Macrothink Institute™

Network Protocols and Algorithms
ISSN 1943-3581
2017, Vol.9, No.1-2

Resource Allocation for Heterogeneous Cloud Computing

Ha Huy Cuong Nguyen

Quang Nam University, Quang Nam, Viet Nam

Tel: 084-0935-019-929 E-mail: nguyenhahuycuong@gmail.com

Vijender Kumar Solanki

Department of Computer Science and Engineering, CMR Institute of Technology, Hyderabad India

Tel: +91-9911513926 E-mail: spesinfo@yahoo.com

Doan Van Thang

Industrial University of Ho Chi Minh, Ho Chi Minh, Viet Nam

Tel: +84-0935-011-076 E-mail: vanthangdn@gmail.com

Thanh Thuy Nguyen

VNU University of Engineering and Technology, VietNam

Tel: 084-0913-531-184 E-mail: nguyenthanh.nt@gmail.com

Received: May 8, 2017 Accepted: June 8, 2017 Published: June 30, 2017

DOI:10.5296/npa.v9i1-2.11076 URL: https://doi.org/10.5296/npa.v9i1-2.11076

Abstract

Cloud Computing and the Internet of Things (IoT) have been converging, creation of the distributed computing system large scale, called IoT Cloud Systems. One of the main advantages of Cloud Computing is reflected in its support for self-service, on-demand resource consumption, where users can dynamically allocate appropriate amount of infrastructure resources (e.g., computing or storage) required by an application. In this paper, we develop a method to predict the lease completion time distribution that is applicable to making a sophisticated trade off decisions in resource allocation and rescheduling. Research methods have improved the

efficiency and effectiveness of heterogeneous cloud computing resources.

Keywords: IoT Cloud System; Deadlock Prevention; Resource Allocation; Distributed Computing; Virtualization.

1 Introduction

Recently, there has been a dramatic increase in the popularity of cloud computing systems that rent computing resources on-demand, bill on a pay-as-you-go basis, and multiplex many users on the same physical machine. These cloud computing environments provide an illusion of infinite computing resources to cloud users that they can increase or decrease their resource [1] [2]. Cloud Computing and the Internet of Things (IoT) have been converging, creation of the distributed computing system large scale, called IoT Cloud Systems [3] [4]. One of the main advantages of Cloud Computing is reflected in its support for self-service, on-demand resource consumption, where users can dynamically allocate appropriate amount of infrastructure resources (e.g., computing or storage) required by an application [5] [6]. As more and more people are using virtual machine technology in the data center, especially as many IoT Cloud Systems are deployed in the layer infrastructure as a services. Because, we have witnessed a lot of benefits of this utility-based supply model in flexible and cheaper IT operations [7] [8]. Current approaches dealing with IoT Cloud provisioning mostly focus on providing virtualization solutions for the Edge devices, such as IoT gateways [7] [8]. In this paper, we develop a method to predict the lease completion time distribution that is applicable to making a sophisticated trade off decisions in resource allocation and rescheduling. Research methods have improved the efficiency and effectiveness of heterogeneous cloud computing resources. It can be seen that previous studies have suggested that device virtualization is one of the prerequisites for providing a utility-based approach, which is often used to support a task. Specifically, such as data integration or data linking and largely rely on rigid supply models. With our automated resource provisioning model, our proposed alternative has effectively responded to the growing resource use needs of IoT Cloud Systems. The work is organized in the following way: in section 2, we introduce the related works; in section 3, we introduce existing models V VM-out-of-N PM, this model we build a resource optimization objective function; in section 4, we present deadlock avoidance algorithm; in section 5, we present our conclusions and suggestions for future work.

2 Related Work

Resource allocation in cloud computing has attracted the attention of the research community in the last few years. Cloud computing presents a different resource allocation paradigm than either grids or batch schedulers [9]. In particular, Amazon C2 [10], is equipped to, handle may smaller computer resource allocations, rather than a few, large request as is normally the case with grid computing. The introduction of heterogeneity allows clouds to be competitive with traditional distributed computing systems, which often consist of various types of architecture as well. Like traditional distributed system before we can see a heterogeneous distributed system consists of a set of processes that are connected by a communication network. The communication delay is finite but unpredictable. Eugen Feller et al. [11] showed a resource allocation based on a homogeneous platform model to provide the number of physical machines to virtual machine and proved that the energy consumption of the system is reduced if the number of used physical machines is diminished. We will explore the heterogeneous platform such as resources of the physical machines are not the same. According to some reviews pointed out

