Computers, Materials & Continua
DOI: 10.32604/cmc.2023.033300
Article

Tech Science Press

TRS Scheduling for Improved QoS Performance in Cloud System

G. John Samuel Babu[1] and M. Baskar[2,*]

[1]Department of Computer Science and Engineering, School of Computing, College of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur, Chengalpattu, Tamilnadu, 603203, India
[2]Department of Computing Technologies, School of Computing, College of Engineering andTechnology, SRM Institute of Science and Technology, Kattankulathur, Chengalpattu, Tamilnadu, 603203, India
*Corresponding Author: M. Baskar. Email: baashkarinfo@gmail.com
Received: 13 June 2022; Accepted: 14 December 2022

Abstract: Numerous methods are analysed in detail to improve task scheduling and data security performance in the cloud environment. The methods involve scheduling according to the factors like makespan, waiting time, cost, deadline, and popularity. However, the methods are inappropriate for achieving higher scheduling performance. Regarding data security, existing methods use various encryption schemes but introduce significant service interruption. This article sketches a practical Real-time Application Centric TRS (Throughput-Resource utilization–Success) Scheduling with Data Security (RATRSDS) model by considering all these issues in task scheduling and data security. The method identifies the required resource and their claim time by receiving the service requests. Further, for the list of resources as services, the method computes throughput support (Thrs) according to the number of statements executed and the complete statements of the service. Similarly, the method computes Resource utilization support (Ruts) according to the idle time on any duty cycle and total servicing time. Also, the method computes the value of Success support (Sus) according to the number of completions for the number of allocations. The method estimates the TRS score (Throughput Resource utilization Success) for different resources using all these support measures. According to the value of the TRS score, the services are ranked and scheduled. On the other side, based on the requirement of service requests, the method computes Requirement Support (RS). The selection of service is performed and allocated. Similarly, choosing the route according to the Route Support Measure (RSM) enforced route security. Finally, data security has gets implemented with a service-based encryption technique. The RATRSDS scheme has claimed higher performance in data security and scheduling.

Keywords: Cloud; task scheduling; TRS; quality of service; RSM; route security; data security; SDE; RATRSDS

1 Introduction

The increasing volume of organizational and customer data challenges the storage capacity of different organizations. The organization maintains various data, including business and personal information about its employees and customers. Also, they need more time to be ready and capable of purchasing huge volume data centres due to their cost. The cloud platform is deployed to support the organizations. The cloud environment opens the gate for different scale organisations in such a way as to keep their data secure as well as provide access to their users at the exact and least cost.

The cloud can be classified as public, private, or hybrid, each dedicated to providing different services. Various service providers offer the same sets of services. However, the services differ with different constraints, and they differ in Quality of Service (QoS) performance. Scheduling and selecting services according to various parameters is necessary to achieve effective results. Several scheduling approaches consider different parameters like throughput, makespan, latency, and so on [1].

Further, the services can be ranked in several ways according to their throughput, latency, completion ratio, success rate, and popularity. Yet, the entire system's performance depends on how the services are selected and ranked. It is necessary to rank the services based on multiple parameters.

On the other side, the services are accessed through several devices where the requested data is passed through many hops in the network. Internet of Things (IoT) devices will be located along the route. But the devices cannot be trusted, and it is necessary to choose an efficient route to improve the performance of QoS. The routing is performed in several ways, considering parameters like hop count, traffic, energy, mobility, and direction. Still, choosing an efficient route with higher security is necessary. The security of any route can be measured by considering a variety of parameters. In another way, data security has been enforced with blockchain technology. By incorporating the blockchain methodology, data security in the cloud can be improved. This research is geared towards maximising the cloud's security in data and routing with high-quality scheduling of services. Based on the above analogy, scheduling performance can be improved by enforcing a strategic scheduling scheme that considers the maximum number of parameters and features. Instead of considering just makespan and popularity or completion time, it is necessary to consider various features like throughput, resource utilization, and success rate. Further, performance achievement dramatically depends on the way services are ranked. So, the selection of services should be performed according to the support requirements.

