Journal of Theoretical and Applied Information Technology
20th July 2015. Vol.77. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EFFICIENT LOAD BALANCING USING ANT COLONY OPTIMIZATION

MOHAMMAD H. NADIMI-SHAHRAKI, ELNAZ SHAFIGH FARD, FARAMARZ SAFI

Department of Computer Engineering, Najafabad branch, Islamic Azad University, Najafabad, Iran
nadimi@iaun.ac.ir, shafighfard@azaruniv.edu, fsafi@iaun.ac.ir

ABSTRACT

Workload and resource management are two essential functions provided in the service level of a Grid software infrastructure. Consistently, efficient load balancing algorithms are fundamentally important to improve the global throughput of these environments. Although previous works show that, ant colony algorithm works well for load balancing, the cost is a very important factor in this subject. In this paper, a grid load balancing algorithm is proposed by using an ant colony optimization which is able to consider shortest path, type of resource, and running speed of resource. The experimental results show that the proposed algorithm by using this ant colony optimization can reduce the cost of load balancing in comparison with standard algorithm DASUD.

Keywords: *Grid computing, Load balancing, Ant colony*.

1. INTRODUCTION

Although load balancing problem in conventional distributed systems has been intensively studied, new challenges in Grid computing still make it an interesting topic, and many research projects [1-5] are underway. This is due to the characteristics of Grid computing and the complex nature of the problem itself. Load balancing algorithms in classical distributed systems, which usually run on homogeneous and dedicated resources, cannot work well in the Grid architectures. Grid has lots of specific characteristics such as heterogeneity, autonomy, scalability, adaptability and resources computation-data separation, which make the load balancing problem more difficult.

Ant Colony System (ACS) [6] is one the most successful algorithms used in combinatorial optimization problems such as the Traveling Salesman Problem (TSP). The algorithm is inspired by the foraging behavior of a colony of ants communicating through chemical substances called pheromones which act like a memory preservation mechanism and provide guidance for ants in searching for shortest paths. However, considering only the shortest path can sometimes be useless, especially for processors in which response time is a very important factor. Instead, there have been introduced some efficient Ant Colony Optimizations (ACO), in which another type of resources such as their running time and their workload are considered to enhance efficiency [7-9].

Balanced Ant Colony Optimization (BACO) was proposed by Chang et al. (2007) [8] to minimize the computation time of job executed in Taiwan in grid environment. It focused on load balancing factors of each resource. By considering the resource status and the size of the given job, BACO algorithm chose optimal resources to process the submitted jobs. In this algorithm, finding a good resource might be difficult due to the limited longevity of ants, or the long delay in the response time. A Multiple Ant Colony Optimization (MACO) is proposed [9] for load balancing in circuit–switched networks. MACO uses multiple ant colonies to search for alternatives to an optimal path. Each group of mobile agents corresponded to a colony of ants, and the routing table of each group corresponded to a pheromone table of each colony.

Although previous works show that, ant colony algorithm works well for load balancing, the cost is still a very important factor in this subject. In this paper, a grid load balancing algorithm is proposed by using an ant colony optimization which is able to consider shortest path, type of resource, and running speed of resource. It consists of 5 steps as followed: obtaining job requirements, creating an ant for a job, calculating the scale of every workload of resource and depositing pheromone depending on its throughput of every resource's path, assigning job to the resource with the highest pheromone value, and performing global pheromone update after completely processing the job. The experimental results show that the proposed algorithm by using this ant colony optimization can reduce the cost of load balancing in comparison with standard algorithm DASUD.

The rest of paper is organized as follows. In section 2, the background and related works are reviewed. Then, the proposed algorithm is described in section 3. Afterwards in section 4, the proposed algorithm is experimentally evaluated. Finally, section 5 concludes the contribution and introduces some future research ideas.

2. BACKGROUND AND RELATED WORK

2.1. Load Balancing

The essential objective of a load balancing primarily consists of optimizing the average response time of applications, which often means maintaining the equality of workload proportion in all resources of a system. Conceptually, load balancing algorithms can be classified into two categories: static and dynamic [10, 11].

