Advances in Science, Technology and Engineering Systems Journal Vol. 6, No. 3, 40-44 (2021)

www.astesj.com

Special Issue on Innovation in Computing, Engineering Science & Technology

ASTESJ

ISSN: 2415-6698

Plummeting Makespan by Proficient Workflow Scheduling in Cloud Environment

Juhi Singh, Shalini Agarwal*

Department of Computer Science, Shri Ramswaroop Memorial University, Lucknow, 225003, India

ARTICLE INFO

Article history:
Received: 21 January, 2021
Accepted: 12 April, 2021
Online: 05 May, 2021

Keywords:
Cloud Computing
Deadline
Decisive Path
Makespan
Scheduling
Workflow

ABSTRACT

Cloud is an Internet-based computing technology in which on-demand shared resources such as software, platforms, repositories, and information are delivered to customers. In the emerging era of computing cloud environment provide the use of resources with the concept of virtualization. Workflow of the tasks has vital role for the improvement of computing performance which leads to improved quality of service. As per the demand of user's number of tasks are scheduled in such a way so that better performance is computed using partial deadline of the workflow. In this paper we have introduced with the workflow concepts, further we aim to diminish makespan for the proposed workflow scheduling algorithm. Here makespan refers to overall time duration taken for the sequence of tasks, by the resources so as to complete the execution of each and every task.

1. Introduction

Every day, everyone is connected in one way or another to this digital world, and this is because the field of information technology is escalating. User-friendly environment is the main factor of internet, so that diverse groups of individuals like students, researchers and business people complete their work by providing numerous opportunities to accomplish their goals. Many users connect to the Internet and use their IT infrastructure to meet their daily needs. As the demand for the internet increases, the services provided over the internet such as software, platforms, database services, storage services, etc. are also escalating. Here the imperative term cloud computing comes into existence. It provides countless diverse services to its users over a network. Because of concept "Pay as you Go", end user can get the most out of using this service at a lower cost.

1.1. Workflow

The increasing demand and heterogeneity of cloud computing is gaining recognition among the scientific commune to leverage cloud services to implement large-scale electronic applications. These applications are in the form of a set of tasks representing workflow. Computations performed considering task dependencies are regarded as Workflows. It allows users to straightforwardly elucidate multifaceted multi-step calculation tasks. Workflow tasks are associated to the mechanization of procedures where tasks, information or documents are delivered between partakers in accordance with a specific policy set and allow the formation of different applications in Directed Acyclic Graph. Each and every in DAG is represented as task and each edge symbolizes the dependencies [1], [2].

A mechanism to manage various workflows on cloud, is known as Workflow Management System (WMS). Figure 1 depicts some significant constituents of WMS. Workflow can be categorized as:

- Single workflow - It includes one or more instances using the same architecture
- Multiple workflow – It can take into account manifold cases of diverse structures of workflow.

Figure 1: Components of Workflow Management System

1.2. Workflow Design

Figure 2 portrays the workflow design mechanism. It is concerned with how to delineate and configure constituent of a

*Corresponding Author: Juhi Singh, Department of Computer Science, SRMU, Lucknow, India, 225003, juhisingh.srmcem@gmail.com

workflow. The workflow portrays the relationship and dependencies between workflows. DAG and Non-DAG [3] are two categories of workflow design. Directed Acyclic Graph is further characterized as Selection, Sequence and Parallel, while Non-DAG can be classified as Repeat, Parallel and Selection. In sequence architecture, the tasks are executed in a sequence, whereas in a parallel architecture, the workflows can be executed synchronously.

In a selection structure, workflows can run in sequence or in parallel. The recurrence pattern structure performs tasks iteratively [4]. Another is the workflow model, which is a constituent of workflow design, delineates the workflow at both the task and structure levels in both abstract and concrete workflows. Abstract workflows are characterized as a nonfigurative template with no commit to cloud resources to carry out the tasks, while concrete workflows are called actionable workflows. The workflow configuration permits users to coalesce diverse components through user-oriented and stand-alone systems.

