BEYOND SIMULATED ANNEALING IN GRID SCHEDULING

R. Joshua Samuel Raj
Assistant Professor, CSE, VV College of Engineering,
VV Nagar, Arasoor, Thisaiyanvillai

Dr. V. Vasudevan
Prof. & Head/IT, Kalasalingam University Srivilliputur, India

Abstract: In Grid Environment the number of resources and tasks to be scheduled is usually variable and dynamic in nature. This characteristic emphasizes the scheduling approach as a complex optimization problem. Scheduling is a key issue which must be solved in grid computing study and a better scheduling scheme can greatly improve the efficiency.The objective of this paper is to explore and investigate Simulated Annealing with limited iterations to promote compute intensive grid applications to maximize the Job Completion Ratio based on the comprehensive understanding of the challenges and the state of the art of current research. Experimental results demonstrate the effectiveness and robustness of the proposed algorithm. Further the comparative evaluation with other scheduling algorithms such as First Come First Serve (FCFS), Earliest Deadline First (EDF) is plotted.

Key words: grid computing, workflow, scheduling problem, Simulated Annealing, Memory

INTRODUCTION

Grid Computing a pioneer technique in harnessing the geographically dislocated computer power, has changed the perception on the utility and availability of the computer power, which has carved a new technology that openly ventures and amalgamates an infinite number of computing devices into any grid environment, augmenting to the computing capability and providing resolutions to the various tasks within the operational grid environment basically by enabling, sharing, selection and aggregation of geographically distributed autonomous resources dynamically at runtime, depending on their availability, capability, performance and cost, thereby shifting the focus to collaborative environments, federating services and exchanging transactions in a mutual manner to share resources and thereby achieve common goals to enhance productivity and speed up progress in much the same way that the Internet did in yesterdays economy, paving the way for numerous research efforts in grid scheduling mechanisms. Grid Computing is our greatest hope for delivering computing as utility to homes and offices.

Simulated annealing (SA) is a kind of global optimization technique based on annealing of metal. It can find the global minimum using stochastic searching technology from the means of probability. Simulated annealing is a concept courtesy to the way in which crystalline structures are formed into a more ordered state by use of the annealing process, which repeats the heating and slowly cooling a structure. Other names of Simulated Annealing are Monte Carlo annealing, Statistical cooling, Probabilistic Hill Climbing, Stochastic Relaxation, Probabilistic Exchange algorithm. The idea in SA to escape local maxima is to allow some bad moves but gradually decrease their size and frequency

PROBLEM DESCRIPTION

The Super Schedule (SSGA) Grid Architecture described with eight nodes Grid environment example is shown in the Fig 1. This architecture can be utilized for any practical applications for the normal grid environments. The setup is experimented in TIFAC Core in Network Engineering under DST project.

The goal of the SSGA is to find the allocation sequence of workflows on each Grid site. Four major entities are involved in this architecture.

- The grid users submit their request for job completion to the local grid managers.
- All the tasks should be received by the grid managers and the decision for the scheduling is made on deploying the request to the Intra Grid schedulers.
- The Intra-Grid schedulers have the updated information of the grid resources that are idle during time t. This information is frequently updated. The smaller jobs can be scheduled within their deadlines by the Intra-Grid schedulers in their respective Administrative Domains. Here scheduling is often dynamic.

- For data intensive applications where the jobs are larger it requires the necessity of the resources worldwide. At that moment, there is a necessity of Inter-Grid schedulers which is static often.

