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 model of problem-oriented cloud environment. Using this model, we propose a list-based algorithm of problem-oriented planning of execution of applications in a cloud environment that considers the applications' execution profiles, based on a Heterogeneous Earliest-Finish-Time (HEFT) algorithm.

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

I. INTRODUCTION

Today a lot of complex e-Science 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

The reported study was partially supported by RFBR, research project No. 14-07-00420-a and by Grant of the President of the Russian Federation No. MK-7524.2015.9

describe the cloud-based problem solving environment model. 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 Workflow Sim's cloud environment simulation package. In section VI we summarize the results of our research and give further research directions.

II. 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. A holistic model increases the precision of cloud environment simulation 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 users 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.

III. CLOUD-BASED PROBLEM SOLVING ENVIRONMENT MODEL

Let us define a model for cloud problem solving environment, so we could simulate the task scheduling algorithm. We would work in the conditions, where a set $\mathfrak{M}$ of virtual machines $\mathfrak{m} \in \mathfrak{M}$ was distributed between all available nodes $\mathfrak{n} \in \mathfrak{N}$ of cloud platform.

Let us define a virtual machine image performance factor as:

$$\pi: \mathfrak{m} \to \mathbb{Z}_{>0},$$

where $\mathfrak{m}$ is a virtual machine image.

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

In order to maximize the quality of task characteristics prediction on specified machine, we need to take into account several performance characteristics, including such characteristics as the number of available processors and memory; CPU frequency; hard drive data exchange speed; LINPACK test results and so on. Thus, let us define a vector Π of the performance characteristics of virtual machines deployed in the cloud-based PSE:

$$\Pi = [\pi_1, \pi_2 \dots \pi_r].$$

Each machine $\mathfrak{m} \in \mathfrak{M}$ in a cloud-based PSE is comparable to the performance characteristics of the vector Π, which reflects the values of the performance of the machine:

$$\Pi: \mathfrak{M} \to \mathbb{Z}_{>0}^{r}.$$

Let's define a set of tasks, that can be executed in a PSE as a set $\mathcal{F}$ of functions $f \in \mathcal{F}$. Each function $f: \mathcal{C}^{in} \to \mathcal{C}^{out}$ receives n information objects $\mathcal{I}^{in} = (I_1^{in}, \dots, I_n^{in})$ of classes $\mathcal{C}^{in} = (C_1, \dots, C_n)$. The result of the function is m new information objects $\mathcal{I}^{out} = (I_1^{out}, \dots, I_m^{out})$ of classes $\mathcal{C}^{out} = (C_1', \dots, C_m')$. We assume that in our model, each task of a workflow is allocated to one virtual machine. Direct access to components of the computing system is not provided.

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) output data volume estimation function;
2) task execution time estimation function on a machine with a given performance characteristics values vector.

To implement the problem-oriented scheduling, let us define two operators, which should be implemented in the PSE:

1) *The operator of the expected output* $\nu(f, \mathcal{I}^{in})$ – it is the operator that returns the expected total size in bytes of output data objects $\mathcal{I}^{out}$ for the function $f: \mathcal{C}^{in} \to \mathcal{C}^{out}$:

$$\nu(f, \mathcal{I}^{in}) = |\mathcal{I}^{out}|.$$

2) *The operator of the expected function's execution time* $\tau(f, \mathcal{I}^{in}, \Pi)$, that returns the estimated run time (in seconds) of a function $f: \mathcal{C}^{in} \to \mathcal{C}^{out}$ for a given set of input data $\mathcal{I}^{in}$ on a given machine, with the performance characteristics vector Π:

$$\tau: (f, \mathcal{I}^{in}, \Pi) \to \mathbb{N}.$$

Execution time of a function $f: \mathcal{C}^{in} \to \mathcal{C}^{out}$ on a given machine with a performance values vector Π can be defined as an operator that takes input information objects vector $\mathcal{I}^{in}$. 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 $\mathcal{I}^{out}$ 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 take into account this inherent inaccuracy, execution time estimate can be modelled as a random value that is a sum of two parts:

