OPTIMIZATION OF SCIENTIFIC WORKFLOW SCHEDULING IN CLOUD ENVIRONMENT THROUGH A HYBRID SYMBIOTIC ORGANISM SEARCH ALGORITHM

Nazia Anwar, Huifang Deng
School of Computer Science and Engineering, South China University of Technology, China
Corresponding: naziaanwar12@yahoo.com

ABSTRACT*— With the emergence of e-Science, work flow scheduling has become a significant component for the representation of complex multiple dependent tasks and the flow of control between them. Optimization of scientific workflow scheduling is a well-known NP-hard [1] optimization problem and is more challenging in the cloud computing environment. Therefore, it is almost intractable to find exact solution for dynamic and heterogeneous environments such as a cloud. The complexity of intricately connected tasks in scientific workflow applications is compelling researchers to explore hybrid techniques to get an optimum solution of the workflow scheduling problems. The systematic reasoning ability and strong global exploration ability of Symbiotic Organism Search (SOS) algorithms can be enhanced by involving an efficient heuristic and modifying the ecosystem organisms to find better solutions on local solution regions. These features are employed in the proposed algorithm to get better schedules in terms of efficient resource utilization, and reduced makespan.*

1. INTRODUCTION

Cloud computing has emerged as an effective distributed computing utility which may be exploited for running large and complex scientific workflow applications [1, 2]. However, heterogeneous distributed environments such as cloud may incur significant overheads to the workflow execution. This may adversely affect the overall performance and makespan of the workflow [3]. Although, clouds provide an easily accessible, flexible, and pay-as-you-go model, but these delays may add additional financial overheads to consumers for execution of their scientific workflow applications. Thus, in order to minimize the overheads incurred, one of the most challenging problems in workflow scheduling algorithms is minimizing the time of workflow execution.

Various scientific domains such as planetary science, astronomy, physics, bioinformatics, and environmental science, often involves the use of simulations of large-scale complex applications for validating behavior of different real-world activities. Most of such scientific applications are constructed as workflows containing a set of computational tasks linked via control and data dependencies. The workflow is partitioned into tasks which might have different sizes and require different running times ranging from tens of seconds to multiple minutes. Efficiently executing such workflows within a reasonable amount of time usually require massive storage and large-scale distributed computing infrastructures such as cluster, grid, or recently the cloud. Among such infrastructures, cloud computing has been emerged as an economical and scalable environment for the design and execution of workflow applications.

Several heuristic based and metaheuristic based workflow scheduling algorithms in cloud computing have been proposed. Heuristic based techniques attempt to find optimum solution on the basis of some pre-defined rules. Hence, the schedule map for complex task scheduling problems, acquired by these techniques are infeasible and dependent on the problem size and underlying rules. Moreover, in heuristic methods, the computations needed to find optimal solutions increased exponentially in large search space such as the cloud.

The main contribution of this study is a novel workflow scheduling method, namely the Hybrid Symbiotic Organism Search (SOS). Our approach takes into account a heuristic to achieve an optimum schedule efficiencies in cloud environment. SOS algorithms have strong global exploration and faster convergence capabilities by using only a few parameters. The heuristics wer employed to enhance the performance of meta-heuristic optimization technique, SOS, to explore the search space in less makespan and produce a near-optimal workflow schedule.

The standard way to represent a scientific workflow application is a Directed Acyclic Graph (DAG) in which the set of vertices represent different tasks of the workflow and the directed edges between the vertices represent data or control dependencies between the tasks. The same DAG model is implemented in our study. The behavior of the proposed scheduling method was validated with diverse set of benchmark workflow traces from synthesized and real-world applications, such as CyberShake, and Montage.

Our main contributions are as follows.

- Design and implementation of hybrid version of SOS algorithm for scheduling of scientific workflows in a cloud computing environment.
- An objective function to find a strong global optimal solution for workflow scheduling of scientific applications that take into account improved resource utilization, and reduced makespan, load imbalance, and response time.

2. RELATED WORK

2.1 BACKGROUND

Several metaheuristic methods [4, 5] have been implemented to solve the issues of task assignment in order to minimize makespan and response time. The most commonly employed nature-inspired heuristics in cloud computing systems are ACO, PSO, GA, and their variants. Workflow scheduling problems have been widely studied using PSO with the aim of reducing communication cost and makespan.

2.2 THE SOS ALGORITHM

Symbiotic Organism Search (SOS) algorithm [6] is a comparatively new population-based metaheuristic approach for the solution of numerical optimization that simulates the symbiotic associations (mutualism, commensalism, and parasitism) between organisms in an ecosystem. Figure 1 shows a group of organisms in an ecosystem with their symbiotic relations.

