ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 14, March 2017

SLA-based Virtual Machine Scheduler In Cloud Environment

D.Dhivya
Department of CSE
Kongu Engineering College
Perundurai
Erode
divyaduraisamy1995@gmail.com

M.Deebiga
Department of CSE
Kongu Engineering College
Perundurai
Erode
deepikamathan96@gmail.com

P.Gowthaman
Department of CSE
Kongu Engineering College
Perundurai
Erode
gowthaman.p1996@gmail.com

B.Krishnakumar
Assistant Professor
Department of CSE
Kongu Engineering College
Perundurai
Erode
Krishnakumarpri@gmail.com

ABSTRACT
Cloud computing is emerging as a new paradigm of large-scale distributed computing. In order to utilize the power of cloud computing completely, we need an efficient task scheduling algorithm. The traditional Min-Min algorithm is a simple, efficient algorithm that produces a better schedule that minimizes the total completion time of tasks than other algorithms in the literature. However the biggest drawback of it is SLA.Service-level agreement (SLA) is a major issue in cloud computing because it defines important parameters such as quality of service, uptime, downtime, period of service, pricing, and security. To overcome this drawback SLA-based minmin Scheduling algorithm aims at minimizing the total completion time as well as reducing the SLA violation during VM scheduling. The proposed algorithms support three levels of SLA determined by the customers.Furthermore, the algorithms incorporate the SLA gain cost for the successful completion of the service and SLA violation cost for the unsuccessful end of the service. The results of the proposed SLA-Min-Min have four performance metrics, namely makespan, average cloud utilization, gain, and penalty cost of the services in VM.

Keywords
Cloud computing; Service-level agreement; Task scheduling; Min-Min Algorithm; Minimum completion time; makespan;

1. INTRODUCTION
Currently Cloud computing has evolved as great potential technology that is known as a provider of dynamic services using very large scalable and virtualized resources over the Internet. The term cloud refers to the service provider that organizes all categories of resources like storage, computing, etc. In the cloud computing environment, three types of services in the form of infrastructure, platform and software are provided for the customers on the cloud market. IaaS provides the infrastructure for different functions such as storage and computing. Secondly, PaaS gives platform to the client so that the users can effortlessly make the applications. At last, SaaS provides software to the clients and it does not require installing the software.

Efficient allocation of resources is naturally associated with a Service Level Agreement (SLA) in service computing. SLA provides a wide range of services and the cost is decided between users accessing cloud service request model with their constraints. Cloud is subjected to User Requirement and other constraints that have direct effect on user consumption of resources controlled by cloud provider. In order to utilize the power of cloud computing completely, we need an effective and efficient task scheduling algorithm. Task scheduling algorithm is responsible for dispatching tasks submitted by users to cloud provider onto heterogeneous available resources. This paper focuses on the efficient tasks scheduling considering the total completion time of tasks, resources utilization and various service levels in a cloud environment. The SLA specifies various parameters such as quality of service (QoS),uptime, downtime, period of service,pricing,data protection,security and backup policy. The CSP earns money on successful completion of the customer's job. Otherwise, they pay the penalty cost for the violation of the SLA. The SLA for a particular cloud service may vary from one CSP to another CSP. However, in the multi-cloud environment, there is a requirement of one common SLA so that a unified service can be delivered to the customer. Therefore, task scheduling in heterogeneous multi-cloud environment is more challenging and not well studied in the current literatures.

The rest of this paper is organized in the following manner: The previous research on scheduling is discussed in Section 2.Section 3 introduces an existing algorithm and presents the task scheduling problem. Section 4 presents the proposed algorithm of SLA-based min-min scheduling. The experimental results are discussed in the Section 5. The conclusion is given in Section 6 along with the direction for future work.