On the other side, data security should be enforced to improve the entire QoS of the environment because the presence of malicious nodes in the network or route would spoil service access and degrade service performance. Finally, route security should be enforced rigidly, supporting performance development. By considering all these, an efficient real-time application-centric TRS schedule and service-based data encryption are discussed in this paper. The method estimates Requirement Support (RS) values for the task submitted toward identifying the resource and measures Route Support Measure (RSM) towards selecting the route. The RSM value is measured according to the presence of IoT devices and other route features. By incorporating the proposed scheduling model, the performance of the cloud environment would be improved by including the available IoT devices in the scheduling, which would support the development of QoS performance. The QoS performance of cloud scheduling can be improved not just by choosing effective resources and sequences but also depending on the route selected for data transmission. This claim says the proposed model produces higher QoS performance. The TRS approach increases performance on various QoS factors and improves security by adapting RSM-based route selection. Also, the adaptation of service-based data encryption improves data security.The article is organised to present different details. Section 1 offers a detailed introduction to cloud scheduling and the methods and issues involved in scheduling tasks in the cloud. Section 2 summarises the literature survey with different approaches to the problem. Section 3 details the working methodology of the proposed scheduling approach, and Section 4 details the performance analysis results and discusses different parameters in detail. Finally, Section 5 describes the conclusion of the approach and future work.

2 Literature Survey

Several approaches are discussed in the literature around task scheduling and securing data in the cloud. This part analyses and presents different approaches related to the problem.

An efficient workflow management scheme with scheduling to support big data applications is presented in [2], which works on two tiers by generating several combinations of tasks and schedule accordingly. An energy and resource efficient scheduling (ERES) model over a virtual environment is presented in [3], which targets minimizing energy consumption and maximizing resource utilization. Also, the model considers the deadline constraints by reducing the makespan. A detailed review of multi-objective task scheduling is presented in [4], which considers different approaches and classifies the schemes based on the optimization adapted. A low-power task scheduling scheme is introduced in [5], which uses a loss comparison rule to select specific clusters of resources with minimum loss and power usage. A dynamic scheme for resource allocation and task scheduling to handle power management is discussed in [6]. The method uses a resource prediction scheme and a resource table updating scheme to select efficient resources in scheduling with the least amount of power.

Towards efficient service selection, a context-based service similarity measure-based approach is presented in [7], which computes the similarity according to the input and output parameters in measuring the correlation. According to the correlation measure, service composition is performed. A DoC2Vec-based functionality clustering with Deep-FM score prediction is presented in [8], which conducts functionality clustering with Doc2Vec and uses Deep-FM to extract the relationship among the services. A composite description-based reconstructed profile for web service recommendation is presented in [9], which dynamically constructs the service profiles with different historical information to perform service selection and recommendation.

A location-aware service composition and mixture ranking scheme is presented in [10], which analyses the QoE (Quality of Experience) features of different services in ranking them and uses GLLB (a global most significant number of service requests first, local best-fit service candidate first) in service ranking and scheduling. A trust-based routing scheme with blockchain security and reinforcement learning is presented in [11], which uses blockchain and reinforcement learning techniques for data security in WSNs (wireless sensor networks). The blockchain is used to identify the nodes involved in routing, gets routing information, and supports avoiding tampering. By adopting the blockchain in routing, the nodes can quickly identify the next hop to be forwarded and cannot be tempered. In [12], a secure transaction for bitcoin is presented, which performs routing according to the properties of the data. The method involves securing the data according to the assumption of data and considering different properties of the data. The route selection is performed by considering data properties to improve data security.

The challenges in scheduling tasks in cloud environments have been well studied, analyzing different issues in task scheduling [13]. Further, a detailed analysis of scheduling nomenclature is presented in [14], with a lot of experimental research. A heuristic approach that combines analytical hierarchy and bandwidth-aware scheduling is presented, including the longest-exempted preemption scheduling approaches [15].

A load-based scheduling algorithm named LIF is presented that identifies the integrated loads of different tasks and performs scheduling based on the features [16]. Similarly, a scheduling scheme is introduced where the resources are minimal according to the market-oriented scheme, and the tasks are distributed among the clients [17]. Furthermore, the participants utilize the same market approach in service allocation. It identifies the order in which combinations of services can be allocated to perform scheduling [18].

