

Performance evaluation of two stage scheduling algorithm in cloud computing

ABSTRACT

Cloud computing is an evolutionary approach that completely changes how computing services are produced, priced and delivered. Cloud computing allows to access services that reside in a distant datacenter, other than local computers. Resource provisioning is the key process in cloud computing. A virtual machine (VM) is a software implementation of a machine that executes programs like a physical machine. Two stage scheduling is a novel approach in cloud computing. In this case a job may request two virtual machines in sequence to complete their needs. This paper presents a novel two stage scheduling algorithm to schedule the given requests in cloud environment by extending Johnson's Scheduling algorithm. Simulation results show that this algorithm reduces average waiting time and total elapsed time when compare to other scheduling algorithms.

Keywords: Cloud Computing, Two Stage Scheduling

1. INTRODUCTION

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. The cloud computing environment refers to the hardware and systems software in the datacenters that provide computing resources as services. Cloud computing provides on-demand computing resources for various applications. This paradigm provides a cost effective solution for running business applications by using Virtualization technologies, scalable distributed technologies and data management techniques with on demand Pay-As-You-Go (PAYG) pricing model. Cloud computing has three deployment models Public Cloud, Private Cloud and Hybrid Cloud.

Cloud computing has three service models Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

Scheduling is the process of deciding how to commit resources between a variety of possible tasks. In cloud computing, a typical datacenter consists of computing machines connected by high speed network links. This environment is well suited for the computation of large, diverse group of tasks. Tasks belonging to different users are no longer distinguished one from another. Scheduling algorithms in Cloud computing comes under the class of NP-complete. Heuristic is often applied as a suboptimal algorithm to obtain relatively good solutions. Scheduling heuristics can be static or dynamic. Static heuristic is suitable for the situation where the complete set of tasks is known prior to execution, while dynamic heuristic performs the scheduling when a task arrives.

A virtual machine (VM) is a software implemented abstraction of the underlying hardware, which is presented to the application layer of the system. Virtual machines may be based on specifications of a hypothetical computer or emulate the computer architecture and functions of a real world computer. Virtual Machines are considered an ideal vehicle for resource

provisioning in Cloud Computing. In VM model, hardware, software, and availability can be provisioned for job requests. Additionally, VMs' capability in getting suspended, resumed, or migrated without major utilization loss has proved to be useful in resource management.

2. RELATED WORK

2.1 Task Scheduling

Samuel [1] presented an algorithm for job sequencing with resource constraints to achieve minimum total elapsed time using partial enumeration. This algorithm uses partial enumeration of mixed integer program. This algorithm employs a maximum flow computation as a check for feasibility with respect to available resources. Richard.J [2] discussed a classic three machine scheduling model with optimal permutation of n items by considering total elapsed time as objective function. This scheduling algorithm uses Integer Linear Programming approach to solve this problem. Edward Ignall [3] proposed flow shop scheduling using branch and bound technique for n jobs and three machines by considering mean completion time as objective function. This scheduling algorithm uses lower bound for makespan of all nodes emanating from a given node.

2.2 Workflow Scheduling

Jia Yu and Rajkumar Buyya [4] presented a budget constrained scheduling of workflow applications on utility grids using genetic algorithms and proposed a budget constraint based scheduling using genetic algorithm, which minimizes execution time while meeting a specified budget for delivering results. Jia Yu, Rajkumar Buyya and Chen Khong Tham [5] further enhanced cost-based scheduling of scientific workflow Applications on Utility Grids with a novel algorithm which minimizes execution cost while meeting the deadline for delivering results. Hamid and Radu [6] discussed a pricing model and truthfull mechanism for task scheduling by considering monetary cost and completion time as objective functions.

2.3 Virtual Machine Scheduling

Quyet Thang Nguyen [7] explained Virtual Machine allocation using mixed linear programming method in cloud computing for minimizing total execution time on each machine where each job needs to use a number of virtual machines during the given fixed time. Xiao cheng Liu [8] described a Priority-based Consolidation of parallel workloads in the Cloud. Zhen Xiao [9] discussed a Dynamic Resource Allocation using Virtual Machines for Cloud Computing Environment. In this approach the concept of skewness was introduced to measure the unevenness in the multi-dimensional resource allocation of a server. E. Ilavarasan and P. Thambidurai [10] identified a Scheduling of an application modeled by Directed Acyclic Graph (DAG). and explained a scheduling algorithm based on list scheduling, namely, low complexity Performance Effective Task Scheduling (PETS) algorithm for heterogeneous computing systems with complexity $O(e)(p + \log v)$, which provides effective results for applications represented by DAGs.

