Problem-Oriented Scheduling of Cloud Applications: PO-HEFT Algorithm Case Study

E.A. Nepovinnykh and G.I. Radchenko

\*South Ural State University, Chelyabinsk, Russia

nepovinnykhea@susu.ru, gleb.radchenko@susu.ru

Abstract - Today we see a significantly increased use of problem-oriented approach to the development of cloud computing environment scheduling algorithms. There are already several such algorithms. However, a lot of these require that the tasks within a single job are independent and do not account for the execution of each task and the volume of data transmitted. We propose a list-based algorithm of problem-oriented planning of execution of applications in a cloud environment that considers the applications' execution profiles. It provides payroll algorithm for the problem-oriented scheduling applications in the cloud environments based on their computing profiles. Scheduling on the basis of lists suggests prioritization of computing tasks and running in blocks to perform according to the obtained priorities. The proposed approach allows us to take into account the costs of the transfer of data between nodes, thereby reducing the total run time of the workflow. The proposed algorithm is based on an algorithm of Heterogeneous Earliest-Finish-Time (HEFT), but contains modifications in calculation of a node level objectives and takes into account the cost of incoming communications of its parent task.

*Keywords* – *scheduling, execution planning, cloud computing, grid computing, HEFT*

# Introduction

Today a lot of complex e-Sceince tasks are solved using computer simulation which usually requires significant computational resources usage [1]. Moreover, the solutions, developed for such tasks are often characterized by structural complexity, which causes different resources (informational, software or hardware) to be integrated within a single solution. The complexity of the solutions grows as the multidisciplinary tasks are considered.

Today’s common approach for building composite solutions is based on Service-Oriented Architecture [2] which forms the basis from interconnection of services and hiding their complexity behind their interfaces. Interconnection of the services within complex tasks is usually implemented in a form of workflow structures, which exploits graph-based structures to describe interconnection of used services. On the other hand, today the Cloud Computing concept is developed as a business framework for providing on-demand services supporting computing resources’ consolidation, abstraction, access automation and utility within a market environment. The service-oriented architecture in the cloud is best implemented using the microservice approach. The microservice model describes a cloud application as a suite of small independent services, each running in its own container and communicating with other services using lightweight mechanisms. These services are built around separate business capabilities, independently deployable and may be written by different development teams using different programming languages and frameworks [3].

To provide scientists and engineers a transparent access to the computing resources a “Problem Solving Environment” (PSE) concept is commonly used. A PSE is a system that provides all the computational facilities necessary to solve a target class of problems. It uses the language of the target class and users need not have specialized knowledge of the underlying hardware or software [4]. At present, PSE researchers are investigating a variety of fields, e.g., Cloud computing support, education support, CAE usage support, document generation support, and so on.

Today most of the systems that provide a problem-oriented approach to e-science problems on the basis of high performance computing resources use workflows to organize a computational process [5]. Nodes of such workflows represents separate tasks implemented by individual services, and the edges define the data or control flow. In this paper, under the “Problem Solving Environment” term we would understand a set of services, software and middleware focused on the implementation of workflows to solve e-Science problems in a specific problem domain, using resources of cloud computing system [6].

Within a problem domain of PSE, a set of tasks, forming the workflow, is predetermined. Those tasks can be grouped into a finite set of classes. Task class is a set of tasks that have the same semantics and the same set of input parameters and output data. On the one hand, this imposes restrictions on the class of problems that can be solved using the PSE. On the other hand, such restriction allows to use a domain-specific information (such as task execution time on one processor core, scalability limits, and the amount of generated data) during resources allocation and scheduling, increasing the efficiency of use of available computational resources.

So, in order to increase efficiency of distributed problem-oriented computer environments it is feasible to use problem-oriented task scheduling methods that use domain-specific information in order to predict computational attributes of a particular workflow.

The main goal of the research is to develop a scheduling algorithm for a workflow-based problem-solving environment, which would effectively use a domain-specific information (such as task execution time, scalability limits, and the amount of data transfer) for prediction of cloud computing environment resources load.

This paper is organized as follows. In section II we present the concept and the basic idea of scheduling applications in cloud environments. In section III we describe the results of the analysis of existing algorithms of resource scheduling. In section IV we describe HEFT and PO-HEFT cloud scheduling algorithms complete with a mathematical task model. In section V we describe the implementation of PO-HEFT algorithm in CloudSim cloud environment simulation package. In section VI we describe the results of benchmarking PO-HEFT against CloudSim's built-in scheduling algorithm. In section VII we summarize the results of our research and give further research directions.

