International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6110
***Article Received**: 14 December 2022 **Revised**: 14 January 2023 **Accepted**: 22 January 2023*

Novel Load Balancing Optimization Algorithm to Improve Quality-of-Service in Cloud Environment

Mr. Rupesh Mahajan[1], Dr. Purushottam R. Patil[2], Dr. Amol Potgantwar[3], Dr. P. R. Bhaladhare[4]
[1]Research Scholar,
School of Computer Sciences and Engineering, Sandip university,
Nashik, Maharashtra, India
e-mail: mhjn.rpsh@gmail.com
[2]Associate Professor,
School of Computer Sciences and Engineering, Sandip university,
Nashik, Maharashtra, India
e-mail: mhjn.rpsh@gmail.com
[3]Professor,
Sandip Institute of Technology and Research Centre,
Nashik, Maharashtra, India
e-mail: mhjn.rpsh@gmail.com
[4]Professor & Head,
School of Computer Sciences and Engineering, Sandip university,
Nashik, Maharashtra, India
e-mail: mhjn.rpsh@gmail.com

Abstract— Scheduling cloud resources calls for allocating cloud assets to cloud tasks. It is possible to improve scheduling outcomes by treating Quality of Service (QoS) factors as essential constraints. However, efficient scheduling calls for improved optimization of QoS parameters, and only a few resource scheduling algorithms in the available literature do so. The primary objective of this paper is to provide an effective method for deploying workloads to cloud infrastructure. To ensure that workloads are executed efficiently on available resources, a resource scheduling method based on particle swarm optimization was developed. The proposed method's performance has been measured in the cloud. The experimental results prove the efficiency of the proposed approach in reducing the aforementioned QoS parameters. Several metrics of algorithm performance are used to gauge how well the algorithm performs.

Keywords- load balancing, QoS, PSO, algorithm, cloud computing

I. INTRODUCTION

In cloud computing, clients have access to shared computing resources over the Internet using methods from parallel and distributed computing. As a result of the "pay as you go" pricing model, cloud computing is quickly approaching widespread accessibility. Participants in this stage of software deployment include cloud providers, service providers, and end users. Businesses may hire the computing power of the cloud, thanks to service providers (VMs). These virtual machines are used by service providers to provide application-level client services. Algorithms for scheduling tasks are used by service providers to distribute work from their clients among multiple virtual machines in order to speed up responses, guarantee a high level of service quality, and make the most efficient use of available resources [1]. Job scheduling algorithms are thus an essential component of any cloud infrastructure. The numerous scheduling methods currently used in different computing environments need to be modified to accommodate cloud computing. It's possible that a scheduling approach that works well on a cluster won't fare as well in the cloud. The steps of the procedure must be introduced into the issue space before the algorithm can handle the cloud environment's structure. The number of possible task configurations grows in proportion to the number of different virtual machines and the total amount of the workloads being managed. One of the hardest issues in computer science is determining the shortest path among all possible combinations. This work has developed a novel load-balancing variant of the original PSO approach for cloud scheduling, though metaheuristic algorithms have been used previously to aid cloud scheduling.

Our Contribution

- We need better load balancing to ensure that requests are spread evenly between machines. Significantly more time was saved because to better VM load balancing than was shown in earlier studies.
- An efficient fitness function that makes more efficient use of resources is necessary in order to keep both the

service provider and the customers happy.

- Improved PSO algorithm tested using Cloudsim by using varying load, and result shows algorithm improved various performance parameters.

Scheduling methods for tasks in the cloud have the potential to increase processing times and slow down the system as a whole. To this end, cloud computing seeks to optimise the utilisation of computing resources in a distributed network of devices so as to boost overall performance. Multiple methods of task scheduling are used in cloud computing, including ACO, PSO, and GA.

II. Load Balancing Algorithms

1. Ant Colony Optimization Technique: In this rule, the ant moves forward in a linear fashion whenever a call for participation is made, stopping at each node in turn to record whether or not the node is overloaded. As soon as the ant discovers a full node, it begins to crawl backward to the previous underneath-loaded node in order to transfer its newly acquired knowledge [1].

