Scheduling in CloudSim of Interdependent Tasks for SLA Design

Bogdan ȚIGĂNOAIA[1], George IORDACHE[1]*, Cătălin NEGRU[1], Florin POP[1,2]

[1] University "Politehnica" of Bucharest, Computer Science Department, Splaiul Independentei, No. 313, Bucharest-Sector 6, Bucharest, 60042, Romania
bogdantiganoaia@gmail.com, george.iordache@cs.pub.ro (**Corresponding author*), catalin.negru@cs.pub.ro

[2] National Institute for Research and Development in Informatics (ICI), Bucharest, Romania
florin.pop@cs.pub.ro

Abstract: One of the most recent concepts in the framework of today's distributed systems is Cloud Computing. A very difficult problem that needs to be addressed is the management of the Cloud. When designing a Cloud scheduling strategy, the design trade-offs of the Cloud architecture should be evaluated. The easiest way to evaluate this infrastructure is to use a simulation tool (in this case CloudSim simulation toolkit). This article examines three different algorithms that consider the scheduling of tasks in Cloud, which are described as Directed Acyclic Graphs (DAGs) (interdependent tasks). The results of the scheduling algorithms are valuable because they are helpful for the design of the Cloud Computing scheduling algorithms and for the design of a Service Level Agreement (SLA) contract whose members are the Cloud Service Provider (CSP) and the Cloud Service Customers (CSCs). This paper shows that the selection of the scheduling algorithms is useful in the design of the SLA contract.

Keywords: Scheduling in Cloud, CloudSim, SLA parameters, Directed Acyclic Graphs.

1. Introduction

The Cloud Computing paradigm is considered as one of the main techniques to operate with data using a network of computers that is available through Internet by paying a certain price. This paradigm is based on the interaction between CSCs that rent the Cloud Computing Service from a CSP. In this type of environment one of the most important operations is to schedule the tasks from one or various CSC to the Cloud Computing resources that are offered by the CSPs. The process of task scheduling is crucial when discussing about this type of paradigm because based on the scheduling algorithm the CSC can decide which CSP is the most suitable for scheduling its DAG of tasks.

Three different scheduling algorithms were modelled and simulated in order to design a SLA contract for the Cloud. One used a very well-known Cloud Computing paradigm simulator that is called CloudSim (Calheiros et *al.*, 2011). The reason behind the choice of CloudSim was the fact that this tool is considered a well-established modeling and simulation software for the Cloud. The CloudSim is a simulation toolkit that can help design, evaluate and implement various Cloud architectures and various scheduling algorithms. The process of experimenting in Cloud is very difficult and involves high costs when dealing with real environments. Thus, the choice of simulating the algorithms in CloudSim is well-motivated.

This paper elaborates on the scheduling of tasks with dependencies on a Cloud Computing platform. For this purpose, three scheduling algorithms (that are some of the most usual when working with DAGs) are considered and the differences between them are analyzed. These differences are useful when trying to optimize certain SLA parameters. By optimizing the SLA parameters, one refers to the minimization of the values of these parameters in the negotiation phase (between the CSC and the CSP) when knowing the DAG of tasks of the CSC. These SLA parameters can be enumerated as follows:

- **Total time of execution** (Iordache et *al.*, 2017): although this parameter is hard to minimize because the actual execution time on a certain resource it is usually unknown, in this paper it is considered that this parameter can be reduced because the DAG of the CSC is already known. This parameter is useful when designing the CSP infrastructure because based on it one can decide for example the number of Virtual Machines that are offered by the provider to the customer.

- **Total Execution Time of a specific tenant**: because the execution times of tasks on Virtual Machines and the communication costs between the Virtual Machines of the Cloud are known, the Cloud infrastructure can be designed so that it can minimize the

https://doi.org/10.24846/v28i4y201911

total execution time of a specific tenant.; thus this parameter is useful from the user (tenant) point of view.

- **Percentage of Disponible Resources (PDR)**: this parameter is used by the CSP in the scheduling process of certain tasks on certain Virtual Machines in the Cloud.