# scheduling applications in cloud environments

Analysis of the main trends in resource scheduling research in distributed problem-oriented environments shows that the theme of the problem-oriented scheduling and prediction of environment load is an urgent task.

In the cloud computer data centers Holistic Model for Resource Representation is used in virtualized cloud computing data [7]. This model is designed to represent physical resources, virtual machines, and applications in cloud computing environments. The model can be applied to represent cloud applications, VMs, and physical hosts. Each of these entities is described by multiple resources: computing, memory, storage, and networking. The model is scalable, as it does not increase the simulation time and creates a limited memory overhead of less than 10% with the respect to the Simple model. This additional memory is required for the creation a separate objects for each resource demand and provision. A holistic model increases the precision of simulations and enables a number of new simulation scenarios focused on heterogeneity of the hardware resources and virtualization. The model distinguishes between computing, memory, storage, and networking types of resources. However, the model can easily scale to include other types of resources as well, e.g., additional GPGPU units.

New cloud-related techniques for resource virtualization and sharing and the corresponding service level agreements call for new optimization models and solutions. Computational Intelligence proves to be applicable to multiple resource management problems that exist at all layers of Cloud computing. Standard optimization objectives for scheduling are to minimize makespan and cost, but additional objectives may include optimization of energy consumption or communications. Solutions to this multi-objective optimization problem include but are not limited to: Improved Differential Evolutionary Algorithm combined with the Taguchi method, Multi-Objective Evolutionary Algorithm based on NSGA-II, Case Library and Pareto Solution based hybrid GA Particle Swarm Optimization, Auction-Based Biobjective Scheduling Strategy etc. [2]. The main drawback of mentioned algorithms is the fact that they do not use information about previous executions.

The main reason that traditional cluster and grid resource allocation approaches fail to provide efficient performance in clouds is that most of cloud applications require availability of communication resources for information exchange between tasks, with databases or the end users [10]. CA-DAG model for cloud computing applications, which overcomes shortcomings of existing approaches using communication awareness. This model is based on Directed Acyclic Graphs that in addition to computing vertices include separate vertices to represent communications. Such a representation allows making separate resource allocation decisions: assigning processors to handle computing jobs, and network resources for information transmissions. A case study is given and corresponding results indicate that DAG scheduling algorithms designed for single DAG and single machine settings are not well suited for Grid scheduling scenarios, where user run time estimates are available.

For practical purposes quite simple scheduler MaxAR with minimal information requirements can provide good performance for multiple workflow scheduling [11]. In real Grid environments this strategy might have similar performance comparing with the best ones when considering approximation factor, mean critical path waiting time, and critical path slowdown. Besides the performance aspect the use of MaxAR does not require additional management overhead such as DAG analysis, site local queue ordering, and constructing preliminary schedules by the Grid broker. It has small time complexity. This approach is related with offline scheduling which can be used as a starting point for addressing the online case. Online Grid workflow management brings new challenges to above problem, as it requires more flexible load balancing workflows and their tasks over the time .

Nowadays the shifting emphasis of clouds towards a service-oriented paradigm has led to the adoption of Service Level Agreements (SLAs) [12]. The use of SLAs has a strong influence on job scheduling, as schedules must observe quality of service constraints. In terms of minimizing power consumption and maximizing provider income Min-e outperforms other allocation strategies. The strategy is stable even in significantly different conditions. The information about the speed of machines does not help to improve significantly the allocation strategies. When examining the overall system performance on the real data, it is determined that appropriate distribution of energy requirements over the system provide more benefits in income and power consumption than other strategies. Min-e is a simple allocation strategy requiring minimal information and little computational complexity. Nevertheless, it achieves good improvements in both objectives and quality of service guarantees. However, it is not assessed its actual efficiency and effectiveness.

One of the most popular algorithms is scheduled list-based algorithm Min-min [13]. Min-min sets high scheduling priority to tasks which have the shortest execution time. The main drawback of scheduled list-based algorithms is that they do not analyze the whole task graph.

