

# Dynamic Scheduling in Real Time with Budget Constraints in Hybrid Clouds

Ovidiu-Cristian Marcu<sup>(✉)</sup>, Catalin Negru, and Florin Pop

Computer Science and Engineering Department,  
University Politehnica of Bucharest, Bucharest, Romania  
ovidiu21marcu@gmail.com, {catalin.negru, florin.pop}@cs.pub.ro

**Abstract.** In this paper we handle the problem of scheduling tasks in hybrid clouds for small companies which can spend only a fixed budget in order to handle specific situations where the demand is high and cannot be predicted. We describe a model with important characteristics for the resource utilization and we design an algorithm for scheduling tasks which are sent continuously for execution, optimizing the schedule for tasks with high priority and short deadline. We propose an architecture that meets the challenges encountered by small business in their systems for tasks scheduling. We describe the main components, Configuration Agent and Task Scheduler, and we analyze different test scenarios, proving the efficiency of the proposed strategy.

**Keywords:** Hybrid clouds · Dynamic scheduling · Task priority · Deadline · Budget

## 1 Introduction

Every company wants to offer the best services to its customers. A service is characterized by the execution of a task in an existing IT environment in which there is a limited number of virtual machines of different types. For the most part of the system's life, the applications are responding well during the execution of the tasks. However, at some point in time there can be an increase in demand for these services and in such cases of peak, we need to analyze the internal systems which can manage only a limited bag of tasks.

There are peak situations when a company system must deal with an increased and difficult to predict number of tasks with different priority ranks. The problem becomes complicated when the tasks are continuously sent for execution and the environment is very dynamic. In such a case we are unable to predict what will be the complete image of the entire scheduling system. We acknowledge that in these peak situations the private systems cannot handle all the tasks in order to successfully meet all the tasks' deadline.

With the arrival of the services offered on demand by different Cloud providers we have at our disposal many offers for the adjacent resources we may need in order to handle the extra workload that reaches the system in an

unplanned way. A company generally opts for a cloud provider based on both the feedback received from its internal consultants and the price of the virtual machines offered in the geographical region of the respective business.

In this paper we build an algorithm for dynamic scheduling in real time with budget constraints in hybrid clouds. The main goal is to minimize the number of tasks which exceed the deadline in the context of scheduling in hybrid clouds, having at our disposal a fixed monthly budget and considering that every task has known required characteristics like CPU, memory etc.

The problem of scheduling tasks under budget constraints in hybrid clouds needs to be approached since there are many small companies which do not have the option to invest in costly systems for an optimal scheduling, but they want to develop and do what is necessary in order to address efficiently the problem of scheduling. We need to take into consideration both the capacity of the private resources and the demand for services observed in different time periods in order to optimally schedule tasks in hybrid clouds.

We consider that the private system is composed of a very limited number of virtual machines with different types of resources like CPU, memory etc. Every week their system can handle a limited number of tasks. The problem occurs when the demand for services grows unexpectedly and the IT department does not have enough time to handle the situation properly. Thus, there is a high risk of losing customers due to the fact that the private system is not prepared for such a situation.

Although the recent research puts forward different approaches to solve the problem of scheduling tasks in hybrid clouds, we failed to identify one simple solution apt to provide an advantageous algorithm which can be implemented in several contexts and easily simulated in order to assess its profit. Thus, it is of utmost importance to offer an efficient solution for scheduling tasks in hybrid clouds for small companies having a limited budget.

The paper is structured as follows. Section 2 presents several strategies for scheduling in Cloud with some limitations. Then, Sect. 3 describes our proposed model, while Sect. 4 is presenting in details the scheduling algorithm with budget constraints. Using simulations, Sect. 5 shows that the obtained results sustain the optimality of proposed scheduling algorithm. The paper ends with conclusions.

