Completion of Circle Material Problems Using the Jumping Task Approach to Junior High School Students

by Nadya Alvi

Submission date: 21-Jun-2023 06:57PM (UTC+0700)
Submission ID: 2120285595
File name: 31744-104897-1-PB.pdf (474.62K)
Word count: 3219
Character count: 18254

Transcendent Journal of Mathematics and Application
ISSN 2964-1845
Vol. 2, No. 1, Apr 2023, pp. 34-42

Completion of Circle Material Problems Using the Jumping Task Approach to Junior High School Students

Nadya Alvi Rahma[1,*] and Nissa Kurnia Sari[1]
[1] Department of Mathematics Education, Universitas Islam Negeri Sayyid Ali Rahmatullah Tulungagung, Indonesia
*Corresponding email: nadyaalvirahma@uinsatu.ac.id

ARTICLE INFO

Article History
Received : 11-03-2023
Revised : 11-04-2023
Accepted : 12-04-2023

Keyword
Adversity Quotient; Circle Problem; Critical Thinking; Jumping Task; STEM.

ABSTRACT

In order to practice critical thinking skills, it is necessary to get used to practicing questions such as questions based on jumping tasks. The jumping task question is a question with a level of difficulty above the curriculum. In solving math problems based on jumping tasks, students often face obstacles. This study aims to describe the profile of students' critical thinking towards solving mathematical problems based on jumping tasks on circle material. The research was conducted at MTsN 2 Blitar with 2 students each having adversity quotient types of climber, camper, and quitter. The data collection technique used was an adversity response profile questionnaire, a jumping task-based math problem solving test and interviews. The research results obtained showed that at the stage of understanding the problem both climber, camper and quitter subjects were able to show interpretation after reading the questions. However, the quitter subject is less precise in interpreting important information from the problem. In the problem-solving planning stage, the plan presented by the climber subject was more appropriate than the plan presented by the camper subject, while the quitter subject was unable to develop a settlement plan and the work process stopped at this stage.

DOI: …

1. INTRODUCTION

Nowadays, the advancement of technology has made cloud computing become a new paradigm for delivering resources or services through the network. Cloud computing is internet-based computing in which different services, such as: servers, computing resources, storages, software platforms or even applications, are delivered to the device or computer through the internet (Manzoor et al., 2020). Resource allocation is very crucial concern in the cloud environment. Resource allocation is the process of assigning the number of resources needed by cloud applications (Liu et al., 2017; Naik et al., 2021). If the resource allocation is not managed precisely, the cloud services may lack the resources or waste the resources. Auto-scaling mechanism is one approach used in cloud environment in which service providers can maintain the resources and reduce waste resources by automatically increasing or decreasing them when needed (Kumar, 2022). It will monitor information about Central Processing Unit (CPU) utilization, disk input/output (disk I/O) and network I/O on server side to trigger system configuration (Huang et al., 2023).

Several works have been done with different approaches. Lin et al. (2011) proposed threshold-based approach for dynamic resource allocation scheme implemented using CloudSim that dynamically allocates the virtual resources based on their load change. Kang et al. (2013) proposed an auto-scaling method that meets SLAs such as deadlines, cost-oriented or performance-oriented policies to provide efficient resource utilization in hybrid cloud. Ighare and Thool (2015) addressed automated resource allocation system which uses the utilization threshold to make decision of the virtual machine migration. Sultana and Kawadkar (2022) used thresholds with attributes and amount (TAA) in request for task scheduling and

resource allocation strategy to improve the quality of service. Ashawa et al. (2022) presented a framework to optimize cloud resource allocation using the LSTM machine learning algorithm. However, few studies have been done on allocating resources at the application level and know exactly the performance from client-side experiences.

In this paper, we focus on allocating resources at the application level rather than mapping the physical resources to virtual resources for better cloud resource utilization. For this experiment, we employ Amazon Elastic Compute Cloud (Amazon EC2) instances. We propose a novel cloud resource management framework which supports auto-scaling. The proposed system will monitor the end-user's response time directly from client-side. This paper defined several thresholds with Quality of Services (QoS) considerations to indicate the process of allocating or terminating the virtual resources when workload of application changes. The monitoring system will be deployed in client hosts which will send requests for CPU utilization, response time and error rate sampling. We compared the sampling with threshold to optimize the decision of resource reallocation. The goal of this study was to dynamically allocate virtual resources among the cloud applications based on their workload changes in order to improve resource utilization, prevent resource starvation and reduce the user usage cost.

