DRAGONFLY SOFT-COMPUTING APPROACH FOR WORKLOAD SCHEDULING RESOURCE UTILIZATION MAXIMIZATION USING MULTI-CLOUD PLATFORM

Arundhati Nelli, Rashmi Jogdand*

Department of Computer Science and Engineering, KLS Gogte Institute of Technology, Belagavi, Karnataka
India

* Corresponding Author, e-mail: arundhatinelliresearch@gmail.com

Abstract: The execution of the real-time workload in the multi-cloud platform with the Service Level Agreement (SLA) requirements is a challenging task. Existing workload scheduling models have addressed issues related to minimizing execution time, cost, and energy with application reliability prerequisite. However, these models are not efficient in maximizing resource utilization under a multi-cloud platform. In addressing resource utilization issues, this paper presents a Workload Scheduling Resource Utilization Maximization (WS-RUM) technique for the multi-cloud platform. The WS-RUM technique leverages a multi-objective such as energy, processing efficiency, and fault-tolerant offloading mechanism employing a dragonfly soft computing algorithm. The WS-RUM improves resource utilization by minimizing both energy and processing time for real-time workload execution in comparison with existing workload execution.

Key words: Cloud computing, Multi-cloud, Multi-objective parameter, Resource provisioning, Resource Utilization, Soft computing, Workflow scheduling.

1. INTRODUCTION

Cloud services provide various kinds of offers and make the user utilize them and have easy access to the current virtual technology. The Cloud Service Providers (CSP) provides various kinds of resources to the users such as memory, storage, and computing resources for the computation of a given task. Nowadays, as users have adapted to use virtual technology, Cloud Service Providers have enabled the virtualization method so that the resources required by the users can work effectively during the execution of the task. Moreover, the cloud network also utilizes the

Resource Management Technique (RMT) to procure and manage the resources when different resources are allocated to the users. Whenever a task arrives at the cloud network, the Resource Management Technique allocates the given task to a new set of virtual machines or allocates the existing virtual machines for the execution of the task. Furthermore, the Resource Management Technique attempts to increase the performance of the cloud network by considering various QoS prerequisites and various challenges. The Resource Management Technique also provides maximized profit, better fault tolerance, attempts to increase energy efficiency, and also improves load balancing. Most of the recent work done in the field of the cloud face challenges during the execution of real-time workload scheduling. Another challenge is to provide better energy efficiency methods by using Resource Management Techniques for real-time workload scheduling. Further, most of the existing Resource Management techniques use energy optimization techniques in the homogenous cloud network to reduce energy consumption. Moreover, all these methods when used in multi-cloud networks provide poor results and consume more energy for the execution of the task. Hence, in this paper, we present an SLA-based workload scheduling model under a multi-cloud platform that utilizes the Soft Computing Technique and a modified dragonfly algorithm to reduce energy consumption and SLA violation.

