Ingénierie des Systèmes d'Information
Vol. 26, No. 2, April, 2021, pp. 225-230
Journal homepage: http://iieta.org/journals/isi

Secured Data Transmission with Integrated Fault Reduction Scheduling in Cloud Computing

Annabathula Phani Sheetal[1*], Giddaluru Lalitha[1], Arepalli Peda Gopi[2], Vejendla Lakshman Narayana[3]

[1] Department of CSE, School of Technology, GITAM (Deemed to be University), Hyderabad, Telangana 502329, India
[2] Department of CSE, Vignan's Nirula Institute of Technology & Science for Women, Peda Palakaluru, Guntur 522009, India
[3] Department of IT, Vignan's Nirula Institute of Technology & Science for Women, Peda Palakaluru, Guntur 522009, India

Corresponding Author Email: sheetal.klu@gmail.com

https://doi.org/10.18280/isi.260209

Received: 11 November 2020
Accepted: 5 March 2021

Keywords:
cloud computing, failures, fault tolerant, critical tasks, scheduling, fault recovery, overhead

ABSTRACT

Cloud computing offers end users a scalable and cost-effective way to access multi-platform data. While the Cloud Storage features endorse it, resource loss is also likely. A fault-tolerant mechanism is therefore required to achieve uninterrupted cloud service performances. The two widely used defect-tolerant mechanisms are task relocation and replication. But the replication approach leads to enormous overhead storage and computing as the number of tasks gradually increases. When a large number of defects occur, it creates more overhead storage and time complexity depending on task criticalities. An Integrated Fault Reduction Scheduling (IFRS) cloud computing model is used to resolve these problems. The probability of failure of a VM is calculated by finding the previous failures and active executions in this model. Then a fault-related adaptive recovery timer is retained, modified depending on the fault type. Experimental findings showed that IFRS reached 67% lower storage costs and 24% less response time when comparing with the current technique for sensitive tasks.

1. INTRODUCTION

Cloud computing in today's technical world is one of the new fields. It is used to connect services via the internet, such as hardware, infrastructure and applications on demand. The key goal is to provide a large amount of services and/or on-demand resources [1]. There is a large size of heterogeneous tools, a broad user base and various kinds of application tasks in the cloud computing environment. They handle a wide range of user activities and huge data [2]. In financial and scientific applications, cloud computing is very useful. In the face of setbacks, the assignment of resources for jobs with a tight time frame is challenging [3].

Fault tolerance is also a critical aspect of cloud computing in addition to missed task deadlines. In the event of failure, this guarantees prompt and efficient performance of real-time work. Since backup significantly increases overhead storage, alternative methods are required that produce high resource utilization [4]. Although the cloud features are appealing and the uninterrupted performance of cloud services requires an inaccurate tolerance mechanism [5].

Any of the flaws come from inside and outside defects. The two normal fault tolerant mechanisms in cloud computing are task reallocation and replication. In the re-allocation of tasks, after a mistake occurs, a task is re-submitted. This process improves the system's use of resources. However, it can extend the time for response, which does not meet the task deadlines [6]. Requests for resources in data centers can fail, as power is increased during allocation of resources [7].

It is mainly aimed at reducing latency and service overheads and enhancing the efficiency and capacity of the cloud. The approach relies on the categorization of devices that can request in three groups by service category. These classes are timely, time-tolerant and central. A preprepared executive list of devices maps any time-sensitive request to one or more edge devices. One or more devices on the cloud or cloud core may be allocated per time tolerant request. Key applications are delegated to the cloud core resources. The proposed approach selects the most suitable defect tolerant technique from the duplication, check pointing and re-submission techniques for each application to achieve defect tolerance whereas the majority of current methods consider only one technique.

Adapting to the fault-resolving mechanism involves all the necessary steps for strict system reliability and heartiness as well. The probability of the system to terminate internally or fail could be reduced to a large extend using fault tolerance considering the fact that it produces better results in dynamic improvements in the execution time of the system, failure recovery and an economically lower budget cost. Meanwhile unmistakably cloud environment has diversified in short term run which was able to develop a circulated application which was severely developed to improve the layers of virtualization where a designing application where able to reflect effective adaptability. The framework highlights the work which consists of accessibility and reliability of course as mentioned which is depicted by the frameworks QoS.