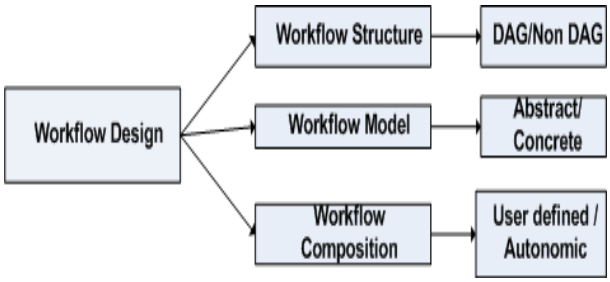

Figure 2: Workflow Design Components

1 . 3 . Workflow Scheduling

The workflow scheduler is necessity for the arrangement of workflows task on the cloud resources that are utilized to implement the workflow. The components that must be delineated to schedule a workflow are revealed below in Figure 3:

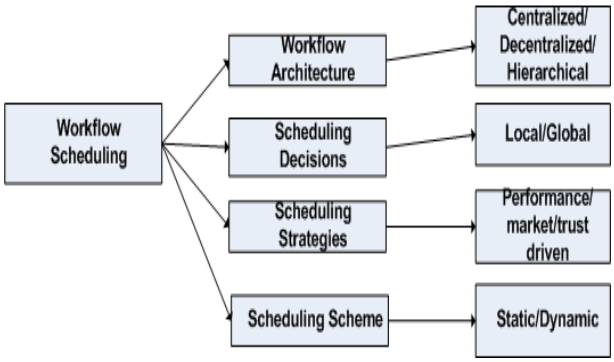

Figure 3: Workflow Scheduling Components

1.4. Fault Tolerance in Workflow

Fault tolerance is linked to tackle errors that can take place during the scheduling and execution phase of workflow tasks for various reasons such as unavailability of resources, resource breakdown, task malfunction, resource overload, network collapse, out of memory, etc

2. Literature Review

Arrangement of tasks can affect cloud system performance, so numerous workflow techniques as well as scheduling algorithms for scientific workflows have been studied and discussed below:

- Enhanced Scheduling of Resources: In [5], the author introduced a scheduling algorithm to attain optimization or more precise to sub-optimization for scheduling tasks. The authors exploited IGA (Improved Genetic Automated Scheduling Policy) to produce better results.
- Transaction Exhaustive Cost Restraint: In [6], author introduced an algorithm for scheduling tasks that took into account time and cost. Their simulation showed that this algorithm diminishes costs while adhering to deadlines.
- In [7], a superior algorithm based on cost was proposed by author. This algorithm capably assigns tasks to available resources in the cloud. Resource cost, computing performance, convalescing computing connectivity ratio is evaluated in this algorithm.
- In [8], author introduced a new cost scheduling algorithm based on deadlines. It took into account the features of cloud computing to hold cost-concentrated and limited workflows. It diminishes execution time and cost while enabling consumer input instantly.
- Inferences based on PSO for programming workflow applications: In [9], author introduced computation and data communication for applications that include the cost for both and provided a guideline based on Particle Clustering Optimization (PSO). This algorithm can be utilised for workflow applications that have dissimilar computing and communication overheads. Experiential results showed that PSO can attain cost savings and well distribute workloads to cloud resources. Also in [10], authors expanded PSO to provide deadline-based resource scheduling and provisioning. However, these authors did not explain resource failures or extreme dependence on essential tasks.
- Market-oriented hierarchical scheduling technique was proposed by author in [11], which includes both the levels of scheduling that is task-level scheduling and service-level scheduling, where task-level scheduling concept copes with optimizing device allocation from a task to a VM on cloud data centers and service-level scheduling concept copes with task assigned to service.
- Stretchy workflow scheduling: In [12], author proposed SHEFT workflow scheduling algorithm which is a stretchy workflow scheduling in cloud environment. Investigational upshots showed that this algorithm performs better than various other workflow scheduling algorithms. This algorithm perks up workflow uptime, as well as it also facilitates resources to flexibly measure uptime.
- Multi-workflow Multi-QoS (MQMW) constrained scheduling strategy: In [13], research work author introduced a scheme for workflows with multiple QoS. Authors boosted the rate of access to scheduling and also diminished the duration and outlay of workflows for the cloud platform, so improved Quality of Service was proposed for multiple workflows using constrained scheduling strategy.
- In [14], research work author suggested SA to schedule all the tasks on platform with the aspire of plummeting execution time, but they did not analyse malfunctioning.

