ISSN:2348-2079

Volume-6 Issue-2

International Journal of Intellectual Advancements and Research in Engineering Computations

Workload Based Resource Provisioning and Scheduling

[1]Mr. C. Mani, M.C.A., M.Phil., ME., Associate Professor/MCA
[2]Mr. S. Duraishankar, Final MCA
Department of MCA, Nandha Engineering College (Autonomous), Erode-52.
E-Mail ID: cmanimca@gmail.com, duraishankar4@gmail.com

***Abstract*— In the cloud environment, the workflows have been frequently used to model large-scale problems in areas such as bioinformatics, astronomy, physics and arithmetic process. This project proposes a resource provisioning and scheduling strategy for scientific workflows on Infrastructure as a Service (IaaS) and Platform as services clouds (PaaS). This project presents an algorithm based on the Superior Element Multitude Optimization (SEMO), which aims to minimize the overall workflow execution cost while meeting deadline constraints. The main scope of the project is used to analyze best available resource in the cloud environment depend upon the total execution time and total execution cost which is compare between one process to another process. If the provider satisfies the time least time, then the process becomes to termination.**

***Index Terms*— task, workflow, cloud.**

I. INTRODUCTION

THIS document main contribution are workflows scheduling strategies and dynamic resource allocation for executing IaaS in cloud environment. The scenario is modeled as implement the multi cloud environment and its optimization problem is to minimize the overall execution cost. The proposed approach incorporates basic IaaS cloud principles heterogeneity, multi cloud, and cloud provider of the resources. The dissertation is workflow scheduling strategies implementing superior element multitude optimization execution cost reduced to particle swarm optimization execution. It aims minimize the overall execution cost and time in executing the processes in multicloud environment.

Dynamicity resource scheduling model offered by IaaS providers, there is no initial set of available resources can be use as an input to the algorithm. Instead, to have the illusion of an unlimited pool of heterogeneous VMs that can be acquired and released at any point in time. Consequently, astrategy to define an initial pool of resources that the algorithm can use to explore different solutions and achieve the scheduling objective needs to be put in place. Such strategy needs to reflect the heterogeneity of the VMs and give PSO enough options so that a suitable particle (i.e., solution) is produced. If this initial resource pool is limited, then so will be the resources that can be used to schedule the tasks. If it is very large, then the number of possible schedules becomes very large and so does the search space explored by PSO, making it difficult for the algorithm to converge and find a suitable solution.

II. RELATED WORKS

Baeza-Yates.R[1] presents a particle swarm optimization for reactive power and voltage controlconsidering voltage stability. The proposed method determines a control strategy with continuous and discrete control variables such as AVR operating values, OLTC tap positions, and the amount of reactive power compensation equipment. The method also considers voltage stability using a continuation power flow technique. The feasibility of the proposed method is demonstrated on model power systems with promising results.

Reactive power and voltage Control (Volt/VarControl: VVC) determines an on-line control strategy for keeping voltages of target power systems considering varying loads in each load point and reactive power balance in target power systems. Conventionally, VVC is usually realized based on power flow sensitivity analysis of the operation point considering execution time and available data from the actual target power system.

Cristianini N. and Shawe-Taylor J [2] presents Effective scheduling is a key concern for the execution of performance driven Grid applications. In this paper, to propose a Dynamic Critical Path (DCP) based workflow scheduling algorithm that determines efficient mapping of tasks by

calculating the critical path in the workflow task graph at every step.

It assigns priority to a task in the critical path which is estimated to complete earlier. Using simulation, It have compared the performance of the proposed approach with other existing heuristic and meta-heuristic based scheduling strategies for different type and size of workflows. The results demonstrate that DCP based approach can generate better schedule for most of the type of workflows irrespective of their size particularly when resource availability changes frequently.

Fergus.R, Fei-Fei.L, Perona.P[3] Over the last few years, Grid technologies have progressed towards a service-oriented paradigm that enables a new way of service provisioning based on utility computing models. Users consume these services based on their QoS (Quality of Service) requirements. In such "pay-per-use" Grids, workflow execution cost must be considered during scheduling based on users' QoS constraints. In this paper, they propose a budget constraint based scheduling, which minimizes execution time while meeting a specified budget for delivering results. A new type of genetic algorithm is developed to solve the scheduling optimization problem and test the scheduling algorithm in a simulated Grid testbed.

