CHAPTER 6

A NOVEL APPROACH FOR SCHEDULING PARALLEL JOBS BASED ON PRIORITY BASED CONSOLIDATION METHOD

6.1 INTRODUCTION

Cloud Technology standardizes and groups the Information Technology (IT) resources and automates many of the maintenance tasks done manually today. Its architecture model facilitates self-service, elastic consumption, and pay-as-you-go pricing. One of the advanced concepts of cloud computing is virtualization. It is a process of dividing the resources into various execution environments, using several techniques such as hardware and software partitioning approaches, machine simulation, time-sharing approaches, etc. With virtualization, it is possible to run multiple operating systems and multiple applications on the same server at the same time, by improving the utilization and flexibility of hardware. The main objective is to manage the load across the resources by converting the existing architecture into a scalable and effective architecture. Virtualization enables the business enterprises to

- Combine multiple servers without the need for separating the applications.
- Scale the infrastructure with respect to the increase in their needs and requirements.
- Increase resource availability through dynamic provisioning and relocation of the critical systems.
- Easy allocation of the memory and computing resource.
- Minimize the hardware cost and energy consumption through the dynamic sharing of the computing resources.

The main benefits of the virtualization are resource pooling, rapid and easy deployment of new servers, optimization of the physical resources and easy reconfiguration of the resources during the execution of applications or services. Using the virtualization concept, separation of applications or services on each Virtual Machine (VM) is achieved. Thus, the security of the applications is improved. There are two types of virtualization.
6.1.1 Full Virtualization

In the case of full virtualization [143], a complete installation of one virtual machine is done on another machine. It will result in a virtual machine having all the software present in the actual server. The remote data center delivers the service in a fully virtualized manner. Fig. 6.1 illustrates the full virtualization. Full virtualization is highly successful for the following purposes:

- Resource sharing among multiple users
- Separating the users from each other
- Emulating hardware on another virtual machine

![Fig. 6.1 Full virtualization](image)

6.1.2 Para Virtualization

In para-virtualization [143], the hardware allows multiple operating systems to run on the single virtual machine through the efficient utilization of the system resources such as memory and processor. Here, all the services are not fully available, rather the services are provided partially. Fig. 6.2 illustrates the para virtualization.

![Fig. 6.2 Para virtualization](image)
6.2 LOAD BALANCING

Load balancing [143], [144], [145] is an issue in the cloud data centers. Load balancing is the method of reassigning the total load to the individual resource of the cloud computing system. It make resource utilization effective and improve the response time of the job. The load balancing algorithm is dynamic in nature which does not consider the previous state. During the migration of job between the resources in the middle of the job execution, load balancing is a critical task. Hence, distribution of load between the resources should be balanced so as to complete the execution of job within a short time and improve the throughput of the whole system. A good load balancing approach should be scalable and minimize the computational overhead of the cloud computing system. Fig. 6.3 illustrates the load balancing in the cloud computing environment. Fig. 6.4 shows the types of the load balancing algorithms. There are two main types of load balancing approaches

- Static load balancing approaches
- Dynamic load balancing approaches

![Image](image-url)

Fig. 6.3 Load balancing in cloud computing environment
6.2.1 Static Load Balancing Approaches

Static load balancing approaches [143], [144], [145] divide the load equivalently between the resources. These approaches work properly only when the nodes have a low variation in the load. The static load balancing approaches are generally based on the information about the performance of the distributed cloud computing system. It also uses a priori knowledge of the applications or services and statistical information about the system. During the static load balancing, the operating performance of the computing resources is determined during the commencement of the execution process. Depending upon the performance, the data center assigns the workload to the resources. The resources calculate their own allocated workload and submit the workload rate to the data center. A task is always executed on the resource to which it is assigned. The main goal of the static load balancing method is to reduce the job execution time and minimize the communication delay between the resource and data center. As the load varies with respect to time, these algorithms are not suitable for cloud environments. The main disadvantage of the static load balancing approaches is the resource is selected before starting the execution of the process and it cannot be changed during the execution of the process.

