Journal of Computer Science

Original Research Paper

Rationalizing Resource Utilization in Cloud Computing Using Coalition Formation Strategy

[1]Hend Fakhri Noureldin and [2]Mai Fadel

[1]Center of Excellence in Genomic Medicine Research, Bioinformatics Unit, King Abdulaziz University, Jeddah, Saudi Arabia
[2]Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

Article history
Received: 15-03-2021
Revised: 19-04-2021
Accepted: 27-05-2021

Corresponding Author:
Mai Fadel
Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
Email: mfadel@kau.edu.sa

Abstract: Even though the contribution of cloud computing towards the Sustainable Development (SD) of communities is still under research investigation, cloud computing has become an integral part of many ICT solutions that shape our daily lives. Thus, some researchers recommend taking considerable actions to point cloud computing development towards supporting SD. In this research, an approach to designing energy efficient cloud architecture as a way of supporting SD is proposed. Resource allocation is a challenging process in cloud management, the goal is to allocate the exact amount of resources needed throughout the service duration; tight enough to avoid unnecessarily wasting resources and loose enough to prevent any degradation in Quality of Service (QoS) that may lead to the violation of the Service Level Agreement (SLA) between the service provider and the cloud user. This study aims to achieve the desired balance by benefiting from the history of the user's behaviour and from sharing resources – more specifically Virtual Machines (VM) – among a coalition of users. Coalition formation strategy is used to build groups of cloud users based on their cloud behaviour history. Users are grouped in a way that their usage patterns complement each other, either to avoid the loss stemming from VM excess reserved space or from idle times. A type of architecture that fulfils this improvement process is proposed and implemented on Google Compute Engine (GCE). The contribution of this research is that it applies the Coalition formation strategy in cloud computing resource management in a novel way and experiments show that there are scenarios where the efficiency of resource management has improved. Evaluation of the performance of the proposed architecture is done by comparing resource utilization for both the cloud following this architecture and the cloud that runs the basic GCE strategy. In conclusion, it is observed that improvements depend on accuracy of the prediction of usage pattern of the user. Results show that in certain scenarios, improvements can be made to up to 24% of VM usage and, in other scenarios, it can minimize the number of required VMs, thus contributing to green computing.

Keywords: Sustainable Development, Cloud Computing Middleware, Resource Utilization, Service Level Agreement, Resource Utilization, Virtual Machine, Coalition Formation, IaaS, Resource Allocation

Introduction

ICT solutions has become an integral part of people's daily living, actually there is a line of research called Activities of Daily Living (ADL) covering many topics such as the one found in (Thakur and Han, 2021). Cloud computing is part of ICT solutions as it provides users with computing power and applications delivered via the Internet. It shifts much of the provisioning of applications, configuration and maintenance to the responsibility of cloud providers rather than cloud users.

Efficient computing strategies are provided by centralizing storage, memory, processing and bandwidth to form the cloud. Cloud architecture is composed of

three layers: Software as a Service SaaS, Platform as a Service PaaS and Infrastructure as a Service IaaS (Sunyaev, 2020). In addition, the computer software, middleware, operates in different parts within this architecture. This research focuses on middleware on the IaaS level which is responsible for managing resources, such as memory and Central Processing Unit (CPU) among other functionalities.

Resource management allocates resources to user requests and ensures they are suitably matched by respecting the QoS terms specified in the Service Level Agreement (SLA); a process called Resource Allocation (RA). Resource management is responsible for keeping track of these terms and reserving additional resources to avoid violating SLA's terms. Management also releases unused resources to cut unnecessary costs. As cloud computing embraces the business model of pay-as-you-use, this poses additional challenges. For example, RA can hinder efficiency if it does not consider changes in the available environment, the users' requests, or network traffic. Consequently, RA should be dynamically configured to achieve resource optimization. Resource management should consider how to avoid common pitfalls such as presence of deadlock, starvation, significant response time, that may lead to violating the terms of SLA.

Resource allocation has been extensively studied by researchers as reported in several surveys (Dewangan *et al*., 2020; Saidi *et al*., 2019; Barán and López-Pires, 2017), some of the techniques used are auction-based, optimization-based and Autonomic-based techniques. Other techniques focus on monitoring or improving certain aspect of the resource allocation process such as SLA-base, QoS-based and cost-based techniques (Dewangan *et al*., 2020). Clustering techniques such as k-means clustering are also used (Barán and López-Pires, 2017).

In this research, an architecture of middleware that is based on coalition formation strategy, to improve the efficiency of the RA process is proposed. The rationale behind this idea is that forming groups of users will assist in the specification of more accurate resource sizes when reserving resources. Additionally, these groups are derived from analysing the characteristics of the history of users' requests. Furthermore, allocation of resources will change periodically based on predicted incoming requests, thus satisfying the dynamic requirement described above. The contributions of this study are as follows:

- Applying coalition formation strategy that is based on predicting users' behaviour to improve the efficiency of the resource allocation process
- Identifying scenarios where the strategy provided improvement up to 24% of VM usage and also being able to minimize the number of required VMs, thus contributing to green computing

The new architecture is implemented with Google Compute Engine (GCE). The effect of implementing this new strategy within both the execution time and resource utilization is explored. Synthesized workload will be used to test the effectiveness of the proposed middleware.

The paper is organized in the following sections. After the introduction will follow the next section which presents the literature review, followed by section III that describes the rationale behind the stated solution. In Section iv, the proposed coalition formation-based resources management architecture is explained. A description of our algorithm is given in Section V and evaluation of our proposed algorithm and experiment results are detailed in section VI. Findings and limitations are discussed in section VII. Finally, section VIII concludes this work.

Literature Review

Researchers have done extensive work regarding improving resource allocation in cloud computing. In this section, resource allocation research is organized on the following topics: Workload profiling and characterization, improving resource utilization, resource usage prediction and coalition formation.

Some researchers have focused on workload characterizing and analysis as they provide useful insights into cloud performance (Singh and Chana, 2014; 2015; Moreno *et al*., 2014; Ravi *et al*., 2018). Such information supports cloud providers in making decisions concerning daily operations and enables them to benefit from it when making resource usage predictions.