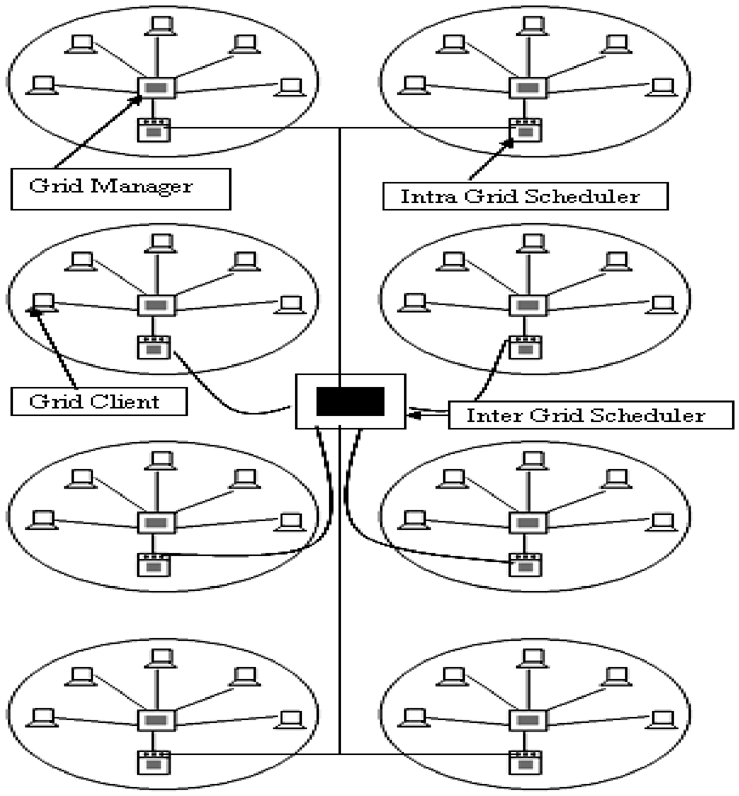

Fig 1: Super Schedule Grid Architecture

The workflow allocation strategy in a Grid environment differs from the traditional ones. The goal of the Inter-Grid Scheduler is to receive the request from different Intra-Grid Schedulers and make an optimistic scheduling such that it accommodates many workflows completing within its deadline. The following DAG workflows are considered for experimental purpose.

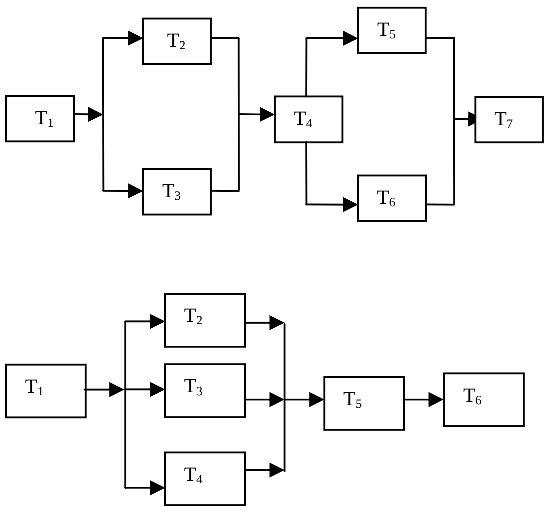

Fig 2: DAG workflow model

The settings of the experiment consist of workflows with following assumptions:

- Each workflow received in the Inter-Grid Scheduler consists of a set of Tasks T1, T2, and T3 and so on.
- The task in each workflow is a Directed Acyclic Graph (DAG) model. (Fig. 2)
- The output from a task can be transferred to other tasks as per the DAG graph model and all jobs are available at time zero.
- At any time a task can be executed only on a Grid site which is reported to the Inter-Grid scheduler as idle via Intra-Grid scheduler.
- There is no pre-emption of tasks or workflows.
- The sequential order of workflow allotment changes.

Here we present a scheduling approach for the wide area problem where in the resources and jobs are dispersed geographically. We have n jobs to be scheduled on m machines. Each machine cannot be simultaneously assigned to two jobs. Our main objective is to minimize the makespan.

SIMULATED ANNEALING

In the general optimization for Grid Scheduling we start with an initial configuration where in the neighborhood is exhaustively searched for and a neighbor is selected as a candidate. We evaluate some optimization function and accept the candidate if it is better else we proceed on to select another neighbor as candidate until some stopping criteria is reached. For further improvisation we find a method to find an initial configuration, a transition function to find a neighbor as candidate, an optimization function, an evaluation criteria and a stopping criterion. Simulated annealing uses a more complex optimization function. It sometimes accepts candidates with lower optimization values based on a probability function to escape from local minimum. It adapts the parameters of the evaluation function during execution. It is based on the analogy with the simulation of annealing of solids. Simulated annealing is an optimization algorithm that exploits an analogy between the annealing process and the search for the optimum in a more general system.

The annealing process is a two step wherein the first one corresponds to raising the temperature to a very high level (melting temperature) thereby the atoms posses' high energy state and a high possibility to rearrange the crystalline structure. The second step corresponds to cooling down slowly wherein the atoms have a lower and lower energy state and a smaller and smaller possibility to rearrange the crystalline structure. The physical system we considered so far is analogous to our optimization problem. The solution is analogous to the State (Configuration) of the system; the optimization function is analogous to the Energy in the system, the optimal solution to the ground state of the system, the iterative improvement to the rapid quenching of the system and the simulated annealing to the careful annealing of the system. In simulated annealing global optimal solutions can be achieved as long as the cooling process is slow enough. At very high temperatures Simulated Annealing explores the solutions space and at low temperature limits the exploration. The normal simulated annealing algorithm is as follows