Scheduling policies which consider resource prices and also users budget and deadlines are called market oriented scheduling policies. Mohsen Amini, Rajkumar Buyya [11] discussed market oriented scheduling policies which aims to satisfy application deadlines. Navendu Jain [12] constructed a resource allocation algorithm which provides small approximation factor (≤ 2.0) as the number of servers increasing. Nguyen and Nam [13] discussed power aware allocation for user request in virtual computing by proposing host selection policies like MAP(Minimum of Active Ports)

3 MODEL

Consider the resource allocation in cloud computing for each job request in physical machines. Each request requires a series of different virtual machines to complete its task. Virtual machine allocation for each job request in proposed model has been described in the Fig. 1.

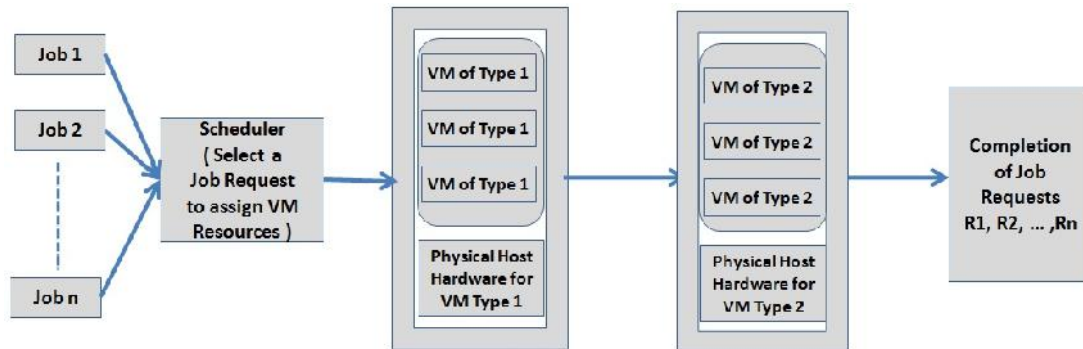


Fig. 1 Model of Resource Allocation

Let $r_1, r_2, \dots, r_n$ be the set of requests in the cloud computing at an instance of time. Each request r_i requires a VM of type1 for a time of T_{i1} , VM of type2 for an amount of T_{i2} . Each request has to process its work on VM_1 and later on VM_2 . Total elapsed time and average waiting time are performance metrics for scheduling algorithm. An efficient scheduling algorithm is required to find a scheduling sequence among the n job requests on two virtual machines types to reduce Total Elapsed Time, Average Waiting Time and Average Turnaround Time. Idle time of Virtual Machines is also one of the performance evaluations.

Terminology and notations used in this paper is given below.

n : Number of jobs.

r_i : i^{th} Job Request.

T_{i1} : The time required on Virtual Machine of type 1 (VM_1) for request r_i .

T_{i2} : The time required on Virtual Machine of type 2 (VM_2) for request r_i .

p : Maximum no of virtual Machine can be created for a Virtual Machine Type.

s_i : Starting time of job request r_i in Virtual Machine of type1.

c_i : Completion time of job request r_i in Virtual Machine of type2.

4 TWO STAGE SCHEDULING IN CLOUD COMPUTING

S M Johnson [14] proposed a scheduling algorithm for two stage problem. A typical two stage problem for the given n job request to process on Resource type-1 and then on

Resource type-2, and assumed that only one instance of each Resource type are available. Wing [15] described an efficient implementation Johnson algorithm. Johnson scheduling algorithm to schedule n job request on two resource types can be expressed as follows.

Alg. Johnson($T[1..n][1..2]$)

```

1. begin
2.    $i=1, j=n,$ 
3.   Solution_seq = Empty;
4.   for each job request  $r_i$  with shortest time period among all unprocessed requests do
5.     if  $T[i][1] < T[i][2]$  then
6.       add the job request  $r_i$  to the Solution_seq at index  $i$ 
7.        $i = i + 1;$ 
8.     else
9.       add the job request  $r_i$  to the Solution_seq at index  $j$ 
10.       $j = j - 1;$ 
11.    end if;
12.  end;
13. end;
```

Johnson Scheduling algorithm accepts n number of job requests, and each job request r_i to perform it operations on Resource type-1 for an amount of T_{i1} units time initially, later on Resource type-2 for an amount of T_{i2} units time. Johnson's algorithm recursively identifies a request r_i with shortest time quantum T_{i1} or T_{i2} among unprocessed requests. If the shortest time is T_{i1} of the request r_i Resource type-1, then the request r_i is added to the solution vector from front, otherwise if shortest time is T_{i2} , then the request is added in solution from the end. Consider n job request on the two virtual machine types, Table 1 shows each request r_i process times on the given two virtual machine types.