2. Honey Bee Foraging Algorithm: It's a method for balancing the load across the different nodes of the cloud that's inspired by the way nature does it. This formula determines if a node is overloaded [2], underloaded, or balanced by its initial current load. Priority-based job migration involves relocating a task from a node that is currently overburdened to one that is currently underutilized.

3. Biased Random Sampling Algorithm: This regulation envisions the network as a digital graph. When thinking of servers, think of them as nodes; the available resources at any given node correspond to their in-degree. The load balancer distributes jobs to the nodes based on the principle of in-degree. The in-degree is reduced when a job is assigned and increased when it is completed.

4. Resource Allocation Scheduling Algorithm (RASA): An initial step of this algorithm is to create a network of fictitious nodes. All of the virtual nodes' estimated response times are determined. If we look for the least occupied node as a metric, we can zero down on a highly effective virtual node. If the number of available resources is divisible by 3, then MinMin is used; otherwise, MaxMin.

III. Related Work

In order to boost performance in an HPC setting, the literature has extensively studied the load balancing problem. The dynamic mapping problem has been studied by experts from several fields, such as job scheduling and distributed systems.

In their study [3], YuAng Chen and Yeh-Ching Chung examined Cache-aware Reorder (Corder), a lightweight task reordering solution that makes use of the cache hierarchy of multi cores computers. It improves cache performance by using more nuanced approach to the vertex order in the local region and supports uniform distribution of computational loads across several cores on the shared-memory level. Corder's efficiency is thoroughly tested using a wide range of graph applications and datasets. Corder is only applicable to vertices that have been labelled by their outgoing edges, despite its remarkable benefits in portability across platforms and reordering overhead.

A CA Model, dynamic load balancing strategy for the concurrent execution of spatially-related applications has been proposed by Andrea Giordano et al. [4]. Through the use of a CA model developed for the express purpose of graphically displaying the development of the system and the consistent load distribution across the nodes throughout the execution, it demonstrates how the LB operates from a qualitative perspective. This study's parallel execution was achieved through a variety of parallel execution settings determined by starting MPI processes locally and across multiple nodes.

Applications can be dynamically optimised for time, energy, communications, or their combinations using the multi-Objective technique proposed by Alberto Cabrera et al. [5]. Using a flexible objective function that can evolve over time, it takes the approach that maximises resource utilisation in the present. Where workload is consistent between iterations, energy measures have been shown to be very helpful; where workload is inconsistent, they have been shown to be less so. It can be reduced using a multi-goal strategy.

For data-parallel clusters, Yinghao Yu et al. [6] created SP-Cache, a load-balanced, redundancy-free cluster caching system. To distribute the read requests for popular files more fairly over various servers, it divides them into multiple divisions based on their size and popularity. Short-term popularity swings, like spikes in access to specific files, have so far proven too much for SP-Cache to handle without resorting to frequent load balancing.

To improve load balancing performance, Mahdi Jafari Siavoshani et al. [7] presented and analysed a content delivery and caching scheme that uses a secret code that outperforms the nearest replica approach and the power of two choices baseline techniques. Even though it may not be computationally feasible in some practical scenarios, Design and Development of an Efficient Approach for Task Allocation in Distributed Systems using Heuristics Environment derives closed-form expressions for a grid network that achieves nearly perfect load balancing without sacrificing communication cost.

One load balancing strategy, called Computing load Aware and Long-View, has been examined by Guoxin Liu et al. [8]. If a server is overloaded at one point in time, CALV will select the blocks that provide the most work at that time, and the blocks that contribute the least work at that time will be selected instead. CALV employs a sluggish data block transfer technique to enhance the load balancing functionality. To avoid burdening the receiving servers too much, and to free up bandwidth during network peaks, it strategically schedules all data transfers. CALV is superior than competing approaches because it increases data locality and decreases task delay, network burden, and reallocation overhead.