2. LITERATURE SURVEY

Project replication approach is used in the programming of the fault tolerant workflow [8]. You can concurrently perform several copies of tasks here. But as the number of tasks grows, it leads to large overhead storage and computing. In addition, the technique to find the exact number of duplicate copies was not presented.

The constructive fault tolerance approach [9] has taken into account strength, memory and other network parameters to improve resource reliability. After estimating the reliability of each VM based on the success rate of performance, VM is selected for work scheduling with high reliability.

Lu et al. [1] proposes two algorithms for scheduling. Energy efficient defect-free scheduling is carried out in the first algorithm, and the second algorithm reserves slack times required for recovery of defects.

They proposed a selective reflected task method in the fault tolerance algorithm (Fault Tolerance Algorithm) [10], considering the balance between the parallelity and the topology of the application. By minimizing the mapping and calculation price, the fault tolerance for DAG-based applications is solved.

For critical tasks or tasks with permanent failures, the dynamic fault-tolerant working flow schedule [11], the spatial re-execution (SRE) system is used while the temporal re-execution scheme (TRE) is used for non-critical tasks with temporary failures. However, the SRE system can incur more overhead storage if the number of faults for critical tasks is high. Also, when the amount of defects in non-critical tasks is high, time complexity is increased [12]. In addition, the chances of VM backup failure will not be reviewed.

Reactive defect tolerant methods are responding to a malfunction. Here, after beginning the application, reactions are implemented. The cloud status is hereby constantly monitored for failure detection. Replication, inspection and re-submission may be used for reactions. The requested submission could lead to a breach of the contract for service level because of delays in the fulfilment of the requests. Checkpoints could lead to delays and storage resources in exhaust cloud. Although replication is the most comprehensive resource reactive technique, it is the most common. The reason is that the replication method decreases lateness to almost nil. Amazon Ec2 uses the auto scaling community to simultaneously build and operate several copies of the same software.

Cloud knowledge of fault tolerance relates to the mechanism to allow a technique to withstand faults in the framework of the task execution [13]. One of the advantages of improving cloud tolerances is failure prevention, cure, cost savings and higher performance measurement [14]. Once several cloud tasks are performed on several VMs, then several of the servers crash, which means that there is a failure and that the defect tolerance mechanism is usually taken [15]. There are a server instance failure and thus a failure of the tasks. Sometimes one case of failure stimulates another [16]. These factors may include hardware breakdown, device failures, network partitions, power loss and unforeseen software results. There are currently several fault tolerance mechanisms in line with the cloud scheduling [17]. These include: reprocessing, healing, submission, replication, software rejuvenation, masking and migration, but most of these are susceptible to high overheads and often lead to local trappings. In this section we examine some associated literature that used intelligent optimization technologies in the cloud computing setting to solve the dynamic task scheduling problem [18].

3. PROPOSED MODEL

In this research work, a cloud computing model is proposed for Integrated Fault Reduction Scheduling (IFRS). The block diagram of the CHFTS model is shown in Figure 1.

The probability of failure in each VM is calculated in the IFRS model by finding previous failures and successful executions. The expected runtime (EET) is calculated for each Ti mission. Then tasks are classified as important tasks with short deadlines. Some VMs are then designated as defect-tolerant VMs based on the FoP. A collection of main and backup VMs is assigned for each mission. If a fault occurs in any VM, a fault recovery timer is started, depending on the task type. If it is impossible to recover the failure in the same period, the failed VM will automatically be notified and from that moment the output resumes.