A market-oriented strategy in scheduling the resources considers the cost of tasks in accessing the resources. The selection of the scheduling sequence is made according to the task [19]. A log-based approach is presented that considers the previous logs of scheduling and, based on the performance of resources for the task assigned, performs the scheduling [20].

A randomised online stack algorithm (ROSA) based on booking calculation is presented in [21], which uses a certain proportion of resources at the lower bound and finds resources with the lowest cost in resource allocation. A policy-based load-balancing approach is presented that uses the ant colony to perform optimization to reduce the makespan [22]. A hybrid heuristic and genetic-based task scheduling algorithm are presented in [23], which combines a genetic algorithm and a list-based scheme for scheduling in cloud environments. A user-centric adaptive fuzzy clustering-based resource scheduling scheme is presented in [24], which works based on the trust sensitivity being used in splitting the resources around the user and task-based scheduling. For each of them, it would compute the trust measures based on which the resource selection would be performed.

An online reinforcement learning-based scheduling is presented in [25] to handle the lack of knowledge of resources prior to foresighted task allocation. A modified ACO (ant colony optimization) technique is proposed to improve the performance of scheduling the VMs [26]. An energy-based task allocation scheme and resource allocation strategy over the Green-Sched model is presented, where the method considers the deadline, cost, and so on, in scheduling [27]. On the other side, a genetic algorithm-based meta-heuristic scheduling strategy is presented to reduce the cost of workflow. Also, the method considers the deadline at a reduced price [28]. In [29], a novel approach, WOA (Whale Optimization Algorithm), is presented to improve the scheduling performance.

All the methods discussed above suffer from poor scheduling performance and introduce poor throughput performance. This article considers all these and presents a TRS scheduling model for the QoS development of the entire environment.

2.1 *Problem Statement*

The problem of task scheduling in the cloud has been well studied, and various approaches are analyzed for their performance in different metrics. The existing methods consider only limited features and factors in selecting resources for any task that challenges the methods'z ability to achieve higher performance metrics. The scheduling performance can be improved by considering multiple and maximum metrics and features.

2.2 *Motivation*

Around the problem identified from the literature, the author is motivated to improve the performance in scheduling. The selection of resources can be enhanced by adapting the scheduling algorithm to throughput, resource utilization, and success rate. By enforcing the scheduling algorithm according to the above features and computing requirement support for the tasks, the performance of scheduling can be improved.

3 Real-Time Applications Centric TRS Scheduling and Data Security (RACTRSDS) Model

The proposed RACTRSDS model maintains a set of services for various tasks. Also, the traces of different services and resources are maintained in the cloud. By receiving the user request, the method extracts the features of the service claimed and estimates the requirement support (RS) value. Similarly, the method estimates the TRS value for different sets of services to choose the best one. The scheduling is performed according to the TRS values. Again, route security is enforced by measuring RSM (Route Support Measure), and data security is enforced using service-based data encryption schemes. The detailed approach is discussed in this section.

The schematic diagram of the proposed RATRSDS model and its functional components are sketched in Fig. 1. Each is discussed briefly in this part.

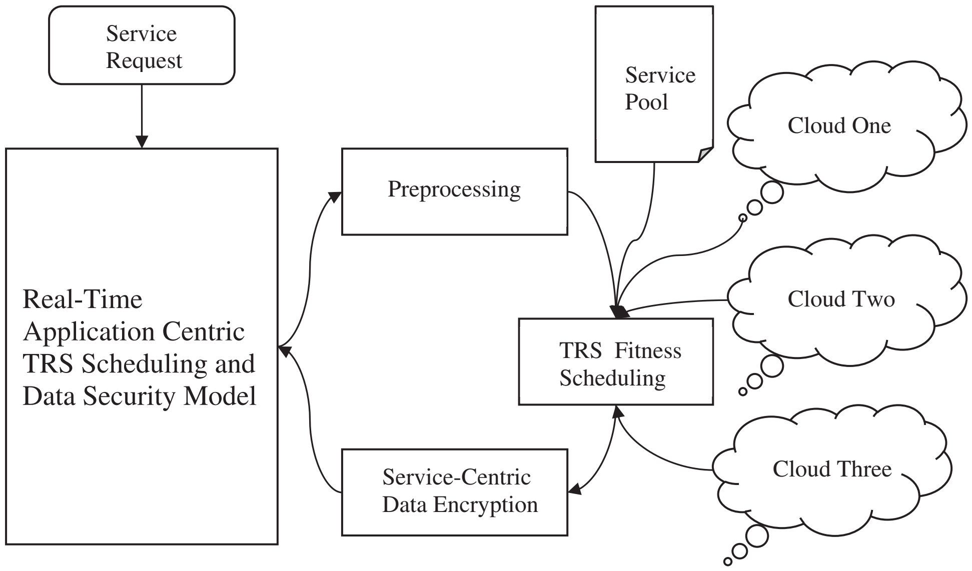
Service Request
Service Pool
Cloud One
Preprocessing
Real-Time Application Centric TRS Scheduling and Data Security Model
Cloud Two
TRS Fitness Scheduling
Cloud Three
Service-Centric Data Encryption

