IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Vol.-7, Issue-6, June 2019 E-ISSN: 2347-2693

A Survey of Backfilling Algorithms in Cloud Resource Allocation

V. Nisha[1*], S. Vimala[2]

[1,2]Department of Computer Science, Mother Teresa Women's University, Kodaikanal, India

Corresponding Author: nishavarnam@yahoo.co.in

DOI: https://doi.org/10.26438/ijcse/v7i6.390394 | Available online at: www.ijcseonline.org

Accepted: 14/Jun/2019, Published: 30/Jun/2019

***Abstract*—** Cloud computing is an information technology (IT) paradigm that enables ubiquitous access to shared pools of configurable system resources and higher-level services that has provisioned with minimal management effort over the Internet. The main enabling technology for cloud computing is Virtualization, which is essentially creating scalable system of multiple independent computing devices. With virtualization, idle computing resources have allocated to user more effectively. Allocation of idle computing resource is one of the major problem faced today in cloud computing. Adopting to right resource allocation algorithms can resolve the problem of resource allocation. Backfilling algorithms are better than the existing First Come First Serve algorithms (FCFS) used for resource allocation. In this paper, various 'Backfilling' algorithms are surveyed. Further analysis on the performance of each algorithm in terms of response time, throughput, waiting time, turn-around time, job migration between queues are measured.

***Keywords*—**Cloud Computing, Virtualization, First Come First Serve, Backfilling, Job Migration

I. INTRODUCTION

In cloud computing, virtualization separates a physical computing device into one or more "virtual" devices, each of which can be used and managed for performing task. FCFS algorithm is the default algorithm used by computing devices (parallel and super computers) for allocating resources.

FCFS uses parallel processing where multiple requests are executed simultaneously. The computing resources are allocated to the incoming request based on the order of arrival rather than the resource required which results in poor allocation and utilization of resource leading to fragmentation. Fragmentation is the major drawback in FCFS where request are kept in queue due to insufficient computing resources and the unallocated resource are kept idle.

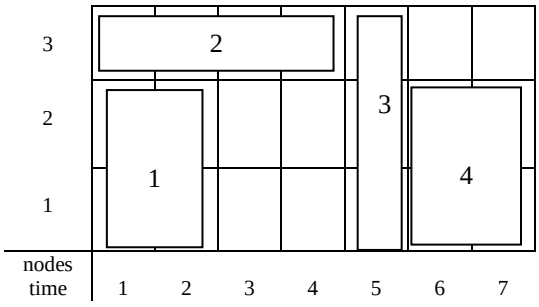

Fig 1. First Come First Serve Resource Allocation

In the above figure, there are 4 incoming request and 3 computing devices available. FCFS allocates the available computing devices in the order of arrival. Incoming Request #1 & #2 are served and Request #3 is not served due to the insufficient computing devices. In the above, Fragmentation is created because the computing devices #1 & #2 are kept idle (no immediate request matching with available computing devices) and Incoming Request #3 is kept in queue (inadequate computing device).

Backfilling algorithm overcomes the drawback of FCFS by allocating the available computing devices to the incoming request based on required number of computing devices irrespective of their incoming order. By doing so, Backfilling manages to allocate the computing devices better and reduces the wait time of incoming request.

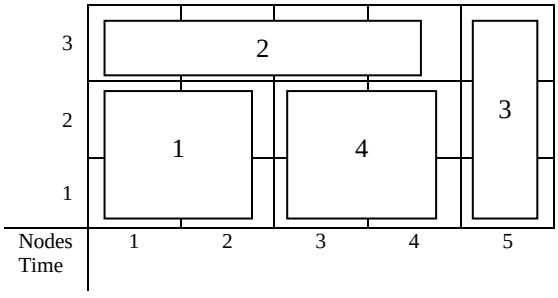

Fig 2. Backfilling Resource Allocation

In figure 2, there are 4 incoming request and 3 computing devices available. Backfilling allocates the available computing devices based on the required number of computing device. Incoming Request #1 & #2 are served, Computing device # 1 & #2 become available after serving Request #1. Backfilling algorithm allocates the available computing devices to next incoming with matching required computing device. So the incoming request #4 is picked and incoming request #3 is kept in queue. Backfilling manages to allocate computing devices to incoming request and avoids fragmentation.