2. LITERATURE SURVEY

This section studies recent workload scheduling methods adopting cloud computation platforms. In [1], they have proposed a model which reduces the consumption of energy and also reduces the SLA violations. In [2], they have used the dragonfly algorithm which provides better results when compared with the existing optimization methods. In [3], they have addressed the issues of resource utilization and energy consumption in cloud networks. They have proposed an algorithm based on the classification of the task and the threshold to provide better resources for the execution of the task. The results of this model show that it utilizes less energy, makespan, and also reduces load balancing. In [4], they have presented an algorithm, Deadline and Budget Constrained Heterogeneous Scheduling (DMW-HDBS) to select the task, check the priority of the task so that the resources should be allocated to the most priority task first and then to the least priority tasks. For the experimental results, they have considered the real-time application data to compare their results with the existing models. In [5], they have proposed a method, Endpoint communication contention-aware List Scheduling Heuristic (ELSH) [13], to reduce the makespan during the workflow. This method utilizes the novel task raking method to schedule data communications to the communication resources in addition to scheduling the tasks to computing resources. Also, a rescheduling method has been used to improve the performance of the scheduling algorithm. In [6], they have proposed a model to reduce energy consumption in the cloud server. For this, they have proposed a scheduling framework. In this framework, they have used different strategies for the division of the deadlines and sorting the tasks. In [7], they have proposed a method to minimize the energy consumption of the systems concerning reliability and response time. In [8], they have proposed a multi-objective workflow scheduling method that provides a trade-off solution to reduce the cost and increase the makespan. They have also explored why the makespan and cost fluctuate during the different levels of workflow. They have used five real-world workflows for the experimentation. The results attained show that it increases the makespan and reduces the cost when compared with other existing workflow scheduling methods. In [9], they have proposed two multi-cloud scheduling techniques for the allocation of resources and the execution of the workflow. This technique reduces the cost of execution of the task in the cloud and increases performance. In [10], they have proposed a Scoring and Dynamic Hierarchy-based NSGA-II (Non-dominated Sorting Genetic Algorithm II), called SDHN, which reduces the cost and makespan during the execution of the workflow. They used five real-time workflows for the experimentation and the attained results show that it provides better results. In [12], they have proposed a strategy to provide a cost-efficient and reliable method for the multi-cloud network. The results show that the model has outperformed when compared with the existing methods in terms of reliability and cost. Moreover, all these existing workload scheduling models has not been developed for the multi-cloud network. Hence, in this paper, we present an SLA-based workload scheduling model under a multi-cloud platform that utilizes the Soft Computing Technique and a modified dragonfly algorithm to reduce energy consumption and SLA violation. The research significance of the WS-RUM method is presented below.

- The WS-RUM method maximizes resource utilization by bringing tradeoffs in reducing energy consumption with minimal task processing time.
- The WS-RUM assures Fault-tolerance by leveraging a multi-cloud platform.
- Task offloading is done through the energy minimization function.
- Multi-objective optimization strategy between energy minimization and resource utilization maximization is done by employing an improved dragonfly algorithm.

3. WORKLOAD SCHEDULING RESOURCE UTILIZATION MAXIMIZATION IN MULTI-CLOUD PLATFORM

This section presents workload scheduling leveraging a multi-cloud platform. The workload scheduling is aimed at maximizing system resources with minimal energy consumption. The workload scheduling tradeoff optimization is done through the soft computing technique. The Workload Scheduling Framework under Multi-cloud platform has been given in Figure 1.

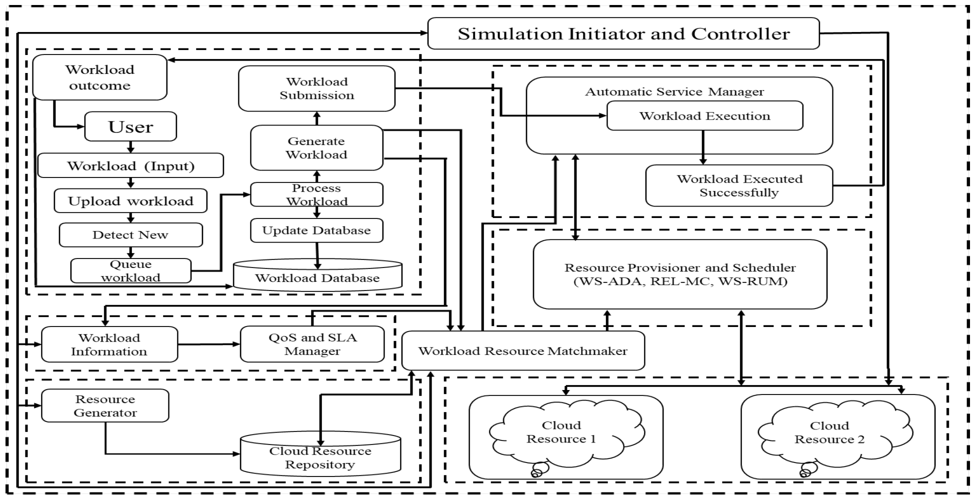


Figure 1. Workload Scheduling Framework under Multi-cloud platform.