Figure 1: Architecture of proposed RATRSDS model

3.1 Preprocessing

The task set submitted has been read at this stage, identifying the tasks submitted. Any task would have several subtasks, and such subtasks are identified. For any task, there should be a specific resource to be used to complete the task. The preprocessing algorithm identifies the tasks and resources required. Also, from the task submitted, the method would find the hold time of each task and its resources. Such features are extracted and converted into feature vectors to perform task scheduling.

The task set submitted to the preprocessing algorithm is presented in Table 1, where, for each task, the method would find the list of resources required and their hold time towards scheduling. According to the rows of Table 1, each task has several subtasks, and to perform such tasks, we would need several resources for a specific time. For example, task no. 1 requires resources R1, R2, and R3 for a certain amount of time, as mentioned, to complete the task. According to the details of Table 1, the TRS scheduling algorithm would select a sequence of tasks to be executed to achieve higher QoS performance.

Table 1: Task set submitted

Task no.	Resource	Time claims
1	R1,R2,R4	4,7,9
2	R2,R3,R5	2,5,9
3	R1,R2,R3,R4	3,6,8,9
4	R3,R3,R5	6,2,8
5	R1,R2,R3	4,7,9

Algorithm 1: Preprocessing Algorithm

```
Given: Task Set Ts
Obtain Feature Set Fs.
Start
    Read Task set Ts.
    Identify set of all tasks Tasks $= \sum_{i=1}^{size(Ts)} Tasks \in Ts(i)$    (1)
    For each task Ti
        Identify the list of sub-task STs $= \sum \mathrm{SubTasks} \in \mathrm{Ti}$    (2)
        Level = 1;
        For each sub-task STi
            Identify the resource required Rr $= \overset{size(STs)}{\underset{i=1}{STs(i).Resource}}$    (3)
            Identify the resource claim time Rct $= \overset{size(STs)}{\underset{i=1}{STs(i).Resource.HoldTime}}$    (4)
            Generate feature vector Fv = {level, Sti,Rr,Rct}
            Add to feature set Fs $= \sum(\mathrm{Fv} \in \mathrm{Fs}) \cup \mathrm{Fv}$
        End
    End
Stop
```

The preprocessing algorithm reads the task set using Eq. (1), and the list of subtasks is identified using Eq. (2) for each task. Further, using Eq. (3), the resource required is identified. Also, the features like the hold time required by the task are identified using Eq. (4) and extracted to produce a feature vector. A generated feature vector is added to the set to support TRS scheduling. According to the content of Table 1, the method finds the set of tasks and their required resources with the claim time to generate the feature vector to support efficient scheduling.

3.2 TRS Fitness Scheduling

The proposed scheduling algorithm works according to the TRS support values. The method reads the feature vectors and service taxonomies to perform. From the taxonomy, the method identifies the set of services according to the service requests identified at each stage. First, the feature vectors are split according to the stage of the task. The method identifies the service type for each task and finds the set of services available from the taxonomy. Now, according to service and resource identification, the method computes support values based on throughput, resource utilization, and success rate measures. Similarly, the method computes the value of required support for each resource identified. Finally, using the value of TRS support values, the method computes the value of the Scheduling Fitness Score (SFS). The services are ranked according to the SFS, and based on that, scheduling is performed.

