Simulated Annealing for Grid Scheduling Problem

Stefka Fidanova
Institute of Parallel Processing-BAS, Acad. G. Bonchev str. Bl. 25A, 1113 Sofia, Bulgaria
E-mail: stefka@parallel.bas.bg

Abstract

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. The goal of grid tasks scheduling is to achieve high system throughput and to match the application need with the available computing resources. This is matching of resources in a non-deterministically shared heterogeneous environment. The complexity of scheduling problem increases with the size of the grid and becomes highly difficult to solve effectively. To obtain good methods to solve this problem a new area of research is implemented. This area is based on developed heuristic techniques that provide an optimal or near optimal solution for large grids. In this paper we introduce a tasks scheduling algorithm for grid computing. The algorithm is based on simulated annealing method. The paper shows how to search for the best tasks scheduling for grid computing.

Keywords: Grid computing, Simulated Annealing, Heuristics.

1.Introduction

Computational Grids are a new trend in distributed computing systems. They allow the sharing of geographically distributed resources in an efficient way, extending the boundaries of what we perceive as distributed computing. Various sciences can benefit from the use of grids to solve CPU-intensive problems, creating potential benefits to the entire society. With further development of grid technology, it is very likely that corporations, universities and public institutions will exploit grids to enhance their computing infrastructure. In recent years there has been a large increase in grid technologies research, which has produced some reference grid implementations.

Task scheduling is an integrated part of parallel and distributed computing. Intensive research has been done in this area and many results have been widly accepted. With the emergence of the computational grid, new scheduling algorithms are in demand for addressing new concerns arising in the grid environment. In this environment the scheduling problem is to schedule a stream of applications from different users to a set of computing resources to minimize the completion time. The scheduling involves matching of applications need with resource availability. There are three main phases of scheduling on a grid [5]. Phase one is resource discovery, which generates a list of potential resources. Phase two involves gathering information about those resources and choosing the best set to match the application requirements. In the phase three the task is executed, which includes file staging and cleanup. In the second phase the choice of the best pairs of tasks and resources is NP-complete problem [5]. A related scheduling algorithm for the traditional scheduling problem is Dynamic Level Scheduling (DLS) algorithm [17]. DLS aims at selecting the best subtask-machine pair for the next scheduling. To select the best subtask-machine pair, it provides a model to calculate the dynamic level of the task-machine pair. The overall goal is to minimize the computational time of the application. In the grid environment the scheduling algorithm no longer focuses on the subtasks of an application within a computational host or a virtual organization (clusters, network of workstations, etc.). The goal is to schedule all the incoming applications to the available computational power. In [2,10] some simple heuristics for dynamic matching and scheduling of a class of independent tasks onto a heterogeneous computing system have been presented. There are two different goals for task scheduling: high performance computing and high throughput computing. The former aim is minimizing the execution time of each application and latter aim is scheduling a set of independent tasks to increase the processing capacity of the systems over a long period of time. Our approach is to develop a high throughput computing scheduling algorithm.

The organization of the paper is as follows. In section 2 the simulated annealing method is discussed and it basic structure. In section 3 grid scheduling algorithm is introduced. We make some experimental tests and conclude this studying in section 4 and 5.

2.Simulated annealing method

The Simulated Annealing (SA) is a generalization of optimization method for examines the equations of frozen states of *n*-body systems. The original Metropolis scheme [11] was that an initial state of a thermodynamic system was chosen at energy *E* and temperature *T*, holding *T* constant the initial configuration is performed and the change of energy *dE* is computed. The current state of the thermodynamic system is analogous to the current solution to the combinatorial problem, the energy equation for the thermodynamic system is analogous to the objective function and ground state is analogous to the global minimum. The key objective of this paper is to find an effective solution in a short period of time close to least cost for a given grid using simulated annealing method.

SA is a heuristic method that has been implemented to obtain good solutions of an objective function defined on a number of discrete optimization problems. This method has proved to be a flexible local search method and can be successfully applied to the majority of real-life problems [1,3,4,9,12,15]. The origin of the algorithm is in statistical mechanics. The fundamental idea is to allow moves resulting in solutions of worse quality than the current solution in order to escape from local minimum. The probability of doing such a move is declared during the search. The algorithm starts by generating an initial solution and by initializing the so-called temperature parameter *T*. The temperature decrease during the search process, thus at the beginning of the search the probability of accepting uphill moves is high and it gradually decreases. This process is analogous to the annealing process of metals and glass, which assume a low energy configuration when cooled with an appropriate cooling schedule. Regarding the search process, this means that the algorithm is the result of two combined strategies:

- Random walk;
- Iterative improvement.