3.1. System Model for Workload Scheduling Under Multi-Cloud Platform

The recent multi-cloud networks usually consist of a set of physical machines I, in which each physical machine $I_l \in I$ consists of a set of virtual machines V_l having storage capacity st_l, memory capacity n_l, frequency level w_l, voltage level g_l, maximum energy level $q_l^{\uparrow}$ and bandwidth o_l. In this model, the virtual machines can migrate among the physical machine, and the required resources are shared among the virtual machines. The energy consumed by the physical machine j_m is evaluated using the given Equation (1) [17]

$$J_u = u_m * r_m^{\uparrow} * a_m^v + \frac{(1-u_m) * r_m^{\uparrow}}{\left(h_m^{\uparrow}\right)^3} * (h_m)^3, \tag{1}$$

In Equation (1), u_m is defined to show the least energy dissipation for the executing machines, $r_m^{\uparrow}$ is defined to show the maximum energy level of the physical machine j_m, $a_m^v \in \{0,1\}$ is defined as a Boolean constraint to identify whether the physical machine j_m is working or not working in a given time period v, h_m is used to define the real processing element frequency at a given time period v, $h_m^{\uparrow}$ is defined to indicate the maximal processing element frequency level at a given time period v. From all these notations the total energy cost on the physical machine o concerning the various time period from xt to yt is given using the following equation

$$\mathcal{E} = \sum_{j=1}^{o} \int_{xt}^{yt} \left(t_l * q_l^{\uparrow} * z_l^u + \frac{(1-t_l) * q_l^{\uparrow}}{\left(g_l^{\uparrow}\right)^3} * (g_l)^3 \right) dt, \tag{2}$$

Both the variables a_m^v and h_m in Equation (1) will vary concerning the various time period. Moreover, other variables in the Equation have a fixed value and do not change concerning the time period.

3.2. Workload Scheduling metric modelling

Let the dependency among the various sub-tasks v_l^k and the virtual computing node $w_{m,n}$ on the physical machine j_m be defined by $z_{l,mn}^k$. If the v_l^k is not mapped to the virtual computing node $w_{m,n}$ the value of the $z_{l,mn}^k$ is set to 1. If the v_l^k is mapped to the virtual computing node $w_{m,n}$ the value of the $z_{l,mn}^k$ is set to 0.

$$z_{l,mn}^k = \begin{cases} 0, & if\ v_l^k\ is\ not\ mapped\ to\ w_{m,n} \\ 1, & otherwise. \end{cases}, \quad (3)$$

The data dependency between the various sub-tasks should follow the SLA bounds for the scheduling of the resources

$$hv_{r,mn}^k + vv_{rl}^k \leq tu_{l,mn}^k, \ \ \forall g_{rl}^k \in G_k \quad (4)$$

In Equation (4), $hv_{r,mn}^k$ is used to define the processing time of the sub-tasks v_r^k, vv_{rl}^k is defined to represent the time period required for communication among the various subtasks v_k and v_r. By utilizing the respected virtual machine, the execution of the task can be completed in the given time period y_k . The maximal time period required for the completion of a given task is given using the following Equation

$$hv_k = \max_{v_l^k \in V_k} \{hv_{l,mn}^k\}. \quad (5)$$

During the execution of the task, the workflow schedule should meet the Service Level Agreement required deadline. To ensure the Service Level Agreement requirement, we define it using Equation (6)

$$hv_k \leq f_k, \ \ \forall y_k \in Y. \quad (6)$$

Furthermore, as the virtual machines are limited in the physical machine, the scheduling of the resources along with meeting all the Service Level Agreement parameters which have been defined using Equation (7) and Equation (8) is a difficult and challenging task.

$$h_m^{\uparrow} - \sum_{n=1}^{|W_m|} h_{m,n} \geq 0, \ \ \forall j_m \in J; \quad (7)$$

$$o_m - \sum_{n=1}^{|W_m|} o_{m,n} \geq 0, \ \ \forall j_m \in J. \quad (8)$$

To meet all the constraints of the Service Level Agreement which have been defined in Equations (4), (6), (7), (8), and also to reduce the consumption of energy for the allocation of the resources to run the real-time workload, Equation (9) has been given

$$Min \sum_{m=1}^{p} \int_{xt}^{yt} \left(u_m * r_m^{\uparrow} * a_m^v + \frac{(1-u_m) * r_m^{\uparrow}}{\left(h_m^{\uparrow}\right)^3} * \left(h_m^f\right)^3 \right) dt. \quad (9)$$