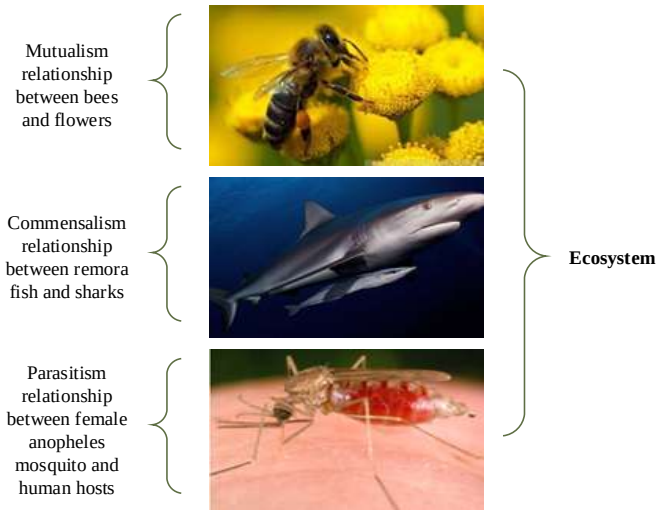

Figure 1. Symbiotic relations among natural organisms in an ecosystem

Mutualism simply means a symbiotic relationship in which both individuals benefit from the interaction. Commensalism is the relationship where one benefits from the interaction and the other is not damaged. Parasitism is the interaction where one specie is helped and other is harmed.

In SOS, the members of the ecosystem that survive in the solution space are denoted as best. SOS has some features which are same as in most of the other nature-inspired methods, such as the population of organisms representing candidate solutions, the use of operators for searching best solution, and the need to initialize population size and termination condition before starting the search process. However, SOS provides more flexibility in selecting the algorithm specific metrics. The pseudo code of SOS is given in Algorithm 1.

Algorithm 1: Pseudocode of SOS algorithm
Input: Initialize Ecosystem size, Initial population of organisms, Termination condition
Output: Optimal schedule
1 Repeat
For (all organisms in the ecosystem)
2 Identify the best organism;
3 Mutualism Phase;
4 Commensalism Phase;
5 Parasitism Phase;
6 End For
7 Until (termination condition is not met)

2.2.1 Step 1: Ecosystem initialization

The SOS method begins with initializing randomly generated population of organisms denoted as ecosystem. Thereafter, some control variables like ecosystem size, maximum number of iterations, are assigned values. The fitness value of an organism represents its extent of adaptation to the desired objective.

2.2.2 Step 2: Best organism identification

The organism with the best fitted objective function is selected represented as X_{best}, where X is a vector of optimization variables:

$$X = \{X_1, X_2, X_3, \ldots, X_n\}, \quad (1)$$

The fitness value of each member is computed iteratively by means of their relevant positions as input in the three phases of the SOS method.

$$X_i = round(X_i \bmod m) \quad (2)$$

where m is the number of resources for executing the submitted tasks.

2.2.3 Step 3: Mutualism phase

In this step, the ith member, X_i of the ecosystem interacts with a randomly selected organism, X_j, for mutual benefit, where $i \neq j$. New candidate solutions X_α and X_β are calculated on the basis of their symbiotic relationship.

2.2.4 Step 4: Commensalism phase

Just like in mutualism phase, X_j, is selected randomly to associate with X_i, $i \neq j$. In this situation, X_i attempts to benefit from the association. However, X_j itself neither gains nor loses from the association.

2.2.5 Step 5: Parasitism phase

In ith iteration, an artificial parasite, X_p is generated by duplicating X_i and is assigned a role analogous to the anopheles mosquito. Afterwards, X_p is modified by using a random number. A random organism X_j is selected as a host to X_p. If X_p is fitter than X_j, it will replace X_j. In other case, X_p will no longer be able to survive.

2.2.6 Step 6: Termination criterion

Steps 2 through 5 are repeated until stopping criterion is reached.

3 System model

This section describes the cloud model, scientific workflow application model, and the overall architecture of the computing framework for workflow execution in cloud.

3.1 Cloud model

The IaaS cloud model, assumed in this work, offers a set of virtualized resources. In this study, the cloud resource model for executing scientific workflow applications consist of a set of homogeneous Virtual Machines (VMs) of a given type. The VMs are acquired before starting the workflow execution and are released after terminating the execution. A task can be scheduled to a single VM at a time and it cannot be started until the execution of its predecessor tasks.

Our main goal is to schedule tasks on virtual machines such that the makespan is minimized and resource utilization is maximized. Therefore, the proposed scheme use Estimated Execution Time (EET) of a task v_i to assign it to virtual machine m_r.

3.2 Scientific workflow application model

The standard way to represent a scientific workflow application is a Directed Acyclic Graph (DAG), $W = (V, E)$ in which $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices representing *n* different tasks of the workflow, where $v_i \in V | 1 \leq i \leq n$

Each task has a weight $w(v_i)$ which denotes the precedence constraint or weight assigned to tasks v_i. It represents the estimated average execution time of the task.