6.2.2 Dynamic Load Balancing Approaches

Dynamic load balancing approaches [143], [144], [145] are advantageous over the static algorithms. But to gain this advantage, there is a need to consider the additional cost associated with collection and maintenance of the load information. In the dynamic load balancing approaches, the workload is distributed among the resources during the execution of the job. The data center assigns new processes to the resources based on the new information collected. The dynamic load balancing is further classified as distributed and non-distributed. In the distributed approach, the load is shared equally among all the resources. The communication between the resources to enable load balancing can be cooperative and non-cooperative. During the cooperative communication, the resources work side-by-side to achieve a common objective. During the non-cooperative communication, each resource works independently towards a local objective. Dynamic load balancing approaches usually generate more messages than the non-
distributed approaches, since each resource needs to communicate with the other resource. The main advantage of this distributed dynamic load balancing approach is there is no interruption in the total load balancing process, even during failure of one or more resources. In the non-distributed dynamic load balancing approach, load balancing is performed on either a single resource or a group of the resources. Fig. 6.4 illustrates the dynamic load balancing approach to avoid overloading on the heavily loaded resource.

![Fig. 6.4 Dynamic Load balancing approach](image)

### 6.3 JOB SCHEDULING

Parallel computing is the simultaneous use of multiple resources to solve a computational problem. Jobs are the sets of smaller components where the applications and services can be decomposed. Virtual machines are the emulation of a particular computer system and their operations are based on the computer architecture and functions of a real or hypothetical computer. And these virtual machines are treated as the processing units of cloud computing environments. The demand for the resources for the execution of jobs varies over time. Optimal scheduling of the jobs in the cloud enables proper utilization of the resources and reduces the execution time of the jobs while managing the load between the resources. Scheduling [146] of the jobs is of great consequence in the cloud computing environment. Schedules for cloud computing is accountable for the efficient scheduling of the jobs by fully utilizing the available resources. Two main categories of the scheduling are static scheduling and dynamic scheduling. In the static method, the scheduling decision should be computed before executing the job.
The dynamic method does not allow the prior knowledge about the time of termination of jobs. The general job scheduling problem mainly includes

- Selection of processing resource for every job
- Selection of job processing order/time for every resource

They are driven by different constraints such as Quality of Service (QoS) requirement of jobs, date/time dependencies between jobs and the processing limitation of the resources. The scheduling process involves discovery of the resource and filtering the resource. Then, a target resource is selected and a particular task is submitted to the target resource. Job scheduling is a most prominent task that greatly improves the performance of the cloud environment. The jobs are scheduled to the resources, based on the job size and resource capacity. Traditional job scheduling approaches are highly deterministic and fast, but it often get stuck on the local optima. The main aim of the job scheduling process is to improve the performance and QoS of the cloud environment while maintaining the fairness among the execution of the job and reducing the job execution time. Fig. 6.5 illustrates the stages of job scheduling.

![Fig. 6.5 Stages of job scheduling](image)
The scheduling process in the cloud computing is divided into three stages [147],

- **Resource discovering and monitoring**
  In this stage, all the resources located in the cloud environment are discovered by the data center broker. Also, it collects the status information corresponding to them.

- **Resource selection**
  This is a decision stage. Here, the target resource is selected based on the different characteristics of the job as well as the resources.

- **Job submission**
  After the selection of an optimal resource, the job is submitted to the selected resource.

The basic operation performed by the components in the scheduling process is briefly described below

**Data Center Broker**

The data center performs modeling of the core infrastructure-level services that are offered by the cloud providers. This data center encapsulates with the set of hosts, which are either heterogeneous or homogeneous. The data Center broker represents the broker acting on behalf of the user. The broker obtains the services from the data center and also provides the cloud services to the VMs. The data center broker is one type of broker with major responsibilities for mediating the negotiations between the Software as a Service (SaaS) and the cloud service providers.

**Cloud Information Services (CIS)**

It is one of the major components in the scheduling of resources. This is an entity that registers the data center entity and performs indexing and discovery of resource.