In Equation (9), p is used to describe the number of PMs, the starting workload execution time period is given by xt and the ending workload execution time period is given by yt. Further, resource utilization is maximized through the following equation

$$\mathrm{Max}\left(\sum_{k=1}^{o}\sum_{l=1}^{|V_k|}\mathrm{cpu}_l^k * \mathcal{U}_l^k\right)\Big/\left(\sum_{m=1}^{p}\mathrm{h}_m^{\uparrow} * \mathcal{B}_m\right), \tag{10}$$

where o represents the workflow size of Y and $|V_k|$ defines workflow y_k task size, cpu_l^k denote task v_l^k's processing frequency requirement, $\mathcal{U}_l^k$ denotes the overall time it has taken for completing workflow tasks, p describes PMs in the multi-cloud platform, and $\mathcal{B}_m$ defines active hours of PMs j_m's. The performance of the WS-RUM can be increased by using a multi-cloud platform. Therefore, the next section provides the workload task-offloading model.

3.3. Fault-Tolerant Optimization through Multi-Cloud Platform

Consider a multi-cloud network in which each of the servers is connected to other servers. In these circumstances, if a server is not able to meet the Service Level Agreement and Quality of Service requirement then it can offload the computation process to a new server. As there have been only a few research works which have been done for workload scheduling in the multi-cloud network, it a very difficult task to offload it to a new server. Moreover, to offload a task v_k to a given new server u_k it takes more bandwidth x_l which is given using the following Equation

$$x_l = Ylog_2\left(1 + \frac{R_{il}}{\sigma^2}\right) \tag{11}$$

In Equation (11), Y defines the data rate, R defines the energy required for the communication of the servers, σ^2 defines the channel noise, and il defines the communication gain among the cloud networks which is given using the following Equation

$$il = f_l^{-\gamma} \tag{12}$$

In Equation (12), f_l defines the distance between the cloud networks u_l, γ defines the path loss constraint which is dependent on the different obstacles which come in the sight of cloud networks, hence, the latency induced to offload a task v_k to another cloud network u_l is given using the following Equation

$$N_{kl}^t = \frac{\delta_k}{x_l} \tag{13}$$

The overall consumption of energy to offload a task v_k to another cloud network u_l is given using the following Equation

$$\mu_{kl}^t = RN_{kl}^t. \tag{14}$$

3.4. Workload Scheduling offloading metric under Multi-Cloud Platform

If a particular server in the cloud fails to meet the Service Level Agreement requirement, then the given task is offloaded to another server. During the offloading, it may induce some latency which has to be computed. The latency experienced during the execution of the task v_k in one cloud network by considering available virtual machines H_0 is evaluated using the given Equation

$$N_k^n = \frac{\alpha_k}{H_0} \tag{15}$$

In Equation (15), the size of the task is defined using α_k, virtual machines are defined using H_0. To evaluate the energy overhead for the computation process in the new cloud network is given using Equation (16)

$$\mu_k^n = \varphi N_k^n \tag{16}$$

In Equation (16), φ represents the consumption of energy of the virtual machines per second. The proposed offloading mechanism is done by following the minimization function