The process of scheduling can be employed when designing a Cloud Computing System, by using the simulation results that come from the CloudSim simulation toolkit. The main idea behind this design process is the fact that one must decide the number of available Cloud Computing resources and the cost that is needed to maintain those resources. Thus, it can be said that having many resources is useful, but this large number is not always necessary. The second situation, which is the implementation of different scheduling strategies on existing Cloud architectures, is very important when dealing with CSCs that want to optimize the SLA parameters of a given Cloud Infrastructure. For this usually a SLA between the CSPs and CSCs must be in place. The design of such a SLA contract can be based also on the scheduling algorithms and results.

The process of scheduling is based on the idea of mapping tasks on the available resources (see Figure 1). There are various types of tasks. From the point of view of interaction between them there are (Hassan et *al.*, 2015), (Kwok, Ahmad & Gu, 1996):

- **Independent** (tasks that run independently from one another)
- **Interdependent** (tasks that have dependencies between them)

From the point of view of the time of execution, the tasks can be **heterogeneous** (with different execution times) or **homogeneous** (with the same execution times).

This paper will focus on heterogeneous offline (in this case the queue of tasks is known a priori) CPU and network communication intensive interdependent tasks.

One of the most important situations in Cloud is the way that the resource management is performed. We know already that the scheduling problem is NP-complete (Ullman, 1975). This means that the matching between the tasks and the resources can be optimized, but it is not certain that an optimal one can be reached. Thus, in scheduling the purpose is finding a "good enough" solution in a short enough time. For this to be feasible the Cloud designer must take proper decisions.

The paper is organized as follows. Section 2 discusses the state of the art regarding scheduling in the Cloud domain. Section 3 presents the DAG model used. Section 4 sets forth the simulation toolkit – CloudSim. Section 5 explores the list scheduling strategies and metrics. In section 6 the implemented algorithms are considered. Section 7 presents the DAG scheduling results and how these algorithms can be used in designing the SLA for the Cloud. In section 8 the conclusions are drawn, and the future work is presented.

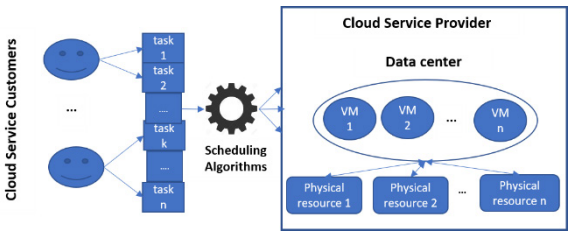

Figure 1. Scheduling in Cloud – a schema.

2. State of the Art

This paper discusses the simulated scheduling in a homogeneous Cloud because the homogeneous Cloud is encountered in practice (the CSPs usually provide the same types of resources) and the scheduling in such environments represents a real research challenge (Singh & Chana, 2016).

DAG-based scheduling strategies in homogeneous and heterogeneous Cloud have been presented in multiple works, (Singh & Chana, 2016; Wang, et *al.*, 2012; Mohammed & ŢĂPUŞ, 2017). Various scheduling strategies with various optimization parameters are discussed by (Zhan et *al.*, 2015), for example load balancing scheduling, energy conservation scheduling, and cost effectiveness scheduling. The scheduling strategies can be offline or online, and the experimental environments can be real clouds or numerical simulators.

In (Emeakaroha et al., 2015) the scheduling process in Cloud by considering multiple SLA objectives. In (Reig, Alonso & Guitart, 2010) a novel mechanism is proposed to prevent SLA violations.

Three scheduling algorithms were implemented in CloudSim with the purpose of choosing (the

chosen strategy is explicitly discussed) the best scheduling strategy when a certain SLA parameter needs to be optimized.

The novelty of this work lies in the fact that it starts from the scheduling strategy and analyses which SLA parameters can be optimized based on the implemented scheduling strategy.

3. The DAG Model

In Cloud scheduling a set of tasks that interact with each other can be represented as a DAG characterized by:

V – the set of nodes (vertexes – tasks), where a node represents an application task. Each application task is defined as a set of instructions that can be executed on a processor, and a set of nodes (vertexes – tasks) can be executed on multiple processors. There can exist two types of nodes: a parent node which is the source node of an edge and a child node which is the destination node of an edge. A node without parents is said to be at an entry level and a node without children is said to be at an exit level. The set of nodes can be used when computing the SLA parameter called **Number and type of virtual machines** (Iordache et *al.*, 2017). For example, based on the V of the DAG and on the scheduling policy one can decide which **number and type of virtual machines** of the CSPs can be rented for scheduling that DAG.

