Grid Scheduling using Differential Evolution (DE) for solving multi-objective optimization parameters

Mrs. A.R. Jayasudha MCA, M.Phil
Lecturer,
Department of Computer Applications,
Hindusthan College of Engg & Tech,
Coimbatore, India

Dr. T. Purusothaman, M.E., Ph.D.,
Assistant Professor,
Department of CSE,
Government College of Technology,
Coimbatore, India

***Abstract*—The computational grid is a collection and aggregation of parallel, distributed, and heterogeneous resources. Grid Scheduling is the complex issue to manage the heterogeneous resources. The proposed approach considers the evolutionary algorithm of Differential Evolution (DE) technique in a modified manner to solve the multi-objective parameters of makespan and flowtime. The proposed grid scheduling approach completes the jobs within minimal time and also it increases the utilization of resources. The proposed DE based grid scheduling algorithm with modified has been tested under the batch mode and the performance of the proposed MDE based algorithm has been compared with Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), and the results outperform the compared one.**

Keywords - ***Grid Scheduling, Differential Evolution, Makespan, Flowtime, Grid Scheduler***

1. INTRODUCTION

Grid is emerging as a wide-scale infrastructure that promises to support resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organization [8]. Grid computing aggregates the resources from multiple computers in a network for a single problem at the same time usually to a scientific or technical one that requires a large number of computer processing cycles or access to large amounts of data. A computational grid is the combination of distributed resources such as personal computers, workstations, supercomputers, processors, clusters, and scientific instruments have emerged as a next generation computing platform for solving large-scale problems in science, engineering, and commerce. Job Scheduling and Resource Management are the critical issue in Grid Computing [9]. It is a big challenge to design an efficient Grid Scheduling algorithm in computational grid. The above grid scheduling problem is multi-objective in nature, the two most important objectives considered in our proposed approach is minimization of makespan and flowtime of the resources.

In the heterogeneous nature of grid the job has to wait in the queue, the waiting time of the job in the queue depends on the following factors such as load and availability of the resources. There are lots of jobs submitted by the user and the submitted jobs have been categorized based on the user requirements such as number of nodes, estimated execution time, and specific input/ output needs. The grid scheduler is responsible for making scheduling decisions to allocate the jobs to the resources in an optimal way. To allocate the resources in an optimal manner, the grid scheduler needs the required information such as size of the job, priority of the jobs and estimated execution time of the job. The way in which the jobs have been prioritized and the prioritized jobs have been allocated to the appropriate resources is called Grid Scheduling. The above grid scheduling problem is called NP-Complete and it is in multi-objective nature. The efficient grid scheduling algorithm minimizes the average completion time of jobs through optimal job allocation on each grid node. The two most important objectives considered in our proposed approach are makespan and flowtime. The remainder of the paper has been organized as follows. The related work has been described in section 2, generic grid scheduling and grid scheduler performance factors and the implemented grid scheduler has been described in section 3, introduction to DE and proposed DE has been given in section 4, implementation details has been discussed in section 5, experimental results has been explained in section 6, conclusion and future work has been explained in section 7.

2. Related Work

Job scheduling is known to be NP-complete [10] and there are lot of meta-heuristics techniques have been examined job scheduling approach. A heuristic approach proposed by Lei Zhang, Yuehui Chen, Bo Yang [1] based on particle swarm optimization is adapted to solving scheduling problem in the grid environment. Each particle is represented a possible solution. The approach aims to generate an optimal schedule so as to get the minimum makespan and maximum resource utilization while completing the jobs. The hybrid particle

swarm optimization algorithm was proposed by M. Fikret Ercan [2] for the application of PSO in scheduling hybrid flow shops with multiprocessor jobs. In order to improve the performance of PSO, hybrid techniques were employed. The experimental results show that the PSO and hybrid methods are more efficient and effective in scheduling basis. Ritchie and Levine [3] have combined an Ant Colony Optimization algorithm with a TS algorithm for the problem. Abraham et al. [4] have proposed hybridization of Genetic Algorithm, Simulated Annealing and Tabu Search heuristics for dynamic job scheduling on large-scale distributed systems. They have combined the local search heuristics such as Tabu Search (TS) and Simulated Annealing (SA) that deals with a single solution at a time. Braun et al [11] & Maheswaran et al [12] have defined the simple heuristics for dynamic matching and scheduling of a class of independent jobs onto a heterogeneous computing. There are lot of research works has been carried out related to our work using Differential Evolution (DE) [13] technique, in this the chromosome has been represented as a feasible schedule that should be converted to a string of real numbers for DE operations, their proposed approach creates the infeasible solutions.

