Efficient Resource Allocation in Cloud Computing Using Hungarian Optimization in Aws

Juliet A Murali ( julietamurali@gmail.com )
Noorul Islam College of Engineering: Noorul Islam Centre For Higher Education

Brindha T
Noorul Islam University: Noorul Islam Centre For Higher Education

Research Article

Keywords: Cloud Computing, Resource Allocation, Make Span, Hungarian Optimization, AWS, Amazon EC2, Lambda Function

Posted Date: February 8th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2543829/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License
Abstract

Cloud computing is a recent technology which allows on-demand availability of computing services as well as resources to users without having direct control by the user. The various resources offered by cloud technology needs appropriate task scheduling strategies to provide better Quality of Service (QoS). Moreover, the Cloud Service Provider (CSP) concentrates more on providing various resources to a user based on demand. Thus, resource allocation occupies a crucial task in offering better QoS and profits to CSP. Although several techniques were deployed for various resource allocations in the cloud, the challenges still risk the CSP in accomplishing improvised resource allocation due to over demand and under availability of resources which results in increased makespan as well as Virtual Machine (VM) utilization factor. For this reason, this paper aims to introduce a novel resource allocation model that comprises appropriate resource discovery, task scheduling, and then resource allocation. The proposed methodology uses the EC2 Lambda instance for the resource discovery process. Depending on the EC2 Lambda instance, the recommended process is grouped to the available resources. Resource discovery is done by finding all the resources that meet the user's needs and guiding them to nearby innovation. Initially, the executed jobs are recognized or analyzed and the available resources are grouped. Simultaneously the resources are allocated with the help of a Hungarian optimization technique. Eventually, the empirical analysis reveals the performance of the proposed resource allocation model in terms of CPU and network utilization.

I. Introduction

Generally, Cloud Computing (CC) can be explained as computer hardware and software resources that can be provided based on on-demand availability by the service providers for the end-users [9][11]. It is comprised of storage, software, platform, computational power, as well as networks. Huge cloud servers frequently provide operations dispersed over several places and all places have a data center for communication [7]. To put it simply, CC is a technology that provides numerous services via the internet. Moreover, the cloud enables a user can access all the computing services virtually even from remote places through the internet [10][15]. Due to the increased advantages with the implementation of cloud systems, there is a huge demand for the growth and development of cloud technology day by day. Moreover, it provides several computing services which simplifies and minimizes the effort and cost of implementation of both software and hardware required to complete numerous computer-based applications [8][12][13]. The various benefits include minimized data losses since hardware failure does not affect the user as it has data backups [16]. Furthermore, the cloud enables users to establish their applications to the market easily and minimizes the equipment cost as it utilizes remote devices [14][17].

For instance, Google Drive is a cloud-based technology that allows users to store their files online and enables them to access the files anywhere through the internet using any electronic gadgets like smartphones, laptops, or computers [19]. Besides, CC offers several computing services among the three services that are majorly used such as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), as well as Infrastructure-as-a-Service (Iaas) [21][22]. Though the cloud provides various resources and
services, managing and allocating the resources to the users based on demand and availability is a challenging task [24][25]. Moreover, a lot of issues arise in allocating resources in CC such as over demand and under availability, security and privacy, reliability, scalability, and maintenance as well as updates. Alloting resources in the cloud needs to overcome the afore-mentioned challenges and needs to enable secure as well as optimized resource allocation [18][23][28].

Previously, several methodologies were implemented to overcome the issues and challenges in cloud resource allocation [20]. Yet increase in the usage of the cloud results in a necessity to develop efficient techniques to handle cloud systems [1]. For this purpose, the efficacy of Artificial Intelligence (AI) is deployed. It includes Machine Learning (ML) models such as several optimizations and Deep Learning (DL) algorithms [5]. The optimization models give better resource allocation results, accurate task scheduling, and resource management. Moreover, the performance, as well as accountability provided by optimization models were incredible [11]. However, for several applications, adopting a suitable optimization model is a challenging task. Besides, computational complexities arise when handling highly latency-sensitive cloud applications [13][15]. Fast computation is another challenge as optimization models need several iterations to find the best solution. Also, implementing an appropriate algorithm for the defined problem needs a deep understanding regarding the ML algorithms as well as the problems and necessity of implementing cloud services [21][23].

For this purpose, several types of research and studies are going on concerning proficient resource allocation models for CC systems [17][28]. In this context, an efficient resource allocation strategy is introduced to handle the above-mentioned issues and challenges in cloud systems. The major contribution of the paper is as follows.