The first phase permits the exploration of the search space. The advantages of the algorithm are:

- Simulated annealing is proved to converge to the optimal solution of the problem;
- An easy implementation of the algorithm.

In order to implement SA for grid scheduling problem a number of decisions and choices have to be made. Firstly, the problem specific choices, which determine the way in which the grid scheduling, is modeled in order to fit into the SA framework. In other words, it involves the definition of the solution space *Q* and neighborhood structure *I,* the form of the objective function *C(V)* and the way in which a starting solution *V* is obtained. Secondly, the generic choices, which govern the working of the algorithm itself, are mainly concerned with the components of the cooling parameters: control parameter *T* and its initial starting value, the cooling rate *F* and the temperature update function, the number of iterations between decreases *L* and the condition under which the system will be terminated [14,15]. The performance of the achieved result is highly dependent on the right choice of both specific and generic choices.

The SA procedure used in this work is designed and developed essentially from practical experience and requirement of the grid. A simple constructive procedure is proposed to obtain an initial (starting) feasible scheduling *V* for the grid. The aim of this simple procedure is to obtain quickly an initial schedule. The structure of the SA algorithm is shown below:

1.The problem Specific Decisions

Step 1. Formulation the problem parameters;

Step 2. Determination of the initial schedule, generate a feasible solution *V*;

2. The Problem Generic Decisions

Step 3. Initialization the cooling parameters:

- Set the initial value of the temperature parameter, $T=8$;
- Set the temperature length $L=3$;
- Set the cooling rate $F=0.9$;
- Set the number of iterations $K=0$;

3. The Generation Mechanism, Selecting and Acceptance Strategy of Generated Neighbors

Step 4.

- Select a neighbor *V'* of *V* where $V' \in I(V)$
- Let $C(V')$=the cost of the schedule *V'*
- Compute the move value $\Delta=C(V')-C(V)$

Step 5.

- IF $\Delta \leq 0$ accept *V'* as a new solution and set $V=V'$
- ELSE
- IF $e^{-\Delta/T}>\theta$ set $V=V'$
- Where θ is a uniform random number $0<\theta<1$
- OTHERWISE retain the current solution *V*

4. Updating the Temperature

Step 6. Updating the annealing scheduling parameters using the cooling schedule $T_{k+1}=F*T_k$ $k=1,2,...$

5. Termination of the Solution

Step 7. IF the stopping criteria is met THEN

Step 8.

- Show the output
- Declare the best solution
- OTHERWOISE GO to step 4.

3.Grid scheduling model

Our scheduling algorithm is designed for distributed systems shared asynchronously by both remote and local users.

3.1.Grid model

The grid considered in this study is composed of a number of hosts, each composed of several computational resources, which may be homogeneous or heterogeneous. The grid scheduler does not own the local hosts, therefore does not have control over them. The grid scheduler must make best effort decisions and then submit the task to the hosts selected, generally as a user. Furthermore, the grid scheduler does not have control over the set of tasks submitted to the grid, or local tasks submitted to the computing host directly. This lack of ownership and control is the source of many of the problems yet to be solved in this area.

3.2. Grid scheduling algorithm

While there are scheduling request from applications, the scheduler allocates the application to the host by selecting the best match from the pool of applications and pool of the available hosts. The selecting strategy can be based on the prediction of the computing power of the host [8]. We will review some terms and definitions [10,13].

The expected execution time ET_{ij} of task t_i on machine m_j has no load when t_i is assigned. The execution time CT_{ij} of the task t_i on machine m_j is defined as the wall-clock time at which m_j completes t_i (after having finished any previously assigned tasks). Let m be the total number of the machines. Let S be the set containing the tasks. Let the beginning time of t_i be b_i. From the above definitions $CT_{ij} = b_i + ET_{ij}$. The makespan for the complete schedule is then defined as $max_{\ ti \in \kappa}(CT_{ij})$. Makespan is a measure of the throughput of the heterogeneous computing system. The objective of the grid scheduling algorithm is to minimize the makespan. It is well known that the problem of deciding on an optimal assignment of tasks to resources is NP-complete. We develop heuristic algorithm based on simulated annealing to solve this problem.