**VM**

A VM is an operating system or application environment that is installed on software that imitates dedicated hardware. The VM act as a physical computer, and also contains individual virtual Random Access Memory (RAM), Central Processing
Unit (CPU), Network Interface Card (NIC), and hardware disk. VMs can be easily moved, copied and reassigned between the host servers to optimize hardware resource utilization.

**Advantages of the job scheduling approach**

The main advantages of the job scheduling approaches are

- Increases throughput and overall working performance of the cloud computing system.
- Low turnaround time.
- Effective utilization of the resources.
- Prevent overloading of the resources.
- Execution of jobs within a minimum time period.
- Low migration time.
- Improves scalability of the cloud computing system.

6.4 **TYPES OF SCHEDULING**

The scheduling approaches are classified as [148]

- Static and dynamic scheduling
- Centralized, Hierarchical and Distributed scheduling
- Preemptive and Non-preemptive scheduling
- Online and batch mode scheduling

6.4.1 **Static and dynamic scheduling**

In static scheduling, the jobs are prescheduled before starting the execution process. The information and tasks are assumed to be available during scheduling of the jobs. It is also assumed that there is no job failure and the resources are available all the time. The static scheduling approach incurs less runtime overhead. The main drawback of this static scheduling approach is the need to estimate the execution time of each job. However, this is not possible in all cases. In dynamic scheduling approach, the jobs are dynamically available for scheduling. The dynamic scheduler should decide about the allocation of resources during the arrival of jobs. Hence, load balancing is to be considered in this concept. The main advantage of the dynamic scheduling approach over the static scheduling approach
is that the system need not possess the knowledge about the execution time of the jobs before it is executed.

**6.4.2 Centralized, Hierarchical and Distributed Scheduling**

The centralized and distributed scheduling approaches differ based on the control capacity of the resources and knowledge of the whole system. During centralized scheduling, there is more control on the resources and complete knowledge about the system by monitoring the current state of the resources. The main advantages of the centralized scheduling approaches are high efficiency and better resource monitoring. However, the centralized scheduling algorithm lack in the stability and fault tolerance capability. Hierarchical scheduling allows coordination of different schedulers. The schedulers in the lower level of the hierarchy possess the knowledge about the resources. The scalability and fault tolerance of the hierarchical approach are better than the centralized schedulers. In the distributed scheduling, there is no central entity for controlling the resources. The scheduling decision is shared by multiple distributed schedulers. The efficiency of the distributed scheduler is less than the centralized scheduler.

**6.4.3 Preemptive and Non-preemptive scheduling**

During preemptive scheduling, the jobs are interrupted during the middle of execution and migrated to another resource, while leaving the previous resources in the idle state and available for the execution of other jobs. This concept is highly useful for the priority-based job scheduling. During the non-preemptive scheduling, the resource is reallocated to another job, until the execution of scheduled job is completed.

**6.4.4 Online and Batch mode scheduling**

In the online scheduling, the jobs are scheduled immediately to the available resources, once they arrive into the system. The online scheduling algorithms are highly suitable for the cloud computing environment, due to the heterogeneity and varying processor speed of the cloud environment. In the batch mode scheduling, the jobs are allocated to the queue, once they arrive into the cloud computing system. The scheduling process starts only after a predefined time. The
main examples of the batch mode scheduling algorithms are First Come First Served scheduling algorithm (FCFS) and Round Robin scheduling algorithm (RR).

### 6.5 SCHEDULING ALGORITHMS

A scheduling policy is a set of rules that prioritizes the order for selecting the jobs for execution. The scheduling policies are described below

**First Come First Serve (FCFS):**

The jobs are prioritized according to the arrival time. The jobs that arrive earlier receive the higher priority over the jobs that arrive later. The jobs are dispatched according to their arrival time on the ready queue. The shortest job at the back of the queue has to wait for the completion of the longer job. FCFS algorithm is non-preemptive.

**Shortest Job First (SJF):**

The jobs are prioritized according to their execution time. The shortest jobs receive higher priority over the longer jobs. However, this policy can lead to the starvation of the jobs with longer jobs. This is the best approach to reducing the waiting time of the jobs in the ready queue. Like FCFS, SJF algorithm is non-preemptive. The main problem with the SJF algorithm is the requirement of precise knowledge about the time required for the execution of the job.