Observations of these researchers and the related findings are reported here. Singh and Chana (2014) have identified types of workload such as web apps, online transaction processing etc. In their study, they clustered them based on workload patterns and identified their QoS requirements. These researchers then continued with the analysis further and presented a shorter list consisting of: Compute, storage, communication and administration clusters (Singh and Chana, 2015). During investigation, Moreno *et al.* (2014) conducted an analysis of large cloud workload trace logs, in order to identify and quantify the diversity of behavioural patterns for users and tasks. Additionally, they identified and later validated model parameters and their values for simulation purposes. The researchers then emphasized that realistic workload models must include parameters describing user behaviour and their links to tasks. Part of their findings is that user behaviour varies a lot and therefore it is recommended that predictions should better depend on recent historical data.

Some researchers have pointed out certain mismanagement in dealing with workload characteristics and subsequently designed RA algorithms avoiding the perceived problems (Wei *et al*., 2015; Singh and Chana, 2015; Dezhabad *et al*., 2019; Deng *et al*., 2013). In their study, Wei *et al*. (2015) emphasized that even though

resources are provisioned based on the assumption that workload is homogenous, in fact, they are not. They therefore designed and evaluated a RA algorithm that corrects this assumption and takes into consideration that the algorithm does not result in skewness among the multi-resources. In their evaluation, Wei *et al*. considered 3 types of synthesized workloads: Growing, pulse and curve (resembles a bell shape to some extent), that can describe the shape of the incoming requests with respect to time.

Dezhabad *et al*. (2019) gave another example of heterogeneity where they described the workload of Alibaba Cluster Dataset, as containing both online services and batch jobs. The former requires a short response time and quick recovery from failure whereas the latter requires intensive processing. Batch instance workload is characterized into low, medium and high profiles according to the following measures: CPU usage, memory usage and job duration. In addition, the researchers' algorithm makes use of the arrival pattern for each type of requests in time.

Deng *et al*. (2013) addressed the problems that arise from executing scientific applications by using portfolio scheduling. Prolonged execution of such applications results in varying application requirements and increases the need to handle requests dynamically. Their algorithm should select the best policy for workload execution based on an abstract model introduced by their research. Part of the workload uses measurements related to job wait time, job runtime and the number of processors requested by a job; however, no description of the workload used for evaluation was found.

Significantly, resource allocation has been thoroughly investigated in previous research. In this study, some of the work focusing on improving resource utilization has been reviewed.

The goal of the research done by Maurer *et al*. (2013) is to confirm to SLA while allocating resources by considering the two issues involving efficient use of resources and improved resource utilization, thus ensuring that the process is done with as little human intervention as possible. In their investigation, they made use of autonomic control to govern the cloud infrastructure. In addition, these researchers introduced the concept that the gradual scaling-up of the level of action to be taken to efficiently allocates resources, such as when reconfiguring VMs, migrating applications, migrating VMs, etc. The results have shown that experiments that applied the rule-based approach yield better performance, with respect to the number of violations as well as utilization and time.

Gong *et al*. (2019) also used Control Theory (CT) for RA. They deal with uncertainty and unexpected workload by using Multiple Input Multiple Output type of CT to ensure that any influence between different resource types is embodied within their RA system. In their study, the researchers make use of Generalized Prediction Control (GPC) that provides feedback correction to make the workload prediction part of their algorithm adaptable. Additionally, Chen (2018) addressed the situation of emergency and sudden demand of workload, in which users/service providers are aware of the urgency of their presence and the subsequent need to take appropriate action.

During their probe into the subject matter, Bi *et al*. (2015) explained that uniformly treating all kinds of workloads, results in a waste of resources as some workloads require intensive computing power, whereas, other workloads have a high demand for storage. Their research aims to maximize the profit of service providers by meeting SLA requirements. These researchers benefit interested parties in their definitions of two levels of services - Gold services and Silver services –when dealing with different resource-intensive workloads. To clarify, achieving this fine-grained resource provisioning requires defining it as an optimization problem and using a probabilistic model to determine workload request arrival rates.

In a research done by Xiao *et al*. (2012), special attention has been given to multiplexing virtual resources into the physical hardware. In their study, they introduced the concept of skewness, which is used to measure the unevenness of the consumption of a resource type on a server with respect to other types of resources, the goal being to minimize skewness to improve resource utilization. In addition, they also designed their algorithm to dynamically allocate virtual resources to adapt to the heterogeneity of workload. Significantly, the work of Xiao *et al*. predicts the future demands of VMs based on past statistics.

The topic of resource usage and workload prediction has gained the attention of the scientific community as seen in the interest taken by Amiri and Mohammad-Khanli (2017 and Vashistha and Verma (2020) in their reports. The main goal of both of these studies was to improve the accuracy of prediction. Similarly, an earlier work done by Islam *et al*. (2012) addressed the problem of the delay caused by initializing a new virtual instance in a cloud by developing prediction-based resource management. They built two models; one using Linear Regression (LR) and the other using Neural Networks (NN); they also used Sliding Window to improve the accuracy of the system. Data gathering were conducted using TPC-W workload generator mimicking a number of simultaneous user sessions for an e-commerce website. The experimental results show that the Neural Network model produced more accurate results while the sliding window increased the accuracy. In the analysis, mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE) were used as evaluation metrics.

The work done by Kaur *et al*. (2019) focused on running scientific applications while using Genetic

Algorithm (GA) to select features in order to minimize search space. They explained that GA is a greedy algorithm that is designed with the goal of selecting the best input to produce the best output, thus, forming a good candidate for performing the task of feature selection. In their analytic study, they ensembled eight regression models to improve the accuracy of the results and gathered experimentation data from running a scientific application called Cybershake. It has been thus concluded that, in comparison to machine learning regression models, their model outperforms them, enhancing accuracy by 2% and reducing execution time by 16.2%. The model has also been seen to perform well with respect to error rate compared to the learning automata-based ensemble approach.