Existing mapping heuristics can be divided into two categories: on-line mode and batch mode. In the on-line mode, a task is mapped onto a machine as soon as it arrives at the mapper. In the batch mode, tasks are not mapped onto the machines as they arrive, instead they are collected in a set that is examined for mapping at pre-scheduled times called mapping events. This independent set of tasks that is considered for mapping at mapping events is called meta-tasks. In the on-line mode heuristics, each task is considered only for matching and scheduling. The minimum completion time heuristic assigns each task to the machine so that the task will have the earliest computational time [6]. The minimum execution time heuristic assigns each task to the machine that performs tasks computation in the least amount of execution time. In batch mode, the scheduler considers a meta-task for matching and scheduling at each mapping event. This enables the mapping heuristics to possibly make better decision, because the heuristics have the resource requirement information for the meta-task and known the expected execution time of a larger number of tasks. Our heuristic algorithm is for batch mode.

The most important part in the application of SA is generation of the initial solution and creating a set of neighbors. We will describe in details the initial solution generation and creating of the neighbor set for grid scheduling algorithm.

Let the number of the tasks in the set of tasks is great than the number of machines in the grid. The result will be triples *(task, machine, starting time)*. The function *free(j)* – shows when the machine m_j will be free. To generate initial solution we use greedy heuristic, the first task in the set will be executed on the first free machine, the machine with minimal *free(j)* function. The value of the *free(j)* function is updated and the same method is used for the second task in the set and so on. If the task t_i is executed on the machine m_j then the starting time of t_i becomes $b_i = free(j)+1$ and the new vlue of the function *free(j)* becomes $free(j)= b_i + ET_{ij}= CT_{ij}$. So the initial solution is a feasible solution.

After the generation of a feasible solution the set of neighbors will be created. As is written above the solution is triple (t_i , m_j, b_i).We can think about a solution like a matrix with three columns, the fist is the tasks, the second is the corresponding machines and the third is corresponding starting times. The order in the columns is by starting time. Thus the tasks with early starting time are before tasks with later starting time. To create new solution we will swap two of the tasks. It changes the starting times and the value of the *free(j)* functions and reorder succeeding tasks.

Two kind of set of tasks are needed: set of scheduled tasks and set of arrived and unscheduled tasks. When we start the execution of scheduled tasks, we collect new arrived tasks in the set of unscheduled tasks. If some of the machines starts to execute the last scheduled on it task, schedule algorithm is started over the tasks from the set of unscheduled tasks. When we start the next schedule the starting time for a machine is the end time for the same machine from the previous schedule. Thus it is guaranteed that the machines will be fully loaded. If some machine becomes out of service, we move all unexecuted tasks, scheduled on this

machine in the set of unscheduled tasks and we schedule the tasks on other machine. The tasks, submitted to the computing nodes bypassing the scheduler, are ran between the tasks of two scheduled sets and when the free time is calculated, there running time are taken in to account. Thus we solve the problem in dynamic way.

4.Experimental results

In this section some experimental results are reported. We compare the often used online algorithm with explained simulated annealing algorithm and ant colony optimization algorithm (ACO), for grid scheduling problem [7], which is a heuristic method too. In online mode the tasks are in a queue towards their arriving time in a scheduler. When some of the machine is free, the first task in the queue is run on this machine. Thus the order of the queue is important and it is arriving order of the tasks on the scheduler. In the tests, the time to send a task on a machine is ignored. To be more realistic it is necessary to add the send time from the scheduler to the machine, for online mode.

We compare our simulated annealing algorithm with ant algorithm too. The both are applied on batch mode and are offline methods. These are intelligent methods for scheduling. In the both the tasks first are collected in a set and after that they are scheduled. Thus the arriving time of the tasks is not important. The tasks scheduled on a same machine form a local queue related with this machine. The tasks queue is sent on the machine during the running tasks from previous queue. Thus the send time do not influence the makespan time. If there is a cluster like a part of the grid, we look at it like one machine, thus the problem becomes sequential tasks scheduling problem.

In our experimental testes we use 5 heterogeneous machines and 20 tasks. The following parameters are used: initial temperature 8, final temperature 3, temperature length 3 and cooling parameter 0.9 for all tests.

Table 1. Makespan for the execution by online-mode, simulated annealing and ant colony optimization algorithms

Online-mode	Simulated annealing	ACO
80	54	67
174	97	128
95	78	80

The results are in minutes. We observe outperform of the heuristic algorithms comparing with online mode. In online mode the arriving order is crucial while in heuristic algorithms the arriving order is not important and the most important is the execution time of the tasks. Comparing the two heuristic algorithms, simulated annealing achieves better results.