## 2 Various Scheduling Algorithms Strategies

The problem of scheduling a bag of tasks in hybrid clouds has been researched on many occasions and different strategies have been proposed in order to make an optimal allocation of new resources when unexpected tasks are arriving to a system which is normally not prepared for additional compute efforts.

An interesting algorithm (BaTS) for scheduling a large bag of tasks under budget constraints is proposed in [1]. BaTS learns to estimate task completion time at runtime and offers significant cost savings when compared to Round Robin scheduler, at the expense of compute time. The tasks are independent of each other and can be preempted and rescheduled. BaTS uses an initial sampling

phase to build a first VM allocation and after a monitoring interval it refines the results. All this effort is made in order to give the user guidance for choosing between Cloud offerings, using an economic model for resource utilization.

In [2] an extension of the BaTS algorithm is researched to give estimates for budget and makespan for different scenarios and finally the user executes the schedule selected within the given budget. BaTS work is useful when we do not know a-priori task execution time. However many systems are in good knowledge of their tasks average execution time, like in our approach.

Tail-Phase optimization for BaTS is added in [3] with the main idea to use the idle machines in the tail phase and to decide which tasks are to be replicated based on task execution time information saved during the execution. BaTS learns runtime distribution properties of the bag of tasks for the considered cloud offerings. In this paper research it is proven that BaTS gives the user freedom in choosing between optimizing costs or improving execution time for a given bag of tasks, using the total number of tasks and the price of the VMs. It does not considers the machine utilization. The results are enhanced for users who can increase their budget and improvement exists in minimizing the makespan.

Preemptive tasks characterized by memory, CPU and data transmission requirements are considered to be scheduled in hybrid clouds having budget constraints in [4] and a binary integer program is formulated with the goal to deploy a fixed number of applications (having a fixed number of tasks and a specific deadline) while minimizing the total execution cost. The algorithm takes into consideration different VM instance types with parameters like CPU, memory and price. The authors acknowledge that in the hybrid setting the solution given has rather a poor performance and invite the users to develop custom heuristics.

HICCAM (Hybrid Cloud Construction and Management) project is proposed in [5] in order to investigate the design of software applications and algorithms for scheduling in hybrid clouds and it focuses on the optimization problem of allocating resources in both private and public infrastructures. This model is considering non preemptive workloads with a hard deadline with characteristics like CPU, memory and network bandwidth. It takes into consideration data transmission speeds and data locality during the scheduling process.

CAMTH (Cost optimal algorithm for multi-QoS constraints for Task Scheduling in Hybrid Cloud) model and algorithm are proposed in [6], considering three steps: private cloud scheduling, provider selection and public cloud scheduling. It supports security and reliability for QoS (Quality of Service) constraints. This model considers jobs consisting of many tasks; a task has a few characteristics like deadline, workload, data size which impact the data transmission and execution cost, different resource slots with compute power, price, storage cost, input and output data costs, network bandwidth, estimated execution time and estimation of VM finish time. In different experiments CAMTH is compared with FIFO (first in first out), Greedy and an efficient heuristic scheduling algorithms.

A hybrid algorithm for workflow scheduling is proposed in [7] in order to reduce the CPU idle time and to ensure a good load balancing. This raises a

general problem of reducing wasted resources when scheduling in hybrid clouds. A technique to ensure the QoS using the reputation of the offered resources is analyzed in [8] and a reputation guided genetic algorithm is build for scheduling independent tasks in inter cloud environments. Cost efficient scheduling heuristics for deadline constraint workloads are proposed in [9], taking into account computational and data transfer costs. There are three components, a public cloud scheduler when given runtime of the VM types are available and given data set size while taking into account the cost for execution and transferring data, a private cloud scheduler and a hybrid cloud scheduler to decide.

