Cost and Efficiency-based Scheduling on a General Framework Combining between Cloud Computing and Local Thick Clients

Nguyen Doan Man
College of Electronics and Information
Department of Computer Engineering
Kyung Hee University
Republic of Korea
Email: doanman1985@gmail.com

Eui-Nam Huh
College of Electronics and Information
Department of Computer Engineering
Kyung Hee University
Republic of Korea
Email: johnhuh@khu.ac.kr

***Abstract*—Today, the rapid growth of the large-scale applications conduces to a difficult challenge to individuals as well as the commercial organizations due to the limitation of the local computing resources. Meanwhile, the extension of the local computing platforms requires a huge investment about both finance and human power. Therefore, the use of nearly-unlimited resources and on-demand scaling of Cloud computing is considered as a potential solution to address the problems above. However, since Cloud computing is built from a pay-as-you-go model, the novel application scheduling approaches are challenged to guarantee the high efficiency of application execution and the reasonable cost for renting Cloud resources. In this paper, we present a novel framework built from the combination between the computing resources on Cloud computing and the computing components in the local systems. The key component of this framework is the Cost with Finish Time-based scheduling algorithm, which provides the balance between performance of application schedule and the mandatory cost for the use of Cloud resources. The experiments and comparison with the other scheduling approaches demonstrated the potential benefits of our proposed algorithm.**

I. Introduction

Nowadays, the quantity and the scale of business workflows or applications that companies or commercial organizations have to proceed each day are growing rapidly and, thus, create a huge pressure for the small-scale individual data centers. The emerging of Cloud computing [1], [3] really attracted to multiple organizations and companies since Cloud computing promises the important benefits ,e.g. nearly-unlimited computing resources, on-demand scaling and pay-per-use metered services, for efficiently executing their large-scale offloading applications. Therefore, the most important challenge for Cloud customers (CCs) is only finding how to utilize their rented Cloud resources in the most efficient way.

Each large-scale workflow is a set of multiple tasks or activities with the diversity of the volumes of workloads. Therefore, the efficient workflow execution is related to find the best schedule to deploy activities of a workflow on multiple processing systems. A workflow scheduling problem as above has been extensively studied for a long time and various heuristics were proposed. Especially, in the heterogeneous environments such as Cloud computing, workflow scheduling becomes a NP-Complete problem and it's impossible to find the optimal solutions without an exhaustive search. For most of the previous scheduling algorithms, the main objective is to minimize the completion time of workflows. However, the recent proposed frameworks where workflow execution can be outsourced to the rented computing resources (i.e. virtual machines (VMs)) of different Cloud providers (CPs) presented an additional objective for the scheduling problem: reducing cost for the use of external resources. The reason is that CPs charge their customers for workflow execution on their own infrastructures. For example, in Amazon EC2 [4], the charging policy is based on the number of virtual instances that CCs initialized and how long CCs used them. Sometimes a workflow schedule can provide a good performance, but requires a large amount of the monetary cost for the external resources. To the budget of a company, this is not a good proposal. This motivates us to find a novel scheduling approach to balance between performance and cost for rented resources.

In this paper, we present a framework built from the combination between virtualized computing platforms as services on Cloud computing and thick clients (TCs) residing on the local systems of CCs. As definition, TCs have been used for providing rich user interface functionalities and also for mobile employees who would like to execute functions without connecting to the network or firewall [2]. In our framework, TCs are also available to process the tasks that are dispatched to them as well as store a specific amount of user data for workflow execution if necessary. PCs, tablets or smartphones are considered as the most popular TCs. Resource Broker, a resource management component in this framework, takes responsible of receiving and solving the scheduling problem for the workflows which are submitted from users at the CCs' side. The major material for solving the scheduling problem in this framework is a novel scheduling heuristics, Cost with Finish Time-based (CwFT), an extension of the popular Heterogeneous Earliest Finish Time (HEFT) scheduling algorithm. A novel utility function, which is based on the performance of scheduling plan and the resource prices, is used to select the best processing unit for each task of a workflow. Our experimental results show CwFT brings CCs the significant cost savings along with a reasonable performance of workflow execution.

