experience is discussed in Section 6. Finally, Section 7 concludes the paper.

2. RELATED WORK

An integrated software environment to support large-scale scientific experiments using various paradigms of high-performance computing is not discussed in depth in the scientific literature. Nevertheless, we can distinguish several interesting projects of great practical importance for the implementation of collaborative computing.

The modular toolbox Distributed and Unified Numeric Environment (DUNE) for solving partial differential equations is presented in [14]. It provides efficiency mechanisms in high-performance computing for scientific applications.

Integration of developed applications, for example, Computer-Aided Engineering systems, into a distributed computing environment can be performed on the basis of the service-oriented approach proposed in [15].

A new algorithm for computation managing in such an environment has been developed in [16].

The Project SPPEXA supports various studies in integrated developing computational algorithms, system software, applied software, data management and exploration, programming environment, and software tools [17]. A wide spectrum of scientific institutions involved in the project.

Afgan et al. [18] propose the web-based Galaxy platform for a scientific analysis of large biomedical datasets in genomics, proteomics, metabolomics, and imaging. Many scientists across the world actively use it for large-scale experiments.

Ananthakrishnan et al. [19] provide the elastic cloud platform Globus for the implementation, deployment, and operation of services to support similar experiments. Similar tools implemented using the multi-agent approach are offered in [20].

Often, scientists need to execute the composition of various services with certain quality criteria. The problem of creating such a composite combination of services is NP-hard. QoS-aware cloud service composition using eagle strategy is presented in [21].

The concept and functioning principles of an integrated computing environment are proposed in [2]. The authors provide the development tools for implementing the main stages of large-scale computational experiments and supporting a long-life cycle of applied and system software. However, existing projects leave many problems unresolved. The most important ones include [2, 17]:

• Study of application functioning in heterogeneous resources,

• Designing applications considering the architecture of the environment,

• Applying middleware to support the interaction between applications and operating systems,

• Integration of capabilities of middleware and applications for managing jobs and resources,

• Intellectualization of job management,

• Development of flexible interfaces for various categories of users to access to application components,

• Automation of the controlling, building, and testing of system and application software,

• Elicitation, analysis, and transformation of subject data.

These problems are also relevant for scientific workflows [22]. Their solution facilitates mitigating the uncertainty of the workflow distribution.

Scientific workflow systems and distributed applied software packages have several differences. Usually, a scientific workflow applications such as LIGO [23], Montage [24], CyberShake [25], SIPHT [26] or Epigenomics [27] are oriented to solve a single problem in some subject domain. A package is designed to solve a class of problems. Problem-solving schemes reflect the variety of algorithms used in the package. Each workflow has no large variability [28]. Only the input data are usually changed. Thus, there is no need to support the continuous integration of their software.

On the contrary, packages are actively developing by their users. The permanent changes of algorithmic knowledge are expected.

In the packages, new problem-solving schemes can be designed and existing schemes can be modified. In [29], we provide an analysis of the state of the art in this field. It shows that, currently, there are no tools, which would allow us to implement the continuous integration of software for distributed application packages.

To solve this problem, the development of new specialized tools is required. Such tools should implement special functions for continuous integration caused by the package specifics and interact with existing continuous integration tools that implement traditional functions. The provided comprehensive comparative analysis of the possibilities of known tools for continuous integration shows that the GitLab system is most suitable for implementing such interaction [30].

There are various categories of package users (developers, administrators, and end-users). Developers create and modify the package. End-users apply it to solve a class of problems provided by the package. They can formulate new problems and create various schemes for their solving using existed, modified or newly applied software. Administrators determine resource use policies for running the package. Users may have various affiliations and be in different geographically distant regions.

Unlike most modern scientific workflows [31], packages are developed to run in a heterogeneous distributed computing environment that can integrate resources of cloud platforms, grid-systems, and public access supercomputing centers.

Resources that are integrated within such an environment can be dedicated and non-dedicated. Scientific workflow systems usually apply traditional virtualization tools [32]. Their capabilities are not enough to integrate the aforementioned resources into unified virtual environment.

A detailed comparative analysis of virtualization systems is represented in [12]. It showed that in a dynamic environment it is necessary to apply middleware to support effective interaction with various hypervisors and container management systems. Such a middleware that can be used in dedicated resources is the OpenStack platform [33]. However, for non-dedicated resources, it is necessary to develop a specialized tool for launching virtual machines (VM).

The aforementioned differences cause various types of uncertainties that arise in computation planning and resource allocation. Among them are uncertainties in the resource capabilities, jobs characteristics, environment states, preferences of resource owners and users, quality of services, etc. From uncertainty point of view, the main challenges in scalable scientific applications while tackling applications at scale are computation planning and resource allocation. Studies of improving computation planning and resource allocation through decreasing the above-listed uncertainties are actively developing.

Methods for the analysis of workflow structures and their decomposition are considered in [34–36]. The workflow analysis allows clarifying the components properties. The decomposition is carried out in order to rationally distribute the components on resources.

In addition, automation of planning workflows (problem-solving schemes) for non-procedural formulations of problems enables significantly simplify the selection of applications needed for solving problems if there is uncertainty in the software use [37].

A new algorithm for decreasing the jobs distribution uncertainty of problem-oriented applications is proposed in [38]. It considers both the specifics of problems being solved and features of available resources.

Ramírez-Velarde et al. [39] study adaptive grid-resource allocation under uncertainty. They propose a special model of execution delay for job makespan prediction and a new strategy for adaptive stochastic allocation of jobs.

Tchernykh et al. [40] propose an approach to modeling cloud computing with uncertainty addressing resource provisioning in a hybrid public-private cloud environment. This study provides improving the efficiency of cloud resource provisioning considering problems of their confidentiality, integrity, and availability.

Babenko et al. [41] proposed the arithmetic error correction codes to reduce uncertainty completion time, deadlines, reliability, and cost. This approach focuses on redundant computing systems.

Uncertainty of quality of services associated with the preference coordination for owners and users of resources can be successfully provided using multi-agent technologies. For example, Singh [42] describes a Multi-Agent System (MAS) to support the scalability in cloud computing by finding the clouds satisfied to user's requests when the currently used cloud becomes overloaded. Kalyaev et al. [43] present the multi-agent scheduling in cloud computing environments considering job completion deadlines. Improving multi-agent scheduling is due to emergent knowledge arising in the process of local interactions of agents.

Keeping order in computation planning and resource allocation within an unspecialized heterogeneous distributed computing environment becomes increasingly complex. Often, some practical situations are complicated. Thus, they simply have no workable deterministic solution that can be obtained in an acceptable time.

At the same time, probabilistic methods provide simple and effective prediction mechanisms using retrospective statistic data and expert judgment for the future. For example, Prieto et al. [44] demonstrate that a probabilistic approach to the decentralized management of a computing network allows them to solve the challenges of the efficient resource usage, scalability, robustness, and adaptability.

In this paper, we represent a new subsystem of the OT framework. This framework is a foundation for an integrated computing environment for packages. It implements the necessary tools for mitigating uncertainty in the job distribution on resources.

3. OT: FOUNDATION FOR INTEGRATED COMPUTING ENVIRONMENT

The architecture of the integrated computing environment is shown in Fig. 1. The OT subsystems represent its basis. External information and computation systems and resources are used in developing and applying distributed applied software packages.

The following main subsystems of are distinguished:

• A user interface that provides a variety of ways for different user categories to access to the package components, subsystems, and information and computation resources,

• Designer of a conceptual model of the environment that implements the knowledge specification about the subject domain of solved problems,

• Designer of module libraries that supports the development and modification of applied software for packages,