Algorithm 2: TRS Scheduling Algorithm

```
Given: Feature Vector Set Fvs, Service Taxonomy ST, Trace Tr
Obtain: Null
Start
    Read Feature Vector Set Fvs, Service Taxonomy ST, Trace Tr.
    Find the number of stages Ns = $\underset{i=1}{\overset{size(Fvs)}{Max}}(Fvs(i,..j).Stage)$
    At each stage s
        For each task T
            Service set ss = $\sum_{i=1}^{size(ST)} ST(i).service.Type == Task$
            For each service sr
                Compute throughput support Thrs.
                $$Thrs = \frac{\sum_{i=1}^{size(Tr)} Tr(i).service == sr\&\&Tr(i).execution\ statements}{\sum_{i=1}^{size(Tr)} Tr(i).service == sr} \quad (5)$$
                Compute resource utilization support Ruts.
            $$Ruts = \frac{\sum_{i=1}^{size(Tr)} Tr(i).service == sr\&\&Tr(i).HoldTime}{\sum_{i=1}^{size(Tr)} Tr(i).service == sr\&\&Tr(i).completionTime} \quad (6)$$
                Compute success support Sus.
            $$Sus = \frac{\sum_{i=1}^{size(Tr)} Tr(i).service == sr\&\&Tr(i).state == Complete}{\sum_{i=1}^{size(Tr)} Tr(i).service == sr} \quad (7)$$
            Compute scheduling fitness score SFS.
            $$SFS = \frac{(Thrs \times Sus)}{Ruts} \quad (8)$$
          End
            Rank services according to SFS.
            Choose the most valued service and assign the task.
    End
Stop
```

The TRS fitness-based scheduling scheme discussed above represents how the scheduling is performed according to the value of TRS measures, the number of services accessed and the number of statements executed using Eq. (5). Similarly, the value of ruts is measured according to the completion time and hold time of different services using Eq. (6). The value of Sus is calculated according to the number of completions and total service access using Eq. (7). Finally, the value of SFS is measured by computing throughput support, resource utilisation support (Ruts), and success support (Sus). The value of SFS is used to find a task that supports the enhancement of QoS.

3.3 Service-Centric Secure Data Encryption

The security of the service data is enforced in two ways. The proposed approach enforces data security by choosing the data encryption scheme according to the type of service. Similarly, the selection of the transmission route is performed according to the value of the Route Support Measure (RSM). To perform this, the method first identifies the set of routes available to access the service. For each route identified, the method estimates the RSM according to the number of transmissions made earlier, the number of them made exactly, and the number missed and retransmitted. According to the value of RSM, the method selects an optimal route, the data is encrypted with the selected service and the key has been transmitted through the chosen route. The selection of keys is made from the set of keys dedicated to the service. As the model maintains the key set and scheme sets for different services dedicatedly, the method selects a distinct key from the key set and scheme set, which will be sent to the receiver in a secure channel as an index. The receiver gets the index values and performs a lookup on the set maintained by it, which has been given to the user at the start of the session. By mapping the key and scheme, the user can get the original data.

The service-centric secure data encryption scheme presented above represents how the data transmission is performed through the secure route and how the data security is enforced with the service-based scheme. The RSM value of any route is measured according to the successful transmission of packets through the route and the number of retransmissions that occur on the route considered. According to the value of RSM, the most effective route is selected, and data transmission is performed.

4 Results and Discussion

The Real-time Application Centric TRS Scheduling and Data Security (RACTRSDS) scheme is hard-coded, and its efficacy is measured on various parameters. In this section, the obtained results are compared with those of different other approaches.

The details of services and resources considered for the performance evaluation have been measured and displayed in Table 2.