2. METHODS

2.1. System architecture

The proposed framework consists of Cloud Services, Service Builder, Resource Manager and Elastic Load Tester. Cloud Services provides web services to the end-users. Service Builder provides infrastructures, management, monitoring, maintenance and other features needed to run services (Swain et al., 2022). Resource Manager defines user-policies to assigns sufficient instances needed by the services according to the real end-user experience. It ensures that the running services meet the user requirement with minimum usage cost. Elastic Load Tester is a flexible load-testing tool which generates a certain scale of distributed testing framework to simulate a large number of clients for load testing (Praveenchandar and Tamilarasi, 2022). The detail of each component is described separately in the following sections.

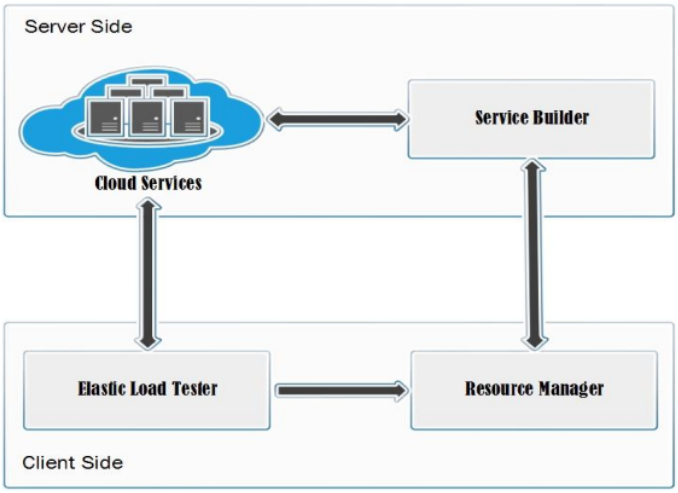

Figure 1. System architecture

2.1.1. Cloud Services

Cloud Services can be accessed directly by the end-user. It is a group of virtual web servers or instances that work together in an Auto Scaling Group which is attached to Load

Balancer. Auto Scaling Group ensures that services provided can satisfy all user requests. Auto Scaling is a special feature in which instances can be launched or terminated automatically based on user-defined policies. Policies are defined by Resource Manager which intends to assign instances efficiently. By providing efficient instances needed by Cloud Services, the performance of the services can be assured and the cost usage can be reduced. For example: during high workload traffic, a certain number of instances can't satisfy huge number of requests; therefore, additional instances are required. During low workload traffic, terminating instances precisely can help to reduce cost usage. On the other hand, Load Balancer will automatically distribute the incoming application request across multiple virtual web servers. So that the workload of each server will remain balance which can prevent one instance from fault because of the workload excess (Dharmara and Kavitha, 2022).

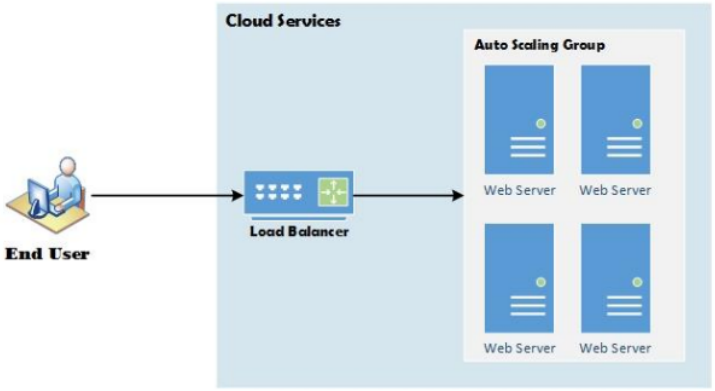

Figure 2. Cloud services

2.1.2. Service Builder

Service Builder is a tool set of service features offered by cloud providers to give costumer the ability to manage the infrastructures. As Figure 3 shows, the Service Builder architecture consists of three layers: Service Infrastructure, Service Features and Application Programming Interface Tools (API Tools).