**Best Fit (BFit):**

The jobs are categorized according to the number of the resource. The scheduler looks for the job that best matches with the number of resources. The Best Fit scheduling algorithm calls the binary search procedure for searching the best job for the resource. This improves the resource utilization efficiency. However, the job queue should be reordered after each scheduling process.

**Worst Fit (WFit):**

The jobs are prioritized according to their size. The scheduling process proceeds from the shortest job to the longest job. The main goal is to fill the idle resources with small jobs. In the case of Worst Fit algorithm, the resources are sorted in the descending order of remaining capacity. But, this policy can also lead to the waiting state of large jobs.
6.6 PARALLEL JOB SCHEDULING

Parallel job scheduling [149], [150] is defined as the aggregation of multiple jobs and find an optimal resource to execute the jobs efficiently. The main objective of parallel job scheduling is efficient sharing of the resources for the execution of jobs, while ensuring required level of service. It improves the resource utilization efficiency and load sharing efficiency. Some of the scheduling algorithms may require a job to wait in the queue until all the resources are available. But, in the time slicing-based scheduling algorithms, the arriving jobs are executed immediately through the resource-sharing option. The parallel scheduling enables temporal and spatial sharing of the resources by dividing the overall execution time into multiple time slices. Hence, the jobs are scheduled to run concurrently.

6.7 PRIORITY-BASED JOB SCHEDULING

Priority of the jobs is an important issue in the scheduling process since there is a need to execute some jobs earlier than the rest of the jobs. The priorities are categorized as high, medium and low. Low priority is assigned to a job that exhibits low parallelism and requires lowest computational power for execution. Medium priority is assigned to a job that exhibits a medium level of parallelism and requires medium computational power. High priority is assigned to the job that exhibits high parallelism and requires high computational power. The fastest free resource available in the cloud is allocated to the job of high priority. The high priority jobs are executed first while the low priority jobs are held in the waiting state. The priority-based job scheduling algorithm optimizes the computational speed of the cloud computing system and reduces the usage of the resources. Hence, a consistent performance is achieved during the execution of the assigned jobs [151]. The priorities are classified into two types: Fixed and dynamic priorities. If the same priority is allocated to all jobs, it is called as fixed priorities. If different priorities are allocated to individual jobs, it is called as dynamic priorities. The priorities are defined internally or externally. Internally defined priorities use a measurable quantity to calculate the priority of a job. The examples of the internal priorities are time limits, memory requirements, Central Processing Unit (CPU) vs Input / Output (I/O) requirements. The external priorities are defined using specific criteria.
Furthermore, the priority-based scheduling approaches are classified as

**Pre-emptive**

This type of scheduling may preempt the resource if the priority of the newly arrived jobs is higher than the priority of the existing jobs. If the equal priority process is in running state, the resource is allocated to the high priority job, after the completion of the currently executing job.

**Non-preemptive**

This type of scheduling algorithm places the newly arrived jobs at the top of the ready queue. In this type of scheduling, the resource is allocated to the high priority jobs after completing the currently executing jobs.

### 6.8 BACKFILLING ALGORITHM

Backfilling [149] allows execution of the small jobs from the back of the queue before the execution of larger jobs that arrived earlier. Backfilling is a space-sharing optimization technique that is used with anyone of the scheduling policies. Using one of the scheduling policies, the job scheduler can build a schedule for all jobs in the waiting queue. This schedule determines a specific start time for each job. With backfilling, the priority order imposed by the scheduling policy is bypassed. Backfilling tries to balance the goals of resource utilization and maintaining the FCFS order. It requires that each job should specify its maximum execution time.