Kholidy (2020) study, a new Swarm Intelligence Based Prediction Approach (SIBPA) was developed for resource needs prediction. His research illustrates use of Multiple Support Vector Regression (MSVR) and ARIMA for non-linear and linear feature selection, respectively and integrates the Particle Swarm Optimization (PSO) and the Kernel Adatron (KA) algorithms to enhance the prediction of ARIMA and MSVR models by estimating their parameters. In the investigative study, experiments were done by conducting 150 user requests on a bookstore online system via the TPC-W emulator. The results show that the accuracy of SIBPA has outperformed LR, NN and Support Vector Machines approaches in terms of CPU utilization, response time and throughput memory utilization.

The idea of the system presented by the analytic research in this study is based on forming coalitions among cloud users. The strategy of coalition formation is applied in cloud computing to support cloud federations. As cloud federation enables a group of cloud providers to cooperate and dynamically share their resources, coalition formation in this context is used as one means to deal with an elastic demand of resources in the environment, as described in the work done by Hadjres *et al*. (2018). Bairagi *et al*. (2016) extended the concept further to allow for a cloud provider to participate in more than one coalition instead of being limited to a single one, to overcome the problem of resource under-utilization. Li *et al*. (2019), cloud federation is used to support hosting of the fifth Generation (5G) of networks. The authors have made use of Bayesian Coalition Formation Game, to address the problem of insufficient amount of information that is needed by the coalition of cloud providers.

Coalition formation can be viewed as being as simple as the formulation of groups. From the point of view of forming groups of cloud users, the work of Alsadie *et al.* (2017; 2018) use group clustering to support the process of RA. However, they differ slightly in that they form clusters of users' tasks. Also, they allocate resources to a group of tasks that are similar although each task is allocated within an individual VM. In summary, research shows that RA can be improved by considering workload characterization and using prediction. Further findings are that multiplexing CPU-intensive and Memory-intensive workloads improve resource utilization and user behaviour should be considered during workload modelling. Additionally, RA algorithms are seen to use arrival request rates, CPU utilization and memory utilization as parameters. Finally, even though coalition formation strategy is used, to our knowledge, no work has yet used coalition formation to form groups of users within the same cloud, or even among users of different clouds.

Rationale Underlying Our Solution

Optimizing resource management stems from allocating VMs to a group of users (coalition) that has been defined previously based on their behavioural patterns. VM utilization is targeted on two levels: The level of an individual user and the level of coalitions of users. In both levels, predicting users' incoming requests is based on analysis of historical data of their behaviour.

Optimization on the user level is achieved by computing the average size of the most dominant feature of its request, such as its CPU size, thus, ultimately reserving a virtual machine that can deal with the derived size of the requested CPU. Consequently, resource optimization becomes higher whenever the predicted size of VM gets closer to the actual request of the user.

It is noted that forming coalitions helps in optimizing the use of virtual machines in two ways: One that achieves space optimization (parallel users' mode) and the other achieves time optimization (interchangeable users' mode).

Regarding parallel users' mode, cloud service providers offer a fixed template size of Virtual Machines (VM), whereas user requests may not fit these templates. In this case, users either over-reserve VM capabilities, thus paying extra for unused resources, or under-reserve VM capabilities and suffer from delay in accomplishing the task due to scheduling purposes or boosting a new VM. It is noted that there may be benefit gained from grouping users so that the total of their requests matches one of the offered VM's templates, thus, achieving space optimization.

On the other hand, Interchangeable user's mode allows for the more preferable use of a static VM that works all day rather than the need to boost a VM with every request, thus providing benefit from grouping users that utilize the machine interchangeably. For example, users that work at night may be grouped with other users who work during the day. To exemplify, living in different time-zones may lead to such compatible needs. Consequently, the machine ends up working all day, thus, achieving time optimization. In addition, users do not need to reserve an external disk to save their data if they are using the same machine – static VM - contrary to using dynamic VMs when they get terminated (GCE, 2014).

Architecture of the Proposed Resource Management Tool

The resource management tool proposed in this research uses coalition formation strategy. Its architecture is composed of the following four modules, as shown in Fig. 1.

Classifying user behaviour module: This module is responsible for analysing the behaviour of each cloud user to classify the user as High-CPU, High-Memory or Standard consumer. This classification, referred here as *User Class* (UC), is based on determining the most dominant QoS feature specified in user requests during specified interval. In this research the focus is on two features of QoS, which are CPU and memory.

Forming user coalitions module: Coalitions are formed by this module. Members of each coalition are selected based on the determined UCs and whether the coalition is anticipated to achieve space optimization or time optimization as described in Section III. In addition, some users can remain alone and do not become part of a coalition as explained in section V. The size of coalitions can vary between 2 and 5 inclusively. The rationale underlying this choice is described in Section VI.

VMs creation module: VMs are created periodically by this module. Each machine is reserved to handle the requests of a certain coalition or to handle the requests of an individual user that does not belong to a certain coalition. Related security concerns are discussed in section V.

Handling of requests module: User requests are forwarded - as they arrive - to the VM allocated to the user or to its coalition. This module is also responsible for recording the cost of each request made by a user based on the billing policy of the cloud service provider and for computing their total cost for a specific time interval.

It is important to highlight that in VMs creation and request handling modules, processing is done online, in the sense that it is done when user requests are received.

For the other two modules, processing is done offline and can be conducted on a separate machine that is not linked to the virtual machines directly.

