Scheduling in High Performance Computing Environment using Firefly Algorithm and Intelligent Water Drop Algorithm

Ms. D. Thilagavathi[1], Dr. Antony Selvadoss Thanamani[2]

[1]*Research Scholar, Research Department of Computer Science, NGM College, Pollachi - 642001, Tamilnadu, India.*
[2] *Professor and Head, Research Department of Computer Science, NGM College, Pollachi - 642001, Tamilnadu, India.*

Abstract **- Scheduling of jobs in High Performance Computing environment is a NP-Hard Problem. Many conventional algorithms were used by the researchers to solve this problem. But the results got by using swarm intelligence based algorithms gives a near optimal solution then conventional method. In this paper, we propose two such algorithms like Firefly Algorithm and Intelligent Water Drop Algorithm which outperforms the results of conventional algorithms and also some swarm intelligence algorithms like Ant Colony Optimization, Particle Swarm Optimization comparatively. Both this proposed algorithms are used to dynamically create an optimal schedule to finish the submitted jobs in a High Performance Computing environment showing promising results.**

Keywords**— High Performance Computing, NP-Hard, Firefly, Intelligent Water Drop, Ant Colony Optimization, Particle Swarm Optimization.**

I. INTRODUCTION

The term high performance computing (HPC) refers to any computational activity requiring more than a single computer to execute a task. HPC has the capacity to handle and analyze massive amounts of data at high speed. Tasks that can take months using normal computers is possible in days or even minutes. It might be utilized to model and tackle highly complex issues over a scope of high esteem areas.

Uses are diverse and examples include facial reconstruction modeling, animated graphics, fluid dynamic calculations, nuclear energy research, petroleum exploration, car crash simulations, airflows over aircraft wings, data mining and storage, and visualization.

The objective of task scheduling is to achieve high system throughput and to match the application need with the accessible computing resources. This is matching of resources in a non-deterministically shared heterogeneous environment. The decision of the best pairs of tasks and resources is NP-Hard problem. A good task scheduler should adapt its scheduling strategy to the changing environment and the types of tasks. In recent years there has been a large increase in HPC technologies research, which has produced some reference implementations. Various sciences can benefit from the use of HPCs to solve CPU-intensive problems, creating potential benefits to the entire society.

In this paper, we address a job scheduling problem on High Performance Computing environment, in which to obtain near optimal solution so as to complete the task in minimum period of time as well as utilizing the resources in an efficient way is considered as the objective. To tackle this problem, Firefly Algorithm (FA) and Intelligent Water Drop (IWD) algorithm is proposed to search for the optimal schedule which in turn gives the solution to complete the batch of jobs in minimum period of time.

The rest of the paper is organized as follows. Section 2, related works is described; Section 3 presents the problem statement related to job scheduling. Details of the proposed Firefly algorithm and Intelligent Water Drop Algorithm are reported in Section 4 and 5 respectively. Comparison with other conventional algorithms is discussed in Section 6 and the conclusions are presented in Section 7.

II. RELATED WORKS

Both Grid and Cloud environment is HPC environment. Grid computing is a form of distributed computing that involves coordinating and sharing computing, application, data storage or network resources across dynamic and geographically dispersed organizations[1]. Cloud computing, the long-held dream of "computing as a utility", is emerging as a new paradigm of large-scale distributed computing driven by economies of scale, in which a pool of highly scalable, heterogeneous, virtualized, and configurable and reconfigurable computing resources(e.g., networks, storage, computing units, applications, data) can be

rapidly provisioned and released with minimal management effort in the data centers [2-6]. Scheduling in Grid and cloud though has been intensively studied only during the recent year; there exists a great variety of the algorithms for scheduling in Grid and Cloud. This section discusses some of the research works on the algorithms for scheduling in HPC used.

