International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Adaptive Fault-Tolerance During Job Scheduling in Cloud Services Based on Swarm Intelligence and Apache Spark

Sadoon Hussain Abdullah*[1], Al-Hakam Ayad[2], Nadia M. Mohammed[3], Redhwan M. A. Saad[4,5]

Submitted: 12/11/2022 **Accepted:** 13/02/2023

Abstract: Cloud services are generally seen as a promising technique developed to achieve the highest computation service needs. However, such high-performing level of computing services can lead to the highest level of failure rates owing to a wide range of components and host servers which are filled with intensive job scheduling problems. Therefore, failure which occurs in one component or sub-system will lead to the unavailability of the computation services for the system. In this research, we suggest a new effective model called adapting fault-tolerant model (AFTM) which aimed to examine the optimization of job scheduling problem in computing infrastructure based on Particle Swarm Optimization (PSO), Apache Sparka and Ant Colony Optimization (ACO). The proposed approach covers the implementation and analysis of virtualizations with the job task selection to health monitoring for fault diagnoses based on Apache Spark. The objective is to find the cost trade-off between the allocated memory and CPU execution time required by virtualization services created by the end-users. The evaluation of the empirical performance of the proposed approach results outperforms PSO algorithms and traditional Genetic Algorithm (GA) in terms of the allocated memory and the time of CPU execution.

Keywords: *Fault-tolerance, Job Scheduling, Cloud services, Apache Spark.*

1. Introduction

Cloud computing is a model which allows the end-users to access the distributed virtual machines and shares available resources in infrastructure as a service (IaaS) as well as job scheduling techniques [1], tolerance techniques, and virtualization technologies. There are two technique approaches for adopting the fault-tolerance: reactive and proactive fault-tolerance techniques. The difference between the two techniques lies in the fact that the proactive fault-tolerance techniques are expecting and predicting (fault is expected before occurrence) [2], while the techniques of reactive fault-tolerance are interacting and responding (fault is handled after occurrence). Therefore, based on the present status of cloud service network [3], the adaptive technique of fault-tolerance travels between the two techniques. Concerning cloud computing and virtualization, it can be easy to run swarm intelligence algorithms such as ACO and PSO simulation on multiple VM nodes. Cloud service infrastructures use the technique for fault-tolerance to tolerably form the faults. In this new method, the function of operating and management such as job scheduling aims at presenting cost-effective and high-performing services demanded by VMs and implemented by physical resource machines (host server) at certain times. This can efficiently solve the job scheduling problem by reducing the total execution time without any failure. As replication refers to duplication, different jobs are duplicated and implemented on various resources to reach effective implementation and desired outcomes. The tools that are used in this work are Hadoop and Apache Spark replication for implementing replication [4][5]. The job of scheduling problem in virtualization aims to assign various job tasks to VM nodes found at the network boundary. Host server machines/virtualization are more specifically distributed to various tasks of given jobs marking the services demanded by end-users so the CPU execution time, and the distributed memory requested by all jobs are reduced.

In this study, a new model called adapting fault-tolerant model (AFTM) is proposed. The focus of this study is on the implementation and analysis of the adaptive fault-tolerance through the job scheduling problem for the request resource services in infrastructure as a service (IaaS). The job scheduling problem governs the best job of various tasks implemented at the minimum level of cloud resources (e.g., less memory) in the shortest time of CPU

[1] *Sadoon Hussein , Department of Physic, College of Science, University of Mosul, Mosul, IRAQ*
ORCID ID : 0000-0002-4629-7466
[2] *Al-Hakam Ayad , Department of Arabic Language ,College of Arts, Tikrit University, IRAQ*
ORCID ID : 0000-0003-0107-3028
[3] *Nadia M. Mohammed, Department of Software, College of Computer Science and Mathematics, University of Mosul, Mosul, IRAQ*
[4] *Redhwan M. A. Saad,[4]Department of Electrical Engineering, Faculty of Engineering, Ibb University, Ibb 70270,YEMEN*
[5] *Department of Computer Engineering, Faculty of Engineering, Cairo University, Giza 12613, EGYPT*
ORCID ID : 0000-0001-7241-9327
Correspondence Author Email; sadosbio113@uomosul.edu.iq

execution based on PSO, ACO, Hadoop, and Apache Spark platforms. As a result, the tasks of the end-users achieve request with quicker implementation time of one's job tasks at the minimum cost with fault-tolerance. The algorithms of the job scheduling should have a new technique of the fault-tolerance ability, which means executing combined job scheduling in despite of failure in a host server or VMs arbitrary. There are many studies conducted to overcome the fault based on the fault-tolerance.