Proposed Algorithm

A description of the proposed algorithm is given in this section. The description, divided into the offline part and online part of the algorithm, is shown in Fig. 2 and 3, respectively:

a. Offline part of the algorithm: As described in the previous section, analysis of user behaviour and the determining of coalitions are done offline. As can be seen in Fig. 2, the algorithm starts by determining the UC for each user, creating combinations of all possible coalitions, then goes on to conducting a series of checks to filter the combinations into a final set of coalitions and individual users. Details of the checks will be visited shortly
b. Online part of the algorithm: Figure 3 shows this part of the algorithm, depicting the steps needed to form coalitions based on the analysis of users' behaviour. It determines all possible combinations of coalitions then excludes coalitions that have the same users. The algorithm starts with the creation of VMs for the final set of coalitions and individual users, then it is ready to receive users' requests. When a request arrives, it is treated differently depending on whether the requesting user is new or not. Requests of existing users are forwarded to reserved VMs. However, a new VM is created in two cases: Whether the user is new, or if the size of the request is higher than originally predicted, hence, the size of the reserved VM is not enough to handle the current request.

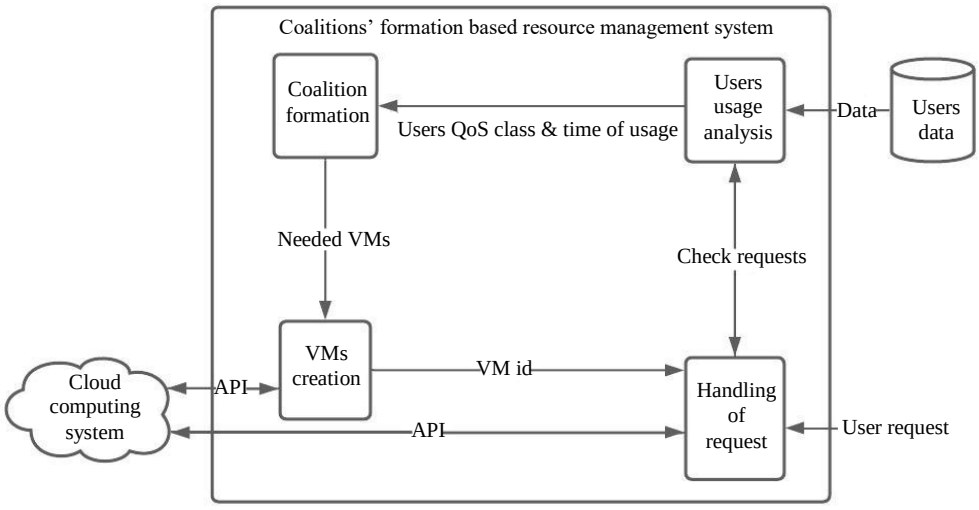

Fig. 1: Architecture of the resource management tool

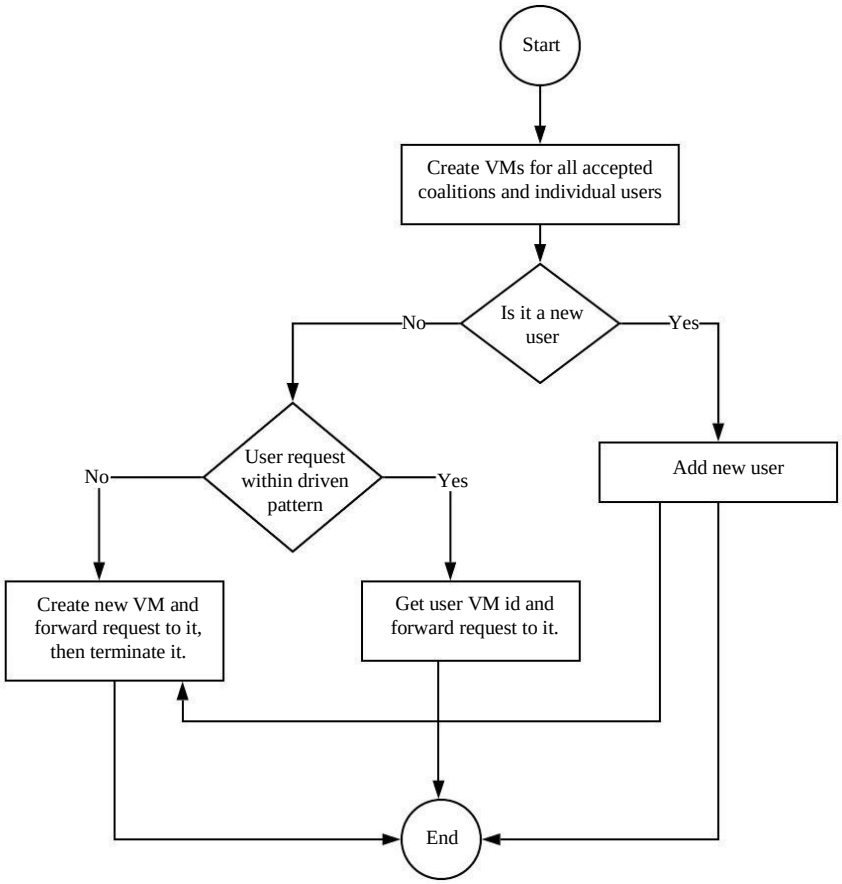

Fig. 2: The online part of the algorithm

These extra VMs are considered temporary and are terminated when they finish fulfilling the requests. Revisiting the step of VM creation, it is important to highlight that combining the requests of different users (of the same coalition) within the same VM provide some flexibility but, at the same time, may cause security concerns. Allocating a certain VM capacity to a group of requests, gives room for drifting of request sizes from original prediction, for example having one request larger than expected, while having the other request smaller than expected. In this research, security concerns are not addressed as this is a subject for additional research. However, some insights are given in how to deal with it. From one aspect, a constrain may be imposed to limit the degree of user security to be low, i.e., they use the cloud to perform routine computations only.

Another technical solution is to use containers (Sunyaev, 2020). Another issue to highlight is that consideration has been given to whether the request matches the size of a predefined template or not, even though there is the opportunity to define customized sizes of VM. However, a preferred action of the current researchers is to stick to the template configurations and exploit the advantages of coalitions, because template sizes are more likely to yield gaps that are more likely to fit the needs of other requests.