Several heuristic algorithms [7, 8] have been proposed for the job scheduling problem. Ant Colony Optimization (ACO) is a heuristic algorithm with efficient local search for combinatorial problems. ACO imitates the behavior of real ant colonies in nature to search for food and to connect to each other by pheromone laid on paths travelled. Many researches use ACO to solve NP-hard problems such as travelling salesman problem, graph coloring problem, vehicle routing problem, and so on. In paper [9] Ruay-Shiung Chang et al suggests modified ant algorithm as Balanced ACO (BACO) algorithm which reduces makespan time and also tried to balance the entire system load. This work was implemented in the Taiwan UniGrid Platform. The BACO algorithm selects a resource for submitting the request (job) by finding the largest entry in the Pheromone Indicator (PI) matrix among the available jobs to be executed. This work was carried for independent jobs and not for workflow jobs.

PSO is a population-based search algorithm based on the simulation of the social behavior of bird flocking and fish schooling. HU Xu-Huai et al [10] proposes an Immune Particle Swarm Optimization (IPSO) algorithm. The basic idea of the IPSO is to record the particles with a higher fitness in the evaluating process, and make the new particles which satisfy neither the assumption nor the constraint condition replaced by the recorded ones. In addition, immune regulation should be done to maintain the species diversity while it decreases. This paper mainly discusses the independent task scheduling. Experiments show that the PSO algorithm has the best integrate performance.

Genetic algorithm may be used to solve optimization problems by mimicking the genetic process of biological organisms. In [11] authors investigate the job scheduling algorithm in grid environments as an optimization problem. This paper gives an improved genetic algorithm with limited number of iteration to schedule the independent tasks onto Grid computing resources. The evolutionary process is modified to speed up convergence as a result of shortening the search time, at the same time obtaining a feasible scheduling solution. The scheduling creating process in GA algorithm costs the longest time.

III. THE PROBLEM STATEMENT

Suppose that $R = \{r_1, r_2, r_3, \ldots, r_m\}$ are m resources and $J=\{j_1, j_2, j_3, \ldots, j_m\}$ are n independent client jobs. The speed of each resource is expressed in the form of MIPS (Million Instructions Per Second), and the length of each job is expressed in the form of number of instructions. Define C_{ij} as the time that resource r_i needs to finish job j_i; ΣC_i is the total time that resource r_i completes all the jobs submitted to it. $C_{max} = \max \{\Sigma C_i\}$ is makespan time, which is the maximum completion time or the time when the HPC system completes the latest job. The flowtime, which is the total of execution times of all tasks submitted to the HPC.

Makespan and flow time are critical factors in scheduling problems; moreover the efficiency and effectiveness of each algorithm depend mainly on the makespan and the flow time.

Scheduling the Longest Job on the Fastest Resource (LJFR) rule minimizes the makespan time. However, to minimize the flowtime we should use scheduling Shortest Job on the Fastest Resource (SJFR) rule. Flowtime minimization tries to decrease the average job completion time; at the cost of the longest job finishing in a long time.

While, makespan minimization strives to make no job finishes in too long time; at the cost of most jobs finish in long time. So it is obvious that, the minimization of makespan will consequently maximize the flowtime and vice versa. The goal of job scheduling process is to dynamically allocate the n jobs to the m resources in order to complete the tasks within a minimum makespan and flowtime as well as utilizing the resources effectively.

IV. FIREFLY ALGORITHM

A. Standard Firefly Algorithm

Firefly algorithm (FA) is a metaheuristic algorithm, inspired by the flashing behavior of fireflies. The Firefly Algorithm (FA) is a population-based technique to find the global optimal solution based on swarm intelligence, investigating the foraging behavior of fireflies [12].

The main function of the firefly's flash is to operate as a signal method to attract other fireflies. The flashing signal by fireflies is to attract mating partners and preys and share food with others.

Similar to other metaheuristics optimization methods, firefly algorithm generates random initial population of feasible candidate solutions. All fireflies of the population are handled in the solution search space with the aim that knowledge is collectively shared among fireflies to guide the search to the best location in the search space. Each particle in the population is a firefly, which moves in the multi-dimensional search space with an attractiveness that is dynamically updated based on the

knowledge of the firefly and its neighbors. Firefly optimization algorithm illustrated by [12, 13] can be described as follows:

- The firefly x attracts all other fireflies and is attracted to all other fireflies.
- The less bright firefly is attracted and moved to the brighter one.
- The brightness decreases when the distance between fireflies is increased.
- The brightest firefly moves randomly (no other fireflies can attract it).
- The firefly particles are randomly distributed in the search space.

