CHAPTER 4

OPTIMIZED LOAD BALANCING, JOB SCHEDULING AND DATA AWARE SCHEDULING IN CLOUD COMPUTING

4.1 INTRODUCTION

Cloud computing is nowadays being used for on-demand storage and processing power, which can be also used for dispatching user tasks or jobs to the available system resource like storage and software. It allows leasing of resources to improve the locally available computational capacity when necessary. The huge amount of computations in a cloud computing can be fulfilled in a specific time which cannot be done even by best super computers. However, cloud computing performance can still be improved by making sure all the VMs available in the cloud computing are utilized by a good scheduling algorithm. In cloud computing, scheduling also plays a major role for dispatching user tasks. The purpose of such algorithms is to make sure all nodes are equally involved in computations.

The work proposes a swarm intelligence which uses scheduling algorithm based on IPSO, Neural Network with Fuzzy logic and Data Aware Scheduling Algorithm for MTC. The good load balancing algorithm of IPSO effectively overcome the inherent flaw of getting local optimal value by Particle Swarm algorithm and finds the global optimum value in the search space again. Neural network and fuzzy algorithms are needed to carry on the task classification, fairness function definition of user tasks, the task and
resource parameterization, task resource mapping, and etc. The scheduling algorithm for Many-Task scheduling is data aware scheduling algorithm.

As demand increases, more resources are acquired, thus allowing faster response to subsequent requests that refer to the same data; when demand drops, resources are released. The algorithm analysis has evaluated the parameters like bandwidth, memory, CPU utilization and size which minimises the load, power, computation cost and processing time.

4.2 **OPTIMIZED LOAD BALANCING OF VIRTUAL MACHINE RESOURCES USING IPSO**

The rapid growth of data and computational needs in Distributed Systems and Cloud computing are gaining more and more attention. Clouds are playing an important and growing role in today’s networks. In cloud, the huge amount of computations can be done in a specific time which is difficult by best super computer. However, cloud performance can be improved by utilizing good load balancing algorithm, which makes sure that all nodes are equally involved in computations. The current research establishes a swarm intelligence which in term uses load balancing algorithm based on IPSO. This algorithm ahead’s for the cross operation, variation operation and select operation of the Genetic Algorithm (GA) to the Particle Swarm Optimization algorithm, it effectively overcome the inherent flaw of getting local optimal value by particle swarm algorithm and find the global optimum value in the search space again. The performance of this algorithm will be evaluated using several performance criteria (e.g. make span and load balancing level). The use of particle swarm optimization in cloud can yield better performance results in many scenarios than Genetic approach.
4.2.1 Architecture Diagram

Cloud environment is constructed by using VM resources, task and routers. The load balancing is achieved by using GA and IPSO algorithm. By comparing the GA and IPSO, the performance has been calculated and IPSO algorithm’s performance was always found to yield better results. Figure 4.1 shows the architecture of an IPSO algorithm.

Figure 4.1 Architecture of IPSO Algorithm
4.2.2 Methodology

The system starts with reading distinct number of machines and different number of tasks. For every individual machine, 3-Virtual Machines (VM) are created, then VM identifier is automatically generated by the system. Average load of CPU of a physical machine was also calculated. After getting tasks to Virtualization technology, it is able to carry out remapping between VM and physical resources according to the load change. This is done to achieve the load balance of the whole system in dynamic manner. There might be dynamic change in VMs, and there also might be an increase of computing cost of virtualization software and some unpredicted load wastage with the increase of VM number started on every physical machine. While considering the VM scheduling in cloud computing environment, with GA advantage, this module presents a balanced scheduling strategy of VMs based on GA. Starting from the initialization in cloud computing environment, the research was focused on the best scheduling solution by GA in every scheduling.

As same as GA, in his consideration of the VM scheduling in cloud computing environment, with the advantage of IPSO algorithm, this module also presents a balanced scheduling strategy of VMs based on IPSO algorithm. Starting from the initialization, every particle is encoded in the Particle Swarm algorithm. Ignoring the operation that uses the particle’s speed to update its location, here the arithmetic crossover mutation operator and choice operators to update the individual particles by GA is adopted. A particle is selected randomly from the newly generated set and was made to get into the next generation cycle. A best scheduling solution by intervention of IPSO algorithm in every scheduling was expected with possible outcomes.

The average load of every VM in period T, load of physical machine and migration cost was evaluated for both the algorithm. Obtained
result shows that improved particle swarm optimization algorithm have generated better results for all three parameters.

4.2.3 Performance Evaluation

The performance of the IPSO algorithm for load balancing was compared with the GA algorithm to evaluate the performance metrics such as cost, completion time and load which are required for optimized load balancing in cloud environment.

Figure 4.2 Performance of Genetic Algorithm

Figure 4.2 shown above has given the simulation of the performance evaluation of Genetic algorithm in cloud environment.
Figure 4.3 has given the simulation of performance evaluation of Improved Particle Swarm Optimization algorithm in cloud environment.

Table 4.1 shown below gives the evaluation of cost for the resources used for both GA and IPSO algorithm. Here, the total cost for the variable number of resources used in GA algorithm is higher than the proposed IPSO algorithm.