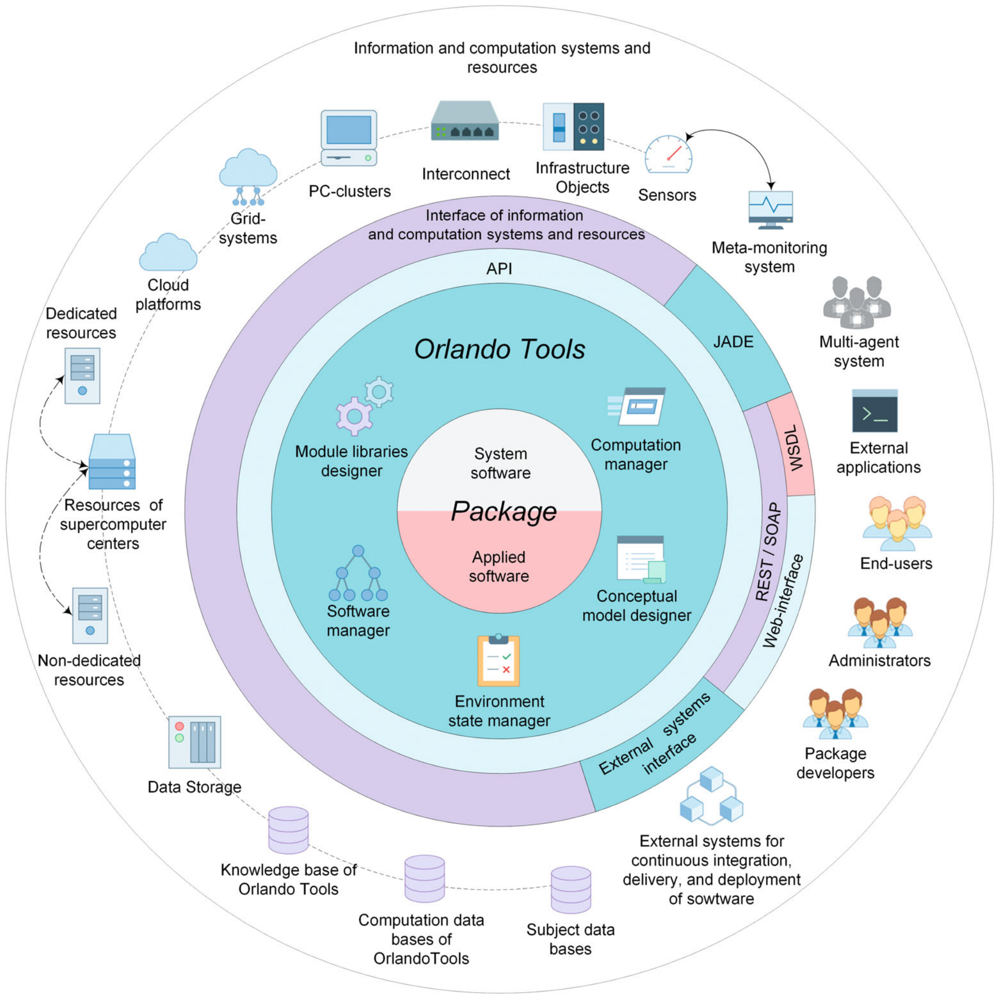

Information and computation systems and resources
PC-clusters
Interconnect
Infrastructure Objects
Sensors
Meta-monitoring system
Grid-systems
Cloud platforms
Dedicated resources
Resources of supercomputer centers
Non-dedicated resources
Data Storage
Knowledge base of Orlando Tools
Computation data bases of OrlandoTools
Subject data bases
External systems for continuous integration, delivery, and deployment of sowtware
Multi-agent system
External applications
End-users
Administrators
Package developers
Interface of information and computation systems and resources
API
JADE
WSDL
REST / SOAP
Web-interface
External systems interface
Orlando Tools
Module libraries designer
Computation manager
Software manager
Conceptual model designer
Environment state manager
System software
Package
Applied software

Fig. 1. The architecture of an integrated computing environment.

• Software manager, which implements automation of the version control, testing, delivery, and deployment of both the applied software of packages and system software of the environment,

• Computation manager that implements computing management on heterogeneous resources of the environment,

• Environment state manager that observes resources,

• API for access to external information and computation systems, and resources.

Knowledge about the subject domains of packages, information about environment resources and computation systems are stored in the corresponding sections of the OT knowledge base. To elicit such knowledge, various databases can be used. The source data and computation results are stored in the computation databases. In the process of computing, additional storages can be used.

3.1. User Interface

Components of the OT subsystems are represented by services. We implemented web-based access for users to the operations. Services that represent components accessed by other software entities (other components, package modules, or external software systems) have an additional description in the Web Services Description Language (WSDL) [45].

3.2. Conceptual Model Designer

A conceptual model of the environment formed by this designer consists of three parts. The first part describes the algorithmic knowledge about solving problems of the subject domain with the package. The second part contains information about infrastructural objects of the environment. Knowledge about the continuous integration, delivery, and deployment of software on the environment resources forms the third part of the model.

Traditionally, algorithmic knowledge is the result of a structural analysis of the subject domain. In OT, its description is formed using a special set of concepts. Among them are such basic concepts as parameters, operations, modules, problem-solving schemes, etc.

Parameters represent the characteristics of the subject domain necessary in the problem-solving process. Operations are defined above a set of parameters. They associate a subset of the input parameters, whose values must be known, to another subset of the output parameters, whose values need to be calculated. Information relations between operations are determined based on their input and output parameters. Productions determine the logical conditions of applying operations in the problem-solving process. These conditions take into account the intermediate results of computations and environment resources state.

Package developers and their end-users create problem-solving schemes on the procedural or nonprocedural problem-formulations. These formulations are based on information relations between operations. The procedural formulation is a list of operations required to solve the problem and partially ordered within a tiered-parallel form of DAG. A nonprocedural formulation is represented in the following form: "Using the initial subset of parameters, whose values are known, calculate the parameter values of the target subset". In the case of such a problem-formulation, the list of operations required to solve the problem and their partial order are determined automatically.

Modules are associated with software implementations of operations. Thus, each problem-solving scheme through its operations determines a set of modules necessary for the problem-solving process implementation.

A module description can include the following information: module source code; programming language; type and purpose of input, output, and transit parameters; methods for passing parameters, handling the non-standard execution, required compiler and its options, call format, etc.

We represent new features of OT for the automation of version control and testing of software. In this regard, the module description is expanded with additional information. Now, the description includes the following additions: links to repositories with source and binary module files, scenarios for building and testing modules, information about test data, and methods of delivery and deployment.

OT supports testing the modules in both individually and as part of problem-solving schemes. The conceptual model constructor of OT provides the development of necessary scripts for external systems. These scripts implement scenarios of continuous integration, delivery, and deployment of software.

Information about infrastructure objects of the environment includes the characteristics of nodes, network devices, network topology, and other facilities. In addition, these characteristics include statistics on software and hardware failures. Information about administrative policies defined for resources is taken into account. It includes data on users and their problems, access rights to resources and quotas for their use, properties of resource management systems, their characteristics, and disciplines of computation dispatching.

Information about the hardware and software infrastructure and administrative policies is elicited using various information and computation systems, controlling and measuring means, and monitoring systems. It is reflected in the conceptual model using system parameters, operations, and modules.

A description of the conceptual model is stored in the OT knowledge base. Sections of the knowledge base are distributed between OT, a meta-monitoring system, and external systems for continuous integration, delivery, and deployment of software.

3.3. Module Libraries Designer

The applied software of packages can be formed on the basis of existing software libraries or created directly by package developers. In both cases, the module description is included in the conceptual model of the environment. OT provides the text editor of the module source code and tools of interaction with external systems for the building, debugging, and testing of modules.

The system software of packages is formed on the basis of the OT system libraries.

3.4. Software Manager

The software manager is a new subsystem of OT. It is designed to identify and partially eliminate potential troubleshooting during the continuous integration, delivery, and deployment of new versions of both the applied and system software of packages. Its main functions include the following operations:

• Interacting with repositories of source and binary files of the applied and system software, including the support of its version control,

• Automation of building and testing the applied and system modules individually,

• Creating the VM images for executing modules,

• Automation of testing the applied modules as part of problem-solving schemes,

• Predicting the runtime of the applied modules in the environment nodes,

• Automation of the delivery and deployment of the applied and system modules in the environment.

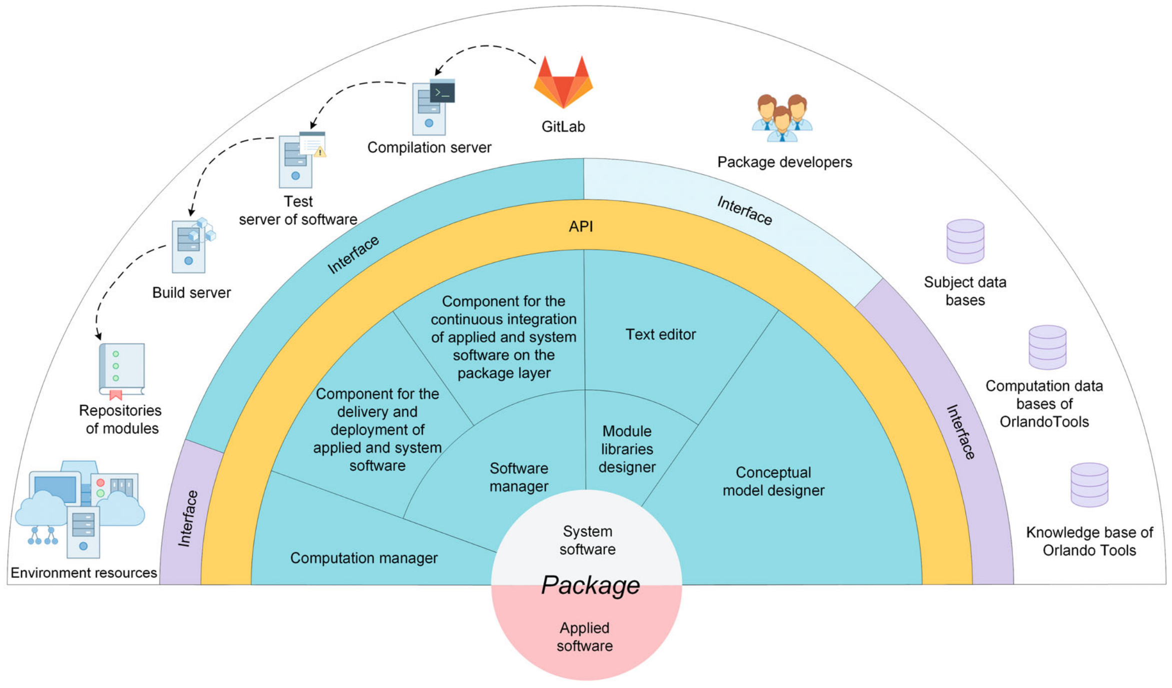


Fig. 2. Scheme of the continuous integration, delivery, and deployment of software.

The general scheme of continuous integration, delivery, and deployment of applied and system software is shown in Figure 2.

New tools for a meta-description of performing all the above-listed operations are first implemented in the conceptual model designer. Information about nodes performance obtained during the testing of modules allows us to determine how much does resource performance matches the module execution criteria.

Such determining is implemented by the environment administrators in advance, before the start of the resources allocation in the process of computational experiments. This ensures significant mitigation of the uncertainty in the subsequent process of the distribution of jobs to resources.

3.5. Computation Manager

A computation manager includes the components for representing problem-formulations and creating problem-solving schemes, generator of jobs for executing the created schemes, scheduler, and interpreter. The scheduler decomposes the initial problem-solving schemes represented in jobs into subschemes in order to optimize the distribution of computing and communication loads on the environment resources.