2. LITERATURE REVIEW
Many SLA-based algorithms have been proposed that focus on the resource management, negotiation, fault tolerance, agreement violation, and cost analysis. Gao et al. have presented a dynamic resource management approach

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 14, March 2017

to achieve SLA in the data centers. Ranaldo et al. have proposed a SLA negotiation process of cloud services. However, they have taken workload as a precondition for SLA negotiation. Moreover, it does not incorporate the user and market requirements. Therefore, Maurer et al.have proposed an adaptive SLA matching approach which gives a flexibility to the cloud users to define mappings between SLA templates in the cloud market. Lu et al. have presented a new approach for optimized SLA negotiation. Nonetheless, dependent SLAs lead the SLA management system inefficiently. As a result, multi- domain SLA management is considered as a future work. Garcia et al. have introduced an algorithm that handles the resource life cycle of the generic SLA model. Emeakaroha et al. have proposed an architecture, called detecting SLA violation infrastruscture to monitor the agreement violations. However, detection of violations depends on the service-level objectives. Most of the literatures discussed above are not well suited for heterogeneous multi-cloud environment. In multi-cloud environment, the CSPs may have different SLAs in providing their services. Therefore, it is very difficult for the customer to select the CSP. One solution of this problem is that the CSPs can collaborate each other to provide services under a common SLA. The European commission DG CONNECT has taken initiative to set up cloud select industry group on service-level agreement (C-SIG-SLA), and they have prepared a document to provide standard guidelines for CSPs.However, this is limited to Europe nation boundary rather than spreading across the world. Later, this objective is achieved by ISO/IEC 19086. Baset et al. have studied the SLA guarantees and violation of CSPs. They have provided a guidance for the future SLA. In our proposed algorithms, we consider a common SLA for all the CSPs. Aazam et al. have provided a model to handle the service reservation and presented a set of customer characteristics in the dynamic environment. Franke etal. have presented an investigation on SLA decision making. The investigation consists of pairwise choices between various alternatives and uncertainty. Abawajy et al. have introduced an SLA management framework to optimize the QoS. This framework helps the service provider to avoid the violation of the SLA. However, the framework is limited to a single cloud environment. Ivanovic et al. have defined the SLA in the form of execution time, availability, and cost and modeled a constraint satisfaction problem for SLA violation.

Many one-phase and two-phase task scheduling algorithms have been proposed for grid computing environment and which are extended to cloud computing environment. Ibarra et al. have proposed Min-Min algorithm which is a two-phase scheduling algorithm.This algorithm also does not consider the execution cost. We have incorporated the cost in MCT and Min-Min and call these algorithms as Profit-MCT and Profit-Min-Min which aim to minimize the execution cost only. Recently, Farokhi et al. have presented a hierarchical SLA for the multi-cloud environment which is applicable for software as a service (SaaS) providers. The lack of service selection and SLA management make some issues, to move from multi-cloud SaaS providers to Infrastructure as a Service (IaaS). Li et al. have proposed two scheduling algorithms, namely cloud list scheduling and cloud Min-Min scheduling for multi-cloud environment. We have also proposed various scheduling algorithms for heterogeneous multi-cloud environment. However, the main objective of these algorithms is to minimize the makespan and to maximize the average cloud utilization. Moreover, the execution cost matrices (i.e., gain and penalty cost) are not considered in these algorithms. Christo Ananth et al. [4] discussed about a system, In this proposal, a neural network approach is proposed for energy conservation routing in a wireless sensor network. Our designed neural network system has been successfully applied to our scheme of energy conservation. Neural network is applied to predict Most Significant Node and selecting the Group Head amongst the association of sensor nodes in the network. After having a precise prediction about Most Significant Node, we would like to expand our approach in future to different WSN power management techniques and observe the results. In this proposal, we used arbitrary data for our experiment purpose; it is also expected to generate a real time data for the experiment in future and also by using adhoc networks the energy level of the node can be maximized. The selection of Group Head is proposed using neural network with feed forward learning method. And the neural network found able to select a node amongst competing nodes as Group Head.

