International Journal of MC Square Scientific Research Vol.3, No.1 Nov 2011

# AN EFFECTIVE METHOD FOR QOS -CONSTRAINED WORK FLOW SCHEDULING OF CLOUD SERVICES

*Sabarinathan.N*
*Computer science Engineering,*
*Sri Jayaram Engineering College,*
*Anan University, Chennai, Tamilnadu, India*
*sabari.studies@gmail.com*

**ABSTRACT -** Cloud computing is the most recent promising trending that provides hardware infrastructures and software applications as a service. Users can use these services through Service Level Agreement (SLA) which defines user's essential Quality of Service (QoS) parameters on pay-per-use basis. In cloud computing the workflow scheduling is a difficult problem to be solved. Normally the scheduling methods are tried to diminish the execution time of the workflows. There are several existing approaches to solve the difficulty of multi-objective scheduling in cloud but, there exists the problem of computational complexity and the budget constraints. To overcome this problem, in existing the SaaS Cloud Partial Critical Paths (SC-PCP) algorithm was enhanced, which is an extension of the preceding one for the SaaS Clouds. The idea of the SC-PCP algorithm is to form a schedule that decreases the total execution cost of a workflow, while satisfying a user defined deadline for the total execution time. In this work the main problem is that the time and cost are the only considered as parameters for the deadline. To overcome this problem, in proposed work, there are three proposals are followed. The first proposal is; QoS needed by the customers for selecting a Cloud service provider is based on: Accountability, Agility, Assurance of Service, Security and Privacy, and Usability where the drawback of existing method is solved. To solve this problem as a second proposal work, a (Price- and-Time-Slot-Negotiation) PTN mechanism devised that enables both providers and customers to do the following: 1) specify their preferences for price and time slot and 2) search for mutually acceptable prices and time slots. Finally, quantifying the performance of scheduling and allocation policy on a Cloud infrastructure (hardware, software, services) for different application and service models below unreliable load energy performance (power consumption, heat dissipation), and system dimension is an extremely challenging problem to tackle. To overcome this problem, the SC-PCP algorithm is expanded to support other IaaS Cloud computing model. This final proposal work can be implemented with the use of cloud sim.

**Keywords**: Cloud computing; SaaS Clouds; Grid computing; Workflow scheduling; QoS-based scheduling

## 1. INTRODUCTION

Recently many researchers contain considered the benefits of by Cloud computing for systematic applications. Currently, these services are categorized into three major classes: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). IaaS Clouds like Amazon, provide virtualized hardware and storage on top of which the users can deploy their own applications and services. PaaS Clouds, like Microsoft Azure [17], provide an application development environment in which the users can implement and run applications on the Cloud. According to cloud [17], there are two types of Cloud [17], which deliver software applications to the users. Examples of these Clouds are Google office automation services like Google Document or Google Calendar. The second group [17] provides rudimentary web services to the users (known as on-demand web services).

Workflow scheduling is the problem of mapping each task to a suitable resource and of ordering the tasks on each resource to satisfy some performance criterion [16], [17]. As task scheduling is a well-known NP-complete problem [16], many heuristic methods have been proposed for homogeneous and heterogeneous distributed systems like Grids. These scheduling methods [16], [17] try to minimize the execution time (makes pan) of the workflows [16] and, as such, are suitable for community Grids.

Most current workflow management systems [17], like the ones above mentioned, use such scheduling methods [16]. However, in Clouds, there are many other potential QoS attributes [16], [17] like execution time, like reliability, security, availability [17] and so on. Besides, stricter QoS attributes mean higher prices for services [17]. Therefore, the scheduler faces a QoS cost tradeoff in selecting appropriate services [16], which belongs to the multi-objective optimization problems family. There are several existing approaches [16], [17] to the problem of multi-objective scheduling. One method is to find pare to desirable solutions, and let the user select the best schedule according to his requirements. The problem is that the pare to sets are usually very large and hard to examine [17].