<table>
<thead>
<tr>
<th>Genesitic Algorithm</th>
<th>IPSO Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources</td>
<td>Cost</td>
</tr>
<tr>
<td>5</td>
<td>1500</td>
</tr>
<tr>
<td>8</td>
<td>2400</td>
</tr>
<tr>
<td>10</td>
<td>3000</td>
</tr>
<tr>
<td>15</td>
<td>4500</td>
</tr>
</tbody>
</table>
As per the findings of the cost applied for the number of resources used in the GA and IPSO algorithms, the performance evaluation was plotted as a graph as shown in the below Figure 4.4 with X-axis representing Number of Nodes and Y-axis refers to the Cost. These performance curves clearly notify that the proposed IPSO algorithm reduces the cost in better manner.

![Figure 4.4 Cost Vs Number of Resources](image)

**Figure 4.4 Cost Vs Number of Resources**

Table 4.2 below gives the average load distributed to the VM’s based on the number of resources used for GA and IPSO algorithm. Here, the result of load distribution in IPSO is better than GA which is clearly with the obtained value after the performance evaluation.

**Table 4.2 Average Load Distribution for the Number of Resources Used**

<table>
<thead>
<tr>
<th>Resources</th>
<th>Load</th>
<th>Resources</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>111691</td>
<td>5</td>
<td>78471</td>
</tr>
<tr>
<td>8</td>
<td>103872</td>
<td>8</td>
<td>75560</td>
</tr>
<tr>
<td>10</td>
<td>96400</td>
<td>10</td>
<td>66318</td>
</tr>
<tr>
<td>15</td>
<td>96810</td>
<td>15</td>
<td>77256</td>
</tr>
</tbody>
</table>
Based on the obtained values of average distribution of load in IPSO and GA, the graph has been plotted with X-axis denoting Number of Resources and Load by the Y-axis respectively in Figure 4.5.

![Graph showing Load Vs Number of Resources](image)

**Figure 4.5 Load Vs Number of Resources**

Table 4.3 shown below explains the iterations required to manage the load and the associated computation time for both the GA and IPSO algorithm. The results expressed the computation time per iteration are less for IPSO algorithm comparatively with GA.

**Table 4.3 Load computation time for various distributed loads**

<table>
<thead>
<tr>
<th>Genetic Algorithm</th>
<th>Iteration</th>
<th>Computation time</th>
<th>IPSO Algorithm</th>
<th>Iteration</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>960</td>
<td>0</td>
<td>893</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>960</td>
<td>10</td>
<td>893</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>960</td>
<td>20</td>
<td>785</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>926</td>
<td>30</td>
<td>785</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>926</td>
<td>40</td>
<td>785</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>926</td>
<td>50</td>
<td>730</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>874</td>
<td>60</td>
<td>730</td>
<td></td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>874</td>
<td>70</td>
<td>730</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>874</td>
<td>80</td>
<td>666</td>
<td></td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>840</td>
<td>90</td>
<td>666</td>
<td></td>
</tr>
</tbody>
</table>
The investigation of load balancing among the load applied to the VMs, the computation time has been calculated based on the variable iterations as plotted as a graph shown in Figure 4.6. As per the findings the iterations the computation time merely reduced for sequence of iterations in IPSO compared with GA. Here, the X-axis representing Number of Iterations and Y-axis refers to the Computation time.

![Figure 4.6 Computation Time Vs Number of Iterations](image)

4.3 JOB SCHEDULING USING NEURAL NETWORK

In Cloud Computing, based on the prior analysis the existing Berger Model worked with resource pool and the resource request can utilised the services from it. The theory of distributive justice model was used in the Berger Model for resource allocation in cloud computing. The proposed model used the Neural Networks (NN) and fuzzy for the job scheduling in cloud environment to carry on the task classification. This method initiate with task classification as the first step which is based on the QOS parameters like bandwidth, memory, CPU utilization and size. Then the classified tasks
are given to fuzzier, then proceed with NN and finally given for de-fuzzifier. In this method the main drawback is the high turnaround time. To overcome such an issue, a novel Fuzzy Neural Network model has used for job scheduling.

4.3.1 Architecture Diagram

The Fuzzy NN Architecture has shown in Figure 4.7 given below. The user task can be mapped with system resources and the classification of user tasks is mainly based on the parameters like bandwidth, memory, and completion time. The classified tasks are given to the fuzzification where it is used for conversion. Fuzzification is mainly used for converting the different range of input values into the range between 0 and 1. In Neural Network, the training set is maintained for learning. The training sets are stored in the centralized database. The training data in centralized database is compared with input for making decision. The converted input values are passed to the neural network. Neural network will make decision for mapping the system resources with the user tasks. With the help of de-fuzzification, the fuzzy range values are converted into their original values.

