



**INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH
TECHNOLOGY**
**RECOVERY AND USER PRIORITY BASED LOAD BALANCING IN CLOUD
COMPUTING**

Er. Rajeev Mangla *, Er. Harpreet Singh

* Computer Science and Engineering, Swami Vivekanand Institute of Engineering and Technology, India

ABSTRACT

Cloud Computing becomes an important aspect in computer science. For systematic use of the cloud, an efficient load balancing algorithm is required for scheduling the tasks in a well organized and logical manner. The Min-Min algorithm is an efficient approach to enhance the total completion time of the tasks. The major shortcoming is, it let load imbalanced on the resources. This drawback can be removed by using an improved load balancing Min-Min algorithm (LBIMM). User priority- an another aspect, which plays a vital role in terms of pay-per-use base. Cloud providers offers different type of QoS to put up the demands for different type of users. To provide the guarantees, load balancing algorithm must consider User Priority and Failure Recovery. Availability is considered as the growing and reoccurring concern in software intensive systems. PA-LBIMM considers user priority and seeks to minimize the total completion time. It fails to define, what will happen if a resource fails/crashes? To remove this constraint, a failure recovery policy is proposed in this paper. So that, if a resource fails then the tasks must be rescheduled to achieve minimum completion time. At last, the introduced policy is simulated using Matlab toolbox. The results show that the policy can led to significant rise in performance of the resource utilization.

KEYWORDS: Cloud Computing; Load Balancing; Recovery Policy; PA-LBIMM Algorithm; Makespan; Task Scheduling.

INTRODUCTION

Cloud Computing is a resourceful technology which has a great potential to provide dynamic services on large and scalable virtualized resources over the network [1][2]. Cloud is a mishmash of various homogenous as well as heterogenous resources which subject to fulfill the requirements of the cloud user and load balance the usage of the resources for the cloud providers [12]. Various characteristics such as On demand self-service, Resource pooling, Multi-tenancy, Rapid elasticity, etc. must be possessed by a cloud [8],[9]. In order to efficiently utilize the resources of the cloud, an efficient load balancing and scheduling algorithm is required. In this paper, a recovery policy is proposed which describes the action to be taken when a resource fails.

In a cloud environment, tasks are submitted by clients to the scheduler. The Scheduler check for the availability of the resources. Then scheduling is done on the resources according to the task's requirements[3]. Now, tasks are ready to be executed. After execution results are provided to the users as a reply from the cloud. Scheduling tasks efficiently is a challenge as clouds can be heterogenous in nature by

architecture, resource providers and consumers, operating system, etc.[7]. Problem arises when a resource under execution or ready state fails. The users must wait for the results until the resource recover from failure and become ready to execute the task. It may led to large task queues which a cloud provider never wants. The main purpose of scheduling algorithm is to decrease the completion time of all the submitted tasks, efficiently utilize the resources and provide an effective failure recovery policy[11]. Min-Min algorithm enhances the completion time but didn't able to do proper load balancing [6]. LBIMM enhances the overall completion time, do load balancing efficiently but never considers user-priority [7]. PA-LBIMM considers user priority and minimize the completion time according to user priority. In pay-per-use base, cloud users may be offered with different level of services i.e. VIP level or Ordinary level. Availability is the growing and reoccurring concern in software intensive systems as there is always the probability of the system failure. A system failure can take place due to various factors i.e. may be of operational

deadlock, power failure, security breaches or due to some other reasons. In the above discussed algorithms, No algorithm considers resource failure [6]. To provide the guaranteed service, Resource failure must be considered during execution of the tasks as availability is the main concern in cloud computing [10].

Recovery policy ensures the availability of the resources to the users even under the condition of resource failure. According to the simulated results, the proposed policy outperforms the other algorithms which never use this recovery policy in terms of makespan. The remaining part of the paper is organized as follows: Section II presents outline of the previous work about task scheduling algorithm with emphasis on Min-Min, LBIMM and PA-LBIMM. Section III, the proposed recovery policy is introduced. In Section IV, the implementation and results are given. Finally, Section V concludes the paper and presents future works.