$$\chi(f, \Pi, \mathcal{I}^{in}) = \tau(f, \Pi, \mathcal{I}^{in}) + \alpha,$$

where $\tau(f, \Pi, \mathcal{I}^{in})$ – a deterministic function that represents a dependency of execution time of the function f that is running on a computer with a performance values vector Π on input information objects vector $\mathcal{I}^{in}$, α – a stochastic value with the expected value ($M[\alpha] = 0$), that represents factors that our model does not account for.

In conditions, when a specific function (task) of PSE is implemented on the basis of a virtual machine with a pre-defined performance characteristics vector Π_0, to evaluate

the expected value $\hat{E}[\chi(f, \Pi_0, \mathcal{J}^{in})]$ we can use the k-nearest neighbor method, based on records of execution time of the previous launches of function f on the same machine with close values of input parameters:

$$\hat{E}[\chi(f, \Pi_0, \mathcal{J}^{in})] = \hat{\tau}(f, \Pi_0, \mathcal{J}^{in}) = \frac{1}{k}\sum_{i=1}^{k} W_i(\mathcal{J}^{in}) t^f_{\mathcal{J}_i^{in}, \Pi} \quad (1)$$

where $\hat{\tau}(f, \Pi_0, \mathcal{J}^{in})$ is a weighted average execution time estimation for the function f with the $\mathcal{J}^{in}$ input parameters on the basis of k previous observations of the execution time $t^f_{\mathcal{J}_i^{in}, \Pi_0}$ of the function f with the input parameters $\mathcal{J}_i^{in}$ on a machine with performance characteristics vector Π_0. The weighting function $W_i(\mathcal{J}^{in})$ assigns greater weight to the function execution time records, where the values of the input parameters are closer to the $\mathcal{J}^{in}$.

To take into account a possibility of execution of the function f on the virtual machine with the characteristics vector that is different from Π_0, we can extend the definition of the evaluation parameter vector, adding to the vector $\mathcal{J}^{in}$ of the input parameters the virtual machine performance vector $\Pi = [\pi_1, \pi_2 \dots \pi_r]$. Thus, we assume that as a characteristic we use the vector P of dimension $n + r$, where n - number of input parameters of the function f, r - the number of virtual machine performance characteristics, defined as follows:

$$P = [I_1^{in}, \dots, I_n^{in}, \pi_1, \dots, \pi_r].$$

In this case, (1) can be transformed into the following form:

$$\hat{E}[\chi(f, \Pi, \mathcal{J}^{in})] = \hat{\tau}(f, \Pi, \mathcal{J}^{in}) = \frac{1}{k}\sum_{i=1}^{k} W_i(P) t^f_{P_i},$$

where $t^f_{P_i}$ is the value of the previous observation of execution time of the function f with the evaluation parameter vector P_i.

Similarly, we can estimate the amount of output data for the function f:

$$\hat{v}(f, \mathcal{J}^{in}) = \frac{1}{k}\sum_{i=1}^{k} W_i(\mathcal{J}^{in}) v^f_{\mathcal{J}_i^{in}},$$

where $\hat{v}(f, \mathcal{J}^{in})$ is an estimate value of the volume of output parameters of the function f on the basis of information on output volumes of k previous runs of the same function with the $\mathcal{J}_i^{in}$ input parameters, using the weighting function W.

IV. HEFT ALGORITHM FOR THE PROBLEM-ORIENTED SCHEDULING

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 [17].

