RESEARCH ARTICLE OPEN ACCESS

Hybrid Approach to Minimize Makespan and Flowtime in Grid Environment

Malkeet Kaur [1], Mr. Karan Mahajan [2]
Research Scholar of M. Tech [1], Assistant Professor [2]
Department of Computer Science and Engineering
Global Institute of Management & Emerging Technologies, Amritsar
Punjab, India

ABSTRACT

Jobs are executed in a multitasking system on the basis of processor scheduling, bandwidth scheduling. Job scheduling in parallel processing use different types of algorithms and techniques which are used to reduce the number of delayed jobs. There are different kind of scheduling algorithms and techniques used to reduce the execution time of tasks. We can execute our jobs using sequential or parallel job scheduling method. In sequential scheduling method we can use FCFS, SJF, and Priority algorithm to schedule our jobs but in case of parallel scheduling we can use honey bee, genetic algorithm, ant colony algorithms to resolve the problems of parallel job scheduling. Ant colony algorithm can be used when the distance between resource and job is within 14km. but if our resource is more away from the 14km distance then we can use honey bee algorithms. In our proposed system we will hybrid ant colony and honey bee algorithm to implement job scheduling as it improves the speed to 35% to improve the scheduling. By using this technique the overall throughput and performance of the system will be improved by using this immense method.

Keywords:- Job Scheduling, Parallel Scheduling, Ant Colony, Honey Bee, Hybrid

I. INTRODUCTION

Job scheduling is a process of allocating system resources to different tasks by using operating system. [1]The system handles prioritized job queues that wait for CPU time and it should also determine which job to be executed first from list of jobs. By using the above criteria jobs can be executed in a fair manner.

Job scheduling is performed by using job schedulers. Job schedulers are programs which enable scheduling and, at times, track computer "batch" jobs, or units of work as like the operation of a payroll program. Job schedulers have the ability that they can start and control jobs automatically by running prepared job-control-language statements or by means of similar communication with a human operator. Generally, the present-day job schedulers include a graphical user interface (GUI) along with a single point of control.

[2]Parallel processing is used in job scheduling due to some reasons, i.e. it provide concurrency, save time, solve larger problems, maximize load balancing and make a good use of parallel hardware architecture. In multiprocessor environment parallel processing has two kinds of processors heterogeneous and homogeneous, in heterogeneous the processors are of different kind of speed and cost while in homogenous there are same kind of processors in all perspectives.

1.1 Types of Scheduling

A. *Long-term Scheduling*

[3]Long term scheduling is performed when a new process is created. It is shown in the figure below. If the number of ready processes in the ready queue becomes very high, then there is a overhead on the operating system i.e. processor for maintaining long lists, context switching and dispatching increases. Therefore, allow only limited number of processes in to the ready queue.

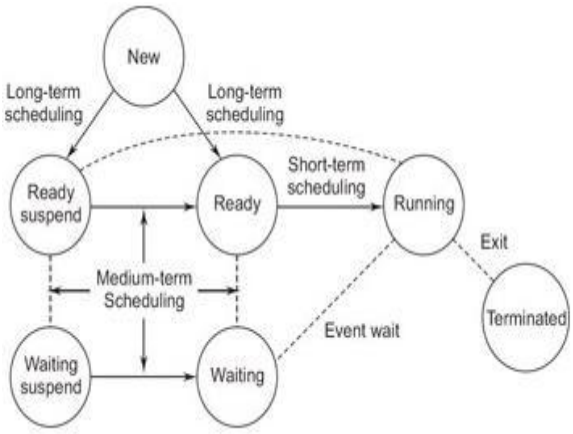

Figure 1: Shows types of scheduling

The long-term scheduler limits the number of processes to allow for processing by taking the decision to add one or more new jobs, based on FCFS (First-Come, first-serve) basis or priority or execution time or Input/output requirements. Long-term scheduler executes relatively infrequently.

B. *Medium-term Scheduling*

Medium-term scheduling is a part of the swapping function. When part of the main memory gets freed, the operating system looks at the list of suspend ready processes, decides which one is to be swapped in (depending on priority, memory and other resources required). This scheduler works in close conjunction with the long-term scheduler. It will perform the swapping-in function among the swapped-out processes. Medium-term scheduler executes somewhat more frequently.