2. Related Works

Much research has been conducted on the fault-tolerance in cloud computing services. Cloud service infrastructure has presented problems pertaining real-time computing services. For example, Kalanirinika GR, et al. [2] suggested an approach of reactive fault-tolerance to facilitate exploit check pointing to accept the liability. Another study conducted by Egwutuoha I.P. et al. [6] in which a fault-tolerance (FT) technique was proposed for managing faults proactively in HPC systems. The objective of the study was to shorten the implementation case of the barricade chronometer in the occurrence of responsibility and enlarge a standard FT algorithm. For creating and managing cloud fault-tolerance, Jhawar et al. [7] found an advanced system level modular perspective and suggested all-inclusive complicated approach to share application details with developers and users on implementation of the fault-tolerance technology by adapting a devoted service layer. In the same vein, Hwang et al. [8] proposed a model targeting failure detection service through notification and scalable framework for allowing grid failures using simulation as to evaluate parameter.

Due to the dynamicity and heterogeneity of cloud resource availability, the fault-tolerance intelligent algorithm-based job scheduling techniques in cloud computing services were found significant in order to avoid job failure [9, 10]. In [11], the MTCT technique is presented, which is minimum-based time and cost trade-off for multipurpose workflow scheduling, to assist fault retrieval in the cloud. To evaluate this method, simulations are used to test its strength using four different real-life scientific workflow scenarios. The results display that fault retrieval has a significant effect on the criteria of performance and the MTCT algorithm is treasured for real-life workflow systems when both optimization goals are considered.

A job scheduling method in cloud services is introduced by Kumar and Aramudhan in [12], using hybridization of BA method with a gravity scheduling algorithm that takes into consideration the scheduling checks and the trust model. Depending on the reliability level, jobs are mapped to resources. The hybrid algorithm has been tested and it intelligently minimizes the time period and the number of failed operations compared to GVSA. However, the BA is known for its weak local research in solving complex problems.

NSGA-II is an intelligent technique that presents a security-driven solution for scheduling failure probability [13]. The Pareto dominance relationship is used to provide a set of non-dominant optimal solutions. This technique is integrated with GA which is evidenced by a number of experimental results. The average response time results are closely related to makespan results. However, the general trend seems to be more challenging to explain. The corresponding results display the usefulness of the provided method and GA for both medium and small planning jobs. However, the empirical results did not solve large and large-scale scheduling job in the cloud.

It has been found that the schemes of PSO are nature-inspired population extraction techniques. The social features of flock of birds and fish are simulated by the algorithms. The scheme attempts to develop resolutions based on the measure of quality called the fitness function by initializing a set of randomly distributed particles called potential solutions [14]. An improved PSO for managing the scheduling job of VMs in a cloud computing environment is presented by Yuan et al. [15]; they proposed a VM scheduling scheme which takes into consideration the calculation power of the processing fundamentals and also takes into account the calculation the system density.

The resources currently available in the cloud should be used for each scheduling point so that hobs failure can be avoided due to essential machine failure or overload. Modern dynamic scheduling methods do not take into account the fault-tolerance parameters or they are used to a certain degree at different levels of scheduling.

3. System Model

In this study, the suggested cloud service infrastructure consists of "M" local VMs of jobs service located in the IaaS of the virtualization as shown in Fig. 1. Hypothetically, the system is composed of administrator VMs who takes over job scheduling after getting all given job parameters (i.e. a set of tasks). Each job is provided by an end-user using a service order form to be operated in the cloud-computing infrastructure.