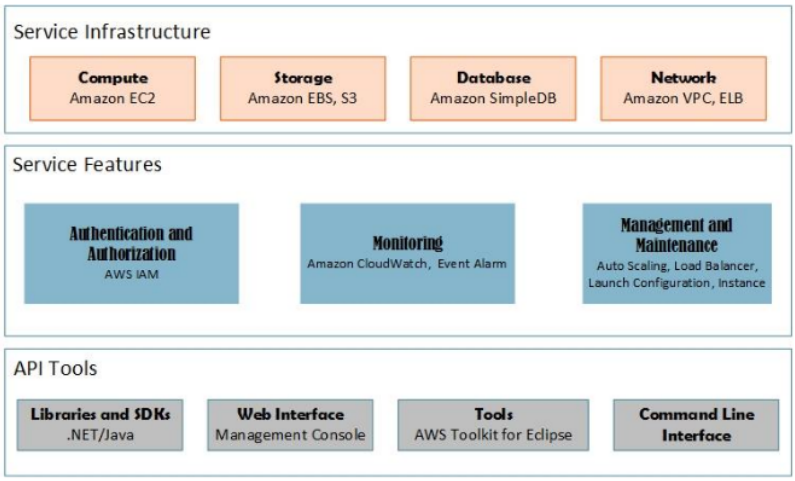

Figure 3. Service builder layer

The top layer "Service Infrastructure" interacts directly with Cloud Services. Service Infrastructure layer provides main resources to run Cloud Services which includes computing, storage, database and networking resources. This framework architecture currently only supports Amazon Web Service (AWS) products, such as: Amazon Elastic Compute Cloud (Amazon EC2) for compute, Amazon Elastic Block Store (Amazon EBS) and Amazon Simple Storage Service (Amazon S3) for storage, etc.

Service Features is the main function of the Service Builder. It contains all features needed to deploy, manage and monitor instances. In general, Service Features is categorized into three components: Authentication and Authorization, Monitoring, and Management and Maintenance. Authentication and Management is required for access control. It enables users to securely control access to AWS services and resources. AWS Identity and Access Management (IAM) is the example of Authentication and Management provided by AWS. By using IAM, users can control the access to AWS resources to create, delete or manage resources. Monitoring offered by AWS such as: AWS CloudWatch is used to monitor cloud resources utilization, application performance and operational health. AWS CloudWatch can collect data metrics, monitor log files and set alarms for specific instances. Management and Maintenance plays a role in managing Auto Scaling, Load Balancer, Launch Configuration, Instances, etc.

Application Programming Interface (API) Tools includes all libraries, Software Development Kits (SDKs), command line interfaces, web interfaces and other tools link Resource Manager and other Management tools to Service Builder. It is a set of commands, libraries and protocols used to provide easy access for managing AWS services. The Amazon API tools serve as the client interface to the Amazon web service. These tools can be utilized by both users and developers. User API tools can help user to manage their AWS services while developer tools will assist developers to optimize their application products.

2.1.3. Resource Manager

Resource Manager ensures that Cloud Services has enough resources derived from Service Builder to run web application. Resource Manager defines user-policies to assigns sufficient instances needed by the services according to the real end-user demand. It ensures that the running services meet the user requirement with minimum usage cost. Resource Manager consists of two modules: Monitoring module and Allocation module.

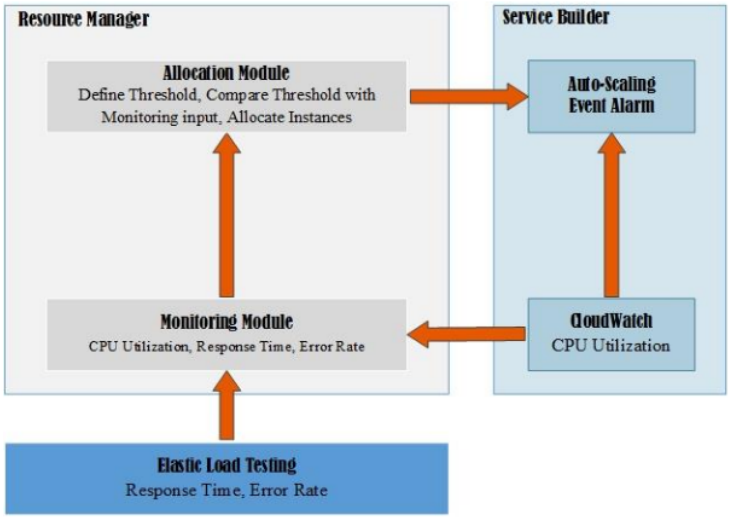

Figure 4. Resource manager

Monitor module measures and collects the workload and performance metrics of web server. There are two types of metrics collected:

1. Server performance

Server performance metric collected by Monitoring module is CPU utilization of the web server. AWS CloudWatch periodically get the percentage of CPU utilization and pass it to monitoring module and Event Alarm to trigger Auto Scaling function whenever meet the policies defined by Allocation module.

