International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 2 Issue: 8 2384 – 2389

Parallel Asynchronous Particle Swarm Optimization For Job Scheduling In Grid Environment

Mr.S.Umarani
Department of Information Technology,
Shree Venkateshwara Hi Tech Engineering College, Gobi,
Tamilnadu, India.
umaranisks@gmail.com

Dr.T.Senthilprakash
Department of Computer Science and Engineering
Shree Venkateshwara Hi Tech Engineering Collge, Gobi,
Tamilnadu, India.
jtyesp@yahoo.co.in

Abstract-Grid computing is a new, large and powerful self managing virtual computer out of large collection of connected heterogeneous systems sharing various combination of resources and it is the combination of computer resources from multiple administrative domains applied to achieve a goal, it is used to solve scientific, technical or business problem that requires a great number of processing cycles and needs large amounts of data. One primary issue associated with the efficient utilization of heterogeneous resources in a grid environment is task scheduling. Task Scheduling is an important issue of current implementation of grid computing. The demand for scheduling is to achieve high performance computing. If large number of tasks is computed on the geographically distributed resources, a reasonable scheduling algorithm must be adopted in order to get the minimum completion time. Typically, it is difficult to find an optimal resource allocation for specific job that minimizes the schedule length of jobs. So the scheduling problem is defined as NP-complete problem and it is not trivial. Heuristic algorithms are used to solve the task scheduling problem in the grid environment and may provide high performance or high throughput computing or both. In this paper, a parallel asynchronous particle swarm optimization algorithm is proposed for job scheduling. The proposed scheduler allocates the best suitable resources to each task with minimal makespan and execution time. The experimental results are compared which shows that the algorithm produces better results when compared with the existing ant colony algorithm.

Keywords- *grid computing; task scheduling; NP-Complete problem, Parallel asynchronous particle swarm optimization, ant colony algorithm*

*****

I.INTRODUCTION

Scheduling in a grid computing system is not as simple as scheduling on a many machine because of several factors. These factors include the fact that grid resources are sometimes used by paying customers who have interest in how their jobs are being scheduled. Also, grid computing systems usually operate in remote locations so scheduling tasks for the clusters may be occurring over a network. It is because of these reasons that looking at scheduling in grid computing is an interesting and important problem to examine [1].

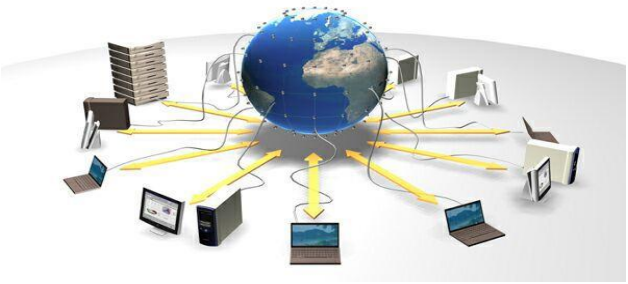

Fig 1.1 Grid Computing Environment

. For the static scheduling, jobs are assigned to suitable resources before their execution begin. However, for the dynamic scheduling, reevaluation is allowed of already taken assignment decisions during job execution [2]. It can trigger job migration or interruption based on dynamic information about the status of the system and the workload.

This job scheduling is the mapping of jobs to specific physical resources, trying to minimize some cost function specified by the user. This is a NP-complete problem and different heuristics may be used to reach an optimal or near optimal solution. Effective computation and job scheduling is rapidly becoming one of the main challenges in grid computing and is seen as being vital for its success.

In this paper, we propose a new improved parallel asynchronous particle swarm optimization approach for achieving optimal task scheduling in grid computing. This paper is organized as follows. Section II describes the use of ant and multiple ant colony algorithms in grid computing. In Section III, It covers problem formulation and methodology used in existing system. In Section IV, our proposed parallel asynchronous particle swarm optimization for the scheduling problem. In Section V, computational experiments and comparison studies are reported. Some concluding remarks are made in Section VI.

II.RELATED WORKS ON ACO AND MACO IN GRID COMPUTING ENVIRONMENT

One motivation of grid computing is to aggregate the power of widely distributed resources, and provide non-trivial services to users. To achieve this goal, an efficient grid scheduling system is an essential part of the grid. Rather than covering the whole grid scheduling area, survey provides a review of the subject mainly from the perspective of scheduling algorithms. The state of current research on scheduling algorithms for the new generation of computational environments will be reviewed in this section. Here, a collection of some commonly used heuristics algorithms

literature has been selected and discussed in the following section.