E – the set of oriented edges that defines the nodes' dependencies. An edge, e(i, j), denotes the communication costs between a parent node (task) v(i) and a v(j) node (task) that is denoted as a child node. Additionally, it must be said that when dealing with a DAG a child node, v(j) cannot be executed until its parent node, v(i), is executed and until it communicates the data resulted from its execution to the child node, v(j). **E** can be used to represent the communication costs of the network that is in place between the computational nodes of the Cloud.

C – the set of communication-related costs, meaning that each edge e(i, j) has an associated communication cost, c(i, j). The communication costs can be used when designing the Cloud based on the SLA parameter **Network capacity**. Starting from the DAG of the CSC and based on the upper bounds of the network (the **Network capacity** parameter) the communication costs of the set C can be recomputed.

W – the set of computation costs (or computation weights), where an element of the set, for example the node (task) v(i), has the computation weight w(i).

It can be said that a DAG has some entry nodes, some intermediate nodes and some exit nodes.

Based on the W set the CSP infrastructure configuration can be designed. When doing this, the SLA parameter called **CPU capacity is** of interest. For example, W is computed based on the **CPU capacity** SLA parameter of each virtual machine from the Cloud (e.g. as the time necessary for the execution of a certain task on a certain resource).

The DAG is used because the scheduling of tasks in the Cloud when having a CSC's DAG can be optimized. This optimization can be used when defining the SLA parameters agreed with the CSP. This will be further discussed in section 7.

4. CloudSim Toolkit

CloudSim (Calheiros et *al.*, 2011; Buyya, Ranjan & Calheiros, 2009) is a well-known simulation tool that was designed to evaluate the performance of different Cloud infrastructures and model the interactions between the CSPs and the CSCs. The CloudSim implementation classes model Cloud Components such as: datacenters (Datacenter), hosts (Host), virtual machines (Vm), processing elements (Pe) and tasks (Cloudlets) (see Fig. 2).

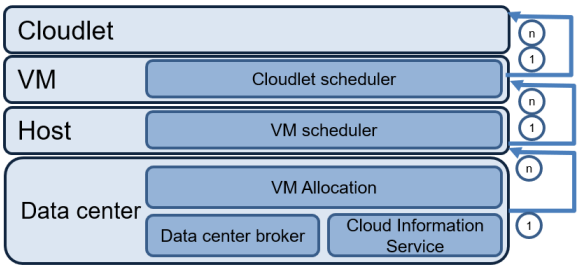

Figure 2. Basic Cloud infrastructure (Calheiros et *al.*, 2011)

As shown in Figure. 2, the main entities of a Cloud architecture implemented in CloudSim are:

1. CIS (Cloud Information Service) functional part registers Data Center entities and ensures the correspondence between the Cloud Service Customers and the suitable Cloud Service Providers.

2. Data Center represents a model of hardware resources that are offered by the Cloud Service Providers and that are either homogeneous or heterogeneous.

3. Data Center Broker is an entity that has the purpose of selecting a Data Center by analyzing its available virtual machines and the Cloudlets (tasks) that need to be allocated to the virtual machines and implements the execution of tasks on the selected Data Center.

4. Host represents a physical computing machine. A Data Center can have one or many Hosts.

5. VM represents a Virtual Machine. A Host can have one or multiple Virtual Machines.

6. Cloudlet represents a Task that needs to be scheduled.

The simulator has some specific properties:

- a task can be executed on a single processing element (Pe);

- a processor cannot use preemption; this means that before executing the next task on the current Pe, the current task must be executed on the current Pe;

- a Pe can execute the tasks that are mapped to it;

- for tasks with dependencies a task can be executed only after the execution of all its parents;

- if a task is mapped on a Pe, the task is ready to run and the Pe is available, the task will run on that Pe.

The aim of this research is to optimize the scheduling process based on SLA constraints in a Cloud environment by using simulations. The authors chose to simulate the scheduling process because it offers different configuration capabilities at lower costs.

