Optimizing Grid Scheduling with Particle Swarm Optimization

T. R. Srinivasan, R. Shanmugalakshmi,

Professor, Department of IT,
Vidyaa Vikas College of Engineering and Technology, Tiruchengode, India.
Associate Professor, Department of CSE & IT,
Government College of Technology, Coimbatore, India
E-mail: trssrinivasan838@gmail.com

Abstract: Grid computing is a computing framework to run different grid enabled applications. This paper proposes a neural network to capture user requirements automatically and to use it for resource selection. It also introduces a Particle Swarm Optimization (PSO) based approach to schedule computational grid jobs. Representations of position and particle velocity in conventional PSO are extended to real vectors. The proposed approach plans to dynamically generate an optimal schedule to finish jobs within a specific minimum time duration. It also uses resources efficiently.
[T. R. Srinivasan, R. Shanmugalakshmi. **Optimizing Grid Scheduling with Particle Swarm Optimization.** *Life Sci J* 2013; 10(4s):559-563] (ISSN: 1097-8135). http://www.lifesciencesite.com. 85

Keywords: Grid, Resource Selection, Grid Scheduling, Neural Network, Particle Swarm Optimization (PSO)

I Introduction

Grid Computing performs distributed computing, high-throughput computing, on-demand computing, data-intensive computing and collaborative computing/multimedia computing. Grid technology combines physical dynamic resources and differing applications. It is currently a technology with high potential for good resource utilization [1]. Scheduling is a grid computing implementation issue. Computational requirements resources increase daily as computational tasks are intensive. As for scientific experiments, simulations, problem solving, the capability/capacity of a stand alone machine is not enough [2] organizations opt for dedicated resources like supercomputers and mainframes at high prices. Development of powerful, low cost miroprocessors and high speed computer networks has changed computing paradigms in the last two decades.

Presently mainframes have been replaced by low-cost and high powered collaborative infrastructure which can effectively use an organization's computational resources. However, such potential cannot be used till users can access such resources. Transparency means users can access resources sans physical locations which in urn motivated evolution of Resource Management Systems (RMS). Grid computing outgrew its original aim of linking supercomputers.

A scheduler has various tasks – including resource discovery and resource selection - necessary for efficient application execution.
Resource discovery: This provides available resources list and authorizes specific tasks/users. A Scheduler surfs databases to check various resources and availability drawing up a availability list. Monitoring and Discovery Services (MDS) in Globus is a scheduler resource discovery module. Resource discovery keeps tabs on resources entering/leaving Grids.

Resource selection: A scheduler has many resources from a route discovery process provided list that meet user requirements like computing speed, deadline, and cost. Those available including the least-loaded and fastest resources for a specific job, are identified Thus resource selection increases job performances and/or lowers communication time.

Scheduling systems allocate resources for required tasks, improve resource load balance, security, reliability and fault-tolerance. Grid resource scheduling performs user's tasks and satisfy user requirements through coordination and resource configuration available which includes network resources, computing and storage ensuring that each task has own resource. Scheduling ensures task completion/performance by resources simultaneously. n user submitted tasks are queued awaiting scheduling. The scheduler reads and removes the task from the queue when idle, allocating correct resources so that task gets proper execution needs. Once completed, the system updates resource nod as to whether it was successful with the resource being re-added to the resource queue.

Grid computing scheduling aims to find an optimal resource to improve system performance and also use resources better. The 4 resource scheduling strategies are: centralized approach, distributed approach, hierarchical approach and multi-agent scheduling (Hamscher, et al., 2000). Centralized approach ensures one scheduling center which is responsible for resources. A distributed approach will

have many scheduling centers to schedule its own resources. A Hierarchical approach is a combination of both centralized and distributed approaches. The Multi-Agent approach includes computer resources discovery and management through a mini agent program dominating all network scheduling functions. The above mentioned approaches are efficient in scheduling grid resources - with their application to a different environment to some level – to a certain extent.

Highly sophisticated resource management systems are needed to correct computing resources to solve scientific/engineering problem in a Grid. This is possible only if strategies and technologies can master modern large-scale networks and computing complexities. Grid computing when in a line with a service-oriented approach has fostered a new vision where economic aspects represent central issues rather than using computing as an utility. Hence, design, data execution and compute-intensive applications are simplified by adopting model-driven workflow based approaches.

This paper proposes to automatically capture user requirements, using it for resource selection through neural network use. It also introduces a Particle Swarm Optimization (PSO) based approach to schedule jobs on computational grids. The reminder of the paper is as follows: Section 2 relates works in literature with regard to optimization in grid scheduling. Section 3 deals with methodology, section 4 provide results and a follow up discussion with section 5 concluding the paper.