The interpreter performs the implementation of schemes and subschemes for solving problems.

In addition, the computation manager can transfer jobs for processing to MAS [46]. It is implemented using the Java Agent DEvelopment framework (JADE) [47].

This MAS belongs to the class of meta-schedulers such as GridWay [48] or Condor DAGMan [49]. At the same time, unlike similar tools, it allows us to control the job flows in an environment that integrates the resources of the cloud platforms, grid systems and public access supercomputer centers.

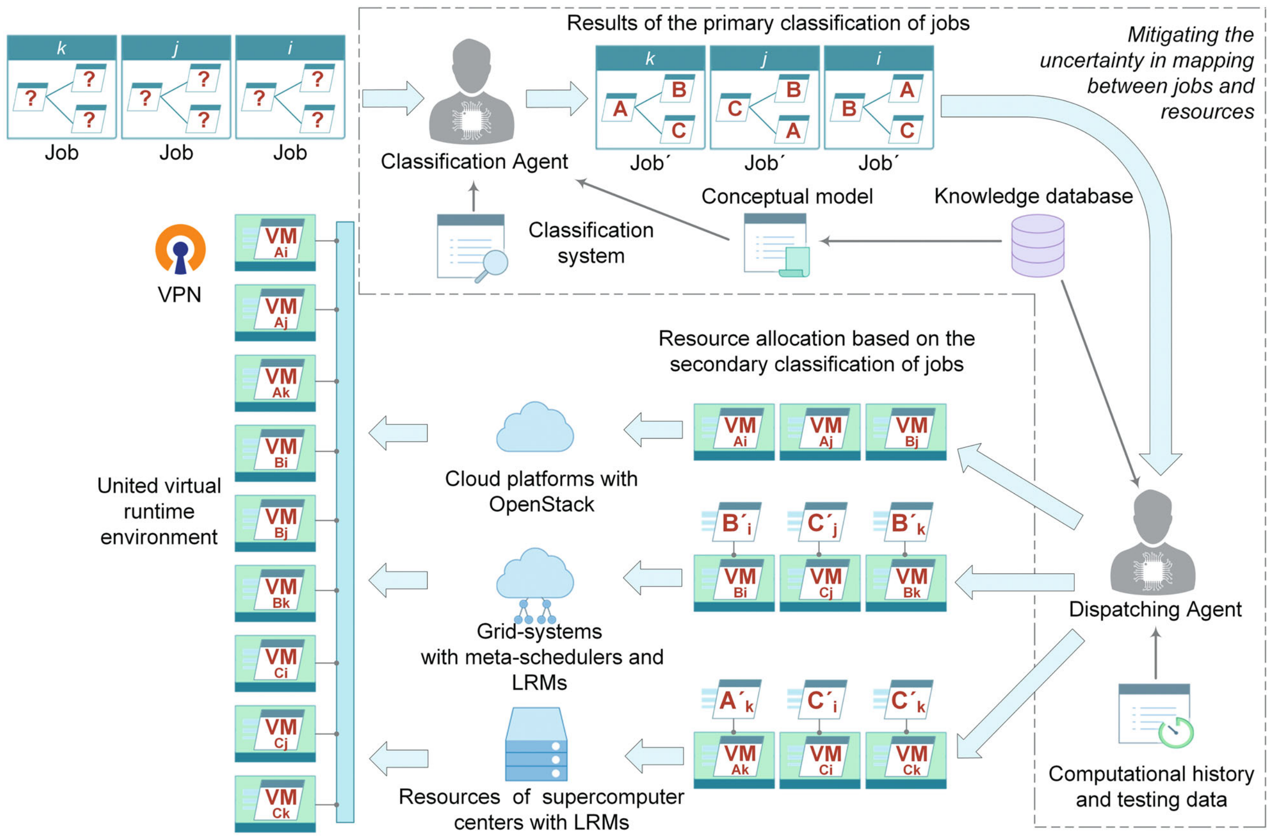
MAS includes agents for classifying jobs, monitoring and dispatching environment resources, as well as agents directly representing integrated resources (resource agents). Resource agents come together in a new virtual community to execute each job.

A condition for including an agent in a community for executing the specific job is the agent has the resources that have the necessary capabilities to execute the modules of this job.

The allocation of resources by agents is shown in Fig. 3. The classification agent determines the class of the job and forms a virtual community of resource agents. The job classification system is discussed in detail in [50]. The classified job is queued to the dispatching agent.

In processing the job queue, the dispatching agent decomposes them into subjobs, which are then distributed among the segments of the integrated computing environment, representing cloud platforms, grid systems, and public access supercomputer centers.

When modules of subjobs are executed, special VMs are launched. Software manager previously creates the images of these VMs. Running VMs are com-



Results of the primary classification of jobs
Mitigating the uncertainty in mapping between jobs and resources
k
j
i
?
Job
Job
Job
Classification Agent
A
B
C
Job´
Job´
Job´
Conceptual model
Knowledge database
Classification system
VPN
VM Ai
VM Aj
VM Ak
VM Bi
VM Bj
VM Bk
VM Ci
VM Cj
VM Ck
United virtual runtime environment
Resource allocation based on the secondary classification of jobs
Cloud platforms with OpenStack
B´i
C´j
B´k
Grid-systems with meta-schedulers and LRMs
Dispatching Agent
A´k
C´i
C´k
Resources of supercomputer centers with LRMs
Computational history and testing data

Fig. 3. Resource allocation by agents.

bined into a united virtual runtime environment for a package.

Cloud resources are managed using the OpenStack platform. To start VMs on the resources of grid-systems and public access supercomputer centres, new jobs are formed for Local Resource Managers (LRMs) installed on these resources. New jobs are sent to the common queues of LRMs. To speed up the process of executing jobs, nodes with free slots are searched in their job service queues during the resources allocation. Each slot determines the number of cores that are used or can be used to execute a job over a period of time. Overall, agents provide mitigation of uncertainty in the job distribution by resources. Such mitigation is carried out automatically in four stages.

At the first stage, the primary classification of a job is implemented by analyzing the job specification using a special characteristic function. The formal correspondence of job characteristics and their values to the characteristics of the predefined classes and their value domains is checked. In consequence of the primary classification, a job may belong to several classes.

The selection of resources corresponding to the job classes is performed in the second stage.

In the third stage, the secondary classification of a job is performed based on the analysis of statistical information about the computational history of job executions and their correspondence to the predefined classes. The analysis results are used to adjust the outcomes of the primary classification.

At the last stage, the results of the secondary classification are refined using the previously obtained data in testing applied software by the software manager. For this purpose, we can also use information about the predicted runtime of modules.

The agent for classifying jobs performs the first two stages. The agent for dispatching environment resources carries out the last two stages. Preliminarily, administrators define the necessary job classes in the classification system. Then they establish the correspondence of all their resources to the defined classes. Administrators are made the decisions based on their practical experience.

The final lob distribution by resources is performed by resource agents using a tender of computational works (jobs) [51]. Within the tender, bidding is based on the closed second-price Vickrey auction [52]. It is held in one round. During the bidding, additional conditions for job execution are taken into account. These include the preferences of resource owners and the criteria for the problem-solving efficiency, determined by users of the environment.

3.6. Environment State Manager

This manager receives information about the environment state from a special meta-monitoring system implemented by applying a multi-agent approach [46]. The meta-monitoring system collects information about the state characteristics of environment infrastructure objects using the controlling and measuring means (Sensors, APC PowerChute, IPMI, SMART, etc.) and local monitoring systems (Ganglia, Nagios, Zabbix, ZenOSS, etc.). In addition, it observes the performance parameters of control systems that operate on these objects.

Based on the information collected, the meta-monitoring system assesses the current computing situation, predicts its development, and generates control actions on infrastructure objects and control systems in order to prevent or partially eliminate failures of software and hardware. In case of a critical situation, when such actions cannot be performed automatically, the meta-monitoring system sends a notification to the environment administrator.

3.7. Access to Information and Computation Systems and Resources

OT provides web-oriented access for users to information and computations systems and resources of the integrated computing environment. This framework interacts with the systems and resources using the protocols supported by them.

4. MITIGATING UNCERTAINTY

We consider uncertainty in terms of a deficiency of the available knowledge in relation to the knowledge needed to make the near-optimal decision. In our case, such necessary knowledge is information about the module runtime in heterogeneous nodes of the environment, job classification in accordance with their characteristics, and correspondence of resource capabilities to job classes.

4.1. Evaluating the Module Runtime

We developed a special model for evaluating the runtime of modules in the problem-solving scheme. In the process of testing, the module runs on the reference node from the software environment for profiling programs. We execute the module and determine its time spent on working with various components of the node taking into account its computational characteristics. The module runtime on the target node is predicted with some error by comparing the characteristics of the reference and target nodes.

Within the developed model, the sets $CR = \{cr_1, cr_2, \ldots, cr_m\}$ and $CT = \{ct_1, ct_2, \ldots, ct_m\}$ of characteristics of the reference and target nodes are formed. The «one-to-one» correspondence is set between cr_i and ct_i, $i = \overline{1,m}$.

We form the set $P = \{p_1(d), p_2(d), \ldots, p_n(d)\}$. The element $p_j(d)$, $j = \overline{1,n}$ reflects the computational load of a node when the module is run. Among them are the number of integer operations performed, the number of floating point operations, the number of accesses to main memory and cache memory of different levels, the number of misses of such accesses, the number of disks read and write sessions, etc. The load depends on the data d processed by this module.

Then, the functions $f_l(CR, P, g(d))$ are defined. They calculate the work-time of the lth component of a node when it performs the computational load, $l = \overline{1,k}$. The parameter $g(d)$ is an interpolation function that determines the dependence of $\hat{T}_r(d)$ from the amount of data d experimentally.

