International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 06 | June 2019 www.irjet.net p-ISSN: 2395-0072

A Particle Swarm Optimization Algorithm for Total Cost Minimization in the Cloud Manufacturing

S.A.A. Jude[1], S. Selva Prabhu[2], P.Thangababu[3], S. Muruga Perumal[4], M. Sathish[5]

[1]*Assistant Professor, Mechanical Engineering, PSN College of Engineering and Technology, Tirunelveli, India*
[2]*Research Scholar, PSN College of Engineering and Technology, Tirunelveli, India*
[3,4,5]*UG Student, BE Aeronautical Engineering, PSN College of Engineering and Technology, Tirunelveli*

Abstract - *Cloud manufacturing is a hybrid model that provides both hardware and software resources through computer networks. Data services (hardware) together with their functionalities (software) are hosted on web servers rather than on single computers connected by networks. Through a computer, a browser and an internet connection, each user accesses a cloud platform and asks for specific services. This model creates a brand new opportunity for enterprises. It is largely impossible to carry out some manufacturing tasks without the support of suppliers, contractors and business partners. There is a need to connect manufacturers together to share risks, benefits, competiveness and costly resources. More than just deploying manufacturing-related software applications in the computing Cloud, Cloud Manufacturing is an integrated solution that provides a pool of machine capabilities provided by Cloud participants.*

Key Words: Cloud manufacturing, Particle swarm optimization algorithm, Cloud computing

1. INTRODUCTION

Cloud Manufacturing is a model for enabling ubiquitous, convenient and on-demand network access to a shared pool of configurable manufacturing resources (e.g., manufacturing software tools, manufacturing equipment, and manufacturing capabilities) that can be rapidly provisioned and released with minimal management effort or service provider interactions. Like Cloud Computing concept, manufacturing infrastructure, platform and software application in Cloud Manufacturing can be offered as a service to a Cloud User. By extending the concept to a broader scope, all the production objects and features can be treated as a service, hence everything as a service. The rest of this section discusses the Cloud Manufacturing structure and related technologies.

Cloud concept presents a promising future for computing business and the same can be said for manufacturing business. Cloud Manufacturing is described as a computing and service-oriented manufacturing model developed from existing advanced manufacturing models (e.g., Application service provider, Agile Manufacturing, Networked Manufacturing, and Manufacturing Grid), enterprise information technologies under the support of cloud computing, Internet of things, virtualization and service-oriented technologies, and advanced computing technologies. In this paper, we focus on minimizing the total execution cost of applications on these resources provided by Cloud service providers, such as Amazon and GoGrid3. We achieve this by using a meta- heuristics method called Particle Swarm Optimization (PSO).

2. PROBLEM FORMULATION

The mapping of tasks of an application workflow to distributed resources can have several objectives. We focus on minimizing the total cost of computation of an application workflow. Figure 5 depicts a workflow structure with five tasks, which are represented as nodes. The dependencies between tasks are represented as arrows.

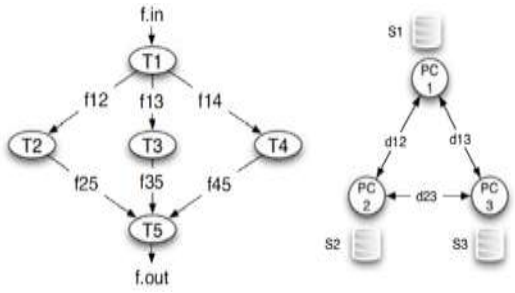

Figure1 An example workflow, compute nodes (PC) & storage (S).

This workflow is similar in structure to our version of the Evolutionary Multi-objective Optimization (EMO) application. The root task may have an input file (e.g. f.in) and the last task produces the output file (e.g. f.out). Each task generates output data after it has completed (f12, f13, ..., f45). These data are used by the task's children, if any. The numeric values for these data is the edge-weight (ek1,k2) between two tasks k1 Є T and k2 Є T . The figure also depicts three compute resources (PC1, PC2, PC3) interconnected with varying bandwidth and having its own storage unit (S1, S2, S3). The goal is to assign the workflow tasks to the compute resources such that the total cost of computation is minimized.