RELATED WORK

As described in section I, Uniqueness of the resources in the cloud made it more challenging to do scheduling. A number of algorithms were discussed in the literature review. Tracy D et al [11] have studied the various scheduling algorithms such as Minimum Completion Time (MCT), Minimum Execution time (MET), Min-Min, Max-Min, etc. The result shows that Min-Min algorithm executes the tasks in minimum makespan which means it produces a better schedule than others. Huankai Chen et al [7] studied the Traditional Min-Min algorithm and considered it as a base algorithm to propose Load Balance Improved Min-Min scheduling algorithm (LBIMM) and User-Priority Aware Load Balance Improved Min-Min scheduling algorithm (PA-LBIMM). To test the results a simulation basis is also provided.

TRADITIONAL MIN-MIN SCHEDULING ALGORITHM

Min-Min algorithm is a simple algorithm that starts with an unmapped task set S [4]. Resource R with minimum completion time for tasks in the set S is selected from the available resources. Then the smallest task T from S is assigned to the corresponding resource R . T is removed from the task set S and the procedure is repeated until task set S is empty [5].

Various steps of Min-Min algorithm are represented in Fig. 1. Assume we have task set $S=(T_1, T_2, T_3 \dots T_x)$ of x tasks, which we want to schedule over y resources (R_1, R_2, \dots, R_y) . Expected

completion time C_{ij} produced by resource j ($1 \leq j \leq y$) for task i ($1 \leq i \leq x$) is calculated as in (1)

$$C_{ij} = E_{t_{ij}} + R_{t_j} \quad (1)$$

Here $E_{t_{ij}}$ denotes time required by task T_i to execute on resource R_j . R_{t_j} represents ready time of the resource R_j .

```

Step 1: For tasks in set  $S$ ;  $T_i$ 
Step 2: For all resources;  $R_j$ 
Step 3:  $C_{t_{ij}} = E_{t_{ij}} + R_{t_j}$ ; End For; End For;
Step 4: Do while task set is not empty
Step 5: Find task  $T_k$  that cost minimum execution time.
Step 6: Assign  $T_k$  to the resource  $R_j$  which gives minimum expected complete time
Step 7: Remove  $T_k$  from the tasks set
Step 8: Update ready time  $r_{t_j}$  for select  $R_j$ 
Step 9: Update  $C_{ij}$  for all  $T_i$ 
Step 10: End Do

```

Figure 1: Traditional Min-Min Scheduling Algorithm

A major weakness of Min-Min algorithm is that it emphasizes on the minimum completion time of all the tasks without considering work load on each resource [4]. This led some resources busy all the time and other may be ideal.

LOAD BALANCE IMPROVED MIN-MIN SCHEDULING ALGORITHM (LBIMM)

LBIMM algorithm is enhanced form of Min-Min algorithm. LBIMM improves the load balancing which is a major weakness in case of Min-Min algorithm [7]. LBIMM stresses to remove the load unbalance and to minimize the execution time of each resource effectively.

LBIMM algorithm's pseudo code is represented in Fig 2. As it is based on Min-Min algorithm, therefore it executes Min-Min at first step. It then searches for the most heavily loaded resource and the smallest task allotted to that resource. Then the completion time is calculated for that task on all other resources. Then the makespan produced by Min-Min is compared with minimum completion time of that task. If completion time is less than makespan then the task is reassigned to that resource which produces it. The ready time is updated for both resources. This process repeats until the makespan produced by the heavy load resource becomes less than completion time on the newly selected resource. In this way, load balancing is achieved and LBIMM