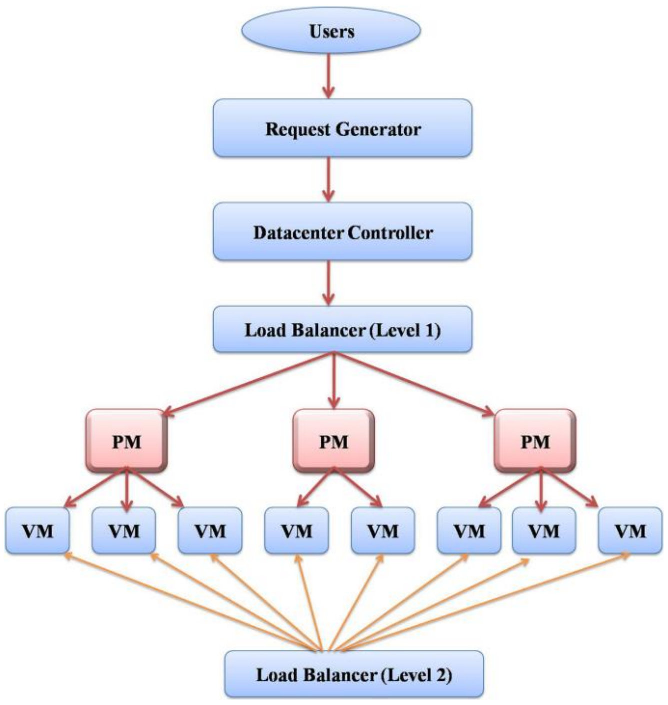

Figure 1. Block diagram of IFRS model

3.1 System model

Each server in Cloud system can be divided into a set of heterogeneous virtual machines (VMs), by means of virtualization. Hence a VM is considered as a fundamental element in a cloud system.

Each data center c owns a set Vc of virtual machines, which can be represented as:

$$Vc=\{vc1,vc2,\ldots,vcn\}.$$

The machine model is shown in Figure 2. In particular, VM(k) is defined by the P(k) and the cost per hour C processing power (k). The virtualization principle allows users to access an infinite number of VMs in the cloud computing platform. In addition, the bandwidth between the VMs should be homogeneous, all VMs are placed in one cloud data centre.

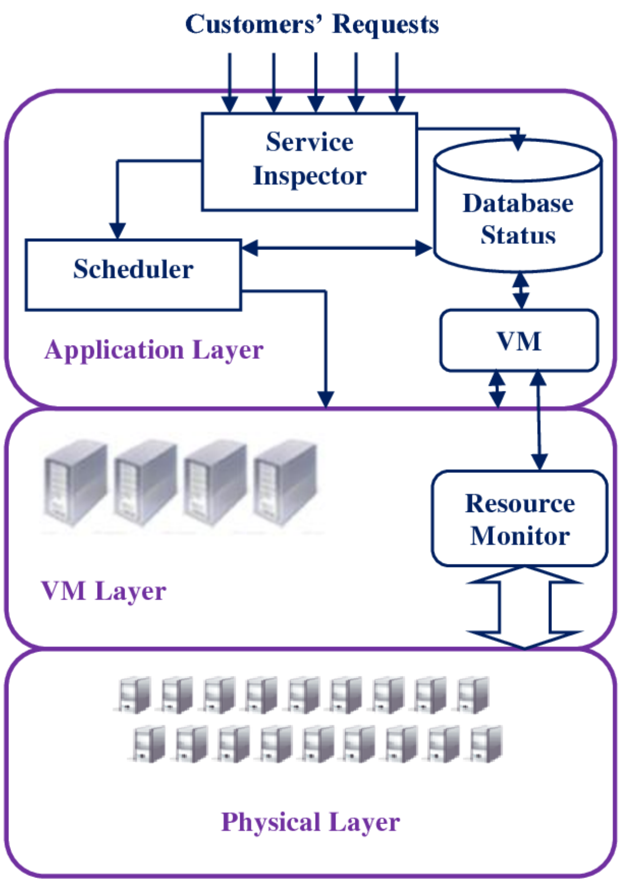

Figure 2. System model

3.2 VM management

The main reason for the calculation is to improve the performance of the cloud by limiting as well the time used by the cloud application as the effect of dissatisfaction. The calculation is based on choosing VMs and Cloud Analysts that have the most timely completion of customer application energy. It also relies on the replication framework for producing multiple duplicates of identical applications that are running concurrently on different VMs and Cloud Analyst. The segments include the fast, the VM server, the replication manager and the cloud VMs and Cloud Analyst. In addition to the QoS requirements, users or customers present their applications or jobs to the cloud through the cloud entry. In the representative line are the occupations integrated. The expert receives a vocation from the representative line in addition to its requisite QoS. The VM Monitoring Server will be approached at that time by a review of relevant VMs and cloud analysts to perform the operation. The server responds to the client application with the VMs which, in addition to usual completion times, will play out the application. Each VM shall be sorted by the official according to the completion date of the application. The first VM is chosen as the basic VM for carrying out activities in the arranged cycle, the VM with the time limit.