Simulated Annealing algorithm:

```
Select an initial temperature $t_i$ (a large number)
Select an initial solution s
Select an optimization function f
Select a neighbourhood structure for the solution space
Select a temperature reduction function alpha( t )
Repeat
  Repeat
    randomly select s1 in N(s)
    diff = f(s1) - f(s)
    if diff< 0 then s = s1
    else generate random x in (0, 1) and if $x < e^{-diff/t}$
                            then s = s1
  until iteration count = max_number_iteration
  t = alpha(t)
until stopping condition
s is the approximation solution
```

The input of the algorithm is an initial solution which is constructed by assigning a resource to each task at random. There are several steps that the simulated annealing algorithm needs to go through while the temperature is decreased by a specific rate. The cooling schedule used in our paper is among the most popular ones where the temperature decrease is exponential and defined according to the following equation.

$$T_{n+1} = \alpha T_n$$

where $0 < \alpha < 1$.

To ensure the cooling schedule is sufficiently slow the parameter α should be given values close to unity. The control parameter decreasing rate is one of the most important factors in deciding the nature of the simulated annealing process. If the decreasing rate is too high, the process would be terminated quickly due to the stricter acceptance criteria. Thus, the so called “annealing” effect cannot be achieved, instead the “quench” effect would emerge, as the desirable solution cannot be obtained. If the decreasing rate is too low, the simulated annealing would require a very long process to terminate to a desirable solution state so that it becomes very inefficient or even prohibited.

The perturbation scheme used in our paper is pair wise exchange. The proposed perturbation scheme namely the pair wise exchange can be best explained with an example. Consider the workflow sequence 1, 2, 3, 4, 5 as a seed sequence and that the integers i, j (i, j <= w_n) are randomly generated. Suppose in the first instance i =2 and j=3 the pair wise exchange technique will generate the new sequence 1, 3, 2, 4, 5 which is the result of interchanging the second and third integers in the starting sequence. Consider ten workflow sequences, the effect pair wise exchange when i=3 and j=9 is shown below

Before pair wise exchange

W1, W2, $\underline{\text{W3}}$, W4, W5, W6, W7, W8, $\underline{\text{W9}}$, W10

After pair wise exchange

W1, W2, $\underline{\text{W9}}$, W4. W5, W6, W7, W8, $\underline{\text{W3}}$, W10

Here the optimization function we deal is with maximizing the number of workflows to be completed within the deadline. At each iteration, a neighbor s' N(s) of the current configuration s is generated by our pair wise perturbation scheme and a decision is taken to decide whether s' will replace s based on evaluation function. . If s' is a better than s ie $\Delta = f(s')-f(s) \leq 0$. We move from s to s', otherwise we move to s' with a probability $e^{-\Delta E/T}$

SA algorithm stops when a fixed number of non improving iterations are realized with a temperature or when a limit of iterations is reached. There exist theoretical schedules guaranteeing asymptotic convergence of the algorithm towards an optical solution. That is why much simple schedules are preferred even if they don’t guarantee an optimal solution

SOLUTON CONSTRUCTION

In simulated annealing a substantial amount of run time is required for a good optimal schedule to be found. This is because the algorithm spends a lot of time in doing repetitive iterations as the system is being cooled. We use the addition of memory scheme in our algorithm to construct the solution as a possible means to reduce the number of iterations by restricting the neighborhood structure as the temperature decreases. In SA sometimes the optimal solution that has been visited might be lost as the algorithm proceeds and also the algorithm might be spending a lot of time unnecessarily by exploring the search space with no feasible solutions. So by augmenting some memory to the simulated annealing algorithm we forcefully reduce the number of iterations and keep the solution space as small as possible.

The main characteristic of our proposed method is the use of a short term memory which keeps track of the best solution visited during the last n number of iterations and the simultaneous track of the best solution. As the temperature is decreased the solution accepted is always compared with the overall best solution. Only if the solution improves the quality of the overall best solution, it is added to the memory. The completion of the memory list marks the end of the iterations. At the end of the iteration the memory list is emptied and the best solution of the current solution is selected as the initial solution for the next iteration. In case of improvements the memory list is always lengthened else it is reduced by a factor that is proportional to the neighborhood size. The list is initially populated by randomly generated solutions which are gradually replaced by overall best solutions as the search progress.