- The proposed resource allocation strategy includes two basic features as resource discovery as well as allocation.
- Resource discovery is done by the EC2 Lambda function which groups the available resources in the cloud. Besides, resource discovery is performed by finding each resource that meets the user’s demands and guiding them to nearby innovation.
- At first, the jobs which are supposed to be computed are recognized/analyzed and grouped the available resources via the implemented EC2 lambda instances.
- Instantaneously, the efficiency of the Hungarian optimization technique is utilized to allocate the resources.
- At last, the experimental results exhibit the efficiency of the proposed resource allocation model through CPU and network utilization.

The rest of this paper is assembled as follows. A detailed literature about various resource allocation strategies in CC is described in Section II. Section II provides the proposed architecture and the objectives. Moreover, Section IV represents the methodologies implemented in the proposed work. The empirical results and the performance of the proposed work are deliberated in Section V. Eventually, Section VI concludes the paper.
ii. Literature Review

A. Related Works

In 2020, Guang Zheng et al. [1] have addressed a resource allocation technique in 5G cloud service that used dynamic optimization technique based on heuristics for allocating the virtual resources. Moreover, a cooperative Q learning paradigm was employed to assign the cloud resources. Here, the cloud system is comprised of a cloud access level, dispersed micro-clouds, as well as a data center. From the experimental conclusions, it was clear that the dynamic optimization technique achieved minimized learning time as well as better throughput.

In 2020, Mahdi Abbasi et al. [2] has designed a cloud resource allocation model which implemented XCS Learning Classifier Systems (LCS) called XCS and BCM-XCS for managing the power utilization as well as minimizes the computational delays. Furthermore, the BCM-XCS technique disperses the workloads in an efficient manner which helped to minimize the delays at both computation and communication of the nodes. The empirical evaluation proved the proficiency through minimized delay and power utilization and outperformed the conventional XCS model.

In 2020, Xiang Wu et al. [3] have developed a resource allocation for cloud systems using Natural Language Processing (NLP), as well as Gray Relational Analysis (GRA) strategies for computing as well as minimizing the feature attributes. Besides, the Bayesian model was employed to classify the chosen applications which maximize the VM's CPU. Further, the efficiency of VM resources such as CPU and memory load was predicted using the ANFIS system. The efficacy of the implemented model was verified through the experimental analysis which enhanced the VM performance as well as resource allocation.

In 2021, Fatemeh Ebadifard, and SeyedMortezaBabamir [4] have presented two issues and their solutions in cloud resource allocation through automated as well as end-to-end multi-objective resource sharing models. In addition to this, the implemented model accomplished enriched profits, minimized network latency, as well as data transmission traffic loads. Finally, the experiments were conducted using the CloudSim platform which attained improvements concerning success, revenue, as well as computational time.

In 2021, Ning Ai et al. [5] have proposed a resource allocation strategy through Cuckoo Search Algorithm (CSA) to accomplish better QoS. Initially, the resource shortage was eliminated using multi-agent CSA and the QoS was assured to the users by assigning appropriate resources by optimally allocating the resources. The experimental performance demonstrated the efficiency of the CSA model in terms of maximized energy as well as improved adaptability.

In 2020, JingLi [6] have established a resource allocation model in a 5G cloud environment using Hierarchical Distributed Cloud Service Network (HDCSN) technique to properly perform task scheduling and there by attaining enhanced resource allocation. Moreover, a hierarchical resource vector, along with a graph were developed for creating a resource portrayal to enhance the IaaS cloud system. Through the
simulation analysis, it was evident that the implemented model attained better resource scheduling criteria and minimized the response time.

B. Review

The various features and challenges associated with the above-mentioned works were described in this section. The dynamic optimization technique [1] attained enhanced throughput and minimized learning time. Yet, training the cloud services for multiple dimensions was complex. The BCM-XCS [2] accomplished reduced delay and power utilization but it was restricted by the implementation cost. The NLP and GRA [3] model obtained improved VM performance and resource allocation. However, it was limited due to computational complexities in training the Bayesian model regarding a large VM environment. The automated and end-to-end multi-objective resource sharing [4] model achieved minimized network latency, as well as data transmission traffic loads still, the risks such as processing end-to-end resources in the cloud is time-consuming for large networks. The multi-agent CSA [5] assured better QoS and task scheduling but facing issues due to high computational time as it required several iterations to find the best solution. Although HDCSN [6] model guaranteed better resource scheduling criteria and minimized the response time, it was bounded due to complications in depicting exact resource graphs. Thus, these challenges motivate the researchers to concentrate and develop innovative ideas regarding efficient resource allocation strategies for CC systems.