Where an internal or external problem makes a task incomplete, it can be called a task failure. During performance of scientific work flows in cloud-based environments, failures can occur. Behind these defects there are various explanations. The key explanation for the task breakdown is the VM failure. The other explanations for failures include insufficient resources, overloading of resources, delayed execution, etc.

During execution, there are two instances of failure in the Cloud. The first is a lifelong mistake and the other a temporary mistake. The failures can be restored in temporary defects in a short time period while the faults can be rectified in permanent defects only after a failed part has been fixed or replaced. A fault detection system widely used is a failure signal or acceptance measure.

As a result of job delays, workflows are more time consuming and the service level contract (SLA) is being infringed. Reexecution is one of the least costly fault tolerant techniques widely used to increase workflow reliability. The re-execution of space with other resources (SREs) and temporary re-implementation with the same resources after recovery of faults can be achieved in two ways: (TRE).

3.3 Estimation of expected end time

Let P(k) be the processing capacity of VM_k, k=1,2...K.

Let $S(t_i)$ and $W(t_i)$ be the size and workload of the input task t_i.

Task execution is started if and only if the input data is received from all its previous tasks $pre(t_i)$.

Then start time of t_i is represented by Equation

$$T_{start}(t_i) = max_{pri}\,(T_{start}) + min_{pri}\,(T_{end})$$

The scheduling time is calculated as

$$T_{sch}(ti) = CS(i)/\,Th$$

where, CS is the cloud server and Th is the Threshold limit.

The fault levels are identified as

$$T_{exec}(ti, VM(k)) = W(ti) \big/ P(k)$$

Thus, the end time of task t_i is given by

$$T_{end}(ti) = T_{start}(ti) + T_{trans}(ti) + T_{exec}(ti, VM(k))$$

The proposed algorithm indicates the fault recognition and reduction for improving the accuracy and performance levels of the model.

3.4 Algorithm Integrated Fault Reduction Scheduling (IFRS)

```
Step-1:  Input: jobs, money, and tasks.
Step-2:  Output: the length of the process and the use of
         resources.
Step-3:  Set the resource list [Ressource number] to start
Step-4:  Start the task list [Task Number] Not empty
         though job-list
Step-5:  For each task 'Ji' do
Step-6:  VMi random nodes will be selected uniformly,
         do the task 'J' from the front of the job-list.
Step-7:  Applications are given for
Step-8:  VM0(J0), VM1(J1),)... VMn(Jn) and Jobs.
Step-9:  For all workers set rep=0.
Step-10: Set rep=0.
Step-11: Present Assign Work Ji with a lower load to the
         Vmi
Step-12: If the tie for the least loaded node is present then
         Assign Job to a selected random end if the
```

```
           default list is initialized [Resource Faulty No.]
Step-13:   The Job 'Ji' assigned is taken from the defective
           resource.
Step-14:   Rep=1 set;
Step-15:   Set Send jobs back to queue if idle available
           resources
Step-16:   Return Job to the less loaded node Reallocate
           the Work J and divide and assign VM to the
           randomly selected node if there is a tie for the
           least loaded node.
Step-17:   Wait before idle load-less resources are
           provided.
Step-18:   End if
Step-19:   End
Step-20:   End for
```

The primary step of the programmer is to divide the task into smaller tasks, when the user submits the work. The Task Controller shall have a list of tasks from the beginning of each task that is prepared for execution. The allocator will keep a resource list, which will randomly pick the 'd' grid nodes from the available nodes. Scheduler transmits each grid node with a query message. Thus, Scheduler receives the actual load data of each of the d nodes. Finally, the planner assigns the less loaded assignment to d nodes that have been randomly selected. There may be cases in which resources cannot perform the role assigned to them during the task assignment process. This could be because the resources allocated are already full of tasks. The error occurred in such a case prevents the programmer from properly programming the job. The task allocator thus determines the defective resources and the error handler provides a list of defective resources [19]. Scheduler reviews in the resource list for idle resource. The fault list resource is used and the fault tolerance is carried out when this specific task is allocated with the least load to the available resource [20]. This helps to plan properly and all tasks with minimal runtime can be completed successfully. Tolerate errors during the assigning of tasks and re-schedule the tasks to various resources to assist the system to cope with heavy loads to produce very significant results.