Table 2: Details of evaluation

Parameter	Value
Tool used	Cloud sim
Number of resources	100
Number of instances	5
Number of tasks	200
Number of services	300

Algorithm:

```
Given: Data D, Scheme set Scs, and Key set Keys, service taxonomy ST
Obtain: Encrypted data ED
Start
        Read Data D, a scheme set scs, and key set keys.
        Find service type SType = $\sum_{i=1}^{size(ST)} ST(i).Service == s \&\& ST(i).Type$
        Encryption scheme Es = $\sum_{i=1}^{size(Scs)} Rand(Scs, Scs(i).Type == SType)$
        Encryption key Ek = $\sum_{i=1}^{size(keys)} Rand(keys, keys(i).Type == SType)$
        For each feature f
                ED = Encrypt(Data.f,Es,Ek)
        End
        Identify the list of routes Rl to the service point.
        For each router
                Compute route support measure RSM.
                $\text{Rsm} = \frac{\sum_{i=1}^{size(Tr)} Tr(i).Route == r\&\&Tr(i).state == success}{\sum_{i=1}^{size(Tr)} Tr(i).Route == r} \times \frac{\sum_{i=1}^{size(Tr)} Tr(i).State == retransmit}{\sum_{i=1}^{size(Tr)} Tr(i).Route == r}$   (9)
        End
        Choose the route with maximum RSM and transmit the chain through the route.
Stop
```

The efficiency of various approaches to scheduling is evaluated and compared with the result of the proposed RACTRSDS approach. The RACTRSDS scheme has claimed higher performance in all categories of the number of tasks than other schemes, as displayed in Table 3.

Table 3: Analysis of scheduling

Scheduling Performance % *vs.* No of tasks			
	50 Tasks	100 Tasks	200 Tasks
EREC	76	71	69
EAT	83	77	73
PESVMC	87	91	91
RACTRSDS	91	95	98

The efficacy of scheduling achieved by different approaches is evaluated and compared in Fig. 2, where the RACTRSDS scheme has achieved a higher rate than other schemes.

The efficiency of energy utilisation reduction is measured for different approaches and presented in Table 4. The proposed RACTRSDS approach has claimed higher results than other approaches.

Fig. 3 denotes the comparative result on energy efficiency produced by different methods with varying numbers of tasks. In each case, the RACTRSDS scheme has achieved higher performance.

The resource utilisation performance produced by various approaches is counted in the presence of different numbers of tasks. In each case, RACTRSDS produced higher performance, as in Table 5.

Performance in resource utilisation is evaluated with different-sized task buckets. The RACTRSDS scheme introduced higher performance in each bucket and is plotted in Fig. 4.

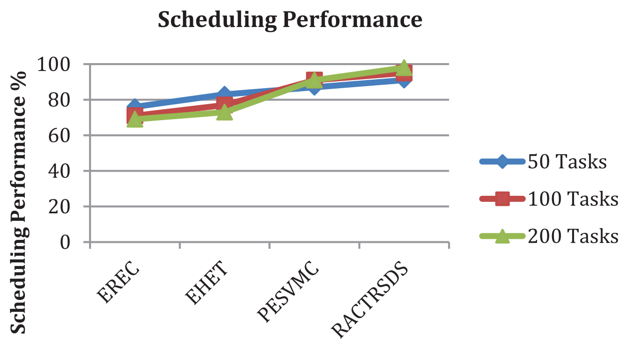

Figure 2: Analysis of scheduling performance

Table 4: Analysis of energy efficiency

Energy efficiency % *vs* No of tasks			
	50 Tasks	100 Tasks	200 Tasks
EREC	74	71	68
EAT	81	74	72
PESVMC	86	89	92
RACTRSDS	91	94	97

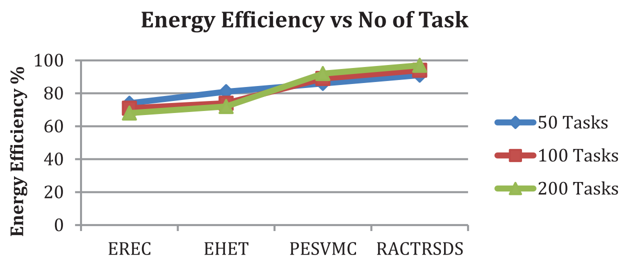

Figure 3: Analysis of energy efficiency

Table 5: Analysis of resource utilization

Resource utilization performance vs No of task			
	50 Tasks	100 Tasks	200 Tasks
EREC	71	74	78
EAT	74	79	83
PESVMC	81	87	92
RACTRSDS	87	92	97