While the job at the head of the ready queue is waiting, it is possible to schedule the smaller jobs, especially if these jobs do not delay the start of the job on the head of the queue. Hence, there is a need to estimate the job execution time, as there is no delay incurred in the execution of higher priority jobs. By allowing execution of some jobs, the other jobs may get delayed. Backfilling approach never violates the FCFS order. In particular, a reservation for some future time is allocated for the jobs that need to wait. Backfilling may cause delays in the execution of other waiting jobs. The main goal of the backfilling approach is to improve FCFS by increasing the utilization rate of the resources and reducing the average waiting time of the jobs in the queue. However, this scheduling approach does not balance the computational load in the cloud computing system.
6.9 BEE BACKFILL WITH MIGRATION

A load balancing algorithm should attempt to improve the response time of the user-submitted jobs while improving the utilization rate of the available resource. In order to overcome the challenges and difficulties of the existing algorithms, a new algorithm called as “Bee Backfill with Migration” [152] is introduced.

6.9.1 System Architecture

![System architecture diagram](image)

Fig. 6.6 System architecture of the proposed approach
Cloud environment is created by implementing the data centers, cloud broker/scheduler, and virtual machines. Our proposed work introduced a new algorithm and compared makespan, waiting time, response time, number of migrations, and idle time of different algorithms. The main contribution of the work is to analyze different scheduling algorithms and to reduce the migration cost, waiting time, makespan, and response time. The number of migrations required in the proposed Bee Backfill with Migration algorithm is less than the existing scheduling algorithms.

Cloud broker is the mediator to perform the job in the resources. Upon the submission of a job to the job queue, the scheduler arranges the jobs according to the order of their arrival time and will specify their node requirement. ‘Bee Backfill with Migration’ schedules the jobs according to the order of their arrival time when there is enough number of nodes. When the number of inactive (idle) nodes is inadequate for a job, another job with a later arrival time but smaller node number requirement may be scheduled and the remaining jobs are backfilled to the queue. The backfilled jobs can check the loads on the virtual machines while deciding to migrate to some other nodes. Based on the loads, the virtual machines can be categorized into three groups: Partially loaded, Fully loaded and Balanced virtual machines. Fig. 6.6 shows the system architecture of our proposed approach.

The pseudo code for the proposed algorithm is described in the next section. Initially, the job is obtained from the queue and it is checked whether the job queue is null or not. The number of nodes required by the job and number of idle nodes are checked. If the number of the nodes is less than or equal to the number of idle nodes, the job is removed from the queue and dispatched to the idle nodes. The allocation matrix is updated. If the job is not at the head of the queue, the job is inserted into the backfill queue. The capacity of the remaining VMs, load on the VMs and time required for processing a job in the VM are calculated, if the number of nodes is less than or equal to the summation of the number of nodes running jobs arriving later than the job and number of idle nodes. The best suitable VM for migration is selected based on the load of the VM. The job is removed from the queue and dispatched to idle nodes and again the load is checked. The allocation matrix is updated and the next job is obtained from the job queue.
### Pseudo code for Bee Backfill with Migration

**Input:** The queue of incoming jobs  

**Allocation matrix:** Rows containing nodes and columns with jobs.  

**Output:** The updated allocation map  

1. **Begin**  
2. **job** ← Get the first job from job_queue  
3. **While** queue is not null do  
   1. check the number of nodes \( num_{nodej} \) required by job  
   2. check the number of idle nodes \( num_{idlej} \)  
   3. If \( num_{nodej} \leq num_{idlej} \)  
      (a) Remove job from job_queue(Q) and dispatch it to any \( num_{idlej} \) idle nodes  
      (b) Update the allocation matrix  
      (c) If job is not at the head of the job_queue (Q) then insert the job into \( backfill_{queue} \)  
   4. Else  
      (a) If the \( num_{nodej} \leq num_{backfill} + num_{idlej} \) is less than or equal to the summation of the number of nodes running jobs arriving later than the job and number of idle nodes then  
      (i) Suspend the jobs in backfill queue that arrive later than job and move them back to queue according to descending order of their arrival time until the number of idle nodes is greater than the number of nodes required by job.  
      (a) Find the capacity of remaining virtual machines  
         \[ Capacity = \frac{\text{sum of job size}}{\text{MIPS}} \]  
         MIPS is millions of instruction per second.  
      (b) Find the loads on virtual machines  
         \[ Load = \frac{\text{number of tasks}}{\text{service rate of virtual machines}} \]  
      (c) Find the time required to process a job in the virtual machine  
         \[ Time \text{ to process a job} = \frac{\text{load}}{\text{capacity}} \]  
      (d) Based on the loads (Partially loaded, Fully loaded or Balanced) of the virtual machines select the best suited Virtual machine for migration.  
      (ii) Remove job from queue (Q) and dispatch it to \( num_{idlej} \) idle nodes and call check load function  
      (iii) Update the allocation matrix  
      (iv) Get the next job from job_queue.
6.10 RESULTS AND DISCUSSION