Extreme workload variability is a common problem in large-scale multi-server distributed queuing systems, Jonatha Anselmi and Josu Doncel [9] have mentioned a class of size-based routing algorithms. It demonstrates that when all servers have the same processing speed, with infinitely increasing system size, the size-interval task assignment policy's mean waiting time converges., the average wait time reduced by a Size-Interval Task Allocation policy that distributes work fairly among all servers. However, when servers are heterogeneous, SITA-performance E's may suffer for no apparent reason.

When it comes to dynamically balancing computing workloads, a self-organized criticality strategy has been presented by Juan Luis Jimenez Laredo et al. [10]. It explains how the sandpile model might be modified to accommodate the challenge of coordinating several, unrelated activities. The sandpile's decentralized execution is self-organizing, thus it can adjust amount of available assets to the unique characteristics of incoming workloads. The system's emergent load-balancing response is studied to determine how best to balance these two goals—low energy consumption and high quality of service.

In their paper, Qiong Chen et al. [11] discussed task allocation for MTL scenarios in the frontier. It demonstrates that the problem of task allocation with task importance for MTL (TATIM) is a special case of the NP-complete Knapsack problem, requiring complex computations to be performed repeatedly in different settings. To accomplish this, we employ a Data-driven Cooperative Task Allocation (DCTA) strategy, which is shown to be 3.24 times faster than the state-of-the-art Design and Development of an Efficient Approach for Task Allocation in Distributed Systems with Heuristics Environment, and which provides a useful and effective mechanism for decreasing the necessary resource for MTL to be performed on the edge.

Soft real-time task fault-tolerant allocation (FTAOA) in WSNs was developed by Wenzhong Guo et al. [12] Using the principle of "earliest deadline first," FTAOA prioritizes the tasks in order of their respective priority levels, giving more precedence to those with earlier due dates. The job allocation problem is addressed, and a utilization function is developed, both of which are utilized to assess the overall performance of the nodes in this study.

TCHAP, proposed by Ashraf Suyyagh et al. [13], is a partitioning method that minimises energy consumption while still accounting for the inherent heterogeneity of the cluster and the inherent performance/energy variance of the underlying hardware. An example of a software-level strategy where the allocation of tasks to heterogeneous clusters affects overall system energy consumption is discussed; this strategy is called energy-efficient partitioning. The power consumption of the platforms is increased as a result of the coupling of the task-aware scheduling for energy-efficient partitioning on single-ISA heterogeneous platforms.

PathGraph is a path-centric technique to doing efficient iterative graph calculations on enormously large graphs, and it was recently presented by Pingpeng Yuan et al. [14]. Both the storage and compute levels exhibit the path-centric abstraction. The idea of a task queue with various "stealing points" that can be utilized to steal work is explored in this paper. It shows that for a range of iterative graph algorithms, PathGraph beats X-Stream and GraphChi on real-world graphs of varied sizes, achieving both superior balance and speedup.

Load balancing algorithms fall into one of three types depending on who started the process:

- Sender Initiated: If the load-balancing algorithm is started up by the sender
- Receiver Initiated: If the receiver starts the load-sharing routine
- Symmetric: It's a hybrid between "sent" and "received" messages.

Load balancing algorithms can be broken down into two groups, depending on the system's current configuration:

- Static: This is independent of the system's current condition. The system requires prior knowledge.
- Dynamic: The present state of the system is taken into account while making load balancing decisions. In other words, no prior knowledge is required. Therefore, it's preferable to a purely static method.

In a distributed system, load balancing can be performed either statically, before any tasks are executed, or dynamically, while the tasks are being executed. Static load balancing has the advantage of requiring less resources to complete due to its reliance on the system's typical behaviour when making load decisions. Policy decisions made by static load balancers don't change as needed to accommodate for variations in the amount of work being done. In contrast, dynamic load balancing makes an effort to balance loads in real time, taking into account the present state of the system and factors that are believed to enhance performance.

As a result of its adaptability, dynamic load balancing is well suited to applications with variable processing needs and irregular communication patterns. During execution, Dynamic Load-Balancing redistributes the workload across the computers such that each one is doing roughly the same amount of work as the others. With the goal of decreasing the reaction time of the task, dynamic load balancing shifts work from overworked computers to those with less on their plates through the use of a dedicated network.