2. Client-side experience

Client-side experience metrics including response time and error rate are used to guarantee the QoS. Elastic load testing using Apache JMeter plays important role to simulate the real client behavior. Apache JMeter gets the response time and the percentage of error rate while making requests for the services to the server. Then, these metrics will be forwarded to allocation module (Qureshi et al., 2020).

Allocation module reads the values from monitoring module and user-defined configuration. These values will be used to make decision for Auto Scaling. The detail metric values are shown in Table 1.

Table 1. Metric values used in Allocation module

	Variable Name	Description
User-defined values	*upThr*	Upper Threshold
	lowThr	Lower Threshold
	tOut	The period in which condition must be met to trigger a scale out (second)
	tIn	The period in which condition must be met to trigger a scale in (second)
	vResponse	Response time defined value for QoS considerations (ms)
	vError	Error rate defined value for QoS considerations (%)
	n	The number of instances to be allocated/terminated
	tCooldown (optional)	The amount of time need before another scaling activity can start (second)
Monitoring metrics	*pCPU*	CPU utilization (%)
	tResponse	Response time (ms)
	nError	Error rate (%)

Allocation module compares pCPU value from monitoring module against the threshold values (upThr or lowThr) for a specific tIn second in order to determine whether the server needs more virtual resources or excessive virtual resources owned by web application based on the actual need (Sahu and Agrawal, 2015; Shahapure et al., 2021). On the other hand, the client-side monitoring values, such as: tResponse and nError are compared against vResponse and vError from user-defined values in order to guarantee the QoS. The pseudo code of allocation procedure is depicted as follows:

Scale out:

if pCPU > upThr || tResponse > vResponse || nError > vError for tOut then
    nInstances = nInstances + n and
    do nothing for tCooldown seconds

Scale in:

if pCPU < lowThr for tIn then
    nInstances = nInstances - n and
    do nothing for tCooldown seconds

2.1.4. Elastic Load Tester

Elastic Load Tester is a load-testing tool designed to test load behavior and measure performance. There are many different testing tools, such as Apache JMeter, Load Impact and Locust, for load test of web services. In this experiment, Apache JMeter was used as a testing tool. Apache JMeter is an open-source application designed for testing and measuring the performance of server which is written in Java. It supports testing performance both on static and dynamic resources which can simulate a heavy load on the servers. This load testing tool can used to load performance test on many different types of server/protocol, such as: web via Hypertext Transfer Protocol (HTTP/HTTPS), Simple Object Access Protocol (SOAP) / Representational State Transfer (REST), File Transfer Protocol (FTP), database via Java Database Connectivity (JDBC), Lightweight Directory Access Protocol (LDAP), message-oriented middleware via Java Message Service (JMS), mail via Simple Mail Transfer Protocol (SMTP) / Post Office Protocol version 3 (POP3) / Internet Message Access Protocol (IMAP), MongoDB, Transmission Control Protocol (TCP), etc.

Apache JMeter can simulate a large number of users and avoid a bottleneck arose on the tester-side by implementing distributed testing. Apache JMeter supports distributed mode for testing, in which we can remote many hosts concurrently to simulate a high workload. Figure 12 shows the basic concept of distributed testing. JMeter Slaves deploy on many different hosts, and these slaves are responsible to process job assigned by JMeter Master. JMeter Slaves send requests periodically to the testing target for sampling. Finally, JMeter Master collects the testing result from the testing target.

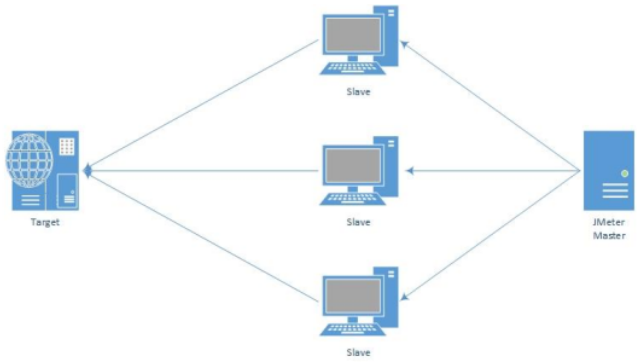

Figure 5. Distributed testing on Apache JMeter

2.2. Experiment setting

We used Resource Management Framework tool to deploy web server target and tester clients. The target system tested on the experiment is a simple web service which is located on Amazon Web Service us-east-1 region. The target web server employs m3.medium instances with Bitnami LAMP Stacks installed on it and linked to AWS Elastic Load Balancer (ELB). The tester clients consists of five t1.micro instances which are located on Amazon Web Service ap-northeast-1 region.