C. *Short-term Scheduling*

Short-term scheduler is also called as dispatcher. Short-term scheduler is invoked whenever an event occurs, that may lead to the interruption of the current running process. For example clock interrupts, I/O interrupts, operating system calls, signals, etc. Short-term scheduler executes most frequently. It selects from among the processes that are ready to execute and allocates the CPU to one of them. It must select a new process for the CPU frequently. It must be very fast

II. LITERATURE SURVEY

In the proposed technique we accomplish a way through which overhead during the load enhancement can be reduced. The processor that used within the parallel computation could be of similar types of dissimilar types. The similar type of processor used within the network form array processors and dissimilar types of processor within the network are known as vector processors. The processors can be used within the network executing instructions either in serial or parallel manner. The work has been diverted from humans to machines. The human workload has been diverted because that much load cannot be [1][4]accommodated on the single user. Same is the case with the machines. Now days cloud is used most often hence the load on the cloud system is increasing. The cloud will use data centers and with the increase in number of users the load on data centers is increasing. In that case if the data center goes down then all the data stored over the VMs(Virtual Machine) will be lost. The proposed paper will suggest the technique for offloading the data to multiple data centers. In the proposed paper migration will be performed by the use of live VM migration which means that the migration does not required to switch off the devices which is the case in offline migration.

Offloading is an effective method for extending the lifetime of handheld mobile devices by executing some components of applications remotely (e.g., on the server in a data center or in a cloud).[2][5] To achieve energy saving while satisfying given application execution time requirement, we present a dynamic offloading algorithm, which is based on Lyapunov optimization. The procedure elaborated has low complexity to solve the offloading problem (i.e., to determine which software components to execute remotely given available wireless network connectivity). The offloading in mobile computing will serve as mechanism for storing more data within the mobile device than its capacity. The mobile computing offloading becomes critical in a situation where larger applications are need of the hour which is to be executed on the mobile system.

Computation Offloading, sending computational tasks to more resourceful servers, is becoming a widely-used approach to save limited resources on mobile devices like battery life, storage, processor, etc. [3][6]Given an application that is partitioned into multiple tasks, the offloading decisions can be made on each of them. However, considering the delay constraint and the extra costs on data transmission and remote computation, it is not trivial to make optimized decisions. Existing works have formulated offloading decision problems as either graph-partitioning or binary integer programming problems. The first approach can solve the problem in polynomial time but is not applicable to delay constraints. The second approach relies on an integer programming solver without a polynomial time guarantee. We provide an algorithm, DTP (Deterministic delay constrained Task Partitioning), to solve the offloading decision problem with delay constraints. DTP gives near-optimal solution and runs in polynomial time in the number of tasks.[4][7] Going beyond prior work on linear delay constraints that apply only to serial tasks, we

generalize the delay constraints to settings where the dependency between tasks can be described by a tree. Furthermore, we provide another algorithm, PTP (Probabilistic delay constrained Task Partitioning), which gives stronger QoS guarantees. Simulation results show that our algorithms are accurate and robust, and scale well with the number of tasks.

III. PROPOSED SYSTEM

A. ANT COLONY OPTIMIZATION

[8] [9]In natural world, ants of some species (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but instead to follow the trail, returning and reinforcing it if they eventually find food. Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. Increase evaporation also has the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the Inspection of the solution space would be constrained. The impact of pheromone evaporation in real ant systems is unclear, but it is very important in artificial systems.

The overall result is that when one ant finds a good path from the colony to a food source, other ants are more likely to follow that path, and positive feedback ultimately leads to all the ants following a single path. The idea of the ant colony algorithm is to mime this behaviour with simulated ants walking around the graph representing the problem to solve.

B. HONEY BEE OPTIMIZTION

[10] [11] A colony of honey bees can extend itself over long distances (over 14 km) and in multiple directions simultaneously to harvest pollen from multiple food sources (flower patches). A small fraction of the colony constantly searches the environment looking for new flower patches. These recruiter bees move randomly in the area surrounding the hive, evaluating the profitability (net energy yield) of the food sources encountered. When they return to the hive, the scouts deposit the food harvested. Those individuals that found a highly profitable food source go to an area in the hive called the “dance floor”, and perform a ritual known as the waggle dance Through the waggle dance a scout bee communicates the location of its discovery to idle onlookers, which join in the exploitation of the flower patch. Since the length of the dance is proportional to the scout’s rating of the food source, more foragers get recruited to harvest the best rated flower patches. After dancing, the scout returns to the food source it discovered to collect more food. As long as they calculate as profitable, rich food sources will be advertised by the scouts when they return to the hive. Enlisted foragers may waggle dance as well, increasing the recruitment for highly rewarding flower patches. Thanks to this autocatalytic process, the bee colony is able to quickly switch the focus of the foraging effort on the most profitable flower patches.