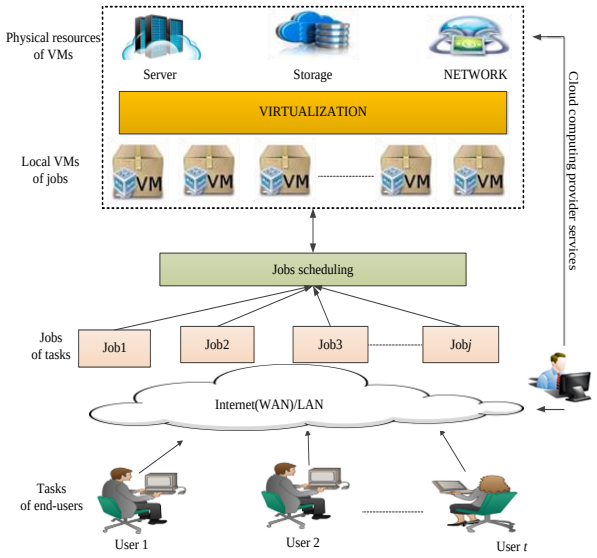


Fig 1. Architecture System of Cloud Provider Service and Jobs Scheduling

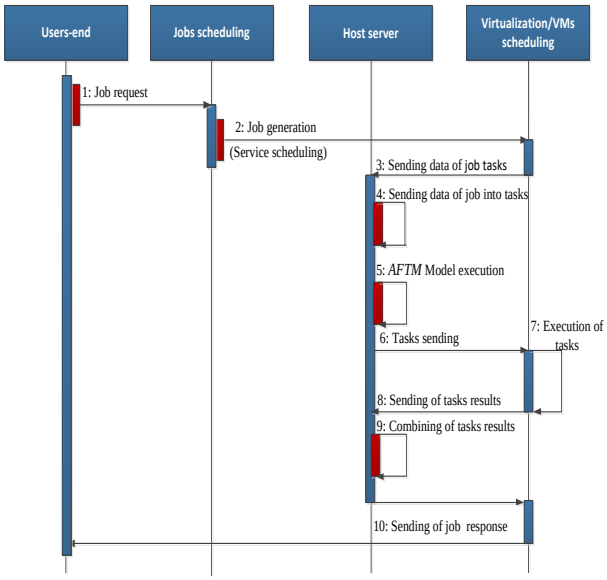


Fig 2. A Host Server System Model and Jobs Scheduling of Virtualization

The fault state in cloud service is a popular issue as well job scheduling as shown in Fig. 2. To guarantee job scheduling, a new method of algorithm called AFTM was proposed and done by the administrator VMs to figure out the ideal request which is implemented via virtualization VMs. Thus, the virtualization VMs can ensure the performance of the memory of the scheduled services and CPU. The model process was described by outlining the stages or steps of operating a scheduled service as illustrated in Fig. 2.

First, an end-user makes a service request to a job scheduling machine found at the VM of the virtualization of this cloud services infrastructure (stage 1). Next, the job generation of this request as service scheduling is sent by job scheduling machine (stage 2). Then, the VM sends data with parameters of the request to the host server machine which is far-off from the end-user (stage 3). The job is, then, decomposed by the host server machine into a set of tasks (stage 4). For finding an optimal job scheduling, the AFTM is implemented (stage 5). Next, each VMs gets its allocated task (stage 6). The task is implemented at the level of these VMs (stage 7). Each VMs results are sent to the host server (stage 8). The result is prepared by the host server based on the partial given results from the VMs (stage 9) to the end-users receives the final result as a service response (stage 10).

4. Problem Formulation

The job scheduling problem formula in the virtualization infrastructure is explained in this section. The job is defined as an operation that is similar to a service request created by the end-users of computers. Then, a series of 'n' jobs is scheduled and implemented by the infrastructure of virtualization. The Jobs are distinguished as follows:

$$Job\_tasks = \{J_{t1}, J_{t2}, \cdots, J_{ti}, \cdots, J_{tn}\}$$

Each *Job task* '*ti*' within '*n*' jobs can be divided into a set of '*s*' tasks, where ($1 \leq i \leq n$). Every task '*t*' within '*s*' tasks where ($1 \leq t \leq s$) is distributed to one VM '*j*' within '*k*' VMs where ($1 \leq j \leq k$) to be implemented, such as:

$$Job_{ti}Tasks = \{JTask^{a}_{ti1}, JTask^{b}_{ti2}, \cdots, JTask^{j}_{tit}, \cdots, JTask^{k}_{tis}\}$$

For example, tasks of job '*ti*':

$$Job_{ti}Tasks = \{JTask^{3}_{ti1}, JTask^{5}_{ti2}\}$$

are carried out as follows: first task ($JTask_{ti1}^{3}$) is performed in VM_3 and second task ($JTask_{ti2}^{5}$) is performed in VM_5. Consequently, each VMs VM_j can perform a set of disjoint subset of the spoiled jobs set. For its allocated jobs, VM_j assurances the performance of its tasks as the following:

$$VM_jTasks = \{JTask^{j}_{ax}, JTask^{j}_{by}, \cdots, JTask^{j}_{it}, \cdots, JTask^{j}_{ns}\}$$