Table 1. Requests and Time required to process on each type of VM.

<i>Request (r_i)</i>	T_{i1}	T_{i2}
1	t_{11}	t_{12}
2	t_{21}	t_{22}
3	t_{31}	t_{32}
4	t_{41}	t_{42}
...	...	...
...	...	...
n	t_{n1}	t_{n2}

164
165
166
167
168

Alg. Two-stage Scheduling($T[1..n][1..m], \text{No_of_instances } p$)

```

1. Begin
2.   Optimal_Seq = Johnson( $T$ );
3.   for  $i=1$  to  $p$  do
```

172

```

173     4.      begin
174     5.      tval = i % p;
175     6.      append Optimal_Seq[i] to sub sequence S[tval];
176     7.      end;
177     8.      for i=1 to p do
178     9.      begin
179     10.     Calculate MakeSpan for each subsequence S[i];
180     11.     end;
181     12. end;

```

182
183

184 Waiting Time of a request is the time elapsed between the arrival time of request and when
185 the request starts its work on VM of type1, plus the time elapsed between when it completes
186 its work on VM of type1 and starts its work on VM of type2. The Total Elapsed Time of the
187 entire schedule is the time when all requests completed their work on both virtual machines
188 of type1 and type2 respectively. Total Elapsed time of this schedule is the c_k , where k is the
189 last request in the schedule given by the scheduling algorithm. The performance metrics can
190 be computed by the following computations for a given scheduling sequence.

191
192

193 Average Waiting Time(AWT), Average Turnaround Time(ATT), Total Elapsed Time (TET)
194 of all requests can be computed as follows

195

$$196 \text{ AWT} = \sum_{i=1}^n s_i((c_i - t_{i2}) - (s_i + t_{i1})) / n$$

197

$$198 \text{ ATT} = (\sum_{i=1}^n c_i) / n$$

199

200 TET = c_k , where k is the last request in schedule.

201

202 In Two-stage Scheduling algorithm consider a case that multiple VM (p) can be created for
203 each virtual machine type on respective physical machines. Scheduling sequence can be
204 divided in to p ($p \geq 2$) sub scheduling sequences. Each sub scheduling sequence can be
205 processed with one instance of each virtual machine type. A scheduling sequence
206 $S = \{r_1, r_2, r_3, \dots, r_n\}$ can be divided into p sub sequences as $S_1 = \{r_i, \text{ for all } i \text{ where } i \bmod p = 1\}$,
207 $S_2 = \{r_i, \text{ for all } i \text{ where } i \bmod p = 2\}$, ..., $S_p = \{r_i, \text{ for all } i \text{ where } i \bmod p = 0\}$ Each sub
208 sequence has to schedule on a single VM instance among available multiple VM of each
209 virtual machine type. For example the scheduling sequence which consists of 9 request
210 $\{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9\}$ can be divided into 3 sub sequences as follows. $S_1 = \{r_1, r_4, r_7\}$, $S_2 =$
211 $\{r_2, r_5, r_8\}$, $S_3 = \{r_3, r_6, r_9\}$

212

213 5 EXPERIMENTAL EVALUATION

214

215 A custom simulation has been used to analyze the FCFS Scheduling and Two-stage
216 Scheduling Algorithm with p instances for each virtual machine type. Gaussian distribution is
217 used to generate process times for virtual machine for each request. The following instance
218 is considered where eight job requests on two virtual machine types such that only two($p=2$)
219 instances of virtual machine can be created for each virtual machine type. Each request r_i
220 and required process times on each type of virtual machines are shown in Table 2.

221

Table 2. Requests and Time required to process on each type of VM.

222

Request (r_i)	Ti ₁	Ti ₂
----------------------	-----------------	-----------------

1	25	65
2	13	37
3	82	50
4	7	65
5	10	17
6	87	99
7	33	83
8	57	54

223
224
225
226
227
228
229
230
231
232
233

The given requests are divided into two sequences using FCFS's scheduling, and processed each sub sequence with an instance of each virtual machine type. The above sequence of eight requests with two virtual machine types having two instances for each type, can be divided into two sub sequences $S_1 = \{1\ 2\ 3\ 4\}$ and $S_2 = \{5\ 6\ 7\ 8\}$.