Ant Colony Optimization (ACO) [3] has been used as an effective algorithm in solving the scheduling problem in grid computing. ACO is inspired by a colony of ants that work together to find the shortest path between their nest and food source. Every ant will deposit a chemical substance called pheromone on the ground after they move from the nest to food sources and vice versa. Therefore, they will choose the shortest or optimal path based on the pheromone value. The path with high pheromone value is shorter than the path with low pheromone value. This behavior is the basis for a cooperative communication.

Balanced job assignment based on ant algorithm for computing grids called BACO was proposed by [4]. The research aims to minimize the computation time of job executing in grid environment which focused on load balancing factors of each resource. By considering the resource status and the size of the given job, BACO algorithm chooses optimal resources to process the submitted jobs by applying the local and global pheromone update technique to balance the system load. Local pheromone update function updates the status of the selected resource after job has been assigned and the job scheduler depends on the newest information of the selected resource for the next job submission. Global pheromone update function updates the status of each resource for all jobs after the completion of the jobs. From the experimental result, BACO is capable of balancing the entire system load regardless of the size of the jobs.

It is based on the improved ACO that proposed by [5]. The pheromone update function in this research is performed by adding encouragement, punishment coefficient and load balancing factor. The initial pheromone value of each resource is based on its status where job is assigned to the resource with the maximum pheromone value. The strength of pheromone of each resource will be updated after completion of the job. If a resource completed a job successfully, more pheromone will be added by the encouragement coefficient in order to be selected for the next job execution. If a resource failed to complete a job, it will be punished by adding less pheromone value. The load of each resource is taken into account and the balancing factor is also applied to change the pheromone value of each resource.

An Enhanced Ant Colony Algorithm proposed in [6] is allocating an application to a host from a pool of available hosts and applications by selecting the best match. The proposed algorithm uses two types of ET (Execution Time) matrices and finds the list of available resources in the grid environment, form the ET matrix and start the scheduling. Based on the ET matrix the algorithm calculates the probabilistic makespan. The proposed scheduler allocates adopt the system environment freely at runtime. This resource optimally and adaptively in the scalable, dynamic and distribute controlled environment.

Multiple Ant Colony Optimization (MACO) approach [7] presented for load balancing in circuit-switched networks. MACO uses multiple ant colonies to search for alternatives to an optimal path. One of the impetuses of MACO is to optimize the performance of a congested network by routing calls via several alternatives paths to prevent possible congestion along an optimal path.

In MACO, each group of mobile agents corresponds to a colony of ants, and the routing table of each group corresponds to a pheromone table of each colony [8]. By adopting the MACO approach, it may be possible to reduce the likelihood that all mobile agents establish connections using only the optimal path [8]. The advantage of using MACO in circuit-switched routing is that it is more likely to establish connections through multiple paths to help balance the load but does not increase the routing overhead.

IMACO framework is proposed in [9]. In this framework there are two levels of interaction the first one is the colony level and the second one is the population level. The colony level interaction can be achieved through the pheromone depositing process within the same colony; the pheromone updating mechanism is responsible for the implementation of this kind of interaction. The population level interaction is achieved by evaluating the pheromones of different colonies using some evaluation function; the responsibility here is of the pheromone evaluating mechanism [10].

IMACO-AVG method [11] proposed exploration and exploitation process. Exploration and exploitation is controlled by the parameter q_0 whose value is in [0, 1]. It is usually used in ant's probabilistic decision as trade-off between exploitation and exploration. Setting q_0 to zero means that the algorithm uses a pure exploration while pure exploitation is reached by setting q_0 to one. This technique enables the utilized ant colonies to work with different levels of exploration. Some will prefer high exploration of new areas of search space while other colonies will prefer high exploitation search history.

Information exchange in multiple ant colony algorithms [12] is proposed that parallelization where an information exchange between several colonies of ants is done every k generations for some fixed k. They show by using simulations how much the running time of the algorithm decreases with an increasing interval between the information exchanges. But it is not discussed how this influences the quality of the solutions.