Also, each edge has a weight $w(e_{ij})$ which denotes the amount of data needed to be communicated and the precedence constraint from tasks v_i and v_j, if they are executed on different resources. It is predetermined and known a priori.

4 The proposed algorithm HSOS

The proposed algorithm is a hybrid scheme that combines a heuristic and symbiotic organism search algorithm.

The HSOS algorithm starts by creating and initializing a set of schedules (i.e. initial population, X). The population size n, and the number of VMs k are supposed to be user input. Algorithm begins with a set of organisms named as initial population. First, the tasks of the workflow are assigned priorities in order to guarantee task dependencies. For assigning priorities to the workflow tasks, their upward ranks [7] are calculated. The priority of tasks calculated using this technique has several benefits, which make it a worthy candidate for directed search. Its implementation is simple. It considers the execution time of the workflow tasks, transfer time between each pair of the tasks and the critical path of the workflow. Besides, this technique ensures that every task of the workflow has a higher priority over its successor tasks. The priorities of the nodes are calculated upwards starting from the exit task v_{exit} of the workflow. Finally, the task schedule contains tasks sorted in descending order of their priorities. This initial schedule benefits HSOS to converge after a less number of iterations.

The members of the population are organisms. The organisms are represented in a matrix. The rows of the matrix represents the number of tasks which shows their scheduling sequence. The columns represent the VM allocation. In order to find a valid schedule, the dependency constraints should not be violated. Each organism represents a valid schedule. Each organism in each schedule is evaluated on the basis of the fitness function. Then the organism with the best fitness value X_{best} is updated. Afterwards, the organisms undergo mutualism, commensalism and parasitism procedures. The Load Balancing procedure allocates the tasks across the VMs in such a way that all the VMs execute the tasks with minimum difference in FT. As a result, VM utilization is improved by avoiding the VMs from being overloaded or stay idle for a long time.

5 Performance analysis and discussion

The performance of the proposed algorithm was evaluated. The HSOS was evaluated by using real scientific datasets with diverse characteristics and the obtained results were compared with the selected algorithms. We selected heuristic HEFT, hybrid heuristic algorithm HSGS, and LBACO for comparative analysis with the proposed algorithm. These algorithms are based on different schemes so they offer sound grounds to study and compare the behavior of the HSOS. HEFT is a well-known heuristic that provides good schedules. We generate an initial schedule using the HEFT schedule, that accelerates the process to reach an optimal schedule with better makespan.

5.1 EXPERIMENT CONDITIONS

WorkflowSim [8] is a Java-based open source discrete event workflow engine that has been used to model a cloud execution environment for executing scientific workflow applications. WorkflowSim offers support for workflow DAGs and provides execution environment for workflow level resource provisioning, resource management, task clustering, and task scheduling. For our experiments, WorkflowSim was adopted to evaluate the performance of the proposed method on real traces under varying system metrics, such as number of VMs, size of workflows, and average data size.

WorflowSim provides support for creating data centers, which include VMs to run workflows. The simulated computing platform was composed of a VM cluster including 20 homogeneous single core virtual machines. Each simulated VM has 512 MB of memory, 1024 MIPS processing capacity. This is similar to a typical distributed system, for instance Amazon EC2 [9] and FutureGrid [10]. Five real-world scientific applications were chosen [11], namely: Cybershake (data-intensive, memory-intensive, resource-intensive), Epigenomics (CPU-bound), LIGO Inspiral Analysis (memory-intensive, resource-intensive), Montage (I/O bound), and SIPHT.

A comprehensive evaluation of the proposed algorithm and its comparison with the baseline algorithms was performed to identify its ability in improving the overall makespan, VM utilization, and load balancing of the workflow. The plot in Figure 1 shows the performance of algorithms in terms of makespan when the number of workflow tasks was increased. The proposed algorithm showed improvement over HEFT, HSGA, and MBACO for Montage and CyberShake with varied number of tasks. Figure 5 shows that for 30 workflow nodes, the average percentage improvement of the HSOS over LBACO, HEFT, and HSCGS was 74%, 30%, and 14% respectively. Similar experiments were performed for CyberShake and of 50, and 100 workflow tasks. The results showed that the proposed method has increased performance as the size of the workflow was increased. However, the HSOS significantly outperforms the LBACO in case of the CyberShake workflows. The better performance of the HSOS with the increasing size in CyberShake workflows proves the scalability of the HSOS.