In static load balancing, a task is assigned to an available resource when it is generated or admitted to the system using a fixed schema. In contrast, the dynamic load balancing allocates/reallocates tasks to resources at runtime based on no priori task information, which may determine when and which tasks can be migrated. In this way, load imbalances can be resolved by redistributing tasks in real-time, solving the shortcoming of static load balancing. However, network traffic for transmitting load information to the load balancing system would increase the due in the decision dynamicity.

Load balancing algorithms can be defined by their implementation of the following policies: 1) information policy specifies what load information to be collected, when it is to be collected and from where, 2) triggering policy determines the appropriate moment to start a load balancing operation 3) resource type policy classifies a resource as server or receiver of tasks according to its availability status and capabilities, 4) location policy uses results of the resource type policy to find a suitable partner for a server or receiver, and 5) selection policy defines tasks that should migrate from overloaded resources to idlest ones.

2.2. LOAD BALANCING USING ANT COLONY

Nature is a good source of inspiration to solve the problems humans face. Ants provide a good example for the case of transporting goods or finding shortest paths. They are social insects which cooperate through group communication, laying down chemical substances called pheromones to mark locations that have already been visited. The pheromones also serve as a reference for the return route back to their nest. These pheromones are then used by other ants as an indicator of the best path between the nest and food sources. The amount of laid pheromone determines whether the path is desirable to be taken by others; higher pheromone levels indicate more desirable routes [12, 13].

ACO algorithms make use of simple agents called ants which iteratively construct candidate solutions to a combinatorial optimization problem [14]. The ants’ solution construction is guided by (artificial) pheromone trails and problem-dependent heuristic information. In principle, ACO algorithms can be applied to any combinatorial optimization problem by defining solution components which the ants use to iteratively construct candidate solutions, and on which they may deposit pheromone. An individual ant constructs candidate solutions by starting with an empty solution and then iteratively adding solution components until a complete candidate solution is generated. We will call each point at which an ant has to decide which solution component to add to its current partial solution a choice point. After the solution construction is completed, the ants give feedback on the solutions they have constructed by depositing pheromone on solution components which they have used in the process.

To avoid the search getting stuck, typically, before the pheromone trails get reinforced, all pheromone trails are decreased by a factor. The ants’ solutions are not guaranteed to be optimal with respect to local changes and hence may be further improved using local search methods. Based on this observation, the best performing ACO algorithms for many NP-hard static combinatorial problems are in fact hybrid algorithms combining probabilistic. There have been introduced some efficient load balancing algorithm by using ACO algorithms. Moallem and Ludwig introduced two distributed artificial life-inspired algorithms, ACO and Particle Swarm Optimization (PSO) to solve the static grid load balancing problem [15]. Distributed load balancing was categorized as a robust algorithm that could adapt to any topology changes in a network. In the study, an ant acted as a broker to

find the best node in terms of the pheromone value stored in the pheromone table. The node with the lightest load was selected as the best node. The position of each node in the flock could be determined by its load in PSO. The particle compared the load of nodes with its neighbors and moved towards the best neighbor by sending assigned jobs to it. The proposed algorithm performed better than ACO in job scheduling where jobs were submitted from different sources and in different time intervals. PSO showed better results than ACO in terms of the makes pan. However, PSO used more bandwidth and communication compared to ACO. The main drawback of Ant Colony was that jobs were not scheduled efficiently and therefore load among the resources were not balanced. This problem was fixed by increasing the number of ants that could explore the entire grid system to find resources with the lightest load.
Ali and Belal et al. proposed an ACO algorithm for dynamic load balancing in distributed systems through the use of multiple ant colonies [16]. In their algorithm, information on resources is dynamically updated at each ant movement. Load balancing system is based on multiple ant colonies information. Multiple ant colonies are adopted such that each node sent a colored colony throughout the network. The colored ant colonies are used to prevent ants of the same nest from following the same route and also force them to be distributed all over the nodes in the system. Each ant acts like a mobile agent which carried newly updated load balancing information to the next node. The algorithm is compared to the work-stealing approach for its load balancing in grid computing. Their experimental results show that multiple ant colonies work better than work-stealing algorithm in terms of the efficiency. However, the multiple ant colonies do not consider resource capacity and job characteristics. This can make matching the jobs with the best resources a difficult task for the scheduling algorithm. From the above research, ACS is the most popular variant of ACO that has been successfully used in grid load balancing.