One of the important classes of computational problems is problem-oriented workflow applications executed in distributed computing environment [14]. A problem-oriented workflow application can be represented by a directed graph whose vertices are tasks and arcs are data flows. Problem-oriented scheduling (POS) algorithm is proposed. The POS algorithm takes into account both specifics of the problem-oriented jobs and multi-core structure of the computing system nodes. The POS algorithm is designed for use in distributed computing systems with manycore processors. The algorithm allows one to schedule execution of one task on several processor cores with regard to constraints on scalability of the task.

Cloud computing can satisfy the different service requests with different configuration, deployment condition and service resources of various user at different time point. With the influence of multidimensional factors, it is unreality to test with different parameters in actual cloud computing center. Typical Tools for Cloud Workflow Scheduling Research are CloudSim and WorkflowSim [9]. CloudSim is a toolkit (library) for simulation of cloud computing scenarios. It provides basic classes for describing data centers, virtual machines, applications, users, computational resources, and policies for management of diverse parts of the system (e.g., scheduling and provisioning).

WorkflowSim extends the CloudSim simulation toolkit by introducing the support of workflow preparation and execution with an implementation of a stack of workflow parser, workflow engine and job scheduler. WorkflowSim is used for validating Graph algorithm, distributed computing, workflow scheduling, resource provisioning and so on. Compared to CloudSim and other workflow simulators, WorkflowSim provides support of task clustering that merges tasks into a cluster job and dynamic scheduling algorithm that jobs matched to a worker node whenever a worker node become idle.

In the following sections we would present a new problem-oriented resource scheduling algorithm for distributed computing environments, which uses heuristic score-based approach based on the HEFT algorithm for the task of the problem-oriented scheduling in cloud environments.

# Computing environment model

We define a *computational node* as a computational system with shared memory that is represented as 3 components:

where is an ordered set of node’s computational cores; ; is node’s available RAM ; – is node’s computational parameters vector .

We define a *virtual machine* prototype as :

where is an ordered set of virtual machine’s computational cores ; is virtual machine’s available RAM; is virtual machine’s computational parameters vector .

We define a computational unit’s performance factor as :

where is a virtual machine prototype that exists in the computational system .

Numerical characteristics of a machine, synthetic tests results or existing functions’ test execution results can serve as performance characteristics examples [15, 16] .

Obviously, in order to maximize the quality of parameters prediction to perform the tasks on specified machines, we need to take into account the maximum possible number of performance characteristics, including characteristics such as the number of available processors; CPU frequency; speed data exchange with the hard drive; characteristics of LINPACK machines and so on. Thus, we define the vector of the performance characteristics of virtual machines deployed in the cloud computing system :

Each machine in a cloud computing system is comparable to the performance characteristics of the vector, which reflects the values of the performance of the computer:

In what follows we assume that in the framework of the provision of computational resources, each task is allocated one or more virtual machines. Direct access to components of the computing system is not ensured.

One particular feature of a problem-oriented computing environment is the fact that said environment uses information about task classes’ features during scheduling and resource provisioning. We require that every task class should have these functions defined for prediction of task execution process depending on input parameters: :

1) input data volume estimation function;

2) task execution time estimation function on a computer with a given performance values vector. .

Thus, for each function of the domain that is running in a task-oriented environment , define the following set of statements:

1. *The operator of the expected output*  ,) – it is the operator that returns the expected total size in bytes of output data objects :

|  |  |
| --- | --- |
|  |  |

1. *the operator of the expected function’s execution time* , hat returns the estimated run time (in seconds) of a function for a given set of input data on a given machine, with the performance characteristics vector :

|  |  |
| --- | --- |
|  |  |

Execution time of a function on a given computer with a performance values vector can be defined as an operator that takes input information objects vector . Unfortunately it is impossible to estimate a function execution time with absolute accuracy due to the fact that the computations involved in output information objects preparation might indirectly depend on multiple factors that our model does not account for, including, but not limited to, background processes, available cache volume, branch prediction rate, etc.. In order to account for this inherent inaccuracy, execution time estimate can be modelled as a random value that is a sum of two parts.:

|  |  |
| --- | --- |
|  |  |

where – a deterministic function that represents a dependency of execution time of function f that is running on a computer with a performance values vector on input information objects vector , – a stochastic value with the expected value (), that represents factors that our model does not account for.

In order to estimate execution time we need to store a statistical data about every task execution. After every execution this data gets saved into the database: execution parameters, performance values vector, execution time and generated data size. Performance values vector includes but is not limited to a number of processing cores and allocated RAM size.