$$min \sum_{k=1}^{p} \mu_k \tag{17}$$

3.5. Optimization through the soft-computing technique

In [2] showed the dragonfly algorithm is efficient in addressing NP-Deterministic problems for optimizing multi-objective parameters. However, the work is focused on the execution of simple applications under a single cloud platform; further, obtaining optimal solutions takes a significantly longer time. In addressing the aforementioned limitation in this work an improved searching mechanism through Brownian motion considering multi-objective parameters is incorporated into the dragonfly algorithm as shown in Algorithm 1.

```
Algorithm 1. Multi-objective dragonfly algorithm
Step 1. Start
Step 2. Initialize the dragonflies population $X_i(i = 1,2,..,n)$
Step 3. Initialize step vectors $\Delta X_i(i = 1,2,..,n)$
Step 4. Define the maximum number of hyperspheres (segments)
Step 5. Define the archive size
Step 6. while the end condition is not satisfied
Step 8.     Calculate the objective values of all dragonflies
Step 9.     Find the non-dominated solutions
Step 10.    Update the archive with respect to the obtained non-dominated solutions
Step 11.    If the archive is full
Step 12.        Run the archive maintenance mechanism to omit one of the current archive members
Step 13.        Add the new solution to the archive
Step 14.    end if
Step 15.        If any of the new added solutions to the archive is located outside the hyperspheres
Step 16.        Update and re-position all of the hyperspheres to cover the new solution(s)
Step 17.     end if
Step 18.     Select a food source from archive: =SelectFood(archive)
Step 19.     Select an enemy from archive: =SelectEnemy(archive)
Step 20.     Update step vectors
Step 21.     Update position vectors using Brownian motion
Step 22.     Check and correct the new positions based on the boundaries of variables
Step 23. end while
Step 24. Stop
```

The dragonfly algorithm uses Eq. (9), Eq. (10), and Eq. (17) as the multi-objective parameter for obtaining optimal resources in a multi-cloud platform. More details of the dragonfly algorithm can be obtained from [2]. The WS-RUM technique optimized with the dragonfly algorithm aid in minimizing energy and maximizing resource utilization meeting the SLA prerequisite which is shown in the next section.

4. RESULTS AND DISCUSSIONS

This section presents studies of the outcome obtained using WS-RUM over existing workload scheduling techniques such as Workload Scheduling Adaptive Dragonfly Algorithm (WS-ADA) and Reliability Multi-cloud Scheduler (REL-MC). The processing time and energy efficiency parameters are considered for studying the resource utilization performance of different workload scheduling. Complex bioinformatics applications such as SIPHT and Epigenomics are considered for studying the performance of WS-RUM, WS-ADA [2], [14] and REL-MC [12], [13]. The SIPHT and Epigenomics scientific workflow data used for studying models are available in the following link [15].

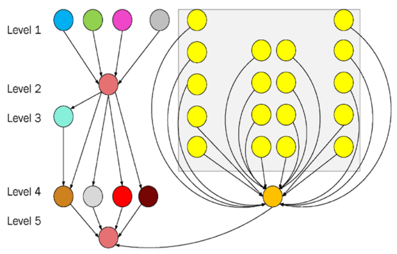


Figure 2. SIPHT Workflow

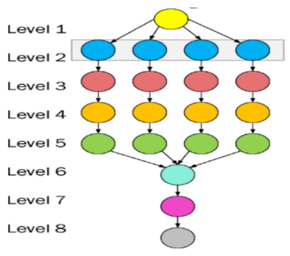
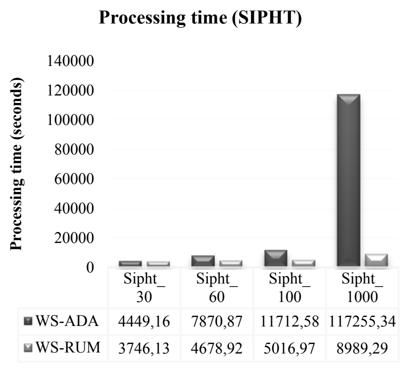


Figure 3. Epigenomics Workflow.