We choose Simulated annealing with memory integration as shown in fig 3 in order to reduce the number of iterations as repeated annealing sometimes unnecessarily slows the whole process. In our approach, a pre-defined number of starting solutions are chosen from widely separated regions in the sample space, and used to obtain a set of locally optimal solutions. The belief is that a large enough set of locally optimal solutions collectively contain predominantly those features that are present in globally optimal solutions and rarely contain features that are absent in globally optimal solutions.

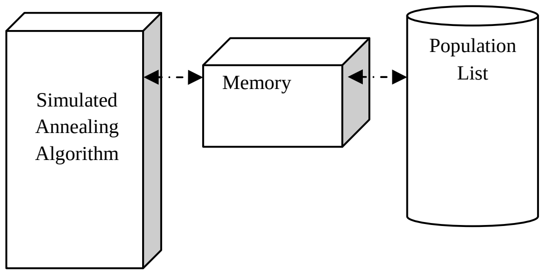

Fig 3: Simulated Annealing algorithm with memory

The solution state in our optimization problem is equivalent to states of a physical system, and the objective function value of a solution is equivalent to the energy of a state. The temperature controlling the whole annealing process is equivalent to a parameter that is called control parameter in our simulated annealing algorithm. Our Simulated annealing algorithm with augmented memory is as follows and the completion time is our only evaluation.

General Scheme of SA with augmented memory:

The structure of simulated annealing augmented memory algorithm is formalized as shown below and the main use of the augmented memory is to cache the optimization functions schedule (partial and complete) that have already been evaluated so that they can be retrieved from the cache rather than having to go through the evaluation function again.

Step 1 (start):

We start with a random solution or with a solution that has been heuristically built

Step 2 (Initialization):

We initialize the various parameters. Let C be the initial makespan, T_s be the maximum temperature, T_f be the final temperature, C_b be the best solution found so far in the memory initially being set to C, n(k) be the number of iterations to be performed before the algorithm terminates

Step 3 (Next solution):

Perturb randomly the current state to a new state. Create next solution state *j* from current solution state *i*. $f(j)$, $f(i)$ is objective function value for the States *j* and *i* . Compute the cost difference between the previous state and the current state based on the optimization criteria

Step 4 (Comparison of solutions):

```
If $\Delta \leq 0$ then
C'=C
Else
Generate a random number r     $U(0, 1)$
If $r \leq e^{\Delta/T_k}$ then
C'=C
Else
Retain S
Endif
Endif
If $C(S)<C(S_b)$ then
$S_b$=S and $T_b=T_k$(the temperature at which the best Solution is found)
Else
Retain S.
```

Step 5 (Termination):

Stop if k = n(k) or if there is a very small change in objective function values or no entries added in the memory for a long time. Then report the best solution as the best makespan C, otherwise, go to step 3.

RESULTS AND DISCUSSION

The methodology is such that an initial job sequence is selected at random among the set of job sequences and the objective function for the solution is defined as a best optimal schedule.

The comparative increase in the completion of workflows by Simulated annealing augmented with memory considering other algorithms such as FCFS, EDF and SA are shown in Fig 4 and Fig 5.

The parameters used in the algorithm are

Simulated annealing with memory parameters	
Parameters	Selection
Initial Temperature	499
Temp reduction factor	0.9
Final Temperature	90
Iterations	k
Memory Size	m * n/10

No of Workflows	**Average CPU Time in Milliseconds**			
	FCFS	**EDF**	**SA**	**SAWM**
10	5.8	5.4	7	5
20	6.7	6.7	10	6.3
30	10	9	11	9.2
40	29	27	26	24
50	63	68	60	53
60	88	83	85	77
70	96	97	90	87
80	117	113	107	101
90	137	133	117	109

Fig 4: Job completion time in milliseconds

It can be analyzed that Simulated Annealing with augmented memory outperforms ordinary SA in the number of workflow completions. As per the methodology SA augmented with memory succeeds the other scheduling mechanisms in consideration.