Initially, one can process r_0 on first instance of virtual machine type-1, r_4 on second instance of virtual machine type-1. Later each request process its operation on an instance of virtual machine type-2. Table 3 and Table 4 show start time, turnaround time and waiting time of the given request.

Table 3. FCFS Scheduling for sub sequence S_1				Table 4. FCFS Scheduling for sub sequence S_2			
Request (r_i)	Start time of r_i	Turn Around time of r_i	Waiting Time of r_i	Request (r_i)	Start time of r_i	Turn Around time of r_i	Waiting Time of r_i
1	0	90	0	5	0	27	0
2	25	127	77	6	10	196	10
3	38	177	45	7	97	279	163
4	120	242	170	8	130	324	222

234 Table 5 shows the Total elapsed time, Average turnaround time and Average waiting time of
235 FCFS Scheduling with two sub sequences. These performance evaluation metrics show the
236 advantages of scheduling the request with multiple instances of each virtual machine type.

237
238
239

Table 5. Performance Evaluation Metrics in FCFS Scheduling with two VM Instances for each VM type

	S_1	S_2	Avg(S_1, S_2)
Total Elapsed Time	266	324	295
Average Turnaround Time	153	206	179.5
Average Waiting Time	52	98	75

240
241
242
243
244
245
246

Two-stage Scheduling for the given problem will results in a scheduling sequence $\{4\ 5\ 2\ 1\ 7\ 6\ 3\ 8\}$, such as the scheduling sequence is split into two sub sequences $S_1 = \{4\ 2\ 7\ 3\}$ and $S_2 = \{5\ 1\ 6\ 8\}$. Table 6, shows the start time, turnaround time and waiting time of sub sequence S_1 , and Table 7, shows the same metrics for sub sequence S_2 .

Table 6. Two-stage Scheduling for sub sequence S_1

<i>Request (r_i)</i>	<i>Start time of r_i</i>	<i>Turn Around time of r_i</i>	<i>Waiting Time of r_i</i>
4	0	72	0
2	7	109	59
7	20	192	76
3	53	242	110

Table 7. Two-stage Scheduling for sub sequence S_2

<i>Request (r_i)</i>	<i>Start time of r_i</i>	<i>Turn Around time of r_i</i>	<i>Waiting Time of r_i</i>
5	0	27	0
1	10	100	10
6	35	221	35
8	122	266	164

Table 8, shows the Total elapsed time, Average turnaround time and Average waiting time of Two-stage Scheduling with two sub sequences. These performance evaluation metrics have minimum time compared to FCFS Scheduling with multiple instances.

Table 8. Performance Evaluation Metrics in Two-stage Scheduling with two VM Instances for each VM type

	S_1	S_2	$Avg(S_1, S_2)$
Total Elapsed Time	242	266	254
Average Turnaround Time	153	153	153
Average Waiting Time	61	52	56.5

Table-9 shows the three evaluation metrics of FCFS Scheduling, and Two-stage Scheduling. Total elapsed time in Extended-Scheduling with multiple VM instances is 218 which is 246.5 in FCFS Scheduling with multiple VM instances. Average waiting time of all requests is 52 units of time in Two-stage Scheduling with multiple VM instances and 94.5 in FCFS Scheduling with multiple VM instances.

Table 9. Comparison of Performance Evaluation Metrics in FCS and Two-stage Scheduling strategies

	FCFS Scheduling with multiple VM instance ($p=2$)	Two-stage Scheduling with multiple VM instance ($p=2$)
Total Elapsed Time	295	254
Average Turnaround Time	179.5	153
Average Waiting Time	75	56.5

figure 3 depicts the comparison of FCFS and Two stage scheduling algorithm with multiple VM instances with respect to the given three valuation metrics.