The module runtime on the reference node is evaluated by the following expression:

$$\hat{T}_r(d) = \sum_{l=1}^{k} f_l(CR, P, g(d)),$$

where $g(d)$ is a manually fitted interpolation function that determines the dependence of $\hat{T}_r(d)$ from the amount of data d experimentally. The error ε of this evaluation is determined from the difference in $T_r(d)$ and $\hat{T}_r(d)$: $\varepsilon = T_r(d) - \hat{T}_r(d)$. A theoretical evaluation of the upper and lower bounds of the ε error is not given since in each particular case it depend on the implementation of the functions $f_l(CR, P, g(d))$.

The time $\hat{T}_t'(x)$ of the module run in the target node is evaluated by the following expression:

$$\hat{T}_t'(x) = \hat{T}_t(d) + \varepsilon \frac{\hat{T}_r(d)}{\hat{T}_t(d)}, s$$

$$\hat{T}_t(d) = \sum_{l=1}^{k} f_l(CT, P, g(d)).$$

To evaluate the module runtime $\hat{T}_t'(x)$ in the target node, we propose the algorithm that includes the following steps:

• Defining the reference and target nodes,

• Determining the characteristics $cr_1, cr_2, \ldots, cr_m$ of the reference node based on the knowledge about resources from the conceptual model of the environment,

• Determining the characteristics $ct_1, ct_2, \ldots, ct_m$ of the target node based on the knowledge about resources from the conceptual model of the environment,

• Preparing the data d in the computation database of the OT,

• Deployment the profiler, module, and prepared data on the reference node using the software manager,

• Starting the profiler on the reference node,

• Launching the module from the profiler environment,

• Executing the module on the reference node,

• Getting the values of profiler counters,

• Determining the real module runtime $T_r(d)$ using the obtained values,

• Calculating $p_1(d), p_2(d), \dots, p_n(d)$ using the obtained values,

• Evaluating the module runtime $\hat{T}_r(d)$ on the reference node,

• Evaluating the module runtime $\hat{T}_t(d)$ on the target node,

• Calculating the error ε,

• Evaluating the module runtime $\hat{T}_t'(x)$ on the target node.

In the algorithm, the characteristics $cr_1, cr_2, \dots, cr_m$, $ct_1, ct_2, \dots, ct_m$, components of d, the module runtime T_r, evaluation $\hat{T}_r$, evaluation $\hat{T}_t'$, and error ε are the system parameters. The functions $p_1, p_2, \dots, f_1, f_2, \dots, f_k$, g, and implementations of the expressions for calculating $\hat{T}_r$, $\hat{T}_t'$, and ε are the system modules. The system parameters and modules are described in the conceptual model of the environment.

The evaluation of the module runtime obtained with the help of the proposed algorithm is rather approximate. In addition, the automatic selection of the interpolation function $g(d)$ is difficult to implement for some classes of modules. However, such an evaluation in some cases can significantly improve end-user evaluations and evaluations based on computational history.

4.2. Mitigating Uncertainty in the Job Distribution by Resources

To evaluate the degree of uncertainty mitigation in the job distribution by resources, we apply an informational binary entropy. This parameter is often used to measure the uncertainty of a complex system. It is assumed that a decrease in entropy leads to mitigation of uncertainty.

Let m classes be found from the results of the primary classification of a job that is satisfied with the characteristics of these classes. The environment includes n heterogeneous resources. One class can correspond to k resources, $k \in \overline{1,n}$. Information about the correspondence between classes and resources is represented by a Boolean matrix Y of dimension $n \times m$. The matrix element $y_{ji} = 1$ ($y_{ji} = 0$) shows that the jth resource corresponds (not corresponds) to the ith class.

We define the entropy Ec, which reflects the uncertainty degree in belonging the job to one of the classes, as follows:

$$E^c(x_c) = \sum_{i=1}^{m} p_i(x_c) E_i^c(x_c),$$

where xc is a random event that determines that the job belongs to one of m classes, $p_i(x_c)$ is the probability of such an event relative to the ith class, $\sum_{i=1}^{m} p_i(x_c) = 1$, $E_i^c(x_c)$ is the entropy corresponding to this event. The value of $E_i^c(x_c)$ is calculated as follows:

$$E_i^c(x_c) = \begin{cases} 0, & \text{if} \quad p_i(x_c) = 0, \\ -\log_2 p_i(x_c) & \text{otherwise.} \end{cases}$$

In the same way, we define the entropy E^r, which reflects the uncertainty degree in belonging the job to one of the classes. It will take the following form:

$$E^r(x_r) = \sum_{j=1}^{n} p_j(x_r) E_j^r(x_r),$$

where x_r is a random event that leads to allocate one of n resources to the job, $p_j(x_r)$ is the probability of such an event relative to the jth resource, $E^r(x_r)$ is the entropy corresponding to this event. The values $p_j(x_r)$ and $E_j^r(x_r)$ are calculated as follows:

$$p_j(x_r) = \sum_{\forall l: y_{jl}=1} p_l(x_C) \Big/ \sum_{j=1}^{n} \sum_{\forall l: y_{jl}=1} p_l(x_C),$$

$$l \in \overline{1,m}, \quad \sum_{j=1}^{n} p_j(x_r) = 1,$$

$$E_j^r(x_r) = \begin{cases} 0, & \text{if} \quad p_j(x_r) = 0, \\ -\log_2 p_j(x_r) & \text{otherwise.} \end{cases}$$

Let us to consider an illustrative example of mitigating uncertainty in the job distribution by resources.

Suppose that a job was sent into the environment to execute a module specified in this job. According to the results of the primary classification, 8 classes were found whose characteristics are satisfied by the job. These classes differ among themselves in the value domain of one characteristic that determines the admissible job makespan. The value of this characteristic is not specified in the job specification.

Table 1. Event probabilities and entropy parameters

No.	$p_1(x_c)$	$p_2(x_c)$	$p_3(x_c)$	$p_4(x_c)$	$p_5(x_c)$	$p_6(x_c)$	$p_7(x_c)$	$p_8(x_c)$	E^c	$p_1(x_r)$	$p_2(x_r)$	$p_3(x_r)$	E^r
1	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	3.00	0.333	0.333	0.333	1.06
2	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	3.00	0.444	0.556	0	0.99
3	0.115	0.741	0.132	0.012	0	0	0	0	1.14	0.988	0.012	0	0.09
4	0	0.936	0.064	0	0	0	0	0	0.34	1.000	0	0	0

The environment includes 3 heterogeneous resources (3 clusters, the nodes of which differ in their computational characteristics). The correspondence of resources to classes is represented by a matrix

$$Y = \begin{bmatrix} 1\,1\,1\,1\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0 \end{bmatrix}.$$

There is a computational history of performing such a job on the environment resources.

Table 1 shows the number of stages, values $p_1(x_c)$, $p_2(x_c)$, ..., $p_8(x_c)$, Ec, $p_1(x_r)$, ..., $p_3(x_c)$, and Ec for each stage of mitigating uncertainty in the job distribution by resources.

At the first stage, the probabilities of belonging the to one of 8 classes are equal to each other, just as the probabilities of allocating one of 3 resources to the job are equal. In this case, $E^c = 3.00$ and $E^r = 1.06$.

At the next stage, two resources were selected that are most suitable for the job execution from the point of view of expert experience of resource administrators. In this case, the uncertainty degree in allocating resources to the job decreases to $E^r = 0.99$.

The secondary classification of the job based on the analysis of statistical information about the computational history concretizes the probabilities of belonging the job to classes obtained at the third stage. The value of Ec becomes 1.14. In accordance with this fact, the probabilities of allocating resources to the job change. The uncertainty degree in allocating resources to the job is significantly reduced to $E^r = 0.09$.

At the last stage, the results of the secondary classification are refined based on the predicted module runtime or previously obtained data of its testing. As a result, $E^c = 0.34$ and $E^r = 0.09$. Thus, resource 1 is allocated to execute the job.

It should be noted that according to the results of the secondary classification of jobs, they could belong to wider sets of classes in comparison with the refined classification based on module testing. This is because the computational history may contain information about the execution of modules in debugging and modules executed using incorrect data. In this case, some job characteristics, for example, their makespan, may go beyond the boundaries of value domains in classes that really correspond to these jobs.

5. EXPERIMENTAL ANALYSIS

5.1. Package for Problem-solving the Search for Directions of Sustainable Development of the Energy Sector

The search for directions of sustainable development of both the energy sector as whole and individual energy systems is a typical problem. The need to solve it repeatedly arises in the process of their operation and development. To solve this problem, the Corrective package was developed. It has been used to study the sustainable development of the energy sector of Vietnam [53].

We study the energy sector development taking into account all combinations of large-scale disturbances that can lead to insufficient provision of end-consumers in energy resources. Such a study causes a large number of possible states of the energy sector and difficulties in analyzing these states by the classical methods.

In this regard, in the package, we use a new approach to the study of the energy sector through combinatorial modeling. It is based on considering various combinations of possible energy sector states and transitions between them for the given period T. Since the entire set of possible states is very large, a set of admissible states is only examined.

We represent a set of states as directed graph $G=<V,E>$_x005F_x0001_, where V_x005F_x-0001_ is a set of vertices (possible states of the energy sector), and E_x005F_x0001_ is a set of edges (transitions between states). Edges are directed from the state s_0 at the initial time τ_0 towards states _x005F_x0001_at the next time moment τ_1 and further towards states at the next time moment $\tau_2, \tau_3, \ldots$ (Figure 4). Each vertex reflects a state of interconnected objects of the energy sector. Such state includes the operation, reconstruction, modernization, and creation of objects. Each object performs one or more technological work (storing, production, processing, transformation, and transportation of energy resources).

