Enhanced Bat Optimization Algorithm and Low Latency Fault Tolerance Model for Efficient Resource Allocation in Grid Computing

R.Ananthi Lakshmi[1], Dr.S.Vidhya[2]

[1]*PhD Scholar Department of Computer Science KG College of Arts and Science Coimbatore.*

[2]*Assistant Professor Department of Information Technology KG College of Arts and Science Coimbatore*

Abstract - **Effectual allocation of resources with fault tolerance is one of the important targets in any computational grid environment to accomplish the task execution on time. In the existing system, computational complexity and error rates are still an issue. Also the requirements and grid services are not satisfied effectively. Hence, the overall grid computing performance is reduced prominently. To overcome the above mentioned issues, in this work, Enhanced Bat Optimization (EBO) algorithm and Low Latency Fault Tolerance (LLFT) model is proposed to improve the optimal resource allocation and fault tolerance over grid computing environment. The proposed system includes main phases are such as system model, load balancing, resource allocation and fault tolerance system. Initially, consider the number of resources, number of tasks, Virtual Machine (VM) and number of grid users over the grid computing. In this work, load balancing is done by using MMH algorithm which is used to equalize the total workloads over grid. Load balancing is achieved by transferring tasks from over-loaded nodes to under-loaded nodes. Then the resource allocation is done by using EBO algorithm which is used to select more optimal resources effectively. The best fitness values are used to choose the available optimal resources. The fault tolerance is performed using LLFT which provides fault tolerance for distributed applications deployed within a grid computing using the leader/follower replication. The simulation result concludes that the proposed EBO+LLFT algorithm provides better performance by means of higher accuracy, lower error rate, cost complexity and time complexity than the existing algorithms.**

Index Terms - **Grid computing, Max-Min Heuristic (MMH), Enhanced Bat Optimization (EBO) algorithm, load balancing, resource allocation, fault tolerance, Low Latency Fault Tolerance (LLFT) model.**

I.INTRODUCTION

Grid computing, in which a network of computers is integrated to create a very fast virtual computer, is becoming ever more prevalent. It is the future computing paradigm for enterprise applications. Large scale grids are complex systems, composed of thousands of components belonging to disjoined domains. Planning the capacity to guarantee Quality of Service (QoS) in these environments is a challenge because global Service Level Agreements (SLA) depends on local SLAs, i.e., SLAs established with components that make up the grid. These components are generally autonomous and join the grid as part of a loose federation. Only if all these partial SLAs are satisfied, the global SLA will be satisfied [1].

Grid computing can be used to solve problems related with computation intensive applications related with different areas. Ecommerce applications can also benefit from grid computing in the form of faster decision making and accurate forecasting. Grid computing permits researchers to use the grid resources spread across the world to solve problems related to the earth observation, marine, and environmental sciences [2]. In the field of medical sciences, grid computing can give benefits in the form of decision making at remote sites. Specialized practitioners with their expertise can take decisions based on computing intensive operations and analysis through grid computing resources. In the field of bioinformatics, the day by day need of grid computing is growing as the distributed information related to bio-informatics is increasing [3].

Due to uneven job arrival patterns and unequal computing capabilities, some nodes may be overloaded while others may be under-utilized. Load balancing mechanism aims to equally spread the load on each node of the grid, optimizing their utilization, throughput and response [4] [5]. In order to achieve these goals, the load balancing mechanism should be ‘‘fair’’ in distributing the load across the nodes; by being ‘‘fair’’ it mean that the difference between the ‘‘heaviest-loaded’’ node and the ‘‘lightest-loaded’’ node should be minimized

To achieve its goal of efficient resource sharing, efficient and effective resource allocation algorithms are essentials [6]. The efficiency of the resource allocation algorithms can be affected by many factors, such as the number of tasks arriving in the system, the available resources, the network characteristics, etc. To study the effectiveness of the resource allocation algorithms, there is a need for a formal model to represent the Grid resource allocation problem and to identify and analyze the impact of the different Grid parameters whose changes would have an impact on the efficiency of the algorithms [7]. This model is essential for comparing different algorithms. Having a formal model to compute the efficiency of a resource allocation algorithm is particularly important, as algorithms developers, such as researchers and students, may not have access to a real Grid for computing the efficiency of their algorithms.

Fault tolerance is an important property for large scale computational grid systems, where geographically distributed nodes co-operate to execute a task. In order to achieve high level of reliability and availability, the grid infrastructure should be a foolproof fault tolerant. Since the failure of resources affects job execution fatally, fault tolerance service is essential to satisfy QOS requirement in grid computing [8]. Commonly utilized techniques for providing fault tolerance are job check pointing and replication. Both techniques mitigate the amount of work lost due to changing system availability but can introduce significant runtime overhead.

The main problem of this research work is the resource allocation and fault tolerance over grid computing. There is several research and methodologies introduced but the fault tolerance is not achieved significantly. The existing approach has drawback with computational complexity and error rates. To overcome the abovementioned issues, in this research, Enhanced Bat Optimization (EBO) algorithm and Low Latency Fault Tolerance (LLFT) model is proposed to progress the overall grid system performance. The main contribution of this research is system model, load balancing using MMH algorithm, resource allocation using EBO algorithm and fault tolerance method using LLFT algorithm for better improvement of grid computing. The proposed method is used to increase the overall performance using effective algorithms over grid computing.