3. Introduction to Grid Scheduling

3.1 Grid Scheduling

Grid scheduling [6] is the process of scheduling applications over grid resources under different administrative domains. Grid Scheduler is responsible for implementing the efficient Grid Scheduling Algorithms.

The Grid Scheduling has been classified into three main phases.

- Resource Discovery – In the first phase of grid scheduling, resource discovery returns the list of capable resources available in the grid environment.
- Resource Selection – In the second phase of grid scheduling, resource selection retrieves the exact resource for job execution has been selected from the list of capable resources.
- Job Execution – In the third phase of grid scheduling, job execution involves the submission of jobs and monitors the job execution.

3.2 Grid Scheduling Performance Factors

There has been lot of optimization criteria can be considered for the grid scheduling problem and the problem is multi-objective in nature. In our proposed approach we have classified the performance factors into grid resource performance parameters and scheduling optimization criteria. The grid resource performance related factors are CPU utilization of Grid resources, Load Balancing, Queuing Time, Throughput, Turnaround time, Waiting Time, Response Time, Deadline, User Priority, and etc. The scheduling optimization criterion related factors are Makespan, Flowtime, Resource Utilization, Load Balancing, Turnaround Time, Total Completion time and Total Response time. The combination of grid resource performance factors and scheduling optimization criterion related factors improve the overall grid resource performance.

3.3. Implemented Grid Scheduler

The implemented grid scheduler architecture has been classified into two layers such as Broker Layer and Middleware Layer. The grid scheduler retrieves the user request using the Request Handler The request handler parses the resource information which is available in the resource repository. Once the resource information has been matched with the user request, the matched resource information has been sent to the Job Dispatcher. The job dispatcher maintains the job queue which contains the dispatched and undispatched jobs. The Differential Evolution (DE) based resource allocator has been periodically invoked by the dispatched for allocating the resources to jobs in an optimal manner. Once the resource has been scheduled, the dispatcher invokes the Transfer Manager for transferring the input files and executable to the scheduled resource. Once the input file and executable has been transferred successfully the dispatcher invokes the Execution Manager to invoke the job execution. The job execution has been periodically monitored by the execution manager and the job information has been updated and sent to the Request Handler.

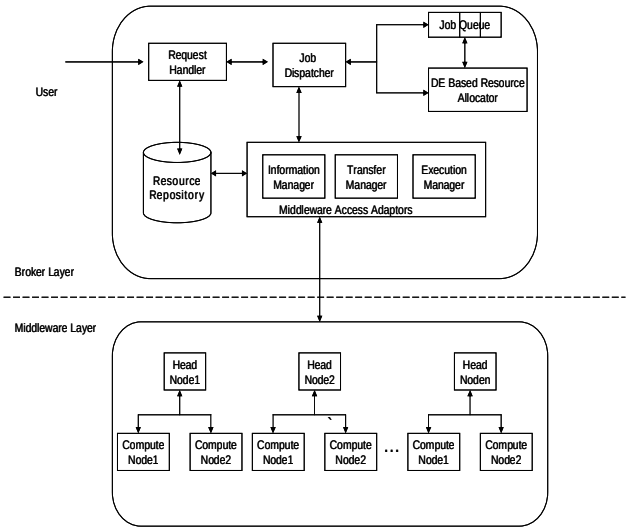

Figure. 1. Grid Scheduling Architecture

4. Differential Evolution Algorithm for Grid Scheduling

Differential Evolution is a Stochastic Direct Search and Global Optimization algorithm and it has been emerged from the field of Evolutionary Computation. It has been closely related to the algorithms such as Genetic Algorithm and Evolutionary Programming and also it shows some similarities to Particle Swarm Optimization (PSO).Differential evolution algorithm (DE) has been first introduced by R.Storn and K.Price in 1995 which is basically a random parallel searching algorithm [7]. The idea of differential evolution algorithm is to obtain a new individual by adding the weighted

difference vector of any two individuals to another individual with certain rules. Suppose if the value of the fitness has better than another individual the new fitness value will replace the existing value else the last fitness value kept for next generation.