II Related Works

Jin Xu, et al., [4] proposed many versions of Chemical Reaction Optimization (CRO) algorithm to handle grid scheduling issues. A population-based metaheuristic, CRO, was inspired by molecular interactions in chemical reactions. This study was due to the fact that though Grid computing solved high performance/high-throughput computing problems through resource sharing - ranging from personal computers to supercomputers - task scheduling was still a big problem. ie; allocating tasks to different resources. In addition to Makespan and Flowtime, the study accounted for resource reliability. Task scheduling was considered a three objective optimization problem. A metaheuristic approach was chosen to locate an optimal solution as this was an NP-hard problem. This paper compares CRO methods with four other metaheuristics on a range of instances. Simulation results revealed that CRO methods outperformed 5 existing methods with improvement being quite high in large-scale applications. The study also showed that for independent task grid scheduling issues,vector-based representation was better than permutation-based procedures.

To ensure reliability in normal grid services, SuchangGuo, et al., [5] introduced Local Node Fault Recovery (LNFR) mechanism in grid systems. The study looked in-depth into grid service reliability modeling and analysis with the earlier mentioned fault recovery type. Though there was a little research on tools development and grid system techniques, yet important issues like grid service reliability and grid task scheduling were not taken seriously. Grid services reliability was rather low for some grid services with large subtasks needing large computation. Constraints like subtasks lifetimes and recoveries performed in grid nodes, were introduced to ensure a practical LNFR. Grid service reliability models were introduced under such constraints. A multi-objective task scheduling optimization model based on proposed grid service reliability model, was presented followed by development of ant colony optimization (ACO) algorithm to find a successful solution. Experiments revealed that when grid service reliability increased so did the cost which was unexpected. The reason was that resources were priced arbitrarily, whereas in reality price was linked to grid resources performance like CPU processing capability and reliability. Hence, grid resources price played a big share in grid task scheduling.

Abraham, et al., [6] addressed dynamic job scheduling to geographically distributed computing resources. Conventionally, scheduling variety and richness of scheduling problems proved that one scheduling method was never enough.. Nature derived heuristics showed a surprising effectiveness and generality when handling combinatorial optimization problems. This paper initially introduces computational grids and then a brief description of three nature heuristics including Genetic Algorithm (GA), Simulated Annealing (SA) and Tabu Search (TS). Hybrid usage of the three algorithms was demonstrated proving that were suitable for a computational grid environment required for job scheduling. The GA-SA algorithm had better convergence than pure GA search and GA-TS algorithm improved GA's efficiency. Because of solution space complexity and other constraints, enumeration of (even implicitly) solution space points was problematic. While GAs deal with solution populations, TS and SA are search procedures dealing with a solution at a time.

Gao, et al., [7] proposed two models to predict service grid job completion time. The single service model predicted this providing a single type of service, whereas multiple services model predicted job completion time in a Grid with multiple services. Two algorithms were developed to schedule jobs at system and application levels Genetic algorithms minimized average job completion time through optimal job

allocation on every node in application-level scheduling. Experiments revealed that a scheduling system using adaptive scheduling algorithms allocated service jobs correctly.

III Methodology

Neural Network for Resource Selection

A human brain has around ten billion densely interconnected neurons with structural and functional complexity resulting in a complex architecture and an intelligence level not achieved by any artificial system. Many mathematical models were developed to represent interconnected neurons. Artificial neural networks (ANN) attempt to reproduce human brain potentialities on a limited scale, including its learning ability. McCulloch and Pitts first neuron mathematical model had a binary output and inputs, each of different excitatory or inhibitory gains, known as synaptic weights (or weights). Input signals values and related weights determine neuron output. ANN architecture includes a set of weight (wi) related n inputs (xi) and an activation function (fi). Neurons form layers with all being linked with distinct outputs. ANN topology solves classification problems with non-linearly separable patterns and is a universal function generator [8]. ANN consists of learning and execution phases. The basis learning is as follows:

1. Initialise network, with randomly numbered weights set between -1 and +1.
2. Present first training pattern and obtain output.
3. Compare network output with target output.
4. Propagate error backwards.

(a) Correct output weights layer with the formula below.

$$w_{ho} = w_{ho} + (\eta \delta_o o_h)$$

where w_{ho} is the weight connecting hidden unit h with output unit o, η is the learning rate, o_h is the output at hidden unit h. δ_o is given as follows:

$$\delta_o = o_o (1 - o_o)(t_o - o_o)$$

o_o is output at node o of output layer and t_o is target output for that node.