A family of heuristics for BoT scheduling are defined in [10] and consists of two phases: task ordering and task mapping. The authors propose an algorithm that implements the Contract Net Protocol, a task sharing protocol where the collection of nodes is the contract net and each node can be a manager or a contractor. The heuristics consider unordered, ordered by size large to small or small to large tasks and a few task mapping policies like Random, maximum expected remaining allocation time, Maximum current remaining, the same with the minimum order. It does not take into consideration neither the budget nor makespan minimization. When dealing with scheduling problems in practice there are a number of general proposed procedures for deterministic scheduling [11]. The following techniques are heuristics so they do not guarantee an optimal solution but rather reasonably good solutions: Dispatching Rules: Service in Random Order, First Come First Served; Composite Dispatching Rules; Local Search: Simulated Annealing and Tabu-Search. There are two types of Heuristics: constructive (starting without a schedule but building one in time) and improvement (starting with a complete schedule and trying to obtain a better one by updating the previous) [11].

Because most of the scheduling problems in the real world are NP-hard, it is difficult to find an optimal solution just using a single VM resource. In practice, most systems have a very high utilization so investing in adjacent resources just for the sake of the scheduling problem is not taken into consideration because of the low budget. Recent research approaches are considering different aspects when trying to solve the BoT scheduling problem when dealing with hybrid clouds. Task characteristics like priority, arrival rate, execution time, deadline or data size input or output constraints are important and mainly considered in the studied papers. In practice we need simple algorithms which can be implemented easily based on different task patterns. Having all of these options in mind when designing a solution for a practical scheduling will give a better chance for modeling the system in order to apply some heuristics based on a possible input of the problem.

### 3 Model Description of the Problem

We will further present you a mathematical model that will be considered in solving our scheduling problem and we describe the notations that we use in this paper. We define two participants of the hybrid clouds, Tasks ( $T$ ) and Virtual

Machines (VM), and we describe the most important properties that we consider to be used in the developed algorithm presented in this paper. We acknowledge that in the real systems there may be other properties that can influence the scheduling problem and we specify that such properties are not considered.

A task has the following definition:

$$T_j = (Type_j, TaskReq_j, e_j, d_j, w_j, input_j, output_j) \quad (1)$$

where  $Type_j$  is the type of a task, basically an unique name which describes its resources, role and execution time;  $TaskReq_j$  are requirements of the task:  $TaskReq_j = \{CPU_j, Memory_j\}$ ;  $e_j$  represents the date time when the task execution is completed;  $d_j$  represents the deadline of the task expressed as a date time;  $w_j$  is a priority index and its value ranges in a scale of 1 to 10, for example it can be LOW=1, MEDIUM=5, HIGH=10;  $input_j$  is the size of data input the task needs to process when the execution of the task begins;  $output_j$  is the size of the data output that the task produces after its execution.

A Virtual Machine has the following definition:

$$VM_i = (VMType_i, CPU_i, Memory_i, Bandwidth_i, VMPrice_i) \quad (2)$$

where  $VMType_i$  is the type of the VM and it is basically a name hiding the resources offered. We assume that the public cloud providers offer similar VMs like those provided by the private data center;  $VMPrice_i$  is the price of a VM expressed in units per hour, for example 1 u for one hour.

A task will be mapped only to a  $VM_i$  which corresponds to at least its characteristics described by  $TaskReq_j$ . The private data center offers a limited number of VMs with different characteristics  $VMType_i$  and these resources are offered at no extra cost. The budget  $B$  is limited for a fixed period (a number of days) and is expressed in units, for example  $B = 1000u$  for seven days. We do not have restrictions to retain a part of the budget for a specific task or period of the day. Our goal is to minimize the number of tasks for which the execution time exceeds the deadline, especially those with higher priority, in the given budget, as defined by the Eq. (3).

$$\min \left\{ \sum_{d_j - e_j < 0} j \right\}, \quad B - fixed. \quad (3)$$

## 4 Scheduling Algorithm with Budget Constraints

The first steps to take before implementing a strategy for a hybrid cloud computing is to understand the current state of our own private data center. After careful analysis we have to determine the following architectural and model elements: describe the current infrastructure in terms of VM types and resources they offer ( $VMType$ ); give each task a priority, a type and check its requirements ( $w_j, TaskReq, Type \leftarrow j$ ); save the average execution time for each task and the

VM type used for scheduling ( $E_{ji}$ ); create a database to save the current state that will be used later for scheduling. We acknowledge the fact that a task can be executed on different VMs based on its characteristics and this could influence the execution time in some cases.

