Amirkabir University of Technology
(Tehran Polytechnic)
Vol. 46, No. 1, spring 2014, pp. 19- 29

Amirkabir International Journal of Science & Research
(Modeling, Identification, Simulation & Control)
(AIJ-MISC)

A Clustering Approach to Scientific Workflow Scheduling on the Cloud with Deadline and Cost Constraints

Arash Deldari[1], Mahmoud Naghibzadeh[1]*, Saeid Abrishami[1], and Amin Rezaeian[1]

1- Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, IRAN

ABSTRACT

One of the main features of High Throughput Computing systems is the availability of high power processing resources. Cloud Computing systems can offer these features through concepts like Pay-Per-Use and Quality of Service (QoS) over the Internet. Many applications in Cloud computing are represented by workflows. Quality of Service is one of the most important challenges in the context of scheduling scientific workflows. On the other hand, the remarkable growth of the multicore processor technology has led to the use of these processors by service providers as building blocks of their infrastructure. Therefore, scheduling scientific workflows on the Cloud requires especial attention to multicore processor infrastructure which adds more challenges to the problem. On the other hand, in addition to these challenges users' QoS constraints like execution time and cost should be regarded. The main objective of this research is scheduling workflows on the Cloud, considering a multicore based infrastructure. A new algorithm is proposed which finds clusters of the workflow that can be executed in parallel while having large data communications. These kinds of clusters could be appropriate candidates to be executed on a multicore processor. In contrast, there are other clusters which should be executed in serial. This algorithm investigates whether serial execution of these clusters is possible or not. The experimental results show that the algorithm has a positive effect on execution time and cost of the workflow execution.

KEYWORDS

High Throughput Computing, Cloud computing, Workflow scheduling, Clustering, Time Overlap

*email: naghibzadeh@um.ac.ir

1. Introduction

The concept of High Throughput Computing (HTC) with loosely coupled applications such as bags of tasks or scientific workflows has been popular for years. One of the greatest features of HTC systems such as Grid and Cloud is that processing resources with variable sizes and capabilities are accessible on-demand. The well-known Community Grid offered required resources free of charge. Utility Grids use the economic concept such that storage and processing resources with different Quality of Service (QoS) characteristics can be provided at different prices [1]. The required QoS specifications of users are guaranteed through contracts between the users and the service providers in these systems called Service Level Agreements (SLAs). As Cloud Computing is considered as an extension of Utility Grids, it uses the concepts of Pay-Per-Use and Quality of Service to offer this feature over the Internet. Instead of investing in the required resources and their maintenance, they can be leased as required. The on-demand and Pay-Per-Use usage of Cloud resources makes this infrastructure highly scalable and cost effective. Storage and processing resources are presented on the Cloud through the virtualization process which is an important feature of the Cloud. The virtual environment offered in the Cloud is completely independent of other environment as well as the physical hardware [2]. This environment also allows the service providers to customize their resources based on the users' requirements. Thus, Cloud computing is regarded as an appropriate infrastructure for executing HTC applications such as scientific workflows.

The main purpose of scheduling algorithms is increasing system performance and throughput. Optimal scheduling in the Cloud is considered to be a multi objective problem and it is hard to solve [3]. Thus, most of the traditional scheduling algorithms for other systems cannot be applied to the Cloud environment.

A workflow can be described as a set of tasks in which each task is dependent on the data produced by its predecessors. Therefore, this communication between tasks causes precedence constraints in the workflow. This model has been used in a lot of scientific processes, including chemistry, computer science, physics, biology, etc. A scientific workflow can be described as a Directed Acyclic Graph (DAG) in which the vertices represent the tasks and the edges represent control and data dependencies. Two of the most important items of Quality of Service in scheduling workflows on the Cloud are time and cost.

On the other hand, the widespread development of the multi-core processor technology has caused the service providers to choose these kinds of processors as their infrastructure. Consequently, executing a workflow on the Cloud must consider scheduling on multi-core processors in such a way that execution time and cost are decreased.

In this paper, a new scheduling algorithm called Cluster Combining Algorithm (CCA) is proposed. This scheduling algorithm minimizes the execution time of the workflow (i.e. makespan) and the cost considering the multi-core processor infrastructure. First of all a clustering algorithm is applied to the workflow DAG and each cluster in the primary clustering phase is mapped onto a single core processing resource for execution. Then the CCA combines these primary clusters with the purpose of scheduling the combined clusters on multi-core processors. Most of the work done in this context does not consider the time interval intersection in parallel execution of the workflows. This approach uses a new concept called Time Overlap to decide how to execute these combined clusters. The clusters with a high Time Overlap are executed in parallel while others with no Time Overlap or low Time Overlap are executed in series. A scoring approach is devised to decide serial or parallel execution of these clusters. A coefficient has been used to make a tradeoff between time and cost in the scoring function. The importance of the workflow's makespan or execution cost can be determined using this coefficient. The CCA algorithm pays especial attention to minimizing execution time and cost by maximizing the utilization of rented time periods of processing resources and also minimizing the inter cluster data communications. Experiments show that this approach has a positive effect on time and cost.

The rest of this paper is organized as follows. A background on workflow scheduling and resource management and related work on distributed systems is presented in section II. The proposed algorithm is described in section III. In section IV the simulation results and the performance evaluation of execution time and cost of the proposed algorithm are presented. The concluding remarks appear in the last section.