Hand D., Mannila H. and Smyth P[4] presents Large-scale applications expressed as scientific workflows are often grouped into ensembles of inter-related workflows. In this paper, to address a new and important problem concerning the efficient management of such ensembles under budget and deadline constraints on Infrastructure- as-a-Service (IaaS) clouds. To discuss, develop, and assess algorithms based on static and dynamic strategies for both task scheduling and resource provisioning. To perform the evaluation via simulation using a set of scientific workflow ensembles with a broad range of budget and deadline parameters, taking into account uncertainties in task runtime estimations, provisioning delays, and failures. To find that the key factor determining the performance of an algorithm is its ability to decide which workflows in an ensemble to admit or reject for execution. The results show that an admission procedure based on workflow structure and estimates of task runtimes can significantly improve the quality of solutions

III. SYSTEM METHODOLOGY

The existing system develops a static cost-minimization, deadline-constrained heuristic for scheduling a workflow application in a cloud environment. The approach considers fundamental features of IaaS providers such as the dynamic provisioning and heterogeneity of unlimited computing resources. To achieve this, both resource provisioning and scheduling are merged and modeled as an optimization problem. PSO is then used to solve such problem and produce a schedule defining not only the task to resource mapping, but also the number of nodes to be assigned. In the thesis the process referred in the single cloud provider which is used to compute the consumption time and execution cost for running the process in the environment. The scheduling process is done in the basis of set of resources, number of task which are defined to that resource in the environment. Here to computing the result of total consumption cost and total execution time using PSO logic.

- Adaptable only in situations where same initial set of resource availability.
- Suitable for single cloud service provider environment only.
- Data transfer cost is not considered between different cloud data centers.

The dissertation presented the algorithm named SEMO (Superior Element Multitude Optimization) which is compare the total execution time and total execution cost between one processes to another process. In addition, it extends the resource model to consider the data transfer cost between data in cloud environment so that nodes can be deployed on different regions.

Also, it assigns different options for the selection of the initial resource pool. For example, for the given task, the different set of initial resource requirements is assigned. In addition, data transfer cost between data environment are also calculated so as to minimize the cost of execution in multi-cloud service provider environment.

- Adaptable in situations where multiple initial set of resource availability.
- Suitable for multiple cloud service provider environments.
- Data transfer cost is reduced between different cloud area.

A. Cloud Providers

This module is used to add the cloud provider details to the database table. The cloud provider id and the cloud provider name are added to the table. All the record details can be viewed using the Grid View control in a form. The 'CloudProviders' table is used to store the records. The resource details must include which cloud provider id it belongs to.

B. Resources

This module is used to add the resource details to the database table. The resource id and the resource

name and cloud provider id are added to the table. All the record details can be viewed using the Grid View control in a form. The 'Resources' table is used to store the records. The cloud provider ids are fetched from the 'CloudProviders' table and any one id is selected as resource type for this record.

C. Process

This module is used to add the process details to the database table. The process id and the process name are added to the table. All the record details can be viewed using the Grid View control in a form. The 'Processes' table is used to store the records. The details regarding which process use which resource is added later. The task details contain which process it belongs to.

D. Tasks

This module is used to add the task details to the database table. The task id and the task name and process id, resource id and time taken in that resource are added to the table. All the record details can be viewed using the Grid View control in a form. The 'Task' table is used to store the records. The process ids are fetched from the 'Processes' table and any one id is selected as process id for this record. The relationship between process id and task id is one-to-many relationship.

E. Execution Time Matrix Generation

This module generates the execution time matrix in which number of resources is taken as columns and tasks are taken as rows and the time the tasks taken to complete in those resources are stored as values.

F. Transfer Time Matrix Generation

This module generates the transfer time matrix in which number of taken are taken as columns and rows (square matrix is prepared) and the time a task transfers the data to other task is stored as values. So the diagonal elements are always zero since same task has no data transfer operation.