(b) Correct input weights using following formula.

$$w_{ih} = w_{ih} + (\eta \delta_h o_i)$$

where w_{ih} is weight connecting node i of input layer with node h of hidden layer, o_i is input at node i of input layer, δ_h is calculated as follows:

$$\delta_h = o_h (1 - o_h) \sum_o (\delta_o w_{ho})$$

5. Calculate error by taking average difference between target and output vectors. The following function is an example

$$E = \frac{\sqrt{\sum_{n-1}^{p} (t_o - o_o)^2}}{p}$$

Where p is number of units in output layer.

6. Repeat steps from 2 for each training set pattern to complete one epoch. Step 2 is repeated for a specific number of epochs, or till error ceases to change.

A training set containing user resource selection examples are used in the learning phase. Figure 1 shows Grid resource selector process. Inputs are job description and computing resource status. Inputs are vectors, job description is given by $J\{j_1, ...j_h\}$ and computing status is given by $C_i = \{s_{i1}, ..., s_{ik}\}$. Output is the selected computing resource. Table 1 shows input parameters used.

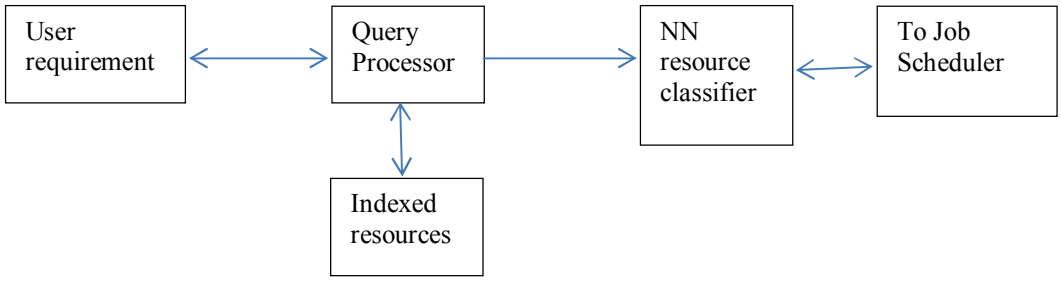

Figure 1: Grid resource selector

Table 1: Input parameter for Neural Network

Parameter	Unit of Measurement
Spec cpu2006	GHz
Spec viewperf	[]
Actual BW	MHz
Available BW	MHz

Particle Swarm Optimization for Grid Scheduling

Particle Swarm Optimization(PSO) simulates bird flocking behaviour. PSO learned and used the scenario to solve optimization problems [9]. Every solution is a "bird" in the search space, called as "particle" in PSO. All particles have fitness values, evaluated by fitness function to be optimized, and have velocities which direct particle flying. Particles fly through problem space following optimum particles.

PSO starts with random particles (solutions) searching for optima through updating generations. Each iteration, each particle is updated by two "best" values. The first is best solution (fitness) achieved till then. (Fitness value is stored.) This value is called pbest. Another "best" value tracked by the particle swarm optimizer is best value, obtained till then by any population particle. This best value is a global best called gbest. When a particle takes portion of the population as topological neighbor, best value is a local best called lbest.

For problems formulation *Jn* independent user jobs n={1,2,….N} is considered with the aim of reducing completion time and resource use. Each resource's speed is expressed in number of cycles per unit time, and job length in number of cycles. Each job *Jn* has processing requirement *Pj* cycles and resource R*m* has speed of *Si* cycles/second. Any job *Jn is* to be processed in resource R*m,* till completed. To formulate our objective, Cj is defined as completion time last job *j* finishes processing. Define Cmax = max {*Cj*, j=1,…N}, makespan and ∑*Cj*, as flowtime. An optimal schedule is one which optimizes flowtime and makespan [10]. A conceptually obvious rule to minimize ∑*Cj* is through scheduling Shortest Job on the Fastest Resource (SJFR). A simple rule to minimize Cmax scheduling Longest Job on Fastest Resource (LJFR). Minimizing ∑*Cj* the average job finishes quick, at the largest job's expense taking longer, whereas minimizing Cmax, asks that no job take long, at most jobs expense taking long. To summarize, minimization of Cmax will ensure maximization of ∑*Cj*.

IV Results

Scheduling methods effectiveness is evaluated using evaluation metrics like makespan and flowtime. Makespan is the time the grid takes to complete the latest task; and flowtime is total execution times for all tasks presented to the grid [11-12].