2. Related Work

Cloud computing can be considered as the extension of the Grid, parallel and distributed computing which offers 4 types of services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and Network as a Service (NaaS). The exceedingly high popularity of Cloud computing has drawn the attention of a lot of researchers in recent years. Scheduling and resource management are regarded as one of the most important issues in this context. The primary objective of such scheduling is increasing the performance of processing resources in distributed systems such as Clouds and Grids.

A great deal of research effort has been dedicated to the problem of scheduling on distributed systems like Grid and Cloud [4]-[8]. Energy efficient scheduling of HPC applications on Cloud data centers has been investigated by Gang et al. [9]. The main objectives of this research was reducing environmental pollution and increasing service provider’s profits, whereas other related research studies have mainly considered reducing cost. Beloglazov et al. [10] have also proposed a heuristic approach for resource provisioning in Cloud data centers. The main objective of this research was reducing energy by dynamic adaptation of Virtual Machine allocation during runtime and also turning idle nodes to sleep mode.

A large number of researchers have considered reliability, load balancing, accessibility and performance in scheduling and resource management on the Cloud [11] - [14].

Workflows are a common model for describing a wide range of scientific applications on distributed systems. The problem of scheduling workflows can be described as mapping tasks to proper resources and meeting certain QoS attributes. Generally workflow scheduling is divided into two categories: static and dynamic. In static scheduling, the plan is completed before runtime and the scheduling scheme does not change during execution. On the other hand, scheduling is done during runtime without any primitive plans in the dynamic mode [1].

The problem of optimal task scheduling is considered to be NP-complete. Therefore, many heuristic and meta-heuristic techniques have been proposed with polynomial time complexity [17-20].

Batch mode best effort scheduling algorithms usually schedule a group of ready tasks on the resources to be processed. In this context, Min-Min and Max-Min [15] are two of the most well-known algorithms. In both methods, firstly the earliest finish time of each task is computed. The Min-Min approach selects the task with the minimum earliest finish time to be scheduled. On the other hand, the Max-Min method chooses the task with the biggest earliest finish time. In a similar approach proposed by Etminani et al. [16], the group of ready tasks is divided into two and then one part is scheduled by Min-Min and the other part is scheduled by Max-Min.

The high computing power of distributed systems has made it an appropriate platform for workflow execution. One of the famous research works done by Topcuoglu et al. [17] in this context considers scheduling workflows on a heterogeneous computing environment. The HEFT algorithm assigns an upward rank to each task and maps it to the processor that has the earliest finish time. This research study has also introduced the CPOP algorithm which uses the summation of upward and downward rank as the priority of the tasks.

The DSC clustering algorithm which has been proposed by Yang and Gerasoulis [18] executes a set of tasks on each processor. Sarkar [19] also introduces a two-step procedure for scheduling: 1- Primary clustering based on the assumption that there exist infinite processors. 2- Aggregation and scheduling of these clusters to meet the number of the available processors.

Bittencourt and Madeira [20] have considered scheduling related tasks on the Grid. Their objectives were reducing runtime, maximizing utilization and throughput. In a similar research they proposed a method that minimized the required scheduling time and maximized the fairness between processes [21].

The HCOC algorithm for scheduling workflows on hybrid Clouds consisting of private and public Clouds has also been suggested by Bittencourt and Madeira [2]. The main objective of this research was to reduce makespan to meet a specified deadline and execution cost. Therefore, the general tendency has been to schedule the workflow on the available resources in the private Cloud. In cases where the makespan does not meet the specified deadline, the resources on the public Cloud have been leased on a Pay-Per-Use basis. Cost and the number of tasks have been considered when scheduling the workflow on the time slots in which the public Cloud is available.

The PCP algorithm has been proposed by Abrishami et al. [1] to schedule scientific workflows on IaaS Cloud which minimizes execution cost considering time constraint using the Partial Critical Path concept. Therefore, a sequence of tasks is mapped onto a processor for execution. They also proposed Budget-PCP [22] which minimizes the execution time with respect to the cost constraint. This approach distinguishes the sequence of tasks pretty well, but does not recognize the relationship between the sequences. Accordingly, this method does not function properly in scheduling on multi-core processors.

Poola et al. [23] have presented a workflow scheduling algorithm on the Cloud which considers robustness and fault-tolerance with time and cost constraints. This method solves the problem of uncertainties such as performance variations and failures in Cloud environment by adding slack time based on the deadline and budget constraints.

Kanemitsu et al. [24] have proposed a theoretical method for mapping jobs to computers in such a way that it considers parallel execution and this increases resource utilization. In this research, it was assumed that each computer has two or more processors that have been connected through infinite bandwidth. This method does not cover the time intersection between clusters that are going to be executed in parallel on different processing elements which might lead to a large amount of free time slacks on the processors.

In this work, the Time Overlap concept has been introduced to compute the time intersection between the clusters. Therefore, clusters with a high time overlap are executed in parallel. Moreover, this method increases the utilization of the processing resources by minimizing the free time gaps.

3. The Proposed Method

In this section, the proposed approach is described in detail. First the system model for the algorithm is illustrated and then the CCA method is demonstrated.

