American Journal of Applied Sciences 9 (7): 1101-1106, 2012
ISSN 1546-9239
© 2012 Science Publications

An Optimal Load Sharing Technique For Grid Computing

[1]Ramesh, D. and [2]A. Krishnan
[1]Department of CSE, Anna University of Technology, Tiruchirappalli, India
[2]K.S. Rangasamy College of Technology, Tiruchengode, India

Abstract: Problem statement: Grid and Cloud Computing is the fast growing industry, in which the grid computing shares the resources in the organization in an effective manner. Resource sharing requires more optimized algorithmic structure, otherwise the waiting time and response time are increased and the resource utilization is reduced. **Approach:** In order to avoid such reduction in the performances of the grid system, an optimal resource sharing algorithm is required. The traditional Min-Min algorithm is a simple algorithm that produces a schedule that minimizes the makespan than the other traditional algorithms in the literature. But it fails to produce a load balanced schedule. In our earlier study, a Load Balanced Min-Min (LBMM) algorithm is proposed that reduces the makespan and increases the resource utilization. This is further improved through Ant Colony Optimization (ACO) based optimization methodology. **Results:** In recent days, ACO plays a vital role in the discrete optimization problems. The ACO solves many engineering problems and provides optimal result which includes Travelling Salesman Problem, Network Routing and Scheduling. **Conclusion:** This study proposes an ACO based resource sharing algorithm for effective utilization of grid computing.

Key words: Grid computing, Ant Colony Optimization (ACO), resource sharing, Load Balanced Min-Min (LBMM), First Come First Serve (FCFS), Ant Colony System (ACS)

INTRODUCTION

As the scientific problem grows very complex in the modern computing technology, it requires more computing power and more storage space. Based on these basic requirements, an organization requires higher computational resource when dealing with current technological methodology. The past technologies such as distributed computing, parallel computing are not suitable for recent advancement. Because, the modern computer industry operating very large amounts of data which utilize more processing power and high storage volumes of data. Therefore, the Grid computing is proposed as effective resource management to the organization.

In grid computing, the network status and the resource status are to be managed effectively. If the network status or resource status are not infeasible level, then the total computation time will be increased dramatically. In grid computing, the user will encounter thousands of computers to utilize in effective and efficient manner. The Grid architectures serving as a middleware technology for various purposes likes resource allocation management, job scheduling, data management, security and authorization. Programming in the grid computing involves more complexities which is not only requires a single-machine application. Some of the additional aspects of the grid computing are (1) Dividing and combining data and results, (2) Data security, (3) Application security, (4) Testing and (5) Redundancy and capacity planning.

The purpose of task scheduling in the grid computing is to balance the load of the entire grid system in such a way that completing all the assigned workload as soon as possible and feasible than other system. It is impossible for anyone to manually assign these loads in the large computing resources of the grid system. As the environmental status of grid architecture is changing frequently, the traditional job scheduling algorithm such as ‘‘First Come First Serve’’ (FCFS), ‘‘Shortest Job First’’ (SJF), may not be suitable for the dynamic environment in grids. Therefore, job scheduling in the grid environment is a very important issue. This study proposes an efficient job scheduling algorithm for solving these problems in the grid environment.

Related work: As the Grid is growing in the modern era, it attracts researcher. There are a variety of research activities is identified in the grid environment, in which the task scheduling and load balancing are the major research issue even till date. There are many scheduling algorithms for grid task is

Corresponding Author: Ramesh, D., Department of CSE, Anna University of Technology, Tiruchirappalli, India

proposed in the recent year; the scheduling is classified in a variety of ways. The job scheduling method is composed (Berrichi *et al*., 2010) of the following major steps:

- The user submits a new job to the system
- Based on the predicted finishing time, the resource broker selects the "best" computing element in its execution and the storage element (if the chosen computing element does not provide enough disk space for the input files) the input files must be replicated to
- Agents are sent to every source Storage Element and one to every destination node
- The agents will run on the nodes in the background as daemons (the environment for this will be assured by the agent hosts) and copy the files prior to the execution queue reaching the job requiring them