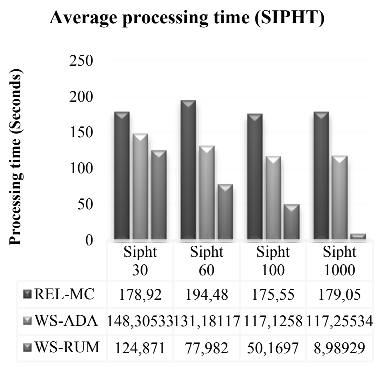
4.1. Case 1

In this section, the SIPHT workload has been executed. The size of the workload varies from 30, 60, 100 and 1000 tasks. The existing system, WS-ADA and WS-RUM have been used for executing the SIPHT tasks which has been shown in Figure 4, Figure 5 and Figure 6. In Figure 4, processing time for executing the SIPHT workload has been shown. The WS-RUM performed well in terms of processing time and attained an improvement of 15.8%, 40.55%, 57.16%, and 92.33% for SIPHT 30, 60, 100 and 1000 tasks respectively. In Figure 5, the average processing time for both the existing WS-ADA and WS-RUM has been evaluated. In this case also, the WS-RUM showed better results when compared with the existing WS-ADA and attained an improvement of 30.20%, 59.90%, 71.42%, 94.98% for SIPHT 30, 60, 100 and 1000 tasks respectively. For proving that our proposed method meets the SLA constraint with minimal violation, we have evaluated the energy consumption of the proposed model which has been shown in Figure 6. The results show that the WS-RUM attained an improvement in energy consumption of 29.1891%, 29.1892%, 29.1892%, and 29.1891% when compared with the WS-ADA for the SIPHT 30, 60, 100 and 1000 tasks respectively.



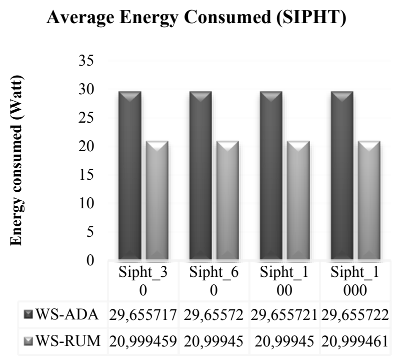
	Sipht_30	Sipht_60	Sipht_100	Sipht_1000
■ WS-ADA	4449,16	7870,87	11712,58	117255,34
■ WS-RUM	3746,13	4678,92	5016,97	8989,29

Figure 4. Processing time for executing SIPHT workload.



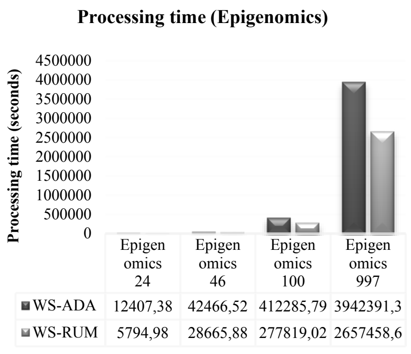
	Sipht 30	Sipht 60	Sipht 100	Sipht 1000
■ REL-MC	178,92	194,48	175,55	179,05
■ WS-ADA	148,30533	131,18117	117,1258	117,25534
■ WS-RUM	124,871	77,982	50,1697	8,98929

Figure 5. Average processing time for executing SIPHT workload.



	Sipht_30	Sipht_60	Sipht_100	Sipht_1000
■ WS-ADA	29,655717	29,65572	29,655721	29,655722
■ WS-RUM	20,999459	20,99945	20,99945	20,999461

Figure 6. Energy consumption for executing SIPHT workload.

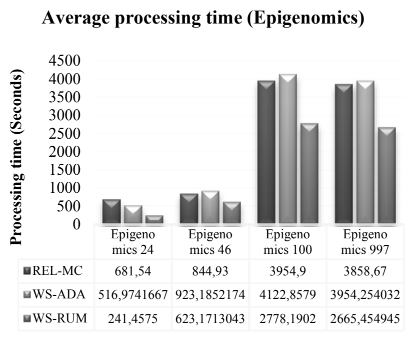


	Epigenomics 24	Epigenomics 46	Epigenomics 100	Epigenomics 997
■ WS-ADA	12407,38	42466,52	412285,79	3942391,3
■ WS-RUM	5794,98	28665,88	277819,02	2657458,6

Figure 7. Processing time for executing Epigenomics workload.