The majority of existing algorithms require that tasks within a single job are independent and do not account for the execution time of each task or the amount of data transmitted. The application is considered to be performed on modern distributed computing systems, and to have a complex structure of the stream, which is a set of ordered tasks related with data dependencies.

We give a formal definition of a task in a distributed computing environment:

A task graph is a labeled weighted directed acyclic graph G=〈T,E,init,fin〉, where T is the set of nodes corresponding to the tasks and E is the set of arcs corresponding to the data streams.

Weight S(e) of arc e defines the amount of data to be transmitted by arc e from the task associated with vertex init(e) to the task associated with vertex fin(e).

One approach to solving the problem is to break the task graph into independent sections. The presence of parallel task blocks, represented in the form of workflow, allows us to immediately use multiple distributed resources to quickly find a solution. When planning task flows, the following criteria must be considered:

* Time to solve a particular task;
* The deployment time of virtual machines. All VM images are stored in the database, and not on on all nodes at once. Image transfer speed in the system is at least equal to the data rate within one group.
* The transmission time of data between nodes in accordance with the network bandwidth. The costs of data transfer can be eliminated by clustering multiple task flow blocks on one resource [10, 12].

The problem is to optimize the distribution of virtual machines on nodes of cloud platform so that the graph solving time is minimal. The problem is reduced to finding sustainable solution between clustering and parallelization for each group, and task is to find the optimal variant of placing workflow at available large number of resources. During our research, we should:

* Analyze existing approaches to cloud workflow scheduling;
* Propose a scheduling algorithm that uses information from previous runs as a heuristic;
* Implement this algorithm in the CloudSim cloud evironment simulation platform;
* Compare this algorithm against existing schedulers within a cloud simulation.

## Restrictions imposed on the computing environment

We assume that in the model of problem-oriented services, the computing environment meets the following conditions.

1. Cloud is a set of nodes connected by a network.
2. The data transfer rate within the same node is infinite.
3. The nodes are combined into groups. We assume that the nodes within the group are connected in the topology of "each-to-each", where a maximum capacity of connections between any two nodes within the same group () is the same and is determined in advance.

C:\Users\Екатерина\Downloads\HEFT.png

Figure 1: The structure of nodes of the cloud platform

1. We assume that the computing nodes belonging to the same group have the same computing power and provide the same amount of resources (memory, storage capacity).
2. Groups of nodes in the cluster are combined. Let us assume that a group within a cluster interconnected topology "each-to-each", with a maximum capacity of connections between any two groups in the same cluster () is the same and is determined in advance.
3. Different groups of nodes with different processing capabilities and different resources vicinity can interact within the same cluster.
4. Clusters are interconnected by high-speed Internet connection. We assume that we are given the maximum network throughput between each pair of clusters. Network bandwidth will be considered the same in both directions (Fig. 1).
5. The network bandwidth between the nodes in the group is greater than the network bandwidth between the groups in the cluster, and the network bandwidth between the groups in the cluster is greater than the network bandwidth between the clusters:
6. Data links are reliable, i.e. there is no loss or duplication of data packets during the transmission.
7. The delays do not affect the data transfer.
8. We will consider two types of nodes in the cloud:

* Computing nodes ensure the provision of computational resources for the functioning of the problem-oriented services through independent virtualized containers. Provide a limited capacity for data storage.
* Storage units enable the storage and remote access to large volumes of data.

1. We assume that the transmission of data in the cloud does not require computational resources and can be conducted in parallel with the computations.
2. We assume that the cloud nodes execute tasks exclusively within the framework of problem-oriented environment, and do not perform any other tasks.

# Approaches to the scheduling

These methods are used for the implementation of scheduling in cloud environments:

* Scheduling on the basis of the theory of management;
* Scheduling using clustering algorithms;
* Scheduling with a time limit;
* Communal method;
* Market method.

Scheduling for all computing blocks in a task graph can be done on the basis of a static or dynamic approach. The static approach involves distribution of computational nodes before starting the job and requires information about the current state of computer network resources, output and consistency of the job prior to the task execution. The dynamic approach allows for the allocation of resources in the course of the assignment, as well as the process of branching in the job structure, which significantly complicates the scheduling process. We propose the introduction of a hybrid approach to scheduling, which uses static methods for primary distribution followed by the dynamic regulation, taking into account the dynamics of the job and the status of network resources to ensure that tasks are rescheduled as needed [7].