By allocating the available CPU's to the incoming job irrespective of their incoming order, backfilling algorithm improves the resource utilization, waiting period and response time of the system.

There are various backfilling algorithms such as

- Extensible Argonne Scheduling System (EASY)
- Gang Scheduling
- Conservative Migration supported Backfilling (CMBF)
- Aggressive Migration Supported Backfilling (AMBF)
- Aggressive Migration and Consolidation supported Backfilling (AMCBF)
- Conservative Migration and Consolidation supported Backfilling (CMCBF).

In this paper, section 2 discusses related work & experiment results and we explain results and discussion in section 3. At last in the in section 4 is conclusion.

II. RELATED WORK & EXPERIMENT RESULTS

For implementation, Consider six processors and ten jobs are waiting in the queue and it is mentioned in following format, Job name (number of processors, execution time).

Jobs are J1(1,10) J2(2,5) J3(2,10) J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15).

A. First-Come-First-Serve(FCFS)

First Come First Serve is the fundamental parallel scheduling algorithm where request are executed in the order of arrival. The immediate available computing device is allocated to the incoming request, in case of inadequate computing device, the incoming requests are kept on hold irrespective of their priority. The incoming request has to be in queue until the availability of next matching computing devices. The result shows increase of wait time of the request and accumulation of un-served request [1].

The implementation result for First-Come-First-Serve (FCFS) is shown in Table 1.

Table 1. First Come First Serve

PROCESSOR → / TIME ↓	P1	P2	P3	P4	P5	QUEUE
T=0	J1	J2	J2	J3	J3	J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=5	J1			J3	J3	J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=10						
	J4	J4	J4	J5	J6	J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=20				J5	J6	
	J7	J7		J5	J6	J8(5,5) J9(4,5) J10(1,15)
T=25	J7	J7		J5		J8(5,5) J9(4,5) J10(1,15)
T=30				J5		J8(5,5) J9(4,5) J10(1,15)
T=-35						
	J8	J8	J8	J8	J8	J9(4,5) J10(1,15)
T=40						
	J9	J9	J9	J9	J9	J10(1,15)
T=45						
	J10					

FCFS algorithm is easy to implement and has an advantage of no migration of request and disadvantage is many times CPU are idle due to the lack of availability of processor. Request with least required resource in queue are not served on time.

B. EASY

Extensible Argonne Scheduling System (EASY) is the default backfilling algorithm, which allows the shortest request to utilize the available computing device when the request at the head of waiting queue does not have enough number of computing devices to execute [2].

The implementation result for Extensible Argonne Scheduling System is shown in Table 2.

Table 2. Extensible Argonne Scheduling System (EASY)

PROCESSOR → / TIME ↓	P1	P2	P3	P4	P5	QUEUE
T=0	J1	J2	J2	J3	J3	J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=5	J1			J3	J3	
	J1	J5	J6	J3	J3	J4(3,10) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=10		J5	J6			
	J4	J5	J6	J4	J4	J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=20		J5				
	J7	J5	J7	J10		J8(5,5) J9(4,5)
T=30				J10		
	J9	J9	J9	J10	J9	J8(5,5)
T=35						
	J8	J8	J8	J8	J8	

EASY is better than FCFS as there is no fragmentation but holds a drawback of request in queue suffer from unbounded delay.

C. Gang-scheduling

Incoming request are split in to threads of same size. These threads are executed based on the number of processor required for the execution of the request and executed simultaneously across all the processors. When the time slice is over, the threads are pre-empted and sent to waiting queue. By the pre-emption mechanism, the shortest jobs are addressed quicker since the long waiting time is avoided [1] [3].