5. List Scheduling

The list scheduling technique is based on the idea that the tasks that must be scheduled are assigned certain priorities and they are ordered into lists (or queues) into the descending order of their priority. The scheduling algorithms include the following steps:

1. If a task from the given list has the highest priority, then it will be selected for scheduling;

2. Based on the selected task a mapping between the task and a resource is done;

3. If no resource is found for the mapping between the task and the resource (Step 2.) continue with the next task from the list.

Priorities are assigned to the tasks based on some metrics of DAG such as (Kwok & Ahmad, 1999):

- **Top level (t-level)** of a node v(i) represents the length of the longest path whose start node is an entry node and whose end node is the node v(i). This length is given by the sum of all computation costs (represented by the costs of nodes) and communication costs (represented by the costs of edges) of all nodes along the top level (t-level) path. For the computation of the t-level the DAG is traversed downwards starting from its entry nodes.

The formula for the computation of t-level of node v(i) is the following:

$$t-level(v(i)) = \max(t-level(v(m)) + w(i) + c(m,i)). \quad (1)$$

• **Bottom level (b-level)** of a node v(i) represents the length of the longest path whose start node is the node v(i) and whose end node is an exit node. Its length is given by the sum of all computation costs (the costs of nodes) and communication costs (the costs of edges) of all nodes along the bottom level (b-level) path. For the computation of the b-level the DAG should be traversed upwards by starting from its exit nodes towards to the node v(i).

The b-level of v(i) can be computed as follows:

$$b-level(v(i)) = w(i) + \max(b-level(v(m)) + c(m,i)). \quad (2)$$

- **Static b-level (SBL)** is computed when the edge weights are not considered. The SBL of v(i) can be expressed as:

$$SBL(v(i)) = w(i) + \max(SBL(v(m))). \quad (3)$$

- **Critical path** is computed as the longest path that starts from an entry node and ends with an exit node.

- **ALAP (LST)** (Wu & Gajski, 1990): As late as possible start time of a given node measures how much the start time of a node can be delayed without affecting the global schedule length.

The formula which expresses the LST of v(i) is the following:

$$LST(v(i)) = \min(LST(v(m)) - c(m,i)) - w(i). \quad (4)$$

- **ASAP** (Wu & Gajski, 1990): – as soon as possible – refers to the earliest start time of a node (vertex) if by scheduling the node (task - vertex) the DAG scheduling constraints are not violated.

All these metrics are used by the considered list of scheduling algorithms for DAGs in CloudSim.

The next section puts forward three algorithms that were developed in order to analyze different SLA characteristics of the scheduling process in the Cloud when tasks are represented as DAGs.

6. Algorithms

The **Earliest Time First Algorithm (ETF)** (Hwang et *al.*,1989). is based on the idea that at each execution step all the earliest start times (EST) are computed for each task and the task with the smallest start time from the batch of tasks that need to be scheduled is chosen for scheduling. The EST of a node is computed by considering exhaustively the start time of each task (node). A list of the nodes that are ready to be evaluated is used. After selecting a node that is ready for evaluation all its successors will be included in the respective list.

The **Highest Level First with Estimated Times (HLFET)** (Adam, Chandy & Dickson, 1974) algorithm produces a schedule based on the priority of nodes (tasks) that is determined by computing its level. For the computation of this level the computation costs are added along a path from a given node (task) to an exit node and the largest sum is chosen. The highest priority of a node is denoted by the highest computed level. For ordering the nodes (tasks) a list with all the nodes of the DAG is sorted in descending order based on the SL of nodes (tasks).

Modified Critical Path (MCP) (Wu & Gajski, 1990) algorithm computes first an attribute called Latest Possible Start Time for each node. This attribute is computed by finding the ALAP metric of each node. For finding this metric one should traverse the graph starting from the exit nodes (tasks) towards the entry nodes (tasks) and to compute downwards where it is possible the start time of the nodes.

7. DAG Scheduling Example

In order to obtain the scheduling results, the DAG presented in Figure 3 is used. In Figure 3 the Critical Path is marked with grey and has a length of 46. In Table 1 the nodes are marked by an asterisk.

