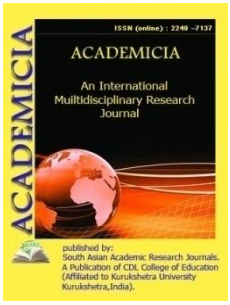
ACADEMICIA
An International Multidisciplinary Research Journal
(Double Blind Refereed & Peer Reviewed Journal)



DOI: 10.5958/2249-7137.2021.02243.6

A REVIEW PAPER ON WORKFLOW SCHEDULING USING CRYPTOGRAPH

Mr Madhav Singh Solanki*

*SOEIT, Sanskriti University,
Mathura, Uttar Pradesh, INDIA
Email id: madhavsolanki.cse@sanskriti.edu.in

ABSTRACT

Cloud computing is a fast-growing technology that allows businesses to use on-demand computer and data services on a daily basis. The most significant contribution is the development of a new genetic algorithm model for workflow planning. One of the major issues here is the scientific planning of large activities in a heterogeneous cloud environment. Other public cloud computing issues are equally significant. These include meeting customer service quality requirements including scalability and reliability, as well as optimizing resource user use rates. Workflow Scheduling is primarily concerned with job assignment in order to achieve the required workload balance while making the greatest use of available resources. Specific workflow planning problems in the cloud computing sector should be addressed by providing various pay-on-demand and cloud-based services that meet the relevant performance criteria and system structure distribution. This paper proposes a novel paradigm for combining cloud-computing resources with local computing components. The finished-time calendar algorithm is at the heart of this system, balancing the performance of the application schedule with the expenses of utilizing cloud resources. The testing and comparisons with other methods revealed the potential benefits of our proposed algorithm.

KEYWORDS: *Cloud computing, Cyber-Physical, Cloud Systems, Dependability, Workflow scheduling,*

1. INTRODUCTION

In recent years, cloud computing has been a growing field of research. A considerable variety of methods to distinct Workflow Scheduling issues have been presented. Due to its direct influence both on the customer and the supplier of services from diverse angles, the cost problem remains of importance (i.e. QoS, system functionality, and system architecture)[1].

Recent developments in computer, communication and control technology offer significant potential to connect physical processes to cyber space and to develop cyberspace systems. For example, automotive, industrial, social, production, smart home, construction, and community-based systems such as CPS applications[2].CPS storage and calculation resources are often restricted, as (1) the size and embedding of cyber components is generally tiny in their surroundings; and (2) some CPS own resources which are suitable, but cost unbeatable for larger usage. Fortunately, cloud computing and IoT technologies would assist the CPS in addressing this problem, because integrating CPS in cloud computing infrastructures may not only enhance cyber-physical device interaction, but also facilitate large-scale data analysis and storages[3]. Cyber-physical cloud systems have therefore garnered considerable interest, both in academia and business, as a viable paradigm.

Cloud computing might in many ways increase system performance. Migration from an internal cybernet to the cloud can, however, lead to a range of security and dependability problems. Due to the complexity of cloud systems, on the one hand, Cyber-Physical Cloud Systems hardware and software are more susceptible to permanent (driven to severe) mistakes and transitory errors (leading to soft errors). On the other hand, owing to the usage of external calculation, communication, and storage resources which cannot be trusted, the security threats to cyber-physical cloud systems are raised.It is therefore very important to create new methodology in cyber-physical cloud systems to solve dependability and security problems[4].

Cloud computing might in many ways increase system performance. Migration from an internal cabernet to the cloud can, however, lead to a range of security and dependability problems. Due to the complexity of cloud systems, on the one hand, Cyber-Physical Cloud Systems hardware and software are more susceptible to permanent (driven to severe) mistakes and transitory errors (leading to soft errors)[5].Cyber-Physical Cloud Systems' security vulnerabilities are raised because of the utilization of external and unsustainable compute, communication, and stocking resources. It is therefore very important to create new methodology in cyber-physical cloud systems to solve dependability and security problems. Workflow has been a successful and widely recognized paradigm for describing data-intensive applications used in cyber-physical cloud systems. There are a number of tasks and dependences of data/control between tasks that create an application for working flow. Directed acyclic graphs that may be used to define work-flow applications with tasks and edges that represent data/control interdependence[6].