A. The System Model

The application model used for scientific workflow is a Directed Acyclic Graph G= (V,E) in which V = {vi | i= 1,…,V} denotes the set of tasks of the workflow and E = { ei,j | (i,j) ∊ {1,…,V} × {1,…,V}} represents the edges between the vertices. In addition, each edge has a weight which denotes the precedence constraint and the amount of communication between tasks vi and vj. A ventry and vexit with zero processing time and zero communication have been added to the DAG since the algorithm needs a graph with individual entry and exit nodes.

In addition, MET(vp) and MTT(ei,j) denote the Minimum Execution Time and the Minimum Transfer Time that are defined as:

$$MET(v_i)= \min_{s \in S_i} ET(v_i, s)$$

$$MTT(e_{i,j}) = \min_{r \in S_i, s \in S_j} TT(e_{i,j}, r, s)$$

Using the following definitions the Earliest Start Time is computed by:

$$EST(v_{entry}) = 0$$

$$EST(v_i) = \max_{v_p \in v_i\text{'s parents}} (EST(v_p)+MET(v_p)+MTT(e_{p,i}))$$

Also the Latest Finish Time of each task is defined as the latest time that the task's computations should be completed so that the workflow meets the specified deadline:

$$LFT(v_{exit}) = deadline$$

$$LFT(v_i) = \min_{v_c \in v_i\text{'s children}} (LFT(v_c)-MET(v_c)-MTT(e_{i,c}))$$

Therefore, the overall execution time or makespan of the workflow is defined as the time between ventry and the completion of vexit. Many techniques such as analytical benchmarking, code profiling, statistical prediction, and code analysis have been used to estimate the execution time of a task on an arbitrary resource [1].

It has also been assumed that TT(ei,j , a , b) and TC(ei,j , a , b) denote the estimated transfer time and the transfer cost between resource a that is processing task i and resource b that is processing task j. The transfer cost can be computed according to the service provider's data communication pricing policy between the resources and also in and out of the specific cloud. Each cluster is modeled as CL(v, config, est, eft) where v is the set of nodes in the cluster, config is the machine onto which the cluster is mapped for processing, and est and eft denote the cluster's Earliest Start Time and Earliest Finish Time. To model the Cloud resources, CONFIG= (C, P, T) which respectively denotes the number of cores, the leasing price for each resource and also the billing time intervals has been used. Although there are many QoS attributes in this context, only execution time and cost which are considered the most important ones have been used. Therefore, QoS(T, C) has been used.

B. The Proposed Algorithm

The main objective of this research is the static scheduling of workflows on the Cloud with especial consideration on multi-core processor platforms.

The increasing growth in the production of multi-core processors, plus the vast usage of Cloud service providers using this technology make the problem of scheduling workflows in multi-core Cloud resources an important challenge to be tackled. Considering the multi-core platform adds a number of difficulties to the scheduling problem. The main purpose of using these processors is parallel computing. The data communications and the precedence constraints between the tasks and the clusters of the workflow make the optimal scheduling problem of the workflow more difficult to solve. Therefore, the workflow must be divided into clusters that can be executed in parallel or series. The concept of clustering, the workflow has been studied in different scheduling algorithms [20]-[21]. An algorithm has been proposed in this research which combines the existing clusters with regard to certain criteria.

Moreover, the utilization of processing cores which are used for the execution of the clusters should be maximized.

The higher leasing cost of multi-core processors compared to single cores significantly increases the importance of cost in the multi-core Cloud. The higher leasing prices increase the importance of the utilization of the available cores. In addition, execution cost and makespan in the Cloud environment have opposite effects. This means that reducing the makespan needs higher processing power resources which results in higher leasing fees. On the other hand, resources with lower computing capacity and lower leasing prices must be used that increases the makespan in order to reduce the leasing cost. Therefore, a proper trade-off between these two concepts is very important.

The proposed algorithm consists of two main stages. At first, a clustering algorithm divides the workflow into primary clusters and schedules each cluster to a single core processor for execution. In the next stage, the main part of the algorithm is performed which combines these primary clusters regarding multi-core processors. Hence, after combining these clusters the algorithm decides where it will be mapped for processing and the attributes of the tasks and clusters such as EST and EFT will be recalculated. Since the combination of two clusters can reduce their inter cluster communications to zero, the attributes must be recalculated after each cluster combination phase.

Depending on the structure of the workflow, different clustering algorithms can be applied in the primary phase. Even each task can be considered as a cluster in the primary clustering phase. In this way, the proposed approach starts by combining these individual tasks and in a way it acts as a clustering algorithm with especial consideration of multi-core processing resources. The default algorithm that is used in the primary clustering phase is the algorithm proposed by Bittencourt and Madeira [21]. By using this algorithm, each primary cluster consists of nodes whose predecessors have already been scheduled or are to be scheduled along with them. To combine these clusters, some important issues must be carefully considered. For instance, there exist clusters that cannot be executed in parallel if they are combined together. Therefore, mapping them onto a multi-core processor for execution significantly increases the free time gaps and is not beneficial in terms of time and cost. The most important issue in the context of multi-core processors is to maximize the parallel execution of workflow clusters. The best case occurs when two or more clusters that can be executed in parallel are combined together and mapped onto a multi-core processor for execution. In this case, the execution time of the workflow is minimized and the inter-cluster communication time can be considered to be zero. Reducing these communication times to zero can have a great effect on the execution time.