3. PROPOSED ALGORITHM

The proposed algorithm consists of 5 steps which are 1) obtaining job requirements, 2) creating an ant for a job, 3) calculating the scale of every workload of resource to running speed of it for all nodes, and depositing pheromone depending on its throughput of every resource's path, 4) assigning job to the resource with the highest pheromone value, and 5) performing global pheromone update after completely processing the job to more information. The proposed algorithm is shown in Fig. 1.

```
The proposed Algorithm

Input: Distance of local node to other neighbors, Number of resources, Number of workload of every node

Output:
  Cost of load balancing

Initialization of parameters

1) For each num_resource,
    1-1) Get cpu_time = getCPUTime ();
    1-2) Calculate the probability of every node to be chosen next time considering (cpu_time, workload, and shortest path)
    End for
2) Calculate initializing pheromone for every node   (num_resource, num_gridlet);
    2-1) Load = get Load ();
3) Assign job to a node which has the highest chance to  select
4) Create Resource (id, cpu_time, load);

5) While (all nodes become balanced based on default workload)
    5-1) For (J = 0; J < num_gridlet; J++)
    5-2) Process (R, J);
    End while
6) Global update (evaporate, R, J);

7) Stopping criteria ?Yes go to Finish, No Go to 8;

8) Process iteration ++ Go to 1;
```

Figure 1: The proposed algorithm

In the proposed algorithm, each ant in a particular node that has overloaded job starts to select the best node by which the overload job can be done. For doing this, the throughput of each node is computed by simple Equation (1) using the count of idle processes and CPU run time.

$$thr_{source(j)} = \frac{\sum_{k \in i}^{n} workload(i)}{cpu\_Running\_time_j} \qquad (1)$$

According to Equation (1), η is the heuristic parameter which is computed by η=1/d_{ij} and thr is standing for the throughput. After computing of thr by using Equation (1) chance of selecting of every node as Equation (2) is computed.

$$P_{ij} = thr_{source(j)} * \frac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum_{k\in N} \tau_{ij}^{\alpha} * \eta_{ij}^{\beta}} \quad (2)$$

We followed the MAX-MIN Ant System (MMAS) for the update of pheromone trials. Regarding pheromone trial limits, an estimate of the upper bound is used to define max; where *best* is the best-so-far solution, and ρ is the evaporation rate of pheromone trails.
After calculating pheromone rate for every path, an amount of pheromone must be updated.
Cost function as shown by Equation (3) is a function that calculates total length of the path traced by an ant.

Ant (i).cost =cost function (ant (i).tour).tour (3)

Ant.cost is a result of cost function based on which costly paths are traced. Thus, as shown by Equation (4) the paths that are more costly will have less pheromone:

$$\tau_{ij} = \tau_{ij} + \frac{1}{(ant(i).cost)} * (thr_{source(j)}) \quad (4)$$

As time passes, as Equation (5) shows, the amount of pheromone evaporates on the paths in distributed system and evaporation will be happened after finishing iteration.

$$\tau=(1-\frac{1}{thr_{source(j)}}\rho)*\tau \quad (5)$$

4. EXPERIMENTAL SETUP

Experiments were run on a Mini Laptop with following properties: intel®Atom™ Cpu N270 @ 1.60GHz with a memory of 1.00 GB.
Matlab is adaptive software for this kind of algorithms which need lots of computing functions. The algorithms have been implemented in such an environment. Table 1 shows all parameters that must be set in our evaluation.