Task 1	Task 2	Task 3	Task 4	Task 5
PC2	PC2	PC3	PC3	PC1

A sample particle for the workflow

	Task 1	Task 2	Task 3	Task 4	Task 5
PC	1	3	3	2	1
	2	3	1	1	3
	1	2	3	3	2
	3	2	3	1	2
	3	2	2	3	1

Table 1: The Taskflow

3. PARTICLE SWARM OPTIMIZATION ALGORITHM

PSO was first introduced by Kennedy and Eberhart as an optimization method for non-linear functions with continuous variables. The initial intention of PSO was to simulate the social behavior of flocking birds searching for food by means of exchanging knowledge among flock members. By applying simple formulae, Kennedy and Eberhart developed an optimization algorithm that mimics this knowledge sharing. Each individual in the flock was represented by a point in a two- dimensional space, and future movement of each point in the search space is determined using a combination of previous experience of the individual, and of other individuals in the group.

The PSO provides a population-based search procedure in which the individuals, called particles, change their positions with time. Each particle adjusts its position according to its own best experience and the best experience of neighboring particles.

The PSO heuristic is applied by first generating a number of random solutions (or positions of particles) in the solution space. PSO searches for optimal solution by updating generations. Each particle is updated by means of two 'best' values, namely pbest (p_k^i) and gbest (p_k^g) in successive iteration. The pbest (p_k^i) is the best solution, a particle has achieved so far. The gbest (p_k^g) is the best solution obtained so far by any particle in the population. The quality of each particle position is then evaluated based on the objective function. To proceed from iteration k to the next iteration k+1, the velocity of a particle i is calculated using the following equation (1).

$$v_{k+1}^i = \omega * v_k^i + C_a * \text{rand}() * \left(p_k^i - s_k^i\right) + C_b * \text{rand}() * \left(p_k^g - s_k^i\right) \quad (1)$$

The new position of the particle is obtained by equation (2),

$$s_{k+1}^i = s_k^i + v_{k+1}^i \quad (2)$$

Where s_k^i is particle i position in the current iteration k. ω can be expressed by the inertia weights approach, Ca and Cb are the acceleration constants which influence the convergence speed of each particle, and rand( ) is a random number between 0 and 1. In equation (1), the first part represents the inertia of the previous velocity, the second part is the "cognition" part which represents the private thinking by itself, and the third part is the "social" part which represents the cooperation among the particles. If the summation in equation (1) causes the velocity $v_k i$, on that dimension, to exceed Vmax, then $v_k i$ will be limited to Vmax. Vmax determines the resolution with which regions between the present position and the target position are searched. If Vmax is too large, the particles might fly over the past good solutions. If Vmax is too small, the particles may not explore sufficiently beyond local solutions. Usually the range of particle is taken as the Vmax. In this project, Vmax = 6. The constants Ca and Cb represent the weighting of the stochastic acceleration terms that pull each particle toward $p\_k i$ and $p\_k g$ positions. Low values allow particles to roam far from the target regions before being tugged back. On the other hand, high values result in abrupt movement toward or passed the target regions. Hence the acceleration constants Ca and Cb are often set to be 2.0 according to the past experiences. Suitable selection of inertia weight ω provides a balance between global and local explorations, thus requiring less iteration on average to find a sufficiently optimal solution. As originally developed, ω often decreases linearly from about 0.9 to 0.4 during a run. In general, the inertia weight ω is set according to the following equation:

$$\omega = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{iter_{max}} \times iter \quad (3)$$

ω Inertia weight;

$v\_k i$ Velocity of particle i in the current iteration k;

$v\_(k+1)i$ Velocity of particle i in the next iteration k+1;

$p\_k i$ The best solution that particle i reached throughout iterations 1,2, …….k,;

$p\_k g$ The best solution that the group has reached throughout iterations 1,2, …..k,;

$s\_k i$ Particle i position in the current iteration k;

$s\_(k+1)i$ Particle i position in the next iteration k+1;

rand() Uniformly distributed random number generated between 1 and 3;

Ca& Cb Learning factors.