Combining these general divided groups are a complete set of jobs. For instance, tasks are allocated to the *VMs VMj* after scheduling as the following:

$$VM_jTasks = \{JTask^{j}_{23}, JTask^{j}_{61}\}$$

Then, *VMj* carries out 3rd job task 2 and 1st job task 6. Side by side, the overall time of CPU execution allocated to *VMj* of all *'s'* tasks will be:

$$CPU\_Execution\_time(VM_jTasks) = sum_{1\le k\ge s}\left(JTask_{it}^{j}.startTime + JTask_{it}^{j}.ExecutinTime\right), i \in jobs\ of\ selected\ tasks \quad (1)$$

where $JTask_{it}^{j}$.StartTime symbolizes the initial point of task *'t'* of a job *'i'* is performed on *VMj* and $JTask_{it}^{j}$.ExeTime symbolizes the time of CPU execution at *VMj* of task *'t'*. In addition, the assigned memory to task *'t'* allocated to *VMj* is computed using the following formula:

$$Memory(VM_jTasks = max_{\ 1\le t\ge r}\left(JTask_{it}^{j}.AllocatedMemory\right), i \in jobs\ of\ selected\ tasks \quad (2)$$

Consequently, the scheduling of job task in the infrastructure of cloud computing can be calculated using the following formula:

$VMTasks = \{VM_1Taks, VM_2Taks, \cdots, VM_kTaks\}, where: VM_1Taks = \{JTask_{ax}^{j}, JTask_{by}^{j}, \cdots, JTask_{it}^{j}, \cdots, JTask_{ns}^{j}\}$,as described above.

5. Cost Function

The cost function is used to evaluate the performance of the expected result (*VMTasks*). As a minimized function, the cost function is used to gauge the optimization of the two objectives mentioned above. Allocated memory size and CPU implementation time are formulated as following:

$$Cost_{function(VMTasks)} = Min\left[\sum_{m}^{j=1}\left(Cost_{unction\left(JTask_{ik}^{j}, VM_j\right)}\right)\right]$$

$$, where, Cost_{unction\left(JTask_{ik}^{j}, VM_j\right)} = w_1.CPU_{Execution_{Time\left(VM_jTasks\right)}} + w_2.Memory\left(VM_jTasks\right) \quad (3)$$

Where, *w1* and *w2* are used to stress the significance of both assessed objectives, allocated memory and CPU execution time, as a fixed weight factors. In other words, the preference of the decision maker using PSO and ACS in this work is reflected by the selection of the weights of the multi-objective optimal approaches. The general parameter values and PSO parameter are shown in Tables 1 and 2.

Table 1. General Parameter Values

Parameter	*value*
VM numbers	20
Number of tasks	[20 100]
Processing rate	[50,500] MIPS
Bandwidth	10,100,512,1024 Mbps

Table 2. Pso Parameter

Parameter	*value*
W	1 or 0.99
c_1	0.5
c_2	0.5
r_1	0.5
r_2	0.5
Max iteration	200

6. Swarm Intelligence Algorithm (SIA)

SIA is built on the combined performance of self-ordered systems. Standard swarm intelligence systems comprise ACO, PSO, Stochastic Diffusion Search, and the Artificial Bee Colony (ABC) [16], [17].

6.1. Practical Swarm Optimization (PSO)

PSO is, to some extent, a new randomly optimized technique that mimics the bird flocking swarm behavior. According to PSO, an individual in the swarm, named a particle, embodies a prospective solution. Every particle has a fitness value and a velocity; it learns the experiences of the swarm to exploration for the global optima [18]. Traditional PSO can be represented in Fig. 3.

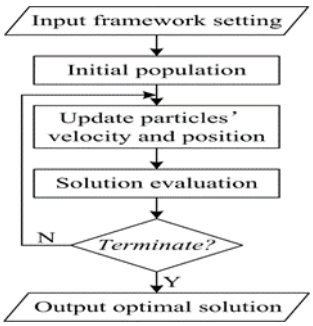


Fig 3. Flow Chart of Standard PSO Algorithm [19]

6.2. Ant Colony Optimization (ACO)

The ACO is a potential technique used for handling computational issues. This technique attempts to minimize

optimal paths through graphs [20] as shown in Algorithm 1. The environment is used as a tool for Ant communication. By the aid of the deposited pheromone, an altercation of information occurs, such as the status of their "work". Owing to the information tradeoff has a local scope, the location of ants are perceived by left the pheromones.

Algorithm 1: ACO Algorithm

1. Initialization:

The heuristic information, the pheromone trails, and the restrictions are configured.

