Towards task scheduling in a cloud-fog computing system

Xuan-Qui Pham
Department of Computer Engineering
Kyung Hee University
Suwon, South Korea
Email: pxuanqui@khu.ac.kr

Eui-Nam Huh
Department of Computer Engineering
Kyung Hee University
Suwon, South Korea
Email: johnhuh@khu.ac.kr

***Abstract*—In recent years, with the advent of the Internet of Things (IoT), fog computing is introduced as a powerful complement to the cloud to handle the IoT's data and communications needs. The interplay and cooperation between the edge (fog) and the core (cloud) has recently received considerable attention. In this paper, we consider task scheduling in a cloud-fog computing system, where a fog provider can exploit the collaboration between its own fog nodes and the rented cloud nodes for efficiently executing users' large-scale offloading applications. We first formulate the task scheduling problem in such cloud-fog environment and then propose a heuristic-based algorithm, whose major objective is achieving the balance between the makespan and the monetary cost of cloud resources. The numerical results show that our proposed algorithm achieves better tradeoff value than other existing algorithms.**

***Index Terms*—cloud computing, fog computing, task scheduling, Internet of Things.**

I. Introduction

Fog computing is a promising solution to deal with the demands of the ever-increasing number of Internet-connected devices. The idea of fog computing is to extend the cloud to be closer to the things that produce and act on IoT data. Instead of forcing all processing to back-end clouds, fog computing aims to process part of the services' workload locally on fog nodes, which are served as a near-end computing proxies between the front-end IoT devices and the back-end cloud servers. Putting resources at the edge of the network only one or two hops from the data sources allows fog nodes to perform low latency processing while latency-tolerant and large-scale tasks can still be efficiently processed by the cloud. In addition, the cost and scale benefits of the cloud can help the fog to serve peak demands of IoT devices if the resources of fog nodes are not sufficient. Also, many applications require the interplay and cooperation between the edge (fog) and the core (cloud), particularly for the IoT and big data analysis [1]. From this point of view, fog computing is not aimed to replace cloud computing, but to complement it in a new computing paradigm, cloud-fog computing, which is to satisfy the increasingly sophisticated applications demanded by users.

In this paper, we consider task scheduling in a cloud-fog computing system, where a fog provider can exploit the collaboration between its fog nodes and the rented cloud nodes for efficiently executing users' large-scale offloading applications. The fog nodes are local resources, which can be any devices with computing, storage, and network connectivity such as switches, routers, video surveillance cameras, etc. A simple scenario is that a shopping center can deploy many fog nodes in different floors to provide WiFi access and deliver some engaged services (i.e. indoor navigation, ads distribution, feedback collections) to its customers. However, in peak time, the capabilities of those fog nodes cannot efficiently serve the customers. Meanwhile, the fog provider, here is the shopping center, can extend its infrastructure by paying for the outsourced computation and storage resources of the cloud nodes, which can be virtual machine (VM) rented from cloud providers on a pay-per-use basis. All distributed processing nodes (cloud or fog) are managed by a resource broker, which is a resource management component and scheduler for the workflows submitted from users at the fog's side. In this case, a task schedule, which can minimize the completion time of the workflow, but corresponds to a large amount of monetary cost, is not an optimal solution for fog providers. Thus, in this paper, we propose a task scheduling algorithm that can achieve a good tradeoff between the workflow execution time and the cost for the use of cloud resources. The experimental results show the outstanding performance of our method compared with some other works.

The remainder of the paper is organized as follows. In section 2, we introduce some related works to the task scheduling problem in heterogeneous environments. The architecture of the cloud-fog computing system is described in section 3. In section 4, we formulate the task scheduling problem and present our proposed method. Then we describe some experimental results in section 5, followed by our conclusions and suggestions for future work in section 6.