Another common method is to assign a weight to each scheduling criterion [15], [16], and optimize the weighted sum of these criteria. However, in most cases, the weight assignment is not a simple process for users. Due to the complexity of the development of a general multi-objective scheduling algorithm [17], many researchers [15], [16] try to propose bi-criteria scheduling algorithms [17]. In most bi-criteria scheduling algorithms [17], the user specifies a limitation for one criterion (deadline or budget constraints), and the algorithm tries to increase the computing speed under this constraint. This is a easiest way for the users to convey their ideas, and a useful scenario for the researchers to easy to solve the problem and propose fast and high performance solutions.

## 2. RELATED WORK

In previous work [15], [16], they have proposed a QoS-based workflow scheduling algorithm on utility Grids, called the Partial Critical Paths (PCP). Also they have proposed the SaaS Cloud Partial

Critical Paths (SC-PCP) algorithm, which is an extension of the preceding one for the SaaS Clouds. The objective function of the SC-PCP algorithm is to create a schedule that reduce the total implementation cost of a workflow as satisfying a user defined deadline for the total execution time. First, the SC-PCP algorithm tries to schedule the (overall) critical path of the workflow, such that it is finished previous to the user deadline, and execution cost is minimized. Then, it finds the partial critical path to each scheduled task on the critical path and executes the same procedure in a recursive manner.

The existing algorithm was based on a similar heuristic, to schedule the critical nodes first, yet not to minimize the execution time, but to reduce the price of executing the critical path before the user-specified deadline. After scheduling all critical nodes, each of them has a start time which is a deadline for its parent nodes, i.e. its (direct) predecessors in the workflow. So, then they can carry out the same procedure by considering each critical node in turn as an exit node with its start time as a deadline, and creating a partial serious path that ends in the critical node and that leads back to an already scheduled node. In the SaaS Cloud-Partial Critical Paths (SC-PCP) algorithm, this process continues recursively until all tasks be successfully scheduled.

The existing algorithm has following problems such as (i) They have considered only time and cost as parameters for the deadline, (ii) The QOS constraints should be improved, (iii) Price and time slot have to be negotiated simultaneously and (iv) Need of new generalized and extensible simulation framework that should support for modeling and instantiation of large scale Cloud computing infrastructure.

## 3. SCHEDULING SYSTEM MODEL

The planned scheduling system model consists of an application model, a Cloud model, with a performance for scheduling [17]. An application is viewed by a directed acyclic graph w (T, E), where T is a set of n tasks {t1, t2. . . tn}, and E is a set of dependencies. Each dependency, ei,j = (ti , tj ), represents a precedence constraint, which indicates that task ti should complete execution before task tj can start. In a given task graph, a task without any parent is called an entry task, and a task without any child is called an exit task. As our algorithm requires a single entry and a single exit task, we always add two dummy tasks tentry and texit, to the beginning and end of the workflow, respectively. These dummy tasks contain zero implementation time and are connected with zero-weight dependency to the real entry and exit tasks.

### 3.1 The SC-PCP scheduling algorithm

Algorithm 1 shows the pseudo-code of the overall SC-PCP algorithm for scheduling a workflow [16],[17]. After some initialization, the algorithm generates the fastest schedule for the input workflow in line 4. This is a preliminary schedule, which obviously has the highest cost. In the next phases, the algorithm tries to refine this preliminary schedule [14] by changing the selected service of each task, such that the overall execution time extends to the user‘s deadline, and the cost decreases as much as possible. We define a scheduled task as a task whose selected service is finalized and which never changes in the next phases of the algorithm. Obviously, all tasks are still unscheduled in the preliminary schedule.

### 3.2 The parents scheduling algorithm

The pseudo-code for Schedule Parents is shown in Algorithm 2. This algorithm 2 receives a scheduled service as input and schedules all its unscheduled parents before the start time of the input node itself (the while loop from line 2 to 14). First, Schedule Parents try to find the Partial Critical Path of unscheduled nodes ending at its input node and starting at one of its predecessors that has no unscheduled close relative. For this cause, it uses the concept of Critical Parent.