Through our literature survey on current scheduling algorithms are working in the grid computing scenario, we can find that heterogeneity, dynamism, computation and data separation are the primary challenges concerned by current research on this topic.

III.MULTIPLE ANT COLONY ALGORITHM AND PROBLEM DESCRIPTION

Recently, ant colony optimization (ACO) has been suggested to solve the task scheduling problem. But ACO approaches using a single colony system may suffer from specific local optima because of its tendency to use the positive feedback mechanism of pheromone, multiple ant colonies optimization (MACO) is employed to avoid this by using several ant colonies to solve combinatorial optimization problems cooperatively. To improve the performance of ACO approaches, MACO considers both positive and negative feedbacks in searching solutions, sharing the search information, and exploring a large area of the search space with mutual cooperation of ant colonies. So MACO

approaches have been explored for several optimization problems.

MACO seems to be appropriate approach to improve the performance of ACO algorithm. This algorithm offers good opportunity to explore a large area of the search space and find optimal solution. All colonies construct their solutions in parallel [13] and interaction mechanisms are designed for sharing experiences among colonies. Assume that M ant colonies would be used to tackle the scheduling problem, and each colony contains N ants for the search procedure.

A.The framework of the MACO approach:

- First the initialization of the algorithm is presented,
- Second Local and global phenomenon updating is performed,
- Third construct the solution for determine the makespan time and then terminal test of the algorithm is take place.

Initially several colonies of ant system are created, and then they perform iterating and updating their pheromone arrays respectively until one ant colony system reaches its local optimum solution. Every ant colony system owns its pheromone array and parameters and records its local optimum solution. Furthermore, once an ant colony system arrives at its local optimum solution, it updates its local optimum solution and sends this solution to global best-found center. In general, the approach can be briefly sketched as follows.

B.Initialization of algorithm:

Assume that *M* ant colonies would be used to tackle the scheduling problem, and each colony contains *N* ants for the search procedure. We denote by ant (*m*, *n*) the *nth* ant in the *mth* colony. At the beginning, ants are distributed on computing nodes randomly, and the pheromone value τ_j^m of the *mth* colony on node *Cj* is initialized as a small value.

C*Interaction of Multiple Ant Colony System*

Pheromone is used as the interaction mechanism not only between the ants of the same colony but also among ant colonies. As traditional ACO approaches, each colony has its own pheromone to interact between the ants of the same colony. Furthermore, pheromone information is also used for the interaction of ant colonies. The colony level interaction is achieved by evaluating the pheromones of different colonies. More specifically, the evaluated pheromone *τj* on node *Cj* in terms of pheromone values of all colonies is defined as follows,

$$\tau_j = \Sigma_{m\varepsilon [1, M]} \tau_j^m / M \qquad (1)$$

Where M-no of ant colonies in system, τ_j - is computed in terms of pheromone values of all colonies

The pheromone evaluation mechanism averages the pheromone values of all colonies, which represents information of all colonies. Based on the average of the available experiences of ants of all colonies, an ant will decide how to choose an edge.

D. *Pheromone Updating*

As ant (m, n) completes its tour, local pheromone updating is applied on the visited nodes. More specifically, if ant (m, n) assigns task Ti to node Cj, the local pheromone update is given by:

$$\tau_j^m = (1-\rho)\, \tau_j^m + \rho\, \Delta\, \tau_j^{mn} \qquad (2)$$

Where ρ- Evaporation rate

In global pheromone updating phrase, after all ants of all colonies construct their solutions, the ant achieving the best-so-far solution in its colony will deposit an amount of pheromone on the edges of its path according to the following rule

$$\tau_j^m = (1-\lambda)\, \tau_j^m + \lambda\, \Delta\, \tau_j^{mn} \qquad (3)$$

Where λ- Evaporation

Thus, the ant finding the solution with the minimum makespan can lay a larger intensity of the pheromone on its tour.

E.Solution construction

In this phase, ant (*m*, *n*) moves through computing nodes and assigns task *Ti* to node *Cj* probabilistically in terms of pheromone and heuristic information until all tasks have been allocated. The probability $P^{mn}_{ij\ for}$ ant (m,n) to assign the task to node is defined as;

$$P^{mn}_{ij} = \tau_j\, Ɒ^{mn}_{ij} / \sum \tau_j\, Ɒ^{mn}_{ij} \qquad (4)$$

Where $Ɒ^{mn}_{ij}$ the heuristic information value for evaluating the assignment of task *Ti* to node *Cj* for ant (*m*, *n*). τ_j - is computed in terms of pheromone values of all colonies.