In this paper, we deal with SLA-based task scheduling algorithms to make a balance between execution time and execution cost. The algorithms have following notable differences with the existing ones. (1) The proposed algorithms enable the customers to choose one of the SLA levels (i.e., 1, 2, or 3) that aim to minimize the execution time, cost, or both, whereas the existing algorithms deal with either execution time or cost. (2) The algorithms also facilitate the customers to select the weight values for execution time and execution cost parameters. As a result, the customers can determine which parameter is more important. However, the existing algorithms do not provide such facilities. (3) Although our proposed algorithms have the same complexity as that of existing algorithm existing algorithm, but they balance between the makespan and execution cost.

3. Traditional Min-Min Scheduling Algorithm

The Min-Min algorithm is simple and still basis of present cloud scheduling algorithm. It starts with a set S of all unmapped tasks. Then the resource R which has the minimum completion time for all tasks is found. Next, the task T with the minimum size is selected and assigned to the corresponding resource R (hence the name Min-Min). Last, the task T is removed from set S and the same

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 14, March 2017

procedure is repeated by Min-Min until all tasks are assigned (i.e., set S is empty).

The pseudo code of Min-Min algorithm is represented in Fig 1 assuming we have a set of n tasks (T_1, T_2, T_3 … T_n) need to be scheduled onto m available resources (R_1, R_2, R_3 … R_m). We denotes the Expected Completion Time for task i
(1≤i≤n) on resources j (1≤j≤m) as Ct_{ij} that is calculated as in (1), Where rt_j represents the Ready Time of resource R_j and Et_{ij} represents the Execution Time of task T_i on resource R_j.

$$Ct_{ij} = Et_{ij} + rt_j$$

1. **For** all submitted tasks in the set; T_i
2. **For** all resources; Rj
3. $Ct_{ij}=Et_{ij}+rt_j$; End **For**; End **For**;
4. **Do** while tasks set is not empty
5. Find task T_k that cost minimum executic time.
6. Assign T_k to the resource R_j which give minimum expected complete time
7. Remove T_k from the tasks set
8. Update ready time rj for select R_j
9. Update C_{ij} for all T_i
10. End **Do**

Traditional Min-min algorithm

An Illustrative Example Of Min-Min Scheduling Algorithm

In order to illustrate the Min-Min algorithm, assume we have five tasks submitted by different users for scheduling on two available resources. Table I, represents the processing speed and service level of each resource while Table II, represents the task size and the user group of each task. Data given in Table I and Table II are used to calculate the expected completion time and execution time of the tasks on each of the resources.

TABLE I. RESOURCES SPECIFICATION

Resources	Processing Speed(MB/sec)
R1	10
R2	8
R3	5

TABLE II. TASKS SPECIFICATION

Tasks	Task size(MB)
T1	10
T2	15
T3	20
T4	25
T5	50

Table III demonstrates calculated execution time of the tasks and expected complete time at the same time. On next step of the algorithm iteration, data in Table III will be updated until all tasks are allocated.

TABLE III. **EXECUTION TIME (EXPECTED COMPLETE TIME)** OF TASKS ON EACH OF THE RESOURCES : MIN-MIN SCHEDULING ALGORITHM

Task/Resources	R1	R2	R3
T1	1(1)	~~1.25(1.25)~~	~~2(2)~~
T2	~~1.5(2.5)~~	1.875(1.875)	~~3(3)~~
T3	2 (3)	~~2.5(4.375)~~	~~4(4)~~
T4	~~2.5 (5.5)~~	3.125(5)	~~5(5)~~
T5	5(8)	~~6.25(11.25)~~	~~10(10)~~

Challenges Of Min-Min Scheduling Algorithm In Cloud Computing

Based on the experimental result from the illustrative example Min-Min algorithm fails to utilize the resources efficiently which lead to a several problems like load imbalancing , SLA and so on.To reduce SLA violation in task scheduling an efficient SLA-based Min-Min scheduling Algorithm was proposed.