#### 4.1 Dynamic Scheduling Algorithm

In every moment tasks can arrive in the system and we need to correctly schedule each one so that the number of tasks that exceeds the deadline is minimized. We cannot know how many tasks (and their deadline) will arrive at some point and it is difficult to predict the number of extra resources we need to acquire from the public cloud in order to reach our goal. Scheduling tasks in the hybrid clouds raise two major problems: the first problem is to estimate at some point in time the total number of extra resources needed in order to satisfy each task's deadline, and this is equivalent to the current need for on-demand public cloud VMs we need to acquire; the second problem is to make a good choice for what task to schedule first and to which VM from either private or public cloud and what task characteristics to consider when scheduling. Both problems will be translated into decisions based on the bag of tasks arrived in the task's queue.

To resolve the first problem we propose a Configuration Agent (CA) that will have the role to inspect all the tasks currently scheduled or not, but with their work execution not completed, and to calculate the configuration of necessary public VMs in an optimal estimation. The CA runs based on a configuration time (a few seconds), it calculates the list of extra VMs it needs to acquire from the public cloud and finally it delegates the role of acquiring the VMs to separate services developed to this end. We, therefore, present the pseudo-code of the algorithms executed by the CA. The following notations are used:  $T$  is the list of tasks not scheduled or not fully executed;  $T'$  is the list of tasks not scheduled;  $N$  is the list of available private VMs;  $M$  is the list of available public VMs;  $B$  is the budget;  $q$  is a constant/configuration parameter;  $m$  is a map with the estimated execution time for each task type;  $L$  is a list of VM types available in the public clouds;  $L'$  is a list of required VMs computed at previous configuration.

*CA Calculate Configuration Needed*

Required:  $T$ ,  $N$ ,  $M$ ,  $B$ ,  $q$ ,  $m$ ,  $L$ .

begin

    sort  $T$  by  $w = (\text{deadline} - \text{currentTime} - \text{estimatedExecution}) * \text{priority}$

    initialize private VMs start time with the current time

    sort VMs by CPU, Memory and price

    Run Tasks scheduling simulation algorithm

    Obtain a list of tasks with the deadline exceeded

    Keep the tasks where  $\text{arrivalTime} + \text{estimatedExecution} \leq \text{deadline}$

    Every task can be scheduled to a VM from  $L$

    Calculate the number of VMs to acquire for each VMType

    return a List of necessary VMs to allocate from the public cloud

end

We can further try to reach a better schedule by giving a second chance to the configuration agent by activating the phase described by the algorithm Check Reconfiguration and Relaunch.

*CA Check Reconfiguration and Relaunch*

```

Required: L', q.
begin
  calculate c = T/q
  if L' is not empty and c>0
    decrease c by one
    Run Calculate Configuration Needed
    update L'
  return L', a list of extra VMs to allocate from the public cloud
end

```

We further present the Tasks Scheduling Simulation algorithm that is used by the CA.