Iii. A Novel Cloud Resource Allocation Model

A. Proposed Architecture

Fig. 1 demonstrates the architecture of the proposed resource allocation model using Amazon Elastic Compute Cloud (Amazon EC2) Lambda instances in Amazon Web Services (AWS). Generally, Amazon EC2 can be defined as a web service that offers adjustable, secured as well as reliable computing facilities via the cloud. The main benefit of this service is to help the users for creating web-scale CC services. Moreover, it is an interface that permits the users for accomplishing as well as configuring more facilities with reduced friction. It offers complete control over the computing resources as well as allows the users to compute the required services. Here, the proposed resource allocation model is implemented with two major phases. Initially, it begins with resource discovery and task scheduling via Amazon EC2 instances. Then, the AWS Lambda is employed for appropriate functioning of the phase one processes. Secondly, the discovered and scheduled resources are allocated using the Hungarian optimization process. Furthermore, depending on the Amazon EC2 instance, the recommended method is grouped to the available resources. Resource discovery is done by finding all the resources that meet the user’s needs and guiding them to nearby innovation. Instantaneously, the resources are allocated with the help of a Hungarian optimization technique. Additionally, Fig. 2 delivers the flowchart of the proposed resource allocation model.

B. Objective Function
Since resource allocation is necessary to accomplish efficient cloud systems. This paper aims to achieve minimized computational time, cost, and overall energy utilization as well as attaining enhanced resource utilization. For this process, the resource allocation is designed as a multi-objective problem which is solved with the implementation of the effectiveness of the Amazon EC2 and Amazon Lambda function. Here, the resource utilization is evaluated by calculating the overall computational time as well as waiting time as given in Eq. (1) where, $RU$ is the resource utilization factor, $N$ indicates the number of tasks, $i$ points out the individual task, $CT_i$ is the computational time of each task, and $w_i$ specifies the waiting time of each task.

$$RU = \sum_{i=1}^{N} \frac{CT_i}{w_i}$$

Eq. (1)

Here, Eq. (2) gives the formula for computing the overall energy consumption where, $M$ is the total time, $j$ indicates the execution time of the individual task, $\Gamma$ is the overall task, and $E$ is the energy required for each task. The computational time required to execute each task is estimated based on Eq. (3).

$$ec = \sum_{i=1}^{N} \sum_{j=1}^{M} \Gamma_{ij} * E_{ij}$$

Eq. (2)

in which, $T_{ij}$ is the total time required to compute each task. The overall cost required to compute the above-mentioned task scheduling is estimated as portrayed in Eq. (4).

$$TC_{ij} = \sum_{i=1}^{N} \sum_{j=1}^{M} T_{ij} * \Gamma_{ij} * E_{ij}$$

Eq. (3)

Now, the multi-objective problem is computed as stated in Eq. (5). In order to define the fitness function for Hungarian optimization algorithm, the multi-objective problem is computed by adding penalty to the parameters which violates certain constraints. Finally, the penalties and the parameters are summed to get the single objective as stated in Eq. (6), where $p_i$ is the penalty for each parameter.

$$RA = \max (RU) \text{ and } \min (ec + CT_{IJ} + TC_{ij})$$

Eq. (5)
Iv. Materials And Methodology Of Proposed Resource Allocation Model

A. Amazon EC2

In the era of computers, CC offers various computing services which drastically enhance the user experiences and betterment of QoS. By AWS, users can accomplish simplified as well as improvised CC experiences. Specifically, Amazon EC2 provides the biggest as well as significant computing platform having several processors, networking, Operating systems (OS), storage, etc. Moreover, Amazon EC2 provides high-speed processors having 400 Gbps Ethernet capacity. Also, it offers cost-effective services and instances for ML and GPU functionalities. The major advantages of having Amazon EC2 include

1. Reliability, scalability, and infrastructure

2. High-speed operating services enable maximize or minimize the capability within minutes.

3. High Service Level Agreement (SLA) commitment.

4. High-level availability of applications, resources, and virtual services.

In addition to this, Amazon EC2 enables flexible services to the users so that the user can indicate the week application or resource which can be rebooted, or terminated by the EC2 instances to avoid unnecessary delays. It schedules uncommon activities like rebooting of the system, termination of a process to the EC2 instances as soon as it finds any irretrievable failure in the host. Using this flexibility provided by EC2, the users can configure the Scheduled Events like rebooting or terminating the insignificant resources. Moreover, it offers the users’ instance IDs/tags through the EC2 console together with the associate event windows. Using this window, the inadvertent maintenance as well as furthered scheduled events can be removed outside of the scheduled event windows. These scheduled events feature can be used by every EC2 instance involved in the computing process.