According to above rules there are two main points in firefly algorithm, the attractiveness of the firefly and the movement towards the attractive firefly.

The main steps of firefly algorithm as described in [14, 15] are as follows:

Create and initialize N firefly particles
Determine the light intensity for each firefly
Determine the distance between each tow fireflies
repeat
for i=1: N
for j=1 : N
if (Ii <Ij) move firefly i towards firefly j end if
Update the attractiveness with distance r by exp[-γr]
Evaluate the new solution and update light intensity
End for j
End for i
Rank the fireflies and find the current global best
until Termination condition is met

B. Merits

FA has two major advantages over other algorithms: automatical subdivision and the ability of dealing with multimodality. It outperforms other optimization methods in terms of convergence and cost minimization in a statistically significant manner [15]. Moreover, FAs are simple, distributed and do not have central control or data source which allows the system to become more scalable.

C. Demerits

Firefly algorithm has some disadvantage such as getting trapped into several local optima. Firefly algorithm performs local search as well and sometimes is unable to completely get rid of them. Firefly algorithm parameters are set fixed and they do not change with the time. In addition Firefly algorithm does not memorize or remember any history of better situation for each firefly and this causes them to move regardless of its previous better situation, and they may end up missing their situations.

D. Algorithm Complexity

Almost all metaheuristic algorithms are simple in terms of complexity, and thus they are easy to implement. Firefly algorithm has two inner loops when going through the population n, and one outer loop for iteration t. So the complexity at the extreme case is O (n^2t). As n is small (typically, n = 40), and t is large (say, t = 5000), the computation cost is relatively inexpensive because the algorithm complexity is linear in terms of t. The main computational cost will be in the evaluations of objective functions, especially for external black-box type objectives. This latter case is also true for all metaheuristic algorithms. After all, for all optimization problems, the most computationally extensive part is objective evaluations. If n is relatively large, it is possible to use one inner loop by ranking the attractiveness or brightness of all fireflies using sorting algorithms. In this case, the algorithm complexity of firefly algorithm will be O (ntlog (n)).

E. Applications

- Digital Image compression and Image processing
- Feature selection and fault detection
- Antenna design
- Structural design
- Scheduling
- Semantic web Composition
- Chemical base equilibrium
- Clustering
- Dynamic problems
- Rigid image registration problems

V. INTELLIGENT WATER DROP ALGORITHM

A. Standard Intelligent Water Drop Algorithm

Intelligent Water drops Algorithm was introduced by Shah-Hosseini, H. in 2007 [16]. It is a population based constructive optimization algorithm which has been inspired from natural rivers and exploit the path finding strategies of rivers. A natural river often finds good paths among lots of possible paths in its ways from the source to destination. These near optimal or optimal paths follow from actions and reactions occurring among the water drops and the water drops with their riverbeds. In the IWD algorithm, several artificial water drops cooperate to change their environment in such a way that the optimal path is revealed as the one with the lowest soil on its links. The solutions are thus incrementally constructed by the IWD algorithm.

In the original IWD algorithm [17, 18] the water drops are created with two main properties:

- Velocity
- Soil

Both of above mentioned properties of IWD may change during its lifetime. The IWD begins its trip from a source to reach some destination with an initial velocity and zero soil. During its trip, an IWD travels in the environment from which it removes some soil and may gain some speed. This soil is removed from the path joining the two locations. IWD is supposed to flow in discrete steps. From its current location to its next location, its velocity is increased by the amount that is non-linearly proportional to the inverse of the soil between the two locations and the amount of soil added to the IWD is non-linearly proportional to the inverse of the time needed for the IWD to pass from its current location to the next location. Therefore, a path having less soil lets the IWD becomes faster than a path having more soil and the time interval is calculated by the laws of physics of linear motion. Thus, the time taken is proportional to the velocity of the IWD and inversely proportional to the distance between the two locations. An IWD prefers the paths having low soils than the paths having high soils.