II. Related work

In heterogeneous environments, despite numerous efforts, task scheduling still remains a big challenge. As usual presentation, each application is comprised of multiple interdependent tasks and each of which is specified by an amount of processing works. It can be modeled as a Directed Acyclic Graph (DAG), in which vertices represent application tasks and edges represent intertask data dependencies. The primary goal of task scheduling is to schedule tasks on processors and

minimize the makespan of the schedule, which has been shown to be NP-complete problem. The most common task scheduling algorithms are list-scheduling heuristics. For example, the Earliest Time First (ETF) algorithm [2] computes, at each step, the earliest start times of each tasks on all processors and then selects the one with the smallest start time. The Dynamic Level Scheduling (DLS) algorithm [3] selects the task-processor pair that maximizes the value of the dynamic level (DL), which is the different between the static level of a task and its earliest start time on a processor. Meanwhile, the heterogeneous earliest-finish-time (HEFT) algorithm [4] selects the tasks with the highest upward rank and then assigns it to the processor that minimizes its earliest finish time. However, how to achieve good tradeoff value between the makespan and the monetary cost is not considered in these algorithms.

For a large scale environment, e.g. cloud computing system, there had been also numerous scheduling approaches proposed with the goal achieving both the better application execution and cost saving for cloud resources. Bossche at al. [5] introduce a cost-oriented scheduling algorithm to select the most proper system (private or public cloud) for executing the incoming workflows based on the ability of meeting the deadline of each workflow and cost savings. The budget constraints for using the cloud resource are considered in ScaleStar [6], whose task assignment is based on a novel objective function Comparative Advantage (CA). This algorithm achieves good balance between cost savings and schedule length, however, the high complexity of CA hinder the algorithm to be applied to the large-scale workflows.

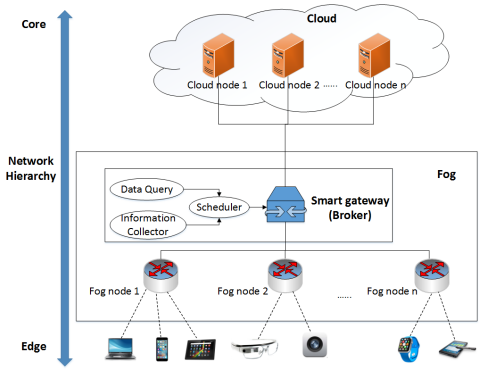

Fig. 1. System architecture

III. System Model

Our cloud-fog computing system has three layers in a hierarchy network, as represented in Figure 1. The front-end layer consists of IoT devices, which serve as user interfaces that send requests from users. The fog layer, which is formed by a set of near-end fog nodes, receives and processes part of a workload of users' requests. The cloud layer, which hosts a number of computing machines or cloud nodes, provides outsourced resources to execute the workload dispatched from the fog layer. Because the computing resources of our system are dispersed into cloud nodes and fog nodes, there is a smart gateway or broker, which is a centralized management component and task scheduler. The broker (1) receives all requests of users; (2) manages available resources on cloud and fog nodes (e.g. processing capacity, network bandwidth) as well as processing and communication costs together with results of data query returned from nodes; and (3) creates the most appropriate schedule for an input workflow.

IV. Task Scheduling in Cloud-Fog Computing System

A. Task graph

A task graph is represented by a Directed Acyclic Graph (DAG), $G = (V, E)$, where the set of vertices $V = \{v_1, v_2, ..., v_n\}$ denotes the set of parallel subtasks and each edge $e_{ij} \in E$ represents the precedence constraint such that task v_i should complete its execution before task v_j starts.

Each task $v_i \in V$ has positive workload w_i representing the amount of computing works (e.g. the number of instructions), which have to be processed at the computing resources. And each edge $e_{ij} \in E$ has nonnegative weight c_{ij} representing the amount of communication data transfered from task v_i and used as input data for task v_j. We assume that the sufficient input data of each task is gathered not only from the preceding tasks but also from other data sources (i.e. data storages) on both cloud and fog infrastructure. A task cannot begin execution until all its inputs have arrived.

The set of all direct predecessors and successors of v_i is denoted as $pred(v_i)$ and $succ(v_i)$ respectively. We assume that G has an entry task, v_{entry}, without any predecessors and an exit task, v_{exit}, without any successors.