some practices of systems resources allocation with the minimal energy consumption and just focused on power utilization on the physical machines CPU. We believed that this burning is not only upon CPU but also over other appliances such as hard disk, bandwidth, ect. In [12] [13] [14] studied a lot of the problem of a request and grant computing resources rescheduling for multi-tiered web applications in heterogeneous distributed systems in order to minimize energy consumption while meeting performance requirements. They proposed an algorithms heuristic for a multidimensional packing problem as an algorithm for workload consolidation. In previous articles, we have published two algorithms. Which were used to request and detect deadlock in resources allocation heterogeneous distributed platforms [14] [15]. After we provide prevent and avoid deadlock. The optimization problems of resources based the recovery of resources allocated. The resources of the physical server to provide resources for the virtual machine requirements are usually finite at a time. Therefore, resource-based research solutions need to be addressed. In order to improve and respond effectively, it is necessary to update resource allocation policies to re-schedule resource allocation in heterogeneous distributed platforms. Sotomayor et al. [7] proposed a lease-based model and implemented First-Come-First-Serve scheduling algorithm in homogeneous physical machines. With an algorithm greedy-based virtual machine mapping to map leases that include some of the virtual machines with/without start time. In the context of the dynamic cloud environment, the number of computing resource requirements increasing, moreover the tendency of service-oriented applications (IoT). The proposed solution [8] will no longer be appropriate; instead, support mechanisms need to be automated, self-updating and effectively utilizing physical server resources.

In [16] [17] have presented the strategies of power aware virtual machine placement techniques on a survey. They have discussed the used optimization algorithms to save power. They have classified the energy saving techniques in a data center into static and dynamic methods. They have included in the Static Power Management class and Dynamic Power Management the techniques. Therefore, when proposing energy-efficient resource allocation, one needs to be aware of the SLA to avoid performance degradation of the consumer applications, including increased response times, timeouts or even failures. Therefore, Cloud providers have to establish QoS requirements to avoid SLA violations and meeting the QoS requirements while minimizing energy consumption.

3 A Resource Allocation System

3.1 The M VM-out-of-N PM model

A heterogeneous distributed platforms are composed of a set of an asynchronous processes (p_1, p_2,. . .,p_n) that communicates by message passing over the communication network [15], [2]. Based on the basic work of the authors Kshemkalyani-Singhal, and other works such as Menasce-Muntz, Gligor - Shattuck, Ho - Ramamoorthy, Obermarck, Chandy, and Choudhary [17]. They have the same opinions that the requested resource model of distributed system is divided into five model resource requirements. It's simple resource models, resource models OR, AND resource models, models AND/OR, and model resource requirements P-out-of-Q. Through this model, the researchers have discovered a technical proposal deadlock corresponding to each model. In this work, we use model P-out-of-Q as a prerequisite for developing research models provide resources in the cloud. The *n* VM-out-of-1PM problem depicts on-demand resource allocation to n VMS residing in N servers, where each VM may

use resources in more than one server concurrently. Thus, we model it to guide the design of algorithm prevents deadlock in resource allocation among VMs each of which may use the resource in various servers concurrently. In Fig. 1 the model resource allocation *M* VM-out-of-N PM, Fig.1 is shown the intended implementation of the developed trust management systems P2P.

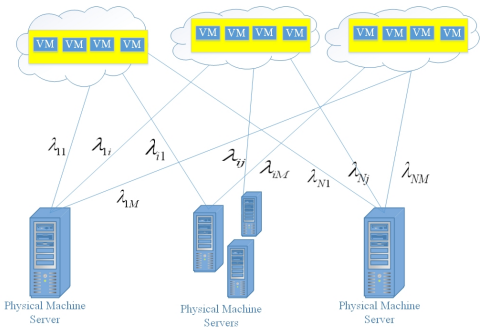

Figure 1: The model *M* VM-out-of-N PM

E_{ijt} is the amount of resources allocated to VM_{ij} at time t, where

$$E^{CPU} = A^{CPU} + \sum_{i=1}^{n}\sum_{j=1}^{m} C_{\mathrm{ij}}^{CPU} \tag{1}$$