Taking into account the loads as real numbers that can be split arbitrarily, numerous writers in the literature have presented iterative load balancing techniques. While jobs in fine-grained applications can be divided indefinitely, those in medium and coarse grains cannot. Even after the load balancing method has finished running, the resulting load distribution will not be globally balanced if it was developed for a discrete load model. The challenges faced by a dynamic load balancing method include determining when to initiate a load balancing operation, which computer makes the load balancing decision, collecting relevant data, and moving the load between computers. An involved computer will balance loads by taking into account the following four criteria:

- **Load Evaluation:** A computer's load index will be used as the foundation for locating a discrepancy in the system. Every machine's utilization was included in to the overall load when calculating the severity of the imbalance. Those with a low load index value indicate a lightly loaded computer, while those with a high load index value indicate an overworked machine. A few tasks from the overloaded computer were transferred to the lightly laden computer in order to even out the load index of the machines in the system.
- **Load balancing profitability determination:** When there is a discrepancy between the load on each computer, the load balancing mechanism shifts work from the busier machines to the less busy ones. The financial viability of load balancing is evaluated by considering factors such as communication lag and migration expense while allocating work to faraway computers.
- **Task migration strategy:** With the task migration approach, the overcrowded computer is selected by the load balancing algorithm to have its queued tasks removed and moved to the queue of a less busy computer. The communication overhead of load balancing is kept to a minimum by carefully selecting the source and destination computers.
- **Task selection strategy:** The source computer must exercise caution while choosing which jobs to offload. Load balancing should be taken into account while choosing which jobs to perform.

Distributed methods, on the other hand, require fewer computers, therefore they incur less overhead when making load balancing decisions. The load evaluation instant is the time at which the load index of each computer is measured. It is important to strike a balance between how often load data is collected and how old it is when using computerized load data. The following three conditions for collecting load information will prevent the load balancing algorithm from using out-of-date values:

- On-demand When a load balancing procedure is about to begin or is launched, computers exchange load information with one another.
- Load information is shared between computers on a regular basis. The other computers in the system may or may not find this data useful.
- On-state-change-driven When the amount of work being done on a computer reaches a certain threshold, the system notifies the other machines of the increased load.

On-demand collection of load information from computers in a system reduces communication overhead but can lengthen task processing times. In a bidding system, lightly burdened computers request load information from other computers in the system and then select the most suitable computer for balancing the load.

IV. Problem Statement and Description

The goal of any scheduling method should be to find the best method of distributing resources among tasks in order to boost as many variables as possible. Each user, R = {$user_1$, $user_2$, $user_3$ … $user_R$}, submits n requests, each of length l(£), to be carried out on a single physical host, M = PH_1, PH_2 , PH_3 … PH_M}, in a cloud datacenter. For a given time interval t, the total number of requests (TR) made by all users is equal to R*n. In order to meet the needs of its users by the deadline set by those users I each PH is made up of m virtual machines (q1, q2, q3,..., qm). Our goal is to optimize QoS parameters by determining how best to map each task/application to the available cloud resources. As shown in Figure 1, a controller node component maps all incoming application requests to available cloud resources based on resource requirement data like memory $_{MMmem}$, storage s_c, processing speed p_s, and CPU g, and then schedules the application to run on the best possible virtual machine (VM). Minimizing the time and cost of application execution simultaneously is a multi-objective scheduling problem due to their inherent incompatibility. Therefore, we've devised a novel method for solving multi-objective problems using the fitness function and a meta-heuristic algorithm.

Fitness Function

The purpose of a fitness function is to identify an optimal solution to a problem with multiple criteria. Here, we have considered two goals: The first goal is to shorten the amount of time it takes for applications to finish processing, and the second goal is to cut down on the amount of money spent on cloud resources during a given time period.