All the resources created and used can be viewed and managed with the EC2 instances. The available resources are discovered and are further subjected to scheduling are handled by the EC2 instances, along with the flexibility provided by the EC2 instances, and the scheduling features, phase 1 including both resource discovery and task scheduling is efficiently managed.

B. Amazon Lambda

\[ \text{obj} = \sum_{i=1}^{N} p_i \times RA \]
After performing phase 1, the scheduled task is needed to be allocated to the requirements. For this purpose, AWS Lambda is employed which can be defined as a serverless computing service that enables the users to execute code deprived of partitioning or controlling the servers, forming workload-aware cluster scaling logic, sustaining event combinations, and so on. Furthermore, this function allows the user to execute programs virtually. After uploading the program, the Amazon Lambda automatically as well as accurately allots the computing services such as computational time, power and executes the program depending on the user demands. The various benefits of Amazon Lambda include

- There is no need for servers for controlling and managing the resources.
- High-level flexibility i.e., continuous scaling.
- Optimal cost due to millisecond (ms) metering.
- Reliable performance at any scale.

AWS Lambda inevitably executes the program without necessitating the user for provisioning/controlling the infrastructure. The user can just create the program, upload it to Lambda and compute the result. Besides, the continuous scaling feature of Lambda automatically adjusts the resource or application through executing the program in correspondence with all events. The user code can be executed parallel as well as the scaling accurately based on the size of the application, from limited requests to numerous requests per second. In addition to this, cost-effectiveness can be achieved through Lambda, as it enables the user needs to pay only for the resources consumed. This feature eliminates the unnecessary payment for excess infrastructure which enhances the savings by up to 17%.

Through the Lambda function, the excess storage space can be eliminated which also avoids under-storage issues as it permits the user to select the appropriate memory size required for the code to execute. By having Provisioned Concurrency feature, it enables high-speed response for every request.

a. Project Work Process

The following steps given below explains the configuration of AWS Lambda.

1. Form a traditional AWS Identity and Access Management (IAM) strategy as well as the computational rules for the Lambda function.

2. Make Lambda function which can automatically begin and terminate the EC2 instances.

3. Make Cloud Watch Events policy which activates the function on a schedule.

With these procedures, the Lambda function creates and schedules the task efficiently. After performing both the resource discovery and task scheduling processes, the created and required resources are allocated using the Hungarian optimization model which is discussed in the next section.
C. Hungarian Optimization

Generally, the Hungarian optimization model [26][27] is used to resolve complex assignment problems. Besides, it is a combinatorial optimization model which perfectly handles multi-objective problems. Hither, $N$ tasks, and $S$ resources have been scheduled and allocated. The aim is to allocate the requested resources to the tasks to complete the job. For optimal allocation of the resources, every job should be completed with reduced total inputs as well as allocate each job with required resources. For every pair of $N$ tasks and $S$ resources, it should be made clear that every task has its required resources. The mathematical model of this problem is defined as explained in Eq. (7).

\[
\text{cost} = \{C_{ij}\}_{N \times S}
\]

Now, Eq. (7) gives the cost matrix function, where $C_{ij}$ indicates the cost of resources $i$ to complete $j$ tasks. This process involves row and column reduction which helps to determine the minimal element in the matrix of resources. After finding the minimal element, subtract the minimal element with all other elements in the matrix. Now, the matrix contains some zeros (at least one zero at every row and one zero at every column, and this arrangement is called 0-percolation). If it is a 0-percolation matrix with redundancy, perform resolvability test to eliminate redundancy. Now, the resolvability test will provide a flag value. If the flag is 0, then the resulting matrix has no redundant values else repeat the process till it reaches the flag value as 0. To perform the resolvability test, the shaker process is significant, as it reduces the redundancy in the matrix using covering segments. Covering segments are nothing but the positions of the horizontal as well as vertical elements in the matrix. Subsequently, the resultant binary matrix is accomplished based on Eq. (8) by satisfying the criteria $B_{ij} = 1$. This criterion can be satisfied only if the $i^{th}$ the resource is assigned to $j^{th}$ task.