4. EXPERIMENTAL RESULTS

The proposed model is implemented in Cloudsim for identification of faults and reducing the faults. The proposed IFRS model exhibits better performance when compared to traditional models. The proposed model considers 34675 tasks for performing scheduling and then the parameters are evaluated and depicted in this section. The fault identification rate of proposed and existing methods are indicated in Figure 3.

Five cloud users with five brokers and two data centers are built in the first scenario. There are three hosts in the first data center, while there are two hosts in the second. Even 10 VMs are generated by Xen as a Virtual Machine Managers (VMM) on Linux OS using the Time – Shared Policy, each with 512 BM, an image size 10 000 BM, and one CPU each. With a stock size of 1,000,000 and a bandwidth of 10,000, the Host Memory is 2048 MB. In addition, the number of tasks submitted (cloudlets) is 10 to 100, each of 800,000 in duration and 600 in file size.

The number of missed tasks for scheduling in the proposed model is very less when compared to the existing models. The Figure 4 indicates the Number of missed tasks in proposed and existing models.

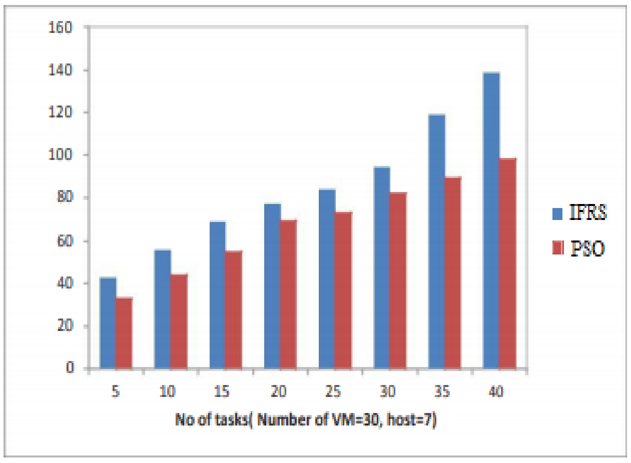

Figure 3. Facult Identification rate

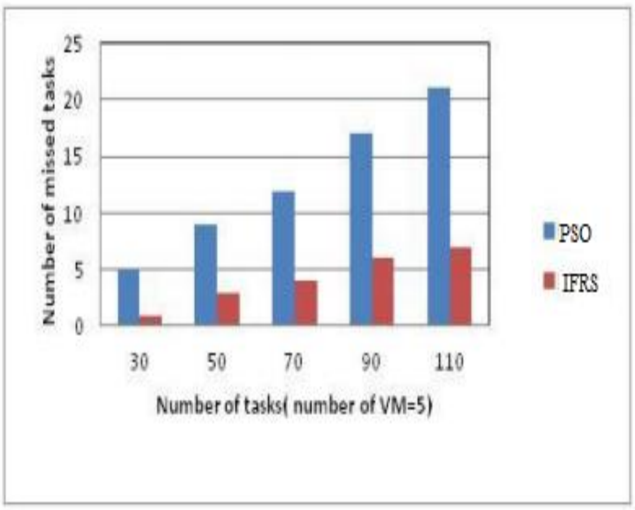

Figure 4. Missed tasks levels

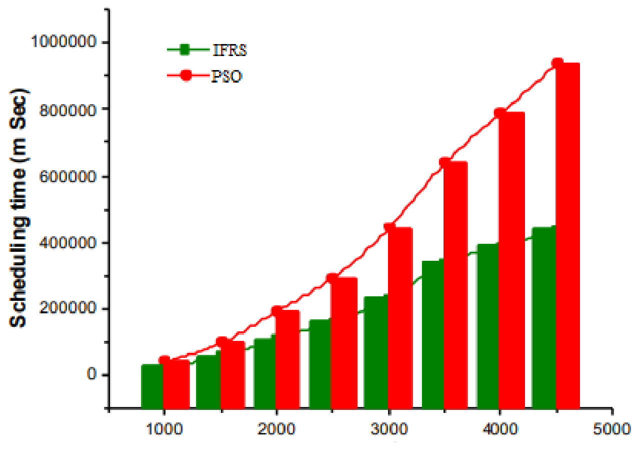

Figure 5. Scheduling time levels

The scheduling time in the proposed model is more accurate and in less time the IFRS model complete the scheduling. The Figure 5 represents the scheduling time levels of the proposed and traditional methods.