V. PROPOSED APPRAOCH

A. *Motivation*

Based on load balancer techniques that can be applied for task distribution to maintain the equal load on each processing element such that all processing elements becomes neither overloaded nor remains idle, we can achieve the benefits of resource pooling, openness, concurrency, scalability, fault tolerance, and transparency. Based on the literature review it is studied that there is a need to develop such a system. It is estimated that the proposed system will provide a generic framework. The problem statement is subdivided into the following objectives to achieve the research goal:

- We need better load balancing to ensure that requests are spread evenly between machines. Significantly more time was saved because to better VM load balancing than was shown in earlier studies.
- An efficient fitness function that makes more efficient use of resources is necessary in order to keep both the service provider and the customers happy.
- Improved PSO algorithm tested using Cloudsim by using varying load, and result shows algorithm improved various performance parameters.

B. *Novel PSO Model*

The scheduling of available resources is the backbone of any cloud computing RMS. It essentially denotes the process of assigning a cloud's workloads to the best resources available. Based on user needs, this procedure finds the most appropriate cloud workload and resources to run them. There are four stages to the scheduling process. Workloads are first categorized by their needs and then analyzed. The next stage involves selecting the necessary resources from the available pool. In the third stage, cloud workloads are mapped to suitable resources according to user-specified Quality of Service requirements. Finally, plan the use of available resources to carry out tasks, thereby doubling down on your assurance that all of your QoS needs will be met to an almost perfect degree. The proposed method satisfies the requirement for optimized resource scheduling in the cloud. Consider a grocery store: if a customer comes in looking to make a purchase, the salesperson there will first find out what kind of restrictions the customer has in terms of price range and other criteria before deciding what to put on display. Choose the right product from among those shown based on your budget, needs, and other considerations.

Resource Provisioning

Specifically, the following components make up the resource provisioning method:

1. Bulk of Workloads: The majority of pending tasks, known as BoW, have arrived and are currently being sorted and queued up for execution.

2. Workload Resource Manager: The Workload Resource Manager (WRM) keeps track of resource data, quality of service metrics, and service level agreements (SLAs) in order to provision resources for the execution of workloads based on the cloud consumer's described QoS requirements.

3. SLA Measure: With the help of the appropriate Service Level Agreement, WRM is able to obtain the necessary data (SLA). WRM verifies availability of resources after analysing and validating the many QoS constraints imposed by the workload.

4. QoS Metric Data: The document details the quality of service metrics that are used to assign relative importance when grouping tasks.

5. Workload Analyzer: Workload Analyzer's purpose is to examine various aspects of a cloud workload in order to ascertain whether or not it is possible to port the application to the cloud. In the cloud, QoS requirements and characteristics vary widely across workloads. The Quality of Service (QoS) requirements of all submitted workloads are analysed by WRM. To ensure quality of service is met, it is necessary to first determine which workload patterns are needed for workload clustering, and then to determine which metrics are needed to assign weights based on the level of measurement described in the QoS requirements specified in the SLA. The workloads are re-clustering for execution on a different set of resources using a K-Means based clustering algorithm.

6. Resource Information: Specs on how many processors are being used, how much memory is available, how much everything costs, what kinds of resources are available, and how many there are in total are all part of the resource breakdown. The resource pool is where we keep all of the shared resources.

Resource Provisioner

When the necessary resources are available in the resource pool, it makes them available to the workload so that it can be run in the cloud. The WRM will request that the workload be resubmitted with the need for an SLA-based Quality of Service guarantee fails if the requisite means are unavailable. The resource scheduler receives requests for work after resources have been provisioned. The resource scheduler will then request the workload be submitted in exchange for the allocated resources. After the resource scheduler sends the results back to the WRM, the cloud workload will already have the necessary resource information. Eighth, a Resource Planner All workloads will be run on the allocated resources. A cloud workload is a type of application submitted by a cloud user that abstracts the work that instance or set of instances would do if they had access to the necessary resources and could then execute. This research takes into account a wide range of workloads, including websites, technological computing, venture software, performance testing, online transaction processing, e-commerce, central financial services, storage and backup services, production applications, software/project development and testing, graphics oriented, critical internet applications, and mobile computing services.