3. Proposed Workflow Scheduling Approach

The literature survey on various workflow algorithms is done one different aspect. The proposed workflow scheduling

algorithm generates planned scheme of tasks which reduces the entire cost of implementing a workflow which meets a deadline defined by user. The algorithm consists of two chief stages:

- Deadline allocation
- Scheduling.

Deadline allocation doles out the time limit for the overall workflow between individual tasks. Scheduling arranges every task with the economical service that is able to run the task earlier than its deadline.

The proposed algorithm focuses on the two concepts of task start times, which is Earliest Begin Time (EBT) and Actual Begin Time (ABT). EBT is calculated prior to schedule of workflow while the ABT is calculated when the tasks are arranged. The EBT of each non-arranged task t_i, EBT (t_i), is delineated as follows:

$$EBT(t_{entry})=0 \quad (1)$$

$$EBT(t_i)= \max_{t_p \in t_i s \ parents} \{EBT(t_p)+LET(t_p)+TT(e_{p,i})\} \quad (2)$$

where

LET(t_i): Least Execution Time of a task t_i, on service $s_j \in S$ with the least ET (t_i, s_j) between all available services.

TT($e_{p,i}$) is the transmission time from parent node to task t_i.

$$LET(t_{entry}) = LET(t_{exit}) = 0. \quad (3)$$

As a result, the Earliest Completion Time (ECT) of an unscheduled task t_i, ECT (t_i), can be delineated as follows:

$$ECT(t_i) = EBT(t_i) + LET(t_i) \quad (4)$$

In addition, we delineate Latest Completion Time (LCT) of task that are unscheduled t_i, LCT (t_i), refers the latest time for t_i which can terminate its execution such that the deadline D of the entire workflow is reduced.

It can be calculated as follows:

$$LCT(t_{exit})=D \quad (5)$$

$$LCT(t_i) = \min_{t_c \in t_i s \ children} \{LCT(t_c) + LET(t_c) + TT(e_{p,c})\} \quad (6)$$

The Service that is selected for every arranged task t_i, SS (t_i) = $s_{j,k}$, is delineated as the selected service for executing t_i while scheduling process, where $s_{j,k}$ is the k_{th} occurrence of service s_j.

The Decisive Parent of a node t_i refers to the not assigned parent of t_i that has the newest arrival time of data at t_i. It is taken as the parent t_p of t_i for which ECT (t_p)+ TT ($e_{p,i}$) which has maximal value.

The partial Decisive Path (PDP) of a node t_i is:

- Blank if t_i does not have some unassigned parents.
- Composed of Decisive Parent t_p of t_i and the PDP of t_p if has any unassigned parents.

Algorithm 1 : Scheduling Algorithm

```
procedure Schedule_Workflow (G (T, E), D)
{
Step 1: verify existing computation services

Step 2: add t_entry , t_exit and their corresponding dependencies to G

Step 3: calculate EBT (t_i), ECT (t_i) and LCT(t_i) for each task in G

Step 4: ABT(t_entry) ← 0, ABT(t_exit) ← D

Step 5: mark t_entry and t_exit as assigned

Step 6: call Assign_Parents(t_exit )
}
```

Algorithm 2 : Algorithm for Parents Assignment (Step 6)

```
procedure Assign_Parents (t)
{
  while (t has any unassigned parent)
  {
    PDP = null , t_i = t

    While (t_i has any unassigned parent)
    {
      add Decisive_Parent (t_i) at the start of PDP

      t_i = Decisive_Parent (t_i)
    }

    Step 7: call Assign_Path(PDP)

    for each t_i ∈ PDP
    {
    update EBT and ECT for every successor of t_i

    update LCT for every predecessor of t_i

    Step 6: call Assign_Parents(t_i)
    }
  }
}
```