The IWD algorithm as specified by Shah-Hosseini H. in is as follows:

1. Initialization of static parameters.
2. Initialization of dynamic parameters.
3. Spread the IWDs randomly on the nodes of the graph.
4. Update the visited node list of each IWD.
5. Repeat Steps a to d for those IWDs with partial solutions.
a. For the IWD residing in node i, choose the next node j, which does not violate any constraints of the problem and is not in the visited node list of the IWD.
b. For each IWD moving from node i to node j, update its velocity.
c. Compute the soil.
d. Update the soil.
6. Find the iteration-best solution from all the solutions found by the IWDs.
7. Update the soils on the paths that form the current iteration best solution.
8. Update the total best solution by the current iteration - best solution.
9. Increment the iteration number.
10. Stops with the total best solution.

B. Merits

- It provides good quality solutions using average values.
- IWD algorithm has fast convergence when compared to other methods.
- It is also flexible in the dynamic environment and pop-up threats are easily incorporated.

C. Algorithm Complexity

Time complexity of IWD Algorithm when there is M Iterations, N Nodes, E Edges and N IWDs Path selection then the complexity is O (M * N * E) which is Very small comparing to O (2^N).

Space complexity for N IWD's, N Solutions is O (N).

D. Applications

- Travelling Salesman Problem (TSP)
- Multidimensional Knapsack Problem
- Air Robot Path Planning
- N-Queen puzzle
- Vehicle Routing Problem
- Economic Load Dispatch
- Continuous Optimization Applications
- Scheduling problem
- Data clustering and Automatic multilevel thresholding

VI. COMPARISON WITH CONVENTIONAL ALGORITHMS

Comparison of bio-inspired algorithms with conventional algorithms can be discussed on the basis of following criteria:

- Intelligence: Bioinspired Algorithms are based on simple rules which take bottom-up approach. While conventional algorithms takes top-down approach.
- Testing: In Bioinspired methods, improvements have to be verified on successive generations taking more time while in conventional, testing results can be obtained immediately.
- Improvement: Improving of the Bio-inspired algorithms is not easy because verifiability compared to conventional algorithms.
- Flexibility to practical situation: Bioinspired algorithms have to be modified when applied to practical problems, while conventional algorithms are built keeping the practical situations and the end result in mind.

VII. CONCLUSION

This paper presents strategies for scheduling jobs in HPC environment using IWD algorithm and firefly algorithm which is able to find optimal solutions. The efficiency and practicability of IWD and FA is proved by testing in experimental environment and the results were better compared to that of conventional algorithms. From the results, it is proved that IWD and FA are more efficient and it also avoids the problem of consuming a large number of iterations. The results of the experimental study support the claims that the proposed

algorithm is superior to other related strategies. As a consequence, further research can focus on the points for amplification of strengths and eliminating the weaknesses. The IWD and FA algorithm demonstrates that the nature is an excellent guide for designing and inventing new nature-inspired optimization algorithms.

REFERENCES

[1] Foster I. *The Grid: Blueprint for a New Computing Infrastructure* (2nd Edition) [M]. Morgan Kaufmann Publishers Inc., ISBN: 1-55860- 993-4, 2004.

[2] Armbrust M, Fox A, Griffith R, Joseph A D, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A and Stoica I, *A view of cloud computing*, Communications of the ACM, Vol. 53, No. 4, 2010, pp. 50-58.

[3] Nidhi Jain Kansal and Inderveer Chana, *Cloud. Load Balancing Techniques : A Step Towards Green Computing*, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No. 1, 2012, pp. 238-246.

[4] Iosup, A., Ostermann, S., Yigitbasi, M.N., Prodan, R., Fahringer, T. and Epema, D.H.J, *Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing*, IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 6, 2011, pp. 931-945.

[5] Almutairi, A., Sarfraz M., Basalamah S., Aref W. and Ghafoor A, *A Distributed Access Control Architecture for Cloud Computing*, IEEE Software Vol. 29, No. 2, 2012, pp. 36-44.