Typically, a time limit (i.e., in real time need) to ensure the quality of the service is linked with each Workflow application. The objective of this article is to address dependability and safety issues in order to plan workflows on cyber physical cloud systems in real time. Given that transient defects are more frequent than permanent defects, and that real-world systems are generally needed to meet certain levels of protection[7], we are specifically looking for problems in maximizing soft-error reliability for Cyber-Physical Cloud Systems workflow applications while respecting lifetime constraints and safety requirements.

1.1 Slack-Aware Fault Tolerance:

Slack is inactive as duties for the workflow application are completed early. The illustration of slack is shown in Fig. 1. The four tasks ¨1-µ4 are performed at instance 10 and time is instance 15 of the application deadline. The slack produced is thus 5. This study largely uses the system slack to allocate recoveries to unsuccessful jobs to improve dependability of the system soft error. In order to assign slack, we utilize the First-coming rule to a core that may be shared via all tasks assigned to this core[8]. Moreover, recuperation is only allocated to a job when there is no lower amount of time available to finish a task which is explained by

$$P_i^{rec} = \begin{cases} R_i(F_{max}) \times \sum_{l=1}^{k} P(\mathcal{M}_{i,l}), & if\ K > 0 \\ 0, & otherwise \end{cases} \quad \ldots (1)$$

Where K is the number of schedule modes meeting the slack constraint that

$$wc_i/f_i + RT(\mathcal{M}_{i,l}) \leq \Omega \quad \ldots (2)$$

For Prec it is essential to calculate the $RT(\mathcal{M}_{i,l})$ as well as $P(\mathcal{M}_{i,l})$ retrieval times of each schedule M mode. Note that slack may only be utilised to retrieve a failed task if the slack is available more than the recovery time for the job[9]. The slack cannot be used for a failed job with a longer recovery time. Therefore, we may search in M-mode and determine RT using an iterative method for the failing tasks $RT(\mathcal{M}_{i,l})$. The $P(\mathcal{M}_{i,l})$ probability is defined by the soft-error reliability of the M-mode tasks are given by

$$P(\mathcal{M}_{i,l}) = \prod_{\acute{\tau}\epsilon\, P(\mathcal{M}_{i,l})}(1 - R\varsigma) \times \prod_{\acute{\tau}\epsilon\, P(\mathcal{M}_{i,l})} R\varsigma\,. \quad \ldots (3)$$

1 initialization: $RT(\mathcal{M}_{i,\ell}) = 0$;
2 **for** each failed task $\tilde{\tau}_\zeta$ in $\mathcal{M}_{i,\ell}$ **do**
3 **if** $\frac{wc_\zeta}{f_\zeta} \leq \mho$ **then**
4 $RT(\mathcal{M}_{i,\ell}) = RT(\mathcal{M}_{i,\ell}) + \frac{wc_\zeta}{f_\zeta}$;
5 $\mho = \mho - \frac{wc_\zeta}{f_\zeta}$;
6 **end**
7 **end**

Figure 1: Algorithmic Step By Step Explanation of the Calculation of the Recovery Time RT

1.2 Determine the Priority of Tasks

The suggested workflow planning algorithm that determines jealously the priority of activities in order to improve soft-error system dependability is given. The algorithm is based on the following findings.

- Due to slack limitations, it is impossible to constantly provide better system soft-error reliability for additional jobs with short run time to get recovery. Instead, it might severely affect the dependability of the soft-error system. This is because jobs with high execution times may not slow down to rework and so reduce the soft-mistake dependability of the system significantly.

- The fact that additional jobs are allowed to recover is only useful when the workflow schedule meets in order to increase system dependability.

Theorem 1.

Input: A workflow meeting the conditions in Theorem 1;
use a DAG $G = (T, E)$ to represent this workflow;
if $(\tau_i, \tau_j) \in E$ **then**
  set $p_i > p_j$;
end
else
  if $\frac{wc_i}{F_{max}} > \frac{wc_j}{F_{max}}$ **then**
    set $p_i < p_j$;
  end
end

Figure 2: Algorithm Explaining Step by Step Implementation of Decidingthe Priority of Tasks in the Workflow.