Similar experiments were performed with the Montage workflows of 25, 50, and 100 nodes. The results obtained show a better performance of th proposed algorithm for Montage workflow, as compared to the other three algorithms. The experiments carried out for 25 nodes Montage workflow showed that the HSOS is better than the LBACO by 37%, HEFT by 6%, and HSCGS by 3%. It is evident that the performance of the proposed method is better for CyberShake workflows as compared to Montage workflows. Both types of workflows have varied features, CyberShake is data-intensive workflow as compared to Montage workflow. The average performance gain of HSOS over HSCGS, HEFT, and LBACO was 6%, 9%, and 27% for

CyberShake and Montage workflows respectively. The proposed algorithm outperforms the aforementioned schemes. It can be seen that the proposed method has significant performance gain over LBACO. In the proposed algorithm the heuristic accelerates the process to find an optimal schedule and modified SOS metrics help to search the problem space efficiently. These features dominate the HSOS among other algorithms and it outperforms completely. The Load Balancing procedure allocates the tasks across the VMs in such a way that all the VMs execute the tasks with minimum difference in FT. As a result, VM utilization is improved by avoiding the VMs from being overloaded or stay idled for a long time.

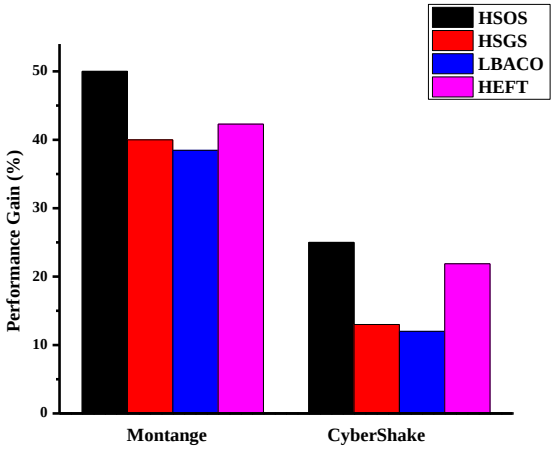

Figure 1. Average makespan of each workflow with 30 tasks

6 CONCLUSION

In this paper, a new hybrid SOS algorithm (HSOS) for scientific workflow scheduling is presented by modifying a metaheuristic optimization based approach, symbiotic organism search. HSOS simulated this natural pattern using the three strategies of mutualism, commensalism, and parasitism. The proposed algorithm HSOS presents efficient scheduling of scientific workflows on VMs. It is a hybrid method that leverage the strength of heuristic based schedule in the initial population that helps the proposed algorithm to achieve an optimal schedule in fewer iterations. The proposed algorithm employs population diversity to identify the fitness of each organism in terms of its capability to produce a better schedule. The proposed algorithm optimizes workflow makespan with efficient utilization and load balancing of virtual machines. The scheduling algorithms with different approaches are compared with the HSOS. The results prove that HSOS outperforms in terms of quality of schedules having minimized makespan, better load balancing, and efficient resource utilization. In future, we intend to develop hyperheurisctics for this issue.

REFERENCES

[1] K. Ostrowski, K. Birman, and D. Dolev, “Extensible architecture for high-performance, scalable, reliable publish-subscribe eventing and notification,” International Journal of Web Services Research, vol. 4, no. 4, pp. 18–58, 2007.

[2] A. Lathers, M. H. Su, A. Kulungowski et al., “Enabling parallel scientific applications with workflow tools,” in Proceedings of the Challenges of Large Applications in Distributed Environments (CLADE '06), pp. 55–60, June 2006.

[3] W. Chen, E. Deelman, Workflow overhead analysis and optimizations, in: Proceedings of the 6th Workshop on Workflows in Support of Large-Scale Science, ACM, 2011, pp. 11–20

[4] Xue Shengjun and Li Mengying and Xu Xiaolong and Chen Jingyi and Xue Shengjun. An ACO-LB algorithm for task scheduling in the cloud environment. Journal of Software. 2014; 9(2):466–473.

[5] Abdullahi Mohammed and Ngadi Md Asri and Abdulhamid Shafi’i Muhammad. Symbiotic Organism Search optimization based task scheduling in cloud computing environment. Future Generation Computer Systems. 2016; 56:640–650.

[6] Cheng, M. Y., & Prayogo, D. (2014). Symbiotic Organisms Search: A new metaheuristic optimization algorithm. Computers and Structures, 139, 98–112.

[7] Haluk Topcuoglu, Salim Hariri, and Min-you Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. Parallel and Distributed Systems, IEEE Transactions on, 13(3):260–274, 2002.

[8] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," IEEE 8th International Conference on E-Science (e-Science), 2012, pp. 1-8.

[9] Amazon.com, Inc., Amazon Web Services. http://aws.amazon.com.

[10] [f-105]FutureGrid. http://futuregrid.org/

[11] Gideon Juve, Ann Chervenak, Ewa Deelman, Shishir Bharathi, Gaurang Mehta, and Karan Vahi. Characterizing and profiling scientific workflows. Future Generation Computer Systems, 29(3):682–692, 2013