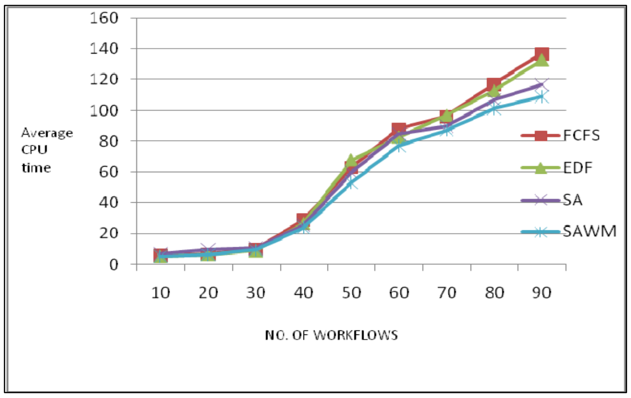

Fig 5: Average CPU time for FCFS, EDF, SA and SAWM

CONCLUSION AND FUTURE WORK

In this paper, we have applied Simulated annealing augmented with memory for the Generalized Grid Scheduling problem a hybrid variant of the ordinary simulated annealing to reduce the number of iterations and thereby reduction in time to obtain the optimal schedule. The simulation results demonstrate that the proposed algorithm has a high performance in the dynamic grid environment. Our computational experience shows us that this hybrid variant of simulated annealing always performs better than the ordinary simulated annealing owing to the augmented memory which restricts exploration in unnecessary search space. The procedures are tested with an example problem environment and the results are compared with that of other available scheduling procedures namely FCFS and EDF.

In the near future we plan to combine simulated annealing and Tabu Search along with sharing method to increase the efficiency. Similarly the ant colony properties can be included for scalability in the existing algorithm. The procedure can also suitably be modified and applied to any kind of Grid scheduling with different problem environment and optimize any number of objectives concurrently.

REFERENCES

[1] P. Shroff, D. Watson, N. Flann, R. Freund, Genetic Simulated Annealing for Scheduling Data-dependent Tasks in Heterogeneous Environments, in: 5th IEEE Heterogeneous Computing Workshop (HCW '96), Apr. 1996, pp. 98-104
[2] I. Foster and C. Kesselman, *The grid: Blueprint for a future computing infrastructure*, San Mateo, CA: Morgan Kaufmann, 1999.
[3] Hamscher, V., U. Schwiegelshohn, A. Streit and R. Yahyapour, 2000. Evaluation of Job Scheduling strategies for Grid comptuing, GRID 2000 first IEEE/ACM International workshop, Proc. Springer Pub., pp:191-202
[4] R. Buyya. Economic-based Distributed Resource Management and Scheduling for Grid Computing [D]. Melbourne, Australia: Monash University, April 12, 2002.
[5] X.S. He, X.H. Sun, G.Von Laszewski. A QoS Guided Scheduling Algorithm for Grid Computing. Journal of Computer Science and Technology, 2003, 18(4):442-451.
[6] A. Mandal, et al. "Scheduling strategies for mapping application workflows onto the grid", *in Proceedings of the 14th IEEE International Symposium on High Performance and Distributed Computing (HPDC-14)*, 2005, pp. 125-134.
[7] A. J. Page, T. J. Naughton. Dynamic Task Scheduling using Genetic Algorithms for Heterogeneous Distributed Computing. Parallel and Distributed Processing Symposium, 19th IEEE International Volume, Issue, 04-08 Apr 2005
[8] A. Afzal, J. Darlington, A.S. McGough, "QoS-constrained stochastic workflow scheduling in enterprise and scientific grids", *The 7th IEEE/ACM International Conference on Grid Computing*, 2006, pp. 1-8.
[9] S. Fidanova. Simulated Annealing for Grid Scheduling Problem. Modern Computing, 2006. IEEE John Vincent Atanasoff 2006 International Symposium on Volume, Issue , Oct. 2006 Page(s):41 – 45
[10] R. Joshua and V. Vasudevan, "Scheduling of workflows in grid computing with probabilistic Tabu Search", IJCSIS: Int. journal of computer science and Information Security, Vol. 8, No. 4, pp 314-319 July 2010

Name:
R. Joshua Samuel Raj

Affiliation:
Assistant Professor / CSE
VV College of engineering.

Brief Biographical History:
2005 -Graduated in 2005 from the Computer Science and Engineering Department from PETEC under Anna University
2007 -Received M.E Degree in Computer Science and Engineering from Jaya College of Engineering under Anna University

2009 Working towards the Ph.D degree in the area of Grid scheduling under Kalasalingam University
Main Works:
Grid computing, Mobile Adhoc Networking, Multicasting and so forth

Name:
V. Vasudevan
Affiliation:
Director, Software Technologies Lab, TIFAC
Core in Network Engineering,
Srivilliputhur, India

Brief Biographical History:
1984- M.Sc in Mathematics and worked for several areas towards
Representation Theory

1992 Received his Ph.D. degree in Madurai Kamaraj University

2008- the Project Director for the Software Technologies Group of TIFAC Core in Network Engineering and Head of the Department for Information Technology in Kalasalingam University, Sirivilliputhur, India
Main Works:
Grid computing, Agent Technology, Intrusion Detection system,
Multicasting and so forth