Clustering of Workloads

Previous research work describes in detail the workloads in the Cloud are grouped together based on their shared characteristics and usage patterns. The end result of grouping work by similar patterns. Furthermore, the process of clustering using a K-means based clustering algorithm has been described in detail in previous research work, and is used to re-cluster the workloads before they are executed on a new set of resources.

Analysis

In the area of resource scheduling, we have developed a particle swarm optimization (PSO) based algorithm while taking into account a variety of quality of service (QoS) factors. The following are some key features that should be included in any effective algorithm for allocating resources: Efficiency QoS-based efficient resource management is essential for the cloud's cost-cutting resource provisioning feature. Minimizing Wasteful Use of Materials Less time and materials should be wasted thanks to well-planned scheduling. Waiting cloud workloads should be carried out in a manner that makes the most efficient use of available resources and optimizes quality of service (QoS) settings (execution time and cost). Equal Timetables Each user should receive the same total amount of resources in the cloud regardless of how many cloud workloads they submit. Capacity for Change and Expansion A resource-aware scheduler can change its behaviour in response to incoming or outgoing resources, allowing for more effective management of both resources and workloads.

VI. EXPERIMENTAL EVALUATION

A. *Platform and Tools Used*

- Operating systems starting with Windows 7, with Windows 10 being the preferred option.
- Since the Cloudsim simulation toolkit is a Java class library, the latest version of Java (JDK) should be installed on your machine; this can be done by downloading it from Oracle's Java portal. Oracle provides extensive documentation and installation instructions for those who need help setting up the software.
- For Java programmers, there is the Eclipse Integrated Development Environment. In accordance with the Linux/Windows distribution that you have currently installed. It is important to determine whether a 32-bit or 64-bit version is needed before initiating a download. The following link will take you to the Eclipse Kepler download page.
- Obtain the CloudSim source code; to date, many versions of CloudSim have been released; the most recent is 5.0, which is built on a container-based engine. To keep things simple for newcomers, we'll be installing the most popular version, 3.0.3, which you can get by clicking any of the following: Select Windows or Linux by clicking the appropriate button.
- The common jar package of mathematical functions is a third-party requirement of Cloudsim; you can get it from the Apache website or just click here to get it downloaded right now.
- Eclipse, Cloudsim, and the Common Math libraries should be unzipped into the same directory.

B. *Metrics of Evaluation*

- Availability (A): Mean Time to Repair (MTTR) is the ratio of the MTBF to the sum of the MTBFs of the individual components (MTTR).
- Reliability (re): Scheduling resources requires verifying their dependability. The fault tolerance of a resource can be evaluated using the reliability parameter.
- Resource Utilization (RU): It's the proportion of a resource's uptime to the time that its workload was actually executed.
- Latency (L): It is the disparity between the planned and actual duration of an operation's execution.

C. *Results and Discussion*

The proposed Novel PSO technique's performance has been evaluated in comparison to those of state-of-the-art existing scheduling algorithms. Multiple cloud workloads and resource

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6110
***Article Received**: 14 December 2022 **Revised**: 14 January 2023 **Accepted**: 22 January 2023*

counts have been used to evaluate the proposed PSO's performance. QoS parameters such as availability, reliability, latency, and resource utilization, as well as execution time, cost, energy, and other metrics have been used to gauge the efficacy of the proposed PSO. The time it takes to execute a workload is an indicator of its complexity, while its cost provides a basis for resource selection. The proposed PSO improves the efficiency of executing workloads. Executing workloads in the cloud is cheaper for consumers as a whole. When funding is increased, more personnel and tools can be made available to speed up the implementation process. The proposed novel PSO runs the with different number workloads with the highest possible uptime and reliability. To put it another way, the proposed novel PSO has a lower mean energy consumption than existing methods.