In this research a Cluster Combining Algorithm (CCA) has been proposed which uses a new concept called Time Overlap. Time Overlap denotes the amount of time intersection between the clusters. This concept is used to decide whether to execute two combining clusters in series or parallel. With each workflow, a deadline or Latest Finish Time (LFT) is submitted

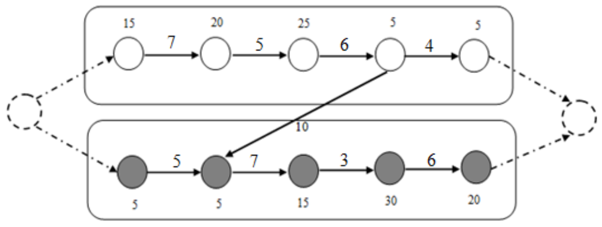

Fig. 1. : An example of serial execution

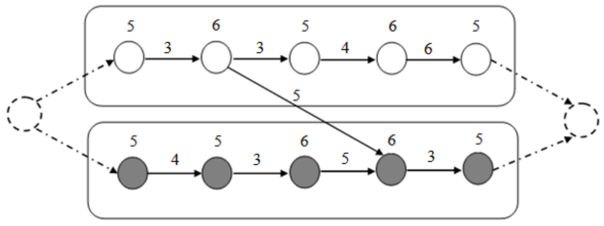

Fig. 2. : An example of parallel execution

by the user. By assigning zero to the Earliest Start Time (EST) to the starting node of the workflow, its EST and LFT can be computed for each task and cluster [1].

An Earliest Finish Time (EFT) has also been considered for each task. The EST and EFT of each cluster can be computed as:

EFT (C_i) =Max (T_j.eft) s.t $T_j \in C_i$

EST (C_i) =Min (T_j.est) s.t $T_j \in C_i$

where Ci denotes the i-th cluster and Tj denotes the j-th workflow task. Having computed the EST and EFT of each cluster, the time overlap of two clusters can be computed by considering EST as the beginning of the clusters period and EFT as its end. By computing the time period, the Time Overlap of these two clusters would be the period of time between the maximum of their ESTs and the minimum of their EFTs. Two clusters without any time overlap should not be executed in parallel. Assume the following example as shown in Figure 1. The numbers that appear on the nodes are supposed to denote the execution time of the tasks and the numbers on the edges are supposed to define the communication time between the tasks. At the first look, the two clusters seem that they could be executed in parallel. However, the communication between them affects the EST of the bottom cluster and consequently affects the cluster time overlap. This makes their parallel execution inappropriate. Parallel execution of two clusters with a low Time Overlap decreases the utilization of the cores of the processor. This is because mapping clusters with a low Time Overlap on multi-core resources increases the free time gaps. Thus, the stress in this approach is to try to execute these clusters in series. In contrast, executing two clusters in series is only possible if the LFTs of all the tasks in the combined clusters are met.

Executing two clusters may exceed one billing unit. In this case the costs of executing them in series would remain the same. To maximize the economic benefits in serial execution, the free time gaps of the available resources should be utilized.

This method increases the utilization of the processing resources by minimizing free time gaps. Now suppose the case with the two clusters shown in Figure 2. In this example, the communication does not affect the clusters EST and time overlap. Therefore, the Time Overlap for the clusters in this example is the highest possible. The proposed algorithm considers these two clusters to be executed in parallel. Parallel execution of these two clusters does not decrease the utilization of the cores of each processor.

In this research, a scoring approach is used to decide how to execute two combining clusters. The score is the ratio of the computation to the billing price of the resource for that particular amount of computational weight. Accordingly, achieving a higher score means more computations should be completed with a lower leased cost. In other words, the utilization of the processing resource in a specific leased period should be increased in order to increase the score. Therefore, a parallel score and a serial score are computed. These scores are then used to decide which twin clusters are the best instances to be combined together. This combination of clusters is done if the maximum score is improved by combining them. This score is also used to decide whether to execute the resulting cluster in parallel or in series. This means that if the parallel score is higher than the serial score, the cluster is executed in parallel and vice versa.

An α coefficient that shows whether execution time has more importance or execution cost, it is considered ($0 \leq \alpha \leq 1$) in order to compute this score. The larger the alpha coefficient is, the more important is the execution time. A tradeoff between execution time and cost is made when this coefficient is used. In cases in which execution time has more importance, the general tendency is parallel execution. Thus, in such cases even the clusters with low time overlaps are considered to be run in parallel which leads to more free time gaps in the resources and also to a higher cost. In other words, these free time gaps impose a higher execution cost on the workflow by leasing more processing resources. On the other hand, in cases where the emphasis is on execution cost, serial execution is preferred. Serial execution does not need a new computing instance. In this case, if there are free time gaps in the rented period, the available instances are used for execution. Otherwise, the instance should be rented for another period. This means that the algorithm executes clusters with low time overlaps in series. However, this is only possible if the LFTs of the tasks are met. This method maximizes the utilization of the rented time periods which leads to a more economical cost.