*CA Tasks Scheduling Simulation*

```

Required: T, N, M, m, B.
begin
  initialize an empty list of tasks with deadline exceeded, TD
  for each task j in T
    bestStartTime = null
    bestVM = null
    for each VM i in M
      if tasks requirements are covered by VM CPU, Memory, Storage
        vmRule = vmStartTime + taskEstimatedExecution - vmEndTime
        if vmRule < 0
          if bestStartTime is null
            bestStartTime = vmStartTime; bestVM = VM
          else if bestStartTime > vmStartTime
            bestStartTime = vmStartTime; bestVM = VM
    endfor
  for each VM i in N
    if tasks requirements are covered by VM CPU, Memory, Storage
      if bestStartTime is null
        bestStartTime = vmStartTime; bestVM = VM
      else if bestStartTime > vmStartTime
        bestStartTime = vmStartTime; bestVM = VM
    endfor
  if bestVM not null
    save task schedule simulation on bestVM
    update task execution time=bestVmStartTime+taskEstimatedExecution
    update bestVmStartTime = task execution time
    if task execution time > deadline
      add task to TD

```

```

else
    add task to TD
endfor
return TD
end

```

To resolve the second problem we propose a Task Dispatcher Scheduler (TDS) that will have the role to schedule all the tasks arrived but not yet proposed for execution. TDS has two components: the first one is a simple listener for new tasks and has the role of saving and proposing for execution each new task arrived (Dispatcher), while the second one is the Scheduler component and it will run based on a configuration time (few seconds but less than the CA parameter). We, therefore, present the pseudo-code of the algorithm executed by the TDS.

### *TDS Tasks Scheduling*

Required:  $T'$ ,  $N$ ,  $M$ ,  $m$ .

```

begin
    sort  $T'$  by  $w=(deadline-currentTime-estimatedExecution)*priority$ 
    initialize private VMs start time with current time
    sort VMs by CPU, Memory and price
    for each task  $j$  in  $T'$ 
        scheduled = false
        for each VM  $i$  in  $M$ 
            if tasks requirements are covered by VM CPU, Memory, Storage
                 $vmRule = currentTime + taskEstimatedExecution - vmEndTime$ 
                if  $vmRule < 0$ 
                    schedule task  $j$  on VM  $i$ 
                    update VM  $i$  not available
                    scheduled = true; break
            endifor
        if scheduled is true
            update  $M$ ; continue
        for each VM  $i$  in  $N$ 
            if tasks requirements are covered by VM CPU, Memory, Storage
                schedule task  $j$  on VM  $i$ 
                update VM  $i$  not available
            endifor
        update  $M$  and  $N$ 
        if  $M$  is empty and  $N$  is empty
            break
        endifor
    end

```



## 4.2 Scheduling System Architecture

We further illustrate in Fig. 1 the architecture for dynamic scheduling in hybrid clouds, consisting of the following components: the task queue being the input of the system; the TDS component having the role of saving and scheduling each task according to the proposed algorithm; the CA component with its role to calculate the configuration of extra VMs to be acquired from the public cloud; the private and public cloud VMs available for scheduling; the DB database to persist the tasks information and VMs configuration.

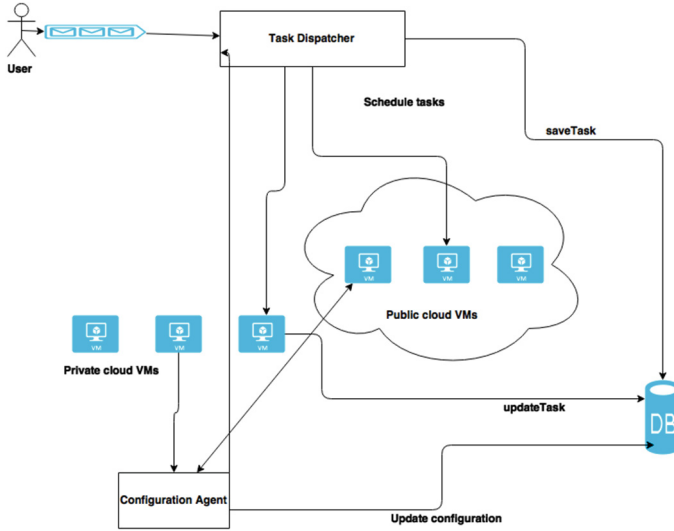


Fig. 1. System components and main events

## 5 Scheduling Test Scenarios and Analysis of the Results

In order to show the manner in which the algorithm works for the considered architecture we have implemented a set of services, for simulation purpose, based on Java technologies. We have built a database with the following tables:

- *Tasks* table contains information on the scheduled tasks such as type, priority, deadline, resources required, arrival and execution time;
- *Vms* table contains information on the available VMs in the private and public cloud;
- *TasksSched* table contains the schedule information of the tasks such as task identifier, VM name and start time of the execution;
- *TasksEstimates* table contains the average execution time for each task type, initially populated with an estimated execution time values.