Where itermax represents the maximum number of iteration, and iter is the current number of iterations. itermax depends on the problem to be optimized. Here, it is

taken as 500. Moreover, ωmax and ωmin are the maximum and minimum weight values respectively.

4. Proposed PSO Algorithm

Set particle dimension as equal to the size of ready tasks

Initialize particles position randomly For each particle, calculate its fitness valueIf the fitness value is better than the previous best pbest, set the current fitness value as the new pbest.

After Steps 3 and 4 for all particles, select the best particle as gbest.

For all particles, calculate velocity using Equation 1 and update their positions using Equation 2.

If the stopping criteria or maximum iteration is not satisfied, repeat from Step3.

5. Experimental Evaluation

In this section, we present the metric of comparison, the experiment setup and the results.

Five possible combinations of tasks are considered in the initialization module. Matrix [1] shows the initial population consists of five particles. Each and every particle in the initial population has a single row.

$$\begin{pmatrix} 1 & 3 & 3 & 2 & 1 \\ 2 & 3 & 1 & 1 & 3 \\ 1 & 2 & 3 & 3 & 2 \\ 3 & 2 & 3 & 1 & 2 \\ 3 & 2 & 2 & 3 & 1 \end{pmatrix} \quad [1]$$

A. Evaluation module

Matrix [1] is called $s\__k i$ and the manufacturing system efficiency is calculated for all particles. The maximum total indexing time in seconds is corresponding to the second particle. Hence, the second particle is selected and is called gbest. $p_k g$ for the first iteration. The $p\__k g$ is then converted into the same size of $s_k i$ (i.e., 5 x 3), by repeating the same row.

B.Scheduling heuristic Algorithm.

Calculate average computation cost of all tasks in all compute resources. Calculate average cost of (communication/ size of data) between resources Set task node weight $w_k j$ as average computation cost Set edge weight $e_k 1_{,k} 2$ as size of file transferred between tasks Compute PSO({ti}) /*a set of all tasks i ∈ k*/repeat for all "ready" tasks {ti} ∈ T do Assign tasks {ti} to resources {pj} according to the solution provided by PSO end for Dispatch all the mapped tasks Wait for polling time Update the ready task list Update the average cost of communication between resources according to the current network load Compute PSO({ti}) until there are unscheduled tasks

C. Scheduling Heuristic

We calculate the average computation cost all tasks on all the computer resources. This cost can be calculated for any application by executing each task of an application on a series of known resources. It is represented as TP matrix in Table 2. As the computation cost is inversely proportional to the computation time, the cost is higher for those resources that complete the task quicker. Similarly, we store the average value of communication cost between resources per unit data, represented by PP matrix in Table 2, described later in the paper. The cost of communication is inversely proportional to the time taken. We also assume we know the size of input and output data of each task.In addition, we consider this cost is for the transfer per second (unlike Amazon CloudFront which does not specify time for transferring). The initial step is to compute the mapping of all tasks in the workflow, irrespective of their dependencies (Compute PSO(ti)).