This algorithm receives a workflow and its related deadline as input. Based on the deadline, the EST and LFT of all the workflow tasks are computed. In Line 1 the primary clustering phase is executed. After the primary clustering phase is executed the internal communications of each cluster is zeroed in Line 2. The attributes of the tasks and clusters are updated and the primary scheduling phase is performed in Line 3. For each cluster pair the parallel score and serial score are computed. Based on these scores, the two clusters with the highest scores are combined together. Two clusters with high time overlap have a higher parallel score than serial score. On the other hand, two clusters with a low time overlap have a higher serial score. This combining clusters is continued until no other combination is possible. This means that the combination of the clusters is repeated until the combination does not improve the maximum score.

When a pair of combining clusters is considered for parallel operation, processors with more cores are used, e.g. if the two clusters use dual core processors for execution, after the combination phase a quad core processor would be used for executing the resulting clusters. Therefore, instead of leasing two dual core processors a quad core processor would be rented. In addition, more inter-cluster communications would be reduced to zero, which leads to a better makespan.

Calculating the serial score of two clusters is only possible if they use the same number of processing cores for execution. The first issue in serial execution is to decide which cluster should be executed first. Wrong precedence in serial execution may lead to a deadlock in workflow execution in some instances. Suppose cluster A which its tasks have parents in cluster B. In this case executing cluster A before cluster B in serial on the same processor leads to a deadlock. Consider the example shown in Figure 1. The correct order for the serial execution of these two clusters is to execute the top cluster before the bottom one. Otherwise, executing the bottom cluster before the top one in serial execution leads to a deadlock. In most cases, executing a cluster before the other cluster imposes a delay to the second cluster. Therefore, in this case serial execution is only possible if the LFTs of all the tasks in the second cluster are met. The general tendency in serial execution of the clusters is to make maximum use of the rented time periods of the processing resources.

```
Algorithm 1: Cluster Combining Algorithm
Compute EST and LFT of all the workflows tasks
Cluster the graph //Primary clustering algorithm
Update communications //zeroing internal communications for each cluster
Compute EST and LFT for each cluster and assign a single core processor to each cluster
While (cluster combination is feasible) do
begin
4-1- Bestscore = -inf; // Bestscore is –infinite at the beginning
4-2- For i=1 to n do // n is the number of current clusters
  begin
   For j=1 to i-1 do
     begin
      Pscore = Parallelscore ( i , j);
      Sscore = Serialscore ( i , j);
      Maxscore = Max ( Pscore , Sscore);
      If ( Bestscore < Maxscore) then
        begin
         Bestscore = Maxscore;
         ii = i;
         jj = j;
        end;
      end;
  end;
4-3- If (combining ii and jj is possible)
     Combine ( ii , jj ); //indices of clusters which makes the // highest combination score
    Else
     Break;
end; // end of while
```

In instances where there is not much difference between the parallel score and the serial score, the α coefficient plays a vital role in deciding whether to execute in parallel or in series.

In cases where the execution time has more importance, the effect of the parallel score is more significant. In contrast, serial execution leads to a better leasing price which is effective in cases where cost is of more importance. Of course, this is affected by the time slots provided by the service provider.

The proposed algorithm has several features which affect the execution time and cost of the workflow. The parallel execution of the clusters with high Time Overlaps and reducing the intercommunications of the combined clusters to zero and also reducing the free time gaps in the scheduling of the resources greatly reduces the makespan and the execution cost of the workflow as well.

4. Performance Evaluation

In this section the simulation results of the Cluster Combining Algorithm are presented. To test the proposed approach, five well known scientific workflows that have been used as benchmarks by researchers [25] [26] were used in order to evaluate the performance of the scheduling algorithms.

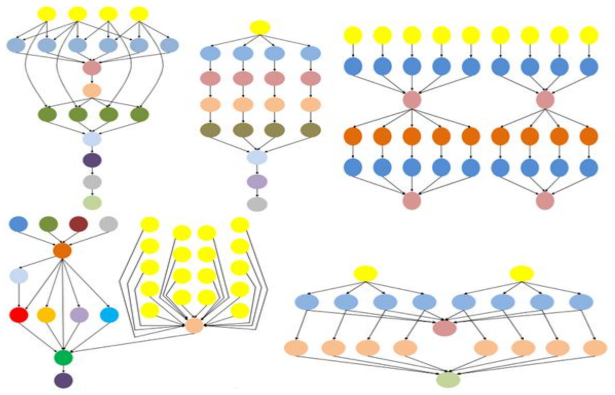

Fig. 3. Scientific workflow DAGs (top row from left: Montage, Epigenomics, LIGO. Bottom row from left: SIPHT, CyberShake)

TABLE 1. : RESOURCE SPECIFICATIONS AND BILLING PRICES

	Vcpu	*Memory (GiB)*	*Windows Usage (per hour)*
M3.medium	*1*	*3.75*	*$0.133*
C3.large	*2*	*3.75*	*$0.188*
C3.Xlarge	*4*	*7.5*	*$0.376*
C3.2Xlarge	*8*	*15*	*$0.752*
C3.4Xlarge	*16*	*30*	*$1.504*
C3.8Xlarge	*32*	*60*	*$3.008*

These benchmarks are based on real scientific workflows in different fields like physics, astronomy, genetics, etc. that have different sizes (i.e. number of tasks): LIGO, SIPHT, Montage, Epigenomics and Cybershake. Figure 3 shows the related DAGs of small samples or similar to these workflows.