The collaboration of both of these algorithms can form hybrid approach for better Makespan and Flowtime than individual approach.

Methodology used in proposed System

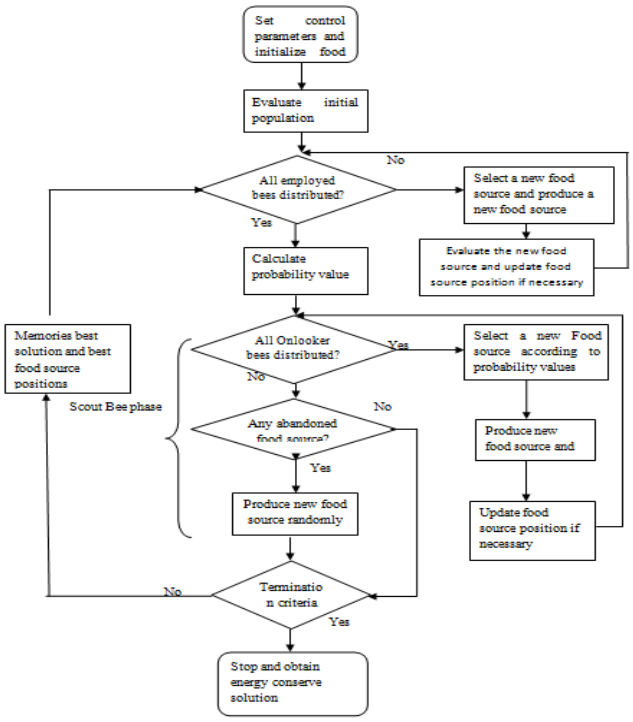

Figure 2: Flow Chart of proposed System

Simulation is carried in NETBEANS and obtained results are better as compared to existing system. Obtained results are described in this section.

When the simulation is carried out following chart is obtained from the JFREECHART simulator present within Netbeans

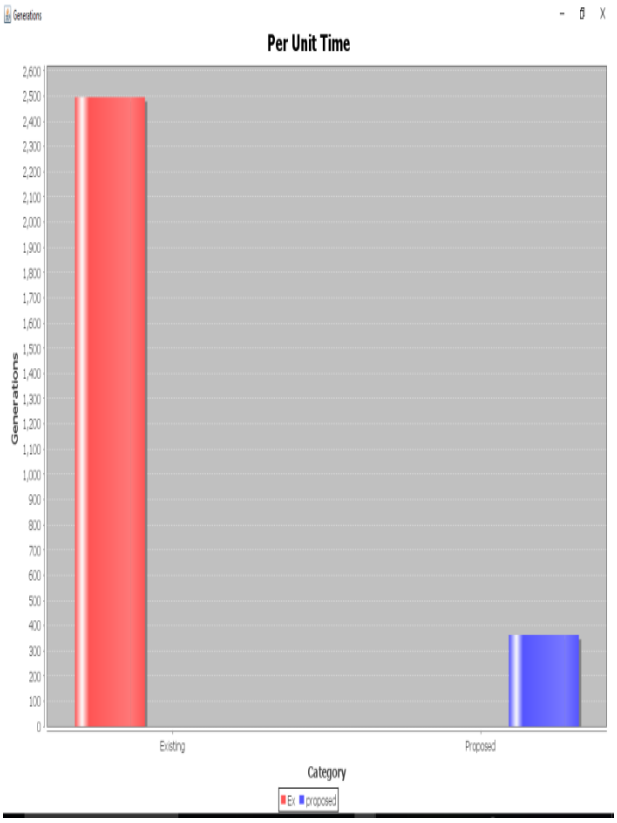

Figure 3: Make span of proposed and existing system

Make span is time the taken to execute entire jobs. The above graph shows the Make span of the proposed and existing system. The proposed system reduces the Make span of the job. The Make span in tabular format is shown as under