The VM requests are handled effectively and the requestions need to be completed in time. The proposed model and traditional models request completed status is indicated in Figure 6.

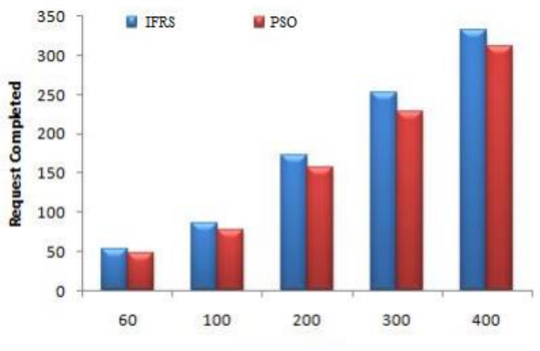

Figure 6. Requests completed levels

The failed request levels of the proposed and traditional models are indicated in Figure 7. The failed requests of the proposed models are high when compared to traditional methods.

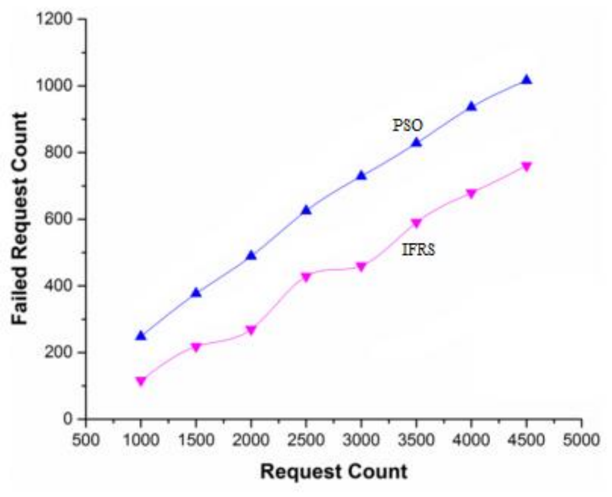

Figure 7. Failed requests levels

5. CONCLUSION

Cloud-based application services need a large amount of data processing. Since network bandwidth is a limited resource, task scheduling algorithms based on data locality is crucial for reducing the job completion time. This article proposes a cloud computing IFRS model for integrated fault reduction planning. In this model, by identifying previous failures and active executions the fault probabilities (FoP) of each VM are calculated. The expected time of success (EET) is calculated for each mission. Then tasks are classified as important tasks with short deadlines. The principal outcomes of the proposed approach are to minimize latencies and service overheads and to enhance cloud trustworthiness and capability. In addition, the approach gives the cloud services greater availability. These results would boost the credibility and increase the benefit from the provider's perspective. For the consumer, the results indicate that the customer's response times and costs are of a distinctive level of service. During a fault which is modified depending upon the form of faults, an adaptive fault recovery period is retained. For critical and non-critical tasks, experiments are performed by different failure rates. The results of experiments showed that the IFRS models achieve a 43% savings in storage costs, as opposed to current technology, and a 13% response delay for essential tasks. Future work concentrates on grouping activities according to their types and demands in order to further reduce CPU use and battery power. We intend to broaden our research to include other forms of transmission failures encountered. Moreover, we intend to concentrate on maximising electricity use.

REFERENCES

[1] Lu, K., Yahyapour, R., Wieder, P., Yaqub, E., Abdullah, M., Schloer, B., Kotsokalis, C. (2016). Fault-tolerant service level agreement lifecycle management in clouds using actor system. Future Gener Comput Syst., 54: 247-259. https://doi.org/10.1016/j.future.2015.03.016

[2] Moon, Y.H., Youn, C.H. (2015). Multihybrid job scheduling for fault-tolerant distributed computing in policy-constrained resource networks. Comput Netw., 82: 81-95. https://doi.org/10.1016/j.comnet.2015.02.030

[3] He, J., Dong, M., Ota, K., Fan, M., Wang, G. (2014). NetSecCC: A scalable and fault-tolerant architecture for cloud computing security. Peer-to-Peer Netw Appl., 9(1): 67-81. https://doi.org/10.1007/s12083-014-0314-y

[4] Nawi, N.M., Khan, A., Rehman, M.Z., Chiroma, H., Herawan, T. (2015). Weight optimization in recurrent neural networks with hybrid metaheuristic Cuckoo search techniques for data classification. Math Probl Eng., 2015: 868375. https://doi.org/10.1155/2015/868375