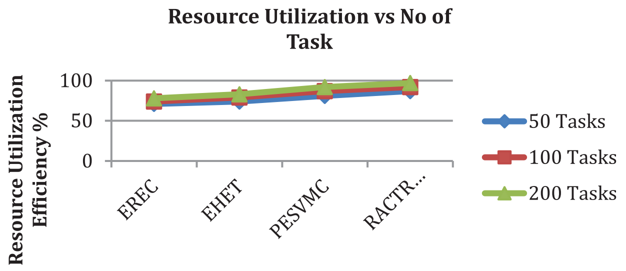

Figure 4: Analysis of resource utilization

The value of time complexity introduced by various approaches for different bucket sizes of tasks is measured and plotted in Table 6, where RACTRSDS has claimed less time complexity in all the buckets.

Table 6: Performance in time complexity

Time complexity *vs*. No of task			
	50 Tasks	100 Tasks	200 Tasks
EREC	32	46	59
EAT	28	34	46
PESVMC	24	28	32
RACTRSDS	21	25	29

Time complexity in scheduling different bucket sizes of tasks is measured for various approaches where the RACTRSDS scheme has introduced less time complexity, as plotted in Fig. 5.

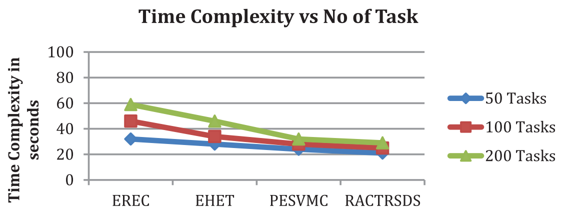

Figure 5: Analysis of time complexity

5 Conclusion

This paper presents an efficient real-time application-centric TRS scheduling system with data security (RACTRSDS). The method receives the user request and finds the task set given. The method applies preprocessing and feature extraction with the task set to generate a feature vector set. With the feature vector set obtained, the method estimates different support measures on throughput, resource utilization, success rate, etc. With the support measures computed, the method computes the value of the scheduling fitness score for different services. According to the value of SFS, the method performs the task scheduling over the services. Also, the secure route is identified with the support of the secure route measure and applies service-centric data encryption to improve data security. The method improves performance in energy efficiency by up to 97%, data security by up to 97%, and scheduling by up to 98%.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Meena, M. Kumar and M. Vardhan, "Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint," *IEEE Access*, vol. 4, pp. 5065–5082, 2016.
[2] Y. Hu and H. Wang, "Intelligent cloud workflow management and scheduling method for big data applications, Springer," *Springer, Journal of Cloud Computing*, vol. 9, no. 39, pp. 1–13, 2020.
[3] N. Garg, "Energy and resource efficient workflow scheduling in a virtualized cloud environment," *Springer, Cluster Computing*, vol. 24, no. 2, pp. 767–797, 2020.
[4] M. Hosseinzadeh and M. Y. Ghafour, "Multi-objective task and workflow scheduling approaches in cloud computing: A comprehensive review," *Springer, Journal of Grid Computing*, vol. 18, pp. 327–356, 2020.
[5] B. Liang, "A low-power task scheduling algorithm for heterogeneous cloud computing," *Springer, Journal of Super Computing*, vol. 76, no. 9, pp. 7290–7314, 2020.
[6] J. Praveenchandar, "Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 4147–4159, 2020.
[7] F. Zhang, Q. Zeng, H. Duan and C. Liu, "Composition context-based web services similarity measure," *IEEE Access*, vol. 7, pp. 65195–65206, 2019.
[8] X. Zhang, "Web service recommendation via combining doc2vec-based functionality clustering and deepfm-based score prediction," in *Proc. IEEE Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, Melbourne, VIC, Australia, pp. 509–516, 2018.
[9] Y. Zhong, "Web service recommendation with reconstructed profile from mash up descriptions," *IEEE Automobile Science and Engineering (ASE)*, vol. 15, no. 2, pp. 468–478, 2018.
[10] J. Lu, Guanfeng Liu, Keshou Wu and Wenjiang Qin, "Location-aware web service composition based on the mixture rank of web services and web service requests," *Hindawi, Analysis and Applications of Location-Aware Big Complex Network Data*, vol. 2019, pp. 16, 2019.
[11] J. Yang, S. He and Y. Xu, "A trusted routing scheme using blockchain and reinforcement learning for wireless sensor networks, MDPI," *Sensors*, vol. 19, pp. 1–19, 2019.
[12] B. Christian, "Bitcoin as a transaction ledger: A composable treatment," in *Proc. 36th Annual Int. Cryptology Conf. Advances in Cryptology*, Barbara, USA, pp. 324–356, 2017.