The rest of the work is organized as follows: a brief review of some of the literature works on load balancing, resource allocation, and fault tolerance methods on grid computing are presented in Section 2. The proposed methodology for MMH, EBO+LLFT model is detailed in Section 3. The experimental results and performance analysis discussion is provided in Section 4. Finally, the conclusions are summed up in Section 5

II.RELATED WORK

In [9], Patel et al (2019) discussed the main problems in managing resources are hardware and software failures, jobs management downtime, etc. To solve this, the PSO is introduced. The PSO algorithm is based on a simplified model of social behaviour exhibited by the buzzing behaviour of insects, birds and fish. In this research, a new algorithm is used for job scheduling, called Improved Particle Swarm Optimization (IPSO). The IPSO algorithm generates a velocity vector that is used to point out that the direction of swarm movement and particle position are updated. Therefore, it refines and improves the efficiency of the execution, the research capacity of the global research, the accuracy of the solution and guarantees the load balancing of the programming of the grid activities. Consequently, the work has been simulated with the help of the OptorSim simulator and it has been shown that algorithm provides an effective solution for planning the resources over grid scheduling network

In [10], Shah et al (2014) discussed that grid computing has enormous impact in the field of scientific research as well as in the realms of application world. The online presence of reliability and ease of flexible operations are the main features. The evolution of distributed computing with the virtualization of various forms of grids is the key to the understanding its utility. The utility management infrastructure consists of diverse nature of numerous virtualization in grid leading to functionality in web services core/hosting, work load and information. The virtualization layers are briefly described here. The

levels of virtualization can be explained enormously owing to the never ending demand of consumer world
In [11], Kfatheen et al (2015) introduced a new scheduling algorithm named as MiM-MaM based on renowned task scheduling algorithms, Min-Min and Max-Min. In the algorithm the drawbacks were rectified by using combined usage of two algorithms. The experimental outcome shows that the algorithm improved the Makespan. The need of effective and efficient scheduling algorithms is necessary to use the capabilities of large distributed systems optimally. For reducing overall completion time and enhancement of load balancing, numbers of algorithms are used.

In [12], Goyal et al (2012) presented Ant based algorithm to solve the load balancing problem in computational grid. In the algorithm, the pheromone is associated with resources, rather than path. The increase or decrease of pheromone represent load and depends on task status at resources. The main objective of algorithm is to map tasks to resources in a way that balance out the load resulting in improved utilization of resources. The simulation result concludes that the algorithm enhances performance in terms of resource utilization

In [13], Yang et al (2012) introduced a new nature-inspired metaheuristic optimization algorithm, called bat algorithm (BA), for solving engineering optimization tasks. BA is based on the echolocation behavior of bats. BA has been carefully implemented and carried out optimization for eight well-known optimization tasks; then a comparison has been made between the algorithm and other existing algorithms. The optimal solutions obtained by the algorithm are better than the best solutions obtained by the existing methods. The unique search features used in BA are analyzed

In [14], Li et al (2021) focused for computing offloading resource allocation strategy based on deep reinforcement learning in Internet of Vehicles. Firstly, the system architecture for Internet of Vehicles is designed; calculation model and communication model of computing offloading strategy are constructed. Then, the resource allocation problem in offloading process is studied for real-time energy-aware offloading scheme in mobile edge computing. Besides, considering the battery capacity of vehicle users, the remaining energy rate is utilized to redefine weighting factors to sense energy consumption in real time. Finally, with the shortest delay and smallest computational cost as optimization goals, Q-learning is used to achieve the optimization of offloading strategy, that is, the optimal allocation of communication and computing resources, and the best system security. The simulation results show that the delay of the algorithm is 0.442 s when the computational complexity is 9000 cycles/byte, and the performance of the delay is improved compared with the other algorithms.

In [15], Lee et al (2005) discussed the fault tolerance service to satisfy QoS requirement in computational grids. In order to provide fault tolerance service and satisfy QoS requirements, it expands the definition of failure, such as process failure, processor failure, and network failure. In grid computing, resource management and fault tolerance services are important issues. The availability of the selected resources for job execution is a primary factor that determines the computing performance. The failure occurrence of resources in the grid computing is higher than in a tradition parallel computing. Since the failure of resources affects job execution fatally, fault tolerance service is essential in computational grids. And grid services are often expected to meet some minimum levels of QoS for desirable operation. And it used resource scheduling service, fault detection service and fault management service and show implement and experiment results.

III.PROPOSED METHODOLOGY

In this work, load balancing is done by using Max-Min Heuristic (MMH) algorithm and resource allocation is performed by using Enhanced Bat Optimization (EBO) algorithm. Then Low Latency Fault Tolerance (LLFT) model is proposed to handle the fault tolerant problem which is focused to improve the grid computing performance significantly. The proposed work involves the formation of system model, load balancing, optimal resource allocation and fault tolerance system.