This mapping optimizes the overall cost of computing the workflow application. To validate the dependencies between the tasks, the algorithm assigns the "ready" tasks to resources according to the mapping given by PSO. By "ready" tasks, we mean those tasks whose parents have completed execution and have provided the files necessary for the tasks' execution. After dispatching the tasks to resources for execution, the scheduler waits for polling time. This time is for acquiring the status of tasks, which is middleware dependent

TP (5x3)=

	PC1	PC2	PC3
T1	1.23	1.12	1.15
T2	1.17	1.17	1.28
T3	1.13	1.11	1.11
T4	1.26	1.12	1.14
T5	1.19	1.14	1.22

Table 2 The Cost of execution of Ti at $P_c j$

Depending on the number of tasks completed, the ready list is updated, which will now contain the tasks whose parents have completed execution. We then update the average values for communication between resources according to the current network load. As the communication costs would have changed, we recompute the PSO mappings. Also, when remote resource management systems are not able to assign task to resources according to our mappings due to resource unavailability, the recomputation of PSO makes the heuristic

dynamically balances other tasks' mappings (online scheduling).

PP (3x3)=	PC1	PC2	PC3
	PC1	0	0.17	0.21
	PC2	0.17	0	0.22
	PC3	0.21	0.22	0

Table 3 The Cost of Comunication between PCi & PCj

Based on the recomputed PSO mappings, we assign the ready tasks to the compute resources. These steps are repeated until all the tasks in the workflow are scheduled.

6. CONCLUSIONS

In this work, we presented a scheduling heuristic based on Particle Swarm Optimization (PSO).We used the heuristic to minimize the total cost of execution of application workflows on Cloud computing environments. We obtained total cost of execution by varying the communication cost between resources and the execution cost of compute resources. We compared the results obtained by our heuristic against "Best Resource Selection" (BRS) heuristic.

We found that PSO based task-resource mapping can achieve at least three times cost savings as compared to BRS based mapping for our application workflow. In addition, PSO balances the load on compute resources by distributing tasks to available resources. The heuristic we proposed is generic as it can be used for any number of tasks and resources by simply increasing the dimension of the particles and the number of resources, respectively**.**

REFERENCES

[1]XunXun " From cloud computing to cloud manufacturing" Department of Technical Engineering, University of Auckland, Auckland1142, NewZealandRobotics and Computer-Integrated Manufacturing 28 (2012) 75–86

[2]OmidFatahiValilai, MahmoudHoushmand "A collaborative and integrated platform to support distributed manufacturing system using a service-oriented approach based on cloud computing paradigm "Robotics and Computer-Integrated Manufacturing 29 (2013) 110–127

[3]Alexander Verla, Armin Lechlera, Stefan Wesnerb, Andreas Kirstädterc, JanSchlechtendahla, Lutz Schubertd, Sebastian Meierc "An approach for a cloud-based machine tool control" Procedia CIRP 7 (2013) 527 – 532

[4]RaminVatankhah "Online velocity optimization of robotic swarm flocking using particle swarm optimization (PSO) method" IntJAdvManufTechnol March 24-26 2009,40 (9) :1257-1284

[5]SandhyaraniBiswas, Mahapatra S.S, "An improved metaheuristic approach for solving the machine loading problem in flexible manufacturing systems", International Journal of Services and Operations Management, Volume 5, Number 1, 2009, pp. 76 – 93

[6]HameshbabuNanvalaPonnambalam S.G, Low SengKiat, "Solving Machine Loading Problem in Flexible Manufacturing Systems Using Particle Swarm Optimization", World Academy of Science, Engineering and Technology 39 , 2008, pp.14-19.

[7]Mahapatra et al., proposed, a metaheuristic approach based on PSO to solve the machine loading problem.2008International Journal of Services and Operations Management

[8]Xi VincentWang and XunW.Xu et al., proposed an interoperable solution for Cloud manufacturing.

[9]G.H. Liu, Y.S. Wong, Y.F. Zhang and H.T. Loh proposed Modeling cloud data for prototype manufacturing. Department of Mechanical and Production Engineering, Journal of Materials Processing Technology, Vol. 138(2003), No. 1-3, p. 53-57

[10]L. Grandinetti , O. Pisacane and M. Sheikhalishahi proposed an approximate constraint method for a multi-objective job scheduling in the cloud Future Generation Computer SystemsVolume 29, Issue 8, October 2013, Pages 1901–1908**.**