produces a schedule reduces the overall completion time [7].

```

Step 1: For tasks in set S; Ti
Step 2: For all resources; Rj
Step 3: Ctij=Etij + Rtj; End For; End For;
Step 4: Do while tasks set is not empty
Step 5: Find task Tk that cost minimum execution time.
Step 6: Assign Tk to the resource Rj which gives minimum expected complete time
Step 7: Remove Tk from the tasks set
Step 8: Update ready time rtj for select Rj
Step 9: Update Cij for all Ti
Step 10: End Do
Step 11: Do while the most heavy load resource is considered, no need of rescheduling
Step 12: Find the task Ti that has minimum execution time on heavy load resource Rj
Step 13: Find the minimum completion time of Ti produced by resource Rk
Step 14: If minimum completion time (Rk) < makespan
Step 15: Reassign Task Ti to Resource Rk
Step 16: Update ready time of both Rj and Rk
Step 17: End If; End Do;

```

Figure 2: Load Balance Improved Min-Min Scheduling Algorithm (LBIMM)

USER-PRIORITY AWARED LOAD BALANCE IMPROVED MIN-MIN SCHEDULING ALGORITHM (PA-LBIMM)

No user-priority is taken into consideration in both Min-Min and LBIMM algorithms. For cloud to be a pay-per-use base, User's must be isolated from each other. Huankai Chen et al [7] proposed a user-priority based PA-LBIMM algorithm.

PA-LBIMM separate the tasks into G1 and G2 groups. The tasks submitted by VIP user's or high priority user's are considered as group G1 and tasks submitted by low priority user's are considered as group G2. Tasks are scheduled to the resources on the priority basis. Firstly, For all the tasks in G1, each task is assigned to the VIP category resource by using Min-Min. Then each task in G2 group is assigned to all the resources by using Min-Min. Now, load balancing function of LBIMM algorithm is executed to load balance all the resources. In this way, an optimal load balanced schedule is generated[7]. The pseudo code for PA-LBIMM scheduling algorithm is given in Figure 3.

PA-LBIMM outperforms both LBIMM and Min-Min as per discussion and results discussed in the

[http:// www.ijesrt.com](http://www.ijesrt.com)

literature of Huankai Chen et al [7]. So, this algorithm can be considered for the further study and research work can be done by considering PA-LBIMM.

```

Step 1: Divide the task set S into two groups VIP G1 and Ordinary G2 agreeing to the user-priority demand
Step 2: For all submitted tasks of G1 group
Step 3: For all VIP resources; Rj
Step 4: Ctij=Etij+rtj;
Step 5: End For; End For;
Step 6: Do while tasks set is not empty
Step 7: Find task Tk with minimum execution time.
Step 8: Assign Tk to the VIP resource Rj which gives minimum expected completion time
Step 9: Remove Tk from the S
Step 10: Update ready time rtj for selected Rj
Step 11: Update Cij for all Ti
Step 12: End Do
Step 13: For all submitted tasks of G2 group
Step 14: For all resources; Rj
Step 15: Ctij=Etij+rtj;
Step 16: End For; End For;
Step 17: Do while tasks set is not empty
Step 18: Find task Tk with minimum execution time.
Step 19: Assign Tk to resource Rj which gives minimum expected completion time
Step 20: Remove Tk from the S
Step 21: Update ready time rtj for selected Rj
Step 22: Update Cij for all Ti
Step 23: End Do
Step 24: Do while the most heavy load resource is considered, no need of rescheduling
Step 25: Find the task Ti that has minimum execution time on heavy load resource Rj
Step 26: Find the minimum completion time of Ti produced by resource Rk
Step 27: If minimum completion time (Rk) < makespan
Step 28: Reassign Task Ti to Resource Rk
Step 29: Update ready time of both Rj and Rk
Step 30: End If; End Do;

```

Figure 3: User-Priority Awared Load Balance Improved Min-Min Scheduling Algorithm (PA-LBIMM)

PROPOSED WORK

In this paper, a recovery policy is proposed which helps the cloud scheduler to reschedule the tasks if a resource fails at the time of execution to achieve the minimum makespan.