The remainder of our paper is organized as follows: In the next section, we define some related works on the scheduling problem for the distributed and heterogeneous environments. The architecture of our framework as well as the important definitions for a scheduling problem will be mentioned in Section 3. Section 4 presents our scheduling algorithm CwFT. The experiments and corresponding results are the main content in the Section 5. We will briefly summarize our work and future works in the last section.

II. Related Works

In the heterogeneous systems such as Grid or Cloud computing, scheduling algorithms play a key role in obtaining high performance. The major objective of scheduling is to map tasks of a workflow onto computing systems and order their executions in order to achieve the minimal schedule length. The most popular presentation of a workflow is directed acyclic graphs (DAG) in which each vertex represents a task and each directed edge represents the precedence constraint. Several scheduling algorithms has been proposed to deal with heterogeneous systems, for example, dynamic-level scheduling [11], levelized-min time [7], Critical Path on Processor [13]. Comparison of the different DAG scheduling algorithms was mentioned in the survey papers [8], [9].

Compared with the algorithms above, the HEFT scheduling algorithm [13] conduced to the better performance about both the time complexity and the completion time (called as makespan). In the first phase of HEFT, each task is set a priority value based on the upward rank value of this task in the workflow. Processor selection begins from the highest-priority task in the sorted list of the unscheduled tasks and finishes when all tasks are assigned. In this case, each task is assigned to the processor which minimizes the completion time of this task. The time complexity of HEFT algorithm is $O(e.q)$ for e edges and q processors. Due to the high efficiency, HEFT was extensively applied in the [5] to meet not only the performance requirement but also the security constraints. In this work, the computation cost of a task in the security-driven scheduling problem includes cost for task execution but also the security overhead. Hence, the decision to assign a task to a computing node is impacted by the security levels of this node.

For Cloud computing, the multitask workflow scheduling problem aims to find the best schedule to either achieve the good performance of workflow execution or satisfy the budget constraint. The scheduling algorithm CCSH [15] is an extension from the HEFT algorithm to schedule the application of large graph processing with considering both cost and schedule length. The impact of cost for Cloud resources is represented

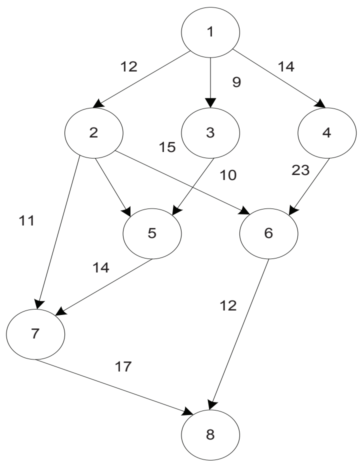

Fig. 1. A sample DAG

by the input cost-conscious factor and used as a weight in the computation of the earliest finish time EFT of each task. Meanwhile, the strictly budget constraints could be satisfied by the ScaleStar [12] algorithm as workflow execution was deployed by Cloud resources. In this case, a novel objective function *Comparative Advantage* (CA) was presented. In the first phase, each task is assigned to the best VM that provides the maximum CA1 value to achieve a good balance between cost savings and efficiency. The budget constraint is considered at the task reassignment phase where the CA2 function is used as the basis to evaluate the improvement of task reassignment to another VM on both criteria, i.e. cost and performance. However, the high complexity of ScaleStar can hinder it to be applied to the large-scale workflows.

Our work can be closest to studies in [6], [14]. The hybrid Cloud model [3], which are generated from a private Cloud and multiple public Clouds, is selected to deploy the large-scale workflows. Authors presented a $Cost-oriented$ scheduling algorithm, a cost-efficient approach with the deadline constraint, to determine the most appropriate system (i.e. public Cloud or private Cloud) for executing the incoming workflows. Decisions are also based on the ability of meeting the deadline of each workflow as well as the cost savings. However, in this work, each application can be processed at only public Clouds or private Cloud, which is different from our framework where workflow execution can be dispersed on both VMs on Cloud and TCs at the local system of CCs.