The resource allocation problem is how to control the resource allocation to VMs with the goal of minimizing the function F_t, giving the limited resources. We get the following formulation:

$$\begin{aligned} F_t &= min \sum_{i=1}^{N}\sum_{i=1}^{V_i} \frac{f_{\mathrm{ij}}(EN_{\mathrm{ij}t}, \sum_{x=1}^{V} EO_{\mathrm{ij}t}^{x}, D_{ijt})}{\Phi_{ij}} \times SP_{ij} \\ &\begin{cases} \sum_{i=1}^{N}\sum_{i=1}^{V_i} E_{ijt} \le E \\ E_{\mathrm{ij}t} \ge C_{\mathrm{ij}} \quad (i = 1, 2, ..., V; j = 1, 2, ..., N) \\ \sum_{i=1}^{V_i} EN_{\mathrm{ij}t} + \sum_{j=1}^{V_i} EO_{\mathrm{ij}t}^{i} \le E_i \\ E_{it} \ge C_{ij} \quad (i = 1, 2, ..., V; j = 1, 2, ..., N). \end{cases} \end{aligned} \tag{2}$$

In this work, extended the basic mean variance model to a cardinality constrained mean variance model. The deails of the model is defined as follows.

$$\begin{aligned} &\min \textstyle\sum_{p=1}^{m}\sum_{i=1}^{l} R_{pi}X_{pi} \\ &subject \sum_{j=1}^{h} C_{\mathrm{ij}}Y_{ji} \le E_{it}; \forall_i = 1, 2, ...l; \\ &\quad \textstyle\sum_i Y_{ji} = 1; \forall_j = 1, 2, \ldots\ , l; \\ &\quad \textstyle\sum_i X_{pi} = 1, \forall_p = 1, 2, ..., m; \\ &\quad X_{pi} \le \textstyle\sum_j Y_{ji} \\ &\quad \forall_p \ \cup i = 1, 2, ..., l; \\ &\quad \cup \textstyle\sum_p C_p X_p \ \le \sum_j C_{ji} Y_{ji}; \forall_i; \\ &\quad \text{vari}able\ X_{pi} \in 0, 1 \cup Y_{ji} \in 0, 1 \end{aligned} \tag{3}$$

Fig. 2 shows the architecture of resource allocation system, which contains two figure lable *(a)*, *(b)*. Here figure *(a)* show a resource allocation contains two laye, input layer and output layer. Input layer show some request resource, output layer show grant resource.

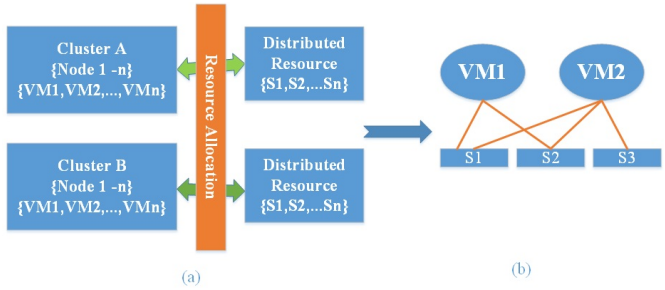

Figure 2: Resource Allocation System

We can use methods to prevent deadlock to solve optimal resource model provides *n* VM-out-of-N PM. Our algorithm is based on wait-for graphs (WFG) algorithm is presented in section 4.

3.2 A Resource Allocation System

In distributed systems, the state of the system can be modeled by directed graph, called a wait for graph (WFG). In a WFG, nodes are processors and there is a directed edge from node P_1 to mode P_2 if P_1 is blocked and is waiting for P_2 to release some resource. A system is deadlocked if and only if there exists a directed cycle or knot in the WFG [15], [12] [17].

A set $P = \{P_1, P_2, P_k\}$ of $k > 1$ entities is deadlocked when the following two conditions simultaneously hold:

- Each entity P_i P is waiting for an event permission that must be generated from another entity in the set;
- No entity P_i P can generate a permission while it is waiting.

If these two conditions hold, the entities in the set will be waiting forever, regardless of the nature of the permission and of why they are waiting for the permission; for example, it could be because P_i needs a resource held by P_j in order to complete its computation. A useful way to understand the situations in which deadlock may occur is to describe the status of the entities during a computation, with respect to their waiting for some events, by means of a directed graph, called wait-for graph [15] [17]. In heterogeneous distributed platforms, the state of the system can be modeled by a directed graph, called a wait for graph (WFG). In a WFG, nodes are processes and there is a directed edge from node P_1 to node P_2 if P_1 is blocked and is waiting for P_2 to release some resources. A system is deadlocked if and only if there exists a directed cycle or knot in the WFG [15], [2], [17].