Recovery Policy

According to this policy, First of all, scheduler looks for the failed resource R_{f_j} . All the tasks that were scheduled by PA-LBIMM to execute on R_{f_j} will be considered as a task set S_f . Now, ready time rt_{f_j} (which is equal to the time span between the start of the execution on the resource and time of failure occurrence) for the failed resource is calculated. So, the completion time of the tasks in task set S_f become equal to the sum of previous completion time on that resource (Ct_{ij}) and ready time (rt_{f_j}).

$$FCt_{ij} = Ct_{ij} + rt_{f_j} \quad (2)$$

One task T_k from the set S_f is selected and its completion time is calculated on all the resources (Here completion time on other resources is equal to the sum of their makespan and execution time of the task T_k on that resource). The task T_k will be assigned to the resource R_k which produces minimum completion time. Ready Time of both R_{f_j} and R_k will be updated. The task T_k is removed from the set S_f . The procedure will be repeated until the task set S_f becomes empty. The psudo code is given in Figure 4.

```

Step 1: For tasks in set S; Ti
Step 2: For all resources; Rj
Step 3: Ctij = Etij + Rtj; End For; End For;
Step 4: Do while tasks set is not empty
Step 5: Apply PA-LBIMM
Step 6: If resource failure occurs
Step 7: For tasks in set Sf; Tk
Step 8: For all resources; Rfj
Step 9: FCtij = Ctij + rtfj;
Step 10: End For; End For;
Step 11: Find the minimum completion time of Tk produced by resource Rk
Step 14: If minimum completion time (Rk) < completion time of (Rfj)
Step 15: Reassign Task Tk to Resource Rk
Step 16: Update ready time of both Rfj and Rk
Step 17: End If;
Step 18: End If;
Step 19: End Do;
    
```

Figure 4: Recovery Policy Based User-Priority Awared Load Balance Improved Min-Min Scheduling Algorithm (RPA-LBIMM)

An Illustrative Example of RPA-LBIMM Scheduling Algorithm

Assume we have a setup of three available resources to which various users can submit their tasks.

<http://www.ijesrt.com>

Suppose five tasks have been submitted by users. Table 1, represents represents the id, size and the user group of each task. Table 2, represents the id, processing speed and service level/type of each resource. Data present in Table 1 and Table 2 is used to calculate the expected execution time and completion time of each task on each of the resources.

Table 1. Task Specification

Task_Id	Task Size (MB)	User Group
T(1)	100	Ordinary
T(2)	150	Ordinary
T(3)	200	Ordinary
T(4)	250	VIP
T(5)	500	Ordinary

Table 2. Resource Specification

Resource_Id	Resource Speed (Mbps)	Type
R(1)	20	VIP
R(2)	16	Ordinary
R(3)	10	Ordinary

Table 3. Execution Time of Tasks on Each of the Resources by PA-LBIMM Scheduling Algorithm

Task/Resource	VIP-R(1)	R(2)	R(3)
T(4) - VIP		15.625	
T(1)		6.25	
T(2)			15
T(3)		12.5	
T(5)	25		

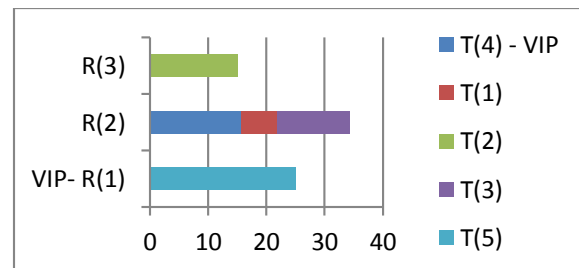


Figure 5: Gantt Chart: PA-LBIMM

Table 4. CompletionTime by PA-LBIMM Scheduling Algorithm After Failure of Resource R(2)