This approach terminates when the global pheromone updating is not improved in successive predefined no of solutions.

IV.PROPOSED PARALLEL ASYNCHRONUS PARTICLE SWARM OPTIMIZATION

Current parallel computing environments and optimization tasks make it difficult to achieve the ideal conditions required for high parallel efficiency with PSPSO. To overcome this problem, we propose an adaptive concurrent strategy, called parallel asynchronous particle swarm optimization (PAPSO) for job scheduling in grid environment, which can increase parallel performance significantly. In addition to parallelization, this strategy involves one significant modification to the sequential asynchronous PSO algorithm.[14] Since the updated velocity and position vectors of particle i are calculated using the best-found position and the global best-found position up to iteration *k*, the order of particle function evaluations affects the outcome of the optimization. Here the asynchronous approach does not need a synchronization point to determine a new search direction or new sets of design variables. Thus, the optimization can proceed to the next iteration without waiting for the completion of all function evaluations from the current iteration. PAPSO algorithm updates particle positions and velocities continuously based on currently available information.

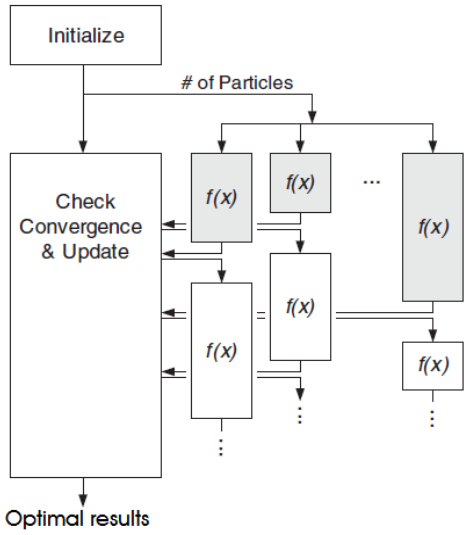

Figure 4.1 Process of Parallel asynchronous particle swarm optimization

PAPSO uses dynamic load balancing to determine processor workload during run time, thereby reducing load imbalance. This Pseudo-code for a asynchronous PSO algorithm[4],

1. Initialize Optimization
2. Initialize algorithm constants
3. Randomly initialize all particle positions and velocities
4. Perform Optimization
5. For k = 1, number of iterations
6. For i = 1, number of particles
7. Evaluate analysis function ($f(x)$)
8. Check convergence
9. Update p^{i}_{k} , p^{g}_{k} ,and particle positions andvelocities x^{i}_{k+1}, $v\,i_{k+1}$
10. End
11.End
12. Report Results

Figure 4.2 Pseudo Code for Asynchronous Particle Swarm Optimization

PAPSO design follows a master/slave paradigm. The master processor holds the queue of particles ready to send to slave processors and performs all decision-making processes such as velocity/position updates and convergence checks. It does not perform any function evaluations. The slave processors repeatedly evaluate the analysis function using the particles assigned to them.

Once the initialization step has been performed by the master processor, particles are sent to the slave processors to evaluate the analysis function. Whenever a slave processor completes a function evaluation, it returns the value and corresponding particle number to the master processor, which places the particle number at the end of the task. Since there is no global synchronization in PAPSO, communication time is hidden within the computation time of the slave processors. Because the master processor can communicate with only one slave processor at a time, each slave processor remains idle for a short period of time while waiting to connect to the master processor after completing a function evaluation. However, this idle time is typically negligible compared to the computation time required for each function evaluation.

The following parameters are used for analysis the algorithms,

- Makespan
- Degree of load imbalance
- Parallel efficiency
- Execution time

Makespan:

Makespan represents the latest completion among all the tasks.

Degree of load imbalance:

Degree of load imbalance represents the imbalance among computing nodes.

Parallel efficiency:

Parallel efficiency indicates the efficiency of the parallel processes.

Execution time:

Execution time represents the overall completion time for processing each gridlets.