Deadlock resolution involves breaking existing wait for dependencies between the processes to resolve the deadlock. Therefore, when a wait for dependency is broken, the corre-

sponding information should be immediately cleaned from the system. If this information is not cleaned in timely manner, it may result in prevention of phantom deadlocks.

The resource providers, consider providing a virtual machine request individually, going through the scheduling algorithm to determine. By applying the algorithm to prevent deadlock, will test the maximum amount of resources of the physical server at the provider. Through the vector components of each type of resource that will be used, as well as the total resource. The allocation is thus represented by two vector components vector element and the vector total.

As can be considered here is the case where a process p_i requires the resources it needs for its session one after the other, hence the name incremental requests. The main issue that has to be solved is the prevention of deadlocks. For instance, the processes p_8 and p_2 in Fig.1 can be taken in consideration when both processes want to acquire both the resources r_1 and r_2. As a matter of fact that prevention deadlock would help in providing resources as good as possible.

As it is commonly known, process must wait for the resources as deadlock occurred since they are being occupied by other processes. However, achieving effectively resource scheduling would greatly lessen such worse situation of deadlock. Therefore, proposed algorithm works as engine springing up the needed information about the process situations whether each one is in underway, lack or waiting and this fact could be graphically expressed through dependence graph presented in this paper.

Therefore, when process in a wait-for the dependency is broken, the corresponding information must be instantly restored from the system, otherwise, deadlock would be happening. Let $\{r_1, r_2, \ldots, r_n\}$ be the whole set of resources accessed by the processes, each process accessing possibly only a subset of them. Let $<$ be a total order on this set of resources. The processes are required to obey the following rule:

- During a session, a process may invoke request resource(r_k) only if the it has already obtained all the resources r_j it needs which are such $r_j < r_k$.
- As p_1 is owning the resource r_a and waiting for the resource r_b, it invoked first request resource(r_a) and then a request resource(r_b).
- As p_2 is owing the resource x_b and waiting for the resource r_a, it invoked first request resource(r_b) and then request resource(r_a).

4 Our Algorithm

In this paper, we will approach proposed algorithm for deadlock prevention maintains property of n-vertex directed graph when the new is added in the graph using two-way search. The time bound for the incremental cycle algorithm for deadlock prevention take O ($\sqrt{m}$) time bound for the m edge insertion in the directed graph. It reports the cycle when the algorithm detects for edge (v,m) that there exist a path from vertex w to v.

Methods of optimizing the use of functions in the formula 3. Optimal recovery method in materials allocated because the process still holds resources when finishing requirements. Data concerning the use of CPU, RAM and HDD were collected from the above physical machine servers at Data Center in every 4 hour during a period of 1 days.

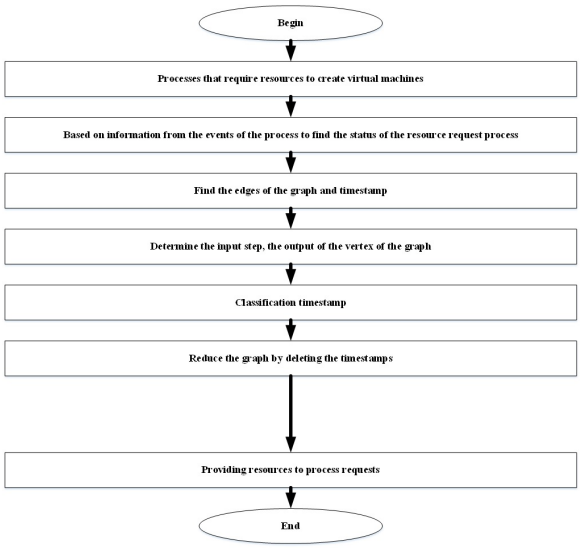

Figure 3: Flow chart of algorithm RRAA

In the flowchart Figure 3 using the demanding method in ideal conditions does not use technical solutions.