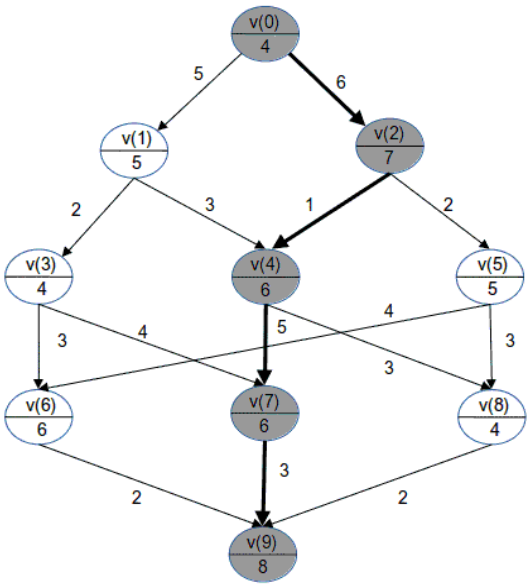

Figure 3. An example of DAG used to highlight the execution of the scheduling algorithm

Table 1 presents the values for the static b-levels (SBLs), t-levels, b-levels and ALAP of the nodes computed for the given graph.

Table 1. The metrics of the DAG graph: Static b-levels (SL), t-levels, b-levels, ALAP; the nodes on the Critical Path (CP) are marked with asterisk (*)

node	SL	t-level	b-level	ALAP
v(0)*	31	0	46	0
v(1)	25	9	36	10
v(2)*	27	10	36	10
v(3)	18	16	25	21
v(4)*	20	18	28	18
v(5)	19	19	25	21
v(6)	14	28	16	30
v(7)*	14	29	17	29
v(8)	12	27	14	32
*v(9)	8	38	8	38

The results of the scheduling algorithms implemented in CloudSim are presented in Tables 2, 3, 4. When designing the above-mentioned Cloud, 1 Data Center with 3 Virtual Machines was chosen. The purpose of the experimental setup was to compare the three algorithms that were implemented in CloudSim. The purpose of the experimental setup was to compare the three algorithms that were implemented in CloudSim.

To compare the results, the following optimization parameters were chosen (Pop, Dobre & Cristea, 2009; Zuo et al., 2015).

Table 2. The scheduling results for the DAG of tasks in the case of the ETF algorithm

Output Earliest Time First (ETF) Algorithm				
Cloudlet ID	**VM ID**	**Task Weigth**	**Start time**	**Finish Time**
0	0	4	0	4
1	1	5	4	14
2	2	7	4	17
3	1	4	14	18
4	2	6	17	23
5	1	4	18	22
6	2	6	23	29
7	1	6	22	28
8	2	4	29	33
9	1	8	28	36

Table 3. The scheduling results for the DAG of tasks in the case of the HLFET algorithm

Output of the HLFET Algorithm				
Cloudlet ID	**VM ID**	**Task Weigth**	**Start time**	**Finish Time**
0	0	4	0	4
1	1	5	9	14
2	0	7	4	11
3	2	4	16	20
4	1	6	14	20
5	0	5	11	16
6	0	6	16	22
7	1	6	20	26
8	2	4	23	27
9	1	8	26	34

Table 4. The scheduling results for the DAG of tasks in the case of the MCP algorithm

Output of the MCP Algorithm				
Cloudlet ID	**VM ID**	**Task Weigth**	**Start time**	**Finish Time**
0	0	4	0	4
1	0	5	4	9
2	0	7	9	16
3	1	4	11	15
4	0	6	16	22
5	1	5	18	23
6	0	6	22	28
7	1	6	23	29
8	2	4	26	30
9	2	8	30	38

1. *Makespan* = max(v(i)) ∈V {ft(v(i))}, where v(i) represents a node (vertex - task) from the set of nodes (vertexes - tasks) V; additionally, the *st(v(i))* is denoted as the start time for node (vertex - task) *v(i)*, and *ft(v(i))* as the finish time for node (task) *v(i)*. The makespan parameter is correlated with the SLA parameter of Total Time of Execution for the DAG that is sent to be scheduled. Based on the results obtained it can be noticed that the HLFET algorithm has the minimum makespan, followed by the ETF algorithm and the last one is the MCP the algorithm (see Table 5.).

Table 5. The makespan for the implemented algorithms: MCP, ETF, HLFET.

Makespan		
MCP	**ETF**	**HLFET**
38	36	34

The makespan parameter is useful when designing a Cloud Service Provider scheduling policy that has the purpose of minimizing the Total Time of Execution for a DAG of tasks. This SLA parameter can be optimized if we know the exact DAG that needs to be scheduled and the scheduling policies to be implemented.