G. Schedule Generation

Initially, the set of resources to lease R and the set of task to resource mappings M are empty and the total execution cost TEC and time TET are set to zero. After this, the algorithm estimates the execution time of each workflow task on every resource $r_iR_{initial}$. This is expressed as a matrix in which the rows represent the tasks, the columns represent the resources and the entry $ExeTime_{i,\ j}$ represent the time it takes to run task ti on resource r_j. This time is calculated using Equation (1). The next step is the calculation of the data transfer time matrix. Such matrix is represented as a weighted adjacency matrix of the workflow DAG where the entry $TransferTime_{i,\ j}$ contains the time it takes to transfer the output data of task t_i to task t_j. This value is taken from database and is zero whenever ij or there is no directed edge connecting t_i and t_j. An example of these matrices is shown in Figure a) and b) below.

At this point the algorithm has all the information needed to begin decoding the particle's position and constructing the schedule. To achieve this, it iterates through every coordinate i in the position array pos and updates R and M as follows. First, it determines which task and which resource are associated to the current coordinate and its value. This is accomplished by using the encoding strategy depicted earlier, which states that coordinate i corresponds to task t_i and its value posi corresponds to resource $rpos_iR_{initial}$. Now that the first two components of a mapping tuple are identified, the algorithm calculates the value of the remaining two, the start ST_{ti} and end ET_{ti} times of the task.

$$exeTime = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \end{array} \begin{array}{c} \begin{array}{ccc} r_1 & r_2 & r_3 \end{array} \\ \begin{bmatrix} 2 & 1 & 4 \\ 4 & 3 & 6 \\ 10 & 6 & 15 \\ 7 & 4 & 12 \\ 8 & 4 & 10 \\ 3 & 2 & 7 \\ 12 & 7 & 18 \\ 9 & 5 & 20 \\ 13 & 8 & 19 \end{bmatrix} \end{array}$$

(a)

$$transferTime = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 \end{array} \\ \begin{bmatrix} 0 & 9 & 9 & 9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

(b)

The start time value ST_{ti} is based on two scenarios. In the first case, the task has no parents and therefore it can start running as soon as the resource it was assigned to is available; this value corresponds to the current end of lease time of resource $rpos_i$, which is $LET_{rpos[i]}$.

In the second case, the task has one or more parents. In this situation, the task can start running as soon as the parent task that is scheduled to finish last completes its execution and the output data is transferred. However, if the resource is busy with another task at this time, the execution has to be delayed until such VM is free to execute t_i.

H. Pso Modeling

There are two key steps when modeling a PSO problem. The first one is defining how the problem will be encoded, that is, defining how the solution will be represented. The second one is defining how the "goodness" of a particle will be measured, that is, defining the fitness function. To define the encoding of the problem, we need to establish the meaning and dimension of a particle.

For the scheduling scenario presented here, a particle represents a workflow and its tasks; thus, the dimension of the particle is equal to the number of tasks in the workflow. The dimension of a particle will determine the coordinate system used to define its position in space. For instance, the position of a two-dimensional particle is specified by two coordinates, the position of a three-dimensional one is specified by three coordinates and so on.

Based on the, the integer part of the value of each coordinate in a particle's position corresponds to a resource index and represents the compute resource assigned to the task defined by that particular coordinate. In this way, the particle's position encodes a mapping of task to resources. There are three resources in the resource pool so each coordinate will have a value between 0 and 3.

- Coordinate 1 corresponds to task 1 and its value of 1.2 means that this task was assigned to resource 1.
- Coordinate 2 corresponds to task 2 and its value of 1.0 indicates that task 2 was assigned to resource 1.

The same logic applies to the rest of the coordinates and their values. Since the fitness function is used to determine how good a potential solution is, it needs to reflect the objectives of the scheduling problem. Based on this, the fitness function will be minimized and its value will be thetotal execution cost TEC associated to the schedule S derived from the particle's position.

How this schedule is generated is explained later in this section. Because ofthe elasticity and dynamicity of the resource acquisition model offered by IaaS providers, there is noinitial set of available resources we can use as an input to the algorithm. Instead, we have the illusion of an unlimited pool of heterogeneous VMs that can be acquired and released at any point in time. Consequently, a strategy to define an initial pool of resources that the algorithm can use to explore different solutions and achieve the scheduling objective needs to be put in place.