4.1 Proposed Differential Evolution Algorithm

The evolutionary theory of the DE algorithm is that chromosomes have special encoding format are decoded for the fitness value of each chromosome of a generation. The fitness value has been saved and used a reference value for the next generating population. The population has been evolved by using crossover operation and mutation operation, decoding operation and selection operation. Once the above process has been completed within certain times the chromosome in a population which corresponds to the best fitness value has been used as a sub-optimal solution for grid scheduling problem.

4.2 Problem Formulation

To formulate the problem the following information needs about the resources in the computation grid environment such as computational load of each job running in the grid resources, computing capability of all the resource, estimation of the computational requirements of each job and the computational load of each resource. From the available resource information of the grid resources the estimated execution time of the job has been calculated by using an Expected Time to Compute (ETC) model and Expected Time to Compute matrix ETC has been constructed and the expected execution time ETij of job ti on resource mj has been defined as the amount of time taken by the resource mj to execute the given job ti by considering the above resource information. Let S be the set of jobs in our testing purpose for testing our grid scheduling algorithm and si be the starting time of the job .By using the above two we can compute the completion time of job by using the following notation CTij = si + ETij where CTij be the completion time of the whole job. Let us assume that the schedule S from the set of all possible schedules Sched.

Algorithm 1 Pseudo Code for DE based Grid Scheduling Algorithm

Let TResources be the total number of grid resources
Let n be the total number of jobs available in queue for a particular schedule
Let ATime be the available time of the grid resources
Let STime be the start time of the job in the grid resources
Let PSize be the population size
Let SFactor be the scaling factor
Let NIter be the number of iterations to be consider
Generate the initial population of random individuals
Match the feasibility for the initial vectors
    For 1 to NIter
    Compute the makespan value for each individual
    For I = 1 to PSize
Select the random integer number rand □ (0, 1, 2,)
Select mutually exclusive random individuals Xa, Xb, and Xc
Calculate the mutant vector V starting from the position rand of each individual.
Select the random value rand □ [0, 1]
Calculate the path vector Ui
Check the feasibility of path vector Ui
end for
Calculate the makespan of path vector set
for i = 1 to PSize
if makespan of Ui is less than Xi then
    Select Ui
else
Preserve Xi
end if
    end for
Select the solution with minimum makespan
end for

4.3 Chromosome encoding

In our proposed approach we have used the double-chromosome encoding principle which has been used for individuals. The first layer has been processed according to the jobs followed by numbered 1, 2, 3… the priorities of the jobs number implied the process constraint of the jobs. The second layer will generate a chromosome and the code number is the number of all jobs L and all codes have been retrieved by using Random Number. The random numbers are used to represent the priority of the job. The coding schemes will generate chromosome faster and easier to use.

A schedule of n independent jobs has been executed on TResources has been expressed as

S=S1, S2, S3….Sn ---------- (1)

Si □ 1, 2…TResources ----- (2)

The value at i in S represents the resource on which ith job has been scheduled in schedule S. The differential evolution has been used for problem encoding real vectors, real coordinates has been used instead of discrete resource numbers.

4.4 Differential operation

The differential operation mainly includes initialization, mutation, crossover, and selection operations. Mutation operation has been based on three individuals that have been randomly selected from the current population by using one individual the disturbance has been deal by other two individuals. Crossover operation has been performed based on the new individual which has been generated by come together the two random individuals which has been randomly selected from the current population. Selection operation has been performed by using the natural selection, preserve the best, and realize the evolution of populations. In Initialization operation chromosome is composed of random [0, 1] which length is L and NP individuals form a population. The local optimum value has been avoided in the mutation operation by using the principle of inter-district operation. In

selection operation the chromosomes have been ordered based on the fitness value into good, medium and bad. The ratio of g: m: b chromosome has been randomly selected for those three sub-populations where g represents good, m represents medium and b represents bad.