Algorithm 3 : Algorithm for Path Assignment (Step 7)

```
procedure Assign_Path(P)
{
  s_i,j = the cheapest applicable existing instance for P
    if (s_i,j = null)
    {
     launch a new instance s_i,j of the cheapest service s_i
     which can finish each task of P before its LCT
     }

   Schedule P on s_i,j and set SS (t_i ), ABT (t_i ) for each t_i ∈ P
   Set all tasks of P as assigned
}
```

4. Performance Evaluation

To appraise the proposed algorithm, we need to measure its performance in some workflow models. One of the structures

we used in this article is the Montage. Figure 4 depicts the approximate structure format of this workflow:

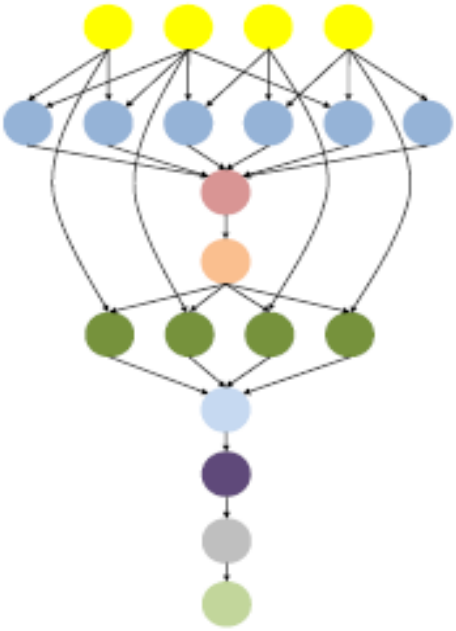

Figure 4: Structure of Montage workflow

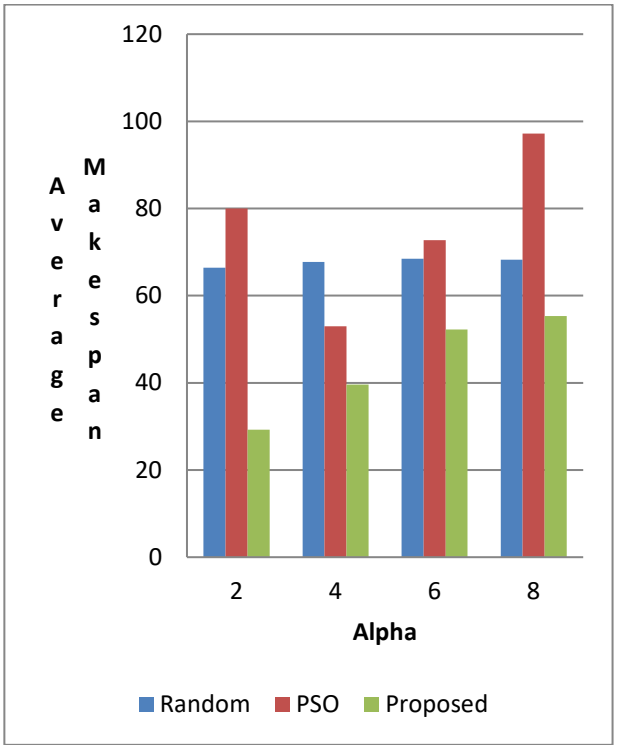

Figure 5: Comparison of average makespan for proposed algorithm

For calculation of the effectiveness of the proposed algorithm, we use makespan as a parameter. Makespan is the total time taken for resources to complete the overall execution of the tasks.

For evaluation of our proposed algorithm, we set a deadline for each workflow. We first delineate the fastest schedule. MF indicates the makespan that is fastest program in a workflow. It is only a least duration for that workflow to execute, so to establish time limit for every workflow, we delineate the deadline with factor α and delineate the new deadline of the workflow with its arrival plus α * MF. Since there is no elucidation for α = 1, we take the range of α from 2 to 8 in our experiment.