B. Processor graph

A processor graph $PG = (N, D)$ denotes the topology of a cloud-fog network, where the set of vertices $N = \{P1, P2, ..., P_n\}$ denotes the set of processors, each of which is cloud or fog node and an edge $d_{ij} \in D$ denotes a link between processor P_i and P_j. Let N_{cloud} and N_{fog} denotes the set of cloud nodes and the set of fog nodes respectively. Hence, $N = N_{cloud} \cup N_{fog}$. Each processor P_i has processing rate p_i and the link d_{ij} between processor P_i and P_j has bandwidth bw_{ij}.

C. Proposed method

Given a task graph $V = \{v_1, v_2, ..., v_n\}$ and a processor graph $P = (N, D)$, we consider to choose the most appropriate schedule to execute the tasks. Our method has two phases:

1) Determining the task priority: In this phase, tasks are ordered by their scheduling priorities that are based on upward ranking. Basically, the upward rank of a task v_i is the length of the critical path from v_i to the exit task, including the computation time of task v_i. Let $pri(v_i)$ be the priority value of task v_i and be recursively defined by:

$$pri(v_i) = \begin{cases} \overline{w(v_i)} + \max\limits_{v_j \in succ(v_i)} \left[\overline{c(e_{ij})} + pri(v_j)\right] & \text{if } v_i \neq v_{exit} \\ \overline{w(v_i)} & \text{if } v_i \equiv v_{exit} \end{cases} \quad (1)$$

where $\overline{w(v_i)}$ is the average execution time of task v_i and $\overline{c(e_{ij})}$ is the average data transfer time between two tasks v_i and v_j. They are computed by:

$$\overline{w(v_i)} = \frac{w_i}{\overline{W}}, \tag{2}$$

$$\overline{c(e_{ij})} = \frac{c_{ij}}{\overline{BW}} \tag{3}$$

with $\overline{W}$ is the average processing rate of all processors and $\overline{BW}$ is the average transfer rate or bandwidth among all processors.

2) Selecting the most appropriate node to execute each task: In this phase, the two parameters Earliest Start Time (EST) and Earliest Finish Time (EFT) need to be defined. A task v_i cannot begin its execution until all its inputs have been available. Let $t_{dr}(v_i)$ be the time when all input data of v_i is ready to be transfered to the selected node for executing the task v_i. It is also the time when the last preceding task of v_i is finished. Thus $t_{dr}(v_i)$ is defined by:

$$t_{dr}(v_i) = \max_{v_j \in pred(v_i), P_m \in N} [t_f(v_j, P_m)] \tag{4}$$

where $t_f(v_j, P_m)$ is the finish time of task v_j on node P_m. For the entry task, $t_{dr}(v_{entry}) = 0$.

Suppose task v_i is assigned to node P_n. Let $c(e_i^{mn})$ be the data transfer time from node P_m to node P_n to execute task v_i, then $c(e_i^{mn})$ is defined as follows [7]:

$$c(e_i^{mn}) = \begin{cases} \left(d_i^m + \sum\limits_{v_j \in exec(P_m)}^{v_j \in pred(v_i)} c_{ji} \right) * \frac{1}{bw_{mn}} & \text{if } m \neq n \\ 0 & \text{if } m = n \end{cases} \tag{5}$$

where d_i^m is the amount of data already stored at processor P_m for executing task v_i and $exec(P_m)$ is the set of tasks executed at node P_m.

When all necessary input data stored from all data storages on either cloud nodes or fog nodes arrive at the target processing node, task execution will begin. Therefore, the values of $EST(v_i, P_n)$ and $EFT(v_i, P_n)$ are computed as follows:

$$EST(v_i, P_n) = \max \left\{ avail(P_n), t_{dr}(v_i) + \max_{P_m \in N} (c(e_i^{mn})) \right\} \tag{6}$$

$$EFT(v_i, P_n) = w(v_i, P_n) + EST(i, j) \tag{7}$$

where $avail(P_n)$ is the earliest time that node P_n completes the last assigned task and be ready to execute another task; $w(v_i, P_n)$ is the execution time of task v_i on node P_n. They are computed as follows:

$$avail(P_n) = \max_{v_j \in exec(P_n)} [t_f(v_j, P_n)], \tag{8}$$

$$w(v_i, P_n) = \frac{w_i}{p_n} \tag{9}$$

Besides, the algorithm also considers the monetary cost that fog provider is charged for the use of cloud resources. The fog provider rents both virtual hosts representing for the computing resources and network bandwidth from cloud providers in order to extend the capabilities of their own fog nodes. Thus, if P_n is a cloud node, the monetary cost $cost(v_i, P_n)$ for executing task v_i on P_n includes two parts: the processing cost $c_{proc}^{(v_i,P_n)}$ of v_i on P_n and the communication cost $c_{comm}^{(v_i,P_m)}$ for the amount of outgoing data from a cloud node $P_m \in N_{cloud}$ to the target node P_n to process task v_i. In contrast, if P_n is a fog node, the fog provider only needs to pay for transferring the outgoing data from cloud nodes to the target fog node in the local system. Therefore, the total cost for executing task v_i on a specific node P_n is defined by:

$$cost(v_i, P_n) = \begin{cases} c_{proc}^{(v_i,P_n)} + \sum\limits_{P_m \in N_{cloud}} c_{comm}^{(v_i,P_m)} & \text{if } P_n \in N_{cloud} \\ \sum\limits_{P_m \in N_{cloud}} c_{comm}^{(v_i,P_m)} & \text{if } P_n \in N_{fog} \end{cases} \tag{10}$$

In (10), the processing cost $c_{proc}^{(v_i,P_n)}$ is calculated as follows:

$$c_{proc}^{(v_i,P_n)} = c_1 * w(v_i, P_n) \tag{11}$$

where c_1 is the processing cost per time unit of workflow execution on cloud node P_n. Let c_2 be the the amount of money per time unit for transferring outgoing data from cloud node P_m, then the communication cost $c_{comm}^{(v_i,P_m)}$ is calculated as follows:

$$c_{comm}^{(v_i,P_m)} = c_2 * \left(d_i^m + \sum_{v_j \in exec(P_m)}^{v_j \in pred(v_i)} c_{ji} \right) \tag{12}$$

From this cost, we can define an utility function which computes the tradeoff between the cost and EFT as follows:

$$U(v_i, P_n) = \frac{\min\limits_{P_k \in N} [cost(v_i, P_k)]}{cost(v_i, P_n)} * \frac{\min\limits_{P_k \in N} [EFT(v_i, P_k)]}{EFT(v_i, P_n)} \tag{13}$$

Then, the task v_i is assigned to the node P_n, which provides the maximal value of the tradeoff $U(v_i, P_n)$. Our method is presented in Algorithm 1.

V. Experimental Results

In this experiment, we present the results to show that our proposed algorithm can provide a good tradeoff between the makespan and the cost of task execution. We compare our algorithm with three others: Greedy for Cost, where each task is assigned to the the most cost-saving processing node and the classical HEFT [4] and DLS [3] algorithms mentioned in section 2. We use Cloudsim for modeling and simulation of the cloud-fog computing infrastructure. All the parameters are presented in Table I. The task matrix size is raised from 20 to 100 with the increasing steps of 20.

In order to prove that our algorithm can achieve better tradeoff value between the makespan and the cost of task execution than other methods, we define a comparision criteria called *Cost Makespan Tradeoff (CMT)* as follows:

$$CMT(a_i) = \frac{\min\limits_{a_k \in SAL} [cost(a_k)]}{cost(a_i)} * \frac{\min\limits_{a_l \in SAL} [makespan(a_l)]}{makespan(a_i)} \tag{14}$$

Algorithm 1: Proposed method

Input: Task graph $G(V, E)$ and processor graph $PG = (N, D)$ $(N = N_{cloud} \cup N_{fog})$
Output: A task schedule
1. Compute the priority level $pri(v_i)$ of each task $v_i \in V$ by traversing graph upward, starting from v_{exit}.
2. Sort all tasks of V into list L by nonincreasing order of priority levels.
3. **For all** $v_i \in L$ **do**
  3.1. Compute $t_{dr}(v_i)$
  3.2. **For all** $P_n \in N$ **do**
    3.2.1. Compute $EST(v_i, P_n)$, $EFT(v_i, P_n)$ and $cost(v_i, P_n)$
    3.2.2. Compute the utility function $U(v_i, P_n)$
  3.3. **end for**
  3.4. Assign task v_i to the processor P_n that maximizes $U(v_i, P_n)$ of task v_i
4. **end for**