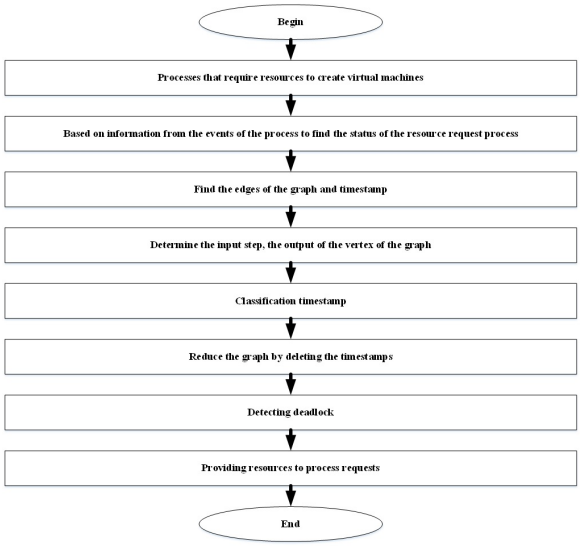

Figure 4: Flow chart of algorithm PDA

The first part was the set of data collected within the Algorithm 1 used for model *M* VM-out-of-N PM and the second part was the remaining data collected within the Algorithm 2 for testing. Compare data set obtained table we noticed that prevention deadlock results also brought confidence and the ability to bring greater efficiency.

In the flowchart Figure 4 using the demanding method in ideal conditions with technical solutions detecting deadlock.

Algorithm 1 Requests Resources Allocaition Algorithm (RRAA)

Input: $P_i^{j(CPU)^*}$, $P_i^{j(RAM)^*}$ from IaaS provider i;

Output: new resource $r_j^{CPU^{(n+1)}}$, $r_j^{RAM^{(n+1)}}$;

BEGIN

Operation request resource (r_i) in the critical section is

$csstate_i \longleftarrow$ trying;

$lrd_i \longleftarrow clock_i$ + 1;

for each $j \in R_i$ do

if ($usedby_i$[j] =0) the send request (lrd_i,i) to p_j end for;

$sentto_i$ [j] $\longleftarrow$ true;

$usedby_i$[j] $\longleftarrow$ R

else $sentto_i$[j] $\longleftarrow$false

end if

end for;

$usedby_i$[i] $\leftarrow k_i$;

$wait(\sum_{j=1}^{n} usedby_i[j] \leq NPM)$;

$csstate_i \longleftarrow$ in;

Operation release resource (r_i) in the critical section is

$csstate_i \longleftarrow$ out;

for each $j \in permdelayed_i$ do send permission(i,j) to p_j end for;

$R_i \leftarrow permdelayed_i$;

$permdelayed_i \leftarrow \oslash$

END.

5 Experiments and results

In this paper, the solution provides effective resources is done through two algorithms. We implement the designed deadlock prevention algorithm on a physical machine server with an Intel E5-2603V3 processor and 16G memory.

Algorithm resource requirements and algorithms prevent deadlock in resource supply. Based on the resource model provides *M* VM-out-of-N PM.

CloudSim supports VM Scheduling at two levels:

- First, at the host level where it is possible to specify how much of the overall processing power of each core in a host will be assigned at each VM.
- Second, at the VM level, where the VMs assign specific amount of the available processing power to the individual task units that are hosted within its execution engine.

Algorithm 2 Deadlock Prevention Algorithm (PDA)

Input: $P_i^{j(CPU)^*}$, $P_i^{j(RAM)^*}$ from IaaS provider i;
Output: new resource $r_j^{CPU^{(n+1)}}$, $r_j^{RAM^{(n+1)}}$;
BEGIN
When REQUEST(k,j) is received from p_j *do*
$clock_i \leftarrow$ max($clock_i$,n);
$prio_i \leftarrow$ (csstate$_i$ = in) $\vee$ ((csstate$_i$ = trying) $\wedge$ ((lrd_i,i) ¡ (n,j)));
if ($prio_i$) then send NOTUSED(1PM) to p_j
else if($n_i \neq$ 1PM) then send NOTUSED(1PM - n_i) to p_j end if
$permdelayed_i \leftarrow permdelayed_i \cup$ j
end if.
When permission(i,j) is received from p_j *do*
$NPM_i \leftarrow NPM_i \setminus$ j;
When NOTUSED(x) is received from p_j do $usedby_i$[j] $\leftarrow$ $usedby_i$[j] -x;
if $((csstate_i = trying) \wedge (usedby_i[j] = 0) \wedge (notsentto_i[j])$
then send REQUEST(lrd_i,i) to p_j
$sentto_i$[j] $\leftarrow$ true;
$usedby_i$[j] $\leftarrow$ N PM;
end if.
END.

For comparison, the other server equipped with the same configuration have prevention deadlock algorithm, named native.The data were separated into two parts.

