Task Classifying Model based on Data Traits for High Efficiency in Cloud Infrastructure Modeling and Simulation Environment

Sunghwan Moon, Jaekwon Kim, Taeyoung Kim, Jeongseok Choi and Jongsik Lee
Department of Computer and Information Engineering
Inha University
Incheon, South Korea
email: shmoon@inhaian.net, jaekwonkorea@naver.com, silverwild@gmail.com,
jeongseokchoi.korea@gmail.com and jslee@inha.ac.kr

***Abstract*—We proposed task Classifying Model based on Data Traits (CMDT) and conducted experiments using this model. CMDT classifies tasks from user taking account of its own data traits. The classified tasks are allocated to each of the nodes which can process them as fast as possible. In conclusion, CMDT improves a service throughput which is the index of efficiency on cloud.**

Keywords-data traits; task classifying model; CMDT.

I. INTRODUCTION

Raising user-level has led to increase the demand for processing highly complex tasks. Service providers meet their demand using high performance computing which is composed of diverse computing resources on cloud service[1]. Clients and providers contract a Service Level Agreement (SLA) for high performance computing service. According to a SLA, a client pays a specific fee and a provider ensures parameters matching agreement[2]. A low expense for task processing is economic to the client. On the other hand, a high performance for service is profitable to the provider. In cloud Infrastructure as a Service (IaaS)[3], a SLA should be guaranteed by allocating physical computing resources efficiently.

Users request processing tasks which include the data. The data comes in a lot of types such as videos, images, audios, texts, logs, etc. The task including data has a dependency on nodes. The nodes are physical computing resources for processing tasks; their performances are closely related with the efficiency of whole system. If the system classifies the tasks regardless of its own data traits, most of tasks may be processed slowly in a long time[4]. This situation results in breach of a SLA.

In this paper, we propose a task Classifying Model based on Data Traits (CMDT) to increase a resource efficiency on cloud environment. CMDT classifies the requested task with its own data traits. The classified task is allocated to the node which has a dependency on the data. CMDT can increase the efficiency by reducing turnaround time at each node and also ensure a SLA degree for stakeholders.

The rest of this paper is structured as follows: In Section 2, we describe our key idea for task classifying in cloud environment. Section 3 explains the experiment settings and results. Finally, we conclude the paper in Section 4.

II. TASK CLASSIFYING METHOD BASED ON DATA TRAITS

We introduce CMDT in this section. CMDT classifies a task according to its own data traits and allocates the task to highly relevant physical resource. Figure 1 shows a designed architecture of the proposed CMDT.

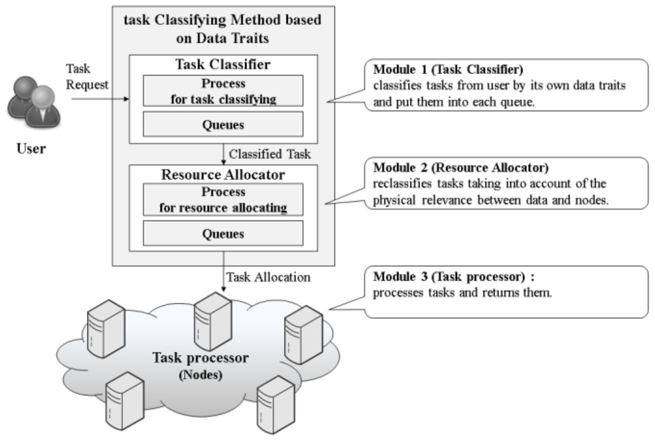
task Classifying Method based on Data Traits
Task Request
User
Task Classifier
Process for task classifying
Queues
Classified Task
Resource Allocator
Process for resource allocating
Queues
Task Allocation
Task processor (Nodes)
Module 1 (Task Classifier) classifies tasks from user by its own data traits and put them into each queue.
Module 2 (Resource Allocator) reclassifies tasks taking into account of the physical relevance between data and nodes.
Module 3 (Task processor) : processes tasks and returns them.

Figure 1. CMDT Architecture

CMDT consists of three modules for resource allocation. These modules take roles as follows. First, Task Classifier stores a requested task from user to each queue according to its own data traits. The data traits depend on the metadata of tasks. Second, Resource Allocator distributes tasks from Task classifiers to each queue according to their physical properties to process efficiently. Third, Task Processor receives tasks from Resource allocators and processes them.