Table 2. Specification of AWS instances used in the experiment

Name	EC2 Units	vCPU	Memory (GiB)	Storage (GB)	Image	Price per Hour
m3.medium	3	1	3.75	1x4 SSD	Bitnami-lampstack Ubuntu-12.04	$0.07
t1.micro	Variable	1	0.615	EBS Only	Ubuntu 14.04 LTS	$0.02

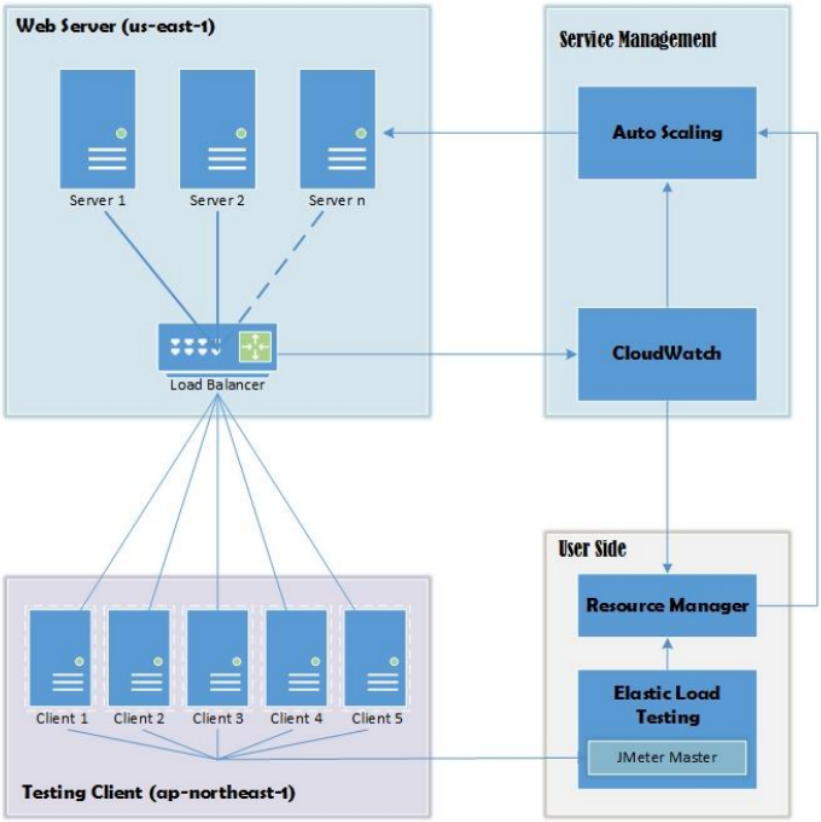

Figure 6. Experiment setting

2.3. Experiment procedure

The experiment procedure steps are explained as follow:

Step 1. Make a workload configuration for simulating user workload pattern. Figure 2 shows the number of user threads in one testing client for duration of half an hour. Initial workload is at 0 thread. The number of threads increases periodically to 40 threads per second and drops back to 0. Each threshold of the experiment uses the same workload patterns.

Step 2. Make a service deployment configuration for web server. The initial number of server nodes is one and maximal node is four with tCooldown is 2 minutes and evaluation period is every one minute.

Step 3. Build a distributed architecture workload generator with five tester-nodes.

Step 4. Start the testing with four different thresholds (table 3) for each level of QoS guarantees (table 4).