For simulation we need a special toolkit named CloudSim. It is basically a Library for Simulation of Cloud Computing Scenarios. It has some features such as it support for modeling and simulation of large scale Cloud Computing infrastructure, including data centers on a single physical computing node.

It provides basic class for describing data centers, virtual machines, applications, users, computational resources, and policies.

We saw infrastructure of Cloud, in which Data Center consist of different Hosts and the Host manages the VM Scheduler and VMs.

Cloud-let Scheduler determines how the available CPU resources of virtual machine are divided among Cloud lets. There are two types of policies are offered: Space - Shared (Cloud-let Scheduler Space Shared): To assign specific CPU cores to specific VMs.

VM Scheduler determines how many processing cores of a host are allocated to virtual machines and how many processing cores will be delegated to each VM. It also determine how much of the processing core's capacity will effectively be attributed for given VM.

Time-Shared (Cloud-let Scheduler Time Shared): To dynamically distribute the capacity of a core among VMs, test data are as follows [18]:

Table 1: Optimal time of our algorithm (RRAA)

Method	Cloudlet ID	PM ID	VM ID	Start time	End time	Finish time (%)
0	1	1	1	0.1	120	25.22%
1	2	2	2	0.1	130	26.00%
2	3	2	4	0.1	132	27.27%
3	4	3	5	0.1	135	33.75%
4	5	3	3	0.1	150	36.39%
5	6	4	6	0.1	170	39.05%
6	7	4	7	0.1	185	42.86%
7	8	1	8	0.1	198	44.44%
8	9	2	9	0.1	199	50.55%
9	10	3	12	0.1	200	54.86%
10	11	4	10	0.1	215	64.86%
11	12	1	13	0.1	220	64.94%
12	13	2	11	0.1	225	70.55%
13	14	3	17	0.1	230	74.86%
14	15	3	24	0.1	235	74.96%
15	16	4	23	0.1	215	64.86%
16	17	1	22	0.1	220	64.94%
17	18	2	21	0.1	225	70.55%
18	19	3	20	0.1	233	74.87%
19	20	3	19	0.1	236	74.97%
20	21	4	18	0.1	235	74.96%
21	22	1	15	0.1	240	80.44%
22	23	2	25	0.1	245	80.55%
23	24	3	16	0.1	250	84.86%
24	25	3	14	0.1	255	86.96%

Table 1 shows that there is data-set simulation with 10 Cloud-let ID, when compared the with some VM ID in Table 1 usually between 8 and 10. According to the results rate has a VM 8 quickly VM 10. The comparative analysis of Table 1 can be seen in many times, after execution, although there were individual time with RRAA algorithm respone time not effectively.

Table 2 show that there is a noticeable difference between there two algorithm. The comparative analysis of experimental result can be seen in many times, apter task execution, although there were individual time improved RRAA algorithm response time was not significantly less than an optimal time algorithm, in most cases, improved algorithm is better than the optimal time algorithm, thus validated the correctness and effectiveness.

Table 2: Optimal time of our algorithm (PDA)

Method	Cloudlet ID	PM ID	VM ID	Start time	End time	Finish time (%)
0	1	1	1	0.1	70	10.22%
1	2	2	2	0.1	86	15.00%
2	3	2	3	0.1	92	17.27%
3	4	3	4	0.1	95	18.75%
4	5	3	5	0.1	100	19.39%
5	6	4	6	0.1	120	20.05%
6	7	4	7	0.1	130	21.86%
7	8	1	8	0.1	148	24.44%
8	9	2	9	0.1	160	25.55%
9	10	3	10	0.1	170	25.86%
10	11	4	11	0.1	120	20.05%
11	12	4	12	0.1	121	21.86%
12	13	1	13	0.1	122	24.44%
13	14	2	14	0.1	130	25.55%
14	15	3	15	0.1	145	25.86%
15	16	4	16	0.1	150	26.05%
16	17	4	17	0.1	151	26.86%
17	18	4	18	0.1	152	26.44%
18	19	4	19	0.1	160	26.55%
19	20	3	20	0.1	165	26.86%
20	21	4	21	0.1	170	30.65%
21	22	4	22	0.1	171	31.76%
22	23	4	23	0.1	172	32.74%
23	24	4	24	0.1	174	33.75%
24	25	3	25	0.1	175	34.88%

The averaged resource usage of different execution strategies are reported in Fig. 5.