We decided to use a CloudSim cloud platform because it allows us to simulate a cloud environment with different scheduling algorithms and to compare their efficiency. CloudSim is a toolkit (library) for simulation of Cloud computing scenarios. It provides basic classes for describing data centers, virtual machines, applications, users, computational resources, and policies for management of diverse parts of the system (e.g., scheduling and provisioning) [1], [2].

# HEFT algorithm for the task of the problem-oriented scheduling

The use of heuristic score-based approach based on the HEFT algorithm for the task of the problem-oriented scheduling in cloud environments.

We offer a list-based algorithm for problem-oriented scheduling in cloud environments based on their computing profiles. List-based scheduling involves the definition of computational units' priorities and starting the execution according to the received priority. The binding to high-priority tasks resources takes place first. The proposed approach allows us to take into account the costs of transmission of data between nodes, thereby reducing the total time of execution of the workflow. The proposed algorithm is based on an algorithm of Heterogeneous Earliest-Finish-Time (HEFT), but contains modifications during the node level computation phase, and takes into account the problem of calculating the incoming communication value of its parent task [11].

Let be the size of the problem , and the be the set of computing resources with an average processing power . Then, the average time to complete the task with all available resources is calculated as

|  |  |
| --- | --- |
|  | (1) |

Let be the amount of data transferred between tasks and , and R be the set of available resources with an average capacity of data transfer . Then the average score on data transfer costs between tasks and for all pairs of p.

(2)

Thus, the priority calculation unit may be defined as

(3)

where is the set of tasks that depend on the task .

Thus, the task priority is directly determined by the priority of all its dependent tasks. Assign tasks to the resources as follows: a task with a higher priority if all the tasks on which it depends, is appointed to the computing resource, providing less time for the task [8].

Fig. 2. Heuristic scheduling algorithm in cloud environments HEFT

|  |
| --- |
| **PROCEDURE:** HEFT  **INPUT:** TaskGraph G(T, E), TaskDistributionList, ResourcesSet R  **BEGIN**  **for** each t T from task graph G  Calculate task execution time according to (1)  **for** each e E from task graph G  Calculate data transfer time according to (2)  Start the width-first search in reverse task order and calculate a rank for each task according to (3)  **while** T has unfinished tasks  TaskList <- get completed tasks from G  Schedule Task (TaskList, R)  Update TaskDistributionList  **END**    **PROCEDURE:** Schedule Task  **INPUT**: TaskList, ResourcesSet R  **НАЧАЛО**  Sort TaskList in reverse task rank order  **for each** t from TaskList  r <- get resource from R that can complete t earlier  schedule t on r  update status of r  **END** |

Taking into account the specifics of the problem-oriented cloud computing environment following modifications apply to this algorithm:

Let be the set of all functions that can be implemented in the subject area. Then a separate problem is nothing more than a single instance of a function with a set of input data objects :

,

We define R as the set of available for the deployment virtual machines with mean production capacity .

In this case, for evaluating the execution time we can apply the following formula:

.(4)

The model of problem-oriented services should take into account the amount of data returned by each task . This may be used by the operator of the expected output ,), which returns the expected total size in bytes of output data objects . Consequently, within the framework of problem-oriented model for the evaluation of data transmission time between two tasks the following formula can be used:

(5)

Where is the bandwidth of data transmission channel in the cloud computing system. During the execution of task it can be estimated as one of the following values:

1) , when the data transmission channel consists of a single node;

2) , when the data transmission channel is shared by a group of nodes;

3) , when the data transmission channel is shared by a cluster of compute nodes.

Thus, the priority of a calculation unit may be defined as: (6)

where is the set of all tasks that are dependent on the task T\_x.