These phases perform on each of the modules as follows:

A. *Task Classifier*

There are various requests in cloud services. Some tasks include complex applications which need high powered computing. Others are just based on web services. Task classifier stores every task with many purposes referring to its own data traits, three roles of which are as follows.

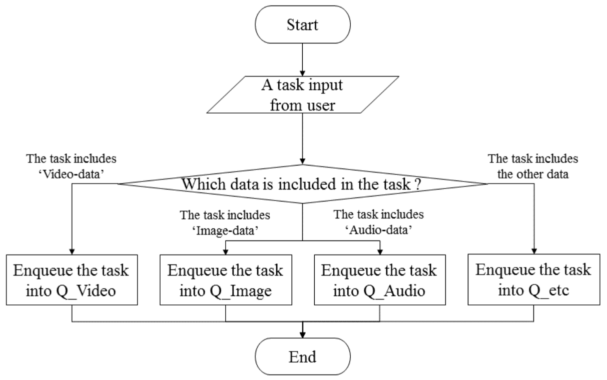
Start
A task input from user
Which data is included in the task ?
The task includes 'Video-data'
The task includes 'Image-data'
The task includes 'Audio-data'
The task includes the other data
Enqueue the task into Q_Video
Enqueue the task into Q_Image
Enqueue the task into Q_Audio
Enqueue the task into Q_etc
End

Figure 2. Process for Task Classification

1) As shown in Figure 2, Task classifier stores every task from users in each queue. The tasks is classified by its own data traits. Classified tasks are stored to pre-deployed queues. We use four queues for each trait. Table 1 shows detailed criteria for classification.

TABLE I. QUEUE IN TASK CLASSIFIER MODULE

Queue	Description	Data
Q_Video	Enqueue the User-Request Job including Video-Data	.avi, .mkv, .mp4, .wmv, etc.
Q_Image	Enqueue the User-Request Job including Image-Data	.jpg, .png, .gif, .tif , etc.
Q_Audio	Enqueue the User-Request Job including Audio-Data	.wav, .ogg, .mp3, .wma, etc.
Q_etc	Enqueue the User-Request Job including Other data	.txt, .log, etc.

2) Task classifier sends stored tasks when Resource allocator requests new task as shown in Figure 3. The request occurs when its queue size downs to less than a certain amount. This amount can be adjusted as high or low depending on the maximum length of the queue.

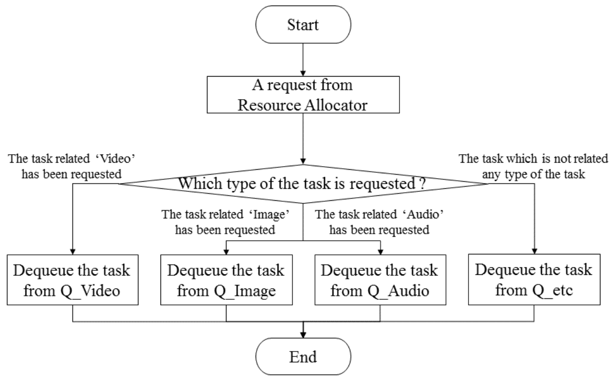
Start
A request from Resource Allocator
Which type of the task is requested ?
The task related 'Video' has been requested
The task related 'Image' has been requested
The task related 'Audio' has been requested
The task which is not related any type of the task
Dequeue the task from Q_Video
Dequeue the task from Q_Image
Dequeue the task from Q_Audio
Dequeue the task from Q_etc
End

Figure 3. Process for Task Selection

3) Task classifier receives finished tasks from Task processor and returns them to the users. Users can request their tasks if uncompleted.

B. Resource Allocator

Resource allocator manages new tasks taking account of acceptable workloads and throughputs of the computing resource on cloud. Two roles of this module are as follows.

1) Resource allocator receives the task which was classified by its own data traits. These tasks are reclassified in view of the performance ratio of Task processor. A classification criteria is described in the following reasons.

- A task including videos and dynamic images:
  Most of tasks need real-time encoding, decoding and storing for large-scale data. Because of this, CPU utilization is extremely high for these kinds of tasks[5].
- A task including graphics and static images:
  Most of tasks need preview and storing images. RAM utilization is high for these kinds of tasks[6].
- A task including audios and voice speech:
  Most of tasks are streaming service and real-time transmission. These kinds of tasks need minimizing of the network delay[7].