In neural network, if the user tasks are not mapped with the system resources and using back propagation algorithm means, the tasks are reclassified and follow the similar operations. The centralized database is used for maintaining and updating the training data required for neural networks. The feedback or feed forward algorithm is used to reclassify the user task if is not matched with the system resources.
4.3.2 Methodology

To achieve job scheduling in cloud computing different types of modules are used, which are as follows:

4.3.2.1 Modules

The Modules are,

i) Creation of Cloud environment
ii) Implementation of Task classification based on QoS
iii) Establishment of Fairness constraint and General expectations constraint
iv) Implementation of Tasks and Resource Mapping
v) Implementation of Fuzzy Neural Networks of QoS feature vector
4.3.2.2 Modules Description

The Module descriptions for the various modules given above are explained briefly which are as follows:

i) **Creation of Cloud Environment**: Cloud computing is a marketing term for technologies that provide computation, software, data access and storage services that do not require end-user knowledge for the physical location and configuration of the system that delivers the services. Similar to the electricity Grid, end-users consume power without the need to understand the component devices or infrastructure required for providing the service. Cloud computing is nowadays being used for on-demand storage and processing power. It allows the leasing of resources to improve the locally available computational capacity when necessary. With the help of Cloud, the user can access computing resources as general utilities that can be leased and released. The ubiquitous access to cloud resources easily enables the simultaneous use of different clouds.

Cloud computing is the advancement of Grid computing, Parallel computing and Distributed computing. It has the ability to provide the computing as a service rather than product such as shared resources, software information, etc. Internet is often represented as a Cloud and the term “Cloud Computing” arises from that analogy. Accenture defines Cloud Computing as the dynamic provisioning of IT capabilities (hardware, software, or services) from third parties over a Network. The basic mechanism of Cloud Computing is to dispatch the computing tasks to resource
pooling which constitutes by massive computers. The commercialization and the virtualization technology adopted by Cloud computing have poured into new features for Cloud architecture. This cloud environment obtained would compile of the resources for computation and worker nodes too. The scheduler for these Networks can schedule the jobs to worker nodes.

- **New VM scheduler:** VM Scheduler Time Shared over Subscription models, a scheduler that allows unbounded number of VMS to be deployed in a single VM, regardless its requirements in terms of number of MIPS. The behaviour of VM Scheduler Time Shared in CloudSim 1.0 Beta had changed in CloudSim 2.0 to accommodate requests with specific amount of MIPS.

- **New Datacenter Network Model:** An internal network model has been added to CloudSim 3.0. It supports definition of switches connecting hosts in arbitrary network topologies. New VM classes and Cloudlet classes were added to take advantage of this feature without breaking compatibility of older code. This new feature also enables modelling of message-passing applications. These are included in the package 'network datacenter'.

- **New VM Allocation and Selection Policies:** Six new VM allocation and four VM selection policies were added to the power package.

  ii) **Implementation of Task Classification Based on QoS:** The task classification is based on QoS parameter. The QoS stems from a parameter of internet performance mechanism. In
cloud computing, QoS is a metrics of user satisfaction with cloud services. Commercial characteristics of cloud computing raises its need to provide satisfaction service for different users. However, the diversity of users set a higher request to job scheduling and resource allocation in cloud computing environment.

The users tasks will be classified according to QoS and the system mainly consider the following QoS parameters.

- **Completion time**: Time To Completion (TTC) is a calculated amount of time required for any particular task to be completed. Completion, is defined by the span from conceptualization to fruition (delivery), and is not iterative. TTC is commonly used when reporting on unmovable dates within a project time line. For real-time demand higher users, the task needs to be completed within as little time as possible.

- **Bandwidth**: In computer networking and computer science, the words bandwidth, network bandwidth, data bandwidth or digital bandwidth are colloquial and metaphoric terms widely used in textbooks as well as scientific papers, patents and standards to refer to various bit-rate measures, representing the available or consumed data communication resources expressed in bits/second or multiples of it (kilobits/s, megabits/s etc.). If user requests need higher communication bandwidth, the bandwidth needs to distribute the jobs to the several nodes.

- **Memory**: One of the main benefits of cloud computing is elasticity. Public Clouds Services such as Amazon Web Services provide services for elastic compute (EC2),
elastic storage (S3), and many other core infrastructure components. One service is remarkably absent though: elastic memory. The reason for this is fairly simple: building a generic service for elastic memory is virtually impossible based on technology available today.

To measure user satisfaction according to different QoS parameters, different quantification evaluation criteria need to be established for different QoS parameters.

**iii) Establishment of Fairness Constraint and General Expectations Constraint:** Two fairness constraints of job scheduling exist in cloud computing. In cloud computing, the fairness of resource can provide reasonably available resources for user tasks according to the characteristics and the preferences of user tasks. It also enables different users to get what end user wanted in with ultimate satisfactory. The user task requires some resources to carry out. At the same time, the task has QoS preferences due to the different characteristics of user applications. The resources allocation in accordance with the corresponding QoS preference is general expectation (namely resource distribution standard in referential structure), i.e., justice allocation. The selection of resources is based on general expectations in local structure. The mapping relation between QoS characteristics and resources of the user tasks corresponds to the relation between Cx and GOX. The mapping between Cx and GOX can be used for overwhelming the constraint in mapping between Cx and GOX, so that tasks tend to obtain a fair resources allocation. In addition, the fairness constraint of the resource selection
process can also be achieved through the general expectations constraint.

Expectation states formed by a series of theories are used to study actor and evaluate the impact of their behaviour. In brief, expectation states theories are to be studied for the following two issues. Firstly actor, how to generate expectations of itself and other individual’s according to the information (such as status, reward and performance differences) around the world; Secondly these expectation, how to affect the behaviour (such as participatory and decision-making influence) of actors and others.