Table I shows the computing resources and their related leasing prices for 1-hour time slots that we have assumed can be provided by the service provider. These leasing prices are based on the Amazon Elastic Cloud2 (EC2) pricing policy. Therefore, one hour billing periods were considered in the proposed scheduling algorithm.

To evaluate the CCA algorithm, a deadline must be defined for each workflow. Therefore, we have assigned the EFT of the exit task of the workflow as the workflows deadline.

A deadline coefficient that is used to set different deadlines for the workflows and is computed as α.EFTExit is also considered.

The results of the experiments carried out in this study show that the CCA algorithm schedules all the workflows before the termination of their deadlines. To compute the success rate of the proposed method, we have considered deadline coefficients between 0.8 and 1.4 and the results show that the success rate of the proposed method is 100% and there are no deadline violations.

The proposed algorithm was also compared with the PCP algorithm proposed by Abrishami et al. [1] and the PCH algorithm proposed by Bittencourt and Madeira [20] which are two of the most cited algorithms in this context. The PCP algorithm divides the workflow into sequences using the Partial Critical Path concept and maps these sequences onto single core processing resources to be executed in a parallel or serial manner. The authors of the Path Clustering Heuristic (PCH) have also proposed a clustering heuristic to divide the workflow into sequences of tasks in such a way that all of the parents of each task in the sequence have been or are to be scheduled with the sequence. These sequences are then mapped onto single core processing resources to be executed.

We have regarded different sizes of the workflows as input and the results show that the proposed algorithm has a superior performance in terms of both time and cost.

To ease the comparison with other methods we have used the logarithmic function to scale the results. The first four charts show the makespan of the method and the other charts show the execution cost. The total billing price of the processing resources to execute the workflow is denoted as the execution cost. The virtual machine specifications used in the implementation are according to the Amazon EC2 service provider given in Table 1.