[6] Junaid Qayyum, Faheem Khan, Muhammad LaL, Fayyaz Gul, Muhammad Sohaib and Fahad Masood, *Implementing and Managing framework for PaaS in Cloud Computing*, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No. 3, 2011, pp. 474-479.

[7] Thilagavathi, D. and A. S. Thanamani, *A Survey on Dynamic Job Scheduling in Grid Environment Based on Heuristic Algorithms*, International Journal of Computer Trends and Technology Vol. 3, Issue 4, 2012, pp. 531-536.

[8] Thilagavathi, D. and A. S. Thanamani, *Heuristics in Grid Scheduling*, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Vol. 2 Issue 8, August 2013, pp. 2427-2432.

[9] Ruay-Shiung Chang, Jih-Sheng Chang, Po-Sheng Lin, *An ant algorithm for balanced job scheduling in grids*, Future Generation Computer Systems 25 (2009) 20–27. (FGCS- Elsevier).

[10] HU Xu-Huai, OUYANG Jing-Cheng, YANG Zhi-He, CHEN Zhuan-Hong, *An IPSO algorithm for grid task scheduling based on satisfaction rate*, International Conference on Intelligent Human-Machine Systems and Cybernetics, 2009.

[11] Hao Yin, Huilin Wu, Jiliu Zhou, Abdul Hanan Abdullah, and Chai Chompoo-inwai, *An Improved Genetic Algorithm with Limited Iteration for Grid Scheduling*, The Sixth International Conference on Grid and Cooperative Computing(GCC 2007), IEEE.

[12] Yang, X.S., *Nature-inspired metaheuristic algorithms*. 2010: Luniver Press.

[13] Senthilnath, J., S. Omkar, and V. Mani, *Clustering using firefly algorithm: Performance study*. Swarm and Evolutionary Computation, 2011.

[14] Yang, X.S., *Firefly algorithms for multimodal optimization*. Stochastic Algorithms: Foundations and Applications, 2009: p. 169-178

[15] Yousif, Adil, et al. *Intelligent Task Scheduling for Computational Grid*. 1st Taibah University International Conference on Computing and Information Technology. 2012.

[16] Shah-Hosseini, H. (2007*). Problem Solving by Intelligent Water Drops*. *Proc. IEEE Congress on Evolutionary Computation*, (pp. 3226-3231). Singapore.

[17] Shah-Hosseini, H. (2008a*). Intelligent water drops algorithm: a new optimization method for solving the multiple knapsack problem*. Int. Journal of Intelligent Computing and Cybernetics, Vol. 1, No. 2, pp. 193-212.

[18] Shah-Hosseini, H. (2008b). *The Intelligent Water Drops algorithm: A nature-inspired swarm-based optimization algorithm. Int. J. Bio-Inspired Computation*, Vol. 1, Nos. 1/2, pp. 71–79.

Ms. D. Thilagavathi received her MCA degree from Bharathidasan University in 2001 and completed her M.Phil. degree in Computer Science from Bharathiar University in 2005. She is currently pursuing her Ph.D. at the Research Department of Computer Science, NGM College, Pollachi, under Bharathiar University, Coimbatore. Her research interests include Object Oriented Analysis and Design, Grid Computing and Cloud Computing. She has 13 years of teaching experience. She is presently working as an Assistant Professor and Head, Department of Computer Technology, NGM College, Pollachi.

Dr. Antony Selvadoss Thanamani is presently working as Professor and Head, Research Department of Computer Science, NGM College, Pollachi, Coimbatore, India. He has published more than 100 papers in international/national journals and conferences. He has authored many books on recent trends in Information Technology. His areas of interest include E-Learning, Knowledge Management, Data Mining, Networking, Parallel and Distributed Computing. He has to his credit 25 years of teaching and research experience. He is a senior member of International Association of Computer Science and Information Technology, Singapore and Active member of Computer Science Society of India, Computer Science Teachers Association, NewYork.