Definition 1. The Critical Parent of node ti is the unscheduled parent of ti that has the latest data arrival time at it; that is, it is the parent tp of it , for which

EST (tp) + ET (tp, SS (tp)) + TT (ep, i) is maximal.

We will now define the fundamental concept of the SC-PCP algorithm.

Definition 2. The Partial Critical Path of node ti is: (i) empty if it does not have any unscheduled parents. (ii) Consists of critical parent tp of ti and the Partial Critical Path of tp if has any unscheduled parents.

Algorithm 2 begins with the input node and follows the critical parents until it reaches a node that has no unscheduled close relative to form a partial serious path (lines 3–7). Note that in the first name of this algorithm, it begins with texit and follows back the critical parents until it reaches tentry, and so it finds the overall real critical path of the complete workflow graph.

```
1:procedure ScheduleWorkflow(w(T, E), D)
2:     determine available services S_i for each task type in w
3:     add t_entry, t_exit and their corresponding dependencies to w
4:     generate the fastest schedule
5:     compute EST(t_i) for each task in w according to Eq. (2)
6:     compute LFT(t_i) for each task in w according to Eq. (4)
7:     if (the fastest schedule is not feasible) then
8:         return(null)
9:     end if
10:    mark t_entry and t_exit as scheduled
11:    call ScheduleParents(t_exit)
12:end procedure
```

**Algorithm 1.** The SC-PCP scheduling algorithm.

```
1: procedure ScheduleParents(t)
2:     while(t has an unscheduled parent) do
3:       PCP ← null, t_i ← t
4:       while (there exists an unscheduled parent of t_i) do
5:         add CriticalParent(t_i) to the beginning of PCP
6:         t_i ← CriticalParent(t_i)
7:       end while
8:       call SchedulePath(PCP)
9:       for all (t_i ∈ PCP) do
10:          update EST for all successors of t_i
11:          update LFT for all predecessors of t_i
12:          call ScheduleParents(t_i)
13:      end for
14:    end while
15: end procedure
```

**Algorithm 2.** Parents scheduling algorithm.

Then, the algorithm calls procedure Schedule Path (line 8), which receives a path (an ordered list of nodes) as input [17], and schedules each node on the path [16], such that it can complete previous to its most recent finish time and the total implementation cost of the path is minimized. We complicated on this procedure in the next sub-section. As the Schedule Path probably changes the selected services of some responsibilities on the path, the ESTs of their successor and the LFTs of their predecessor may change (according to Corollary 3). For this reason, the algorithm updates these values for all responsibilities of the path in the next loop. After that the algorithm starts to list the parents of each node on the partial critical path, from the beginning to the end of the path, by calling Schedule Parents recursively.

### 2.3 The path scheduling algorithm

The Schedule Path algorithm receives a path as input and tries to find a schedule for its tasks that minimizes the total cost of the path and finishes each task before its latest finishes time. We propose three different policies for scheduling a path as follows. We try to find the cheapest schedule that can finish the tasks of the path before their latest finish time.

```
1: procedure SchedulePath(path)
2:     for all tasks t_k ∈ path do
3:         for d = EST(t_1) to EFT(t_k) − 1 do
4:             C[k, d] = ∞
5:         end for
6:         for t = EFT(t_k) to LFT(t_k) do
7:             if (t_k is the first task on the path) then
8:                 compute c[1, t] according to Eq. (5)
9:             else
10:                compute c[k, t] according to Eq. (6)
11:            end if
12:        end for
13:        for t = LFT(t_k) + 1 to LFT(t_l) do
14:            c[k, t] = ∞
15:        end for
16:    end for
17:    find the optimal schedule based on c[l, LFT(l)]
18:    set EST, LFT and SS based on the optimal schedule
19:    mark all tasks of the path as scheduled
20: end procedure
```