3.1 System model

In this work, consider the number of resources, number of tasks and number of grid users over the grid computing. A grid system is designed to complete a set of programs/applications, so that to complete certain tasks. Executing those programs need use some resources in the grid. The fundamental factors that constitute a resource allocation problem in the grid environment include: the task of grid users, grid resources, and the strategy of resource allocation. Grid resource allocation assigns grid task appropriate resources on the premise that user needs are satisfied, so that all the time the task runs is as small as possible. Users share the grid resources through submitting tasks to the grid system, and the grid resource allocation process re-allocates these tasks to the appropriate resources abide by some strategy, so the efficient allocation algorithm can take advantage of the processing power of the grid system to improve the throughput of the whole grid system.

Grid operators have the responsibility for managing grid services, which includes monitoring and responding to the wide range of power system events that may occur. At distribution system level, grid operators typically use a shared communication infrastructure with a certain QoS level provided by communication network operators via service level agreements in order to exchange information. It includes periodic task T_i is completely identified by a pair (C_i, T_i) where C_i is t_i's execution time, Virtual Machine (VM) and T_i is t_i's request period [16]. The request for T_i is periodic with the constant interval T_i between every two consecutive requests, and t_i's first request occurs at time 0. The worst case execution time for all the (infinite) requests of t_i is constant and equal to C_i. Given n independent periodic tasks $t_{1,}t_{2,\dots}t_n$ and a set of identical processors, the scheduling problem consists of finding an order in which all the periodic requests of the tasks are to be executed on the processors so as to satisfy the following scheduling conditions: integrity is preserved, that is, tasks and processors are sequential: each task is executed by at most one processor at a time and no processor executes more than one task at a time; deadlines are met, namely, each request of any task must be completely executed before the next request of the same task, that is, by the end of its period; the number m of processors is minimized. Fig 1 shows the overall block diagram of the proposed system

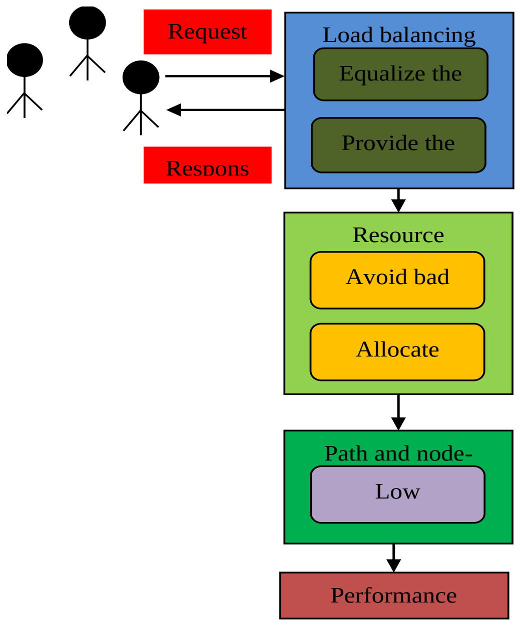

Fig 1 Overall block diagram of the proposed system

3.2Efficient load balancing using Max-Min Heuristic (MMH) algorithm

In this work, efficient load balancing is performed by using Max-Min Heuristic (MMH) algorithm. Load balancing involves distributing tasks among the available resources to ensure that no resource is underutilized or over-utilized. In grid computing load balancing resources are mainly data centers, physical machines, VM or any application software [17].

Need for load balancing in grid computing

In grid computing, tasks and other undertakings to the VM are referred to loads. These loads are categorized into three categories which are;

• Under-loaded
• Over-loaded
• Balanced

Load balancing algorithms are responsible for trying to equalize the total workloads in the grid computing environment. This is achieved by transferring tasks from over-loaded nodes to under-loaded nodes thus maximizing the system's throughput.

A load is the number of jobs in the waiting queue and can be light, moderate and heavy according to their work. Load balancing is a process of improving the performance of computational grid system in such a way that all the computing nodes involved in the grid are uniformly utilized as much as possible so that the throughput is improved, and the execution times are minimized. In dynamic load balancing the scheduling decisions are made when there is need to schedule the job for further processing. Load balancing is recognized as a means of providing satisfactory service performance for users while maximizing the availability and utilization of the whole grid system

The current load will be computed for all the resources shared in VM of grid servers. Once the load index is computed for all the resources, load balancing operation will be initiated to effectively use the resources dynamically with the process of assigning resources to the corresponding node to reduce the load value. So, assigning of resources to proper nodes is an optimal distribution problem so that Max-Min heuristic algorithm is introduced. A scheduling is designed by giving importance to balancing loads on nodes and it is focused to optimize the scheduling queue. In order to achieve the load balancing of grid data centers, it is necessary to choose the optimal physical host efficiently in the process of deploying tasks

In the grid computing environment, the grid resource scheduling controller, through the changing load of real-time monitor server's nodes, assigns resources to the corresponding node dynamically to meet some needs, such as high utilization of resources, excellent performance and fast response. The topology of the server group nodes that involved in the grid resources scheduling can be described in the following Fig 2.