Only few of scheduling algorithms for grid task are focusing the problems with a variety of QoS parameter. (Mohan and Baskaran, 2010) proposed an ACO based task scheduling which is termed as Ant Colony System (ACS) for Grid computing, in which the authors considers the scheduling of tasks in terms of more than one Quality of Service (QoS) requirements using ACO, which is challenging and also it is significantly influences the performance of grids. The proposed ACO is enabling the users to specify their QoS preferences as well as define the minimum QoS thresholds for a certain application and the ACS is tested in ten task applications with at most 120 tasks. The architecture of ACS task scheduling is shown in the Fig. 1 and the system design of ACO is shown in the Fig. 2. This ACO decreases the cost by 10-20% compared with the existing deadline based approach.

Chang *et al*. (2009) proposed a Balanced Ant Colony Optimization (BACO) algorithm for job scheduling in the Grid environment. The main contributions of this study is to balance the entire system load while trying to minimize the make span of a given set of jobs, the BACO focuses on the make-span and system load balance. Compared with the other job scheduling algorithms, BACO can outperform them according to the experimental results.

In apples (Mohan and Baskaran, 2011a), the Parameter study support, event-driven rescheduling, Centralized adaptive scheduling with heuristics and self-scheduled work queues are handled. In EZ-GRID broker (Mohan and Baskaran, 2011b) job handling, transparent file transfer, self-information service with dynamic and historical data, Policy Engine Framework for provider policies are proposed. In GRID BUS Grid Service system (Mohan and Baskaran, 2011c; 2011d), Failure management and application recovery, parameter studies, API support, Economy-based and data aware scheduling are focused for solving. The GRUBER (Suguna, 2011) handles SLA-based resource sharing in multi-VO environment, disk quota considerations, internal site monitoring feature and various users' oriented policies.

Proposed work:

Overview of Ant Colony Optimization (ACO): Swarm intelligence is a new discipline of study that contains relatively optimal approach to problem solving which is the imitations inspired from the social behaviors of insects and of other animals, for ex: Ant colony optimization algorithm, artificial honey bee algorithms. The main idea of ACO is to model the problem as the search for a minimum cost path in a graph. Artificial ants walk through from nest to food, looking for good paths. Each ant has a rather simple behavior so that it will typically only find rather poor-quality paths on its own. Better paths are found as the emergent result of the global cooperation among ants in the colony. The behavior of artificial ants is inspired from real ants, they lay pheromone trails on the graph edges and choose their path with respect to the probabilities that depend on pheromone trails and these pheromone trails progressively decrease by evaporation.

In addition, artificial ants have some extra features that do not find their counterpart in real ants. In particular, they live in a discrete world and their moves consist of transitions from nodes to nodes. Also, they are usually associated with data structures that contain the memory of their previous actions. In most cases, pheromone trails are updated only after having constructed a complete path and not during the walk and the amount of pheromone deposited is usually a function of the quality of the path. Finally, the probability for an artificial ant to choose an edge often depends not only on pheromones, but also to some problem-specific local heuristics. The detailed survey on ACO is available in (Dumitrescu and Foster, 2005) for various engineering optimization problems.

Casanova *et al*. (2003) developed a model of the observed behavior, in which there are four units (A_1, A_2, A_3 and A_4) and two routes (R_1 and R_2) leading to a food source (F_0), where R_1 and R_2 such that $R_1 > R_2$ and $R_1=2*R_2$. Initially, all units are at the decision point N_e and they have to select between $R_1$and R_2 to reach F_o.