**Table 1.** Private cloud VMs and resources

Name	CPU	Memory	Price	Storage price	Bandwidth price	Instances
Small	1	1024	0	0	0	2
Medium	2	1024	0	0	0	1
High	4	4096	0	0	0	1

We describe the Virtual Machines and the tasks that we have considered in our test scenarios (Table 1).

Every private VM is backed by a similar public VM. We consider that the number of the VMs that can be acquired in the public cloud is unlimited, so this should not influence our tests (Table 2).

**Table 2.** Public cloud VMs and resources

Name	CPU	Memory	Price	Storage price	Bandwidth price	Instances
Small	1	1024	1	0.01	0.005	$\infty$
Medium	2	1024	2	0.02	0.005	$\infty$
High	4	4096	4	0.02	0.01	$\infty$

We have considered tasks with different priorities and we define multiple types in order to restrict the execution of a task to different VMs based on the task requirements like CPU, Memory (Table 3).

**Table 3.** Task types and requirements

Type	Priority	CPU	Memory	Execution estimate	Deadline (Random)
1	1	1	1024	5 s	100–150 s
2	1	2	1024	6 s	550–600 s
3	5	1	1024	10 s	250–300 s
4	5	2	1024	12 s	400–500 s
5	10	1	1024	15 s	550–600 s
6	10	2	1024	20 s	350–400 s
7	10	4	4096	25 s	550–600 s

All the tasks can be executed in both private and public cloud VMs. There are tasks having their execution restricted to some VMs, for example the task with the types 2, 4 and 6 can be executed only on *medium* or *high* VMs.

We run multiple tests considering the following distribution of tasks (Table 4):

**Table 4.** Tasks distribution scenarios

Count \ Type	300	600	900	1200	1500	1800	2400	3000
1	20%	25%	40%	30%	35%	50%	40%	40%
2	20%	25%	20%	25%	20%	5%	20%	20%
3	20%	20%	10%	10%	20%	15%	10%	10%
4	20%	20%	10%	10%	15%	5%	10%	10%
5	10%	5%	10%	20%	5%	20%	10%	10%
6	5%	5%	5%	0%	5%	5%	5%	5%
7	5%	0%	5%	0%	5%	0%	5%	5%

When scheduling, in practice, it is important to understand what is the distribution of tasks that a system normally handles. We have considered different patterns for a total number of tasks from 300 to 3000. We have executed each simulation considering our approach where the CA is sorting the list of tasks  $T$  by  $w = (deadline - currentTime - averageExecutionTime) * priority$ . We have configured CA to run every three seconds and the TDS to run every second. We have configured the parameter  $q$  to be equal 20. Every 2 seconds we have calculated the tasks' average execution time that will be used by the CA task scheduling simulation algorithm.

The results are very promising, enabling us to reduce the number of tasks exceeding the deadline to zero. For the considered test cases we present the consumed budget, necessary to obtain the perfect minimization (Table 5).