Table 1. Evaluation setting

Ant population	40
A	1
B	1
numbers of node	20
ρ	.05
Max iterations	30

In this experiment, nodes are completely heterogeneous, and the workload and CPU running time are selected randomly. Depending on the workload in every node, 15 to 25 iterations are done until the system is balanced.
This paper shows the differences of the two algorithms in the following figures. As it is indicated in Figure 2, the proposed algorithm gives a better result than ACO. In Figure 2, total cost of proposed algoritm shows 378.2318, but ACO algorithm shows 388.4078. Thus, in our algorithm, total cost of load balancing has been lessened in comparison to standard ACO algorithm. We tried to show that in the proposed algorithm, overhead in the network is less than that of the ACO algorithm because, as shown in Figure 3, compression of circles that act like a pointer showing time load balancing in a particular node is less than that of Figure 4 which is based on DASUD algorithm.

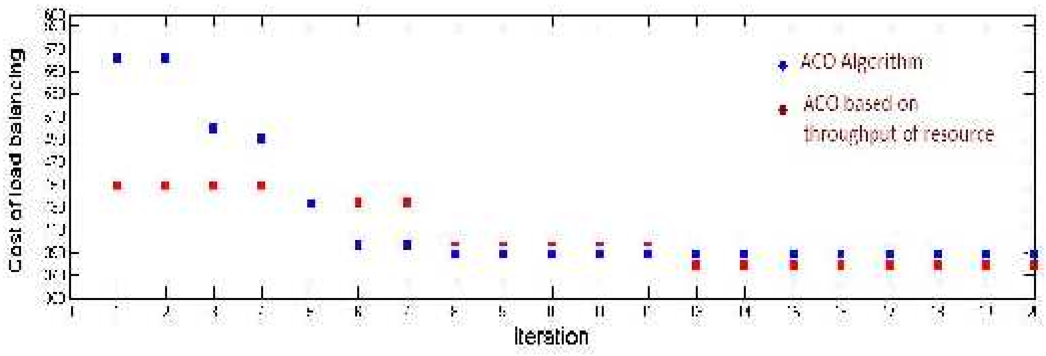

Figure 2: Comparison Of Cost Of Load Balancing

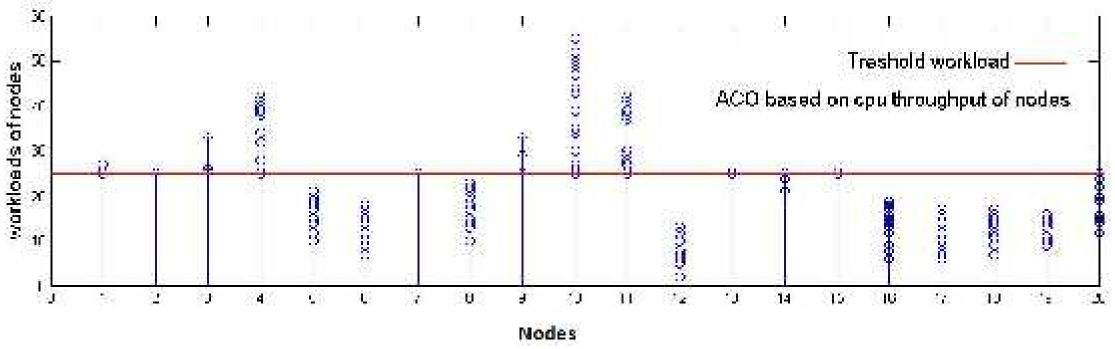

Figure 3: Workload Of Nodes In ACO Algorithm Based On CPU Throughput

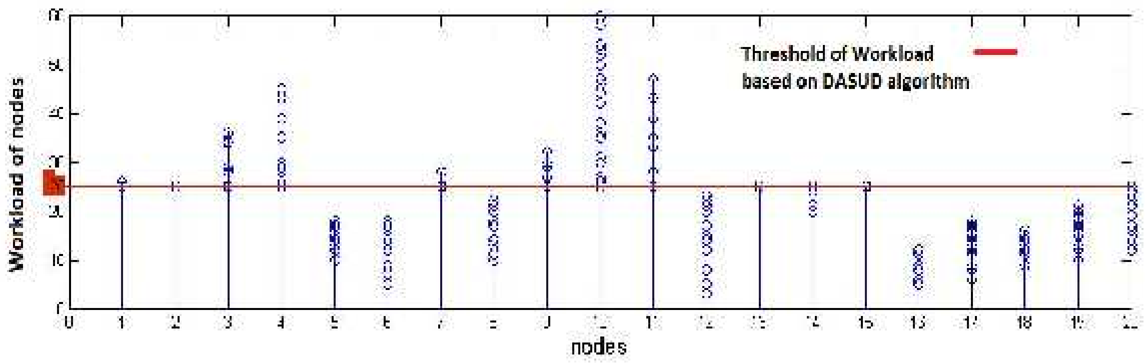

Figure 4: Workload Of Nodes In ACO Algorithm

5. CONCLUSION AND FUTURE WORK

In this paper, an efficient ACO approach to solve load balancing problem for heterogonous grid network had been proposed. It introduces a load balancing strategy based on ACO which offered enhancements by interfering in nodes throughput. The experimental results showed that the proposed approach had better performance than an ACO heuristic, and experimental results showed the reduction of cost. We try to work on projects based on ACO so that the designed hardware as the work station would like a shared local memory for routing, and the communication of the ants about balancing information could take place in these work stations.

ACKNOWLEDGMENT:

The authors would like to thank Islamic Azad University, Najafabad branch for supporting this research.

REFERENCES:

[1] Ali, A., Belal, M., A., & Al-Zoubi, M., B. (2010). Load balancing of distributed systems based on multiple ant colonies optimization. American Journal of Applied Sciences, 7(3), 433-438.
[2] Chang, R., Chang, J., & Lin, P. (2007). Balanced job assignment based on an algorithm for computing grids. Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference, 291-295.