Below is a comparison of the proposed approach with the stochastic approach and the PSO [9]. The graph is plotted for the parameter value of average makespan and alpha factor as deadline of the workflow.

Table 1: Comparison of average makespan for proposed algorithm

Alpha Value	Algorithm	Average Makespan
Alpha Factor =2	Random	68
	PSO	80
	Proposed	30
Alpha Factor =4	Random	68
	PSO	52
	Proposed	40
Alpha Factor =6	Random	68
	PSO	72
	Proposed	54
Alpha Factor =8	Random	68
	PSO	97
	Proposed	56

5. Conclusion

In this research paper, we aimed to design an algorithm that takes an approach to schedule workflow in one phase. The approach is intended to schedule each and every partial decisive path into a one occurrence of a computation service. The time complexity of algorithms is $O(n^2)$, where n indicates the number of tasks in the workflow. The complexity of polynomial time makes it appropriate for bulky workflows. We assess our algorithm by measuring its recital in Montage synthetic workflow. The results show that the proposed algorithm works better than other approaches.

References

[1] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, S. Hu, "Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT," Future Generation Computer Systems, **93**, 278–289, 2019, doi:10.1016/j.future.2018.10.046.

[2] J. Yu, R. Buyya, "Scheduling Scientific Workflow Applications with Deadline and Budget Constraints Using Genetic Algorithms," Scientific Programming, **14**, 217–230, 2006, doi:10.1155/2006/271608.

[3] J. Yu, R. Buyya, "A Budget Constrained Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms," Workshop on Workflows in Support of Large-Scale Science, 2006, doi:10.1109/WORKS.2006.5282330.

[4] J. Yu, R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," Journal of Grid Computing, **3**(3), 171–200, 2005, doi:10.1007/s10723-005-9010-8.

[5] H. Zhong, K. Tao, X. Zhang, An Approach to Optimized Resource Scheduling Algorithm for Open-Source Cloud Systems, 2010, doi:10.1109/ChinaGrid.2010.37.

[6] Y. Yang, K. Liu, J. Chen, X. Liu, D. Yuan, H. Jin, An Algorithm in SwinDeW-C for Scheduling Transaction-Intensive Cost-Constrained Cloud Workflows, 2008, doi:10.1109/eScience.2008.93.

[7] S. Selvarani, S. Sadhasivam, "Improved cost-based algorithm for task scheduling in cloud computing," Computational Intelligence and Computing Research (ICCIC), 2010, doi:10.1109/ICCIC.2010.5705847.

[8] K. Liu, H. Jin, J. Chen, X. Liu, D. Yuan, Y. Yang, "A Compromised-Time-Cost Scheduling Algorithm in SwinDeW-C for Instance-Intensive Cost-Constrained Workflows on a Cloud Computing Platform," IJHPCA, **24**, 445–456, 2010, doi:10.1177/1094342010369114.

[9] S. Pandey, L. Wu, S. Guru, R. Buyya, A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments, 2010, doi:10.1109/AINA.2010.31.

[10] M.A. Rodriguez, R. Buyya, "Deadline Based Resource Provisioningand Scheduling Algorithm for Scientific Workflows on Clouds," IEEE

Transactions on Cloud Computing, **2**(2), 222–235, 2014, doi:10.1109/TCC.2014.2314655.

[11] Z. Wu, X. Liu, Z. Ni, D. Yuan, Y. Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems," The Journal of Supercomputing, **63**(1), 256–293, 2013, doi:10.1007/s11227-011-0578-4.

[12] C. Lin, S. Lu, Scheduling Scientific Workflows Elastically for Cloud Computing, 2011, doi:10.1109/CLOUD.2011.110.

[13] M. Xu, L. Cui, H. Wang, Y. Bi, "A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing," in 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, 629–634, 2009, doi:10.1109/ISPA.2009.95.

[14] M. Kaid, M. Othman, "Simulated Annealing Approach To Cost-Based Multi-QoS Job Scheduling in Cloud Computing Enviroment," American Journal of Applied Sciences, **11**, 872–877, 2014, doi:10.3844/ajassp.2014.872.877.