Existing_Makespan	Proposed_Makespan
2089	656

Table 1: Shows make span of proposed and existing system

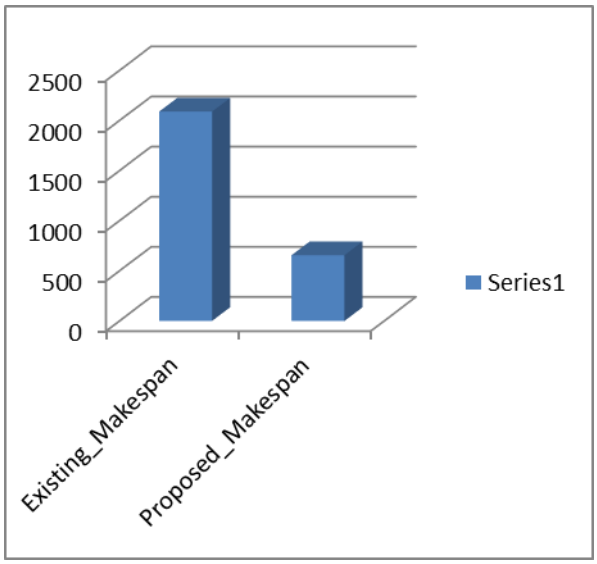

Figure 4: Tabular format of Makespan of proposed and existing system

Flowtime is the time taken to execute single job. This is given as under

Existing_Flowtime	Pro-Flowtime
213	62

Table 2: Shows flow time of proposed and existing system

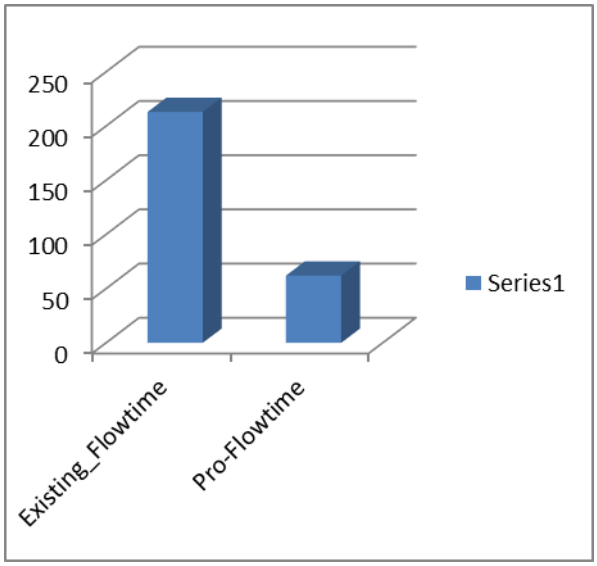

Figure 5: Flow time of proposed and existing system

Latency is the delay from input into a system to desired outcome and also vary from one system to another. It is given as under

Ex-Latency	Pro-Latency
65	54

Table 3: Shows latency in proposed and existing system

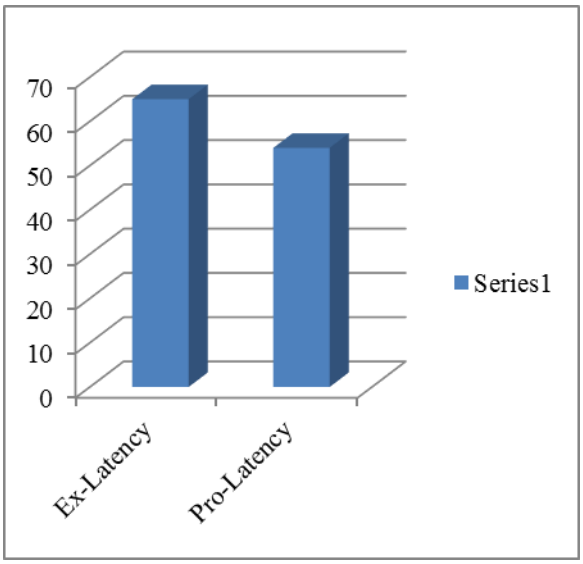

Figure 6: Latency of proposed and existing system

The approach followed is better than existing approach which is proved as simulation is executed multiple times.