Equation (1) presents a priority rule for allocation of each task to physical computing resource because of the reasons mentioned above.

$$\begin{aligned} &\textit{Video-Data} : Q\_CPU > Q\_NetResp. > Q\_RAM > Q\_All \\ &\textit{Image-Data} : Q\_RAM > Q\_CPU > Q\_NetResp. > Q\_All \\ &\textit{Audio-Data} : Q\_NetResp. > Q\_RAM \geq Q\_CPU > Q\_All \end{aligned} \tag{1}$$

The tasks are allocated into each queue in this module. Resource allocator has four specified queues as described in Table 2.

TABLE II. QUEUE IN RESOURCE ALLOCATOR MODULE

Queue	Description	Property
Q_CPU	Enqueue the Job to be assigned Node which has High-Level CPU	Job including Video-Data
Q_RAM	Enqueue the Job to be assigned Node which has High-Level RAM	Job including Image-Data
Q_NetResp.	Enqueue the Job to be assigned Node which has High-NetResponse	Job including Audio-Data
Q_All	Enqueue the Job to be assigned Node on Low-Load	Job including Other data

2) Resource allocator sends a task according to requests from Task processor. This module estimates how much time Task processors would finish the tasks because the processors have different performances. Resource Allocator operates for allocating the tasks as shown in Figure 4.

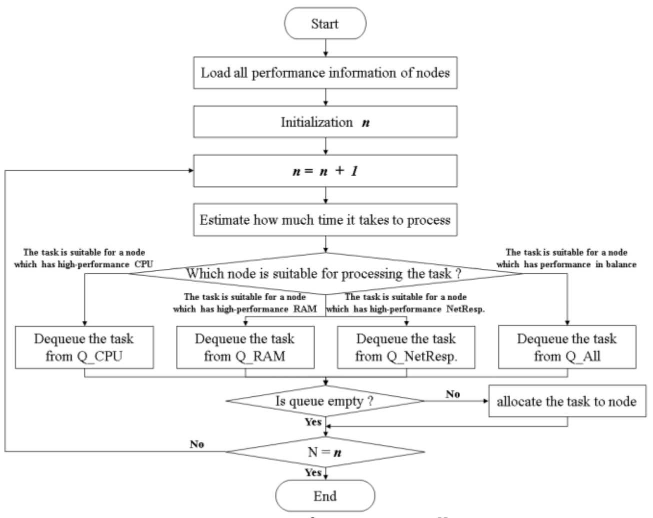
Start
Load all performance information of nodes
Initialization n
n = n + 1
Estimate how much time it takes to process
The task is suitable for a node which has high-performance CPU
The task is suitable for a node which has performance in balance
Which node is suitable for processing the task ?
The task is suitable for a node which has high-performance RAM
The task is suitable for a node which has high-performance NetResp.
Dequeue the task from Q_CPU
Dequeue the task from Q_RAM
Dequeue the task from Q_NetResp.
Dequeue the task from Q_All
Is queue empty ?
No
allocate the task to node
Yes
No
N = n
Yes
End

Figure 4. Process for Resource Allocation

C. Task Processor

Task processor, called 'node' on cloud, is a physical computing resource. This module processes the allocated tasks and sends a finished task to Task classifier. Task processor operates for processing the tasks as shown in Figure 5.

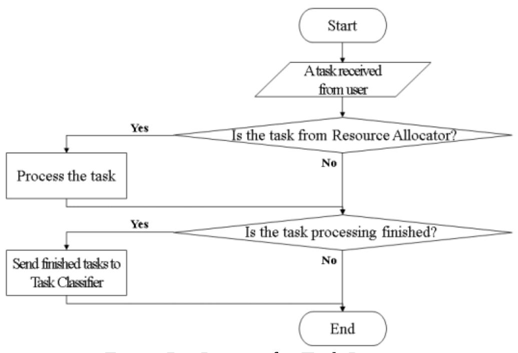
Start
A task received from user
Yes
Is the task from Resource Allocator?
No
Process the task
Yes
Is the task processing finished?
No
Send finished tasks to Task Classifier
End

Figure 5. Process for Task Process

III. EXPERIMENT DESIGN AND RESULT

We designed the cloud environments in order to verify a performance of CMDT. This is a virtual distributed environment based on Discrete Event System Specification (DEVS) formalism[8]. We experiment and measure a throughput as a performance index.