The implementation result for Gang scheduling is shown in Table 3

Time Slice: 2

Table 3. Gang Scheduling

PROCESSOR / TIME	P1	P2	P3	P4	P5	QUEUE
T=0	J1	J1	J1	J1	J1	J2(2,5) J3(2,10) J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=2	J2	J2	J2	J2	J2	J3(2,10) J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=4	J3	J3	J3	J3	J3	J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=6	J4	J4	J4	J4	J4	J3(2,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=8	J5	J5	J5	J5	J5	J3(2,10) J4(3,10) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=10	J6	J6	J6	J6	J6	J3(2,10) J4(3,10) J5(1,25) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=12	J7	J7	J7	J7	J7	J3(2,10) J4(3,10) J5(1,25) J6(1,15) J8(5,5) J9(4,5) J10(1,15)
T=14	J8	J8	J8	J8	J8	J3(2,10) J4(3,10) J5(1,25) J6(1,15) J7(2,10) J9(4,5) J10(1,15)
T=16	J9	J9	J9	J9	J9	J3(2,10) J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5) J10(1,15)
T=18	J10	J10	J10	J10	J10	J3(2,10) J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5)) J9(4,5)
T=20	J3	J3	J3	J3	J3	J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5)) J9(4,5) J10(1,15)
T=22	J4	J4	J4	J4	J4	J5(1,25) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=24	J5	J5	J5	J5	J5	J4(3,10) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=26	J6	J6	J6	J7	J7	J4(3,10) J5(1,25) J8(5,5) J9(4,5) J10(1,15)
T=28	J8	J8	J8	J8	J8	J4(3,10) J5(1,25) J7(2,10) J9(4,5) J10(1,15)
T=30	J9	J9	J9	J9	J9	J4(3,10) J5(1,25) J7(2,10) J8(5,5) J10(1,15)
T=32	J10	J10	J10	J4	J4	J5(1,25) J7(2,10) J8(5,5)
T=34	J5	J5	J5	J7	J7	J4(3,10) J8(5,5)
T=36	J8	J8	J8	J4	J4	J7(2,10)
T=38	J7	J4				

Gang scheduling algorithm has advantages over EASY, Computing devices are maximum utilized, No idle time for computing device, quick response time. At the same time, it has a disadvantage of migration where the requests are shuffled in queue.

D. Aggressive Migration Supported Backfilling (AMBF)

In AMBF, when a job is done and the processor is available to pick up the next job from the queue, the pre-emption is done. Pre-emption is done by the first job in the queue. No other job in the queue is authorized to do the pre-emption.

The implementation result for AMBF is shown in Table 4.

Table 4. Aggressive Migration Supported Backfilling

PROCESSOR / TIME	P1	P2	P3	P4	P5	QUEUE
T=0	J1	J2	J2	J3	J3	J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=5	J1			J3	J3	
	J1	J5	J6	J3	J3	J4(3,10) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=10		J5	J6			
	J4	J5	J6	J4	J4	J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=15	J4	J5	J6	J4	J4	J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=20		J5				
	J7	J5	J7	J10		J8(5,5) J9(4,5)
T=25	J7	J5	J7	J10		J8(5,5) J9(4,5)
T=30				J10		
	J8	J8	J8	J8	J8	J9(4,10) J10(1,15)
T=35						
	J9	J9	J9	J9	10	

AMBF overcomes the drawback of gang scheduling, it limits the number of request migration in queue since no time slice is being used. Computing device might be idle sometime which is the drawback of AMBF [4].

E. Conservative Migration supported Back Filling (CMBF)

In AMBF, as the first job in the queue is authorized to do pre-emption resulting in poor utilization of resource. To overcome the same, CMBF allocates the available resource to the next job in the queue with suitable number of required resource. No restriction to the jobs in queue for pre-emption but preference is given based on the arrival order of the job [4].

The implementation result for CMBF is shown in Table 5.