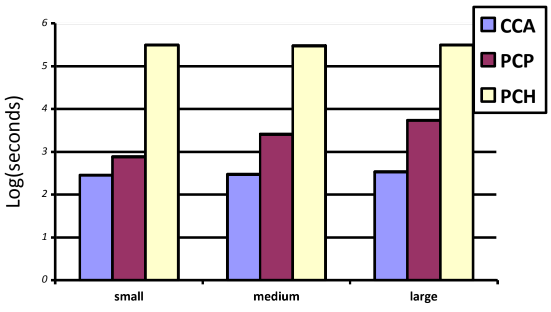

Fig. 4. Makespan comparison on Cybershake workflow with 30, 50 and 100 nodes

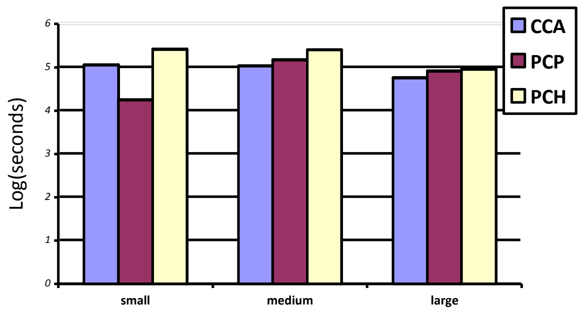

Fig. 5. Makespan comparison on Epigenomics workflow with 24, 46 and 100 nodes

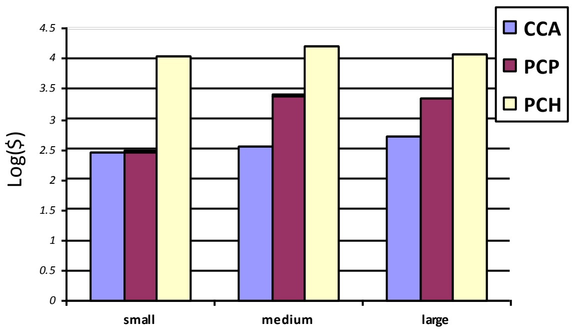

Fig. 9. Execution cost comparison on Epigenomics workflow

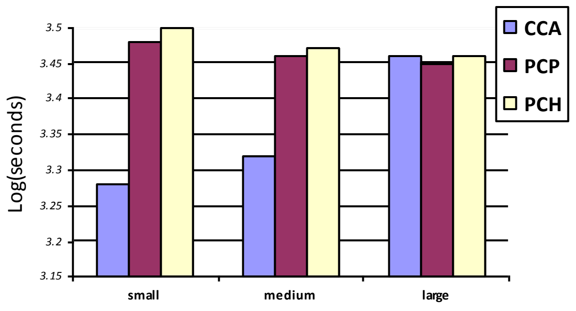

Fig. 6. Makespan comparison on Inspiral workflow with 30, 50 and 100 nodes

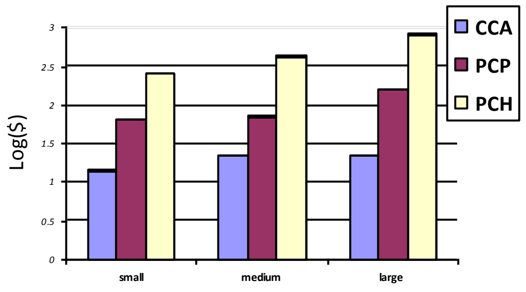

Fig. 10. Execution cost comparison on Inspiral workflow

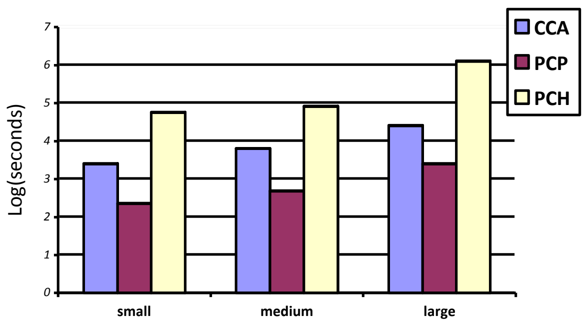

Fig. 7. Makespan comparison on Montage workflow with 25, 50 and 100 nodes

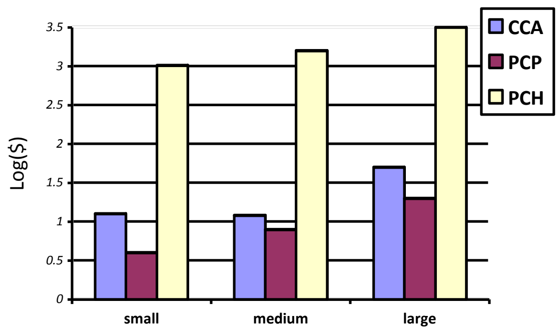

Fig. 11. Execution cost comparison on Montage workflow

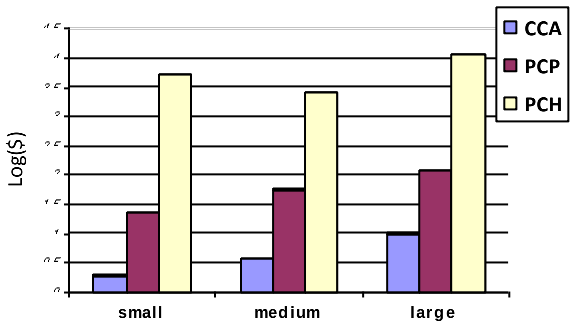

Fig. 8. Execution cost comparison on Cybershake workflow

In most cases, the results show that the CCA algorithm performs better in comparison with the two other methods. In the small Epigenomics workflow, the CCA algorithm considers the serial execution of this workflow because of the structure of the workflow. This serial execution leads to more free time gaps on the leased resources in the scheduling of the tasks which causes a higher makespan and execution cost. This problem has been overcome in the medium and large Epigenomics workflows. This is because in the medium and large cases, more tasks can be parallelized using available leased resources. In other words, other tasks can be scheduled on the free time gaps. Therefore, more tasks can be scheduled on the available time gaps. The structure of the Montage workflow shows that this workflow is made up of small clusters with low Time Overlaps which increases the free time gaps in the scheduling of the resources. Therefore, this increases the makespan and the execution cost. This also increases the number of resources that are needed for the execution of the workflow in the proposed approach in comparison with the PCP algorithm. However, the CCA algorithm performs better compared to the PCH algorithm in this case.

The main reason that the proposed workflow scheduling approach yields superior results in makespan and cost is that by scheduling multiple clusters on a multi-core processor, the communication between these clusters are reduced to zero, especially if resources with a higher number of cores are used. Thus having processing resources with more cores leads to more parallel execution of the workflow's clusters. In addition, mapping more clusters onto the same resource leads to reducing more of the communication between these clusters to zero, and this has an important impact on the makespan. Therefore, workflows with a higher number of tasks have more clusters that can be parallelized and executed on processors with a higher number of cores. As the number of tasks increases, we can imply that this zeroing of communication has a greater effect on the makespan of bigger workflows. We have also tried to maximize the utilization of the processing cores by reducing free time gaps. This approach reduces the number of needed resources to process the workflow.

5. Conclusion

Cloud Computing is considered to be a relevant platform for executing HTC applications like workflows. Thus scheduling workflows on the Cloud is considered to be an important problem to be solved. Service providers offer multi-core processing resources. Because of the payment policy in the Cloud, cost and makespan are important factors which must be carefully considered in the scheduling algorithm. In this research, a new workflow scheduling algorithm that considers multi-core processing resources with especial attention to execution time and cost was proposed. To execute the workflow on multi-core processors, the workflow was divided into clusters and was combined in such a way that can be executed in parallel or in series. Therefore, a new concept called Time Overlap has been presented in this study that was used to combine the clusters as well as to decide the parallel or serial execution of these clusters. Moreover, a scoring approach has been proposed to combine the clusters. Clusters with a high time overlap gain a high parallel score and are executed in parallel and clusters with a low time overlap gain a high serial score and are considered to be run in series. The general tendency is to use resources with a higher number of cores. This leads to more parallel execution and less communication overhead. In cases where the scheduling algorithm executes clusters in serial, the general tendency is to use the available free time gaps.

The proposed approach has been compared with two other well-known methods and the results showed that especially in cases that the structure of the workflow has better characteristics for parallel execution, this algorithm has a superior performance in terms of time and cost. These types of workflows are mapped onto processing resources with a higher number of cores. Therefore, more communication overhead will be reduced to zero which greatly reduces the makespan of the workflow. The general tendency in the proposed approach is to increase the utilization of the processing resources by reducing the free time gaps in the scheduling algorithm which leads to a lower number of resources for the execution of the workflow. Thus, this results in a reduction of the leasing costs.