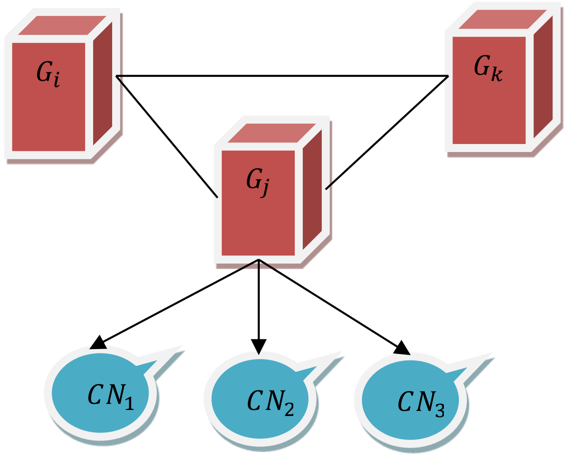

Fig 2 Simple grid system

The topology can be described as an $G\,(V,E)$ undirected graph, G represents the node set, E represents the set of connection between these nodes. C_i represents cluster control server node. CN_i represents node controller, which is a separate physical machine. Each C_i node manages m node controller N_i , you can run k VMs on each CN_i and use V_i to represent VM, So the node which will be mentioned in this study is V_i . In the grid service environment, application, database management system and other software are deployed in several V_i nodes to run.

In the grid computing system, when the user makes a request, the grid controller receives the request and then it will arrange the request in the queue of the server cluster. There have more than one cluster controller below the grid controller, cluster controllers real-time monitor the running load parameters of every virtual resource pool of this cluster, such as CPU occupancy rate, memory occupancy rate, the network bandwidth and process occupancy rate and so on, Therefore, this is a multi-objective problem which searches for the optimal solution. According to monitoring the multi-objective parameter, cluster controller will calculate the best value of each Vi node, then for the whole cluster, use Max-Min heuristic algorithm to search the best solution.

The proposed approach triggers a method for generating an effective load balancing in the grid system to schedule the nodes. Max-Min heuristic algorithm begins with a set made up of unmapped tasks. The scheduling process is preferred to the set of nodes with least amount of load possession. The algorithm later picks the tasks with the highest (maximum) make span and allocates it to the next available VM. When this task has been mapped to the best machine, it is emptied from the set G of tasks. The entire process is iterates until all the tasks have been emptied. Mapping a task with a maximum execution time to the high capacity machine ensures that the task can be executed concurrently with shorter execution times. Max-Min task scheduling algorithm helps in improving the make span. On the other hand, improved make span ensures a more balanced load across all the machines in the grid computing environment.

The nodes with least load are preferred for engaging extra process by the grid network. Each node possesses attributes defined by the scheduling process. The group of nodes is served by the grid system as per the request from the users to the server. As the request comes, the server fetches the node, which is free and severed to the user. This process is mapped on the basis of the nodes and according to their processing time a scheduling list is generated

Selection of Node with Minimum Load: The processes of selecting the node with least load are inspired from the MMH algorithm. The distance calculation between the nodes in the scheduling queues. Before proceeding to the calculation, it find the node with least load values. The node with least load is considered for the queue to calculate least distinct nodes.

The distance values of the node are calculated based on the Cartesian distance, which is given by:

$$dist = \sqrt{\sum_{j=1}^{k}(n_i - n_j)^2} \quad (1)$$

The distance is presented using the expression Dist and the n_i is the selected node and n_j is the comparing node. Once all the distance values have been calculated between the node values, the nodes are rearranged according to the least distinct node to the pivot node. The top queue in the schedule list is considered as the most relevant scheduling queue. Finally, according to grid load balancing, service resource will be assigned to the best server node. It is help them make intelligent decisions on whether any task needs to be assigned to a low-capacity virtual machine or rather wait for a high capacity machine running. MMH also can schedule all the tasks in the waiting mode more efficiently

Assignment of tasks to resources using these heuristic algorithms mainly ensures that all the tasks are executed quickly, thus minimizing the total make span of the virtual machine in use. The max-min algorithm mainly aims to reduce the waiting time for larger tasks and assign them to virtual machines' higher capacity. After all the large tasks have been assigned to their respective devices, the small tasks are assigned to either idle or underutilized machines. This ensures that all the machines are used evenly, reducing the chances of machines being either overloaded with tasks or being underutilized

Algorithm 1: MMH algorithm for load balancing

```
1.  Start
2.  For all the tasks in the set G; Ti
3.  For all the resources ; Rj
4.  Cij=Eij+rj
5.  While the tasks set G has tasks
6.  Select a task Tk with the highest completion time.
7.  Find the distance between nodes using (1)
8.  sort the resources in the order of completion time
9.  Allocate that task to a resource (Rj), which will
    give the minimum execution time
10. Empty the task Tk from set U.
11. Repeat steps 1 to 7 until the set U is empty
12. Equalize the loads in grid
13. End
```

From the above pseudocode, r_j represents the resource R_j which is ready for executing a task. C_{ij} represent the expected completion time and E_{ij} the execution time [18]. It maximizes the utilization of the VM and the CPU. Thus the MMH algorithm provides equalization loads to the grid system efficiently and also it provides more balanced load across all the machines in the grid computing considerably.

3.3 Resource allocation using Enhanced Bat optimization (EBO) algorithm

In this work, Enhanced Bat Optimization (EBO) algorithm is applied for optimal resource allocation. Based on the activities of the bats, the fitness values are computed to decrease the computational complexity effectively for the given grid environment. It is built to perform as a crew of bats finding their prey or foods utilizing their ability of echolocation.