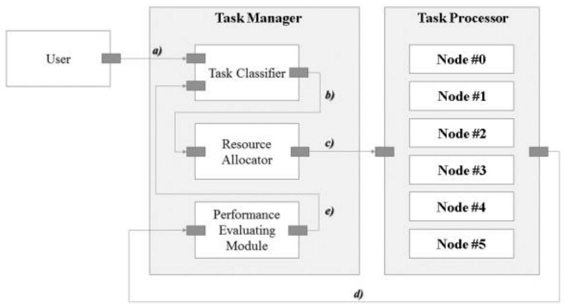

Figure 6. Virtual Environment based on DEVS Formalism

A. Experiment Scenario

In this paper, we build a virtual distributed environment for CMDT using DEVS formalism. This experiment is built to verify the performance of CMDT. Figure 6 shows a test bed and the description is as follows.

1) *User generates and requests a task. It also receives a finished task. This module is a generator model.*

2) *Task Classifier has queues to classify and store a generated task according to its own data traits. This module is a queue model.*

3) *Resource Allocator reclassifies and allocates each task depending on the computing resource. This module is a queue-processor model.*

4) *Node processes a task and sends it to Task classifier. This module is a processor model. It is also called a node.*

5) *Performance Evaluator evaluates a throughput of the task processing. This module is a transducer model.*

We define the performance of nodes for our experiment as shown in Table 3. The higher value it is, the better performance it has.

TABLE III. NODE PERFORMANCE

Node	CPU	RAM	NetResp.
Node #0	9	8	8
Node #1	8	9	8
Node #2	8	8	9
Node #3	9	8	6
Node #4	8	9	7
Node #5	7	8	9

In our experiment, we measure a service throughput with increasing 300 to 3000 for finished time. A service throughput is a performance index which is a total amount of services of each model during designated experiment time. This index is calculated by dividing the number of service response to a finished time as given by (2).

$$\text{Throughput} = \text{The Number of Service Response} / \text{Finished Time} \quad (2)$$

We select two algorithms for applying CMDT because CMDT is an adjunctive method which can be applied to all the task scheduling algorithms. First model is a round robin scheduling algorithm (RR)[9]. RR sequentially allocates tasks to all nodes. In other words, the task is allocated in the order of nodes. Finished tasks are also returned in the order. Second model is a minimum load first scheduling algorithm (MLFS)[10]. MLFS allocated tasks to the node which has the minimum number of task among all nodes on cloud. This model has the merit of load balancing. We applied CMDT to those algorithms.

We finally conduct two comparative experiments. One experiment is comparing RR with RR-CMDT. RR means an original round robin scheduling algorithm. RR-CMDT means an improved round robin scheduling algorithm which

CMDT has been applied to. The other experiment is comparing MLFS with MLFS-CMDT. MLFS means an original minimum load first scheduling algorithm. MLFC-CMDT means an improved minimum load first scheduling algorithm which CMDT has been applied to.

B. Experiment Results

We measure the service throughput in order to compare performance between four scheduling algorithms. They are RR, RR-CMDT, MLFS and MLFS-CMDT. In our experiments, the user requests the tasks which have random sizes. The sizes are based on the Amazon Access Samples Data Set [11], which is opened through UCI Machine Learning Repository. The purpose of these experiments are to verify that CMDT ensures a SLA by increasing the service throughput.

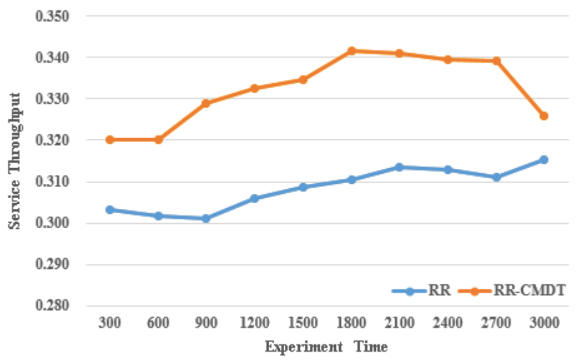

Figure 7. Service Throughput of RR and RR-CMDT

As shown in Figure 7, RR records 0.308 and RR-CMDT records 0.332. This resulting value is an average of the service throughput. It is seen that when CMDT has been applied, a service throughput increased.

RR allocates the requested tasks in order. This method not only classifies the tasks regardless of its own data traits, but it also does not consider the state of nodes. These cause an overload problem at each node. On the contrary, RR-CMDT classifies the tasks taking account of the physical relevance between data and nodes. This method enables the system to process more tasks using limited resources by reducing turnaround time at each node. Service providers can ensure a SLA more easily when the service throughput increases.