**Algorithm 3.** Optimized path scheduling algorithm.

Since the problem of finding the optimal schedule for an ordered list of tasks, or, more precisely, a linear workflow is also an NP-complete problem, there is no polynomial time algorithm [17] to solve it. Fortunately, this problem can be formulated as an extension of a classic problem, known as the Multiple Choice Knapsack Problem (MCKP) [16].

```
1:procedure SchedulePath(path)
2:     repeat
3:         for all (t_i ∈ Path) do
4:             if (scheduling t_i on the next service is feasible) then
5:                 SS(t_i) ← next slower service
6:                 update the EST and the LFT of other tasks on the path
7:             end if
8:         end for
9:     until (no change is done)
10:    set EST, LFT and SS according to best for all tasks ∈ path
11:    mark all tasks of the path as scheduled
12:end procedure
```

**Algorithm 4. Fair Path Scheduling Algorithm.**

This algorithm can professionally solve the MCKP in a lot of cases. However, the most efficient literal algorithm for the MCKP is based on the Branch and Bound approach [15]. These algorithms usually discover the best possible solution for a relaxed version of the problem, e.g. linear programming entertainment, which lets 0 6 xij 6 1, and use it as an upper bound for the original problem. Using this upper bound they abolish partial solutions whose upper bound is less than the current best solution. Finally, there are some polynomial time approximation algorithms which try to find an inexact (estimated) solution with a bounded worst-case relative error denoted by $\epsilon$. It means P – P $*6 \epsilon$ P$*$, where P $*$ is the optimal solution for the problem, and P is the solution found by the approximation algorithm. To find some references to these algorithms, see [11].

## 3. Performance evaluation

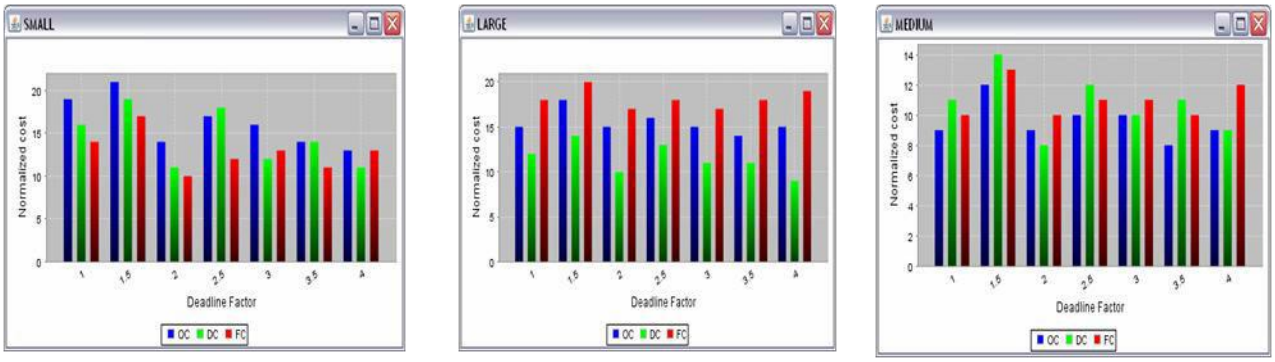
In this segment, we will present our simulation of the Cloud Partial Critical Paths algorithm Quality of Service (QoS) [12] plays a critical role in the affective reservation of resources inside service oriented distributed systems. The Cloud compute is promoting by the commerce rather than academic which determines its focus on user applications. Different users have different QoS Requirements. So according to the given deadline and budget, the proposal is formulated on scheduling model from the user's perspective. The first is how to measure various QOS attribute of a Cloud service. A lot of these attribute differ over time. However, without having precise measurement models for each attribute, it is not possible to compare different Cloud services or still discover them. The attribute be Responsibility, Agility, Assurance of Service, Security and Privacy, and Usability.