TABLE I
SYSTEM CONFIGURATION

Parameter	Value
Topology	LAN, fully connected
Number of tasks	[20 100]
Number of cloud nodes	42
Number of fog nodes	22
Processing rate	[50,500] MIPS
Bandwidth	10, 100, 512, 1024 Mbps
Processing cost per time unit	[0.1, 0.5]
Communication cost per time unit	[0.3, 0.7]

where $SAL = \{a_1, a_2, ..., a_n\}$ is the list of all scheduling algorithms, which we compute the CMT value of each algorithm $a_i \in SAL$. The higher of CMT value, the better tradeoff level on monetary cost and schedule length that an algorithm can provide. The maximum value of this metric is 1, which is reachable if both cost and schedule length of an algorithm are the best compared with the others'.

Figures 2 shows the comparision between our algorithm and the above-mentioned algorithms on the CMT metric. We can see that our algorithm is stable and achieve the highest CMT value compared with the others in most cases. Compared with Greedy for Cost algorithm, which achieves the minimum monetary cost but long schedule length, our algorithm has better CMT value in all cases. The HEFT algorithm achieves the minimum schedule length but it goes with the significant increase of cost. The DLS algorithm also achieves small schedule length, but it requires much more cost for cloud resources and thus gets the worst CMT value, which is about from 15% to 25% lower than our proposed algorithm.

VI. CONCLUSION

In this paper, we introduce a cloud-fog computing system which is the combination of fog nodes owned by a fog provider and cloud nodes rented from cloud providers. For the sake of reaping the most benefit from cloud-fog computing

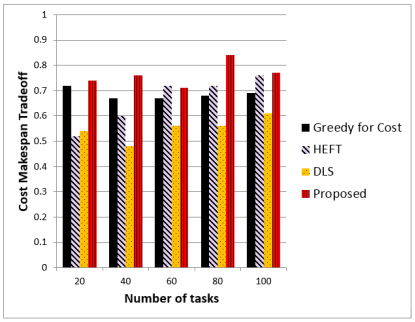

Fig. 2. Comparision on the CMT metric between different scheduling algorithms

system, one must allocate computing tasks strategically at each processing nodes of each layer. We propose a scheduling algorithm which not only guarantees the performance of application execution, but also reduces the mandatory cost for the use of cloud resources. In future, the scheduling algorithm should be made more robust by considering additional constraints, such as fog provider's budget and deadline constraint of a workflow execution.

ACKNOWLEDGMENT

This work was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2016-(H8501-16-1015) supervised by the IITP(Institute for Information & communications Technology Promotion)), and Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R7117-16-0202, The Development of Cloud Edge Computing Technology for Real-time IoT/CPS). Professor Eui-Nam Huh is the corresponding author.

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *Proceedings of ACM SIGOPS*, pp. 8792, 2002.
[2] J.J. Hwang, Y.C. Chow, F.D. Anger, and C.Y. Lee, "Scheduling Precedence Graphs in Systems with Interprocessor Communication Times," in *SIAM Journal on Computing*, vol.18, no. 2, pp. 244-257, Apr. 1989.
[3] G.C. Sih and E.A. Lee, "A compile-time scheduling heuristic for interconnection-constrained heterogeneous machine architectures," in *IEEE Trans. Parallel Distrib. Systems*, vol.4, no.2, pp. 175186, Feb. 1993.
[4] H. Topcuoglu, S. Hariri and Min-You Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260-274, Mar 2002.
[5] R. Van den Bossche, K. Vanmechelen and J. Broeckhove, "Cost-Efficient Scheduling Heuristics for Deadline Constrained Workloads on Hybrid Clouds," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 320-327, 2011.
[6] L. Zeng, B. Veeravalli and X. Li, "ScaleStar: Budget Conscious Scheduling Precedence-Constrained Many-task Workflow Applications in Cloud," in *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, pp. 534-541, 2012.
[7] Nguyen Doan Man and Eui-Nam Huh, "Cost and Efficiency-based Scheduling on a General Framework Combining between Cloud Computing and Local Thick Clients," in *2013 International Conference on Computing, Management and Telecommunications (ComManTel)*, pp. 258-263, 2013.