III. Problem Model

Each workflow is represented by a DAG $G = (V, E)$, where the set of vertices $V = \{v_1, v_2, ..., v_k\}$ represents the set of parallel tasks, and the directed edges $e_{ij} = (v_i, v_j)$ represents communication or the precedence constraint between tasks v_i and v_j. We assume that G has an entry task , v_{entry}, which does not have any predecessors and an exit task, v_{exit}, without any successors. The workload of each task v_i is denoted as w_i to specify the amount of works (e.g the number of instructions) which are processed at the computing resources. Each edge e_{ij} is also set a weight cd_{ij} which represents the amount of data from the preceding task v_i and be considered as input data for executing task v_j. The sufficient input data of each task, in fact, are collected from not only the preceding tasks but also dispersed to multiple other storage sources in the computing infrastructure. However, in the previous approaches, these data sources are aborted to simplify the scheduling problem. In our work, each task v_i can only be executed if all input data from inter-task communications as well as Cloud storages or the fixed storages of TCs were gathered sufficiently.

For the computing resources, we assume that an organization or a home user owns a large computing infrastructure which is divided into two parts; namely, the set N_{in} of m TCs located in the local system and the set N_{out} of heterogeneous VMs residing at n different Clouds. We consider all TCs and VMs as a set $N = N_{in} \cup N_{out}$ of the processing nodes. Each task of a workflow can be executed on any processing node n_i ($n_i \in N$) at the processing rate p_i and the computation power of Cloud nodes n_j ($n_j \in N_{out}$) is always higher than which of TCs n_i ($n_i \in N_{in}$), i.e. $p_j = k * \max_{n_i \in N_{in}} (p_i)$ where $k \geq 2$.

Like the previous scheduling problems, workflow execution also requires communication or data transfer among processing nodes in the whole computing infrastructure. It means that the output data of a task sometimes may be the input of its successors, which is one of the explanations for the precedence constraints. Therefore, for communication between Cloud nodes and other systems, CCs need to reserve a specific amount of network resources at all Clouds. Let bw_i be the data transfer rate between Cloud node n_i ($n_i \in N_{out}$) and the local system; meantime, bw_{in} denotes the average data transfer rate among local TCs. Due to the high stability of Local Area Network (LAN) compared with the Internet, the average data transfer rate of internal communication is always better than that of external communications or $bw_{in} > \max_{n_i \in N_{out}} (bw_i)$.

Because computing resources of CCs are dispersed into multiple Cloud nodes as well as TCs in the local network, there is a need for a centralized management component to receive and process all computation requests of users at the CC's side efficiently. Hence, Resource Broker (RB), an independent management module located at the private system of CCs, is responsible for collecting and managing profiles about not only TCs but also VMs and then producing the most appropriate schedule for each user workflow based on those profiles. When submitting a workflow to RB, a user should specify the necessary parameters of the workflow such as the number of tasks, workload and input data of each task, etc. Based on information about the input data of each task, RB will do queries for the location of relevant data which Cloud nodes and TCs are storing. Profiles about processing capacity and network bandwidth of all nodes as well as computation and communication costs together with results of data query returned from nodes are the mandatory input of the task scheduling problem. RB deploys the scheduling algorithms to find the workflow schedule and then dispatches tasks to the appropriate nodes, which aims to achieve the tradeoff between cost for Cloud resources and the workflow makespan. The architecture of a RB is shown in Fig.2.