The comparative analysis of experimental result can be seen in many times, apter task execution, although there were individual time improved PDA algorithm response time was not significantly less than an optimal time algorithm, in most cases, improved algorithm is better than the optimal time algorithm, thus validated the correctness and effectiveness. The averaged resource usage 25% - 30% of different execution strategies are reported in Fig. 6.

More exactly, the averaged CPU, Memory and Network utilization rates are 30 % show in Fig 5. Due to the barrier synchronization, CPU and network utilization rate fluctuate dramatically in each graph processing super step. In contrast,the memory utilization rate is quite stable, as most memory are used to store the graph structure. At the end of execution,the memory usage reduces slightly as the number of active verticals reduced.

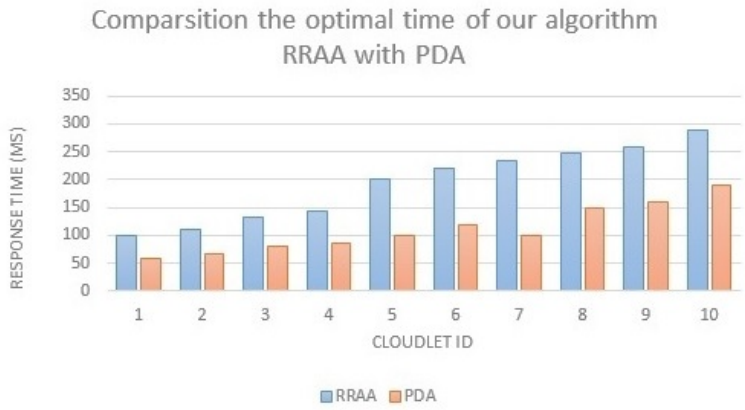

Figure 5: Comparative analysis of experimental result algorithm RRAA with PDA

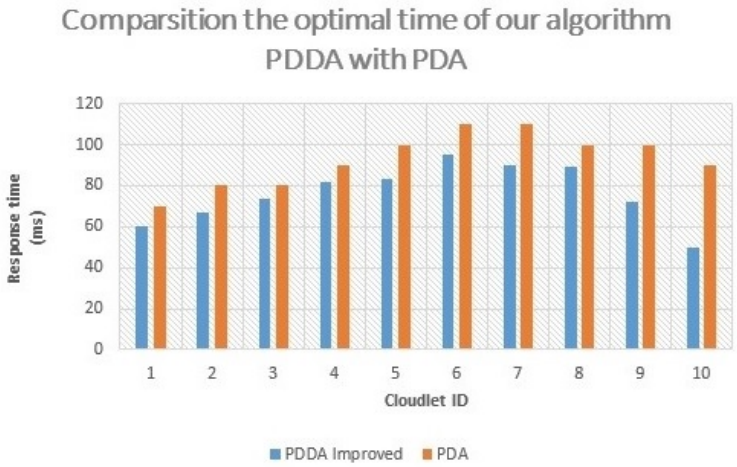

Figure 6: Comparative analysis of experimental result PDDA Improved with PDA

Similar to above observations, instance resources are not fully utilized in all execution strategies. Generally, the higher instance capability, the higher CPU usage and network traffic; and the more instances, the lower CPU usage and network traffic. In the case of 16 instances, the CPU usage increases proportionally to the number of CPU cores, which indicates that the graph vertex programs are run by CPU cores in parallel.

6 Conclusion

A prevention deadlock algorithm is implemented for resource allocation on heterogeneous distributed platforms. The prevent deadlock algorithm has O(m(n-1)/2) time complexity, an improvement of approximate orders of magnitude in practical cases. In this way, programmers can quickly prevent deadlock and then resolve the situation, e.g., by releasing held resources.

We focus on the application of the algorithm to prevent deadlock, when the process enters the critical section. From here we proceed to reschedule delivery system resources in a physical server. The application of the algorithms prevent this impasse in the proposed offer based resources distributed heterogeneous virtual machines require. Through this study, we found that the application of algorithms for preventing deadlock would best performance and take advantage of the physical resources.

The algorithms we have discussed solve the problem of deadlock in two phases: 1) it constructs the WFG of the system, and 2), it searches for the cycles. Due to the lack of globally shared memory, the design of the algorithms is difficult because sites may report the existence of a global cycle after seeing segments of the cycle at different instants, even though all the segments never existed simultaneously.

Through this research, we found that need the application method new would give optimal performance to distributed resources of heterogeneous distributed platforms. In future, we propose algorithm distributed in resource allocation.