[3] Nasir, H.J.A. & Ku-Mahamud, K.R. (2010). Grid load balancing using ant colony optimization. Proceeding of the2nd International Conference on Computer and Network Technology, 23-25April 2010, Bangkok, Thailand, 207-211
.[4] Sathish, K., & Reddy, A. (2008). Enhanced ant algorithm based load balanced task schedulingin grid computing. International Journal of Computer Science and Network Security, 8(10), 219-223.

[5] Moallem, A., & Ludwig, S. (2009). Using artificial life techniques for distributed grid job scheduling. Proceedings of the 2009 ACM Symposium on Applied Computing, 1091-1097.

[6] M. Dorigo, Gambardella L.M.(1997) Ant Colony system: A Cooperative learning approach to the traveling salesman problem, IEEE,Trans.on evolutionary computation, 1(1), 53-66.

[7] M. Dorigo and T. Stützle, Ant Colony Optimization, MIT Press, 2004.

[8] Chang, R., Chang, J., & Lin, P. (2007). Balanced job assignment based on ant algorithm for computing grids. Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference, 291-295

[9] A. D. Ali, and M. A. Belal, "Multiple ant colonies optimization for load balancing in distributed systems," in Proc. Inter. Conf, ICTA'07, 2007.

[10] Ratnesh Kumar Nath, "Efficient Load Balancing Algorithm in Grid Environment", Thapar University, Patiala, May 2007.

[11] Douglas Thain and Miron Livny. "The ethernet approach to grid computing". In Proc. Of 12th IEEE Symposium of High Performance Distributed Computing, 2003.

[12] Nasir, H.J.A. & Ku-Mahamud, K.R. (2010). Grid load balancing using ant colony optimization. Proceeding of the2nd International Conference on Computer and Network Technology, 23-25April 2010, Bangkok, Thailand, 207-211.

[13] Sathish, K., & Reddy, A. (2008). Enhanced ant algorithm based load balanced task schedulingin grid computing. International Journal of Computer Science and Network Security, 8(10), 219-223.

[14] M. Dorigo and T. Stützle, Ant Colony Optimization, IT Press, 2004.

[15] Moallem, A., & Ludwig, S. (2009). Using artificial life techniques for distributed grid job scheduling. Proceedings of the 2009 ACM Symposium on Applied Computing, 1091-1097.

[16] Ali, A., Belal, M., A., & Al-Zoubi, M., B. (2010). Load balancing of distributed systems based on multiple ant colonies optimization. American Journal of Applied Sciences, 7(3), 428-438.