ADVANTAGES OF PROPOSED SYSTEM

In the context of parallel asynchronous particle swarm optimization scheduling, the scheduler attempts to find optimal distribution of work and minimum makespan time.

- Convergence rate is high. So the searching activity of the algorithm produces optimal result when compared with existing methods.
- The Parallel implementations are asynchronous optimization make efficient use of computational resources when a load imbalance exists.
- It uses dynamic load balancing to determine processor workload at compile time, so the completion time is decreased and this optimization is applicable for large complex problems.
- This PAPSO approach applies to the practical situation easily and adaptively in scalable, dynamic and distributed environment.

V EXPERIMENT AND RESULT

The figure 5.1 illustrates the comparison of ant colony algorithm, multiple ant colony algorithm, enhanced multiple ant colony algorithm and parallel asynchronous particle swarm optimization based on the makespan and gridlets. Makespan represents the latest completion time among all the tasks. From the figure 5.1 in ACO decreased roughly as the no of ants increased. Due to the influence of stagnation situation, ACO cannot benefit from the increase of the number of ants [15]. In contrast, the solution quality of MACO was improved through the cooperation of multiple colonies. In Proposed EMACO, it can be improved makespan of the tasks using the inclusion of EETij execution time in local searching process. But Proposed PAPSO updates particle positions and velocities continuously based on currently available information. So the completion time of the tasks is minimized .Proposed PAPSO has minimum makespan when compared with existing algorithms. In Proposed PAPSO, particles position and velocity are updated based on the

 ISSN: 2321-8169
2384 – 2389

currently available information. Therefore the completion time of the task is minimum. From the figure 7.1, makespan of Proposed PAPSO has been reduced 52% with ACO, reduced 28% with MACO and reduced 11% with Proposed EMACO respectively.

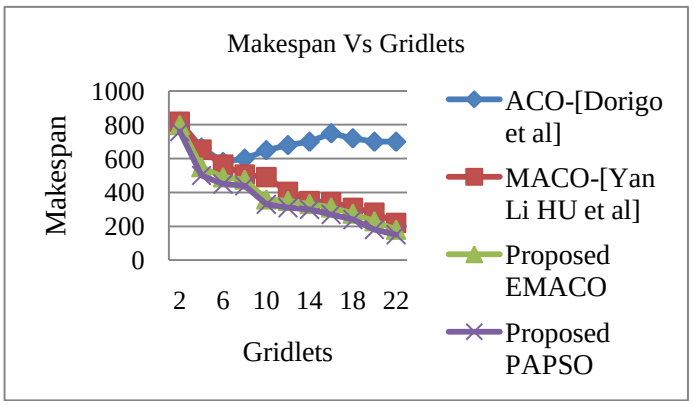

Fig 5.1 Performance Comparisons of ACO, MACO, Proposed EMACO and Proposed PAPSO-makespan

Figure 5.2 illustrates the degree of imbalance of ACO, MACO, Proposed EMACO and Proposed PAPSO. MACO, Proposed EMACO and Proposed PAPSO are parallelly implemented. The parallelization of the process provides the balanced environment. In PAPSO each particle checks the convergence and updates their positions and velocities at different time manner. Here Proposed PAPSO algorithm manages to decrease the degree of imbalance compared with the ACO, MACO and Proposed EMACO. From the figure 5.2, the degree of load imbalance of Proposed PAPSO has been decreased 62%, with ACO, decreased 34% with MACO and decreased 21% with Proposed EMACO respectively.

Figure 5.3 shows that the execution time of MACO, Proposed EMACO and Proposed PAPSO. Here proposed system PAPSO compared with the existing system EMACO. In Proposed PAPSO local and global updates particle positions and velocities continuously based on currently available information. The master processor holds the no of particles ready to send to slave processors and performs all decision-making processes such as velocity/position updates and convergence checks. Based on these processes the execution time of Proposed PAPSO is low when compared with MACO and Proposed PAPSO. From the figure 5.3, the execution time of Proposed PAPSO has been decreased 48% with MACO and decreased 19% with Proposed EMACO respectively.