Expectation status theories have been expanded and applied widely. State value theory of distribution justice is an important theoretical basis of the paper. It describes allocator on how to use referential comparisons to establish the expectation for reward allocation. The expectations are used to evaluate the justice or injustice of distribution in a variety of circumstances. Because computer resources can be quantified, the research work was perforated only with the quantified allocation principle (namely the theory of distribution) in Berger model. The basic ideas of distribution justice is that individual in social system can judge its own gained resources to be fair or not through distribution relations comparisons between itself and other ordinary person in referential structure. The distribution relation of other ordinary person in referential structure is well recognized as the general justice, i.e., fairness.
The Berger model shown in Figure 4.8 of distributive justice characteristic ‘C’ is any aspect of a person. A goal-object, GO, is any object that an actor might want, namely expectation. $C_x / c_x$ and the similarity condition are: $c_x$ is similar to some state $C_x$; $go_x$ is similar to some state $GO_x$. The structure condition is: $c_x$ acquires the status value of $C_x$; $go_x$ acquires the status value of $GO_x$.

Given the similarity and structure conditions, states in the local structure will acquire the status value of the states to which there are similar values in the referential structure. In short, the justice of distribution can be judged through comparison between local structure and reference structure. Moreover, it is worth to mention that the definition of distributive justice evaluation function in Berger model.

### iv) Implementation of Tasks and Resource Mapping:

The task classification and resources mapping were executed in this module. Cloud computing uses virtualization technology to map host resources to the virtual machine layer. The job scheduling in cloud computing is implemented in application
layer and virtual machine layer. Scheduling is to map task to resources according to a certain principle of optimization.

Cloud Computing makes required resources of task manifest in the form of a virtual machine. The resource search process converts the search process of a virtual machine. The next step is to establish the general expectations function of task. In order to restrict the fairness of the selection resource process, it is needed to adjust the performance ratio parameter of the selected virtual machine resources according to the different QoS goals of user tasks. Also mapping the relations of general expectation corresponds to the association between \( C_x \) and \( GO_X \). The minimum Euclidean distance between the vector of general expectation mapping relations and the vector of actual allocation normalized resource parameters is used to constrain the convergence of gox and \( GO_X \).

- **Task Justice**: The actual allocation resources for a task are close, to the maximum extent, to the expected resources of the task, called the task to obtain the fair execution.

- **System Justice**: It assume the system task set is \( T = \{T1, T2, .., Tn\} \) and the corresponding system justice function set is: \( J = \{J1, J2, .., Jn\} \).

According to general expectations, optimization of the selection process for resources in local structure can be done. The mapping relation between QoS characteristics and resources of the user tasks corresponds to the relation between \( C_x \) and \( GO_X \). The mapping between \( C_x \) and \( GO_X \) can be used for constraint
the mapping between $C_x$ and $go_x$, so that tasks tend to obtain a fair resources allocation.

v) **Implementation of Fuzzy Neural Networks of QoS Feature Vector:** Implementation of a Fuzzy Neural Networks for job scheduling environment is described here. The Fuzzy Neural Networks selection section envisages about the various testing and simulations to find a suitable neural networks. Fuzzification can be used for converting the different range of input values into the range of 0 and 1. Fuzzy logic is a form of many-valued logic or probabilistic logic; it deals with reasoning that is approximate rather than fixed and exact. In contrast with traditional logic theory, where binary sets have two valued logic: true or false, fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. Furthermore, when linguistic variables are used, these degrees may be managed by specific functions. Fuzzy logic has been applied to many fields, from control theory to artificial intelligence.

Neural Network can be used to find out or choose the best solution among various solutions. Neural Network that possesses the topology of one input layer, one hidden layer and an output layer is the optimal one with a learning rate of 0.4. Hidden layer is used for adjusting the different range of input values. Before implementing the decided Neural Networks to the assigner, two features of back propagation algorithm were used to test the Neural Networks for best
possible result. It was assumed that the deadline, resource and processing time of each job are known in advance and that the system had a user interface section which accepts user input for each job. Neural Networks can consider the parameters like bandwidth, memory and completion time. It was proved that the Fuzzy Neural Networks can submit the job successfully in sequence order.

Trained data set in Neural Networks represents equivalent form of dispatching rules for back propagation algorithm. Hence the greatest advantage of Neural Networks is that, it will provide ‘rule less’ solution environment for an application. Artificial Neural Networks typically start out with randomized weights for all their neurons. This means that they don’t "know" anything and must be trained to solve the particular problem for which they are intended. Broadly speaking, there are two methods for training an ANN, depending on the problem it must solve.

A back-propagation ANN, conversely, is trained by humans to perform specific tasks. During the training period, teacher evaluates whether the ANN's output is correct. If it's correct, the neural weightings that produced that output are reinforced; if the output is incorrect, those weightings responsible are diminished. This type is most often used for cognitive research and for problem-solving applications. De-fuzzification can be used for reversing the process of fuzzification.
4.3.3 Performance Evaluation

The performance evaluation compares the parameter metrics such as completion time, bandwidth requirement, computing power, that is, which task has the better parameter metrics which is required for scheduling the jobs in Cloud environment.