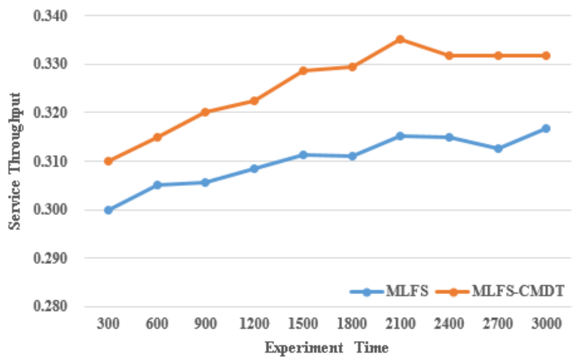

Figure 8. Service Throughput of MLFS and MLFS-CMDT

As shown in Figure 8, MLFS records 0.310 and MLFS-CMDT records 0.326. This resulting value is an average of the service throughput. We see that when CMDT has been applied, a service throughput increased.

MLFS allocates the requested task to the node which has the least number of tasks in its queue. This method balances the load of whole system, but it does not consider the physical relevance between data and nodes like RR. Meanwhile, MLFS-CMDT classifies the tasks taking account of the physical relevance between data and nodes like RR-CMDT. Finally, RR-CMDT and MLFS-CMDT improves the efficiency and ensure a SLA by managing the tasks taking account of the physical relevance between data and nodes.

IV. CONCLUSION

Cloud services provide a high performance computing which can process a large-scale data and complex tasks. There is an outstanding issue ensuring a SLA with the limited resources available on cloud.

We propose a task Classifying Model based on Data Traits (CMDT). This method increases the efficiency of computing resources by applying its own process to the usual scheduling algorithms. CMDT classifies tasks according to its own data traits and allocates tasks depending on relevance between data and physical properties of nodes. It ensures a SLA through improving the service throughput.

Future work will concentrate on applying CMDT to the other task scheduling algorithms. We think CMDT can be applied to more diverse algorithms.

ACKNOWLEDGMENT

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract UD140022PD, Korea.

REFERENCES

[1] G. Kim, W. Lee, and C. Jeon, “Virtualization Technology for Cloud Computing”, Journal of the Korea Society of Computer and Information, Vol. 18, No. 1, 2010, pp. 25-33.

[2] H. Kang, J. Koh, and Y. Kim, “A SLA-based VM Auto-Scaling Method in Hybrid Cloud Computing for Scientific Computational Applications”, Journal of KIISE, System and Theory, Vol. 40, No. 6, 2013, pp. 266-273.

[3] J. K. Kim and J. S. Lee, “Fuzzy Logic-driven Virtual Machine Resource Evaluation Method for Cloud Provisioning Service”, Journal of the Korea Society for Simulation, Vol. 22, No. 1, 2013, pp. 77-86.

[4] B. S. Kim, S. D. Lee, T. G. Kwon, and S. H. Lee, “Design and Implementation of the Unformatted Data Manager for Multimedia Storage System”, Journal of KIISE, Vol. 20, No. 2, 1993, pp. 191-194.

[5] S. J. Lee, E. J. Lee, S. W. Hong, H. N. Choi, and Y. W. Chung, “Secure and Energy-Efficient MPEG Encoding using Multicore Platforms”, Journal of the Korea Institute of Information Security and Cryptology, Vol. 20, No. 3, 2010, pp. 113-120.

[6] H. S. Oh, “Tiled Image Compression Method to Reduce the Amount of Memory Needed for Image Processing in Mobile Devices”, Journal of Korea Game Society, Vol. 13, No. 6, 2013, pp. 35-42.

[7] B. J. Kim, “Service Quality Criteria for Voice Services over a WiBro Network”, The Journal of the Korea Institute of

Electronic Communication Sciences, Vol. 6, No. 6, 2011, pp. 823-829.

[8] Bernard P. Zeigler, H. Praehofer, and T. G. Kim (2000), “Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems”, Academic Press, 2000, pp. 76-96.

[9] S. Pooja and P. Mishra, “Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing”, International Journal of Computer Science and Information Technologies, Vol. 4, No. 3, 2013, pp. 416-419.

[10] T. Janaszka, D. Bursztynowski, and M. Dzida, “On popularity-based load balancing in content networks”, Teletraffic Congress (ITC 24), 24th International. IEEE, 2012, pp. 1-8.

[11] Amazon Access Samples Data Set. [Online]. Available from: http://archive.ics.uci.edu/ml/datasets/Amazon+Access+Sampl es
2015.12.17