4. Proposed Algorithm

SLA-based min-min scheduling algorithm

In this section, we present our proposed scheduling algorithm, SLA-Min-Min. As stated earlier that SLA-Min-Min is a two-phase scheduling (off-line scheduling). In the single-phase scheduling, a task is assigned to a cloud as soon as it arrives in the system. In the two-phase scheduling, the VM are not assigned to the clouds as per their arrival in the system; instead, they are collected in a batch that is assigned at predetermined times. It is noteworthy to mention that two-phase scheduling is feasible in cloud if and only if the mappable VM set is updated in every scheduling step. Let us illustrate two-phase algorithm through the popular Min-Min algorithm. Consider three tasks (*T*1, *T*2, and *T*3) that have execution time 2, 6, and 4 time units and 3, 5, and 7 time units on cloud *C*1 and cloud *C*2, respectively. In phase 1, it finds the minimum completion time of the tasks (over the clouds) which are 2, 5, and 4, respectively. In phase 2, it takes 2 units of time which is minimum out of 2, 5, and 4.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 14, March 2017

Therefore, task $T1$ is assigned to cloud $C1$, and the ready (busy) time of cloud $C1$ is set to 2. The above two-phase process is repeated until there is no task in the mappable set. The key attribute of the proposed algorithms is SLA which is an agreement between CSP and customer.

Now for SLA minmin, consider an example as shown. There are 11 customers' tasks $T = \{T1, T2,..., T11\}$. We assume that the arrival time of these tasks is zero for the simplicity of illustration . These tasks are scheduled to three different clouds with variable computational resources. The ETC matrix in Table 1a represents the execution time of each task on three different clouds. Similarly, the EG and EP matrices in Table 4b, c represent the gain and penalty/violation cost of each task on different clouds. We assume that the violation cost is 50% of the agreement cost for the simplicity of the illustration. Note that each task takes different execution time, gain cost, and violation cost because we present this problem in heterogeneous multi-cloud environment. Table 4d shows the SLA level of the tasks and their weight values. Here, "–" denotes that the customer has not selected the weight values. The type of service with respect to SLA level is shown in Table 5. Note that "X" is used to bypass execution time/execution cost which a customer can adopt in his/her SLA.

Given the matrices in Table 4, the corresponding Gantt chart for SLA-Min-Min (without weight values).. The method is illustrated as follows. SLA-Min-Min finds the minimum completion time for all tasks $T1$ to $T11$ followed by the minimum completion time among them which is 2 for task $T11$ on cloud $C1$. However, the task $T11$ requires SLA level 1 which shows that the task needs the service with minimum execution time. Therefore, it selects cloud $C1$. The completion of task $T11$ results earning of 11 cost units for cloud $C1$. The ready time of cloud $C1$ is updated to 2. Next, it again finds the minimum completion time for all tasks, followed by the minimum completion time among them (i.e., 3 for task $T6$ on cloud $C2$). However, the task $T6$ requires SLA level 3 which shows that the task needs the service with minimum execution cost. Therefore, it selects cloud $C3$. Now the ready time of cloud $C3$ is updated to 18. Similarly, it assigns the other tasks to the respective clouds.

Table 4 Type of service with respect to SLA level

Level	Service	
	Execution time	Execution Cost
1	Yes	X
2	Yes	Yes
3	X	Yes

Notation	Definition
Q	Queue of all the tasks
$ET\ C$	Expected time to compute matrix
EG	Expected gain matrix
EP	Expected penalty matrix
$SL\ A$	Service-level agreement matrix

Expected penalty of the given 11 tasks will be half of EG and service for all the tasks will be generated through user input and its execution cost and time will be generated randomly.