Task/Resource	VIP-R(1)	R(2)	R(3)
T(4) - VIP		19.625	
T(1)		25.875	

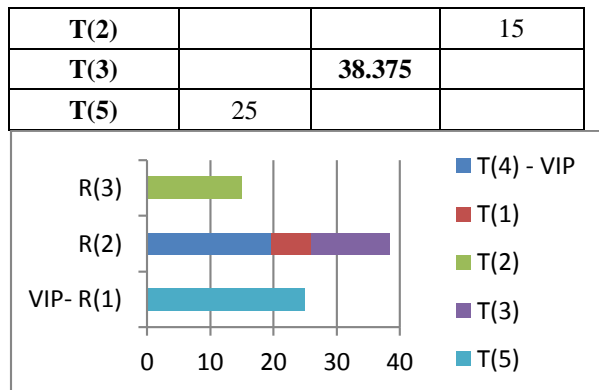


Figure 6: Gantt Chart: PA-LBIMM After Failure of Resource R(2)

Table 5. CompletionTime of Tasks on Each of the Resources by RPA-LBIMM Scheduling Algorithm

Task/Resource	VIP- R(1)	R(2)	R(3)
T(4) - VIP		19.625	
T(2)			15
T(5)	25		
T(3)	35		
T(1)			25

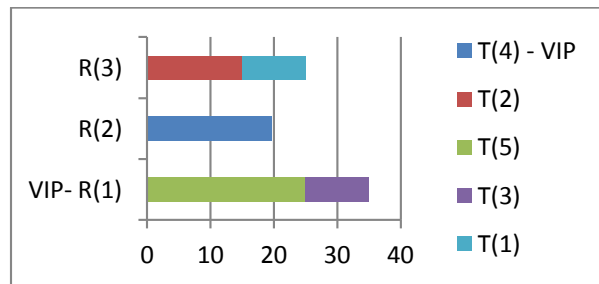


Figure 7: Gantt Chart: RPA-LBIMM

According to the Table 5, the makespan produced by RPA-LBIMM is 35 seconds which is less than the makespan produced by PA-LBIMM i.e. 38.375 seconds (as in Table 4) which is an important improvement. The makespan decreases by 8.88%. Thus RPA-LBIMM algorithm outperforms the PA-LBIMM whenever a resource failure occurs.

RESULTS

To evaluate the rate of increase in efficiency, the experiment is done considering various scenarios. The Scenerios depend upon the time of failure of the resource. Resource may fail:

- A) Before the start of execution of the tasks scheduled to that resource.

- B) At the time of execution.
- C) After the execution of the tasks scheduled to that resource.

So, experimental testing is performed on these all three scenerios.

Makespan can be considered as the throughput of the cloud system. It can be calculated as:

$$\text{Makespan} = \max(rt_j) \tag{3}$$

Here, rt_j denotes ready time of the resource after scheduled. The less the makespan, the better is the throughput.

In scenario A, the failure time can be considered as zero. So, it will not effect the makespan as resource will be in ready state at the time of execution starts.

In scenario B, assume the resource failure occur after 2 seconds of the start of execution. Consider the time required by resource to be ready again is 2 seconds. This scenerio is discussed in Table 4 and Table 5. The makespan reduces significantly by 8.88% as in Figure 8. So, RPA-LBIMM clearly outperformed the PA-LBIMM.

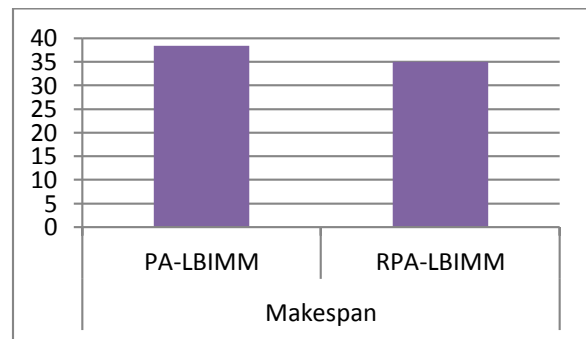


Figure 8: Gantt Chart: Makespan

In scenario C, the failure occurs after all the tasks executed completely. So, it will not effect the makespan as no task is in waiting state.