References

[1] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Cost-driven scheduling of grid workflows using partial critical paths," Parallel IEEE Trans. on Distrib. Syst., vol. 23, no. 8, pp. 1400–1414, 2012.

[2] L. F. Bittencourt and E. R. M. Madeira, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," J. Internet Serv. Appl., vol. 2, no. 3, pp. 207–227, 2011.

[3] R. G. Michael and S. J. David, "Computers and intractability: a guide to the theory of NP-completeness," WH Free. Co., San Fr., 1979.

[4] A. Abraham, R. Buyya, B. Nath, and others, "Nature's heuristics for scheduling jobs on computational grids," in The 8th IEEE international conference on advanced computing and communications (ADCOM 2000), pp. 45–52, 2000.

[5] A. K. Aggarwal and R. D. Kent, "An adaptive generalized scheduler for grid applications," in 19th International Symposium on High Performance Computing Systems and Applications, HPCS 2005., pp. 188–194, 2005.

[6] M. Aggarwal, R. D. Kent, and A. Ngom, "Genetic algorithm based scheduler for computational

grids," in 19th International Symposium on High Performance Computing Systems and Applications, HPCS 2005., , pp. 209–215, 2005.

[7] A. H. Alhusaini, V. K. Prasanna, and C. S. Raghavendra, "A unified resource scheduling framework for heterogeneous computing environments," in Proceedings. Eighth Heterogeneous Computing Workshop, 1999.(HCW'99), pp. 156–165, 1999.

[8] R. Bajaj and D. P. Agrawal, "Improving scheduling of tasks in a heterogeneous environment," IEEE Trans.Parallel Distrib. Syst., vol. 15, no. 2, pp. 107–118, 2004.

[9] S. K. Garg, C. S. Yeo, A. Anandasivam, and R. Buyya, "Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers," J. Parallel Distrib. Comput., vol. 71, no. 6, pp. 732–749, 2011.

[10] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," Futur. Gener. Comput. Syst., vol. 28, no. 5, pp. 755–768, 2012.

[11] A. J. Younge, G. Von Laszewski, L. Wang, S. Lopez-Alarcon, and W. Carithers, "Efficient resource management for cloud computing environments," in International Green Computing Conference , pp. 357–364, 2010.

[12] A. Nathani, S. Chaudhary, and G. Somani, "Policy based resource allocation in IaaS cloud," Futur. Gener. Comput. Syst., vol. 28, no. 1, pp. 94–103, 2012.

[13] W. Wang, G. Zeng, D. Tang, and J. Yao, "Cloud-DLS: Dynamic trusted scheduling for Cloud computing," Expert Syst. Appl., vol. 39, no. 3, pp. 2321–2329, 2012.

[14] M. E. Frîncu, "Scheduling highly available applications on cloud environments," Futur. Gener. Comput. Syst., vol. 32, pp. 138–153, 2014.

[15] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," J. Parallel Distrib. Comput., vol. 59, no. 2, pp. 107–131, 1999.

[16] K. Etminani and M. Naghibzadeh, "A min-min max-min selective algorihtm for grid task scheduling," in 3rd IEEE/IFIP International Conference in Central Asia on Internet, ICI 2007., pp. 1–7, 2007.

[17] H. Topcuoglu, S. Hariri, and M. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Trans. Parallel Distrib. Syst., vol. 13, no. 3, pp. 260–274, 2002.

[18] T. Yang and A. Gerasoulis, "A fast static scheduling algorithm for DAGs on an unbounded number of processors," in Proceedings of the 1991 ACM/IEEE conference on Supercomputing, pp. 633–642, 1991.

[19] V. Sarkar, Partitioning and scheduling parallel programs for multiprocessors. MIT press, 1989.

[20] L. F. Bittencourt and E. R. M. Madeira, "A performance-oriented adaptive scheduler for dependent tasks on grids," Concurr. Comput. Pract. Exp., vol. 20, no. 9, pp. 1029–1049, 2008.

[21] L. F. Bittencourt and E. R. M. Madeira, "Towards the scheduling of multiple workflows on computational grids," J. grid Comput., vol. 8, no. 3, pp. 419–441, 2010.

[22] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds," Futur. Gener. Comput. Syst., vol. 29, no. 1, pp. 158–169, 2013.

[23] D. Poola, S. K. Garg, R. Buyya, Y. Yang, and K. Ramamohanarao, "Robust scheduling of scientific workflows with deadline and budget constraints in clouds," in The 28th IEEE International Conference on Advanced Information Networking and Applications (AINA-2014), pp. 1–8, 2014.

[24] H. Kanemitsu, M. Hanada, T. Hoshiai, and H. Nakazato, "Effective use of computational resources in multi-core distributed systems," in 16th International Conference on Advanced Communication Technology (ICACT), 2014, pp. 305–314, 2014.

[25] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, and others, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," Sci. Program., vol. 13, no. 3, pp. 219–237, 2005.

[26] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in Third Workshop on Workflows in Support of Large-Scale Science, 2008. WORKS 2008., pp. 1–10, 2008.