Figure 3 shows the pseudo-code for algorithm of problem-oriented work flow scheduling in a cloud computing environment based on computing profiles.

|  |
| --- |
| **PROCEDURE:** PO-HEFT  **INPUT:** TaskGraph G(T, E), TaskDistributionList, ResourcesSet R  **BEGIN**  **for each** t T from task graph G  Approximate task execution time according to (4)  **for each** e E from task graph G  Approximate data transfer time according to (5)  Start the width-first search in reverse task order and calculate a rank for each task according to (6)  **while** T has unfinished tasks      TaskList <- get completed tasks from task graph G      Schedule Task (TaskList, R)      Update TaskDistributionList  **END**  **PROCEDURE:** Schedule Task  **INPUT:** TaskList, ResourcesSet R  **BEGIN**  Sort TaskList in reverse task rank order  **for each** t from TaskList  r <- get resource from R that can complete t earlier  schedule t on r  update status of r  **END** |

# Implementation

In order to assess the proposed algorithm’s efficiency, we had to develop a benchmark using CloudSim cloud environment simulation platform. We have implemented the PO-HEFT algorithm itself, as well as a naive bruteforce algorithm that finds and ideal scheduling solution.

Unfortunately, due to a great computational complexity of full search bruteforce, we could not benchmark agains it on all but the most trivial use cases. Hence why the PO-HEFT algorithm have only been benchmarked agains CloudSim’s own built-in space-shared scheduling policy which internally uses round-robin for VM scheduling.

The algorithm was implemented as a number of Java classes so that CloudSim can use it as the simulated cloud environment’s scheduler. We have implemented both a custom DatacenterBroker in order to schedule VMs in a data center and a custom CloudletScheduler in order to schedule tasks (cloudlets in CloudSim’s terminology) in a single VM.

# Performance evaluation

The algorithm was tested in a simulation in which virtual machines with homogeneous characteristics have been deployed. The simulated system was given the same work flow 60 times, which greatly exceeds the capacity of the system. For the distribution of the workflow we have used: a scheduler that does not use the information about the previous system runs that is built in CloudSim itself, the perfect scheduler, which implements the ideal scheduling through complete search space enumeration and a scheduler based on the PO-HEFT algorithm, which uses information about previous runs. The computational complexity of the perfect scheduler does not allow its usage in any non-trivial simulation and, therefore, this algorithm is not present in this comparison.

Experiments have shown that the PO-HEFT algorithm is quite efficient for the task. We have measured a total simulated workflow execution time for each algorithm. In a cloud with 500 nodes the execution time with a built-in scheduler was 43 seconds, and with PO-HEFT algorithm this time was 39 seconds which is a significant improvement. Thus, the algorithm is relevant and effective for planning tasks in problem-oriented cloud environments.

# Conlusion

##### In this article, we described the PO-HEFT scheduling algorithm, which aims to provide an efficient workflow scheduling in heterogenous distributed cloud environments. The main distinctive feature of this algorithm is it's ability to adapt the solution based on previous runs, which allows this algorithm to provide better resource utilization.

##### The algorithm's efficiency was assessed in the CloudSim cloud environment simulation software. As a benchmark we used CloudSim's built-in scheduler called "space-shared scheduling policy" which uses round-robin for resource provisioning and virtual machines creation. Our proposed algorithm have shown significant efficiency gains over this simple scheduler.

##### As a further development we will investigate the possibility of deploying this algorithm at a real cluster in order to assess its real-life, non-simulated performance. We will also compare this algorithm against different algorithms that do not use information about previous runs in order to give an empirical prove that this is a viable heuristic in workflow scheduling.

##### References

[1] S. V. Kovalchuk, P. A. Smirnov, K. V. Knyazkov, A. S. Zagarskikh, and A. V. Boukhanovsky, “Knowledge-based expressive technologies within cloud computing environments,” *Adv. Intell. Syst. Comput.*, vol. 279, pp. 1–11, 2014.

[2] D. I. Savchenko, G. I. Radchenko, and O. Taipale, “Microservices validation: Mjolnirr platform case study,” *2015 38th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2015 - Proc.*, pp. 235–240, 2015.

[3] J. Thones, “Microservices,” *IEEE Softw.*, vol. 32, no. 1, pp. 116–116, 2015.

[4] H. Kobashi, S. Kawata, Y. Manabe, M. Matsumoto, H. Usami, and D. Barada, “PSE park: Framework for problem solving environments,” *J. Converg. Inf. Technol.*, vol. 5, no. 4, pp. 225–239, 2010.

[5] E. DEELMAN, D. GANNON, M. SHIELDS, and I. TAYLOR, “Workflows and e-Science : An overview of workflow system features and capabilities,” *FGCS. Futur. Gener. Comput. Syst.*, vol. 25, no. 5, pp. 528–540.

[6] A. Shamakina, “Brokering service for supporting problem-oriented grid environments,” *UNICORE Summit 2012, Proc.*, vol. 15, pp. 67–75, 2012.