Due to the pay-as-you-go pricing model of Cloud computing, CPs charge CCs an amount of monetary costs for using their Cloud resources. Assume that each CC must pay rc_i per a time unit of workflow execution at Cloud node n_i, as well as dc_i per a unit of outgoing data (e.g. MB or GB) from Cloud node n_i. Let et_i^j denote the time to process workload of task v_i at a node n_j and be defined by:

$$et_i^j = \frac{w_i}{p_j} \tag{1}$$

Therefore, the computation cost of a task v_i at Cloud node n_j can be estimated as follow:

$$pc_i^j = et_i^j * rc_j \tag{2}$$

In the workflow scheduling problem, the communication time and cost are also the important measurements that significantly contribute to the decision of node selection for processing each task. In this framework, we assume data that are exchanged between two certain nodes are always encrypted at the local system to achieve the high-secured communication. Thus, the path that data are transferred from a source node to a target node can be divided into two parts: from the source node to the encryption node at the CC's side and from the encryption node to the target node. We denote d_i^j as the amount of input data stored at node n_j and used for executing task v_i. Let dt_i^{jk} represent the data transfer time between two nodes n_j and n_k for executing v_i at n_k.

Case 1: $n_j \rightarrow n_k$ where $n_j \in N_{out}$ and $n_k \in N_{in}$

$$dt_i^{jk} = \left(d_i^j + \sum_{v \in exec(j)}^{v \in prec(i)} cd_{vi} \right) * \left(\frac{1}{bw_j} + \frac{1}{bw_{in}} \right) \tag{3}$$

where *prec(i)* is the set of the preceding tasks of v_i and *exec(j)* is the set of tasks executed at the node n_j.

Case 2: $n_j \rightarrow n_k$ where $n_j, n_k \in N_{out}$

$$dt_i^{jk} = \left(d_i^j + \sum_{v \in exec(j)}^{v \in prec(i)} cd_{vi} \right) * \left(\frac{1}{bw_j} + \frac{1}{bw_k} \right) \tag{4}$$

Case 3: $n_j \rightarrow n_k$ where $n_j, n_k \in N_{in}$

$$dt_i^{jk} = \left(d_i^j + \sum_{v \in exec(j)}^{v \in prec(i)} cd_{vi} \right) * \frac{2}{bw_{in}} \tag{5}$$

Case 4: $n_j \rightarrow n_k$ where $n_j \in N_{in}$ and $n_k \in N_{out}$

$$dt_i^{jk} = \left(d_i^j + \sum_{v \in exec(j)}^{v \in prec(i)} cd_{vi} \right) * \left(\frac{1}{bw_{in}} + \frac{1}{bw_k} \right) \tag{6}$$

Cost for the amount of outgoing data from a specific Cloud node n_j to process task v_i is specified by:

$$tc_i^j = \left(d_i^j + \sum_{v \in exec(j)}^{v \in prec(i)} cd_{vi} \right) * dc_j \tag{7}$$