5.a. An ETC matrix with 11 tasks and 3 clouds,

ETC	C1	C2	C3
T1	22	8	15
T2	14	25	9
T3	7	23	12
T4	26	15	6
T5	18	21	8
T6	7	3	18
T7	10	13	23
T8	5	8	16
T9	19	7	25
T10	13	19	24
T11	2	3	5

5.b.an EG matrix

EG	C1	C2	C3
T1	2	8	6
T2	6	2	8
T3	8	2	6
T4	1	6	8

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 14, March 2017

T5	**4**	**2**	**8**
T6	**8**	**10**	**4**
T7	**8**	**6**	**2**
T8	**10**	**8**	**4**
T9	**4**	**8**	**2**
T10	**6**	**4**	**2**
T11	**11**	**9**	**6**

Pseudo code for SLA-based task scheduling algorithms

Algorithm 2 SL A-Min-Min

Input: 1. The following 2D matrices: ET C, EG, and EP 2.matrix: SLA, WT , WC , and assign
Output:1.The following 1Dmatrices: M, G, and P
1: **while** Q = NU LL **do**
2: l = |Q|
3: Call FI N D-T ASK -SL A(ET C, EG, EP, SL A, WT , WC, l, m)
4: **end while**

Algorithm 3 FI N D-T ASK -SL A(ET C, EG, E P, SL A, WT , WC, l, m)

```
1: for h = 1, 2, 3, ⋯ , l do
2:     for j = 1, 2, 3, ⋯ , l do
3:         if assign[ j ] == 0 then
4:             minimum = ET C [ j, 1]+ M[1]
5:             taskindex = 1
6:             cloudi index = 1
7:             break
8:         end if
9:     end for
10:    for i = 1, 2, 3, ⋯ , l do
11:        if assign[i ] == 0 then
12:            for j = 1, 2, 3, ⋯ , m do
13:                if minimum > ET C[i, j ]+ M[ j ] then
14:                    minimum = ET C[i, j ]+ M[ j ]
15:                    taskindex = i
16:                    cloudi ndex = j
17:                end if
18:            end for
19:        end if
20:    end for
21:    SL A_Level = SL A[taskindex ]
22:    if SL A_Level == 1 then
23:        Call SC H E DU L E -T ASK S-EXECUT I O N (ET C, EG, EP, taskindex , m)
24:        assign[taskindex ]= 1
25:    else
26:        if thenSL A_Level == 3
27:            Call SC H E DU L E -T ASK S- PRO F I T (ET C, EG, EP, taskindex , m)
28:            assign[taskindex ]= 1
29:        else
30:            Set λ1 = WT [i ] and λ2 = WC [i ]
31:            Call SC H E DU L E -T ASK S-SL A(ET C, EG, EP, taskindex , m, λ1, λ2)
32:            assign[taskindex ]= 1
33:        end if
34:    end if
35: end for
```

2). Then it calculates the ET C, EG, and EP of all the tasks on different clouds. Next, it calls Algorithm 3 to select one task among the batch of tasks for scheduling (Line 3). The while loop iterates until there is no task in the global queue (Lines 1–4).

Algorithm 3 is used to perform the following things. (1) Like Min-Min, it selects a task with minimum completion time in each iteration (Lines 2–20). (2) Then it finds the SLA level of the selected task (Line 21). (3) Like SLA-MCT, it calls the task to the cloud based on the SLA level.

5. EXPERIMENTAL RESULTS

Table 6:Comparison of Min-Min and SLA-based Min-Min

Algorithm / factors	Execution time	Gain	Penalty
Min-Min	70	93	46.5
SLA Min-Min	62	71	35.5