FAIR PATH ALGORITHM

| EST | LFT |
|---|---|
| 163.0483476642171 | -22.74834766421 71 |
| 144.1659490063445 6 | -3.8659490063445603 |
| 866.1640077005805 | -5.887007700580568 |
| 326.66555521025856 | 53.6324447897414 |
| 47.357656317692374 | 119.60901034897 43 |
| 88.39995830159344 | 78.566708365073 23 |
| 759.6046018918066 | 167.359064774859 92 |
| 31.202870437649835 | 198.16953710431 65 |
| 158.09891132650807 | 325.0655779931747 7 |
| 1125.1722927042824 | 375.1033739623841 |
| 1102.1552259392033 | 398.12044072746323 |
| 266.90383845710375 | 433.87050512377044 |
| 150.668603081367 | 710.966603081367 |
| 953.9454356141053 | 1280.9094356141052 |
| 1276.4305068183303 | 1603.3945068183302 |
| 809.0670437642292 | 1616.029043764229 |

**Table 1. Computation time**

**Responsibility**—these groups of QoS attribute is used to measure a variety of Cloud provider exact characteristics. This is important to build the trust of a customer on any Cloud provider. Rejection organization will want to organize its applications in addition to store their critical information in a place where there is no accountability of security exposures and compliance.




(a) (b)

**Agility—**the most important advantage of Cloud computing is that it adds to the agility of an organization. Agility in cloud is measured as a rate of change metric, showing how quickly new capabilities are integrated into IT as desirable by the commerce. When considering a Cloud service‘s agility, organizations want to understand whether the service is portable, adaptable, and flexible.

**Fig 1. Average Cost decrease percentage of the SC-PCP**

**Cost**—the first queries that arise in the mind of organization previous to switch to Cloud computing are whether it is cost effective or not. Therefore, cost is clearly one of the vital attribute for IT and the business. price tend to be the single most scientific metric these days, but



(c)

it is important to express cost in the characteristics which are relevant to a particular business organization. The cost of the VM is Where, and d are weights for each resource attribute and. The weight of each attribute can vary from application to application.

**Performance**—there are a lot of different solutions accessible through Cloud providers addressing the IT needs of special organizations. Each solution has different performance in terms of functionality, service response time and accuracy. Organization need to appreciate how their application will execute on top of the different Clouds and whether these deployments meet their expectations.

**Assurance**—this quality indicates the possibilities of a Cloud service performing as expected or promise in the SLA. Every organization looks to expand their business and provide better services to their customers. Therefore, reliability, resiliency and repair constancy are important factor in selecting Cloud services.

**Security and Privacy**—data protection and privacy are important concern intended for nearly every organization. Hosting data below one more organization‘s manage is always a critical issue which requires stringent security policies employed by Cloud providers. For example financial organizations usually need fulfillment with system connecting data integrity and privacy. Security and Privacy is multi-dimensional in nature and includes lots of attribute such as defensive confidentiality and privacy data integrity and availability.

**Usability**—intended for the rapid acceptance of Cloud services the usability plays a significant role. The easier to use and learn a Cloud service is, the faster an organization can switch to it. The usability of a Cloud service can depend on multiple factors such as Accessibility; Install ability, Learn ability, and Compatibility.

**The sub attributes are calculated as follows**:

**Suitability**--Suitability is distinct as the amount to which a customer‘s supplies be meet through a Cloud provider. Present are two sub-cases before we can define suitability. First, if after filtering the Cloud providers there are more than one Cloud provider which satisfies all the essential and non-essential requirements of the customer, then all are suitable. Otherwise, if filtering results in an empty Cloud provider list then persons providers which satisfy the essential features are chosen. In this case, suitability will be the degree to which service features come closer to user requirements. The resultant metric is:

**Accuracy**—the accuracy of the service functionality measures the degree of closeness to the user‘s definite values when using a service compared to the expected values. For computational resources such because Virtual Machines accuracy‘s first indicator is the number of times the Cloud provider deviated from an assured SLA. It is described as the frequency of failure in fulfilling the promised SLA in terms of Compute units, network, and storage. If is the number of times the Cloud provider fails to convince assured values for user over the service time T , then

accuracy frequency is defined as where n is the number of previous users. An extra indicator of accuracy is the accuracy value which is defined by where α can be computational, network or storage unit of the service and is service time for user.