Next, details of two of the steps of the algorithm are presented as user's usage analysis and coalition formation:

A. User's usage Analysis Step:

1. For each hour in a day, specify usage type according to frequency usage and determine amount of CPU and Memory used for each hour. Usage type can be standard, high memory, or high CPU
2. Then give this hour a score based on its type (1 for high CPU, 2 for high memory and 3 for standard)
3. After that, compute user power score, which is the summation of the day, with the hour score and divide it by number of hours (24) for a day. Use the formula:

$$user\ power\ score = \frac{summation\ of\ hours\ score}{24} \quad (1)$$

To determine UC (standard, high memory, or high CPU), the type of the VM to create and reserve can be evaluated as follows:

1. If the user power score is between (0,1) then the user is high CPU
2. If the user power score is between [1,2) then the user is high memory
3. Elseif it >= 2, then it refers to standard user
4. Then divide user day to slot of time (such as interval of 30 min)
    1. For each time slot of enumerate value 1 indicating the user is running or 0 indicating it is not running during this time slot

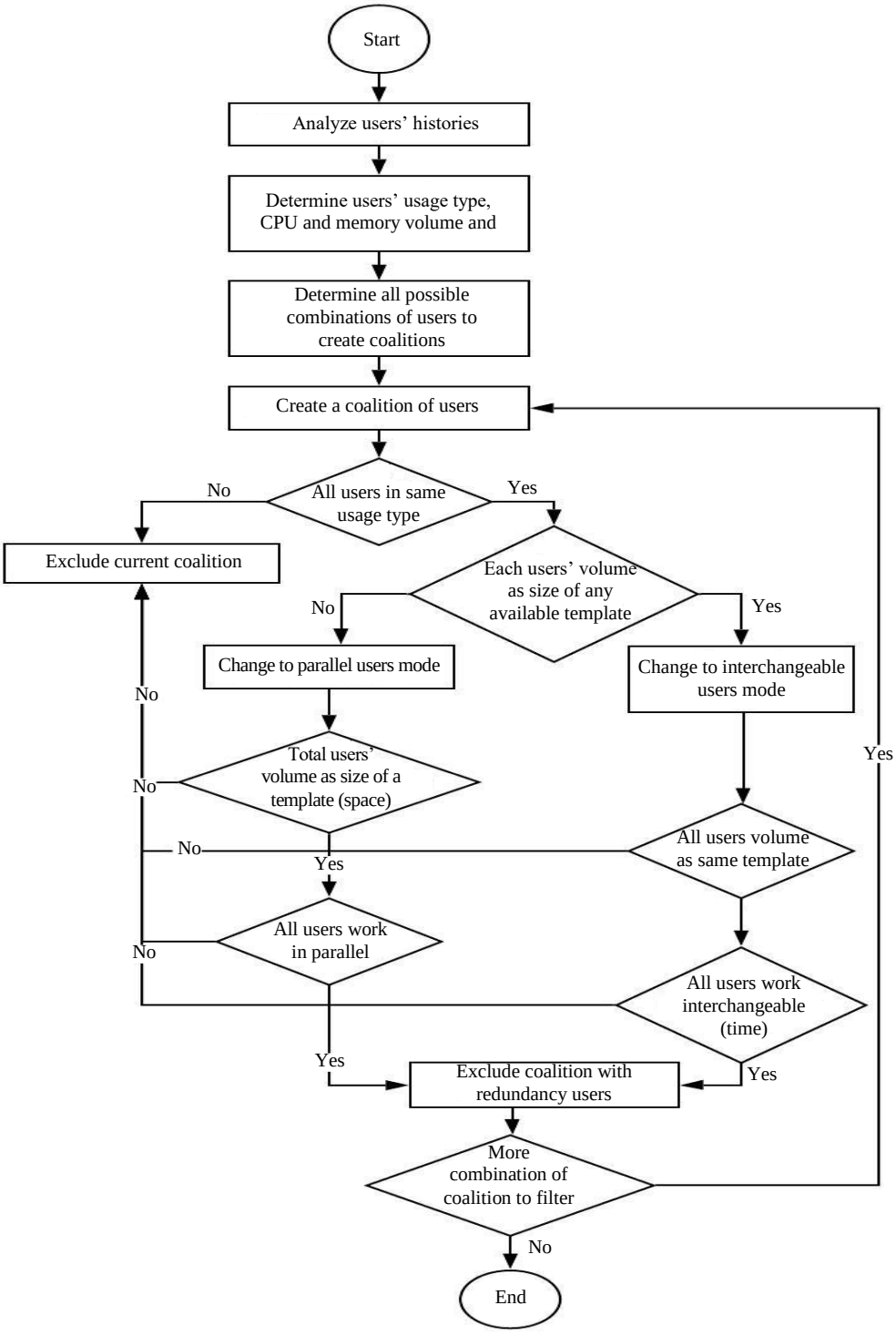

Fig. 3: The offline part of the Algorithm

B. Coalition Formation Step

It is important to note that GCE offers VMs according to certain template sizes. VM types reflect the same classes used for users. i.e., High CPU, High Memory and Standard. Based on the determined UCs, coalitions are computed as follows (see Algorithm 2 for more details):

1 Create all combinations of users based on a specific size
2 Exclude any combination that includes users having different UCs and using different images (operating system)
3 Exclude any combination that includes users that use different images
4 If the request of each member of the coalition is NOT equal to template size, then exclude the combination in the following cases:

   a. The total size of predicted requests of all members exceeds the size of the reserved VM. The comparison is done against the largest feature of the VM type, CPU, or memory
   b. Users are not working at the same time (in parallel). Each user needs to be running 80% of the time, for the coalition to be included

5 Else, the request of each member of the coalition is *equal* to template size, then exclude the combinations in the following cases:

   a. Any user predicted request size does not match the size of the template
   b. Users are working at the same time (users must work interchangeably)

6 Eliminate redundancy by excluding any combinations containing users that are already members of other registered coalitions
7 After forming all coalitions, determine all remaining individual users. For each individual user, determine their VM type according to their UC. They may be included in a coalition when new users register and use the cloud

Ultimately, the algorithm will generate a list of registered coalitions along with their members and VM type, in addition to the list of all users along with VM type and coalition number (if any).