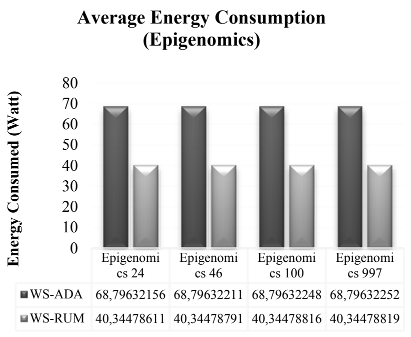
4.2. Case 2

In this section, the Epigenomics workload has been executed. The size of the workload varies from 24, 46, 100, and 1000 tasks. The existing system, WS-ADA and WS-RUM have been used for executing the Epigenomics tasks which has been shown in Figure 7, Figure 8 and Figure 9. In Figure 7, processing time for executing the Epigenomics workload has been shown. The WS-RUM performed well in terms of processing time and attained an improvement of 53.29%, 32.5%, 32.61%, and 32.59% for Epigenomics 24, 46, 100, and 1000 tasks respectively. In Figure 8, the average processing time for both the existing WS-ADA and WS-RUM has been evaluated. In this case also, the WS-RUM showed better results when compared with the existing WS-ADA and attained an improvement of 64.57%, 26.26%, 29.75%, and 30.92% for Epigenomics 24, 46, 100, and 1000 tasks respectively. For proving that our proposed method meets the SLA constraint with minimal violation, we have evaluated the energy consumption of the proposed model which has been shown in Figure 9. The results show that the WS-RUM attained an improvement in energy consumption of 41.36%, 41.356%, 41.35%, and 41.3561% when compared with the WS-ADA for the Epigenomics 24, 46, 100, and 1000 tasks respectively.



	Epigenomics 24	Epigenomics 46	Epigenomics 100	Epigenomics 997
■REL-MC	681,54	844,93	3954,9	3858,67
■WS-ADA	516,9741667	923,1852174	4122,8579	3954,254032
■WS-RUM	241,4575	623,1713043	2778,1902	2665,454945

Figure 8. Average processing time for executing Epigenomics workload.



	Epigenomics 24	Epigenomics 46	Epigenomics 100	Epigenomics 997
■WS-ADA	68,79632156	68,79632211	68,79632248	68,79632252
■WS-RUM	40,34478611	40,34478791	40,34478816	40,34478819

Figure 9. Energy consumption for executing Epigenomics workload.

5. CONCLUSION

The Service-Level-Agreement Workload Scheduling when compared with the existing REL-MC and WS-ADA methods. The WS-RUM method reduces the consumption of energy along with maximizing the utilization of resources and also provides the SLA prerequisite in a multi-cloud environment. In this method, the improved dragonfly optimization algorithm which uses an enhanced searching method for identifying the SLA-aware resources has been proposed for the multi-cloud platform. By using this methodology, the WS-RUM model provides scalability, increases the utilization of resource and enhances the performance of the method for both, large and small tasks of the SIPHT and Epigenomics. However, the major limitation of the proposed work it does not consider memory optimization and reduce last-level cache failures; which is a very important parameter that must be considered in future workload scheduling design. Further, study the performance of the model and how it will perform complex image processing and complex sensor data applications.

REFERENCES

[1] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu, and H. Tenhunen. Energy-awareVMconsolidation in cloud data centers using utilization prediction model. *IEEE Transaction Cloud Computing*, vol. 7, no. 2, 2019, pp. 524536. DOI: 10.1109/TCC.2016.2617374.

[2] Neelima, P., and A. Rama Reddy. An Efficient Load Balancing System Using Adaptive Dragonfly Algorithm in Cloud Computing. *Cluster Computing*, vol. 23, no. 4, 2020, pp. 2891–2899. https://doi.org/10.1007/s10586-020-03054-w.

[3] Malik, Nimra, Muhammad Sardaraz, Muhammad Tahir, Babar Shah, Gohar Ali, and Fernando Moreira. Energy-Efficient Load Balancing Algorithm for Workflow Scheduling in Cloud Data Centers Using Queuing and Thresholds. *Applied Sciences*, vol. 11, no. 13, 2021, pp. 5849. https://doi.org/10.3390/app11135849

[4] Wang, Guan, et al. Dynamic Multiworkflow Deadline and Budget Constrained Scheduling in Heterogeneous Distributed Systems. *IEEE Systems Journal*, vol. 15, no. 4, 2021, pp. 4939–4949. https://doi.org/10.1109/jsyst.2021.3087527.

[5] Wu, Quanwang, et al. Endpoint Communication Contention-Aware Cloud Workflow Scheduling. *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, 2022, pp. 1137–1150., https://doi.org/10.1109/tase.2020.3046673.