The performance of the Bee Backfill with migration (bbm) is compared with FCFS [153], backfill (bf) [154], backfill with migration (bfm) [155], Conservative Migration and Consolidation Supported Backfilling (CMCBF) and Aggressive Migration and Consolidation Supported Backfilling (AMCBF) algorithms [147].

6.10.1 Response Time

The hybrid load balancing and the scheduling models namely FCFS, BF, BFM, CMCBF and AMCBF models. Table 6.1 shows the migration analysis for various scheduling models. Fig. 6.7 graphically illustrates the response time analysis variations with respect to the job size values. For the minimum job size (100), the existing FCFS consume 0.0819 (10^4) ms to provide the response and AMCBF consume 964.6464 (10^4) ms for maximum (1000) job size values. The load-balancing and the effective scheduling through the BBM model reduces the response time values for the respective job size as 0.0765 (10^4) ms and 906.4289 (10^4) ms. The comparative analysis shows that the proposed BBM provides the reduction of response time by 6.59 and 6.04 % compared to FCFS and AMCBF models.

<table>
<thead>
<tr>
<th>Job size</th>
<th>Response time (× 10^4 ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fcfs_rt</td>
</tr>
<tr>
<td>100</td>
<td>0.0819</td>
</tr>
<tr>
<td>200</td>
<td>1.45915</td>
</tr>
<tr>
<td>300</td>
<td>9.4626</td>
</tr>
<tr>
<td>400</td>
<td>39.31811</td>
</tr>
<tr>
<td>500</td>
<td>852.2438</td>
</tr>
<tr>
<td>600</td>
<td>909.5311</td>
</tr>
<tr>
<td>700</td>
<td>962.4715</td>
</tr>
<tr>
<td>800</td>
<td>973.6738</td>
</tr>
<tr>
<td>900</td>
<td>977.7728</td>
</tr>
<tr>
<td>1000</td>
<td>1082.4443</td>
</tr>
</tbody>
</table>

Table 6.1 Response time analysis for diverse job size values
6.10.2 Migration

The increase in the workload maximizes the number of migrations per VM considerably. But, the prioritization in the job execution depends on the size and VM capacity efficiently reduces the migration factor. Table 6.2 investigates the migration variations with respect to the various job size values.

Table 6.2 Migration analysis for diverse job size values

<table>
<thead>
<tr>
<th>Job size</th>
<th>No. of Migrations (× 25)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fcfs_mig</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
</tr>
<tr>
<td>200</td>
<td>2</td>
</tr>
<tr>
<td>300</td>
<td>3</td>
</tr>
<tr>
<td>400</td>
<td>3</td>
</tr>
<tr>
<td>500</td>
<td>4</td>
</tr>
<tr>
<td>600</td>
<td>4</td>
</tr>
<tr>
<td>700</td>
<td>5</td>
</tr>
<tr>
<td>800</td>
<td>5</td>
</tr>
<tr>
<td>900</td>
<td>6</td>
</tr>
<tr>
<td>1000</td>
<td>6</td>
</tr>
</tbody>
</table>
Fig. 6.8 Migration analysis for scheduling models

Fig. 6.8 depicts the migration variations with respect to the various job size values. The number of migrations is less in BF, CMCBF, FCFS models and they are higher in BFM and AMCBF models. The repetitive migrations and the extreme minimum migrations are not suitable for the workload conditions. The estimation of load size and the transfer of jobs among the VMs according to the estimated value provide the necessary mid-way migration performance.