Algorithm 1: userUsageAnalysis(users)

Input: users with their data
Output: determine users type and used volume of CPU and Memory

```
for j = 1 to numOfUser // for each user
    for i = 1 to 24 // for each hour
        if (CPUfreqUsage[i] >MEMfreqUsage[i]) && (CPUfreqUsage[i] >STRDfreqUsage[i])
hourType[i]= CPU;
        else if (MEMfreqUsage[i] > CPUfreqUsage[i]) && (MEMfreqUsage>STRDfreqUsage)
hourType[i] = MEM;
        else
hourType[i] = STRD;
    end for
    for i = 1 to 24 // for each hour
        if (hourType[i] == CPU)
            hourScore[i] = 1;
        else if (hourType[i] == MEM)
            hourScore[i] = 2;
        else
            hourScore[i] = 3;
    end for
        sumHourScore = 0;
    for i = 1 to 24 // for each hour
        sumHourScore += hourScore[i];
    end for
    userPowerScore [j] = sumHourScore/24;
end for
for i = 0 to numOfUser//foreach users
    if (userPowerScore[i] > 0) && (userPowerScore[i]< 1)
userType [i] = CPU;
ComputeCPU_MEM_volume(); // determine the volume based on frequency used volume.
    else if (userPowerScore[i]>=1) && (userPowerScore[i]<2)
userType[i] = MEM;
ComputeCPU_MEM_volume();
    else
userType[i] = STRD;
ComputeCPU_MEM_volume();
// user's day is divided into slot of time interval (such as 30 min)
 for each slot
    If user run
        slot[i] = 1
    else
        slot [i] = 0
    end for
 end for
```

Algorithm 2: CoaliteUser(Users)

Input: users with their behavior and usage amount of CPU and memory
Output: coalitions of users based on their behavior; each coalition has VMid and each user has VMid.

```
1.  cobinationsArray = CreateCombinations(size);
    // Create all combinations of users based on
    specific size; each coalition contain the required
    volume for it
2.  for i = 1 to cobinationsArray.length // for each
    coalition
3.      if (all users in same type && use same image)
4.      cobinationsArray[i].indicator = true;
         // include
5.      else
6.       cobinationsArray[i].indicator = false;
         // exclude
7.      if (!all user usage same as template)
8.          if (cobinationsArray[i].size<=largestSize)
                // determine combination VM volume
9.              if (all user work parallel)
10.             cobinationsArray[i].indicator = true;
                // include
11.             else
12.             cobinationsArray[i].indicator = false;
                 // exclude
13.         else
14.         cobinationsArray[i].indicator = false;
            // exclude
15.     else // as template
16.        if (all user in cobinationsArray[i] are
            same size)
17.          if (all user work in different time)
18.           cobinationsArray[i].indicator = true;
              // include
19.           else
20.           cobinationsArray[i].indicator = false;
              // exclude
21.       else
22.       cobinationsArray[i].indicator = false;
          // exclude
23. end for
24. for i = 1 to cobinationsArray.length // for each
    coalition remove redundancy
25.     if (All user are not inserted collation before)
26.     cobinationsArray[i].indicator = true;
          // include
27.     else
28.     cobinationsArray[i].indicator = false;
        // exclude
29. end for
30. for i = 1 to numOfUser
    // for each user check if he does not belong to
      a coalition
31.     if (user is not inserted to coalition)
32. User.VMRealType=user.getVMGeneralType();
33. end for
```

Evaluation and Experimental Results

Google Compute Engine GCE is used to implement and evaluate the proposed coalition formation-based resource management tool. In this section, evaluation of three aspects of the algorithm is presented. First, the evaluation of user-level optimization is described, followed by the systematic approach to determine coalition size. After that, an evaluation of the coalition formation strategy is presented.

A. Identify user Behaviour (User Level Optimization)

The goal of this experiment is to show that making accurate predictions of users' types of requests improves the utilization of the VM, as a way to validate the designed User's Usage Analysis. In this experiment, prediction accuracy is measured. GCE has been used to create 210 users to identify their behaviours. Each user makes 200 requests during the interval of one day and these requests are equally divided. One hundred randomly picked requests have been used to determine the UC (representing the historical data) and the other 100 requests (representing new incoming requests that need to be predicted) are used to evaluate the accuracy of the prediction.

Figure 4 shows the result of applying User's Usage Analysis as described in Section V. The columns denote the user number, UC, number of cores, size of memory and the image type. From the first row it can be seen that u1 user is classified as high-CPU, whereas u14 in row 14, is classified as High-Memory. Using this classification for prediction - i.e., checking the predicted request against the received one it is noticed that the accuracy ranges between 90 and 95% and the average hit rate for all users is 92.086%. Figure 5 shows the number of hit rates for individual users. The average accuracy of the prediction remains high as long as the user does not change their behaviour. More sophisticated prediction algorithms can be used in future experiments.

B. Selecting Coalition Size

One of the most important design decisions related to the coalition formation algorithm is to determine the size of the coalition; to determine the number of coalition members and whether the size is fixed or variable. Several experiments have been conducted in this study where the coalition size is fixed, i.e., all coalitions have the same size. Another set of experiments have also been undertaken where the coalition size is variable, i.e., where coalitions can be of different sizes, for example, a variable-size policy having a coalition of size 4 may include coalitions of sizes 4, 3 and 2. Regardless of the chosen policy, the goal of the conducted experiments is to decide the size of the coalition that will improve the performance of the system. Two conditions have been considered as indicators of a good candidate coalition size. The first condition is generating the minimum number of individual users that do not belong to a coalition the number must also be less than 50% of the

number of users. The second condition is to generate the minimum number of coalitions, in order to minimize the number of VMs to be created. Again, the experiments were conducted with the same set of 210 users, used in the previous section. Table 1 shows the number of individual users and number of coalitions when a fixed-coalition size policy is used and Table 2 shows the results of when the variable-coalition size policy is used.