The combinatorial analysis is carried out by examining various combinations of object states and transitions between states for the period T. The objective is

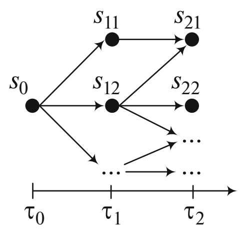


Fig. 4. Fragment of the graph G.

to search a subgraph of the graph G that includes the paths with a minimum cost of energy sector development.

The general scheme for problem-solving is shown in Figure 5.

The operation o_5 prepares these scenarios for the development of the energy sector of Vietnam. It forms the criteria for evaluating the energy sector functioning, list of actions for its development, set of organizational and functional constraints, simulated period, and other initial parameters.

Then, based on the prepared data, the operation o_1 selects a scenario and constructs a variant of the appropriate sector development model. The operation o_2 decomposes the model into submodels taking into account various logical and balance constraints. The submodels are examined in parallel using instances of the operation o_3.

In each case, we solve the following linear programming problem:

$$f(S) = CX + P(R - Y) \to \min,$$
$$S = A, X, Y, C, P, D, R, U, W >,$$
$$AX - Y \geq 0, \quad X \leq D, \quad Y \leq R,$$

where the problem parameters are interpreted as follows:

• S is the generalized state of the energy sector,

• C is the vector of specific costs for the technological methods of functioning the existing, reconstructed, modernized, and newly constructed energy sector objects,

• A is the matrix of technological coefficients of production, processing, transformation, and transportation of resources, $a_{ij} = q_{ij}(U, W)$,

• D is a vector that determines the technically possible intensities of applying technological methods of objects functioning (meaning production and transport of resources), $d_i = h_i(U)$,

• R is a vector that defines the required volume of the resource consumption,

• P is the vector of specific losses due to lack of resources,

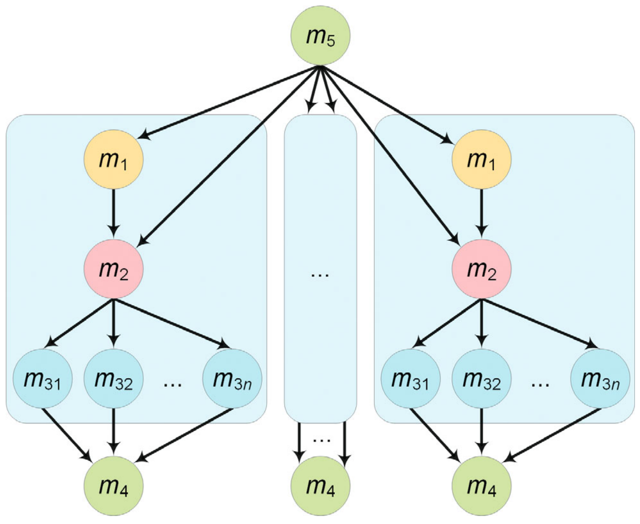


Fig. 5. Problem-solving scheme.

• U is the vector of actions for the energy sector development,

• W is the vector of disturbances.

The first term CX in the objective function shows the costs associated with the energy sector functioning. At the same time, the second term $P(R - Y)$ reflects the damage from insufficient resources supply to consumers due to disturbances. Such disturbances can include large-scale technological accidents, natural disasters, and terrorist acts.

The operation o_4 integrates the simulation results, performs their multicriteria analysis, and generates a set of rational ways of the energy sector development for each scenario.

Modules $m_1 - m_5$ implement operations $o_1 - o_5$. Their frequent modification is due to the evolution of algorithmic knowledge of the package. In particular, it is caused by the applying new criteria for evaluating the energy sector functioning, expanding the list of actions for its development, and using additional constraints on the processes of functioning and development of this sector. In addition, the development of hardware and software of the environment often leads to the need for the recompilation, build, and testing of modules with these updates.

In this regard, we consider two basic versions v_1 [12] and v_2 [13] for solving the problem of searching directions of the sustainable development of the energy sector of Vietnam. The second version is developed using the new software manager developed for OT.

5.2. Computational Resources

Problem-solving using both of the aforementioned versions of the package was performed in the integrated computing environment that includes the following resources:

• Cloud-based virtualized resources under the OpenStack management – 5 dedicated nodes (2x16 cores CPU AMD Opteron 6276, 2.3 GHz, 64 GB RAM) of the HPC-cluster of the public access computer center Irkutsk Supercomputer Center of SB RAS [54],

• Non-dedicated nodes (2 × 18 cores CPU Intel Xeon X5670, 2.1 GHz, 128 GB RAM) of the HPC-cluster,

• Grid-resources – remote personal computers.

The dedicated resources are intended only for solving the problem under consideration. Non-dedicated resources are shared by all users of the public access computer center. They are used in accordance with their administrative policies and quotas. Users who have already been dedicated resources can use only those non-dedicated resources on which there are free slots in the LRM schedule. In the process of solving the problem in an integrated computing environment, its grid-resources can be used by their owners.

5.3. Preparation of a Computational Experiment

In general, software modification includes the following stages:

• Modification of the source code of applied and system software, including searching for the correct version in the repositories on the GitLab server, cloning the necessary repository to the OT server or developer server, launching the programming environment, editing the code, and saving the modified code to the GitLab server,

• Automated build of software in accordance with the scenarios predefined in the conceptual model designer for external continuous integration systems,

• Modification of test data including its search on the GitLab server or testing server, launching the data processing system, and editing them,

• Modification of scenarios of continuous integration, delivery, and deployment of software using the conceptual model designer,

• Automated software testing in accordance with the predefined scenarios,

• Automated build of software installation packages in accordance with scenarios of its continuous integration, delivery, and deployment,

• Modification of the conceptual model (parameters, operations, modules, productions, problem-solving schemes, and relationships between them) of the environment, translation of the resulting modification into an XML-description, and verification of its completeness and correctness,

• Automatic deployment of software on resources using installation packages,

• Automated testing of software on resources in accordance with the predefined scenarios including an evaluation of the resource performance during execution,

• Automated testing of package schemes in accordance with the predefined scenarios.

Usually, these stages are repeated many times (up to several times a day) in the process of modifying applied and system software. When developing version v_1, all of them were executed in non-automated mode.

Evaluations of the average time taken to modify the software of the package are represented in Fig. 6. They include evaluations of overall time costs (Fig. 6a,c) and the developer time costs (Fig. 6b,d) at the modification stages for a one module using the software manager (version v_2) and without it (version v_1).

Figure 6a and Figure 6c reflect the time spent on the stages associated with the modification of the algorithmic knowledge of the package. Their implementation is based on the joint use of OT and external continuous integration systems. The remaining evaluations related to modifying the package only with OT are shown in Fig. 6b,d.

The results represented in Fig, 6 show that the use of the software manager provides a significant decrease in both the time spent by a software developer and overall time taken to prepare the experiment. They are due to the following factors:

• Reducing the time spent to search the software source code in repositories,

• Automation of compilation, testing, and building installation packages, deployment and testing of software on resources, and testing problem-solving schemes,

• Reducing compilation time, testing, and building installation packages by moving these stages to more performance resources,

• Improving the computation reliability in the problem-solving process by preliminary testing software on heterogeneous resources,

• Partial automation of the process of modifying the conceptual model of the environment when changing software specifications in repositories.

Figure 7 shows evaluations of the time spent by the software developer and the overall time taken to prepare and carrying out experiments for version v_2 using and without the software manager. The evaluations are calculated on the basis of a cumulative total and shown in minutes in accordance with the software modification sequence. Obviously, the use of the software manager provides a significant decrease in time of preparing and carrying out of experiments.

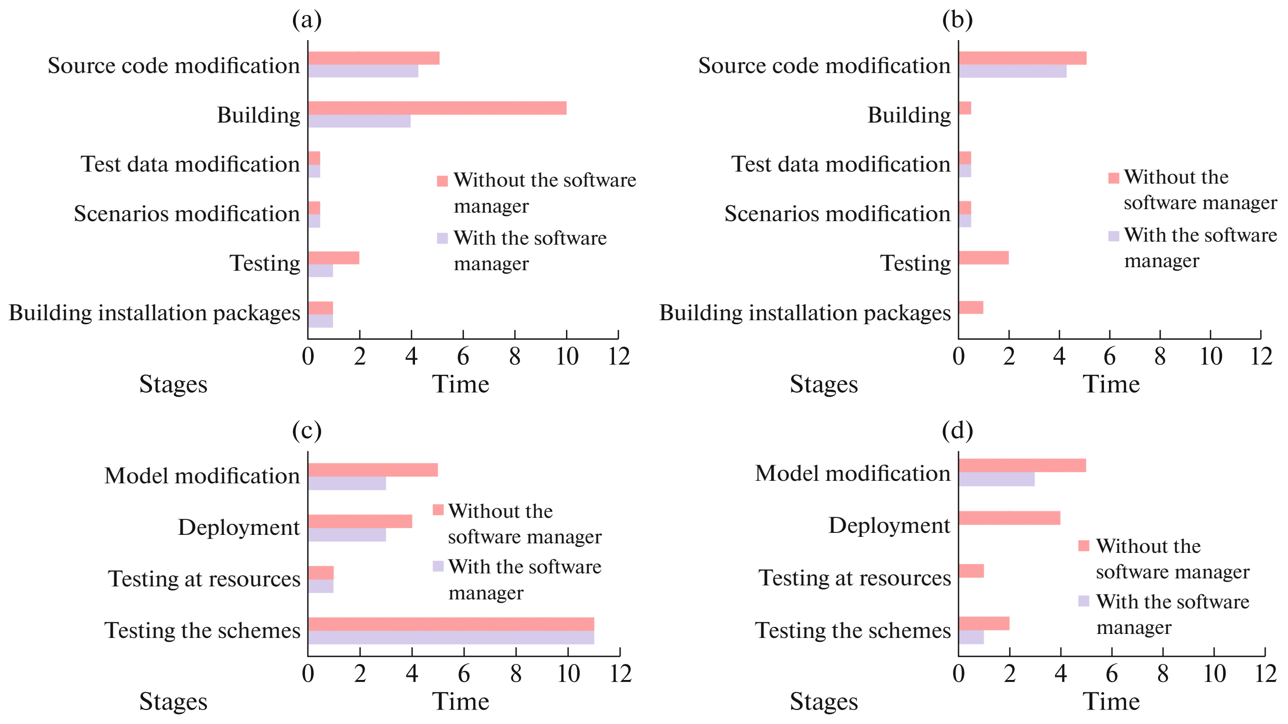


Fig. 6. Evaluations of the overall time (a, c) and developer time (b, d) in minutes at the modification stages for a one module using and without the software manager respectively.