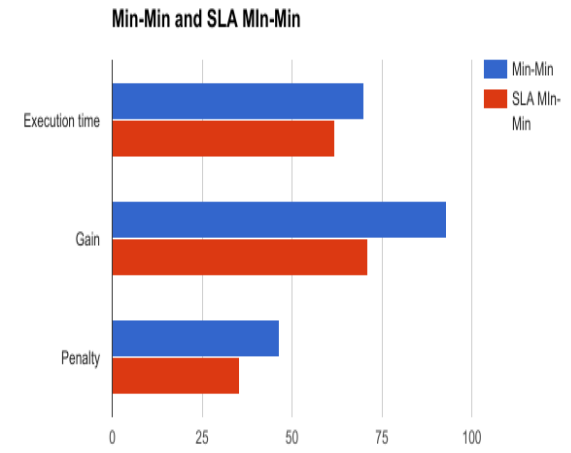

Fig: comparison of Min-Min and SLA-based Min-Min

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 14, March 2017

RESULTS AND DISCUSSION

In order to obtain results of the proposed algorithm the simulation was done using CloudSim 3.0.2 Simulator..In Our simulation scenario, the proposed algorithm is compared to the existing task scheduling algorithm, for this purpose following illustrative example is taken.

CONCLUSION

Cloud computing is a distributed computing which mainly focuses on providing services to the customers and it provides computational as well as storage resources to users. To utilize the resources efficiently, task scheduling provides the solution. Scheduling is the process in which the tasks are assigned to the VM's. In cloud computing environment, many algorithms are available to solve scheduling of tasks and resource allocation problems. Since scheduling of tasks in cloud computing is an NP-hard optimization problem, an efficient task scheduling strategy is required. The proposed task scheduling strategy aims at minimizing the total completion time as well as reducing the SLA problems for all tasks. It considers the process size, time and service type requirement of each task to realize the optimization for cloud computing environment. The proposed scheduling strategy provides the better result compared to traditional min-min scheduling.

In future, the work shall include other important QoS parameters such as performance, availability, and reliability. Our future research work will include all these parameters to develop more efficient algorithms for its applicability in the heterogeneous multi-cloud environment.

References

[1] Gao Y, Guan H, Qi Z, Song T, Huan F, Liu L (2014) Service level agreement based energy-efficient resource management in cloud data centers. Comput Electr Eng 40:1621–1633

[2] Li J, Qiu M, Ming Z, Quan G, Qin X, Gu Z (2012) Online optimization for scheduling preemptable tasks on IaaS cloud system. J Parallel Distrib Comput 72:666–677

[3] Panda SK, Jana PK (2015) Efficient task scheduling algorithms for heterogeneous multi-cloud environment. J Supercomput 71(4):1505–1533

[4] Christo Ananth, A.Nasrin Banu, M.Manju, S.Nilofer, S.Mageshwari, A.Peratchi Selvi, "Efficient Energy Management Routing in WSN", International Journal of Advanced Research in Management, Architecture, Technology and Engineering (IJARMATE), Volume 1, Issue 1, August 2015,pp:16-19

[5] Son S, Jung G, Jun SC (2013) An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider. J Supercomput 64(2):606–637

[6] Cloud Service Level Agreement Standardisation Guidelines. http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?action=display&doc_id=6138. Accessed on 4 June 2015

[7] Liu L, Mei H, Xie B (2016) Towards a multi-QoS human-centric cloud computing load balance resource allocation method. J Supercomput 72(7):2488–2501

[8] Son S, Kang D, Huh SP, Kim W, Choi W (2016) Adaptive trade-off strategy for bargaining-based multi-objective SLA establishment under varying cloud workload. J Supercomput 72(4):1597–1622

[9] Ranaldo N, Zimeo E (2016) Capacity-driven utility model for service level agreement negotiation of cloud services. Future Gen Comput Syst 55:186–199

[10] Baset SA (2012) Cloud SLAs: present and future. ACM SIGOPS Oper Syst Rev 46:57–66

[11] Emeakaroha VC, Netto MAS, Calheiros RN, Brandic I, Buyya R, Rose CAFD (2012) Towards auto- nomic detection of SLA violations in cloud infrastructures. Future Gen Comput Syst 28:1017–1029

[12] Maurer M, Emeakaroha VC, Brandic I, Altmann J (2012) Cost-benefit analysis of an SLA mapping approach for defining standardized cloud computing goods. Future Gen Comput Syst 28:39–47