1.3 Lifetime Reliability-aware Frequency Scaling

Tasks are required for increasing system soft error reliability and slackness at the highest frequency scaling in this job. The highest frequency scaling may, however, lead to excessive temperature on the semiconductor and exceed the reliability limitation. This is why it is important to scale the operating frequency scaling of jobs. However, the reliability of soft errors and time performance are also unfavorable. To solve this problem, we offer a heuristic method that reduces the operating frequency scaling of jobs selectively and dynamically for a lifetime[10]. Taking it into accountthat

(1) If a high priority job fails, it is likely that slack is used for recovery and hence has an influence on the soft-error reliability of low priority activities and

(2) Maximum frequency scaling recovery would not have a major effect upon system reliability, because a task's failure likelihood is generally modest, the suggested heuristic method maintains high priority works and restoration frequency scaling while scaling low priority tasks operational frequency.Saving power efficiency is an energy-time ratio when the operation frequency scaling of a job is scaled between l and z. The effectiveness of the job μ_i is represented by scaling its operation frequency scaling from F_l to F_z

$$PE_i(F_l, F_z) = \frac{Pow_i(F_l) - Pow_i(F_z)}{e_i(F_z) - e_i(F_l)} \dots (4)$$

Here in which the energy consumption as well as completion time of operation i at frequencies (F_l, F_z), correspondingly, is $Pow_i(F_l)$ and $e_i(F)$ along with $Pow_i(F)$ but in general mostly with $e_i(F)$.

```
for τ_i ∈ T_Low do
    l_i = L; // l_i is the frequency level of τ_i
end
while MTTF_sys < MTTF_thresh do
    PE_max = 0, τ_select = τ_N; // τ_N is the task with the
    lowest priority
    for τ_i ∈ T_Low do
        if Ʊ > e_i(F_{l_i−1}) − e_i(F_{l_i}) &&
        PE_i(F_{l_i}, F_{l_i−1}) > PE_max then
            PE_max = PE_i(F_{l_i}, F_{l_i−1});
            τ_select = τ_i;
        end
    end
    f(τ_select) = F_{l_i−1}; // f(τ_select) is the operating
    frequency of τ_select
    Ʊ = Ʊ − (e_i(F_{l_i−1}) − e_i(F_{l_i}));
    update MTTF_sys;
end
```

Figure 3: Algorithm Explaining the Procedure of Scale the Frequency Scaling of Low-Priority Tasks

1.4 Simulation Setups:

Algorithm 3 sums together the overall flow of our heuristic frequency scaling. In view of the schedule of low-priority tasks in set T_{Low}, the algorithm initializes the operating frequency of tasks to the greatest level, and then iterates the operating frequency scaling of jobs to a lifespan limit. The algorithm evaluates each job's power-saving performance when its operational frequency is scaled and finds the task with the highest power-saving effectiveness and meets the slow limit (i.e., the real-time constraint). The slack and system must be updated as long as the specified task's operation frequency scaling is scaled.This paper formulatesthe workflow scheduling problem of maximizing soft-error reliability for Cyber-Physical Cloud Systems under system lifetime reliability, security, and real-time constraints. This paper proposes a hybrid approach that enhances the dependability of soft error by stating tasks and allocating retrieval dynamically. Security services maintain security systems and dependability for life is assured by the scaling of frequency scaling of low priority activities.To justify the effectiveness of our hybrid suggested strategy, this paper conducts a range of simulation tests. Our results are compared with those of one baseline and four competing techniques for the suggested algorithm.To solve the aforementioned problem, we propose a dependable workflow scheduling scheme. Specifically, this paper makes following contributions. Results of simulations showed that, compared to basic and competing techniques, proposal can produce better schedules with a less likely failure and more workability.

This research suggests a CPCS hybrid workflow strategy to enhance the reliability of the software under lifespan limitations, security and working flow deadlines. In our system, slack is used to recover failed jobs and therefore increase the dependability of soft error or to improve the safety of the system through use of security services. All workflow jobs share the slack available

in the system.The hybrid system first assesses the priority of the work and, using three important algorithms, allocates the maximum frequency scaling to all jobs and explains their periodical uses. Using the highest frequency scaling, recovery and security services might be slacked. The hybrid system then examines the dependability of the static workflow schedule. If not, the strategy scales the operation ffrequency scaling of low priority activities dynamically, thereby enhancing system dependability without breaking the deadline.

In order to recover the failed tasks the suggested algorithms employ slack and enable all jobs to share the slack in the available system. The method assesses the priority of tasks first to enhance soft-error reliability and allocates the maximum frequency scalingto every job and lastly dynamically assigns recoveries to tasks. Slack may also be utilized to leverage safety services to meet system security needs. Dynamically reducing the operation frequency scaling of low-priority tasks satisfies the lifetime reliability criterion. Extensive real-world benchmarking trials show that the approach presented decreases the likelihood of failure and improves scheduling.The corresponding softeror reliability of these tasks are 0.90, 0.82, 0.74, and 0.67, respectively, in which the derived system soft-error reliability is 0.37. By assigning recovery to tasks, a higher.