Figure 4.9 shown below has given the simulation details of Berger Model consists of the completion time for the task and resources allocation.

The simulation of Job scheduling algorithm based on the Fuzzy Neural Network model has given in Figure 4.10.
The results from the experiment show that the proposed approach works better than the earlier systems. Tasks that cannot match with the system resources can be re-classified using Back Propagation Algorithm.

The tasks completed after the expected completion time is considered as incomplete tasks. In Neural Network with the help of back propagation algorithm, task can be reclassified and processed for achieving better completion time.

Table 4.4 has given the performance evaluation of Completion Time and Bandwidth Utilization used in both Berger Model and proposed Fuzzy neural network model. The time and Bandwidth performance clearly expressed that the Fuzzy neural network model is better than the existing Model.
Table 4.4 Performance Evaluation of Fuzzy Neural Network Model

<table>
<thead>
<tr>
<th>No of Nodes</th>
<th>Berger model</th>
<th>Fuzzy Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bandwidth</td>
<td>Completion Time</td>
</tr>
<tr>
<td>10</td>
<td>446</td>
<td>2027</td>
</tr>
<tr>
<td>15</td>
<td>1796</td>
<td>7804</td>
</tr>
<tr>
<td>20</td>
<td>4486</td>
<td>21280</td>
</tr>
<tr>
<td>25</td>
<td>7609</td>
<td>41445</td>
</tr>
<tr>
<td>30</td>
<td>12911</td>
<td>76530</td>
</tr>
<tr>
<td>35</td>
<td>19666</td>
<td>131578</td>
</tr>
</tbody>
</table>

Figure 4.11 shown below has given the performance of Bandwidth utilization in Berger Model and Fuzzy neural Network model. The Performance curve explicitly expressed the proposed FNN model gives the better Bandwidth utility and the X-axis represents the Number of Nodes used and Y-axis denotes the Bandwidth utility.
Figure 4.12 denotes the Time Evaluation used in FNN and Berger Model. The Figures also represents number of Nodes in X-axis and Completion Time in Y-axis respectively. From the performance Curve, it is clearly understood that the FNN consumes very less time for completion of assigned Task.

![Figure 4.12 Completion Time Vs Number of Nodes](image)

4.4 DATA AWARE SCHEDULING FOR MANY-TASKS SCIENTIFIC COMPUTING

Cloud Computing proposes an alternative in which resources are no longer hosted by researcher’s computational facilities, but are leased from big data centers only when needed. The Scientific Computing community has started to focus on Many-Task Computing (MTC), that is, on high-performance execution of loosely coupled applications comprising many tasks. The cloud computing services for many tasks computing and applications span a broad range of possible configurations, but utilizing large numbers of computing resources over short periods of time to accomplish many computational tasks. The presence of MTC component in Scientific
Computing workloads and quantification of the presence of these users in Scientific Computing environments are investigated. Besides these evaluation of the work of MTC with the well-known micro benchmarks and application kernels in the performance of four commercial Cloud Computing services that can be used for Scientific Computing. But the scheduling algorithms for MTC were not considered. The proposed approach introduces one scheduling algorithm for Many-Task scheduling through Data Aware Scheduling Algorithm.

4.4.1 Architecture Diagram

Data Aware scheduling algorithm is mainly meant for Many-Task Scheduling. Based on the available system resources the scheduling algorithm takes place on a high performance scientific computing execution. Here, the resources are to be increased as the demand increases.

Figure 4.13 Architecture diagram of Data Aware Scheduling Algorithm
4.4.2  Methodology

Data aware scheduling algorithms follow different modules to schedule Many-Task computing and are as follows:

4.4.2.1  Modules

The Modules are

i) Creation of Cloud Environment
ii) Implementation of Many Task Scheduling
iii) Implementation of Data Aware Scheduling

4.4.2.2  Modules Description

i) Creation of Cloud Environment: As discussed in the previous method the cloud computing environment is consists of the resources for the computation and worker nodes and the scheduler for this network which schedules the jobs to the worker nodes. When using a cloud, the user accesses computing resources as general utilities that can be leased and released. Recently, the scientific computing community has started to focus on Many-Task Computing, that is, on high-performance execution of loosely coupled applications comprising Many Tasks.

ii) Implementation of Many-Task Computing: Many-Task Computing (MTC) applications span a broad range of possible configurations, but utilizing large numbers of computing resources over short periods of time to accomplish many computational tasks, where the primary metrics are in seconds. Here, the investigation implies the presence of a
MTC component in Scientific Computing workloads and quantify the presence of these users in scientific computing environments. And also, further evaluation with the work by well-known micro benchmarks and application kernels, the performance of four commercial cloud computing services that can be used for scientific computing was carried over. The method for identifying proto-MTC users—users with a pronounced MTC-like workload, which are potential MTC users in the future—in existing system workloads is based on the identification of users with many submitted tasks and/or bags-of-tasks in the workload traces taken from real scientific computing infrastructures. Since none of the scheduling algorithms for Many-Task Computing was considered this methodology provide the optimal performance of the system.