4.5 Decoding Operation

In decoding operation the populations has been decoded to retrieve the fitness of chromosomes. In the decoding operation the capacity of every resource has been evaluated. Decoding operation is a process that has been used for satisfying the process constraint and search for period of time that can be allocated to the job.

4.6 Fitness function for makespan and flowtime computation

In our proposed grid scheduling algorithm we have consider the scheduling optimization parameters as makespan and flowtime. The minimization of makespan has been responsible for executing the whole job within a minimal time and the minimization of flowtime has been responsible for utilization of computing resources in an efficient manner.

The above two criteria have been defined as follows:

Makespan: min {max Fj } j □ jobs

Si□ Schedulelength ---------- (3)

Flowtime : min{Fj}j□jobs ----- (4)

Si □ Schedulelength

Here Fj denotes the time when the job j finalizes, Schedulelength is the set of all possible schedules and jobs is the set of all jobs to be scheduled. Makespan is not affected by the particular execution order of jobs in a resource in order to minimize the flowtime the jobs should be executed in an increasing order of their estimated execution time. The makespan and flowtime are ambiguous objectives when we try to minimize one it will not suit the other.

In our proposed approach we have define the fitness function as follows:

$$Fit(S) = Schedulelength \rightarrow R \quad \text{--- (4)}$$

$$Fit(S) = \lambda 1 \cdot makespan(S) + (1 - \lambda 1) \cdot flowtime(S)/m \quad \text{---------- (5)}$$

The function Fit(S) is the sum of two objectives such as the makespan of schedule S and the flowtime of schedule S divide by number of resources m to keep both objectives in approximately the same magnitude. The weights assigned to makespan and flowtime in Fit(S) has been parameterized by the variable λ1.

5. Implementation Details

The algorithm has been implemented on a Pentium IV 2.3 GHZ PC, in Java programming language. The input for the algorithm are job identifier, number of nodes needed for the job, matched resources list in which the job can run, estimated cost of executing the job on the matched resources list, available number of free nodes. Then each cluster has been configured with the collection of computing nodes which can able to simultaneously execute the job in parallel manner. The allocation to jobs to available resources in an optimal manner using our proposed algorithm. The execution of jobs in a cluster has been take care by the local scheduler available in a cluster. The cost of executing a job on a resource includes the time taken to transfer the required input data of the job and the time taken to execute the job on the assigned resource. The exact execution time of each job on each of its matched resource may not be known prior to actual execution of the job on that resource. This information has been required for the proposed algorithm to obtain an optimum matching between the jobs and resources. The approximate execution time of the jobs has been obtained by using the history of jobs execution. The jobs execution time has been calculated based on the following characteristics such as jobs executed processor architecture, CPU cycles, amount of memory needed, number of nodes and etc, the estimated execution time has been computed. For each job, a list of resources where it can execute based on the requirements of architecture, memory, CPU cycles and number of nodes has been maintained. This list contains only those resources which have enough nodes as needed by that job by using these data, cost array in which total cost of execution of each job in each resource is stored.

6. Experimental Results

In our proposed algorithm we have implemented differential evolution for scheduling of independent jobs on heterogeneous grid environments. The results evolved from our modified Differential Evolution (DE) algorithm have been compared with different heuristic algorithms such as Genetic Algorithm (GA), Simulated Annealing (SA) and Particle Swarm Optimization (PSO). The experiment has been repeated for 10 times with different random numbers. Each experiment has 300 m X n iterations where n represents the grid resources and m represents the number of jobs. The m X n has been varied with the following two combinations such as follows:

- Less number of jobs n with less number of resources as m
- More number of jobs n with more number of resources as m

From the above two use cases makespan and flowtime values have been calculated. The makespan results of 300 iterations of use case 1 has been shown below in Table 1 and flowtime values has been shown in Table 2 The performance both makespan and flowtime value of the DE based Grid Scheduling algorithm outperforms the other algorithms. The makespan result of 300 iterations for use case 2 has been shown below in Table 3 and the flowtime results for use case 2 has been shown in Table 4.