\[
BM = \{B_{ij}\}_{N \times S}
\]

Eq. (9) explains the one resource to one task allocation, where $RT$ is the resource to task allocation. On the other hand, the one task to one resource allocation is performed based on the Eq. (10), where $TR$ is the task to resource allocation.

\[
RT = \sum_{j=1}^{N} B_{ij} = 1, \forall i \in 1, N
\]

\[
TR = \sum_{i=1}^{N} B_{ij} = 1, \forall j \in 1, N
\]
Finally, the cost function is achieved as defined in Eq. (11).

\[ CF = \sum_{i=1}^{N} \sum_{j=1}^{S} C_{ij}B_{ij} \rightarrow \min \]

With Hungarian optimization, all the resources are allocated to the tasks effectively and the resource utilization is enhanced with minimized cost and energy.

**V. Simulation Results**

A. Simulation Setup

The proposed resource allocation model using Amazon EC2 and Amazon Lambda with Hungarian optimization approach was implemented in AWS on Intel core® core i5 processor 8th generation, 8 GB RAM, 64-bit operating system. The efficiency and novelty of the implemented model were recorded via the simulation results. Here, the evaluation was implemented through various performance parameters such as CPU utilization, network, CPU credit usage, and network packets utilization factor using different instances.

B. Performance Analysis

In this section, the performance analysis of the proposed resource allocation model using AWS with Hungarian optimization is presented. Furthermore, the performance of the proposed model is portrayed in Fig. 3 through CPU utilization. Here, it shows the CPU utilization percentage of various instances in AWS Lambda. In Fig. 4, the network utilization of multiple resources is demonstrated. Moreover, the resources are allocated using minimized network utilization with high-speed functionality. Figure 5 depicts the CPU credit usages. In Fig. 5(a), the number of CPU credits used by the resources is indicated. The measures micro specifies minimized computational time and small is a little bit higher than micro periods. Besides, AWS Lambda enables the resources to be executed rapidly, and parallelly with minimizes time. Figure 5(b) shows the initialization of the instances and their attributes indicating, the name of the instances, region, period/execution time, and the statistics showing minimum value, maximum value, average, and a sum of the instances created with Amazon EC2. Subsequently, Fig. 5(c) deliberates the credits required for every instance individually. Finally, Fig. 6 portrays the number of network packets each resource utilized during the entire process. Thus, Amazon EC2 and AWS Lambda provide significant features for creating and computing cloud services. Moreover, it exhibits excellent performance in task scheduling and allocating resources in terms of CPU and network utilization.

**Vi. Conclusion**
This paper has addressed a novel resource allocation scheme for cloud systems using the efficiency of the implemented Hungarian optimization model. It includes appropriate resource discovery, task scheduling, and resource allocation processes. For resource discovery, the Amazon EC2 model was employed. Further, the available resources were grouped based on the recommendation of the Lambda function. Resource discovery is done by finding all the resources that meet the user’s needs and guiding them to nearby innovation. At first, Amazon EC2 instances were used to recognize or examine the computed jobs and the available resources are grouped. At the same time, the resources were allocated with the help of a Hungarian optimization technique. Eventually, the empirical analysis reveals the performance of the proposed resource allocation model in terms of CPU and network utilization. Together with the efficiency of Hungarian optimization, the resources are allocated appropriately. Thus, the Amazon EC2 provides a better platform for creating and utilizing cloud resources. Besides, AWS Lambda enhances the user experience and QoS by offering high-level services.

Declarations

Funding: Not applicable.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is not available on any public platform.

Acknowledgments: The authors extend their thanks and appreciation to the Guest Editor for the invitation to submit this work to the Special Issue and to the Associate Editor and anonymous reviewers whose constructive comments and suggestions helped to considerably improve the quality of this paper during the review process.

Conflicts of Interest: The authors declare no conflict of interest.

References


29. Authors biography

**Figures**
Figure 1

Block Diagram of Proposed Resource Allocation Model
Figure 2

Flow chart of Proposed Resource Allocation Model using Amazon EC2 and Amazon Lambda
Figure 3
Performance of Proposed Resource Allocation Model through CPU Utilization

Figure 4
Performance of Proposed Resource Allocation Model through Network Utilization
Figure 5

Performance of Proposed Resource Allocation Model in terms of CPU Credit Usage
Figure 6

Performance of Proposed Resource Allocation Model through Network Packets

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- biography.docx