Table 5. Conservative Migration supported Backfilling

PROCESSOR / TIME	P1	P2	P3	P4	P5	QUEUE
T=0	J1	J2	J2	J3	J3	J4(3,10) J5(1,25) J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=5	J1			J3	J3	
	J1	J5	J6	J3	J3	J4(3,10) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=10		J5	J6			
	J4	J5	J6	J4	J4	J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=15	J4	J5	J6	J4	J4	J7(2,10) J8(5,5) J9(4,5) J10(1,15)
T=20		J5				
	J7	J5	J7	J10		J8(5,5) J9(4,5)
T=25	J7	J5	J7	J10		J8(5,5) J9(4,5)
T=30				J10		
	J9	J9	J9	J10	J9	J8(5,5)
T=35						
	J8	J8	J8	J8	J8	

CMBF is better than FCFS, EASY and GANG, it has lower request migration compared to them. But when compared with AMBF, CMBF have more request migration in queues.

F. Conservative Migration and Consolidation supported Backfilling (CMCBF)

In CMCBF [5], each node has two virtual machines, one is allocated for fore-ground and the other is allocated to background of the environment. CMCBF treats both the foreground and background jobs simultaneously to achieve better node utilisation. When a fore-ground VM is idle or available, the job in background VM is migrated to foreground VM. When background job departs, the scheduler scans the queue based on the arrival time of the job and places the matching job in to the available background VM. All the jobs in queue are authorised to do pre-emption [5].
The implementation result for CMCBF is shown in Table 6.

Table 6. Conservative Migration and Consolidation supported Backfilling (CMCBF)

VM	P1	P2	P3	P4	P5	TIME	QUEUE
FG	J1	J2	J2	J3	J3	T=0	J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
BG	**N**	J4	J4	J4	J5		
VM	**P1**	**P2**	**P3**	**P4**	**P5**	**TIME**	**QUEUE**
FG	J1			J3	J3	T=5	J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
BG	N	J4	J4	J4	J5		
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J1	J5		J3	J3		J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15) **J4(3,5)**
BG	**N**	**N**					
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J1	J5	J6	J3	J3		J7(2,10) J8(5,5) J9(4,5) J10(1,15) **J4(3,5)**
BG	**N**	**N**	**N**				
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J1	J5	J6	J3	J3		J8(5,5) J9(4,5) J10(1,15) **J4(3,5)**
BG	**N**	**N**	**N**	J7	J7		
VM	**P1**	**P2**	**P3**	**P4**	**P5**	**TIME**	**QUEUE**
FG		J5	J6			T=10	J8(5,5) J9(4,5) J10(1,15) **J4(3,5)**
BG		**N**	**N**	J7	J7		
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J4	J5	J6	J4	J4		J8(5,5) J9(4,5) J10(1,15)
BG		**N**	**N**	J7	J7		
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J4	J5	J6	J4	J4		J8(5,5) J9(4,5)
BG	J10	**N**	**N**	J7	J7		
VM	**P1**	**P2**	**P3**	**P4**	**P5**	**TIME**	**QUEUE**
FG		J5	J6			T=15	J8(5,5) J9(4,5)
BG	J10	**N**	**N**				
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J10	J5	J6				J8(5,5) J9(4,5)
BG	**N**	**N**	**N**				
VM	**P1**	**P2**	**P3**	**P4**	**P5**	**TIME**	**QUEUE**
FG	J10	J5				T=20	J8(5,5) J9(4,5)
BG	**N**	**N**					
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J9	J5	J9	J9	J9		J8(5,5) **J10(1,5)**
BG		N					
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J9	J5	J9	J9	J9		J8(5,5)
BG	J10	**N**					
VM	**P1**	**P2**	**P3**	**P4**	**P5**	**TIME**	**QUEUE**
FG						T=25	J8(5,5)
BG							
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J8	J8	J8	J8	J8		
BG							

CMCBF is better than AMBF, CMBF, FCFS, EASY and GANG since it has less waiting time and response time. But migration cost is more when compared with AMCBF due to movement of request from background VM to foreground VM.