Bats use ultrasound [19] to detect optimal resources over the grid environment. In the process of predation, bats can find food by adjusting flight route based on the current resource information with three major categories of the task information. The bats in the feasible region update flight speeds and adjust positions according to their own status and the optimal individual bat position of resources in the grid setting; if the current position of the bat is superior to the position then selected resources are updated to allocation, it will keep its own position. Else, the bat will determine whether to fly to a position near the optimal selected resources are according to the roulette gambling strategy. Each bat uses the change of the acoustic pulse frequency fre_i to detect the gap between its own resource position and the resources position of the food, and then uses it to adjust the speed and direction.

The location of the virtual bats is provided by updating their velocity and position using Equations 2, 3 and 4, as follows:

$$fi = fmin + (fmin - fmax)\beta \quad (2)$$

$$v_i^j = v_i^j(t-1) + [\hat{x}^j - x_i^j(t-1)fi \quad (3)$$

$$x_i^j(t) = x_i^j(t-1) + v_i^j(t) \quad (4)$$

Here β indicates arbitrarily produced number among the interval [0, 1]. $x_i^j(t)$ indicates the value of decision variable j for bat i at time step t. It means $i\ and\ j$ resources computation in time period$t$. The product of fi is utilized to manage the pace and range of the movement of the bats. The variable $\hat{x}^j$ represents the current global best location (solution) for decision variable j, which is obtained by comparing all the solutions given by the $m$bats. To model this algorithm, some rules are idealized as follows:

1. All bats employ echolocation to detect distance, and they as well “know” the dissimilarity between food/prey and environmental obstacles in some miraculous way;
2. A bat bi fly arbitrarily with velocity vi at position xi with a fixed frequency $f$min, varying wavelength λ and loudness $A0$ to search for prey [20]. They can adjust the wavelength (or frequency) of their emitted pulses and fine-tune the rate of pulse emission $r\in$ [0, 1] automatically, based on the closeness of their target;
3. Even though the volume can vary in many ways, it is considered that the loudness varies from a large (positive) $A0$ to a minimum constant value Ami

By using bat optimization, the best resources are selected as follows:

Algorithm: 2 EBO for resource allocation

Objective function (x), x = ($x1$, ..., xn).

Initialize the bat population xi and vi, i = 1, 2, ..., m.

Define pulse frequency fi at xi, ∀i = 1, 2,...,.

Initialize pulse rates ri and the loudness Ai, i = 1, 2,...,.

```
1.  Begin
2.  While $t$<T
3.  For each bat $bi$, do
4.  Generate new solutions using eq (2), (3) and (4)
5.  If rand>ri, then
6.  Find the resources using objective function values
7.  Select a solution among the best solutions
    (resources)
8.  Generate the available resources around the local
    resources
9.  If rand<Ai and f(xi)<f(x’) then
10. Accept the new solutions
11. Increase ri and reduce Ai
12. Rank the bats and find the current best x’
13. Return the optimal resources
```

Initially, the first position xi, velocity vi and frequency fi are initialized for each bat bi. For each time step t, being T the maximum number of iterations. By utilizing objective function, the finest fitness value is calculated optimally. One solution is chosen among the available top solutions, and then to produce a fresh solution for every bat that meets the condition

$$x_{new} = x_{old} + \epsilon \bar{A}(t) \qquad (5)$$

in which $\bar{A}(t)$ stands for the average loudness of all the bats at time t, and $\epsilon\in$ [−1, 1] attempts to the direction and strength of the arbitrary walk.

$$\text{Ai(t+1)}=\alpha\text{Ai(t)} \qquad (6)$$

For every iteration of the algorithm, the loudness Ai and the emission pulse rate r_i are updated [21].

3.4 Fault tolerance using Low Latency Fault Tolerance (LLFT) Model

Fault tolerance is the ability of a system to perform its function correctly even in the presence of faults. The fault tolerance makes the system more dependable [22]. LLFT model provides fault tolerance for distributed applications deployed within a grid computing using the leader/follower replication approach. The LLFT middleware consists of a low latency messaging protocol, a leader-determined membership protocol, and a virtual determinizer framework. The messaging protocol provides a reliable, totally ordered message delivery service by employing a direct group-to-group multicast where the ordering is determined by the primary replica in the group. The membership protocol provides a fast reconfiguration and recovery service when a replica becomes faulty and when a replica joins or leaves a group. The virtual determinate framework captures ordering information at the primary replica and enforces the same ordering at the backup replicas for major sources of no determinism. The LLFT middleware maintains strong replica consistency, offers application transparency, and achieves low end-to-end latency

The LLFT middleware maintains strong replica consistency of the states of the replicas, both in fault-free operation and in the event of a fault [23]. If a fault occurs, the LLFT reconfiguration/recovery mechanisms ensure that a backup has, or can obtain, the messages and the ordering information it needs to reproduce the actions of the primary. They transfer the state from an existing replica to a new replica and synchronize the operation of the new replica with the existing replicas, to maintain virtual synchrony. The LLFT middleware replicates application processes to protect the application against various types of faults. The steps of LLFT model is as follows:

- *Thread identifier* - The identifier of the thread that is carrying out the operation
- *Operation identifier* - An identifier that represents one or more data items that might change during the operation or on completion of the operation
- *Operation count* - The number of operations carried out by a thread for a given operation identifier
- *Operation metadata* - The data returned from the system/library call. This metadata includes the out parameters (if any), the return value of the call, and the error code (if necessary)