A SLA between the CSP and the CSC might state the following: "The Total Execution Time in the case of the given DAG must have a finish time of at most 37" (Iordache et *al.*, 2017). A generic SLA rule of the form *"if-then"* can be the following:

"If the Total Execution Time in the case of the given DAG is greater that the agreed one (e.g. 37) then the CSP needs to pay a penalty of 10% for this violation" (Iordache et al., 2017).

One can see that if this rule is put in practice then the MCP scheduling algorithm should not be considered.

2. Total Execution Time of a specific tenant that represents how the scheduling process can be optimized when it relates to a specific tenant (a Cloud customer can have multiple tenants of the same CSP). For example, if one considers that in the DAG employed for designing the scheduling process in the Cloud the path v(0) – v(2) – v(5) – v(8) represents the tasks executed for a specific tenant (the tasks in the chosen DAG are from multiple tenants) then the Total Execution Time

of this tenant is 27 in the case of the HLFET algorithm, 33 in the case of the ETF algorithm and 30 in the case of MCP algorithm (see Table 6).

Table 6. The makespan for the three implemented algorithms in case of a given tenant

Total execution time for a specific tenant							
Cloudlet ID	Task Weigth	HLFET		ETF		MCP	
		st	ft	st	ft	st	ft
0	4	0	4	0	4	0	4
2	4	4	11	4	17	9	16
5	5	11	16	18	22	18	23
8	4	23	27	29	33	26	30

A SLA between the CSP and the CSC might point out about the Total Execution Time for that tenant (the one in Table 6): "The Total Execution Time in the case of the given DAG for the given tenant must have a finish time of at most 29". A generic SLA rule of the form "if-then" can be the following:

"If the Total Execution Time for the given tenant in the case of the given DAG is greater that the agreed one (e.g. 29) then the CSP needs to pay a penalty of 5% for this violation"

If this rule is put in practice, then only the HLFET algorithm should be considered. The SLA rules can be defined while negotiating the SLA contract. The negotiation of the SLA contract is based on what the Cloud Service Provider can offer in terms of implemented scheduling algorithms.

3. Another SLA parameter that can be optimized is the **Percentage of Disponible Resources (PDR)** when discussing about multi-tenancy in Cloud. If one considers for example that in the aforementioned DAG v(3), v(4), and v(5), are independent tasks, that are part of different Cloud applications, it can be noticed that for example for the task v(3) the only available machine in order to minimize the makespan is the Virtual Machine VM 2, in the case of the HLFET algorithm, it is the Virtual Machine VM 1 in the case of the ETF algorithm, and the Virtual Machine VM 1 in the case of the MCP algorithm. A generic SLA rule in the form *"if-then"* for the Percentage of Disponible Resources (PDR) parameter can be the following:

"If the Percentage of Disponible Resources in the case of the given DAG is smaller than the agreed one then the CSP needs to pay a penalty of 5% for this violation"

and the rule is useful when multiple tenants use the same Cloud architecture and the Percentage of Disponible Resources is used in the scheduling process).

The SLA parameters that were discussed in this work are used when designing a SLA contract if one knows the scheduling DAG and the scheduling strategy. Based on the parameters that need to be optimized the SLA can consider a given scheduling strategy (from among the three ones presented previously or from among others that will be implemented in the future).

8. Conclusions and Future Work

This paper discusses the importance of scheduling algorithms in order to help designing a proper SLA contract, which may prove to be advantageous for both the CSP and for the CSC. One discusses certain SLA parameters that need to be taken into consideration when implementing the scheduling algorithms, then analyze and compare the three algorithms that are employed for scheduling tasks in the CloudSim environment.

As future work it is planned to implement the scheduling algorithms in a real Cloud System (for example Cloud robotics (Wan et al., 2016)) and analyze and design in a better way both the interaction between the CSCs and the CSPs and the scheduling process. In addition, one could analyze and implement other algorithms from literature (for example those described by (Zuo et al., 2015), that are meant to optimize the SLA-based scheduling process. Another aim is to use the scheduling platform together with other technologies (Merezeanu, Vasilescu, & Dobrescu, (2016).