The minimum number of migrations in the existing methods are 2 and the maximum number of migrations are 6 (FCFS). But, the no. of migrations for the proposed BBM are 1 and 4 for minimum (100) and maximum (1000) job size values on the scale of 25 respectively. The comparative analysis shows that the proposed BBF with migration provides 50 and 33.33 % better performance compared to AMCBF and FCFS model for respective job size.

6.10.3 Makespan

The time required for the overall scheduling of each machine is called makespan time. The variation of makespan time directly affects the optimal cost of the system. Table 6.3 shows the makespan variations of various scheduling models against the job size variations.
Table 6.3 Makespan analysis

<table>
<thead>
<tr>
<th>Job size</th>
<th>fcfs_ms</th>
<th>bf_ms</th>
<th>bfm_ms</th>
<th>cmcbf_ms</th>
<th>amcbf_ms</th>
<th>bbm_ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.08</td>
<td>1.89</td>
<td>1.89</td>
<td>2.27</td>
<td>1.89</td>
<td>1</td>
</tr>
<tr>
<td>200</td>
<td>5.92</td>
<td>6.23</td>
<td>3.47</td>
<td>5.98</td>
<td>3.05</td>
<td>1.82</td>
</tr>
<tr>
<td>300</td>
<td>41.3</td>
<td>35.06</td>
<td>6.77</td>
<td>28.06</td>
<td>8.77</td>
<td>5.11</td>
</tr>
<tr>
<td>400</td>
<td>67.32</td>
<td>40.92</td>
<td>20.32</td>
<td>46.92</td>
<td>31.14</td>
<td>12.3</td>
</tr>
<tr>
<td>500</td>
<td>378.68</td>
<td>141.83</td>
<td>23.78</td>
<td>235.43</td>
<td>37.19</td>
<td>21.45</td>
</tr>
<tr>
<td>600</td>
<td>393.31</td>
<td>182.34</td>
<td>33.89</td>
<td>256.32</td>
<td>38.13</td>
<td>29.25</td>
</tr>
<tr>
<td>700</td>
<td>393.95</td>
<td>184.77</td>
<td>40.82</td>
<td>275.88</td>
<td>40.92</td>
<td>35.68</td>
</tr>
<tr>
<td>800</td>
<td>411.46</td>
<td>201.93</td>
<td>43.23</td>
<td>275.89</td>
<td>41.44</td>
<td>36.87</td>
</tr>
<tr>
<td>900</td>
<td>414.45</td>
<td>204.01</td>
<td>49.36</td>
<td>329.85</td>
<td>50.37</td>
<td>42.56</td>
</tr>
<tr>
<td>1000</td>
<td>474.11</td>
<td>271.84</td>
<td>53.81</td>
<td>354.85</td>
<td>61.38</td>
<td>48.75</td>
</tr>
</tbody>
</table>

Fig. 6.9 Makespan analysis for scheduling models

Fig. 6.9 describes the variation of makespan time for the proposed BBM and existing scheduling models. For minimum job size (100), the makespan of FCFS model is less (1.08 X 10² ms) compared to other existing models and BFM provides 53.81 (X 10² ms) for maximum job size (1000). The comparison of proposed BBM scheduling with the existing scheduling models shows that the BBM model offers 7.41 and 9.40 % reduction in makespan for minimum and maximum job size respectively.
6.10.4 Waiting Time

The time period between the request initialization to the response generation refers waiting time. The occupation of a number of jobs and the absence of scheduling leads to high waiting time values. But, the minimum waiting time is the effectiveness of proposed algorithms. Table 6.4 shows the variation of waiting time for the various job size with the different scheduling algorithms.

Fig. 6.10 depicts the waiting time analysis with respect to the various job size. Among the various existing methods, the FCFS model provides the less waiting time performance. The FCFS model waits for $0.0148 \times 10^4$ ms and AMCBF model waits $930.1224 \times 10^4$ ms for the minimum (100) and maximum (1000) job size. The waiting time for FCFS model is less for high load compared to other scheduling mechanisms.