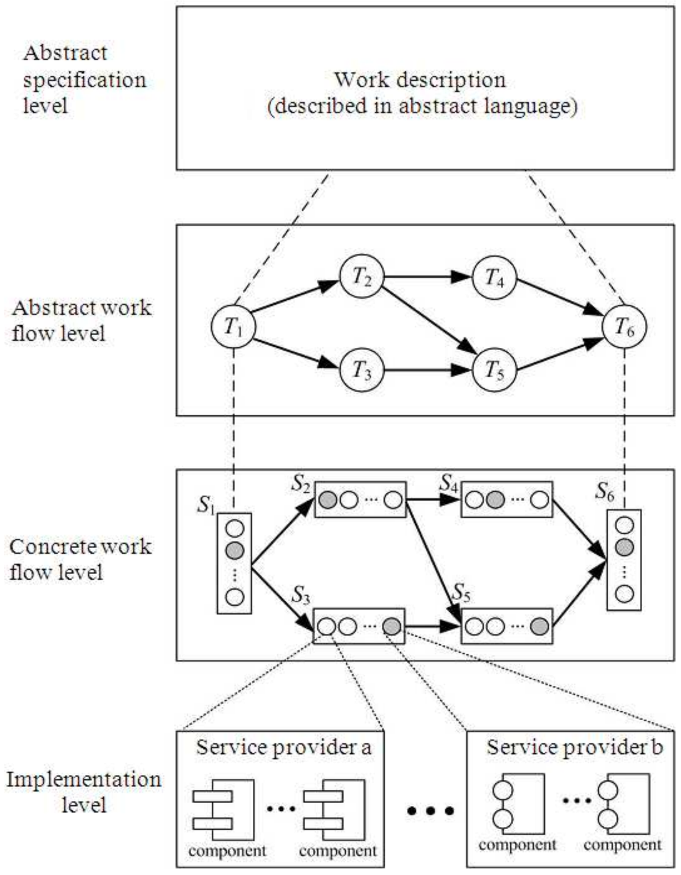

Fig. 1: Architecture of task scheduling in ACS

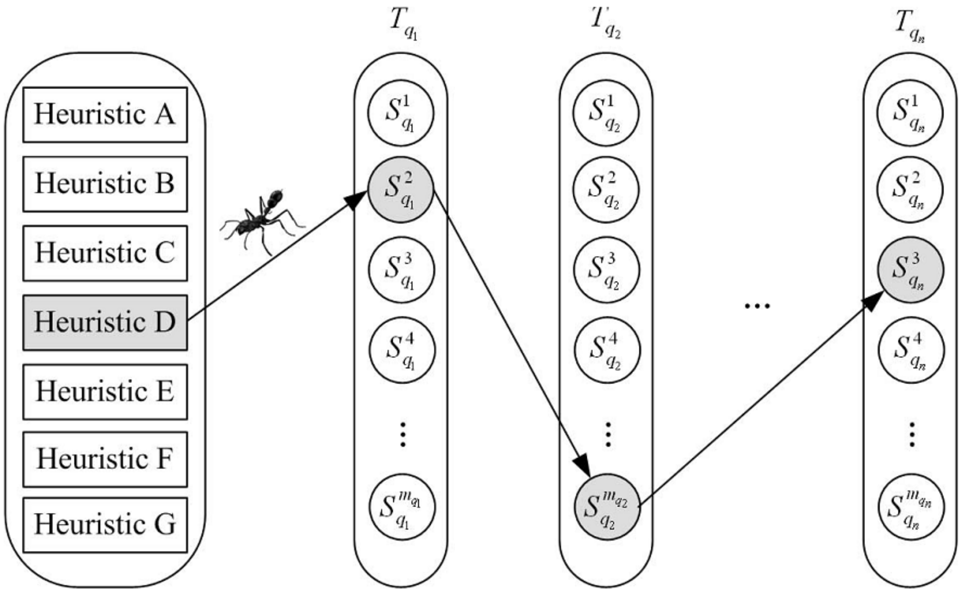

Fig. 2: System design of Ant flow in ACS

At N_e, all units have no knowledge about the location of food (F_0). Hence, they randomly select from {R1, R_2}. Suppose that A_1 and A_2 choose R_1 and A_3 and A_4 choose R_2.

As A_1 and A_2 move along R_1 and A_3 and A_4 move along R_2, they leave a certain amount of pheromone along their paths τ_{R1} and τ_{R2}, respectively.

Since $R_1 > R_2$, A_3 and A_4 reach F_0 before A_1 and A_2. When A_3 and A_4 pass R_2 to reach F_0, $\tau_{R2} = 2$, but A_1 and A_2 have yet to reach F_0 and $\tau_{R1} = 0$. To return to N_e from F_0, A_3 and A_4 have to choose between R_1 and R_2. At F_0, A_3 and A_4 detect that $\tau_{R2} > \tau_{R1}$, hence they are more likely to select R_2.