At the primary, the algorithm maintains a queue, the OrderInfo queue of four-tuples (T, O, N, D), where thread T has executed a call with operation identifier O and with metadata recorded in D, and this is the Nth time in its execution sequence that thread T has executed a nondeterministic call of interest. The entries are transmitted to the backups reliably and in first in first out order, using the piggybacking mechanism of the LLFT messaging protocol. At a backup, for each operation O, the algorithm maintains an O.OrderInfo queue of three-tuples (T, N, D), in the order in which the primary created them. After an entry is appended to the queue, the algorithm awakens an application thread if it is blocked. At a backup, when thread T tries to execute the operation O as its Nth execution in the sequence, if (T, N, D) is not the first entry in the O.OrderInfo queue, the algorithm suspends the calling thread T. It resumes a thread T that was suspended in the order in which (T, N, D) occurs in the O.OrderInfo queue, rather than in the order in which the thread was suspended or in an order determined by the operating system scheduler. It removes an entry (T, N, D) from the O.OrderInfo queue immediately before it returns control to the calling thread T after its Nth execution in the sequence. The algorithm requires the ordering of all related operations, such as claims and releases of mutexes.

At a backup, when it is time to process a mutex ordering information entry, the mechanisms examine the metadata. If the metadata contains an error code, they return control to the calling thread with an identical error status, without performing the call. Otherwise, they delegate the mutex claim operation to the original library call provided by the operating system. If the mutex is not currently held by another thread, the calling thread acquires the mutex immediately. Otherwise, the calling thread is suspended and subsequently resumed by the operating system when the thread that owned the mutex releases it. The Consistent Multi-Threading Service allows concurrency of threads that do not simultaneously acquire the same mutex, while maintaining strong replica consistency. Thus, it achieves the maximum degree of concurrency possible, while maintaining strong replica consistency. Also the security level of communications and enables authentication [24] and access control scheme that is implemented on a multi-layer communication architecture

IV.SIMULATION RESULT

In order to verify the effectiveness of grid resource allocation and load balancing algorithm based on optimized target decision, a comparative test is carried out. Firstly, resource allocation decision planning model is established for a certain same resource requirement. Compared with the efficiency and accuracy of resource allocation management in the resource allocation management method, in order to ensure the accuracy of the experiment before the experiment detection, the experiment detection parameters are first obtained and determined [25], and the specific parameter information is shown in the following Table 1

Table 1 Experimental Test Parameters

Resource capacitymb	allocation capabilities %	memory GB	Maximum detection time（s）
1300	35	110	18
1200	50	140	26
1600	78	220	38
1400	64	210	20
1500	75	156	60
1300	48	230	57
1200	65	170	40

In this research work, performance of the proposed EBO+LLFT algorithm is compared with the existing PSO–GELS, Multi-Task Target Decision (MTTD), Adaptive Particle Swarm Optimization + Fit First (APSO+FF) and Improved Ant Colony Optimization

+ Fit First (IACO+FF) approaches in terms of accuracy, time complexity, cost complexity and error rate. Table 2 shows the comparison metrics of existing and proposed system.

Methods/Metrics	PSO–GELS	MTTD	APSO+FF	IACO+FF	EBO+LLFT
Accuracy (%)	87.04	91.03	93.08	94.34	96.87
Error rate (%)	23.76	20.47	16.13	11.01	8.64
Time complexity (sec)	45	38	29	18	9.31
Cost complexity (GB)	0.13	0.1	0.08	0.04	0.014

Table 2 Comparison metrics of existing and proposed system

1. Accuracy

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

$$\text{Accuracy} = \frac{(\text{TruePositive} + \text{TrueNegative})}{(\text{TruePositive} + \text{FalseNegative} + \text{FalsePositive} + \text{TrueNegative})} \quad (7)$$

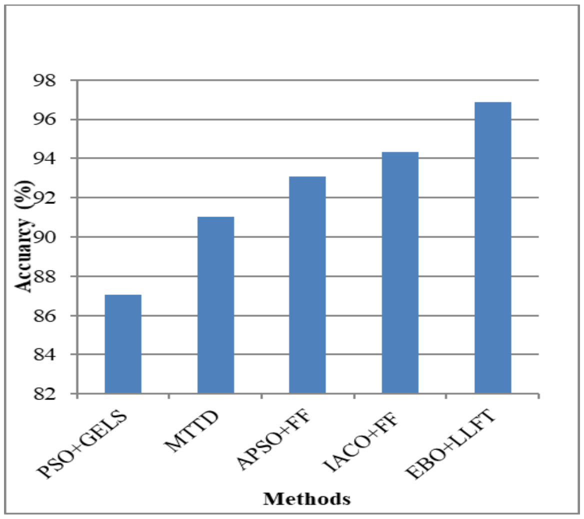

Fig 3 Accuracy comparison

From the above Fig 3, it can be observed that the comparison metric is evaluated using existing and proposed method in terms of accuracy. For x-axis the methods are taken and in y-axis the accuracy value is plotted. The existing methods are such as PSO–GELS, MTTD, APSO+FF and IACO+FF algorithms provide lower accuracy whereas proposed EBO+LLFT algorithm provides higher accuracy. In this proposed research work, optimal resources are selected by using EBO algorithm via best fitness function values. The loads are balanced equally hence the tasks are performed optimally in grid computing. The fault tolerance system is performed using LLFT model which increases the efficiency hence it maximizes the overall grid computing performance. Thus the result concludes that the proposed EBO+LLFT algorithm increase the accuracy of resource allocation over grid computing