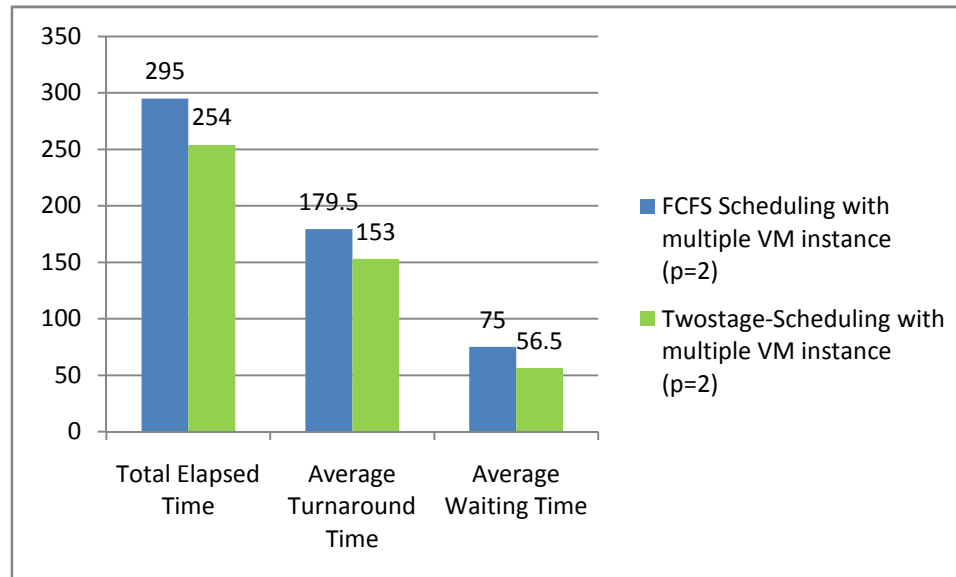


Fig. 3. Comparison of Performance Evaluation Metrics in different scheduling strategies.

6. CONCLUSION

Scheduling the given n job requests in cloud computing on two types of virtual machines using Two-stage Scheduling algorithm gives better performance evaluation metrics. The total elapsed time and Average waiting time are reasonably decreased in Two-stage Scheduling Algorithm when compared to FCFS Scheduling algorithm. Experimental results show that Two-stage Scheduling algorithm has better scheduling parameters compared to FCFS Scheduling.

REFERENCES

1. Samuel Gorenstein, An Algorithm for Project (Job) Sequencing with Resource Constraints, Journal of Operations Research, Vol. 20, No. 4, pp. 835-850, 1970.
2. Richard J. Giglio and Harvey M. Wagner, Approximate solutions to the three machine scheduling problem, Journal of Operations Research, Vol. 12, No. 2, pp. 305-324, 1964.
3. Edward Ignall and Linus Schrage, Application of the branch and bound technique to some flow-shop scheduling problems, Cornell University 1964.
4. Jia Yu and Rajkumar Buyya, A Budget Constrained Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms.
5. Jia Yu, Rajkumar Buyya and Chen Khong Tham, Cost-based Scheduling of Scientific Workflow Applications on Utility Grids.
6. Hamid Mi Fard, Radu and Thomas, A Truthful Dynamic Workflow Scheduling Mechanism for Commercial Multi-Cloud Environments, IEEE T Transaction on Parallel and Distributed Systems, 2012
7. Quyet Thang Nguyen, Quang-hung, Nguyen Huynh Tuong, Van Hoai Tran, Nam Thoai, Virtual Machine Allocation in Cloud Computing for Minimizing Total Execution Time on Each Machine, IEEE, 2013.

- 309 8. Xiaocheng Liu, Chen Wang, Priority-based Consolidation of Parallel Workloads in the
310 Cloud, IEEE TRANSACTIONS
- 311 9. Zhen Xiao, Weijia Song, and Qi Chen, Dynamic Resource Allocation using Virtual Machines
312 for Cloud Computing Environment, IEEE Transaction on Parallel and Distributed Systems,
313 2012.
- 314 10. E. Ilavarasan and P. Thambidurai, Low Complexity Performance Effective Task Scheduling
315 Algorithm for Heterogeneous Computing Environments
- 316 11. Mohsen Amini, Rajkumar Buyya Adapting market-oriented scheduling policies for cloud
317 computing, 2009.
- 318 12. Navendu Jain, Ishai Menache, Joseph Naor, and Jonathan Yaniv, A Truthful Mechanism for
319 Value-Based Scheduling in Cloud Computing, 2010.
- 320 13. Nguyen, Nam, Power aware allocation for user request in virtual computing
- 321 14. Johnson S M, Optimal 2 and 3 stage production schedules with set up time included, Naval
322 Research Logistics, Q1, Page:61-88, 1954.
- 323 15. Wing S, An Efficient Implementation of a Two Stage Production Scheduling Algorithm,
324 Journal of Operational Research Vol 40, No 11, pp 1049-1052, 1989.
325