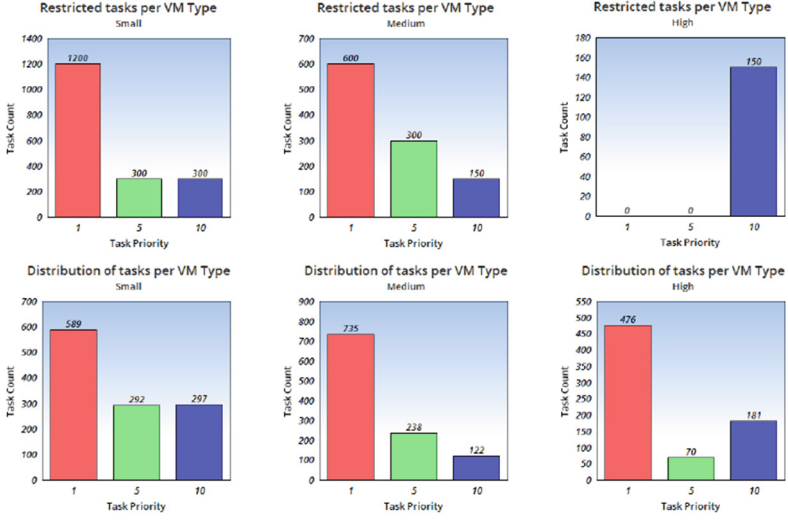
**Table 5.** Budget consumed for public VMs

Total \ Budget	300	600	900	1200	1500	1800	2400	3000
500	13	26	43	98	35	56	121	188

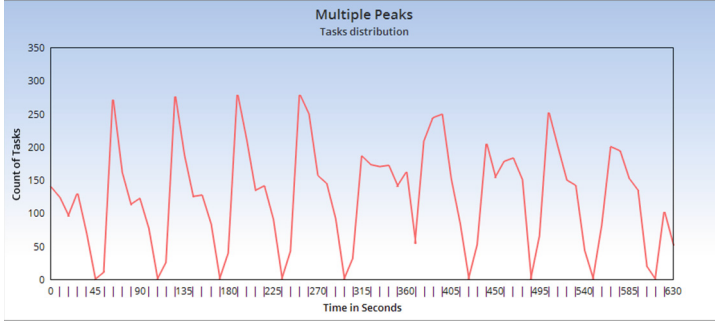
Load balancing of tasks is good in every case, being ensured by the algorithm, as we can see from Fig. 2.

We have considered a Special Test with Multiple Peaks and we have chosen a tasks' distribution with multiple peaks as illustrated in Fig. 3. We send three times the same amount of tasks respecting the distribution proposed. The total number of tasks scheduled is 26250 and is executed in a time interval of 1890 seconds. The reason for choosing this test configuration is to assess the performance of CA and TDS components in a scenario as realistic as possible.

We repeat the test three times, the first one considering the tasks are ordered by the proposed weight  $w$ , the second one considering the tasks are ordered by  $w = estimatedExecutionTime$  and arrival date, the third one considering the



**Fig. 2.** Load balancing. (Color figure online)



**Fig. 3.** Large bag of tasks.

tasks are ordered by  $w = \text{priority}$  and arrival date. The results of the tests, considering the proposed approach, shortest execution first and respectively order by priority, are the following:

- budget consumed is 1513 u, 1998 u and respectively 1597 u;
- total number of tasks with deadline exceeded is 9, 37 and respectively 14;
- the total number of seconds representing the delay in meeting the deadlines is 97, 331 and respectively 127.

We observe the fact that the consumed budget in the executed test scenarios is not always in the same range and we conclude that it depends on the manner in which the tasks are ordered, both for scheduling and for its simulation.

## 6 Conclusions and Future Work

Many organizations are opting for an attractive hybrid cloud approach because a company can leverage its private cloud resources but use public cloud services when a peak demand arises. We considered the problem of scheduling a bag of tasks in hybrid clouds, under budget constraints, with the goal to minimize the number of tasks that exceeds the deadline. We have outlined a model for the main entities considering the tasks to be non-preemptive and characterized by a type, priority and resource constraints for execution and virtual machines from the private and public cloud. We have proposed an architecture for scheduling and we have defined the role of each component. We observed that the manner in which the two components CA and TDS have been configured has a positive influence on the results leading to an optimal dynamic scheduling. Thus, in view of the our goal, respectively to minimize the number of tasks that exceeds the deadline, we conclude that the proposed algorithm is optimal and the objective has been reached.