Acknowledgement

This work has been funded by University Politehnica of Bucharest, through the "ARUT Grants" Program, UPB – GNaC. Identifier: GNaC 2018, Contract: 16/06.02.2019, RM-CYBERSEC.

REFERENCES

1. Adam, T. L., Chandy, K. M. & Dickson, J. R. (1974). A comparison of list schedules for parallel processing systems, *Communications of the ACM*, *17*(12), 685-690.

2. Buyya, R., Ranjan, R. & Calheiros, R. N. (2009, June). Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *International Conference on High Performance Computing & Simulation, - HPCS'09* (pp. 1-11). IEEE.

3. Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, *Software: Practice and experience*, *41*(1), 23-50.

4. Emeakaroha, V. C., Brandic, I., Maurer, M., & Breskovic, I. (2011). SLA-aware application deployment and resource allocation in clouds. In *2011 IEEE 35th Annual Computer Software and Applications Conference Workshops* (pp. 298-303). IEEE.

5. Hwang, J. J., Chow, Y. C., Anger, F. D. & Lee, C. Y. (1989). Scheduling precedence graphs in systems with interprocessor communication times, *SIAM Journal on Computing, 18*(2), 244-257.

6. Iordache, G., Paschke, A., Mocanu, M. & Negru, C. (2017). Service Level Agreement Characteristics of Monitoring Wireless Sensor Networks for Water Resource Management (SLAs4Water), *Studies in Informatics and Control, 26*(4), 379-386. DOI: 10.24846/v26i4y201701

7. Kwok, Y. K., Ahmad, I. & Gu, J. (1996). FAST: A low-complexity algorithm for efficient scheduling of DAGs on parallel processors. In *Proceedings of the 1996 ICPP Workshop on Challenges for Parallel Processing* (pp. 150-157). IEEE.

8. Kwok, Y. K. & Ahmad, I. (1999). Static scheduling algorithms for allocating directed task graphs to multiprocessors, *ACM Computing Surveys (CSUR), 31*(4), 406-471.

9. Merezeanu, D., Vasilescu, G. & Dobrescu, R. (2016). Context-aware control platform for sensor network integration in IoT and Cloud, *Studies in Informatics and Control, 25*(4), 489-498. DOI: 10.24846/v25i4y201610

10. Mohammed, M. A. & Ţăpuş, N. (2017). A novel approach of reducing energy consumption by utilizing enthalpy in mobile cloud computing, *Studies in Informatics and Control, 26*(4), 425-434. DOI: /10.24846/v26i4y201706

11. Pop, F., Dobre, C. & Cristea, V. (2009). Genetic algorithm for DAG scheduling in grid environments. In *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing* (pp. 299-305). IEEE.

12. Reig, G., Alonso, J. & Guitart, J. (2010). "Prediction of job resource requirements for deadline schedulers to manage high-level SLAs on the cloud." In *2010 Ninth IEEE International Symposium on Network Computing and Applications* (pp. 162-167). IEEE.

13. Singh, S. & Chana, I. (2016). A survey on resource scheduling in cloud computing: Issues and challenges, *Journal of Grid Computing, 14*(2), 217-264.

14. Ullman, J. D. (1975). NP-complete scheduling problems, *Journal of Computer and System Sciences, 10*(3), 384-393.

15. Wan, J., Tang, S., Yan, H., Li, D., Wang, S., & Vasilakos, A. V. (2016). Cloud robotics: Current status and open issues. *IEEE Access, 4,* 2797-2807.

16. Wang, W., Zeng, G., Tang, D. & Yao, J. (2012). Cloud-DLS: Dynamic trusted scheduling for Cloud computing, *Expert Systems with Applications, 39*(3), 2321-2329.

17. Wu, M. Y. & Gajski, D. D. (1990). Hypertool: A Programming Aid for Message-Passing Systems, *IEEE Transactions on Parallel and Distributed Systems, 1*(3), 330-343.

18. Zhan, Z. H., Liu, X. F., Gong, Y. J., Zhang, J., Chung, H. S. H. & Li, Y. (2015). Cloud computing resource scheduling and a survey of its evolutionary approaches, *ACM Computing Surveys, 47*(4), 1-33.

19. Zuo, L., Shu, L., Dong, S., Zhu, C. & Hara, T. (2015). A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing, *IEEE Access, 3,* 2687-2699.