VII. CONCLUSION

In this paper, various technologies and issues related to resource allocation and scheduling has been studied and discussed. From the literature survey, it is found out that the there are several task distributions and sharing of resources techniques implemented, but still load balancing optimization needs further extensive research. The proposed novel PSO approach giving a promising solution to this problem. The primary objective of this paper is to provide an effective method for deploying workloads to cloud infrastructure with load balancing optimization. To ensure that workloads are executed efficiently on available resources, a resource scheduling method based on particle swarm optimization is proposed. The proposed method's performance has been measured in the cloud. The experimental results prove the efficiency of the proposed approach in reducing the aforementioned QoS parameters. Several metrics of algorithm performance are used to gauge how well the algorithm performs.

REFERENCES

[1] D. Babu, and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments", (*Applied Soft Computing, 2013*), pp. 292-2303.

[2] Misha Goyal, and MehakA ggarwal, "Optimize Workflow Scheduling UsingHybrid Ant Colony Optimization (ACO) & Particle Swarm Optimization (PSO) Algorithm in Cloud Environment", *International Journal of Advance research, Ideas and Innovations in Technology*, 2017.

[3] YuAng Chen and Yeh-Ching Chung, "Workload Balancing via Graph Reordering on Multicore Systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 33, No. 5, May 2022.

[4] Andrea Giordano, Alessio De Rango, Rocco Rongo, Donato D'Ambrosio, and William Spataro, "Dynamic Load Balancing in Parallel Execution of Cellular Automata", IEEE Transactions on Parallel and Distributed Systems, Vol. 32, No. 2, February 2021.

[5] Alberto Cabrera, Alejandro Acosta, Francisco Almeida, and Vicente Blanco, "A Dynamic Multi–Objective Approach for Dynamic Load Balancing in Heterogeneous Systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 31, No. 10, October 2020.

[6] YinghaoYu , Wei Wang , Renfei Huang , Jun Zhang , and Khaled Ben Letaief, "Achieving Load-Balanced, Redundancy-Free Cluster Caching with Selective Partition", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 2, February 2020.

[7] Mahdi Jafari Siavoshani , Farzad Parvaresh , Ali Pourmiri , and Seyed Pooya Shariatpanahi, "Coded Load Balancing in Cache Networks", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 2, February 2020.

[8] Guoxin Liu, Haiying Shen, and Haoyu Wang, "Towards Long-View Computing Load Balancing in Cluster Storage Systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 28, No. 6, June 2017.

[9] Jonatha Anselmi and Josu Doncel, "Asymptotically Optimal Size-Interval Task Assignments", IEEE Transactions on Parallel and Distributed Systems, Vol. 30, No. 11, November 2019.

[10] Juan Luis Jimenez Laredo, Frederic Guinand, Damien Olivier, and Pascal Bouvry, "Load Balancing at the Edge of Chaos: How Self-Organized Criticality Can Lead to Energy-Efficient Computing", IEEE Transactions on Parallel and Distributed Systems, Vol. 28, No. 2, February 2017.

[11] Qiong Chen, Zimu Zheng, Chuang Hu, Dan Wang, and Fangming Liu, "On-Edge Multi-Task Transfer Learning: Model and Practice with Data-Driven Task Allocation", IEEE Transactions On Parallel And Distributed Systems, Vol. 31, No. 6, June 2020.

[12] Wenzhong Guo, Jie Li, Guolong Chen, Yuzhen Niu, and Chengyu Chen, "A PSO-Optimized Real-Time Fault-Tolerant Task Allocation Algorithm in Wireless Sensor Networks", IEEE Transactions On Parallel And Distributed Systems, Vol. 26, No. 12, December 2015.

[13] Ashraf Suyyagh and Zeljko Zilic, "Energy and Task-Aware Partitioning on Single-ISA Clustered Heterogeneous Processors", IEEE Transactions on Parallel and Distributed Systems, Vol. 31, No. 2, February 2020.

[14] Pingpeng Yuan, Changfeng Xie, Ling Liu, and Hai Jin, "PathGraph: A Path Centric Graph Processing System", IEEE Transactions on Parallel and Distributed Systems, Vol. 27, No. 10, October 2016.