MTC includes loosely coupled applications that are generally communication-intensive but not naturally expressed using standard message passing interface commonly found in High Performance Computing (HPC), drawing attention to the many computations that are heterogeneous but not parallel. There is more to HPC than tightly coupled MPI, and more to HTC than embarrassingly parallel long running jobs. Like HPC applications, and science itself, applications are becoming increasingly complex opening new doors for many opportunities to apply HPC in new ways if the work broadens the research perspective. Some applications have just so many simple tasks and it is obvious that managing them is hard. Applications that operate on or produce large amounts of data need sophisticated data management in order to scale. There exist applications that involve many tasks, each composed of
tightly coupled MPI tasks. Loosely coupled applications often have dependencies among tasks, and typically use files for inter-process communication. Efficient support for these sorts of applications on existing large scale systems will involve substantial technical challenges and will have big impact on science.

There are many differences between the high-throughput computing, high-performance computing (HPC), and many-task computing (MTC). HPC tasks are characterized as needing large amounts of computing for short periods of time, whereas HTC tasks also require large amounts of computing, but for much longer times (months and years, rather than hours and days). HPC Environments are often measured in terms of FLOPS. The HTC community is not concerned about operations per second, but rather operations per month or per year. Therefore, the HTC field is more interested in how many jobs can be completed over a long period of time instead of how fast an individual job can be completed. As a general rule, HPC systems are tightly coupled parallel jobs be such that they must execute within a particular site with low-latency interconnects. Conversely, HTC systems are independent, sequential jobs that can be individually scheduled on many different computing resources across multiple administrative boundaries. HTC systems achieve this using various grid computing technologies and techniques.

Many parallel applications consist of multiple computational components. While the execution of some of these components or tasks depends on the completion of other tasks, others can be executed at the same time, which increases
parallelism of the problem. The task scheduling problem is the problem of assigning tasks in the system in a manner that will optimize the overall performance of the application, while assuring the correctness of the result. The primary goal of task scheduling is to schedule tasks on processors and minimize the make span of the schedule, i.e., the completion time of the last task relative to the start time of the first task. The output of the problem is an assignment of tasks to processors.

MTC aims to bridge the gap between the HTC and HPC. MTC is reminiscent of HTC, but it differs in the emphasis of using many computing resources over short periods of time to accomplish many computational tasks (i.e. including both dependent and independent tasks), where the primary metrics are measured in seconds (e.g. FLOPS, tasks/s, MB/s I/O rates), as opposed to operations (e.g. jobs) per month. MTC denotes high-performance computations comprising multiple distinct activities, coupled via file system operations.

HPC systems are tightly coupled parallel jobs and such they must execute within a particular site with low-latency interconnects. Conversely, HTC systems are independent, sequential jobs that can be individually scheduled on many different computing resources across multiple administrative boundaries. HTC systems achieve this using various grid computing technologies and techniques. Many-Task Computing applications span a broad range of possible configurations, but utilizing large numbers of computing resources over short periods of time to accomplish many computational tasks, where the primary metrics are in seconds.
MTC workloads may comprise tens of thousands to hundreds of thousands of tasks and BoTs, and a typical period may be one year or the whole trace. The method for identifying proto-MTC users—users with a pronounced MTC-like workload, which are potential MTC users in the future—in existing system workloads is based on the identification of users with many submitted tasks and/or bags-of-tasks in the workload traces taken from real scientific computing infrastructures. The work defines an MTC user to be a user that has submitted at least ‘J’ jobs and at least ‘B’ bags-of-tasks. The user part of our definition serves as a coupling between the jobs, under the assumption that a user submits jobs for execution toward an arbitrary but meaningful goal.

The jobs part ensures that the work focus on high-volume users; these users are likely to need new scheduling techniques for good system performance. The bag-of-tasks part ensures that task submission occurs within a short period of time; this submission pattern raises new challenges in the area of task scheduling and management. Ideally, it should be possible to use a unique pair of values for J and B across different systems. To investigate the presence of an MTC component in existing scientific computing infrastructures, the work analyses ten workload traces.

iii) Implementation of Data Aware Scheduling: Data-aware scheduling is central to the success of data diffusion, as harnessing data-locality in application access patterns is critical to performance and scalability. As demand increases, more resources are acquired, thus allowing faster response to subsequent requests that refer to the same data; when demand
drops, resources are released. The work implements four dispatch policies. They are First-Cache-Available (FCA) Policy, Max-Cache-Hit (MCH) Policy, Max-Compute-Util (MCU) and Good-Cache-Computing (GCC).

The first-available policy ignores data location information when selecting an executor for a task; it simply chooses the first available executor, and provides the executor with no information concerning the location of data objects needed by the task. Thus, the executor must fetch all data needed by a task from persistent storage on every access. This policy is used for all experiments that do not use data diffusion.

The max-cache-hit policy uses information about data location to dispatch each task to the executor with the largest amount of data needed by that task. If that executor is busy, task dispatch is delayed until the executor becomes available. This strategy is expected to reduce data movement operations compared to first-cache-available and max-compute-util, but may lead to load imbalances where processor utilization will be sub optimal, if nodes frequently join and leave.

The max-compute-util policy leverages data location information, attempting to maximize resource utilization even at the potential higher cost of data movement. It sends a task to an available executor, preferring executors with the most needed data locally. The Good-Cache-Compute (GCC) policy is a hybrid MCH/MCU policy.