**Transparency**—Transparency is an imperative aspect of Cloud services due to the rapid growth of these services. Transparency specifies the extent to which users‘ usability is affected by any changes in service. Therefore, it can be supposed as a time for which the performance of the user‘s application is concerned through a change in the service. It can also be calculated in terms of the frequency of such effects. So, it can be calculated by where is the number of calculation.

**Interoperability**—Interoperability is the capacity of a service to cooperate with other services presented moreover by the same provider or other providers. It is more qualitative and can be described by user knowledge. Additional than as it be a significant parameter intended for Cloud customers we provide an estimate, which is defined.

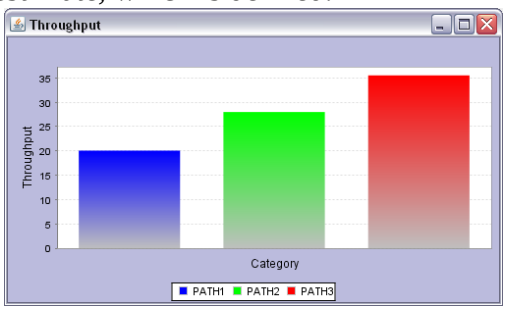



Fig. 2 Throughput Calculation

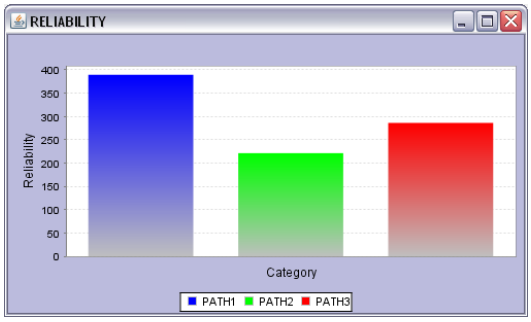



**Fig. 3 Reliability Variation of different factor**

**Reliability**—Reliability reproduces how a service functions without failure during a given time and condition. Therefore, it is defined based on the mean time to failure assured by the Cloud provider and previous failures practiced by the users. It is measured by: Where, the is the number of users who practiced a failure in a time interval less than assured by the Cloud provider is number of users, and is the promised mean time to failure. Reliability of storage can be defined in terms of durability that is the chance of failure of a storage device.

**Stability**—Stability is described as the unpredictability in the performance of a service. For storage, it is the variance in the average read and writes time. For computational resources, it is the divergence from the performance specified in SLAs, i.e. Where α can be computational unit, network unit or storage unit of the resource; is the examined average performance of the user i who leased the Cloud service, is the assured values in the SLA; T is the service time; and n is the total number of users.

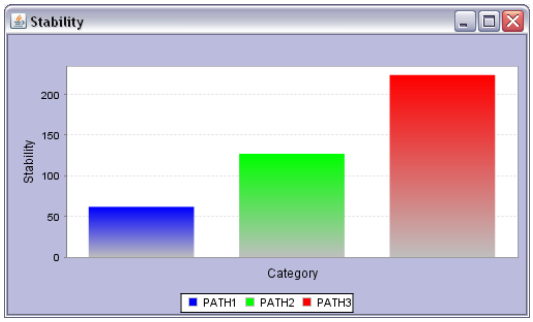



**Fig. 4 Stability for different methods**

## 4. CONCLUSIONS

The Cloud computing model enable user in the direction of get their necessary services by means of preferred QoS (such as deadline) by paying an appropriate price. In this paper, we propose a new algorithm named the SaaS Cloud Partial Critical Path (SC-PCP) for workflow scheduling in SaaS Clouds, which minimizes the total execution cost while meeting a user-defined deadline. We evaluate our algorithm by simulating it with imitation workflows that are based on real scientific workflows with different structures and sizes. The results show that SC-PCP outperforms another highly cited algorithm called Deadline MDP.