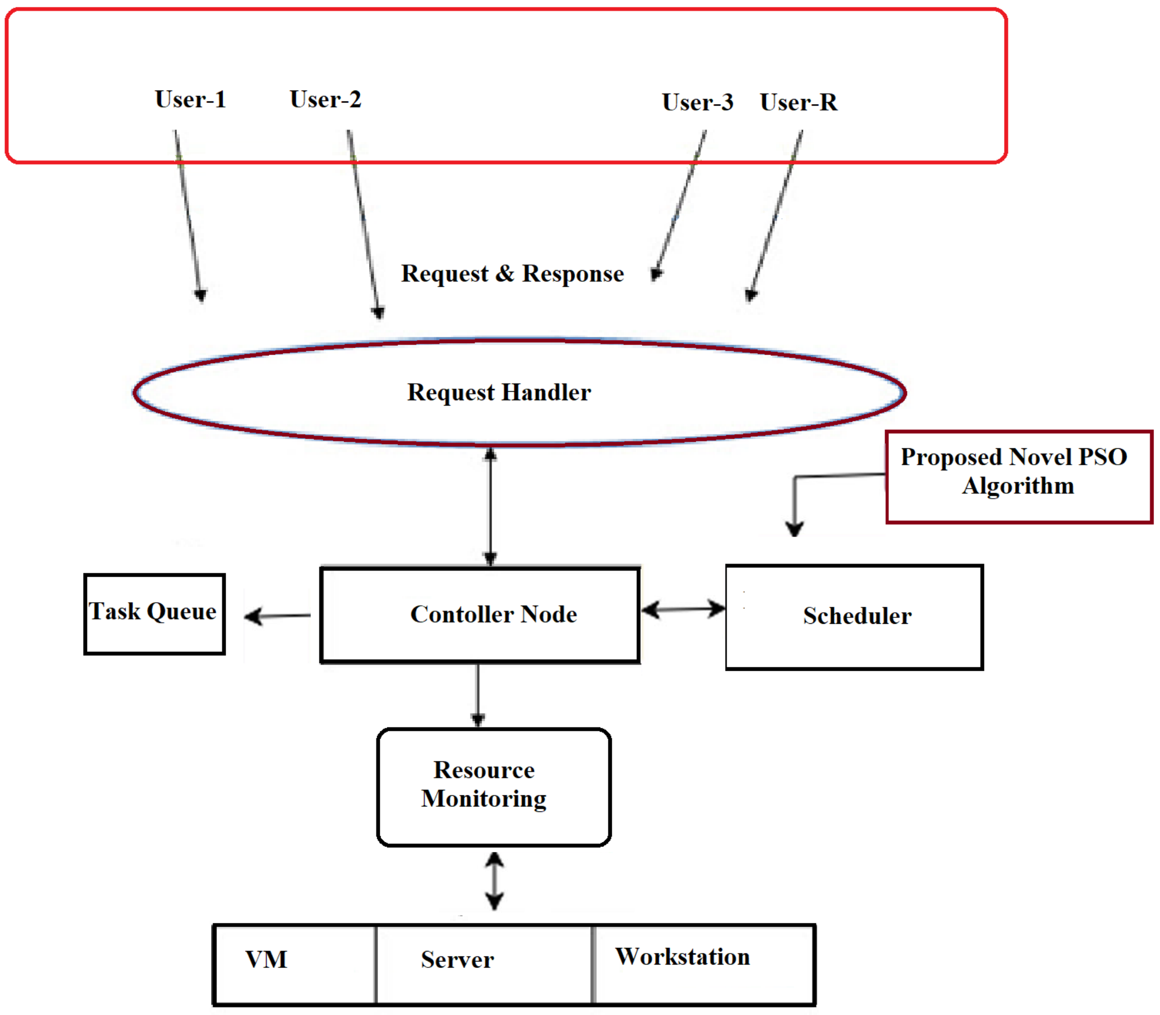

Figure 1. Scheduling tasks using a cloud computing-based processing model

INTERNATIONAL • IJRITCC • COMMUNICATION