2. Error rate

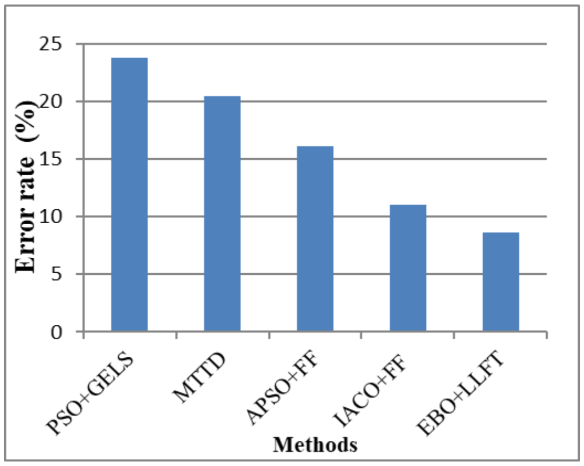

Fig 4 Error rate comparison

From the above Fig 4, it can be observed that the comparison metric is evaluated using existing and proposed method in terms of error rate. For x-axis the methods are taken and in y-axis the error rate value is plotted. The existing methods are such as PSO–GELS, MTTD, APSO+FF and IACO+FF algorithms provide higher error rate whereas proposed EBO+LLFT algorithm provides lower error rate. MMH algorithm provides equalization loads to the grid system efficiently and also it provides more balanced load across all the machines in the grid computing. In this proposed research work, optimal resources are selected by using EBO algorithm via best fitness function values. LLFT model provides fault tolerance for distributed applications deployed within a grid computing using the leader/follower replication. Hence it reduces the failure rates effectively and it maximizes the overall grid computing performance. Thus the result concludes that the proposed EBO+LLFT algorithm increase the accuracy of resource allocation over grid computing

3. Time complexity

The system is better when the proposed algorithm executes in less time consumption.

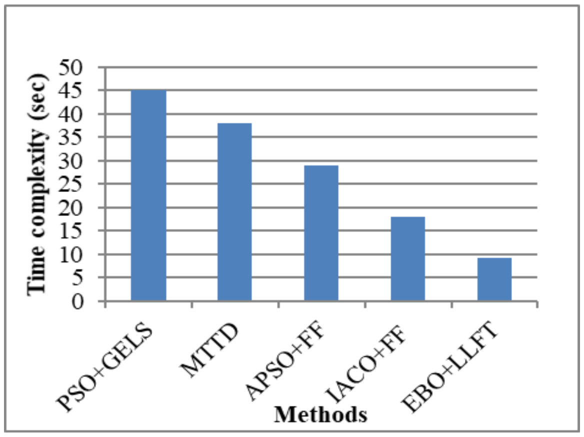

Fig 5 Time complexity

From the above Fig 5, it can be observed that the comparison metric is evaluated using existing and proposed method in terms of time complexity. For x-axis the methods are taken and in y-axis the time complexity value is plotted. The existing methods are such as PSO–GELS, MTTD, APSO+FF and IACO+FF algorithms provide higher time complexity whereas proposed EBO+LLFT algorithm provides lower time complexity. The MMH ensures that all the tasks are executed quickly, thus minimizing the total make span of the virtual machine in use. The max-min algorithm mainly aims to reduce the waiting time for larger tasks and assign them to virtual machines' higher capacity. In this proposed research work, optimal resources are selected by using EBO algorithm via best fitness function values. The fault tolerance is ensured by using LLFT model which handles various types of failures effectively. Thus the result concludes that the proposed EBO+LLFT algorithm increase the accuracy of resource allocation over grid computing

4. Cost complexity

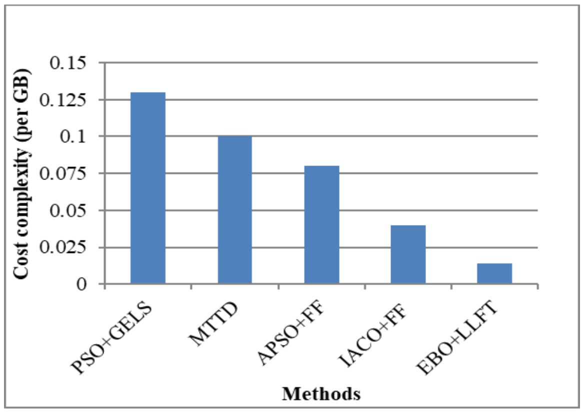

Fig 6 Cost complexity

From the above Fig 6, it can be observed that the comparison metric is evaluated using existing and proposed method in terms of cost complexity. For x-axis the methods are taken and in y-axis the cost complexity value is plotted. The existing methods are such as PSO–GELS, MTTD, APSO+FF and IACO+FF algorithms provide higher cost complexity whereas proposed EBO+LLFT algorithm provides lower cost complexity. Load balancing is improving the performance of computational grid system in such a way that all the computing nodes involved in the grid are uniformly utilized as much as possible. In this proposed research work, optimal resources are selected by using EBO algorithm which reduces the cost complexity. The LLFT model maintains strong replica consistency of the states of the replicas, both in fault-free operation and in the event of a fault. Thus the result concludes that the proposed EBO+LLFT algorithm improve the accuracy and cost complexity of resource allocation over grid computing considerably