G. Aggressive Migration and Consolidation supported Backfilling (AMCBF)

AMCBF is also similar to CMCBF, it has two virtual machines and jobs are run in foreground and background VM's simultaneously. The Pre-emption in AMCBF is done for the first job in the queue which makes it different from CMCBF [5].
The implementation result for AMCBF is shown in Table 7.

Table 7. Aggressive Migration and Consolidation supported Backfilling (AMCBF)

VM	P1	P2	P3	P4	P5	TIME	QUEUE
FG	J1	J2	J2	J3	J3	T=0	J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
BG	**N**	J4	J4	J4	J5		
VM	**P1**	**P2**	**P3**	**P4**	**P5**	**TIME**	**QUEUE**
FG	J1			J3	J3	T=5	J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15)
BG	N	J4	J4	J4	J5		
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J1	J5		J3	J3		J6(1,15) J7(2,10) J8(5,5) J9(4,5) J10(1,15) **J4(3,5)**
BG	**N**	**N**					
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J1	J5	J6	J3	J3		J7(2,10) J8(5,5) J9(4,5) J10(1,15) **J4(3,5)**
BG	**N**	**N**	**N**				
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J1	J5	J6	J3	J3		J8(5,5) J9(4,5) J10(1,15) **J4(3,5)**
BG	**N**	**N**	**N**	J7	J7		
VM	**P1**	**P2**	**P3**	**P4**	**P5**	**TIME**	**QUEUE**
FG		J5	J6			T=10	J8(5,5) J9(4,5) J10(1,15) **J4(3,5)**
BG		**N**	**N**	J7	J7		
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J4	J5	J6	J4	J4		J8(5,5) J9(4,5) J10(1,15)
BG		**N**	**N**	J7	J7		
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J4	J5	J6	J4	J4		J8(5,5) J9(4,5)
BG	J10	**N**	**N**	J7	J7		
VM	**P1**	**P2**	**P3**	**P4**	**P5**	**TIME**	**QUEUE**
FG		J5	J6			T=15	J8(5,5) J9(4,5)
BG	J10	**N**	**N**				
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J10	J5	J6				J8(5,5) J9(4,5)
BG	**N**	**N**	**N**				
VM	**P1**	**P2**	**P3**	**P4**	**P5**	**TIME**	**QUEUE**
FG	J10	J5				T=20	J8(5,5) J9(4,5)
BG	**N**	**N**					
VM	**P1**	**P2**	**P3**	**P4**	**P5**	**TIME**	**QUEUE**
FG						T=25	J8(5,5) J9(4,5)
BG							
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J8	J8	J8	J8	J8		J9(4,5)
BG							
VM	**P1**	**P2**	**P3**	**P4**	**P5**		**QUEUE**
FG	J8	J8	J8	J8	J8		
BG	J9	J9	J9	J9			

AMCBF is better than AMBF, CMBF, FCFS, EASY, GANG and CMCBF as response time, waiting time, turn-around time and migration cost are less. But computing devices may remain idle when compared to the CMCBF.

III. RESULTS AND DISCUSSION

Execution results of the above inputs are depicted in Fig 3 in terms of average waiting time, average response time,

average turnaround time and average waiting time for FCFS, EASY, GANG, AMBF, CMBF, AMCBF, and CMCBF.