Let $|T_x| -$ be the size of the problem T_x, and the R be the set of computing resources with an average processing power $|R| = \sum_{i=1}^{n} |R_i| / n$. Then, the average time to complete the task with all available resources is calculated as

$$E(T_x) = \frac{|T_x|}{|R|} \quad (2)$$

Let $\overline{T}_{xy}$ be the amount of data transferred between tasks T_x and T_y, and R be the set of available resources with an average capacity of data transfer $\overline{R} = \sum_{i=1}^{n} \overline{R}_i / n$. Then the average score on data transfer costs between tasks T_x and T_y for all pairs of p.

$$D(T_{xy}) = \frac{\overline{T}_{xy}}{\overline{R}} \quad (3)$$

Thus, the priority calculation unit may be defined as

$$rank(T_x) = E(T_x) + \\ + \max_{T_y \in succ(T_x)} (D(T_{xy}) + rank(T_y)) \quad (4)$$

where $succ(T_x)$ is the set of tasks that depend on the task T_x.

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 [18].

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

Let F be the set of all functions that can be implemented in the subject area. Then a separate problem T_x is a function $f_x \in F$ with a set of input data objects $\mathcal{J}^{in} = (I_1^{in}, \dots, I_n^{in})$:

$$T_x = f_x(\mathcal{J}_x^{in}),$$

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

$$|R| = \sum_{i=1}^{n} \frac{|R_i|}{n} = \sum_{i=1}^{n} \frac{|\Pi_i|}{n}.$$

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

$$E(T_x) = \hat{\tau}(f_x, |R|, \mathcal{J}_x^{in}) \quad (5)$$

where $\hat{\tau}(f_x, |R|, \mathcal{J}_x^{in})$ is an average execution time estimation for the function f_x on a set of machines with mean production capacity $|R|$ with the set of known values of input parameters $\mathcal{J}_x^{in}$.

The model of problem-oriented services should take into account the amount of data returned by each task T_x. This may be used by the operator of the expected output $v(f, \mathcal{J}^{in})$, which returns the expected total size in bytes of output data objects $\mathcal{J}^{out}$. Consequently, within the framework of problem-oriented model for the evaluation of

data transmission time between two tasks the following estimation can be used:

$$D(T_{xy}) = \hat{v}(f_x, J_x^{in}) * \overline{R}_{xy}, \quad (6)$$

where $\overline{R}_{xy}$ 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) $R = 0$, when the data transmission channel consists of a single node;
2) $R = \beta_{group}$, when the data transmission channel is shared by a group of nodes;
3) $R = \beta_{cluster}$,, when the data transmission channel is shared by a cluster of compute nodes.

Figure 1 shows the pseudo-code for algorithm of problem-oriented workflow 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 (5)
       for each e E from task graph G
           Approximate data transfer time
           according to (6)
       Start the width-first search in
reverse task order and calculate a rank
for each task according to (4)
       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
```

Fig. 1. Problem-oriented heuristic scheduling algorithm PO-HEFT

V. Algorithm Implementation and Performance Evaluation

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

The algorithm was implemented as a number of Java classes so that Workflow Sim 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 and Workflow Sim's terminology) in a single VM.

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 Workflow Sim 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. We have also tested several algorithms such as plain HEFT, particle swarm optimization and genetic algorithm and this will be a topic for further research.

We plan to implement the developed cloud system and model DAG, POS and Min-min algorithms behavior in order to assess their efficiency.

VI. Conlusion

In this article, we assessed current scheduling algorithms and defined a model that allows to evaluate various cloud computing environment metrics for problem-oriented scheduling. Next, we described the PO-HEFT scheduling algorithm, which aims to provide workflow scheduling in heterogeneous 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 with help of Workflow Sim extension 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. We plan to extend the algorithm in order to schedule not only tasks on machines but also to schedule machine provisioning on virtual nodes.

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].

[17] H. Topcuoglu, S. Hariri, and I. C. Society, "Performance-Effective and Low-Complexity," *Parallel Distrib. Syst. IEEE Trans.*, vol. 13, no. 3, pp. 260–274, 2002.

[18] S. Chen, Y. Wang, and M. Pedram, "Concurrent placement, capacity provisioning, and request flow control for a distributed cloud infrastructure," p. 279, 2014.