As A_3 and A_4 pass R_2 for the second time to reach N_e, τ_{R2} is incremented to 4. The increase in τ_{R2} further consolidates R_2 as the shortest path. When A_1 and A_2 reach F_0, $\tau_{R2} = 4$ and $\tau_{R1} = 2$. Hence, A_1 and A_2 are more likely to select R_2 to return to N_e.

In this example, any ant at F_0 (respectively, N_e) will be able to determine the optimal path once A_3 and A_4 reach F_0. If an ant is at a choice point when there is no pheromone (e.g., Initially at N_e), it makes a random decision with a probability of 0.5 of choosing R_1 or R_2. However, when pheromone is present (e.g., When the ant is at F_0), there is a higher probability that it will choose the path with the highest concentration of pheromone.

ACO based task scheduling: Job scheduling problems have a vital role in recent years due to the growing consumer demand for variety, reduced product life cycles, changing markets with global competition and rapid development of new technologies. The Job Shop Scheduling Problem (JSSP) is one of the most popular scheduling models existing in practice, which is among the hardest combinatorial optimization problems. The instance definition of the job scheduling problem is as follows:

- A number of independent (a user / application) jobs to be scheduled
- A number of heterogeneous machines candidates to participate in the planning
- The workload of each job (in millions of instructions)
- The computing capacity of each machine (in mips)
- Ready time indicates when machine m will have finished the previously assigned jobs
- The Expected Time to Compute (ETC) matrix (‘nb’ jobs × ‘nb’ machines) in which ETC[i][j] is the expected execution time of a job ‘i' in machine ‘j

Many approaches, such as, Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithm (GA), Ant Colony Optimization (ACO), Neural Network (NN), Evolutionary Algorithm (EA) and other heuristic approach have been successfully applied to JSSP. (Pasteels *et al*., 1987) proposed an efficient hybrid algorithm for resource-constrained project scheduling. This hybrid algorithm is known as the ACOSS algorithm which combines Scatter Search (SS) with ACO. Research on ACO has shown that improved performance can be obtained by stronger exploitation of the best solutions found during the search (Komarudin and Wong, 2010; Kacsuk *et al*., 2007). Yet, using a greedier search potentially aggravates the problem of premature stagnation of the search. Therefore, the key to obtaining better performance of ACO algorithms is to combine an improved exploitation of the best solutions with an effective mechanism for avoiding early search stagnation. Combining exploitation of the best solutions to a problem-dependent local search algorithm, the authors presents the hybrid algorithm.

In this study, as a first step, all ants in the ACO search the solution space and generate activity lists to provide the initial population for the SS. Then, although the SS improves all the ants’ solutions, only the best solution (thus far) is used to update the pheromone trails. Finally, ACO searches the solution space again using the new pheromone trails. In other words, the SS uses the previous population constructed by ACO, which subsequently updates the pheromone trails using the best solution from the SS and searches again. In addition, a local search strategy is employed to improve the quality of solutions generated by ACO and also as the improved method in the SS. In this scheme, ACO and SS alternatively and cooperatively search the solution space until the termination criterion is satisfied. In each generation, ACO only executes once to generate the initial population for the SS, which then executes one or more times to improve the solutions. The proposed ACO based Task Scheduling (ACOWS) is explained in the following procedure.

ACO based Task Scheduling (ACOTS) algorithm: ACO is proposed for task scheduling of the grid system. The objective of the scheduling algorithm is to minimize the makespan of task. This can be calculated using Expected Time to Compute (ETC) model. The scheduling problem can be defined as follows:

Let task set are T = t1, t2, t3, …. , tn be the group of tasks submitted to the scheduler. Let Resource set are R = m1, m2, m3, …. , mk be the set of resources available at the time of task arrival. Makespan produced by any algorithm for a schedule can be calculated in the following Eq. 1-6:

$$\text{makespan} = \sum_{\text{mimize}} (\text{CTij})(\text{T(ti)}, \text{R(mj)})) \quad (1)$$

$$\text{CTij} = \text{Wj} + \text{ETij} \quad (2)$$

where, CT is the completion time of machines, ETij is the execution time of a job i on resource j, Wj is waiting time of resource j after completing the previously assigned jobs.