Experiments were conducted with 10 resources and 5 jobs. Initial experiment was with random scheduling without resource selection. Experiments were with PSO grid scheduling but without proposed neural network resource selector. Final experiments were with proposed neural network selector and PSO grid scheduling. Table 2 reveals makespan results for various experiments. Figure 2 shows Makespan time vs. number of iterations.

Table 2: Makespan results

Experiment Setup	Makespan
Without NN resource selection and random scheduling	71.82
Without NN resource selection and PSO based grid scheduling	66.27
Proposed system - With NN resource selection and PSO based grid scheduling	61.54

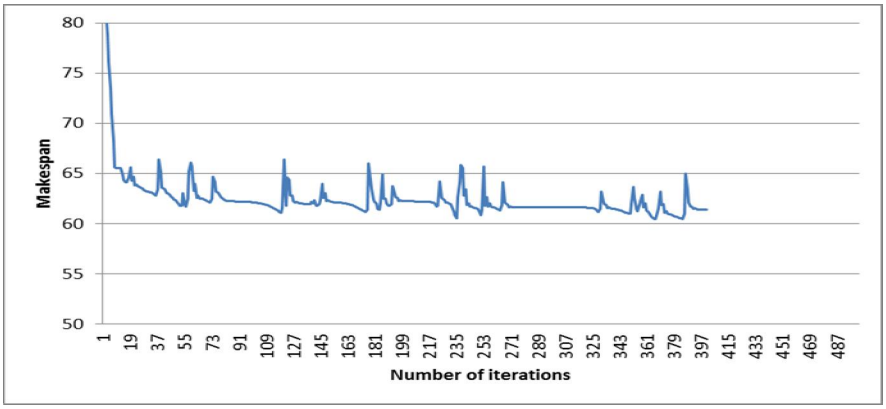

Figure 2: Makespan vs. Number of Iterations.

V Conclusion

This paper proposes to automatically capture user requirements for neural network resource selection. It introduces a Particle Swarm Optimization (PSO) based approach to schedule jobs on computational grids. Position and particle velocity representations in a conventional PSO are extended to real vectors. The aim is generating an optimal schedule to complete tasks in minimum time using resources efficiently. PSO's advantage is convergence speed and ability to get faster/feasible schedules. Experiments provided satisfactory results.

REFERENCES

[1] Subramaniam, N. "Grid computing: An overview", Idea Group, 2003.

[2] Foster, I., Kesselman, C., Nick, J. and Tuecke, S. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Technical Report, Open Grid Service Infrastructure WG, Global Grid Forum, 2002.

[3] Hamscher, U. Schwiegelshohn, A. Streit, R. Yahyapour, *Evaluation of Job-Scheduling Strategies for Grid Computing*, in Proc. of GRID 2000 GRID 2000, First IEEE/ACM International Workshop, pp. 191-202, Bangalore, India, December 2000.

[4] Jin Xu, Albert Y.S. Lam and Victor O.K. Li; Chemical Reaction Optimization for Task Scheduling in Grid Computing; IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 10, October 2011.

[5] Suchang Guo, Hong-Zhong Huang, Zhonglai Wang, and Min Xie; Grid Service Reliability Modeling and Optimal Task Scheduling Considering Fault Recovery; IEEE Transactions on Reliability ; Vol. 6, No. 1, March 2011.

[6] A. Abraham, H. Liu, W. Zhang and T.G. Chang, Job Scheduling on Computational Grids Using Fuzzy Particle Swarm Algorithm, 10th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, B. Gabrys et al. (Eds.): Part II, Lecture Notes on Artificial Intelligence 4252, 500507, Springer, 2006.

[7] Y. Gao, H. Rong and J.Z. Huang, Adaptive grid job scheduling with genetic algorithms, Future Generation Computer Systems, 21(1), 151-161, 2005.

[8] K. S. Narendra and K . Parthasarathy, "Identification and control of dynamical systems using neural networks," IEEE Trans. Neurul Networks, vol. 1, pp. 4-27, Mar. 1990.

[9] Kennedy,J., Eberhart,R.: Swarm Intelligence. Morgan Kaufmann (2001).

[10] Pang,W., Wang,K., Zhou,C., Dong,L.: Fuzzy Discrete Particle Swarm Optimization for Solving Traveling Salesman Problem. In: Proceedings of the Fourth International Conference on Computer and Information Technology, IEEE CS Press (2004) 796–800.

[11] Izakian, H., et al., A novel particle swarm optimization approach for grid job scheduling. Information Systems, Technology and Management, 2009: p. 100-109.

[12] Sayadi, M.K., R. Ramezanian, and N. Ghaffari-Nasab, A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. International Journal of Industrial Engineering Computations, 2010. 1: p. 1-10.

2/20/2013