We acknowledge that the model can be further extended to consider other characteristics such as tasks that are restricted to run only in the private cloud or only on some dedicated virtual machines and we plan, in the future, to take these elements into consideration. We further intend to extend the problem of scheduling in hybrid clouds under budget constraints also by taking into consideration the impact of the task input and output data size on the consumed budget. Moreover we intend to implement the proposed architecture and algorithm in a real client case, using a real platform, that will allow us to adjust the algorithm for the fault-tolerant adjacent impacts.

**Acknowledgements.** The research presented in this paper is supported by the project *DataWay: Real-time Data Processing Platform for Smart Cities: Making sense of Big Data*, PN-II-RU-TE-2014-4-2731 founded by UEFISCDI. We would like to thank the reviewers for their time and expertise, constructive comments and valuable insight.

## References

1. Opreescu, A.-M., Kielmann, T.: Bag-of-tasks scheduling under budget constraints. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), Indianapolis, IN, 30 November–3 December 2010, pp. 351–359. IEEE (2010). Print ISBN: 978-1-4244-9405-7
2. Opreescu, A.-M., Kielmann, T., Leahu, H.: Budget estimation and control for bag-of-tasks scheduling in clouds. *Parallel Process. Lett.* **21**, 219–243 (2011). doi:[10.1142/S0129626411000175](https://doi.org/10.1142/S0129626411000175)
3. Opreescu, A.-M., Kielmann, T., Leahu, H.: Stochastic tail-phase optimization for bag-of-tasks execution in clouds. In: IEEE/ACM Fifth International Conference on Utility and Cloud Computing (2012)
4. Van den Bossche, R., Vanmechelen, K., Broeckhove, J.: Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. In: 2010 IEEE 3rd International Conference on Cloud Computing, pp. 228–235. IEEE (2010). Print ISBN: 978-1-4244-8207-8

5. Van den Bossche, R., Vanmechelen, K., Broeckhove, J.: Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds. *Future Gener. Comput. Syst.* **29**(4), 973–985 (2013)
6. Yanpei, L., Chunlin, L., Zhiyong, Y., Yuxuan, C., Lijun, X.: Research on cost-optimal algorithm of multi-QoS constraints for task scheduling in hybrid-cloud. *J. Softw. Eng.* 33–49 (2015)
7. Nicolae, A.A., Negru, C., Pop, F., Mocanu, M., Cristea, V.: Hybrid algorithm for workflow scheduling in cloud-based cyberinfrastructures. In: *International Conference on Network-Based Information Systems* (2014)
8. Pop, F., Cristea, V., Bessis, N., Sotiriadis, S.: Reputation guided genetic scheduling algorithm for independent tasks in inter-clouds environments. In: *27th International Conference on Advanced Information Networking and Applications Workshops* (2013)
9. Van den Bossche, R., Vanmechelen, K., Broeckhove, J.: Cost-efficient scheduling heuristics for deadline constrained workloads on hybrid clouds. In: *Third IEEE International Conference on Cloud Computing Technology and Science* (2011)
10. Gutierrez-Garcia, J.O., Sim, K.M.: A family of heuristics for agent-based cloud bag-of-tasks scheduling. In: *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* (2011)
11. Pinedo, M.L.: *Scheduling Theory, Algorithms and Systems*, 4th edn. Springer, Berlin (2012). ISBN 978-1-4614-1986-0

Economics of Grids, Clouds, Systems, and Services  
12th International Conference, GECON 2015,  
Cluj-Napoca, Romania, September 15-17, 2015, Revised  
Selected Papers  
Altmann, J.; Silaghi, G.C.; Rana, O.F. (Eds.)  
2016, XV, 323 p. 82 illus., Softcover  
ISBN: 978-3-319-43176-5