The scheduling is defined based on a proactive design which stored in a table called scheduling table, this can be constructed based on the state transition rule and pheromone update policy. In every t unit of times, the forward ant is generated and forwarded to collect the information about the grid systems. The forward ants will collect the information of currently running processes (traffic) and the expected time of completion. The information is stored in the scheduling table. The table contains next optimal resource allocation and also other feasible allocations. From the table, the optimal resource is selected or the load is shared into many feasible resources. The optimal load sharing is explained in the following mathematical models.

The following random proportional rule is applied as State transition rule: For choosing task ti and the probability of selecting a grid / resources mj is:

$$\text{prob}(D, i, j) = \text{Fun}(TD, i, j, \eta) - - - - \text{if}, j \in R \quad (3)$$

where, TD is the pheromone value corresponding to resource j for the task i and 0 < TD < 1 is the local heuristic value. Fun (TD, i, j, η) is a function in TD and η (this function value is high when TD and η are high). Assuming that at a given moment in time, 'm1' ants have used the first resource and 'm2' the second one, therefore the probability p1 for an ant to choose the first bridge is:

$$\text{Fun}(TD, r, s) = \begin{cases} \dfrac{T(r,s) \bullet [\eta(r,s)]^{\beta}}{\sum T(r,s) \bullet [\eta(r,s)]^{\beta}} \rightarrow \text{if} \ldots \\ \text{resource...available} \\ 0 \rightarrow \text{otherwise} \end{cases} \quad (4)$$

where, T(r, s) is the pheromone deposited in the path between 'r and s, η(r, s) is the corresponding heuristic value. β is a parameter which determines the relative importance of pheromone versus execution time (β > 0). The pheromone update policy is as follows:

$$T(r,s) \leftarrow (1-\alpha) \bullet T(r,s) + \sum (1-\alpha) \bullet T(r,s) \quad (5)$$

$$\Delta T_K(r,s) = \begin{cases} \dfrac{1}{CT_K} \rightarrow \text{if} \ldots \text{resource} \ldots \text{found} \\ \\ 0 \rightarrow \text{otherwise} \end{cases} \quad (6)$$

The proposed ACO based Task Scheduling (ACOTS) Algorithm:

```
(A)  Procedure of (ACOTS)
       For all tasks Ti
          For all resources
             $\min \text{imize} \sum_{(i,j)} (\text{CTij}(T(ti), R(mj)))$
             Do until all tasks are mapped
                For each task find the earliest
       completion time and the resource that obtains
       it
                     find the task Tk with the
       minimum earliest completion time
                     assign task Tk to the
       resource Rl that gives the earliest completion
       time
                     delete task Tk from the list
                     update ready time of
       resource Rl
                     update Cil for all i
                End for
                // rescheduling to balance the load
       using TS-(ACO)
             End-do
          End-for
       End-for

(B)  Procedure of TS-(ACO )

     //Initialization Phase
     For every unit of time,
          For each test pair (T, R)
               $\tau$ (t, m) := $\tau_0$
End-for
For k: = 1 to m do
               Let (t1, $m_1$) be the initial solution for
     an ant k
               CT($m_{k,}$ $t_{k1}$) := {1, …, n} – CT($t_{k1}$)
               CT($t_k$) := CT($t_{k1}$)
          End-for
          //This is the phase in which ants build
          their tours (the tour / node is referred to
          finding an optimal resource)
          For i: = 1 to n do
            If i < n
              Then
                For k: = 1 to m do
                Choose the next resource
                $m_k$
                $CT( m_{k,} t_k) := CT (m_{k,} t_k) –$
        $CT(m_k)$
                $Scheduling\_Table (m_j): =$
        $min(CT(t_{k,}\ m_k))$
            End-for
        Else
            For k: = 1 to m do
                $CT(m_k ) := CT(m_{k1})$
                $Scheduling\_Table (mi): =$
                $min(CT(t_k , m_k ))$
            End-for
        End-if
    For k: = 1 to m do
      Update
$CT(m,t) \leftarrow (1-\alpha) \bullet CT(m,t) + \sum (1-\alpha) \bullet CT(m,t)$
    End-for
    End-for
    //In this phase global updating occurs and
    pheromone is updated
    For k := 1 to m do
      Compute $CT_k$
    End-for
    Compute $CT_{best}$
    For each edge (m, t)
        $CT(m,t) \leftarrow (1-\alpha) \bullet CT(m,t) + \sum (1-\alpha) \bullet CT(m,t)$
    End-for
```