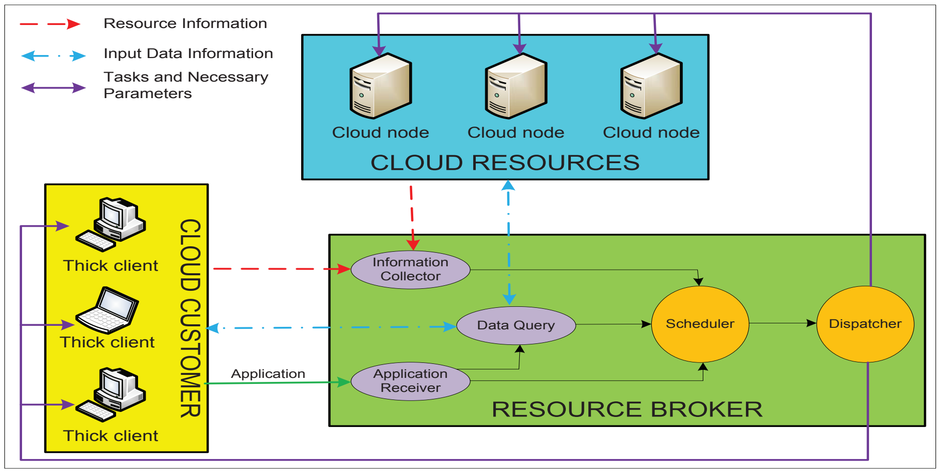

Fig. 2. Architecture of Resource Broker

IV. Cost with Finish Time-based Algorithm

The CwFT algorithm is a workflow scheduling algorithm extended from the HEFT algorithm for distributed environments with multiple heterogeneous processing nodes. Instead of optimizing only the workflow makespan as usual, CwFT algorithm also considers reducing the monetary cost that CCs need to pay in a computing framework with the combination between numerous Cloud node and a local system. Similar to HEFFT, the CwFT algorithm is comprised of two phases: *Task Prioritizing* to mark the priority level for all tasks and *Node Selection* to select tasks in a descending order by the priority level and then schedule each selected task on an appropriate processing node to optimize the value of the utility function.

A. Task Prioritizing Phase

In our algorithm, the priority of a task v_i is estimated by the length of the critical path from task v_i to the exit task, including the computation time of v_i. The critical path of a workflow is the longest execution path between the entry and the exit tasks of the workflow. Thus, the priority value of task v_i is defined as follow:

$$pri(i) = \begin{cases} \overline{et_i} + \max\limits_{j \in succ(i)} [\overline{dt_{ij}} + pri(j)] & v_i \neq v_{exit\_task} \\ \overline{et_i} & v_i = v_{exit\_task} \end{cases} \tag{8}$$

where *succ(i)* is the set of v_i's successors. $\overline{et_i}$ and $\overline{dt_{ij}}$ are the average computation time of task v_i and the average data transfer time between tasks v_i and v_j, respectively:

$$\overline{et_i} = \frac{w_i}{\left(\sum\limits_{n_k \in N} p_k\right) / (n+m)}$$

$$\overline{dt_{ij}} = \frac{dt_{ij}}{\left[\left(\sum\limits_{n_k \in N_{out}} bw_k\right) + bw_{in}\right] / (n+1)}$$

As Equation (8), the priority is defined recursively by traversing the workflow DAG from the exit to the entry task.

Finally, we sort the list of all tasks with a descending order. A benefit of priority values and then ordering tasks in the decreasing order is to provide a topological order of tasks, which preserves the precedence constraints.

B. Node Selection Phase

In the CwFT algorithm, we use two parameters *Earliest Execution Starting Time (EST)* and *Earliest Execution Finish Time (EFT)*, which are mentioned in the HEFT algorithm, but also present another important parameter for task scheduling, *Data Transfer Time (DDT)*. Since the preparation for input data of a task v_i is only started when all preceding tasks of v_i are completed, *DDT(i)* represents the time when the last preceding task of v_i is completed.

$$DTT(i) = \begin{cases} \max\limits_{j \in prec(i)} [AFT(j)] & v_i \neq v_{entry\_task} \\ 0 & v_i = v_{entry\_task} \end{cases} \tag{9}$$

where $AFT(j)$ is the actual finish time of task v_j.

As input data that are collected from all data sources on either Cloud nodes or TCs arrived at the target node, task execution will begin. Thus, the EST and EFT values of task v_i executed on a node n_j are computed recursively as follow:

$$EST(i,j) = \max\left\{avail(j), DTT(i) + \max_{n_k \in N}(dt_i^{kj})\right\} \tag{10}$$

$$EFT(i,j) = et_i^j + EST(i,j) \tag{11}$$

where *avail(j)* is the earliest time that node n_j completes the last assigned task and be ready to execute another one. After a task v_i is scheduled on a processing node n_j, the actual finish time, *AFT(i)*, is equal to the $EFT(i,j)$ value.