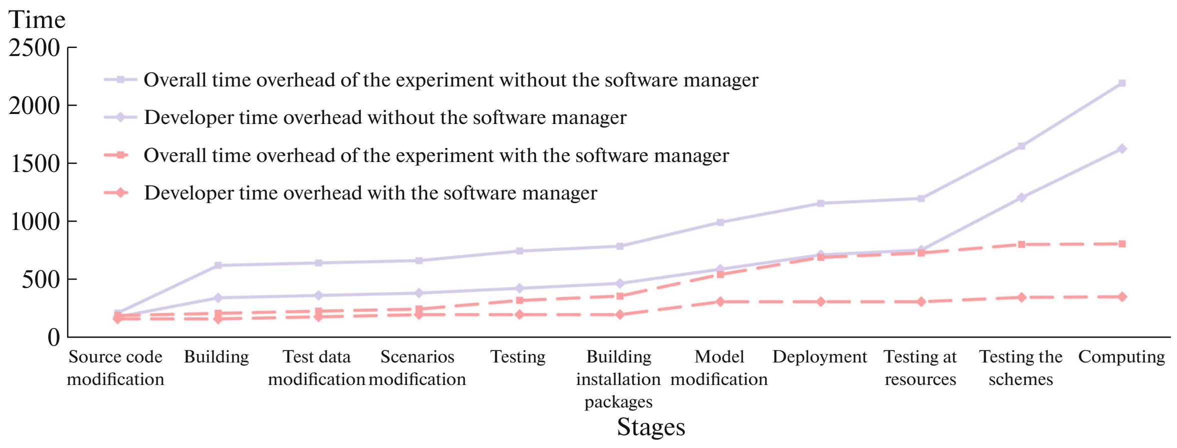


Fig. 7. Cumulative time evaluations.

5.4. Computational Experiment

We compare the module execution parameters of the two versions v_1 and v_2 when solving typical problems characterized by the same computational complexity. In the process of problem-solving, data of the same structure were used. These data are obtained on the basis of statistical information on the functioning of the energy sector of Vietnam in different periods.

In Table 2, we show the following problem-solving parameters for one scenario of the energy sector development:

- n_s is the number of module launches,
- t_{avg} is the average module runtime in seconds,
- n_f is the number of module failures,
- n_r is the number of module restarts,
- k is the coefficient of utilization of the node components at the module execution.

The time of parallel processing of submodels for the energy sector development by instances of module m_3 is the main part of the problem-solving makespan.

Table 2. Module execution parameters

Module	n_s		t_{avg}			n_f		N_r		k
	v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2
m_1	1	1	10.22	10.17	0	0	0	0	90.01	90.70
m_2	1	1	172.64	166.42	0	0	0	0	94.35	94.74
m_3	531442	531497	10.17	9.74	1371	0	1678	0	88.27	98.46
m_4	1	1	130.04	129.23	0	0	0	0	30.00	30.02
m_5	1	1	11.62	10.55	0	0	0	0	27.27	30.63

The results are represented in Table 2 show that the number of failures for the module m_3 instances of version v_2 has decreased. Accordingly, the number of restarts has reduced.

The optimization of the module m_3 operation of the version v_2 and elimination of failures in its functioning led to a significant improvement in the coefficient k for this module by 10.19%. The resource utilization coefficients for the execution of the remaining modules of version v_2 are also improved.

Figure 8 shows the time t_2 and t_3 of computations in minutes when using modules of versions v_1 and v_2 to solve the problem. In both cases, dedicated and non-dedicated resources were used jointly. The jobs distribution was carried out under the MAS management.

In addition, the evaluated time t_1 for problem-solving only on dedicated resources when using modules of version v_1 is given. In this case, the distribution of jobs was carried out under the OpenStack management.

In the case of version v_1, the use of additional non-dedicated resources made it possible to reduce the time it took to solve the problem by 18% in comparison with the evaluated time of problem-solving on non-dedicated resources only. The use of version v_2 reduced this time by an additional 5%.

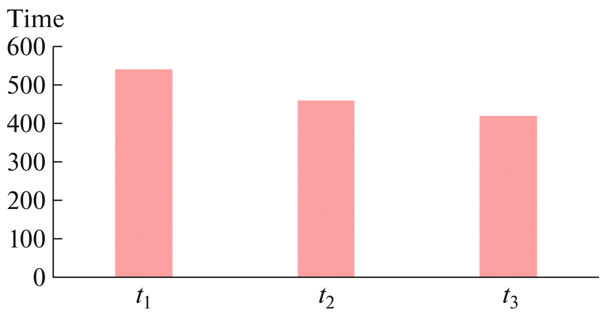


Fig. 8. Problem-solving makespan.

5.5. The Non-dedicated Resource Use

In this section, we consider the advantages of multi-agent management of distributed computing in the joint applying dedicated and non-dedicated resources.

The module m_3 evaluates states every 10 seconds on average. Figure 9 shows the distribution density of the module m_3 runtime for both package versions respectively. In both cases, these results show that the module m_3 runtime in most cases is about from 5 to 10 seconds. This fact enables to provide a good balance of the resources load by agents.

The estimated problem-solving makespan on one core is more than 60 days. The makespan provided by applying only virtualized resources with 160 VMs is about 9 hours for both package versions. This is achieved by starting VMs in non-dedicated resources. A part of the computational load is re-allocated to non-dedicated nodes.

Figure 10 demonstrates numbers of free and used slots for every hour in the problem-solving for both versions of packages.

The numbers of free slots in the LRM schedule of non-dedicated resources are represented by the green color. The numbers of used slots are reflected by the light green color.

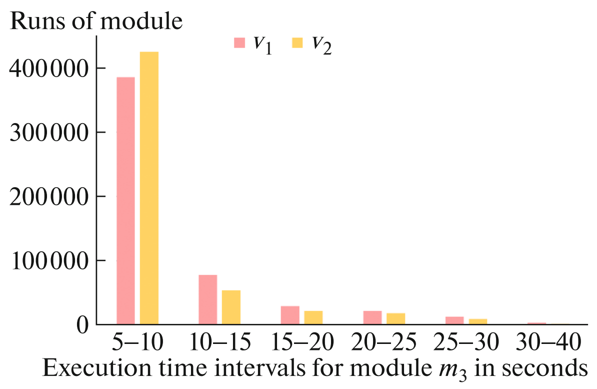


Fig. 9. The distribution density of the execution time.

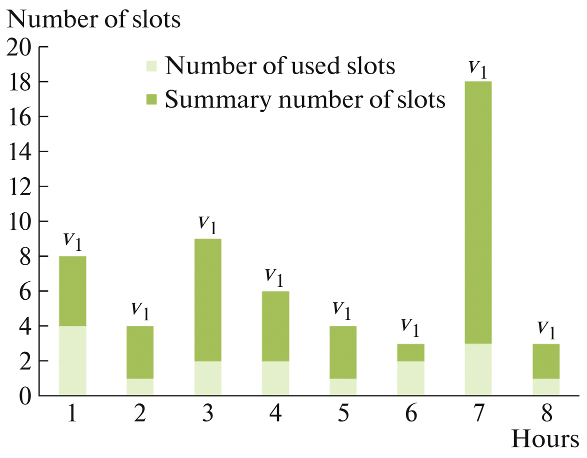


Fig. 10. A number of slots vs. an hour.

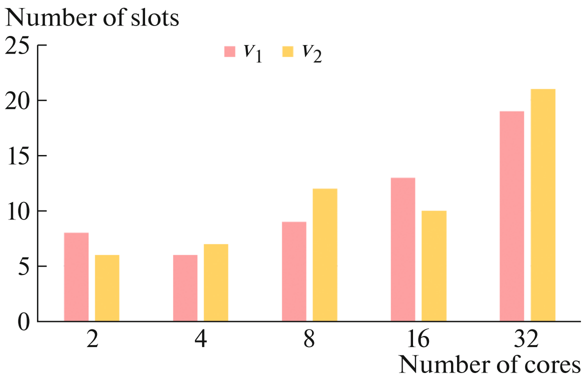


Fig. 11. The distribution density of a number of processors in slots.

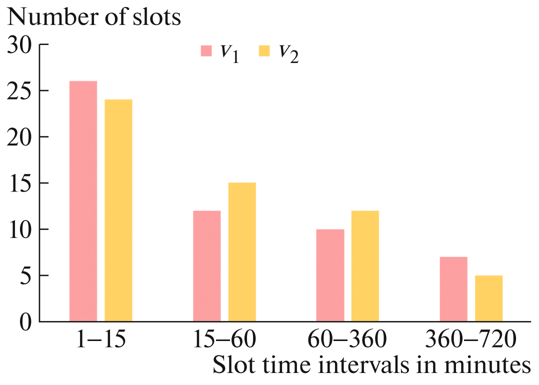


Fig. 12. The distribution density of slot time intervals.