### Table 6.4 Waiting time analysis for diverse job sizes

<table>
<thead>
<tr>
<th>Job size</th>
<th>Waiting time ($\times 10^4$ ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fcfs_wt</td>
</tr>
<tr>
<td>100</td>
<td>0.0148</td>
</tr>
<tr>
<td>200</td>
<td>1.07435</td>
</tr>
<tr>
<td>300</td>
<td>7.2958</td>
</tr>
<tr>
<td>400</td>
<td>37.7546</td>
</tr>
<tr>
<td>500</td>
<td>822.6729</td>
</tr>
<tr>
<td>600</td>
<td>878.6843</td>
</tr>
<tr>
<td>700</td>
<td>929.4579</td>
</tr>
<tr>
<td>800</td>
<td>942.0618</td>
</tr>
<tr>
<td>900</td>
<td>945.1005</td>
</tr>
<tr>
<td>1000</td>
<td>1047.920319</td>
</tr>
</tbody>
</table>

The load balancing algorithm utilization in proposed work further optimize the waiting time as $0.0123 \times 10^4$ ms and $890 \times 10^3$ ms. The comparative analysis shows that the proposed BBF with migration provides 16.89 and 4.31 % better performance compared to AMCBF model for respective job size.
6.10.5 Idle Time

The time duration between the optimal VM placement and the scheduling refers an idle time and during which job execution is stopped and the corresponding machine is in an idle state. The dynamic resource provisioning and an effective load balancing concepts reduce the idle time.

Table 6.5 Idle time analysis for diverse job size values

<table>
<thead>
<tr>
<th>Job size</th>
<th>Idle time($\times 10^2$ ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fcfs_it</td>
</tr>
<tr>
<td>100</td>
<td>1.08</td>
</tr>
<tr>
<td>200</td>
<td>5.92</td>
</tr>
<tr>
<td>300</td>
<td>41.3</td>
</tr>
<tr>
<td>400</td>
<td>67.32</td>
</tr>
<tr>
<td>500</td>
<td>378.68</td>
</tr>
<tr>
<td>600</td>
<td>393.31</td>
</tr>
<tr>
<td>700</td>
<td>393.95</td>
</tr>
<tr>
<td>800</td>
<td>411.46</td>
</tr>
<tr>
<td>900</td>
<td>414.45</td>
</tr>
<tr>
<td>1000</td>
<td>474.11</td>
</tr>
</tbody>
</table>
Fig. 6.11 Idle time analysis for scheduling models

Hence, the algorithm effectiveness depends on the minimum idle time values. Table 6.5 presents the variation of idle time with respect to the job size for various scheduling models. Fig. 6.11 shows the idle time analysis with respect to the various job size values. For minimum job size, the idle time for FCFS model is $1.08 \times 10^2$ ms and the idle time of BFM model are $53.81 \times 10^2$ ms for maximum job size values respectively. The load balancing algorithm utilization in proposed work further optimize the idle time as $1.22 \times 10^2$ ms and $62.34 \times 10^2$ ms. The comparative analysis shows that the proposed BBM provides 11.48 and 13.68 % minimum compared to BF and FCFS model for respective job size.

6.11 SUMMARY

- Cloud environment is created by implementing the data centers, cloud broker/scheduler, and virtual machines.
- Scheduling of the parallel jobs is an important issue in the cloud data centers.
- The requirement of the computational resources for each parallel job is different in the case of complex applications.
- Hence, it is necessary to improve the throughput, response time, and utilization of the resources.
- Existing scheduling mechanisms normally take responsiveness as high priority and need non-trivial effort to make them work for data centers.
To overcome the difficulties of the existing scheduling mechanism, this work proposes a new algorithm that considers the priority of task as the main QoS factor.

The performance of the proposed algorithm is compared with existing scheduling algorithms such as First Come First Serve (FCFS) Backfill, Backfill with Migration, Conservative Migration and Consolidation Supported Backfilling (CMCBF), and Aggressive Migration and Consolidation Supported Backfilling (AMCBF) algorithms.

From the performance analysis results, it is clear that the proposed Bee Backfill with Migration algorithm shows better performance than the existing scheduling algorithms.

The number of migrations, response time, waiting time and idle time for the proposed algorithm are lower than the existing scheduling algorithms.