In addition to the finish time, the CwFT also considers the cost which each CC needs to pay for Cloud resources that are used to execute tasks. Cost for executing on Cloud nodes is comprised of two parts: computation cost pc_i^j and communication cost tc_i^j, which are computed as Equation (2) and (7) respectively. In contrast, if a task is assigned to a TC, CC only needs to pay for the amount of data transferred from Cloud nodes to the target node in the local system. Thus, the total cost for executing task v_i on a specific node n_j can be defined by:

$$cost(i,j) = \begin{cases} pc_i^j + \sum\limits_{n_k \in N_{out}} tc_i^k & \text{if } n_j \in N_{out} \\ \sum\limits_{n_k \in N_{out}} tc_i^k & \text{if } n_j \in N_{in} \end{cases} \tag{12}$$

Replacing the *EFT* value in the HEFT algorithm, the utility function *Cost-on-Makespan Ratio (CMR)* is proposed as a criterion for node selection in the CwFT algorithm. For the CwFT algorithm, the "best" processing node of each task is the one that provides the best tradeoff between cost for the use of Cloud resources to execute that task and the task makespan. For this objective, RB will assign a task v_i to node n_j which maximizes the *CMR* value of task v_i. As task v_i is executed on node n_j, the function *CMR* can be defined as follow:

$$CMR(i,j) = \frac{\min\limits_{n_k \in N}[cost(i,k)]}{cost(i,j)} \text{x} \frac{\min\limits_{n_v \in N}[EFT(i,v)]}{EFT(i,j)} \tag{13}$$

Hence,

$$best\_node(i) = \underset{n_j \in N}{\arg\max}[CMR(i,j)] \tag{14}$$

V. Experiments

A. Scheduling on Gaussian Elimination Program

In this experiment, we applied the CwFT algorithm into addressing the scheduling problem for Gaussian Elimination (GE) program [10], which is considered as a scheduling problem in the real world, and then compared the performance of CwFT with that of other algorithms *Greedy on Cost*, where each task of the workflow is assigned to the processing node which minimizes cost for Cloud resources to execute that task, and *HEFT*. The number of tasks in a GE program with matrix size m is equal to $\frac{m^2+m-2}{2}$. In our experiment, matrix size is raised from *10* to *80* with the increasing steps of *10*. We also simulated a *64*-node computing model which is a combination between *42* VMs with the different configurations and *22* TCs located at the local system of CCs.

Fig.3(a), (b) illustrates the results of scheduling the GE program at the different matrix sizes. Generally, despite the highest cost savings on the use of Cloud resources, the Greedy algorithm obtained the worst results in term of workflow makespan. In contrast, workflow schedule produced by the HEFT algorithm always provides the best performance compared with two other algorithms; however, its efficiency goes with the significant increase of cost. Meantime, the CwFT algorithm conduced to the benefits of balance between the acceptable schedule length for workflows and the corresponding cost for Cloud resources. Specifically, to achieve the better makespan from *12%* to *34%* than Greedy, the CwFT algorithm required the extension of the total consumption no more than *10%*. On the other side, compared with HEFT, CCs can save nearly *25%* to *42%* of cost while reduction on performance is not over *25%*.

Note that VM or TC selections for task execution also rely on the utility function of each algorithm. As Fig.3(c), because HEFT is a performance-intensive algorithm, most of tasks (i.e. more than *90%*) in the GE program are scheduled to VMs where provide the higher processing capability than TCs. Meantime, due to considering the amount of the monetary cost for Cloud resources, both the Greedy and CwFT algorithm produced the task schedules where the majority of tasks had been executed at the TCs to obtain the considerable cost savings. However, in the model using the greedy or CwFT algorithm, there is a trendy increase on the percentage of tasks executed on Cloud nodes along with the increasing number of tasks due to the augmentative complexity of these workflows.