[6] Li, Xiaoping, et al. Energy-Aware Cloud Workflow Applications Scheduling with Geo-Distributed Data. *IEEE Transactions on Services Computing*, vol. 15, no. 2, 2022, pp. 891–903., https://doi.org/10.1109/tsc.2020.2965106.

[7] Hu, Biao, et al. Energy-Minimized Scheduling of Real-Time Parallel Workflows on Heterogeneous Distributed Computing Systems. *IEEE Transactions on Services Computing*, vol. 15, no. 5, 2022, pp. 2766–2779. https://doi.org/10.1109/tsc.2021.3054754.

[8] T. -P. Pham and T. Fahringer. Evolutionary Multi-Objective Workflow Scheduling for Volatile Resources in the Cloud. *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, 2022, pp. 1780-1791. doi: 10.1109/TCC.2020.2993250.

[9] Barika, Mutaz, et al. Scheduling Algorithms for Efficient Execution of Stream Workflow Applications in Multicloud Environments. *IEEE Transactions on Services Computing*, vol. 15, no. 2, 2022, pp. 860–875. https://doi.org/10.1109/tsc.2019.2963382.

[10] Li, Huifang, et al. Scoring and Dynamic Hierarchy-Based NSGA-II for Multiobjective Workflow Scheduling in the Cloud. *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, 2022, pp. 982–993. https://doi.org/10.1109/tase.2021.3054501.

[11] Song, An, et al. Scheduling Workflows with Composite Tasks: A Nested Particle Swarm Optimization Approach. *IEEE Transactions on Services Computing*, vol. 15, no. 2, 2022, pp. 1074–1088., https://doi.org/10.1109/tsc.2020.2975774.

[12] Tang, Xiaoyong. Reliability-Aware Cost-Efficient Scientific Workflows Scheduling Strategy on Multi-Cloud Systems. *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, 2022, pp. 2909–2919., https://doi.org/10.1109/tcc.2021.3057422.

[13] Faragardi, Hamid Reza, et al. GRP-Heft: A Budget-Constrained Resource Provisioning Scheme for Workflow Scheduling in Iaas Clouds. *IEEE Transactions on*

Parallel and Distributed Systems, vol. 31, no. 6, 2020, pp. 1239–1254., https://doi.org/10.1109/tpds.2019.2961098.

[14] Konjaang, J. Kok, and Lina Xu. Multi-Objective Workflow Optimization Strategy (MOWOS) for Cloud Computing. Journal of Cloud Computing, vol. 10, no. 1, 2021, https://doi.org/10.1186/s13677-020-00219-1.

[15] Scientific workflow data, available at: "*https://pegasus.isi.edu/workflow_gallery/gallery/*", last accessed Dec 16th 2022.

[16] Murgesh V, et.al. Service Level Agreement Aware Energy Optimized Scheduling Algorithm for Cloud Computing Environment. *International Journal on Information Technologies and Security*, vol.14, No. 1, 2022, pp. 17-28.

[17] Nelli, Jogdand, R. SLA-WS: SLA-based workload scheduling technique in multi-cloud platform. *Journal of Ambient Intelligence Human Computing*, 2022. https://doi.org/10.1007/s12652-021-03666-z

Information about the authors:

Arundhati Nelli – Arundhati Nelli has completed B.Tech in Information Science and MTech in Computer Science & Engineering from Visvesvaraya Technological University, Belagavi. Currently she is Pursuing PhD in computer Science and engineering from VTU. Her main research interest includes Cloud computing, Machine Learning, Soft computing, Information Retrieval and Database Management Systems.

Rashmi Jogdand – Rashmi Jogdand has completed B.Tech in Computer Science from KUD, Master's degree from BITS, Pilani and Ph.D from Graphic Era University, Dehradun. At Present working as Professor in Department of Computer Science and Engineering, KLS Gogte Institute of Technology, Belagavi. Her areas of interest include Cloud Computing, Information Data Retrieval and Storage Management.

Manuscript received on 28 December 2022
Revised paper received on 29 January 2023