[13] Q. Zhang, L. Cheng and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Springer Journal of Internet Services and Application*, vol. 1, no. 1, pp. 7–18, 2010.

[14] V. Reddy, B. Rao, L. Reddy and P. Kiran, "Research issues in cloud computing," *Global Journal of Computer Science and Technology*, vol. 11, no. 11, pp. 59–64, 2011.

[15] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *Springer, Journal of Cloud Computing*, vol. 7, no. 1, pp. 1–16, 2018.

[16] W. Tianet, "A dynamic scheduling algorithm for cloud data centers considering multi-dimensional resources," *Journal of Information & Computational Science*, vol. 10, no. 12, pp. 3925–3937, 2013.

[17] C. Chien, P. H. Chang and V. Soo, "Market-oriented multiple resource scheduling in grid computing environments," in *Proc. 19th Int. Conf. on Advanced Information Networking and Applications, AINA 2005*, Taipei, Taiwan, vol. 1, pp. 867–872, 2005.

[18] I. Fujiwara, K. Aida and I. Ono, "Applying double-sided combinational auctions to resource allocation in cloud computing," in *Proc. IEEE/IPSJ Int. Symp. on Applications and the Internet*, Seoul, Korea, pp. 7–14, 2010.

[19] Z. Yang, C. Yin and Y. Liu, "A cost-based resource scheduling paradigm in cloud computing," in *Proc. 12th Int. Conf. on Parallel and Distributed Computing Applications and Technologies*, London, United Kingdom, pp. 417–422, 2011.

[20] E. Madhukar and T. Ragunathan, "Efficient scheduling algorithm for cloud," *Elsevier, Procedia Computer Science*, vol. 50, pp. 353–356, 2015.

[21] R. Zhang, K. Wu, M. Li and J. Wang, "Online resource scheduling under concave pricing for cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 1131–1145, 2016.

[22] M. A. Tawfeek, A. El-Sisi, A. E. Keshk and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," in *Proc. 8th Int. Conf. on Computer Engineering & Systems (ICCES)*, Cairo, Egypt, pp. 64–69, 2013.

[23] M. Sulaiman, "An evolutionary computing-based efficient hybrid task scheduling approach for heterogeneous computing environment," *Springer, Journal of Grid Computing*, vol. 19, no. 11, 2021.

[24] J. Yu, "Qualitative simulation algorithm for resource scheduling in enterprise management cloud mode," *Hindawi, Cognitive Computing Solutions for Complexity Problems in Computational Social Systems*, vol. 2021, no. 5, pp. 1–12, 2021.

[25] S. Mostafavi and V. Hakami, "A stochastic approximation approach for foresighted task scheduling in cloud computingpproximation approach for foresighted task scheduling in cloud computing," *Springer, Wireless Personal Communication*, vol. 114, pp. 901–925, 2020.

[26] E. H. Houssein, "Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends," *Elsevier, Swarm and Evolutionary Computation*, vol. 62, 2021. https://doi.org/10.1016/j.swevo.2021.100841 .

[27] T. Kaur and I. Chana, "GreenSched: An intelligent energy aware scheduling for deadline-and-budget constrained cloud tasks," *Elsevier, SimulationModeling Practice and Theory*, vol. 82, no. 5, pp. 55–83, 2018.

[28] X. Chen, "A woa-based optimization approach for task scheduling in cloud computing systems," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3117–3128, 2020.

[29] M. A. Dulebenets, "A delayed start parallel evolutionary algorithm for just-in-time truck scheduling at a cross-docking facility," *International Journal of Production Economics*, vol. 212, no. 2, pp. 236–258, 2019.