B. Effect of Cloud Computing on the CwFT Algorithm

We evaluated the impact of the quantity of VMs on the quality of workflow schedule, the output of the CwFT algorithm. In the computing framework mentioned in Section III, the experimental results can help CCs to determine the most appropriate number of VMs that should be used to satisfy the user-defined deadline or budget constraints. The responsibility of determining the number of VMs belongs to the RB component, which owns all necessary information about either VMs or TCs and provides the solution for the scheduling problem. The number of processing nodes is still fixed at *64*. However, we defined the number of VMs at three levels: *22*, *32* and *42*, respectively. Also, we randomly used several large-scale workflows with over *3000* tasks to run on three computing frameworks and then estimated the workload makespan and monetary costs with the CwFT algorithm.

The experimental results shown in Fig.3(d) demonstrated the quantity of VMs is an important factor to determine the best schedule for a specific workflow. The more the number of high-capacity VMs is, which provide the better processing capability as well as the large amount of available input data, the more tasks are dispatched to Cloud nodes. The total workflow makespan is shortened or the efficiency of workflow schedule is improved. However, offloading workflows to Cloud resources frequently also corresponds to the considerable increase of outsourced costs. Hence, for the budget constraints, the decreasing proportion of VMs out of the total number of processing nodes is considered as a potential solution. Conversely, the expansion of Cloud infrastructure should be deployed to meet the deadline for workflow execution.

VI. Conclusion

In this paper, we introduced a novel computing framework which includes the nearly unlimited processing capacity of Cloud computing and the individual computing resources of organizations or companies which are represented by TCs in the local system. With this framework, CCs can take advantage of the total computing power from both internal and external infrastructures. However, this model also makes increase on the complexity of finding an optimal solution for the workflow

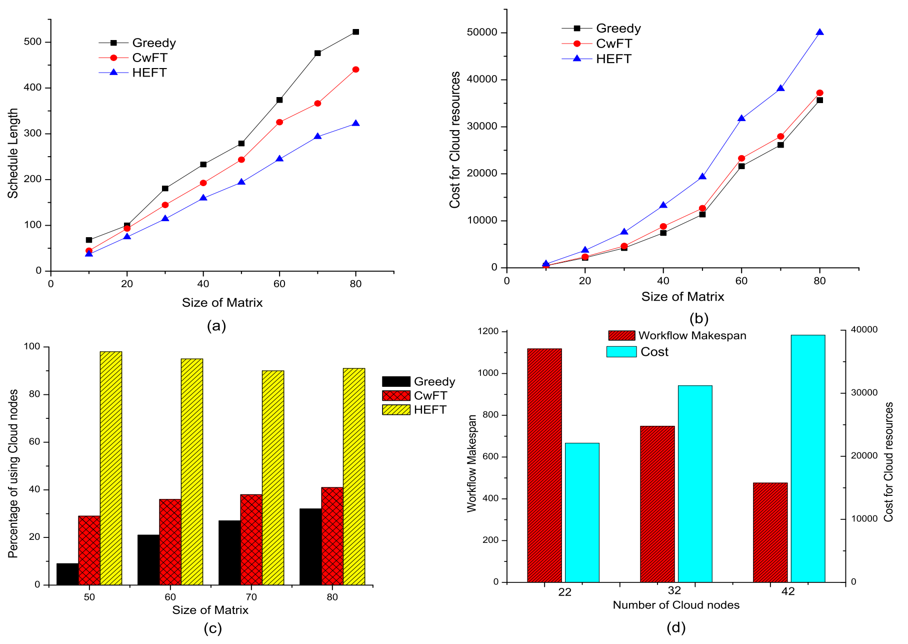

Fig. 3. Comparison between three scheduling algorithms HEFT, CwFT and Greedy on (a) Schedule Length, (b) Cost for using Cloud resources, (c) Percentage of the used VMs; (d) Effect of the quantity of VMs on performance and cost of the workflow schedule

scheduling problem when we confront with a higher level of the heterogeneity on the computing, communication and storage capacity of processing nodes. The CwFT scheduling algorithm with considering both cost savings and workflow performance was proposed to produce the workflow schedule in such computing environments. The experiments demonstrated that the CwFT algorithm provided the better tradeoff between the efficiency of workflow execution and the amount of cost that CCs have to pay for CPs. Furthermore, we also showed the important role of Cloud computing on producing the output schedule by the CwFT algorithm. In future, the budget and deadline constraints should be defined to improve the practicalness of the workflow scheduling problem as well as the quality of the output schedule.