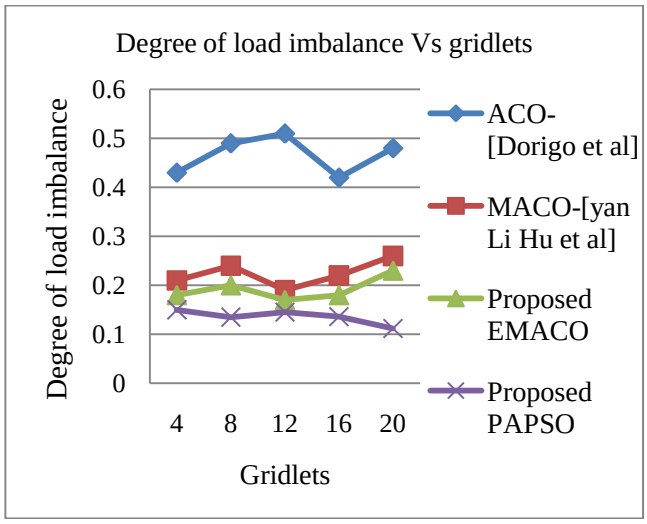

Fig 5.2 Performance Comparisons of ACO, MACO, Proposed EMACO and Proposed PAPSO-Degree of load imbalance

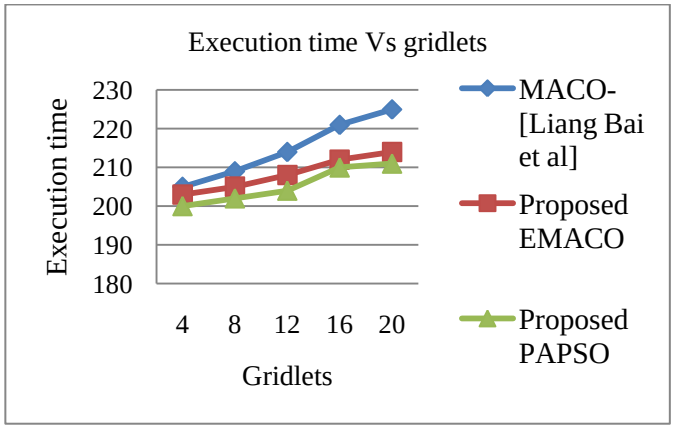

Fig 5.3 Performance Comparisons of MACO, Proposed EMACO and Proposed PAPSO-
Execution time

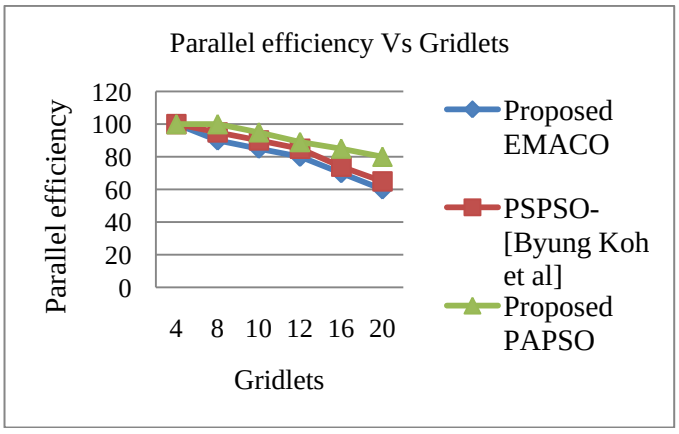

Fig 5.4 Performance Comparisons of Proposed EMACO, PSPSO and Proposed PAPSO-Parallel Efficiency

Figure 5.4 shows that the parallel efficiency of Proposed EMACO, PSPSO [4] and Proposed PAPSO. In

Proposed PAPSO local and global updates particle positions and velocities continuously and also check the convergence based on currently available information in parallelly. And also the degree of the load imbalance is low. Based on the above information the PAPSO increased the efficiency of the process when it is going to implement the large complex problems. If the no of gridlets are increased the parallel efficiency is decreased linearly. But the parallel efficiency of PAPSO is high when compared with other parallel mechanisms such as Proposed EMACO, PSPSO and Proposed PAPSO. From the figure 5.4 the parallel efficiency of the Proposed PAPSO has been increased 20.5 % with Proposed EMACO and increased 7% with PSPAO [4] respectively. Therefore the proposed PAPSO is a best suited method for tracking problem with more no of resources. From the results it is clearly evident that the proposed PAPSO offers better optimization a very fast rate and produced better makespan. In other words, these results prove the effectiveness of the PAPSO algorithm.