Distribution densities of the number of cores are demonstrated in Fig. 11. We can see that most of the slots have 32 cores.

Time intervals for the used slots are represented in Fig. 12. It shows the insufficient time intervals in many slots for the effective launch of VMs. At the histograms, we use the exponential fit.

We used 16 slots from 55 free slots for v_1 and 17 slots from 56 free slots for v_2. For $v_1(v_2)$, 15 (18) free slots occurred in the nodes in which jobs are already executed. The VMs startup could negatively affect the execution time. 24 (21) slots exist a short time. The efficiency of running machines in these slots would be extremely low. In the best case, we use three slots at a time and the maximum number of additional cores is equal to 96 in each scenario. At the same time, the average processor utilization rate in non-dedicated resources for both cases is equal to about 90%.

For jobs of the module m_3 execution, we have created the special class. It includes characteristics for jobs that can be executed both in dedicated and non-dedicated resources. The information about these characteristics is represented in Table 3.

It shows the following data: job characteristic, its types, domain, unit, attributes "mandatory" or "optional", specialized domain for the class, and example of characteristic values in the job specification. In this specification, characteristics 1-3 represent information about the executed file.

Characteristics 4-6 describe requested resources. A selection of the right VM image is provided by characteristics 7 and 8. Middleware in VM is determined by the system variable "framework". This variable "framework" was equal to "DISCOMP" for v_1. Subsequently, it was equal to "OT" for v_2. Characteristics 9 and 10 determine restrictions of slots in LRM schedule. Let us note that characteristics 11-13 are optional. Therefore, their values may not be defined in a job specification.

The job classification determines restrictions of slots in the job class specification, in advance. Further, using this knowledge, agents automatically select slots that ensure efficient use of resources.

For dedicated virtualized nodes, we speed up computations close to linear. In the case of v_1 job re-allocation to non-dedicated resources reduces the problem-solving time by about 18% in comparison with OpenStack platform that cannot provide dynamic elasticity of VM provisioning in non-dedicated resources. At the same time, in the case of v_2 these indicators were improved by 5% in comparison to v_1.

6. PRACTICAL USE

OT is applied in Irkutsk Supercomputer Center of SB RAS. Usually, it is used for supporting large-scale computational experiments within initiative scientific research.

Table 3. Job Class Characteristics

No.	Characteristic	Type	Domain	Unit	Mandatory/ Optional	Specialized domain	Job specification
1	Job name	String	1-255	Char	Mandatory	1-255	Corrective_solver
2	Path to executed file	String	1-255	Char	Mandatory	1-255	~/ framework /corrective/failsets
3	Parameters of executable file	String	1-65535	Char	Mandatory	1-65535	-input_file input.xml -output_file output.xml
4	Number of requested nodes	Integer	1-20	Node	Mandatory	1	1
5	Number of requested cores per node	Integer	1-36	Core	Mandatory	32	32
6	Number of VMs that will be launched in a one node	Integer	1-36	VM	Mandatory	32	32
7	Guest operating system in VM	String	1-256	Char	Mandatory	ReactOS	ReactOS
8	Middleware in VM	String	1-256	Char	Mandatory	framework	framework
9	Number of free cores in a slot of LRM Schedule in non-dedicated resources	Integer	1-36	Core	Mandatory	32	32
10	Minimum time interval for a slot of LRM Schedule in non-dedicated resources	Integer	1-1728000	Second	Mandatory	900-1728000	900
11	Maximum job execution time	Integer	1-1728000	Second	Optional	1-1728000	None
12	RAM size	Integer	8-128	GB	Optional	8	None
13	HDD size	Integer	1-2048	GB	Optional	1	None

At the same time, OT has been also successfully used to solve many practical problems. Among them are the following important problems:

• Studying the development scenarios of the energy sector of Vietnam [13, 53],

• Analysis of the vulnerability of critical infrastructures in the energy sector of Russia [9],

• Optimization of warehouse logistics [55],

• A search of global minima of multiextremal functions [56].

In addition, students of the Irkutsk State University actively use OT within training course Parallel Computing. Students receive both the group training in solving typical problems using traditional parallel programming technologies and individual training in solving specialized problems using the OT technologies.

The community of the OT users includes the following groups:

• A team of scientists developing and supporting the framework,

• Developers, administrators, and end-users of distributed applied software packages that were developed for solving the aforementioned practical problems,

• Students.

7. CONCLUSIONS

OT provides unified environment that integrates clouds, grids, and public access supercomputer centers. It includes tools for development and use of distributed applied software packages using modern methods and means within the paradigms of parallel, distributed, and virtual computing.

In this paper, we present new tools of the framework for mitigating uncertainty of the job allocation on resources. Our contribution is multifold.

We define uncertainty in terms of a deficiency of the available knowledge in relation to the knowledge needed to make the near-optimal decision of the job distributing.

We provide eliciting the additional necessary knowledge about the module runtime estimation in different nodes of a heterogeneous integrated computing environment.

We propose methods for job classification in accordance with their characteristics and matching resource capabilities to job classes.

We extend the systems for continuous integration, delivery, and deployment of software for scalable scientific applications (distributed applied software packages) with workflows.

However, some problems cannot be solved with the help of such systems. To this end, we have implemented our own tools for the meta-description of all processes of continuous integration, delivery, and deployment of both the applied and system software. In comparison with existed systems, these tools support modification the conceptual model of the integrated computing environment and automation of planning and testing problem-solving schemes.

We have provided the results of a comprehensive comparative analysis of two modifications of the distributed applied software package. The versions of this package were developed for solving the problem of studying directions of the sustainable development of the energy sector of Vietnam with the use of the developed tools and without them. These results show the benefits of using the developed tools. In particular, the reliability of distributed computing and problem-solving time are significantly improved.

Our further studies are related to the use of OT in creating digital twins of important infrastructure objects of the Baikal natural territory. The aim of the study is to optimize the applying advanced environmentally friendly technologies at the objects.

OT is based on the technology of creating applied software packages, which has been developing at the institute since the 80s of the last century. It is promoted by the Matrosov Institute for System Dynamics and Control Theory of SB RAS.

ACKNOWLEDGMENTS

The study is supported by the Russian Foundation of Basic Research, projects nos. 19-07-00097 and 18-07-01224. The development of meta-monitoring and resource allocation agents was supported in part by the Basic Research Program of SB RAS, project no. IV.38.1.1.

REFERENCES

1. Inggs, G., Thomas, D.B., and Luk, W., A domain specific approach to high performance heterogeneous computing, *IEEE Trans. Parallel Distrib. Syst.,* 2017, vol. 28, no. 1, pp. 2–15.
2. Il'in, V., Artificial intelligence problems in mathematical modeling, *Commun. Comput. Inf. Sci.,* 2019, vol. 1129, pp. 505–516.
3. Seinstra, F.J., Maassen, J., van Nieuwpoort, R.V., Drost, N., van Kessel, T., and van Werkhoven, B., Jungle computing: distributed supercomputing beyond clusters, grids, and clouds, in *Grids, Clouds and Virtualization. Computer Communications and Networks,* London: Springer, 2011, pp. 167–197.
4. Wang, L., Jie, W., and Chen, J., *Grid Computing: Infrastructure, Service, and Applications,* CRC Press, 2018.
5. Varshney, S., Sandhu, R., and Gupta, P.K., QoS based resource provisioning in cloud computing environment: a technical survey, in *Proc. Int. Conf. on Advances in Computing and Data Sciences,* Singapore: Springer, 2019, pp. 711–723.
6. Voevodin, Vl.V., Antonov, A.S., Nikitenko, D.A., Shvets, P.A., Sobolev, S.I., Sidorov, I.Yu., Stefanov, K.S., Voevodin, V.V., and Zhumatiy, S.A., Supercomputer Lomonosov-2: large scale, deep monitoring and fine analytics for the user community, *Supercomput. Front. Innovations,* 2019, vol. 6, no. 2, pp. 4–11.
7. Shabanov, B.M. and Samovarov, O.I., Building the software-defined data center, *Program. Comput. Software,* 2019, vol. 45, no. 8, pp. 458–466.
8. Mateescu, G., Gentzsch, W., and Ribben, C.J., Hybrid computing – where HPC meets grid and cloud computing, *Future Gener. Comput. Syst.,* 2011, vol. **27**, no. 5, pp. 440–453.
9. Feoktistov, A., Gorsky, S., Sidorov, I., Kostromin, R., Edelev, A., and Massel, L., Orlando tools: energy research application development through convergence of grid and cloud computing, *Commun. Comput. Inf. Sci.,* 2019, vol. 965, pp. 289–300.
10. Feoktistov, A., Kostromin, R., Sidorov, I., and Gorsky, S., Development of distributed subject-oriented applications for cloud computing through the integration of conceptual and modular programming, in *Proc. 41st Int. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO-2018),* Riejka: IEEE, 2018, pp. 256–261.
11. Yu, J. and Buyya, R., A taxonomy of workflow management systems for grid computing, *J. Grid Comput.,* 2005, vol. **3**, no. 3–4, pp. 171–200.
12. Feoktistov, A., Sidorov, I., Tchernykh, A., Edelev, A., Zorkalzev, V., Gorsky, S., Kostromin, R., Bychkov, I., and Avetisyan, A., Multi-agent approach for dynamic elasticity of virtual machines provisioning in heterogeneous distributed computing environment, *Proc. IEEE Int. Conf. on High Performance Computing and Simulation (HPCS-2018),* Orleans, 2018, pp. 909–916.
13. Bychkov, I., Oparin, G., Feoktistov, A., Sidorov, I., Gorsky, S., Kostromin, R., and Edelev, E., Subject-oriented computing environment for solving large-scale problems of energy security research, *J. Phys.: Conf. Ser.,* 2019, vol. 1368, pp. 052030-1–052030-12.
14. Burri, A., Dedner, A., Klofkorn, R., and Ohlberger, M., An efficient implementation of an adaptive and parallel grid in DUNE, *Comput. Sci. High Perform. Comput. II: Notes Num. Fluid Mech. Multidiscipl. Des.,* 2006, vol. 91, pp. 67–82.
15. Radchenko, G. and Hudyakova, E., A service-oriented approach of integration of computer-aided engineering systems in distributed computing environments, *Proc. UNICORE Summit,* Dresden, 2012, pp. 57–66.