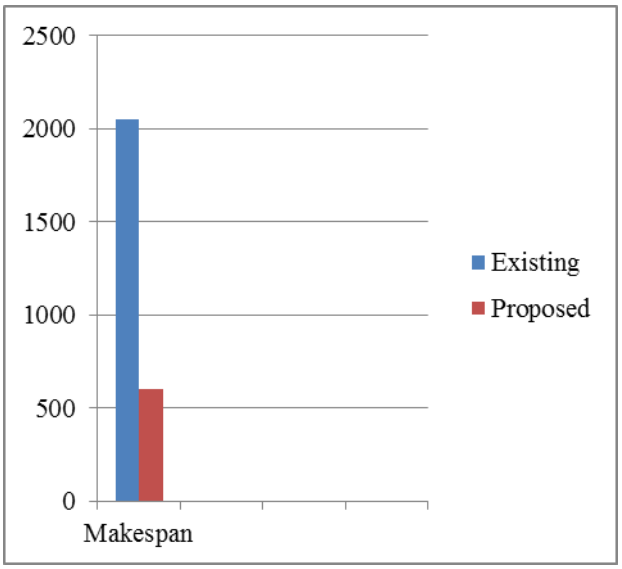

Figure 7: Shows the Makespan of existing and proposed approach

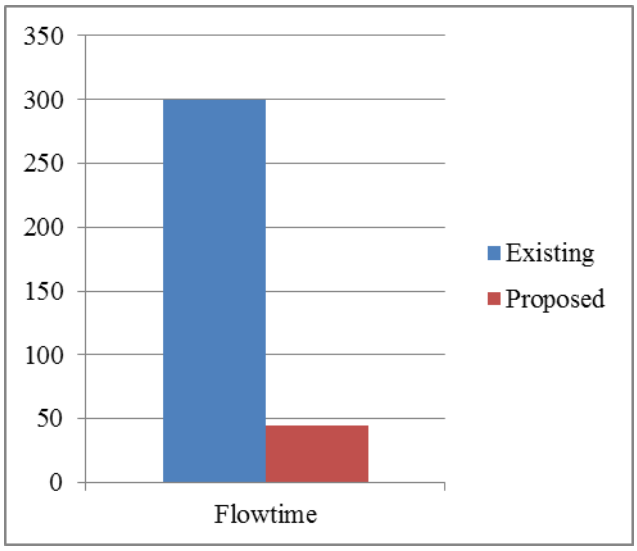

Figure 7: Shows Flow time of existing and proposed system.

Our result indicates that result of proposed approach is better as compared to existing approach.

IV. CONCLUSION

The proposed system uses hybrid approach of ant and honey bee algorithms. The optimized behavior of both the approaches will be considered in this case. Ant colony optimization behaves well as the distance is less and Honey bee is useful as distance increases. The hybrid approach by using simulation produces better result.

REFERENCES

[1] J. H. Abawajy, "Job scheduling policy for high throughput computing environments," in *Ninth International Conference on Parallel and Distributed Systems, 2002. Proceedings.*, pp. 605–610.

[2] H. D. Karatza and R. C. Hilzer, "Parallel Job Scheduling in Homogeneous Distributed Systems," *Simul. Trans. Soc. Model. Simul.*, vol. 79, no. 5, pp. 287–298, 2003.

[3] H. B. Prajapati and V. A. Shah, "Scheduling in Grid Computing Environment," *2014 Fourth Int. Conf. Adv. Comput. Commun. Technol.*, pp. 315–324, 2014.

[4] L. Zuo, L. E. I. Shu, and S. Dong, "A Multi-Objective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing," vol. 3, 2015.

[5] X. Chen, "Decentralized Computation Offloading Game for Mobile Cloud Computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[6] D. Niyato, "A Dynamic Offloading Algorithm for Mobile Computing," *IEEE Trans. Wirel. Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[7] Y.-H. Kao and B. Krishnamachari, "Optimizing mobile computational offloading with delay constraints," in *2014 IEEE Global Communications Conference*, 2014, pp. 2289–2294.

[8] D. Maruthanayagam and R. UmaRani, "Enhanced Ant Colony Algorithm for Grid Scheduling," *Int. J. Comput. Technol. Appl.*, vol. 1, no. 1, pp. 43–53, 2010.

[9] M. T. Aftab, M. Umer, and R. Ahmad, "Jobs Scheduling and Worker Assignment Problem to Minimize Makespan using Ant Colony Optimization Metaheuristic," vol. 6, no. 12, pp. 2823–2826, 2012.

[10] B. Yuce, M. S. Packianather, E. Mastrocinque, D. T. Pham, and A. Lambiase, "Honey bees inspired optimization method: The bees algorithm," *Insects*, vol. 4, no. 4, pp. 646–662, 2013.

[11] D. T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi, "The Bees Algorithm - A Novel Tool for Complex Optimisation Problems," *Intell. Prod. Mach. Syst. - 2nd I*PROMS Virtual Int. Conf. 3-14 July 2006*, pp. 454–459, 2006.