[5] Mills, B., Znati, T., Melhem, R. (2014). Shadow computing: an energy-aware fault tolerant computing model. In: 2014 International Conference on Computing, Networking and Communications (ICNC), pp. 73-77. https://doi.org/10.1109/iccnc.2014.6785308

[6] Kashan, H.A. (2009). League championship algorithm: a new algorithm for numerical function optimization. In: International Conference of Soft Computing and Pattern Recognition, 2009. SOCPAR'09, pp. 43-48. https://doi.org/10.1109/socpar.2009.21

[7] Kashan, H.A., Karimi, B. (2012). A new algorithm for constrained optimization inspired by the sport league championships. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1-8. https://doi.org/10.1109/cec.2010.5586364

[8] Abdulhamid, S.M., Latiff, M.S.A., Ismaila, I. (2014). Tasks scheduling technique using league championship algorithm for makespan minimization in IAAS cloud. ARPN J Eng Appl Sci., 9(12): 2528-2533.

[9] Abdulhamid, S.M., Latiff, M.S.A., Madni, S.H.H., Oluwafemi, O. (2015). A survey of league championship algorithm: prospects and challenges. Indian Jo Sci Technol., 8(S3): 101-110. https://doi.org/10.17485/ijst/2015/v8is3/60476

[10] Yang, Y.G., Tian, J., Lei, H., Zhou, Y.H., Shi, W.M. (2016). Novel quantum image encryption using one-dimensional quantum cellular automata. Inf Sci., 345: 257-270. https://doi.org/10.1016/j.ins.2016.01.078

[11] Dondi, R., El-Mabrouk, N., Swenson, K.M. (2014). Gene tree correction for reconciliation and species tree

inference: Complexity and algorithms. J Discrete Algorithms, 25: 51-65. https://doi.org/10.1016/j.jda.2013.06.001

[12] Abdulhamid, S.M., Latiff, M.S.A., Bashir, M.B. (2014). On-demand grid provisioning using cloud infrastructures and related virtualization tools: A survey and taxonomy. Int J Adv Stud Comput Sci Eng IJASCSE, 3(1): 49-59.

[13] Kushwah, V.S., Goyal, S.K., Narwariya, P. (2014). A survey on various fault tolerant approaches for cloud environment during load balancing. Int J Comput Netw Wirel Mobile Commun., 4(6): 25-34.

[14] Yang, W., Zhang, C., Shao, Y., Shi, Y., Li, H., Khan, M., Hussain, F., Khan, I., Cui, L.J., He, H. (2014). A hybrid particle swarm optimization algorithm for service selection problem in the cloud. Int J Grid Distrib Comput 7(4): 1-10.

[15] Hussin, M., Lee, Y.C., Zomaya, A.Y. (2010). Dynamic job-clustering with different computing priorities for computational resource allocation. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE Computer Society, pp. 589-590. https://doi.org/10.1109/ccgrid.2010.119

[16] Vidhate, D., Patil, A., Guleria, D. (2010). Dynamic cluster resource allocations for jobs with known memory demands. In: Proceedings of the International Conference and Workshop on Emerging Trends in Technology, ACM, pp. 64-69. https://doi.org/10.1145/1741906.1741918

[17] Abd. Latiff, M.S., Shafie, M., Abdulhamid, S.M., Bashir, M.B. (2014). Scheduling techniques in on-demand grid as a service cloud: A review. Journal of Theoretical and Applied Information Technology, 63(1): 10-19.

[18] Abdullahi, M., Ngadi, M.A. (2016). Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Gener Comput Syst., 56: 640-650. https://doi.org/10.1016/j.future.2015.08.006

[19] Madni, S.H.H., Latiff, M.S.A., Coulibaly, Y. (2016). An appraisal of meta-heuristic resource allocation techniques for IaaS cloud. Indian J Sci Technol., 9(4): 1-14. https://doi.org/10.17485/ijst/2016/v9i4/80561

[20] Chiroma, H., Shuib, N.L.M., Muaz, S.A., Abubakar, A.I., Ila, L.B., Maitama, J.Z. (2015). A review of the applications of bio-inspired flower pollination algorithm. Procedia Comput Sci., 62: 435-441. https://doi.org/10.1016/j.procs.2015.08.438