VI.CONCLUSION

Grid computing aims to assign tasks to computing nodes and minimize the execution time of tasks as well as workload across all nodes. Despite of the intractability, the scheduling problem is of particular concern to both users and grid systems. The proposed PAPSO approach achieves optimal schedule than the multiple ant colony algorithm. It has the best integrate performance. In PAPSO all the fitness values are evaluated by parallel processing that depends on the current information. Thus the optimization can proceed to the next iteration without waiting for the completion time of all function evaluations from the current iteration. And also it is implementing on shared memory machine, N particles and P processors in order to reduce the overall makespan of the grid system. From the experimental results the proposed system have minimum makespan ,minimum degree of load imbalance and high parallel efficiency.This process achieve optimal scheduling by completing the tasks with minimum execution time as well as utilizing the resources by uniform load distribution in an efficient way.

VII. REFERENCES

[1] Foster and Kesselman.C, The Grid:Blueprint for a Future Computing Infrastructure, Morgan Kaufman Publishers, USA, 1999.

[2] M. Chtepen, “Dynamic scheduling in grids system,” Sixth Firw PhD Symposium, Faculty of Engineering, Ghent University, pp. 110, 2005.

[3] Stefka Fidanova and Mariya Durchova “Ant Algorithm for Grid Scheduling Problem”, Large Scale Computing, Lecture Notes in Computer Science No. 3743, Springer, 2006, pp.405-412.

[4] Chang.R, Chang.J, and Lin.P "Balanced Job Assignment Based on Ant Algorithm for Grid Computing," presented at Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference, 2007, pp. 291-295.

[5] Yan.H, Shen.X, Li.X, and Wu.M, "An improved ant algorithm for job scheduling in grid computing," presented at Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, vol. 5, 2005, pp. 2957-2961.

[6] Kousalya.K and Balasubramanie.P ,“An Enhanced Ant Colony Algorithm for Grid Scheduling Problem”, IJCSNS International Journal of Computer Science and Network Security, 2008, Vol.8, No.4.

[7] Kwang Mong Sim, and Weng Hong Sun, " Multiple Ant Colony Optimization for Load Balancing" , Department of Information Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong.

[8] Kwang Mong Sim and Weng Hong Sun," Ant Colony Optimization for Routing and Load - Balancing: Survey and New Directions", Ieee Transactions On Systems, Man, And Cybernetics—Part A: Systems And Humans, Vol. 33, No. 5, September 2003.

[9] H. Kawamura, M. Yamamoto, K. Suzuki, and A. Ohuchi, "Multiple ant colonies algorithm based on colony level interactions", IEICE Trans. Fundamentals, Vol. E83-A, No.2, 2000, pp. 371-379.

[10] J. Jong, and M. Wiering, "Multiple ant colony system for the bus-stop allocation problem", Proc of the Thirteenth Belgium-Netherlands Conference on Artificial Intelligence BNAIC’01, Amsterdam, Netherlands, 2001, pp. 141–148.

[11] Alaa Aljanaby , Ku Ruhana Ku-Mahamud , Norita Md. Norwawi,” Interacted Multiple Ant Colonies Optimization Framework: an Experimental Study of the Evaluation and the Exploration Techniques to Control the Search Stagnation,” International Journal of Advancements in Computing Technology Volume 2, Number 1, March 2010.

[12] Martin Middendorf, Frank Reischle, Hartmut Schmeck ” Information Exchange in Multi Colony Ant”, Institut f□ur Angewandte Informatikund Formale Beschreibungsverfahren, Universit□at Karlsruhe, D-76128 Karlsruhe, Germany.

[13] Kruger.F, Middendorf.M, and Merkle.D,” Studies on a Parallel Ant System for the BSP Model” Unpub. Manuscript, 1998.

[14] Byung-I Koh, Alan D. George, Raphael T. Haftka And Benjamin J. Fregly(2006), Parallel Asynchronous Particle Swarm Optimization, international journal for numerical methods in engineering Int. J. Numer. Meth. Engng ; 67:578–595

[15] Yan-Li Hu, Liang Bai, Song-Yang Lao, Wei-Ming Zhang(2010),” Task Scheduling with Load Balancing using Multiple Ant Colonies Optimization in Grid Computing” International Conference on Natural Computation (ICNC).