Step 5. Analyze and compare results, then decide the best threshold by using Cost Performance Index (CPI) formula with some modifications.

$$CPI = \frac{EV}{AC},$$

where EV is Earned Value which is the actual benefit value earned from the services while AC is Actual Cost which is the amount of cost needed to provide the services. In this case, the

3. CONCLUSION

Resource allocation plays a very important role for the service provision. If the allocation of resources is not managed precisely, the cloud services may starve or may waste the resources. This situation will lead the services running on underutilization of resources. To address this problem, we propose a novel cloud resource management framework which supports dynamic auto-scaling. The proposed system monitors the end-user's response time directly from client-side. We defined several thresholds and QoS guarantee levels in order to trigger the decision of allocating or terminating the virtual resources when workload of application changes. The monitoring system will be deployed in client hosts which will send request for CPU utilization, response time and error rate sampling. We compared the sampling with threshold to optimize the decision of resource reallocation. The experimental results show the proposed framework with proper threshold can improve the efficiency of resource utilization and reduce the usage cost. Currently, the proposed system uses static and predefined threshold values. Selecting proper threshold values is key to success in this approach. A lower value leads instances change frequently with high performance while a higher value makes instances less of responsiveness. Employing intelligent system to automatically define threshold values based on the workload history could be considered as the future direction.

ACKNOWLEDGMENTS

Thanks to the Department of Informatics, Universitas Syiah Kuala and NCTU-DSCLAB for the facilities provided to complete this research.

REFERENCES

Ashawa, M., Oyakhire, D., Osamor, J., & Riley, J. (2022). Improving Cloud Efficiency through Optimized Resource Allocation Technique for Load Balancing Using LSTM Machine Learning Algorithm. Journal of Cloud Computing, 11(87), 1-8.

Dharmara, S., & Kavitha, P. (2022). Task Scheduling and Resource Allocation in Cloud Computing Using a Heuristic Approach. YMER, vol. 21 : issue 12, pp. 1769-1778.

Huang, S.-Y., Chen, C.-Y., Chen, J.-Y., & Chao, H.-C. (2023). A Survey on Resource Management for Cloud Native Mobile Computing: Opportunities and Challenges. Symmetry, 15, 53

Ighare, R., & Thool, R. (201 Threshold based Dynamic Resource Allocation using Virtual Machine Migration. International Journal of Current Engineering and Technology, 5(4), 1-7.

Kang, H., Koh, J., Kim, Y., & Hahm, J. (2013). A SLA Driven VM Auto-scaling Method in Hybrid Cloud Environment," Netw. Oper. Manag. Symp. (APNOMS), 15th Asia-Pacifi pp. 1–6.

Kumar, A. (2022). Study on Cloud Computing for Efficient Resource Allocation and Scheduling Approaches. Journal of Contemporary Issues in Business and Government, 28(4), pp. 1379-1387.

Lin, W., Wang, J., Liang, C., & Qi, D. (2011). A Threshold-based Dynamic Resource Allocation Scheme for Cloud Computing, Procedia Eng., vol. 23, pp. 695–703.

Liu, N., Li, Z., Xu, J., Xu, Z., Lin, S., Qiu, Q., Tang, J., & Wang, Y. (2017). A Hierarchical Framework of Cloud Resource Allocation and Power Management Using Deep Reinforcement Learning.

Manzoor, M., Abid A., Farooq, M., Nawaz, A., & Farooq, U. (2020). Resource Allocation Techniques in Cloud Computing: A Review and Future Directions. Elektronika Ir Elektrotechnika, 26(6), 1-10.

Naik, A., Sooda, K., & Rai, K. (2021). A study on Optimal Resource Allocation Policy in Cloud Environment. Turkish Journal of Computer and Mathematics Education, 12(14), pp. 5438-5446.

Qureshi, S., Qureshi, B., Fayaz, M., Zakarya, M., Aslam, S., & Shah, A. (2020). Time and Cost Efficient Cloud Resource Allocation for Real-Time Data-Intensive Smart Systems. Energies.

Praveenchandar, J., & Tamilarasi, A. (2022). Resource Allocation Using Phase Change Hyper Switching Algorithm in the Cloud Environment. Intelligent Automation & Soft Computing, 34(3).

Sahu, Y., & Agrawal, N. (2015). Scheduling Resources in Cloud using Threshold Values at Host and Data Center Level. International Journal of Computer Science and Information Technologies (IJCSIT), 6(6).

Shahapure, N., Rekha, P., & Poornima, N. (2021). Threshold Compare and Load Balancing Algorithm for Resource Optimization in a Green Cloud. Revista GEINTEC, 11(4), pp. 4465-4481.