The GCC policy sets a threshold on the minimum processor utilization to decide when to use MCH or MCU. The work defines processor utilization to be the number of processors with active tasks divided by the total number of processors
allocated. MCU used a threshold of 100%, as it tried to keep all allocated processors utilized. The proceeding work finds that relaxing this threshold even slightly (e.g., 90%) and it keeps processor utilization high and it gives the scheduler flexibility to improve cache hit rates significantly when compared to MCU alone.

The scheduler is a window based one, that takes the scheduling window W size (i.e. |W| is the number of tasks to consider from the wait queue when making the scheduling decision), and starts to build as per task scoring cache hit function. If at any time, a best task is found (i.e. achieves a 100% hit rate to the local cache), the scheduler removes this task from the wait queue and adds it to the list of tasks to dispatch to this executor. This is repeated until the maximum number of tasks work is retrieved and prepared to be sent to the executor. If the entire scheduling window is exhausted and no best task was found, the ‘m’ tasks with the highest cache hit local rates are dispatched. In the case of MCU, if no tasks, the work found that would yield any cache hit rates, then the top ‘m’ tasks are taken from the wait queue and dispatched to the executor. For MCH, no tasks are returned, signalling that the executor is to return to the free pool of executors. For GCC, the aggregate CPU utilization at the time of scheduling decision determines which action to take.

Pre-binding of tasks to nodes can negatively impact cache-hit performance if multiple tasks are assigned to the same node, and each task requires the entire cache size, effectively thrashing the cache contents at each task invocation. In practice, the proceeding work finds that, per task working sets
are small (megabytes to gigabytes) while cache sizes are bigger (tens of gigabytes to terabytes) making the worst case not a common case. The work define several variables first in order to understand the scheduling algorithm pseudo-code and the algorithm is separated into two sections, as the first part decides which executor will be notified of available tasks, while the second part decides which task to be submitted to the respective executor. It provides high performance for Many Tasks Computing Environment. Finally the results oriented to define the model and then analyze the computational time per task, caching performance, workload execution times, arrival rates, and node utilization.

4.4.3 Performance Evaluation

It is a vital fact that the work are evaluating the proposed approach with the earlier approach for identify the utilization of resources. And the work present the evaluation comparison by the parameter metrics such as the computational time per task, caching performance, workload execution times, arrival rates, and node utilization. Based on the comparison and results from the experiment show the proposed approach works better than any other earlier systems.

The work measured the performance of the data-aware scheduler on various workloads, both with Static (SRP) and Dynamic (DRP) Resource Provisioning, and executes experiments on the ANL/UC Tera Grid (up to 100 nodes, 200 processors). The Falkon service was proposed on an 8-core Xeon®@2.33GHz,2GB RAM, Java 1.5, 100Mb/s network, and 2 ms latency to the executors. The persistent storage was GPFS with <1ms latency to the executors.
The three sub-sections cover three diverse workloads: Monotonically-Increasing, Sine-Wave and All-Pairs. The current research uses workloads MI and SI to explore the dynamic resource provisioning support in data diffusion and the various scheduling policies (e.g. FA, GCC, MCH, MCU) and cache sizes (e.g. 1GB, 1.5GB, 2GB, 4GB). To compare data diffusion with active storage AP workloads were used. For all workloads the work was generated at random, using uniform distribution. Although this might not be representative of all applications, the work was progressed with the assumptions that workloads with uniform distribution will stress the data-aware scheduler far more than if the distribution of the work, where few files work extremely popular and many files work unpopular.

In general, zipf distribution workloads require a smaller aggregate cache in relation to the workload working set. From data locality perspective, uniform distribution workloads offer the worst case scenario. It should be noted that zipf distributions would have caused hot-spots to occur for popular files, which does not naturally occur in uniform distribution workloads. However, due to our dynamic resource provisioning, when many new compute and storage resources are joining frequently, the initial local storage population of data can certainly cause the same kind of hot-spots by putting heavier stress on existing nodes. Data diffusion handles these hot-spots naturally, as temporarily popular data (due to new nodes joining and having an empty cache) get replicated at the new storage locations, and subsequent accesses to this data can be served locally. Therefore, in the developed system, these hot-spots are only temporary, while the popular data diffuses to other storage nodes, and subsequent accesses to this data are then localized effectively eliminating the hot-spot.

The aggregate I/O throughput was measured in each of the seven experiments. The work present in each case first, as the solid bars, the average
throughput achieved from start to finish, partitioned among local cache, remote cache, and GPFS, and second, as a black line, the “peak” throughput achieved during the execution. The second metric is interesting because of the progressive increase in job submission rate: it may be work as a measure of how far a particular method can go in keeping up with user demands. It was maintained that the FCA policy had the work average throughput of 4Gb/s, compared to the work 5.3Gb/s and 13.9Gb/s for data diffusion (GCC, MCH, and MCU with various cache sizes), and 14.1Gb/s for the ideal case. In addition to having higher average throughputs, data diffusion also achieved significantly throughputs towards the end of the experiment when the arrival rates are highest, as high as 81Gb/s as opposed to 6Gb/s for the FCA policy. GPFS file system load (the red portion of the bars) significantly the work with data diffusion than for the GPFS-only experiments (FA); in the worst case, with 1GB caches where the working set did not fit in cache, the load on GPFS is still high with 3.6Gb/s due to all the cache misses, while FA tests had 4Gb/s load. The work, as the cache sizes increased and the working set fit in cache, the load on GPFS became as low as 0.4Gb/s; similarly, network load was considerably the work, with the highest values of 1.5Gb/s for the MCU policy, and less than 1Gb/s for the other policies.