Acknowledgement

Authors are very much thankful to the reviewers for providing constructive comments. It has helped a lot to improve the quality of the paper. I am also thankful to my institution for providing me sufficient time and place to execute my research work.

References

[1] Yazir, Y.O., Matthews, C., Farahbod, R.: Dynamic Resource Allocation in Computing Clouds using Distributed Multiple Criteria Decision Analysis. IEEE 3rd International Conference on Cloud Computing, pp. 9198, (2010). DOI: 10.1109/CLOUD.2010.66

[2] Ha Huy Cuong Nguyen, Hung Vi Dang, Nguyen Minh Nhat Pham,Van Son Le, Thanh Thuy Nguyen. Deadlock detection for resources allocation in heterogeneous distributed platforms, *Proceedings of 2015 Advances in Intelligent Systems and Computing*, June 2015, Bangkok, Thailand, Spinger, Volume 361, Issue 2, pp. 285-295, (2015). DOI:10.1007/978-3-319-19024-229.

[3] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D.,Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commune. ACM 53(4), 5058 (2010). DOI:10.1145/1721654.1721672.

[4] Srikantaiah, S., Kansal, A., Zhao, F.: Energy aware consolidation for cloud computing. Cluster Comput. 12, 115 (2009). DOI:10.1016/j.future.2011.04.017.

[5] Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener. Comput. Syst. 28(5),755768 (2012). DOI:10.1016/j.future.2008.12.001.

[6] Buyya, R., Yeo, C., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 25(6), 599616 (2009). DOI: 10.1016/j.future.2008.12.001.

[7] Sotomayor, B.: Provisioning Computational Resources Using Virtual Machines and Leases. Ph.D. thesis, University of Chicago (2010).

[8] Sotomayor, B., Keahey, K., Foster, I.T.: Combining batch execution and leasing using virtual machines. In: HPDC, pp. 8796 (2008). DOI:10.1145/1383422.1383434.

[9] Warneke et al, Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. IEEE Trans. Parallel Distrib. Syst. 22(6), 985-997 (2011). DOI: 10.1109/TPDS.2011.65.

[10] Amazon EC2, https://aws.amazon.com/ec2/.

[11] Eugen Feller, Lavanya Ramakrishnan, Christine Morin, Performance and energy efficiency of big data applications in cloud environments: A Hadoop case study, Journal of Parallel and Distributed Computing, Elsevier, 79-80, pp.80-89, (2015). DOI: 10.1016/j.jpdc.2015.01.001.

[12] Chandy, K. M., Misra, J., and Haas, L. M., Distributed Deadlock Detection, ACM Trans. on Computer Systems, May 1983. DOI: 10.1145/357360.357365.

[13] Vouk, M.A.: Cloud computing: Issues, research and implementations. In: Information Technology Interfaces. ITI 2008. 30th International Conference on, 2008, pp. 3140, (2008). DOI: 10.1109/ITI.2008.4588381.

[14] Ha Huy Cuong Nguyen, Van Son Le, Thanh Thuy Nguyen. Algorithmic approach to deadlock detection for resource allocation in heterogeneous platforms,*Proceedings of 2014 International Conference on Smart Computing*, 3-5 November, HongKong, China, IEEE Computer Society Press, pp. 97-103, (2014). DOI: 10.1109/SMARTCOMP.2014.7043845.

[15] Ha Huy Cuong Nguyen, Dac Nhuong Le,Van Son Le, Thanh Thuy Nguyen. A new technical solution for resources allocation in heterogeneous distributed plaforms, *Proceedings of 2015 The Sixth International Conference on the Applications of Digital Information and Web Technologies(ICADIWT2015)*, 10-12 Feb 2015, Macau, China, IOS Press, Volume 275, Issue 2, pp. 184-194, (2015). DOI: 10.3233/978-1-61499-503-6-184.

[16] Wu, L., Garg, S.K., Buyya, R.: SLA-based Resource Allocation for a Software as a Service Provider in Cloud Computing Environments. In: Proceedings of the 11th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2011), Los Angeles, USA, May 23-26, (2011).DOI: 10.1109/CCGrid.2011.51.

[17] Ajay D. Kshemkalyani, M.S., Distributed Computing Principles, Algorithms and Systems. 2008, UK: Cambridge University Press. DOI: 10.1017/CBO9780511805318.

[18] www.cloudbus.org/cloudsim/