From the experiments, it turns out found that, as the size of the coalition increases, the number of individual users increases as well. i.e., the first condition is not met. As can be seen in Table 1, having 3 members within a fixed-size coalition system gives the minimum number of coalitions with relatively few individual users. However, it can be seen from Table 2, having 4 members or less per coalition gives better results for the variable-coalition-size system. Also, during this investigation, it was found that the variable-coalition-size system generated better results in comparison to the fixed-coalition-size system. The number of individual users of the best system in Table 2 is 73, whereas it is 81 in Table 1. It is also noticed from Table 2, that even though the size 4 gives a smaller number of individual users than the coalition size 5, it gives a higher number of coalitions. After conducting several additional experiments where the number of users vary from 30 to 210, it was found that the size 4 gives better results. Consequently, the decision made is to choose the policy of variable-coalition-size and 4 members or less per coalition.

```
u1, Gtype  n1-highcpu CPU 2 Mem 1.8 image centos-6-v20140415
u2, Gtype  n1-highcpu CPU 3 Mem 2.7 image centos-6-v20140415
u3, Gtype  n1-highcpu CPU 3 Mem 2.7 image centos-6-v20140415
u4, Gtype  n1-highcpu CPU 3 Mem 2.7 image centos-6-v20140415
u5, Gtype  n1-highcpu CPU 2 Mem 1.8 image centos-6-v20140415
u6, Gtype  n1-highcpu CPU 3 Mem 2.7 image centos-6-v20140415
u7, Gtype  n1-highcpu CPU 1 Mem 0.9 image centos-6-v20140415
u8, Gtype  n1-highcpu CPU 1 Mem 0.9 image centos-6-v20140415
u9, Gtype  n1-highcpu CPU 1 Mem 0.9 image centos-6-v20140415
u10, Gtype  n1-highcpu CPU 2 Mem 1.8 image centos-6-v20140415
u11, Gtype  n1-highmem CPU 2 Mem 13.0 image centos-6-v20140415
u12, Gtype  n1-highmem CPU 3 Mem 19.5 image centos-6-v20140415
u13, Gtype  n1-highmem CPU 3 Mem 19.5 image centos-6-v20140415
u14, Gtype  n1-highmem CPU 8 Mem 52.0 image centos-6-v20140415
u15, Gtype  n1-highmem CPU 1 Mem 6.5 image centos-6-v20140415
u16, Gtype  n1-highmem CPU 3 Mem 19.5 image centos-6-v20140415
u17, Gtype  n1-highmem CPU 2 Mem 13.0 image centos-6-v20140415
u18, Gtype  n1-highmem CPU 2 Mem 13.0 image centos-6-v20140415
u19, Gtype  n1-highmem CPU 2 Mem 13.0 image centos-6-v20140415
u20, Gtype  n1-highmem CPU 2 Mem 13.0 image centos-6-v20140415
u21, Gtype  n1-standard CPU 1 Mem 3.75 image centos-6-v20140415
u22, Gtype  n1-standard CPU 1 Mem 3.75 image centos-6-v20140415
u23, Gtype  n1-standard CPU 2 Mem 7.5 image centos-6-v20140415
u24, Gtype  n1-standard CPU 1 Mem 3.75 image centos-6-v20140415
u25, Gtype  n1-standard CPU 3 Mem 11.25 image centos-6-v20140415
u26, Gtype  n1-standard CPU 1 Mem 3.75 image centos-6-v20140415
u27, Gtype  n1-standard CPU 6 Mem 22.5 image centos-6-v20140415
u28, Gtype  n1-standard CPU 1 Mem 3.75 image centos-6-v20140415
u29, Gtype  n1-standard CPU 3 Mem 11.25 image centos-6-v20140415
u30, Gtype  n1-standard CPU 1 Mem 3.75 image centos-6-v20140415
```

Fig. 4: The generated users' classes UCs

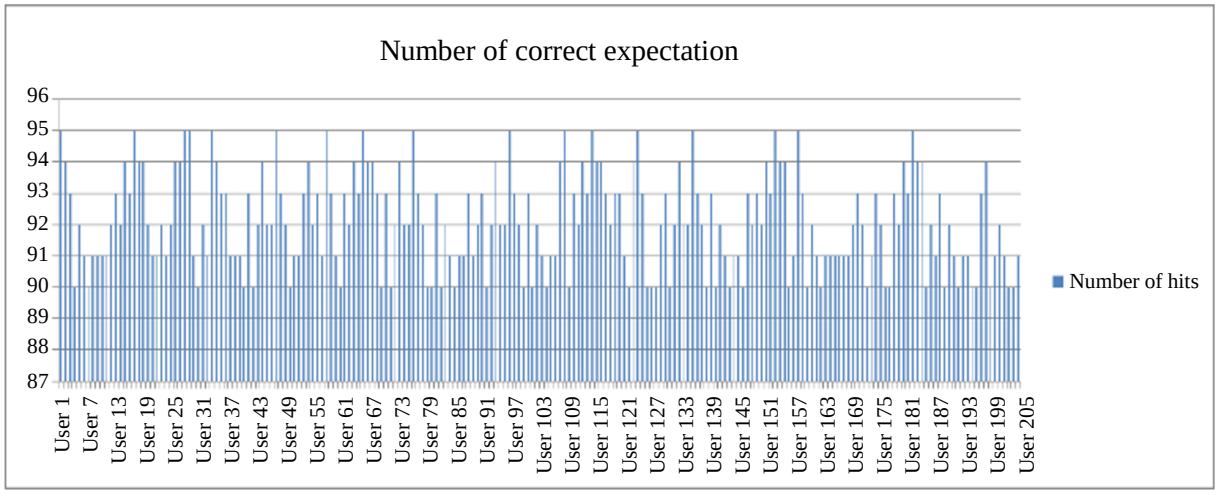

Fig. 5: Users' behaviour expectation rate

Table 1: Number of coalitions and individual users when using a fixed-coalition-size policy for 210 users