2. DISCUSSION

Workflow Management System has become a new cloud computing application in the cloud environment. A number of system models and methodologies have been created for the use of the cloud workflow or Directed Acyclic Graph. A resource supply on demand with the use of the amazon elastic computing cloud for processes.Optimized cost supply for application processes using elastic resources. All of them implemented the supply of resources on the Amazon cloud platform. However, in some directed acyclic graph workflows, just a few study effort has been done to plan the problem. In this study we presented a multiple Reinforcement Learning-based directed acyclic graphscloud workflow scheduling method to resolve multiple directed acyclic graph workflow applications with varied priority submitted at various times in a cloud computing environment. The results of the experiments show that our work schedule is efficient.

Since the objective of this study is to enhance the dependability of the soft error system under lifetime, security and real-time limitations, we will also examine the feasibility of planning four kinds of workflow benchmarks using the suggested method. Of course, among all approaches employed, whatever benchmarks and failure rates are used, the suggested system may always achieve the highest timing feasibility. In comparison with others, the practicality of the suggested arrangement was improved. This improvement is due to our system's safety services and reliability-aware frequency scaling. Similar to Point of delivery observations, the planning of Rand is worse since the two approaches do not use countermeasures to prevent contraventions of restrictions. Moreover, if the rate of failure is larger than that of the lowest failed, feasibilities attained with offered schemes and comparative techniques will not be greater.

3. CONCLUSION

In this paper, we propose several directed acyclic graph scheduling programs for cloud computing based on enhancement learning. In view of the suggested system architecture, we examine the theoretical execution of cloud-based tasks and develop a new work plan using reinforcement learning to improve the time limit for certain cloud-based computing resources.

This article assesses the suggested optimization of work schedules in comprehensive simulations. Based on empirical observations, the proposed work plan may make the best use of cloud resources and load balance in order for the minimum span of time to be achieved with resource limitation.In the cloud computing business, specific tasks of the workflow planning sector should be handled by offering distinct pay-on-demand services and distribution systems structure performance criteria for cloud consumers. This article provides a novel paradigm based on cloud computing resources integration with local computing elements. The essential component of this system is the time algorithm that balances the performance and the cost of using cloud resources in the application schedule. Our proposed algorithms have demonstrated the possible benefits of the tests and the comparisons with other approaches.

REFERENCES

1. F. Wu, Q. Wu, and Y. Tan, "Workflow scheduling in cloud: a survey," *J. Supercomput.*, 2015, doi: 10.1007/s11227-015-1438-4.

2. Sandra V. B. Jardim*, "The Electronic Health Record and its Contribution to Healthcare Information Systems Interoperability," *Procedia Technol.*, 2013.

3. M. Tao, S. Dong, and L. Zhang, "A multi-strategy collaborative prediction model for the runtime of online tasks in computing cluster/grid," *Cluster Comput.*, 2011, doi: 10.1007/s10586-010-0145-4.

4. S. D. Verifier and A. H. Drive, "Simulink ® Verification and Validation ™ Reference," *ReVision*, 2015.

5. S. Committee, *IEEE Standard for Software Verification and Validation IEEE Standard for Software Verification and Validation*. 1998.

6. M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis," *Journal of Network and Computer Applications*. 2016, doi: 10.1016/j.jnca.2016.01.018.

7. E. K. Byun, Y. S. Kee, J. S. Kim, and S. Maeng, "Cost optimized provisioning of elastic resources for application workflows," *Futur. Gener. Comput. Syst.*, 2011, doi: 10.1016/j.future.2011.05.001.

8. M. Bobaru, M. Borges, M. d'Amorim, and C. S. Păsăreanu, *NASA formal methods : third international symposium, NFM 2011, Pasadena, CA, USA, April 18-20, 2011 : proceedings*. 2011.

9. Y. Zhang, "A foundation for the design and analysis of robotic systems and behaviors," 1994.

10. S. Smanchat and K. Viriyapant, "Taxonomies of workflow scheduling problem and techniques in the cloud," *Futur. Gener. Comput. Syst.*, 2015, doi: 10.1016/j.future.2015.04.019.