[7] M. Guzek, D. Kliazovich, and P. Bouvry, “A Holistic Model for Resource Representation in Virtualized Cloud Computing Data Centers,” *IEEE Int. Conf. Cloud Comput. Technol. Sci.*, pp. 590–598, 2013.

[8] P. Bouvry and B. Service, “Review Article A Survey of Evolutionary Computation for Resource Management of Processing in Cloud Computing,” no. may, pp. 53–67, 2015.

[9] C. Chen, J. Liu, Y. Wen, and J. Chen, “CCIS 495 - Research on Workflow Scheduling Algorithms in the Cloud,” *Ccis*, vol. 495, pp. 35–48, 2015.

[10] D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan, and A. Y. Zomaya, “CA-DAG: Modeling Communication-Aware Applications for Scheduling in Cloud Computing,” *J. Grid Comput.*, 2015.

[11] A. Hirales-Carbajal, A. Tchernykh, R. Yahyapour, J.-L. Gonzalez-Garcia, T. Roblitz, and J. M. Ramirez-Alcaraz, “Multiple workflow scheduling strategies with user run time estimates on a Grid,” *J. Grid Comput.*, vol. 10, no. 2, pp. 325–346, 2012.

[12] A. Tchernykh, L. Lozano, U. Schwiegelshohn, P. Bouvry, J. E. Pecero, S. Nesmachnow, and A. Y. Drozdov, “Online Bi-Objective Scheduling for IaaS Clouds Ensuring Quality of Service,” *J. Grid Comput.*, 2015.

[13] J. Yu, R. Buyya, and K. Ramamohanarao, “Workflow Scheduling Algorithms for Grid Computing,” *Springer Berlin Heidelb.*, vol. 146, pp. 173–214, 2008.

[14] L. B. Sokolinsky and A. V. Shamakina, “Methods of resource management in problem-oriented computing environment,” *Program. Comput. Softw.*, vol. 42, no. 1, pp. 17–26, 2016.

[15] J. J. Dongarra, P. Luszczek, and A. Petite, “The LINPACK benchmark: Past, present and future,” *Concurr. Comput. Pract. Exp.*, vol. 15, no. 9, pp. 803–820, 2003.

[16] “WPrime Systems. Super PI. 2013.” [Online]. Available: http://www.superpi.net/ . [Accessed: 14-Nov-2015].

1. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.
2. Cloiudsim documentation [Online] Available from: http://www.cloudbus.org/cloudsim/ [Accessed 22 Jan 2016].
3. C. Chen, J. Liu, Y. Wen, and J. Chen, “Research on workflow scheduling algorithms in the cloud,” *Commun. Comput. Inf. Sci.*, vol. 495, pp. 35–48, 2015.
4. S. Chen, Y. Wang, and M. Pedram, “Concurrent placement, capacity provisioning, and request flow control for a distributed cloud infrastructure,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, 2014, pp. 1–6.
5. G. Radchenko and E. Hudyakova, “A service-oriented approach of integration of computer-aided engineering systems in distributed computing environments,” in *UNICORE Summit 2012, Proceedings*, 2012, vol. 15, pp. 57–66..
6. D. C. Marinescu, *Cloud Computing: Theory and Practice*. Morgan Kaufmann, 2013.
7. G. Radchenko and E. Hudyakova, “Distributed virtual test bed: An approach to integration of CAE systems in UNICORE grid environnement,” *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2013 - Proceedings*. pp. 163–168, 2013.
8. J. Rhoton and R. Haukioja, *Cloud Computing Architected: Solution Design Handbook*. Recursive, Limited, 2011.
9. L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, p. 50, 2008..
10. L. B. Sokolinsky and A. V. Shamakina, “Methods of resource management in problem-oriented computing environment,” *Program. Comput. Softw.*, vol. 42, no. 1, pp. 17–26, Jan. 2016.
11. H. Topcuoglu and S. Hariri, “Performance-effective and low-complexity task scheduling for heterogeneous computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.
12. S. Yassa, J. Sublime, R. Chelouah, H. Kadima, G.-S. Jo, and B. Granado, “A Genetic Algorithm for Multi-objective Optimisation in Workflow Scheduling with Hard Constraints,” *Int. J. Metaheuristics*, vol. 2, no. 4, pp. 415–433, 2013.