2. Iterative Loop:

The starting job can be determined by a colony of ants.

Create a complete schedule for each ant.

3. Repeat:

Selecting the next processing job, using the following:

State transition rule.

Local updating rule.

Until a complete schedule is built, use the following:

Local search process.

Global updating rule.

4. Termination:

If the interactions number is reach to the maximum,

Then STOP.

Else go to STEP No.2.

7. Apache Spark Technology

Like MapReduce, Spark is a distributed calculating framework although it scores elastic allocated dataset that gives a richer model than MapReduce. To support graph calculation algorithms and complex data mining, Spark permits iterations of data in memory algorithms [21]. It also presents simple programming interface (API's) for several analytical algorithms that includes graph processing, machine learning, SQL Queries, and real-time data streaming. The interface enables the developer of application to simply use the storage resources, memory, and CPU through a cluster of servers for processing complex and large datasets. The Spark Streaming is characterized by having some advantages: (a) reaches the second delay and runs on 100+ nodes, (b) memory-based implementation engine, with effective and fault-tolerant attributes, (c) interactive queries and integration of Spark's batch, and (d) Up to 10 times faster than Hadoop MapReduce [22][23].

Apache Spark is seen as a simple construction with two nodes only (Worker and Master) which runs with a cluster manager such as Spark or Hadoop, as shown in Fig. 4.

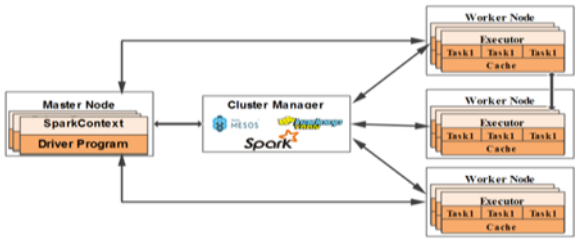


Fig 4. Apache Spark architecture

The batch is processed by the Spark program through operating the interfaces provided by Resilient Distributed Datasets (RDDs) [24]) such as reduction, map, and filter. With regard to the Spark streaming, the process of the DStream (RDD sequence signifying the data flow), which presents the interfaces, are similar to those provided by RDD.

8. Experimental Environment Setting

In the experimental part of the study, a number of simulation tests are conducted based on different computing resource infrastructures. Virtualizations are diverse with regard to their storage capacity and processing capability [25]. To assess the usefulness of the model regarding scheduling the job tasks, each VM has its own operational ability characterized by the rate capacity of CPU clock of the VM (measured in GHz) as well as its own storage capacity characterized by available memory (measured in GBytes). Table 3 shows the availability of the memory size of the CPU clock rate and each one of the 20 simulated VMs. For example, VM No. 1 provides 1.25 GHz as rate of CPU clock requires implementing one job instruction, 1.0 GBytes of available memory.

In this study, 5 jobs are given to be performed: 5, 10, 15 or 20 VM. Every job is composed of 5 tasks which are further composed of a number of instructions and requires space on a memory as shown in Table 4.

9. Performance Evaluation

In this work, the performance evaluation is carried out to assess the potency of the AFTM to job scheduling in virtualization VMs. To reach this aim, the AFTM framework is executed in Java based Hadoop and Apache Spark platforms. For experimental purposes, the suggested model's programs are implemented adopting Java coding and Scale. All programs can be run with different number of nodes by Apache Spark clusters. The setting of the

experiment used 5 VM nodes (VM_1, VM_2, VM_3, VM_4 and VM_5). On Hadoop2.0, all cluster nodes have the same configuration that comprises CPU (3.9 GHz), 500 GB storage, and 32 GB RAM, as shown in Fig. 5

VMj	*Measure*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*
Rate of CPU clock	GHz	1.07	1.02	0.77	1.01	0.93	1.67	0.92	0.70	1.21	1.70	0.77	1.11	1.00	0.50	1.65	0.73	0.93	1.00	1.25	1.00
Size of available Memory	GBytes	1.01	0.78	1.65	1.50	1.44	1.0	1.12	0.19	1.0	1.42	1.21	1.12	1.20	0.81	1.12	1.42	1.23	1.50	1.00	0.9