CONCLUSION AND FUTURE WORK

RPA-LBIMM proposed in this paper can be used to achieve high throughput in case of resource failure. Matlab simulation was used to evaluate the new algorithm under all possible scenerios. RPA-LBIMM produced the makespan of 35 seconds, which is 3.375 seconds less than makespan produced by PA-LBIMM. So, the makespan is improved by 8.88% by RPA-LBIMM.

This paper is concerned with the recovery from the failure, load balancing , makespan and user-priority for task scheduling in Cloud environment. Various



scheduling algorithms such as, Round Robin, Sufferage, First Come First Serve can be devised. Many issues remain open. QOS requirement, the heterogeneity of the resources and many other issues that can be topics of future research. Tasks are independent in this paper, but they may have some precedence relations in real-life situation. We will study and improve RPA-LBIMM for such kinds of tasks in the future.

REFERENCES

- [1] Tushar Desai, Jignesh Prajapati, "A Survey Of Various Load Balancing Techniques And Challenges In Cloud Computing" in INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 2, ISSUE 11, NOVEMBER 2013 ISSN 2277-8616
- [2] Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi and Jameela Al-Jaroodi, "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms" in 2012 IEEE Second Symposium on Network Cloud Computing and Applications
- [3] Mayanka Katyal, Atul Mishra, "A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment" in International Journal of Distributed and Cloud Computing Volume 1 Issue 2 December 2013.
- [4] Kobra Etminani , Mahmoud Naghibzadeh, Noorali Raeji Yanehsari, " A Hybrid Min-Min Max-Min Algorithm With Improved Performance".
- [5] Xiaogao Yu, Xiaopeng Yu, "A New Grid Computation-based Min-Min Algorithm" in 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery
- [6] T. Kokilavani, Dr. D.I. George Amalarethnam, "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing" in International Journal of Computer Applications (0975 – 8887) Volume 20– No.2, April 2011.
- [7] Huankai Chen, Professor Frank Wang, Dr Na Helian, Gbola Akanmu, "User-Priority Guided Min-Min Scheduling Algorithm For Load Balancing in Cloud Computing".
- [8] J.Srinivas, K.Venkata Subba Reddy, Dr.A.Moiz Qyser, "CLOUD COMPUTING BASICS" in International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 5, July 2012.

- [9] Qi Zhang, Lu Cheng, Raouf Boutaba, "Cloud computing: state-of-the-art and research challenges" in J Internet Serv Appl (2010) 1: 7–18 DOI 10.1007/s13174-010-0007-6
- [10] Zenon Chaczko, Venkatesh Mahadevan, Shahrzad Aslanzadeh and Christopher Mcdermid, "Availability and Load Balancing in Cloud Computing" in 2011 International Conference on Computer and Software Modeling IPCSIT vol.14 (2011) © (2011) IACSIT Press, Singapore
- [11] Tracy D, Braun, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems" Journal of Parallel and Distributed computing , Volume 61, Issue 6, Pages 810 – 837, 2001
- [12] Bhushan Lal Sahu, Rajesh Tiwari, "A Comprehensive Study on Cloud Computing" in International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 9, September 2012 ISSN: 2277 128X.

Author Bibliography

	<p>Rajeev Mangla received BEng degree in computer science in 2012 from Punjab Technical University. Currently, studying for a master degree on Computer Science and Engineering in the Swami Vivekanand Institute of Engineering and Technology, Punjab Technical University, India. Main research interests include cloud computing, load balancing strategy and routing strategy.</p>
	<p>Harpreet Singh received MEng degree in computer science in 2013 from Punjabi University. Currently, working as an Assistant Professor in Department of Computer Science and Engineering , Swami Vivekanand Institute of Engineering and Technology, India. Main research interests include cloud computing and image noise removal.</p>