Furthermore, the experiments show that the computation time of the algorithm is very low for the Decrease Cost and the Fair policies, but is much longer for the Optimized poliscy, although still acceptable for the mentioned workflows. In the future, we plan to extend our algorithm to support other Cloud computing models, such as IaaS and other pricing models.

## 5. REFERENCES

[1] Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J. and Brandic, I. __Cloud computing and emerging IT platforms: vision, hype and reality for delivering computing as the 5th utility‘‘, Future Gener. Comput. Syst., 25(6), pp. 599–616 (2009).

[2] Juve, G., Deelman, E., Vahi, K., Mehta, G., Berriman, B., Berman, B.P. and Maechling, P. __Scientific workflow applications on Amazon EC2‘‘, 5th IEEE International Conference on e-Science, Oxford, UK (2009).

[3] Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B. and Good, J. __On the

use of cloud computing for scientific workflows‘‘, Fourth IEEE Int‘l Conference on e-Science (e-Science 2008), Indiana, USA, pp. 640–645 (2008).

[4] Deelman, E. __Grids and Clouds: making workflow applications work in heterogeneous distributed environments‘‘, Int. J. High Perform. Comput. Appl., 24(3), pp. 284–298 (2010).

[5] Weinhardt, C., Anandasivam, A., Blau, B. and Stoesser, J. __Business models in the service world‘‘, IEEE IT Prof., 11(2), pp. 28–33 (2009).

[6] Deelman, E., et al. __Pegasus: a framework for mapping complex scientific workflows onto distributed systems‘‘, Sci. Program., 13, pp. 219–237 (2005).

[7] Wieczorek, M., Prodan, R. and Fahringer, T. __Scheduling of scientific workflows in the ASKALON grid environment‘‘, SIGMOD Rec., 34(3), pp. 56–62 (2005).

[8] Berman, F., et al. __New grid scheduling and rescheduling methods in the GrADS project‘‘, Int. J. Parallel Program., 33(2), pp. 209–229 (2005).

[9] Ramakrishnan, L., et al. __VGrADS: enabling e-science workflows on grids and clouds with fault tolerance‘‘, ACM/IEEE International Conference on High Performance Computing and Communication (SC09), Portland, Oregon, USA (2009).

[10] Ostermann, S., Prodan, R. and Fahringer, T. __Extending grids with cloud resource management for scientific computing‘‘, 10th IEEE/ACM International Conference on Grid Computing, Banff, Alberta, Canada, pp. 42–49 (2009).

[11] Garey, M.R. and Johnson, D.S., Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman (1979).

[12] Kwok, Y.K. and Ahmad, I. __Static scheduling algorithms for allocating directed task graphs to multiprocessors‘‘, ACM Comput. Surv., 31(4), pp. 406–471 (1999).

[13] Topcuoglu, H., Hariri, S. and Wu, M. __Performance-effective and low complexity task scheduling for heterogeneous computing‘‘, IEEE Trans. Parallel Distrib. Syst., 13(3), pp. 260–274 (2002).

[14] Bajaj, R. and Agrawal, D.P. __Improving scheduling of tasks in a heterogeneous environment‘‘, IEEE Trans. Parallel Distrib. Syst., 15(2), pp. 107–118 (2004).

[15] Wieczorek, M., Hoheisel, A. and Prodan, R. __Towards a general model of the multi-criteria workflow scheduling on the grid‘‘, Future Gener. Comput. Syst., 25(3), pp. 237–256 (2009).

[16] Abrishami, S., Naghibzadeh, M. and Epema, D. __Cost-driven scheduling of grid workflows using partial critical paths‘‘, Proceedings of the 11th IEEE/ACM Int‘l Conference on Grid Computing (Grid2010), Brussels, Belgium, pp. 81–88 (2010)

[17] S. Abrishami, M. Naghibzadeh. ―Deadline-constrained workflow scheduling in software as a service Cloud‖,