Table 1 Makespan for GA, SA, PSO, and DE algorithms for use case 1

Algorithm	Makespan Average Value
Genetic Algorithm (GA)	41.82
Simulated Annealing (SA)	46.62
Particle Swarm Optimization (PSO)	44.42
Differential Evolution (DE)	42.82

Table 2 Makespan for GA, SA, PSO, and DE algorithms for use case 1

Algorithm	Flow time Value (in seconds)
Genetic Algorithm (GA)	28.0400
Simulated Annealing (SA)	32.0200
Particle Swarm Optimization (PSO)	30.0004
Differential Evolution (DE)	26.0100

Table 3 Makespan for GA, SA, PSO, and DE algorithms for use case 2

Algorithm	Makespan Average Value
Genetic Algorithm (GA)	41.36
Simulated Annealing (SA)	48.66
Particle Swarm Optimization (PSO)	42.42
Differential Evolution (DE)	40.54

Table 4 Flowtime for GA, SA, PSO, and DE algorithms for use case 2

Algorithm	Flow time Value (in seconds)
Genetic Algorithm (GA)	42.0400
Simulated Annealing (SA)	48.800
Particle Swarm Optimization (PSO)	44.400
Differential Evolution (DE)	40.000

It is evident from the data obtained by using DE based has given excellent results when compared to other techniques. The factor $\lambda 1$ has been set to 0.1 to 0.5 for the equal contribution of makespan and mean flowtime to the fitness value.

6. Conclusion and Future Work

In this paper we have discussed the important concepts from Grid computing related to grid scheduling problems and their resolution using heuristic based approach. The grid scheduling involves the optimization of multiple parameters such as completion time, resource utilization, minimization of total execution cost etc because of the optimization of multiple parameters the grid scheduling has been considered as a multi-objective problem. The optimization technique of Differential Evolution (DE) has been used for solving the multi-objective parameters in grid scheduling. The designed grid scheduling algorithm have been used and tested under batch mode. By using our proposed approach the grid scheduler has been able to reveal the complexity of the scheduling problem in Computational Grids and also it shows the effectiveness for the design of efficient grid scheduling algorithm.

References

[1] Lei Zhang, Yuehui Chen, Bo Yang ”Job Scheduling Based on PSO Algorithm in computational Grid”, 2006 Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications.

[2] M. Fikret Ercan, “A hybrid particle swarm optimization approach for scheduling flow-shops with multiprocessor jobs”, 2008 International Conference on Information Science and Security.

[3] G. Ritchie and J. Levine, “A fast, effective local search for scheduling independent jobs in heterogeneous computing environments” Technical report, Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh, 2003.

[4] Abraham A, Liu H, Zhang W, Chang TG, Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm, Proceedings of 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems, England, pp. 500-507, 2006.

[5] G. Ritchie and J. Levine, “A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments.” in 23rd Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG 2004), 2004.

[6] Schopf J.M, “A General Architecture for Scheduling on the Grid”, special issue of JPDC on Grid Computing, (2002).

[7] K. Price and R. Storn,”Differential evolution: Numerical optimization made easy”, Dr. Dobb’s Journal, pages 18–24, 1997.

[8] I. Foster and C. Kesselman, "The Grid - Blueprint for a New Computing Infrastructure" Morgan Kaufmann Publishers, 1998.
[9] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid", International Journal of Supercomputer Applications, 15(3), 2001.
[10] M.R. Garey and D.S. Johnson, "Computers and Intractability – A Guide to the Theory of NP Completeness" W.H. Freeman and Co., 1979.
[11] Braun T. D., Siegel H. J., Beck N., Bolony L., Maheswaram M., Reuther A. I., Robertson J.P., Theys M. D., Jao B., "A taxonomy for describing matching and scheduling heuristics for mixed-resource heterogeneous computing systems", IEEE Workshop on Advances in Parallel and Distributed Systems, (1998) 330–335.
[12] Maheswaran M., Ali S., Siegel H.J., Hensgen D., Freund R., "Dynamic Mapping of a Class of Independent Jobs onto Heterogeneous Computing Systems", 8th IEEE Heterogeneous Computing Workshop (HCW'99), San Juan, Puerto Rico, (1999) 30–44.
[13] A.C. Nearchou and S. L. Omirou, "Differential Evolution for Sequencing and Scheduling Optimization" Journal of Heuristics.