Table3. Available Memory Size of VM Nodes and CPU Clock Rate

	Job 1	Job 2	Job 3	Job 4	Job_5			
Task 1	*Instructions*		1×10^9		1×10^9	3×10^9	2×10^9	2×10^9
	Memory		0.20		0.10	0.20	0.30	0.20
Task 2	*Instructions*		1×10^9		3×10^9	4×10^9	1×10^9	2×10^9
	Memory		0.30		0.20	0.10	0.20	0.10
Task 3	*Instructions*		4×10^9		1×10^9	2×10^9	3×10^9	2×10^9
	Memory		0.40		0.30	0.20	0.10	0.10
Task 4	*Instructions*		3×10^9		3×10^9	2×10^9	1×10^9	1×10^9
	Memory		0.20		0.10	0.20	0.30	0.30
Task 5	*Instructions*		4×10^9		2×10^9	1×10^9	2×10^9	1×10^9
	Memory		0.20		0.30	0.10	0.30	0.20

Table4. Parameters of Every Task of a Job.

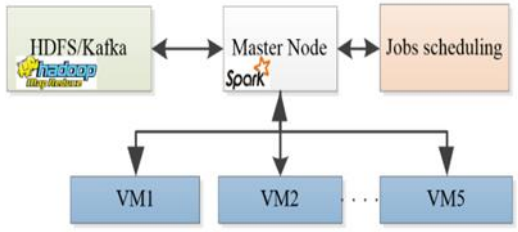


Fig 5. Apache Spark's Cluster Architecture Experiment

Apache Spark cluster is formed by a local server machine service. To assess our proposed model, two evaluation metrics of performance are used as the following:

A. CPU execution time is defined as the period between the beginning and the end of a given task implemented on VMs. We presume that for each instruction, it is necessary to have one clock cycle to be executed. The CPU execution time can be computed as the following:

   Time of CPU execution = instructions number of a task (i.e., clock cycles for a task) /clock rate.

B. Distributed size of memory is defined as the quantity of memory (i.e., the main storage unit) of a VM, devoted to the given task execution.

With reference to metrics above, the results obtained by AFTM are compared with those ones obtained by genetic algorithm and the traditional PSO.

10. Results Analysis for Simulation Experiment

Fig6 and Fig7 show the results of AFTM, PSO, and GA executions of 5 jobs which consist of 5 tasks. For cases (5, 10, 15, and 20 VM nodes in the infrastructure of cloud computing). Fig. 6 shows the CPU execution time yielded after implementing all job tasks by various VM nodes.This demonstrate that more models can bring about lowered implementation time for various simulation tests. The distributed memory with all job tasks after being scheduled with more AFTM, PSO, and GA are illustrated in Fig. 7. Furthermore, the smallest size of memory as in the case of 20 VM nodes is allocated by AFTM

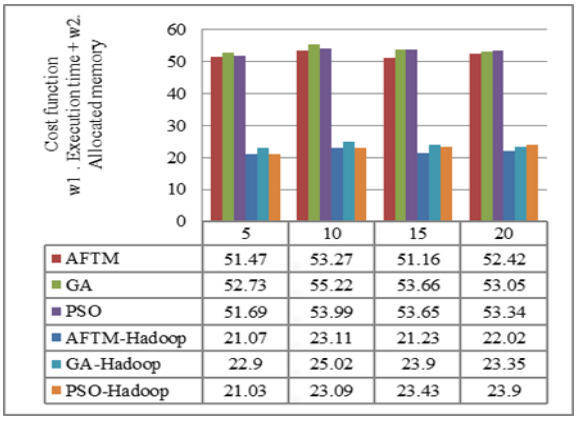
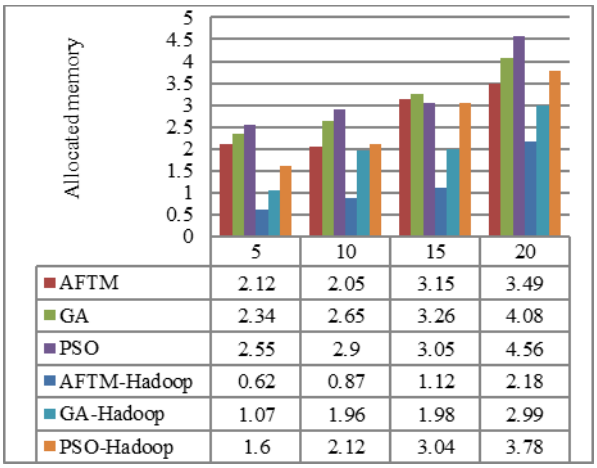