V.CONCLUSION

In this work, Enhanced Bat Optimization (EBO) algorithm and Low Latency Fault Tolerance (LLFT) model is proposed to improve the resource allocation and fault tolerance performance. Initially the system model is constructed with numbers of resources, users, tasks and servers over the grid computing. Then the load balancing is done by using MMH algorithm which is focused to equalize the total workloads over grid computing environment. The resource allocation is done using EBO algorithm is used to produce optimal solutions by improving the time and cost complexity metrics. Then, LLFT model is proposed to handle strong replica consistency of the states of the replicas, both in fault-free operation and in the event of a fault. The results proves that the proposed EBO+LLFT algorithm provides higher performance by means of greater accuracy, lower error rate, cost complexity and time complexity than the existing IACO+FF, APSO+FF, PSO–GELS and MTTD algorithms. In the future work, the effective soft computing and machine learning algorithm can be developed to deal with various attacks prominently

REFERENCES

[1] Patni, Jagdish Chandra. "Centralized Approach of Load Balancing in Homogenous Grid Computing Environment." *Proceedings of the 2020 the 3rd International Conference on Computers in Management and Business*. 2020.

[2] Khoo, BT Benjamin, et al. "A multi-dimensional scheduling scheme in a grid computing environment." *Journal of Parallel and Distributed Computing* 67.6 (2007): 659-673.

[3] Mahato, Dharmendra Prasad, et al. "On scheduling transaction in grid computing using cuckoo search-ant colony optimization considering load." *Cluster Computing* 23.2 (2020): 1483-1504.

[4] Viswanathan, Hariharasudhan, Eun Kyung Lee, and Dario Pompili. "Mobile grid computing for data-and patient-centric ubiquitous healthcare." *2012 The First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT)*. IEEE, 2012.

[5] Rathore, Neeraj. "Performance of hybrid load balancing algorithm in distributed web server system." *Wireless Personal Communications* 101.3 (2018): 1233-1246.

[6] Aggarwal, Anuj, Rajesh Verma, and Ajit Singh. "An efficient approach for resource allocations using hybrid scheduling and optimization in distributed system." *Int. J. Educ. Manag. Eng.(IJEME)* 8.3 (2018): 33-42.

[7] Mohanty, Prasant, et al. "Dynamic resource allocation in Vehicular cloud computing systems using game theoretic based algorithm." *2018 fifth international conference on parallel, distributed and grid computing (PDGC)*. IEEE, 2018.

[8] Balasangameshwara, Jasma, and Nedunchezhian Raju. "A hybrid policy for fault tolerant load balancing in grid computing environments." *Journal of Network and computer Applications* 35.1 (2012): 412-422.

[9] Patel, Surendra Kumar, and Anil Kumar Sharma. "Improved PSO based job scheduling algorithm for resource management in grid computing." *International Journal of Advanced Technology and Engineering Exploration* 6.54 (2019): 152-161.

[10] Shah, Y. K., M. Jukaria, and S. Shah. "Formation and design considerations of grid architecture." Int. J Comp Sci. Emerging Tech 5.4 (2014): 169-176

[11] Kfatheen, S. Vaaheedha, and M. Nazreen Banu. "MiM-MaM: A new task scheduling algorithm for grid environment." *2015 International Conference on Advances in Computer Engineering and Applications*. IEEE, 2015

[12] Goyal, Sandip Kumar, and Manpreet Singh. "Adaptive and dynamic load balancing in grid using ant colony optimization." *International Journal of Engineering and Technology* 4.4 (2012): 167-174.

[13] Yang, Xin-She, and Amir Hossein Gandomi. "Bat algorithm: a novel approach for global engineering optimization." *Engineering computations* (2012).

[14] Li, Xuezhu. "A computing offloading resource allocation scheme using deep reinforcement learning in mobile edge computing systems." *Journal of Grid Computing* 19.3 (2021): 1-12.

[15] Lee, HwaMin, et al. "A resource management and fault tolerance services in grid computing." *Journal of Parallel and Distributed Computing* 65.11 (2005): 1305-1317.

[16] Yagoubi, Belabbas, and Yahya Slimani. "Dynamic load balancing strategy for grid computing." *Transactions on Engineering, Computing and Technology* 13.2006 (2006): 260-265

[17] Maipan-Uku, J. Y., et al. "Max-average: An extended max-min scheduling algorithm for grid computing environtment." *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 8.6 (2016): 43-47.

[18] Xaphakdy, Khampaserth, et al. "Resource optimization in heterogeneous networks using discrete firefly algorithm." *2020 International Conference on Smart Technology and Applications (ICoSTA)*. IEEE, 2020

[19] Rani, A. Sylvia Selva, and R. R. Rajalaxmi. "Unsupervised feature selection using binary bat algorithm." *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*. IEEE, 2015.