16. Shamakina, A., Brokering service for supporting problem-oriented grid environments, *Proc. UNICORE Summit,* Dresden, 2012, pp. 67–75.

17. *Software for Exascale Computing-SPPEXA 2013-2015,* Bungartz, H.J., Neumann, P., and Nagel, W.E., Eds., Cham: Springer, 2016, vol. 113.

18. Afgan, E., et al., The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update, *Nucl. Acids Res.,* 2018, vol. 46, no. W1, pp. W537–W544.

19. Ananthakrishnan, R., Blaiszik, B., Chard, K., and Chard, R., Globus platform services for data publication, *Proc. ACM Conf. of the Practice and Experience on Advanced Research Computing,* Pittsburgh, 2018, pp. 1–7.

20. Sukhoroslov, O., Supporting efficient execution of workflows on Everest Platform, *Commun. Comput. Inf.,* 2019, vol. 1129, pp. 713–724.

21. Gavvala, S.K., Chandrasheka, J., Gangadharan, G.R., and Buyya, R., QoS-aware cloud service composition using eagle strategy, *Future Gener. Comput. Syst.,* 2019, vol. 90, pp. 273–290.

22. Deelman, E., Peterka, T., Altintas, I., and Carothers, C.D., The future of scientific workflows, *Int. J. High Perform. Comput. Appl.,* 2018, vol. **32**, no. 1, pp. 159–175.

23. Abramovici, A., et al., LIGO: the laser interferometer gravitational-wave observatory, *Science,* 1992, vol. 256, no. 5005, pp. 325–333.

24. Berriman, G.B., et al., Montage: a grid enabled engine for delivering custom science-grade mosaics on demand, *Proc. SPIE – Int. Soc. Opt. Eng.,* 2004, vol. 5493. https://doi.org/10.1117/12.550551

25. Maechling, P., et al., SCEC CyberShake workflows-automating probabilistic seismic hazard analysis calculations, in *Workflows for e–Science,* Springer, 2006. https://doi.org/10.1007/978-1-84628-757-2_10

26. Livny, J., Teonadi, H., Livny, M., and Waldor, M.K., High-throughput, kingdom-wide prediction and annotation of bacterial non-coding RNAs, *PLoS One,* 2008, vol. **3**, no. 9, pp. e3197. https://doi.org/10.1371/journal.pone.0003197

27. USC Epigenome Center. http://epigenome.usc.edu. Accessed 08.12.2019.

28. Wangsom, P., Lavangnananda, K., and Bouvry, P., Multi-objective scientific-workflow scheduling with data movement awareness in cloud, *IEEE Access,* 2019, vol. 7, pp. 177063–177081.

29. Feoktistov, A., Gorsky, S., Sidorov, I., and Tchernykh, A., Continuous integration in distributed applied software packages, *Proc. 42st Int. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO-2019),* Riejka: IEEE, 2019, pp. 1775–1780.

30. Gruver, G., *Start and Scaling Devops in the Enterprise,* BookBaby, 2016.

31. Talia, D., Workflow systems for science: concepts and tools, *ISRN Software Eng.,* 2013, art. ID 404525. https://doi.org/10.1155/2013/404525

32. Deelman, E., et al., Pegasus, a workflow management system for science automation, *Future Gener. Comput. Syst.,* 2015, vol. 46, pp. 17–35.

33. Bumgardner, V.K., *OpenStack in Action,* Shelter Island: Manning Publ., 2016.

34. Spruth, I.W.G., Discovering and classifying regions in workflow graphs, *Diploma Thesis in Computer Science,* Publ. of the University of Tubingen, 2005.

35. Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Gaurang, S., and Mei-Hui, V.K., Characterization of scientific workflows, *Proc. 3rd Workshop on Workflows in Support of Large-Scale Science (WORKS 2008),* Austin, 2008, doi 1-10. https://doi.org/10.1109/WORKS.2008.4723958

36. Hirales-Carbajal, A., González-García, J.L., and Tchernykh, A., Workload generation for trace based grid simulations, in *Proc. 1st Int. Supercomputer Conf. in Mexico (ISUM–2010),* Guadalajara University Publ., 2010, pp. 1–10.

37. Bychkov, I., Oparin, G., Tchernykh, A., Feoktistov, A., Bogdanova, V., and Gorsky, S., Conceptual model of problem-oriented heterogeneous distributed computing environment with multi-agent managemen, *Procedia Comput. Sci.,* 2017, vol. 103, pp. 162–167.

38. Sokolinsky, L.B. and Shamakina, A.V., Methods of resource management in problem-oriented computing environment, *Program. Comput. Software,* 2016, vol. 42, no. 1, pp. 17–26.

39. Ramírez-Velarde, R., Tchernykh, A., Barba-Jimenez, C., Hirales-Carbajal, A., and Nolazco, J., Adaptive resource allocation with job runtime uncertainty, *J. Grid Comput.,* 2017, vol. 15, no. 4, pp. 415–434.

40. Tchernykh, A., Schwiegelshohn, U., Talbi, E.-g., and Babenko, M., Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability, *J. Comput. Sci.,* 2019, vol. 36, p. 100581. https://doi.org/10.1016/j.jocs.2016.11.011

41. Babenko, M., Chervyakov, N., Tchernykh, A., Kucherov, N., Shabalina, M., Vashchenko, I., Radchenko, G., and Murga, D., Unfairness correction in P2P grids based on residue number system of a special form, *Proc. 28th IEEE Int. Workshop on Database and Expert Systems Applications (DEXA),* Lyon, 2017, pp. 147–151.

42. Singh, A. and Malhotra, M., Agent based framework for scalability in cloud computing, *Int. J. Comput. Sci. Eng.,* 2012, vol. **3**, no. 4, pp. 41–45.

43. Kalyaev, A.I. and Kalyaev, I.A., Method of multiagent scheduling of resources in cloud computing environments, *J. Comput. Syst. Sci. Int.,* 2016, vol. 55, no. 2, pp. 211–221.

44. Prieto, A.G., Gillblad, D., Steinert, R., and Miron, A., Toward decentralized probabilistic management, *IEEE Commun. Mag.,* 2011, vol. 49, no. 7, pp. 80–86.

45. Walsh, A., *UDDI, SOAP, and WSDL: the Web Services Specification Reference Book,* Pearson Education, 2002.

46. Bychkov, I.V., Oparin, G.A., Feoktistov, A.G., Sidorov, I.A., Bogdanova, V.G., and Gorsky, S.A., Multiagent control of computational systems on the basis of meta-monitoring and imitational simulation, *Optoelectron., Instrum. Data Process.,* 2016, vol. 52, no. 2, pp. 107–112.

47. Java Agent DEvelopment Framework. https://jade.tilab.com. Accessed 08.12.2019.

48. Herrera, J., Huedo, E., Montero, R., and Llorente, I., Porting of scientific applications to grid computing on GridWay, *Sci. Program.,* 2005, vol. 13, no. 4, pp. 317–331.

49. Tannenbaum, T., Wright, D., Miller, K., and Livny, M., *Condor – a Distributed Job Scheduler. Beowulf Cluster Computing with Linux,* The MIT Press, 2002, pp. 307–350.

50. Feoktistov, A., Tchernych, A., Kostromin, R., and Gorsky, S., Knowledge elicitation in multi-agent system for distributed computing management, *Proc. 40th Int. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO-2017),* Riejka: IEEE, 2017, pp. 1350–1355.

51. Feoktistov, A., Kostromin, R., Sidorov, I., Gorsky, S., and Oparin, G., Multi-agent algorithm for re-allocating grid-resources and improving fault-tolerance of problem-solving processes, *Procedia Comput. Sci.,* 2019, vol. 150, pp. 171–178.

52. Vickrey, W., Counterspeculation, auctions, and competitive sealed tenders, *J. Finance,* 1961, vol. 16, no. 1, pp. 8–37.

53. Edelev, A., Zorkaltsev, V., Gorsky, S., Doan, V.B., and Nguyen, H. N., The combinatorial modelling approach to study sustainable energy development of Vietnam, *Commun. Comput. Inf. Sci.,* 2017, vol. 793, pp. 207–218.

54. Irkutsk Supercomputer Centre of SB RAS. http://hpc.icc.ru. Accessed 08.12.2019.

55. Tchernykh, A., Feoktistov, A., Gorsky, S., Sidorov, I., Kostromin, R., Bychkov, I., Basharina, O., Alexandrov, A., and Rivera-Rodriguez, R., Orlando tools: development, training, and use of scalable applications in heterogeneous distributed computing environments, *Commun. Comput. Inf. Sci.,* 2019, vol. 979, pp. 265–279.

56. Bychkov, I.V., Oparin, G.A., Tchernykh, A.N., Feoktistov, A.G., Gorsky, S.A., and Rivera-Rodriguez, R., Scalable application for the search of global minima of multiextremal functions, *Optoelectron., Instrum. Data Process.,* 2018, vol. 54, no. 1, pp. 83–89.