5.Conclusion

To confront new challenges in tasks scheduling in a grid environment, we present in this study heuristic scheduling algorithm. The proposed scheduling algorithm is designed to achieve high throughput computing in a grid environment. This is a NP-problem and to be solved needs an exponential time. Therefore the heuristic algorithm, which finds a good solution in a reasonable time, is developed. In this paper heuristic algorithm based on simulated annealing method is discussed and it basic strategies for a grid scheduling are formulated. This algorithm guarantees good load balancing of the machines and it is applied in dynamic way. In SA technique it is very important how the set of neighbors is created. As a future work we will try with various manner to create the set of neighbors. Another research direction is to create different heuristic based algorithms for problems arising in grid computing.

Acknowledgements. Stefka Fidanova was supported by the European Community project BIS 21++.

References

[1] E. Arts and P. Van Losrhoven, "Statistical Cooling: A general Approach to Combinatorial Optimization Problems", *Philips Journal of Research* 40, 1985,193-226.

[2] T. D. Braun, H. J. Siegel, N. Beck, L. Bolony, M. Maheswaram, A. I. Reuther, P. J. Robertson, M. D. Theys and B. Jao, "A Taxonomy for Describing Matching and Scheduling Heuristics for Mixed-Machine Heterogeneous Computing Systems", *IEEE Workshop on Advances in Parallel and Distributed Systems,* 1998, 330-335.

[3] V. Cerny, "A Thermodynamical Approach to the Traveling Salesman problem: An Efficient Simulated Annealing Algorithm", *Journal of Optimization Theory and Applications* 45, 41-51.

[4] K. Dowsland, "Variants of Simulated Annealing for practical problem Solving", V. Rayward-Smith editor, *Applications of Modern Heuristic Methods,* Henley-on-Thames: Alfred Walter Ltd., 1995.

[5] D. Fernandez-Baca, "Allocating Modules to Processors in a Distributed System, *IEEE Transactions on Software Engineering* 15(11), 1989, 1427-1436.

[6] F. R. Freund, M. Gherrity, S. Ambrosius, M. Camp-Bell, M. Halderman, D Hensgen, E Keith, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust and H. J. Siegel, "Scheduling Resources in Multi-User Heterogeneous Computing Environments with

SmartNet", *IEEE Heterogeneous Computing Workshop,* 1998, 184-199.

[7] S. Fidanova and M. Durchova, "Anr Algorithm for Grid Scheduling Problem", *Large Scale Computing, Lecture Notes in Computer Science* No 3743, Springer, germany,2006,405-412.

[8] L. Gong, X. H. Sun and E. Waston, "Performance Modeling and Prediction of Non-Dedicated Network Computing", *IEEE Transaction on Computer* Vol 51(9), 2002.

[9] S. Kirkpatrick, C. D. Gelatt and P. M. Vecchi, "Optimization by Simulated Annealing", *Science* 220, 1983, 671-680.

[10] M. Maheswaran, S Ali, H. J. Siegel, D. hensgen and R. Freund, "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems", *8th IEEE Heterogeneous Computing Workshop (HCW'99),* San Juan, Puerto Rico, 1999, 30-44.

[11] N. metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, "Equition of State Calculations by Fast Computing Machines, *Journal of Chemistry Physics* 21(6), 1953, 1087-1092.

[12] I. H. Osman and C. N. Potts, "Simulated Annealing for Permutation Flow-Shop Scheduling", *Omega* 17, 1989, 551-557.

[13] M. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Prentice Hall, Englewood Clefts, NJ, 1995.

[14] C. R. Reeves (editor), *Modern Heuristic Techniques for Combinatorial Problems,* Oxford, England, Blackwell Scientific publications, 1993.

[15] V. V. Rene, *Applied Simulated Annealing*, Berlin, Springer, 1993.

[16] M. J. Schopf, "General Architecture for Scheduling on the Grid", *Special issue of JPDC on Grid Computing,* 2002.

[17] G. C. Sih and E. A. Lee, "A Compile-Time Scheduling Heuristic for Inter Connection-Constrained Heterogeneous Processor Architectures", *IEEE transactions Parallel and Distributed Systems* Vol 4, 1993, 175-187.