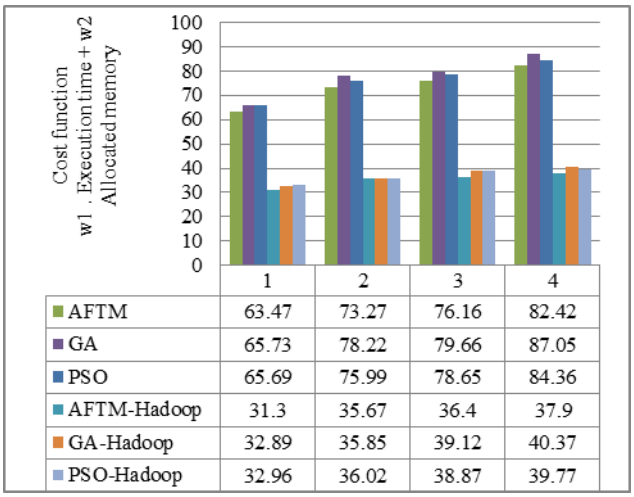
Fig 6. Job Execution Time after AFTM, PSO and GA Scheduling with Apache Spark



	5	10	15	20
AFTM	2.12	2.05	3.15	3.49
GA	2.34	2.65	3.26	4.08
PSO	2.55	2.9	3.05	4.56
AFTM-Hadoop	0.62	0.87	1.12	2.18
GA-Hadoop	1.07	1.96	1.98	2.99
PSO-Hadoop	1.6	2.12	3.04	3.78

Fig 7. Allocated Memory by Jobs after AFTM, PSO and GA Scheduling with Apache Spark

As shown in Fig.8, the multi-objective job-scheduling problem is represented by the cost function in which AFTM outperforms PSO and GA for all tests. AFTM-Hadoop can give minimum cost services out of the allocated memory by taking into consideration transformation and crossover operations with the reiterative times growing in three algorithms. Compared with AFTM, GA, and PSO that consider local, and global scope of the solution space, and also add a random sampling so that the AFTM-Hadoop has a stronger global optimization capability based on Apache Spark.



	1	2	3	4
AFTM	63.47	73.27	76.16	82.42
GA	65.73	78.22	79.66	87.05
PSO	65.69	75.99	78.65	84.36
AFTM-Hadoop	31.3	35.67	36.4	37.9
GA-Hadoop	32.89	35.85	39.12	40.37
PSO-Hadoop	32.96	36.02	38.87	39.77

Fig 8. Cost Function of AFTM, PSO and GA VMs Scheduling Problem with Apache Spark

11. Conclusion

This study aimed at investigating the concept of fault management with the optimization of the job-scheduling problem in the calculating infrastructure. Based on the Apache Spark and swarm intelligence algorithms, a new effective method called adapting fault-tolerant model (AFTM) was proposed for job scheduling in the cloud computing environment. For reducing the resource cost, the performance was analyzed. The suggested model covers the implementation of virtualizations with the job task selection to health monitoring for fault diagnoses with Apache Spark. The objective was to find an optimized cost tradeoff between CPU execution time required by virtualization services and the allocated memory.

Acknowledgements

Authors are sincerely grateful to the anonymous peer reviewers for their precious comments given to improve the quality of this work.

Author contributions

Sadoon AbdulAllah: Conceptualization, Methodology, Writing-Original draft preparation.

Al-Hakam Ayad: Conceptualization, Visualization.

Nadia Maan: Data curation, Validation. Investigation, formal analysis. Software, Writing- Original.

Redhwan M. A. Saad: Investigation, Writing, Reviewing, Editing, and Supervision.

References

[1] TYAGI, Rinki; GUPTA, Santosh Kumar. A Survey on Scheduling Algorithms for Parallel and Distributed Systems. In: Silicon Photonics & High Performance Computing. Springer, Singapore, 2018. p. 51-64.

[2] PRAKASH, Shiva, et al. A Literature Review of QoS with Load Balancing in Cloud Computing Environment. In: Big Data Analytics. Springer, Singapore, 2018. p. 667-675.

[3] Kalanirinika GR, et al." fault tolerance in cloud using reactive and proactive techniques".

[4] Alkasem, A., Liu, H., Zuo, D., & Algarash, B. (2018). Cloud Computing: A model Construct of Real-Time Monitoring for Big Dataset Analytics Using Apache Spark. In Journal of Physics: Conference Series (Vol. 933, No. 1, p. 012018). IOP Publishing..

[5] Ameen Alkasem, Hongwei Liu and Decheng Zuo. CloudPT Performance Testing for Identifying and Eliminating Bottlenecks in Dynamic Cloud Services[C]. 18th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), 2018.