Sultana, F., & Kawadkar, P. (2022). Task Scheduling and Resource Allocation Strategy by Using Thresholds with Attributes and Amount. International Journal of Creative Research Thoughts (IJCRT), 10(3), 1-12.

Swain, S.-R., Singh, A.-K., & Lee, C.-N. (2022). Efficient Resource Management in Cloud Environment.

Completion of Circle Material Problems Using the Jumping Task Approach to Junior High School Students

ORIGINALITY REPORT

21%	18%	11%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repo.uinsatu.ac.id Internet Source	3%
2	researchonline.gcu.ac.uk Internet Source	1%
3	mdpi-res.com Internet Source	1%
4	scholar.archive.org Internet Source	1%
5	www.slideshare.net Internet Source	1%
6	Submitted to Wilmington University Student Paper	1%
7	link.springer.com Internet Source	1%
8	Lin, Weiwei, Chaoyue Zhu, Jin Li, Bo Liu, and Huiqiong Lian. "Novel algorithms and equivalence optimisation for resource	1%

allocation in cloud computing", International Journal of Web and Grid Services, 2015.
Publication

9	aws.amazon.com Internet Source	1 %
10	jestec.taylors.edu.my Internet Source	1 %
11	www.cibgp.com Internet Source	1 %
12	www.techtarget.com Internet Source	1 %
13	Bo Wang, Changhai Wang, Ying Song, Jie Cao, Xiao Cui, Ling Zhang. "A survey and taxonomy on workload scheduling and resource provisioning in hybrid clouds", Cluster Computing, 2020 Publication	1 %
14	Submitted to Zikura International College Student Paper	1 %
15	Swapnil M Parikh. "A survey on cloud computing resource allocation techniques", 2013 Nirma University International Conference on Engineering (NUiCONE), 2013 Publication	1 %
16	www.pria.us Internet Source	1 %

17	Joy, Jimy, and L. Krishna Kumar. "Cost and deadline optimization along with resource allocation in cloud computing environment", 2013 International Conference on Advanced Computing and Communication Systems, 2013. Publication	<1 %
18	Yar Rouf, Joydeep Mukherjee, Marin Litoiu, Joe Wigglesworth, Radu Mateescu. "A Framework for Developing DevOps Operation Automation in Clouds using Components-off-the-Shelf", Proceedings of the ACM/SPEC International Conference on Performance Engineering, 2021 Publication	<1 %
19	ouci.dntb.gov.ua Internet Source	<1 %
20	cloudacademy.com Internet Source	<1 %
21	Submitted to Bayerische Julius-Maximilians-Universität Würzburg Student Paper	<1 %
22	arunj2ee.blogspot.com Internet Source	<1 %
23	www.businessvalueconsulting.com.ng Internet Source	<1 %

24	Submitted to VIT University Student Paper	<1 %
25	www.uin-suka.ac.id Internet Source	<1 %
26	S. Thamarai Selvi, C. Valliyammai. "Dynamic resource allocation with efficient power utilization in Cloud", 2014 Sixth International Conference on Advanced Computing (ICoAC), 2014 Publication	<1 %
27	psasir.upm.edu.my Internet Source	<1 %
28	Tushar Bhardwaj, Subhash Chander Sharma. "An autonomic resource provisioning framework for efficient data collection in cloudlet-enabled wireless body area networks: a fuzzy-based proactive approach", Soft Computing, 2018 Publication	<1 %
29	ebscoapac.stacksdiscovery.com Internet Source	<1 %
30	Yongbao Chen, Zhe Chen. "Short-term load forecasting for multiple buildings: A length sensitivity-based approach", Energy Reports, 2022 Publication	<1 %

31	dcclab.sookmyung.ac.kr Internet Source	<1 %
32	e-journal.iain-palangkaraya.ac.id Internet Source	<1 %
33	press.um.si Internet Source	<1 %
34	what-when-how.com Internet Source	<1 %
35	H Nakanishi, T Nishimoto, R Nakamura, A Yotsumoto, T Yoshida, S Shoji. "Studies on SiO2–SiO2 bonding with hydrofluoric acid. Room temperature and low stress bonding technique for MEMS", Sensors and Actuators A: Physical, 2000 Publication	<1 %
36	Sruthy Santhosh, A Binu. "Auto scaling for various patterns of workflow within deadline time and energy aware VM allocation in cloud environment", 2016 International Conference on Data Science and Engineering (ICDSE), 2016 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		