CONCLUSION

The proposed algorithm is tested using MatLab based simulation and programming models. The proposed ACO method provided the optimlaity in grid environment. The scheduling of the grid is smoothened using ACO. As a consequence, the load on various grids are shared among all available grids and it offers load balancing in the grid environment.

REFERENCES

Berrichi, A., F. Yalaoui, L. Amodeo and M. Mezghiche, 2010. Bi-Objective Ant Colony Optimization approach to optimize production and maintenance scheduling. Comput. Operat. Res., 37: 1584-1596. DOI: 10.1016/j.Cor.2009.11.017

Casanova, H., F. Berman, G. Obertelli and R. Wolski, 2003. The apples parameter sweep template: User-level middleware for the grid. Proceedings of the Proceedings of the 2000 ACM/IEEE conference on Supercomputing, Nov. 04-10, IEEE Computer Society, Dallas, Texas, pp: 60-60. DOI: 10.1109/SC.2000.10061

Chang, R.S., J.S. Chang and P.S. Lin, 2009. An ant algorithm for balanced job scheduling in grids. Future Generat. Comput. Syst., 25: 20-27. DOI: 10.1016/j.future.2008.06.004

Dumitrescu, C.L. and I. Foster, 2005. GRUBER: A Grid Resource Usage SLA Broker. Proceedings of the 11th International European Parallel Computing Conference, (IEPCC' 05), CiteSeerX, pp: 465-474.

Kacsuk, P., T. Fahringer and Z. Nemeth, 2007. Distributed and Parallel Systems: From Cluster to Grid Computing. 1st Edn., Springer, New York, ISBN-10: 0387698574, pp: 222.

Komarudin and K.Y. Wong, 2010. Applying ant system for solving unequal area facility layout problems. Eur. J. Opera. Res., 202: 730-746. DOI: 10.1016/j.ejor.2009.06.016

Mohan, B.C. and R. Baskaran, 2010. Improving Network Performance using ACO Based Redundant Link Avoidance Algorithm. Int. J. Comput. Sci., 7: 27-35.

Mohan, B.C. and R. Baskaran, 2011a. Energy aware and energy efficient routing protocol for adhoc network using restructured artificial bee colony system. High Perform. Architec. Grid Comput., 169: 473-484. DOI: 10.1007/978-3-642-22577-2_65

Mohan, B.C. and R. Baskaran, 2011b. Reliable Barrier-Free Services (RBS) for heterogeneous next generation network. Adv. Power Elec. Instru. Eng., 148: 79-82. DOI: 10.1007/978-3-642-20499-9_13

Mohan, B.C. and R. Baskaran, 2011c. Reliable transmission in network centric military network. Eur. J. Sci. Res., 50: 564-574.

Mohan, C.B. and R. Baskaran, 2011d. Survey on recent research and implementation of ant colony optimization in various engineering applications. Inter. J. Comput. Intell. Sys., 4: 566-582.

Pasteels, J.M., J.L. Deneubourg and S. Goss, 1987. Self-organization mechanisms in ant societies (I): Trail recruitment to newly discovered food sources. Birkhauser, 54: 155-175.

Suguna, N., 2011. An independent rough set approach hybrid with artificial bee colony algorithm for dimensionality reduction. Am. J. Applied Sci., 8: 261-266. DOI: 10.3844/ajassp.2011.261.266