[6] Egwutuoha, I.P., Chen, S., Levy, D., Selic, B. and Calvo, R., 2012, November. A proactive fault tolerance approach to High Performance Computing (HPC) in the cloud. In Cloud and Green Computing (CGC), 2012 Second International Conference on (pp. 268-273). IEEE.

[7] Jhawar, R., Piuri, V. and Santambrogio, M., 2013. Fault tolerance management in cloud computing: A system-level perspective. IEEE Systems Journal, 7(2), pp.288-297.

[8] Hwang, S. and Kesselman, C., 2003. A flexible framework for fault tolerance in the grid. Journal of Grid Computing, 1(3), pp.251-272.

[9] Patra PK, Singh H, Singh G (2013) Fault tolerance techniques and comparative implementation in cloud computing. Int J Comput Appl 64(14):37–41.

[10] Nawi NM, Khan A, Rehman M, Chiroma H, Herawan T (2015) Weight optimization in recurrent neural networks with hybrid metaheuristic Cuckoo search techniques for data classification. Math Probl Eng 501:868375.

[11] Xu H, Yang B, Qi W, Ahene E (2016) A multi-objective optimization approach to workflow scheduling in clouds considering fault recovery. KSII Trans Internet Inf Syst 10(3):976–995. doi:10.3837/tiis.2016.03.002.

[12] Kumar VS, Aramudhan M (2014) Hybrid optimized list scheduling and trust based resource selection in cloud computing. J Theor Appl Inf Technol 69(3):434–442

[13] G\ca\casior J, Seredyński F (2013) Multi-objective parallel machines scheduling for fault-tolerant cloud systems. In: Joanna K, Di Martino B, Talia D, Xiong K (eds) Algorithms and architectures for parallel processing. Springer, Switzerland, pp 247–256. doi:10.1007/978-3-319-03859-9_21

[14] Kaveh A (2014) Particle swarm optimization. In: Advances in metaheuristic algorithms for optimal design of structures. Springer, Switzerland, pp 9–40. doi:10.1007/978-3-319-05549-7

[15] Yuan H, Li C, Du M (2014) Optimal virtual machine resources scheduling based on improved particle swarm optimization in cloud computing. J Softw 9(3):705–708

[16] Kaur, J., Kalra, A., & Sharma, D. (2018). Comparative Survey of Swarm Intelligence Optimization Approaches for ANN Optimization. In Intelligent Communication, Control and Devices(pp. 305-314). Springer, Singapore.

[17] Lin, F. P. C., & Phoa, F. K. H. (2018). An efficient construction of confidence regions via swarm intelligence and its application in target localization. IEEE Access, 6, 8610-8618.

[18] Chu, S. C., Huang, H. C., Roddick, J. F., & Pan, J. S. (2011, September). Overview of algorithms for swarm intelligence. In International Conference on Computational Collective Intelligence(pp. 28-41). Springer, Berlin, Heidelberg.

[19] Zhang, X., & Zhang, X. (2017). Thinning of antenna array via adaptive memetic particle swarm optimization. EURASIP Journal on Wireless Communications and Networking, 2017(1), 183.

[20] https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms

[21] Ameen Alkasem, Hongwei Liu, Muhammad Shafiq, and Decheng Zuo, "A New Theoretical Approach: A Model Construct for Fault Troubleshooting in Cloud Computing," Mobile Information Systems, vol. 2017, Article ID 9038634, 16 pages, 2017. doi:10.1155/2017/9038634.

[22] Salloum, S., Dautov, R., Chen, X., Peng, P. X., & Huang, J. Z. (2016). Big data analytics on Apache Spark. International Journal of Data Science and Analytics, 1(3-4), 145-164.

[23] Mavridis, I., & Karatza, H. (2017). Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark. Journal of Systems and Software, 125, 133-151.

[24] Ameen Alkasem, Hongwei Liu, and Decheng Zuo. "Utility Cloud: A Novel Approach for Diagnosis and Self-healing Based on the Uncertainty in Anomalous Metrics." In Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences (ICMSS '17), Yulin Wang (Ed.). ACM,NewYork,NY,USA,99-107.DOI: https://doi.org/10.1145/3034950.3034967, (2017).

[25] Vasconcelos, P. R. M., & de Araújo Freitas, G. A. (2014, December). Performance analysis of Hadoop MapReduce on an OpenNebula cloud with KVM and OpenVZ virtualizations. In Internet Technology and Secured Transactions (ICITST), 2014 9th International Conference for (pp. 471-476). IEEE.