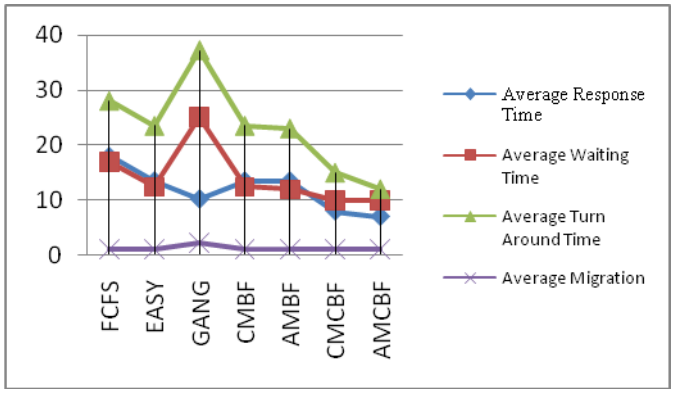

Fig 3: Execution graph for Backfilling algorithm

Average turn around time, Average waiting time and migration time are very high in Gang scheduling. Average response time is very high in FCFS algorithm. AMCBF algorithm got less average turn around time. CMCBF and AMCBF got more or less same result for Average response time, average waiting time and average migration time.

IV. CONCLUSION

Allocation of idle computing resource is one of the major problem faced today in cloud computing. Adopting to right resource allocation algorithms can resolve the problem of resource allocation. Backfilling algorithms are better than the existing First Come First Serve algorithms (FCFS) used for resource allocation. Various 'Backfilling' algorithms are surveyed. AMCBF produce good results when compared to other algorithms in the aspect of turn around time, response time, waiting time and migration time.

REFERENCES

[1] D. Tsafrir, Y. Etsion, and D. G. Feitelson, *"Backfilling Using Runtime Predictions Rather than User Estimates,"* School of Computer Science and Engineering, Hebrew University of Jerusalem, Tech. Rep. TR **2005-5, 2003.**

[2] Bhupesh Kumar Dewangan, Amit Agarwal,Venkatadri M, Ashutosh Pasricha*," Resource Scheduling in Cloud: A Comparative Study",* International Journal of Computer Sciences and Engineering, Vol.**-6**, Issue-**8**, pp. **168-173**,Aug **2018.**

[3] Rajnish Choubey et al., "*A Survey on Cloud Computing Security, Challenges and Threats*", International Journal of Computer Sciences and Engineering, Vol. **3**,No. **3** ,pp.**1227-1231**,Mar **2011.**

[4] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, *"Parallel job scheduling strategies for parallel processing"*, In Proceedings of the 10th International Conference on Job Scheduling Strategies for Parallel Processing, ser. JSSPP'04. Berlin, Heidelberg: Springer-Verlag, pp. **1–16** , **2005**.

[5] Dror G. Feitelson and Ahuva Mu'alem Weil., *"Utilization and predictability in scheduling the IBM SP2 with backfilling",* In Proceeding of the 12th International Parallel and Distributed Processing Symposium, pp. **542–546, 1998**.

[6] U. Schwiegelshohn and R. Yahyapour., "*Fairness in Parallel Job Scheduling", "Journal of Scheduling",* **3(5)** pp:**297-320**. John Wiley, **2000.**

[7] Shahabanath K K, Sreekesh Namboodiri T, *"K-Tier And Selective Backfilling Approach for Parallel Workload Scheduling in Cloud",* International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume **3** Issue **9**, September **2014.**

[8] Xiaocheng Liu, Chen Wang, Bing Bing Zhou, Junliang Chen,Ting Yang, and Albert Y. Zomaya, "*Priority-Based Consolidation Of Parallel Workloads In The Cloud*", IEEE Transactions On Parallel And Distributed Systems, Vol. **24**, No. **9**, September **2013.**

Authors Profile

Mrs. V.Nisha is currently pursuing Ph.D in Mother Teresa Women's University, Kodaikanal and currently working as Assistant Professor in Department of Computer Sciences, Dhanraj Baid Jain College, Chennai since 2011. She has published more than 5 research papers in reputed journals. Her main research work focuses on Cloud Computing.

Dr.S.Vimala is currently working as Associate Professor in Department of Computer Science, Mother Teresa Women's University,Kodaikanal since 1999. She has published more than 40 research papers in reputed international journals and she presented papers in more than 40 national and international conferences. Her main research work focuses on Cloud Computing and Digital Image Compression.