Such strategy needs to reflect the heterogeneity of the VMs and give PSO enough options so that a suitable particle (i.e., solution) is produced. If this initial resource pool is lim-ited, then so will be the resources that can be used to schedule the tasks. If it is very large, then the number of possible schedules becomes very large and so does the search space explored by PSO, making it difficult for the algorithm to converge and find a suitable solution.

Proposed Semo Algorithms

Input: Set of workflow task T, Initial Resources R,
Set DimensionalParticle*dp*, Set EntropyΘ,
Set Optimal Best *opbest*, Set Optimal Global Best*ogbest*

Output: Multi cloud Provider Scheduling

1. Set the dimension of the particle to *dp*
2. Initialized the population of particles with random position and velocities
3. For each particle, calculated its Entropy values Θ
   a. Compare the particle's Entropy Θ value with the particle's *opbest*.
      - If the current Θ values is better than *opbest* then set *opbest* to the current value and location
   b. Compare the particle's Entropy Θ value with Global best *ogbest.*
      - If the current Θ values is better than *ogbest* then set *ogbest* to the current value and location
   c. Update the position and velocity of the particle
      $V X_i (t+1) = V X_i (t) + V V_i (t)$
4. Repeat from Step 3 until the stopping criterion is met

The range in which the particle is allowed to move is determined in this case by the number of resources available to run the tasks. As a result, the value of a coordinate can range from 0 to the number of VMs in the initial resource pool. Based on this, the integer part of the value of each coordinate in a particle's position corresponds to aresource index and represents the compute resource assigned to the task defined by that particular coordinate. In this way, the particle's position encodes a mapping of task to resources.

IV. CONCLUSION

The paper presented the ESEMO (E-Superior Element Multitude Optimization) algorithm which is used to predict the least time computation in the cloud provider area. In addition, the thesis compared the time evaluation work between one dynamic

resource flow to another process flow of dynamic resource in the cloud environment. In addition, it extends the resource model to consider the data transfer cost between data centers so that nodes can be deployed on different regions. Extending the algorithm to include heuristics that ensure a task is assigned to a node with sufficient memory to execute it will be included in the algorithm. Also, it assigns different options for the selection of the initial resource pool. In addition, data transfer cost between data centers are also calculated so as to minimize the cost of execution in multi-cloud service provider environment. The main contribution of paper, the following problem solve in the existing system, they contribution are

- Adaptable in situations where multiple initial set of resource availability.
- Suitable for multiple cloud service provider environments.
- Data transfer cost is reduced between different cloud data centers
- The thesis contribution will be implemented in the real time in cloud environment.
- In addition, the future enhancement will plan toanalyze the load balancing work in the cloud area.
- The application is developed such that above said enhancements can be integrated with current modules.

REFERENCES

[1] Baeza-Yates.R, and Ribeiro-Neto.B, "Modern Information Retrieval". Addison-Wesley, June 1999.

[2] Cristianini N. and Shawe-Taylor J., An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, 2000.

[3] Fergus.R, Fei-Fei.L, Perona.P, and Zisserman.A. Learning object categories from google's image search.

[4] Hand D., Mannila H. and Smyth P., Principles of Data Mining, MIT Press, 2001.

[5] Strachey, Christopher (June 1959). "Time Sharing in Large Fast Computers". Proceedings of the International Conference on Information processing, UNESCO. paper B.2.19: 336–341.

[6] Larose D.T., Discovering knowledge in data: an introduction to data mining, Wiley-Interscience, 2005.

[7] Mitchell T.M., Machine learning, McGraw-Hill, 1997.

[8] Pal S.K. and Mitra P., Pattern Recognition Algorithms for Data Mining, CRC Press, 2004.

[9] Pankaj Jalole ,"An Integral approach to software engineering",Narosa publishing Home-3rd Edition.

[10] Smith, David Mitchell. "Hype Cycle for Cloud Computing", 2013Gartner. Retrieved 3 July 2014.

[11] Webb A., "Statistical Pattern Recognition", Wiley, 2002.

[12] Weber.M, Welling.M, and Perona.P. Unsupervised learning of models for recognition.