Acknowledgment

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012-0006418). Professor Eui-Nam Huh is the corresponding author.

References

[1] B. Furht and A. Escalante, *Handbook of Cloud Computing*, 1st ed. Springer, 2010.

[2] Srinivasan S. Rajan, *Why Thick Clients Are Relevant in Cloud Computing*. http://cloudcomputing.sys-con.com/node/1694221.

[3] *The NIST definition of Cloud computing*. National Institute of Standards and Technology.

[4] *Amazon Web Services*. http://aws.amazon.com/ec2/.

[5] X. Tang, K. Li, Z. Zeng and B. Veeravalli, *A Novel Security-Driven Scheduling Algorithm for Precedence-Constrainted Tasks in Heterogeneous Distributed Systems*. IEEE Trans. on Computers, Vol.60, No.7, pp 1017-1029, July 2010.

[6] Ruben V. den Bossche, K. Vanmechelen and J. Broeckhove, *Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads*. 2010 IEEE 3rd International Conference on Cloud Computing, pp 228-235, October 2010.

[7] Michael A. Iverson, F. Ozguner and Gregory J. Follen, *Parallelyzing Existing Applications in a Distributed Heterogeneous Environment*. The 4th Heterogeneous Computing Workshop, pp 93-100, 1995.

[8] Tracy D. Braun, Howard J. Siegel, N. Beck, Lasislau L. Boloni, M. Maheswaran, Albert I. Reuther, James P. Robertson, Mitchell D. Theys, B. Yao, D. Hensgen and Richard F. Freund, *A Comparison of Eleven Static Heuristics for Mapping a Class of Indepedent Tasks onto Heterogeneous Distributed Computing Systems*. Journal of Parallel and Distributed Computing Systems, Vol.61, No.6, pp 810-837, 2001.

[9] Louis C. Canon and E. Jeannot, *Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments*. IEEE Trans. on Parallel and Distributed Systems, Vol.21, No.4, pp 532-546, April 2010.

[10] M. Cosnard and M. Marrakchi and Y. Robert, *Parallel Gaussian Elimination on an MIMD Computer*. Parallel Computing, Vol.6, No.3, pp 275-296, 1988.

[11] Gilbert C. Sih and Edward A. Lee, *A Compile-Time Scheduling Heuristic for Interconnection-Constrainted Heterogeneous Machine Architectures*. IEEE Trans. on Parallel and Distributed Systems, Vol.4, No.2, pp 175-187, February 1993.

[12] L. Zeng and B. Veeravalli and X. Li, *ScaleStar:Budget Concious Scheduling Precedence-Constrained Many-task Workflow Applications in Cloud*. 2012 26th IEEE International Conference on Advanced Information Networking and Applications, pp 534-541, March 2012.

[13] H. Topcuoglu, S. Hariri and M. Wu, *Performance-Effective and Low-Complexity Task Schduling for Heterogeneous Computing*. IEEE Trans. on Parallel and Distributed Systems, Vol.13, No.3, pp 260-274, March 2002.

[14] Ruben V. den Bossche, K. Vanmechelen and J. Broeckhove, *Cost-Efficient Scheduling Heuristics for Deadline Constrained Workloads on Hybrid Clouds*. 2011 Third IEEE International Conference on Cloud Computing Technology and Science, pp 320-327, November 2011.

[15] J. Li, S. Su, X. Cheng, Q. Huang and Z. Zhang, *Cost-Conscious Scheduling for Large Graph Processing in the Cloud*. 2011 IEEE International Conference on High Performance Computing and Communications, pp 808-813, September 2011.