Analysis and comparison of the performance offered by Many-Tasks Scientific Computing with our proposed method of Data Aware scheduling algorithm is given. Here if the number of resources increased, the caching performance is increased linearly. Based on the results and comparison of the experiment, proposed approach works better than the other earlier systems.

Figure 4.14 has shown below has given the simulation of the Performance Evaluation of Cloud computing services for Many Task Scientific Computing in CloudSim Simulator.
Figure 4.14  Performance evaluation of cloud computing services for Many Task Scientific Computing

Table 4.5 shown below has given the performance evaluation of proposed Data aware scheduling algorithm to the HPC/HTC model with the parameters Execution time and Workload distribution. Based on the analytical results, it very clearly state that the compilation time and work load distribution of Data aware scheduling has done a better performance than existing approach.
Table 4.5 Performance Evaluation of Data Aware Scheduling

<table>
<thead>
<tr>
<th>No. of Resources</th>
<th>HPC-HTC model</th>
<th>Data Aware Scheduling Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Execution Time</td>
<td>Work Load</td>
</tr>
<tr>
<td>5.0</td>
<td>375</td>
<td>150</td>
</tr>
<tr>
<td>8.0</td>
<td>380</td>
<td>2500</td>
</tr>
<tr>
<td>10.0</td>
<td>350</td>
<td>2300</td>
</tr>
<tr>
<td>12.0</td>
<td>390</td>
<td>5500</td>
</tr>
<tr>
<td>15.0</td>
<td>950</td>
<td>6500</td>
</tr>
</tbody>
</table>

Table 4.6 shown below has given the performance evaluation of resource cache and the resource utility by the HPC/HTC model and the proposed Data aware model. Based on the analytical results, it very clearly state that the cache performance and Utility of Data aware scheduling has done a better performance than existing approach.

Table 4.6 Performance Evaluation of Resource Management

<table>
<thead>
<tr>
<th>No. of Resources</th>
<th>HPC-HTC model</th>
<th>Data Aware Scheduling Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Utilization</td>
<td>Cache</td>
</tr>
<tr>
<td>5.0</td>
<td>3</td>
<td>1240</td>
</tr>
<tr>
<td>8.0</td>
<td>2</td>
<td>600</td>
</tr>
<tr>
<td>10.0</td>
<td>2</td>
<td>2800</td>
</tr>
<tr>
<td>12.0</td>
<td>2</td>
<td>6700</td>
</tr>
<tr>
<td>15.0</td>
<td>1</td>
<td>8900</td>
</tr>
</tbody>
</table>

Figure 4.15 meant for the evaluation of Execution Time over the resources used in the Data aware algorithm and HPC/HTC model. The performance curve denotes that the DA scheduling method has better performance. The X-axis indicates the resources used and the Y-axis denoted the execution time.
Figure 4.15 Execution Time Vs No. of Resources

Figure 4.16 meant for the evaluation of work load distribution over the resources used in the Data aware algorithm and HPC/HTC model. The performance curves explained very clearly the DA scheduling method has better performance than the existing approach. The X-axis indicates the resources used and the Y-axis denoted the work load.

Figure 4.16 Work Load Vs No. of Resources
Figure 4.17 meant for the evaluation of work utility over the resources used in the Data aware algorithm and HPC/HTC model. From the performance curves it is clearly understood that the DA scheduling method has better performance than the existing approach. The X-axis indicates the resources used and the Y-axis denoted the work utility.

![Work Vs Resource Utilization](image)

**Figure 4.17 Work Vs Resource Utilization**

Figure 4.18 has a evaluation of Caching performance over the resources used in the Data aware algorithm and HPC/HTC model. The X-axis indicates the resources used and the Y-axis denoted the caching. The DA scheduling method has given a better performance than the existing approach.
4.5 SUMMARY

The cloud computing performance improvement achieved leads to maximize the utilization of resources. In this chapter discussion about various scheduling approaches like Improved Particle Swarm Optimization, Neural Network and Fuzzy Algorithm and Many-Task Scheduling was carried over. The improved Particle Swarm Optimization provided good load balancing by provided local optimal value by using the particle swarm algorithm and found the global optimum value in such places. The second approach explains the job scheduling using fuzzy neural network, which can reduce the bandwidth utilization, compilation time and memory utilization compared with rational algorithm. The third method, Data Aware Scheduling for Many-Task Scheduling meant for many-task computing application span a guard range of possible configuration. This method investigates the presence of MTC compared in scientific computing approach & quantifying the presence of the user scientific computing environment. It also evaluated that the work and
application kernels of cloud computing services, which can be used for scientific computing. Based on the comparison, this method shows the better performance on work load, execution time, arrival rate and more utilization. Thus, the investigations made on the various methods explain the better services in optimizing the cloud based scheduling algorithms.