Coalition size	No. of users outside a coalition	Percent of users outside a coalition	No. of coalitions	No. of needed VMs
2	46	21.90%	82	128
3	81	38.50%	43	124
4	110	52.30%	25	135
5	125	59.50%	17	142

Table 2: Number of coalitions and individual users when using a variable-coalition-size policy for 210 users

Coalition size	No. of individual users	Percent of users outside a coalition	No. of coalitions	No. of needed VMs
2	30	14.20%	90	120
3	57	27.10%	53	110
4	73	34.80%	38	111
5	93	44.30%	32	125

Table 3: Design of the experiments of the coalition formation strategy

	Request size	Running mode	Anticipated result: Achieves resource utilization	Goal
Experiment 1	Off-template	Parallel	Yes	This is a promising scenario that shows how forming coalitions can result in VM that matches template sizes
Experiment 2	Off-template	Not parallel	No	This scenario shows that the algorithm works as expected, treats situations differently and does not always generate positive results
Experiment 3	On-template	Interchangeable	Yes	This is a promising scenario
Experiment 4	On-template	Not interchangeable	No	This scenario shows that the algorithm works as expected, treats situations differently and does not always generate positive results

C. Test the System (Group-Level Resource Optimization)

A number of experiments have been conducted for this study to evaluate the effect of introducing the coalition formation strategy on allocating VMs to users' requests. At the beginning some terminology is defined to facilitate the description of the experiments. Users making requests that match the size of the templates offered by the cloud, will be referred to as *on-template users*. On the other hand, users who make requests that do not match the size of the cloud templates, will be referred to as *off-template users*. The design of the experiments is shown in Table 3.

For each experiment, a proof-of-concept scenario of 2 or 3 users has been defined, their VM requests described and the results of the allocated VM via the cloud compared using two systems. One system is called the *baseline system*, a system that does not use coalition formation and the other is our system. The workload for all users is multiplying a number by itself for a specified interval of time programmed in Python. Following can be found the description of these experiments in detail.

a. Experiment 1

Experiment settings: For the purpose here, it is presumed to have 3 users, referred to as U1, U2 and U3. Table 4 shows the actual requirement of each user. Subsequently, three VMs will be created in the baseline system and two of them will match the actual CPU core requirements for U1 and U2. However, it will also create a third VM with 2 extra cores as this is the nearest template size offered by the cloud.

On the other hand, in our system, part of the decision is made prior to receiving the requests, thus, when the requests from the three users are received, the system will identify that all users belong to the same coalition and, consequently, it will forward the requests to the VM reserved for this coalition.

Running the experiment: As this experiment shows users running in parallel, U1 and U2 will run in parallel with U3 for 15 min. U1 will run for 5 min followed by U2 for 10 min.

In this experiment, measured execution time, CPU usage and cost for both the baseline system and our system are all studied.

Discussion: From Table 4, it have been found that actual execution of the workload is the same for each user, however, in the baseline system there are an extra 30 sec needed for initializing the VM, whereas our system only consumes around one second to determine the group number and the VM id.

Regarding the CPU usage, it can be seen from Table 5 and Fig. 6, that the usage for VM 1 and VM 2 reaches approximately 99%. however, for VM 3, the CPU usage reaches 75%. This wastage is due to confirming the template size provided by the cloud. It is noticed from Fig. 7 that forming the coalition raised the CPU usage to 99%.

Table 4: Execution time and CPU usage for baseline and coalition systems of experiment 1

User	Actual VM requirements	VM Request		Execution Time			CPU Usage	
		Baseline System	Coalition System	Baseline System	Coalition System		Baseline System	Coalition System
U1	Standard type VM that consists of 1 CPU core and 3.75 GB RAM	VM 1 consists of cores as requested	Standard type VM that consists of 8 CPU cores and 30 GB RAM	600 s +30 s = 630 s	600+0.74 600.74s	=	99.14%	99.49%
U2	Same as U1	VM 2 consists of cores as requested		300 s +30 s = 330 s	300+0.85 300.85 s	=	99.3%	
U3	Standard type VM that consists of 6 CPU cores and 22.5 GB RAM	VM 3: Standard type VM that consists of 8 CPU cores and 30 GB RAM (the nearest size available as template that is provided by GCE)		900 s +30 s = 930 s	900+0.77 900.77s	=	74.6%	

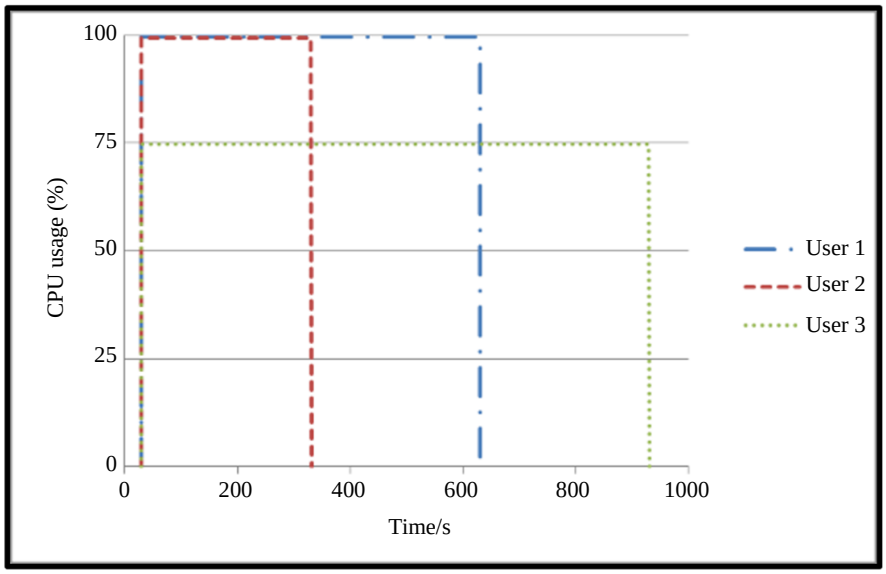

Fig. 6: CPU usages of all users of Baseline system in Experiment 1

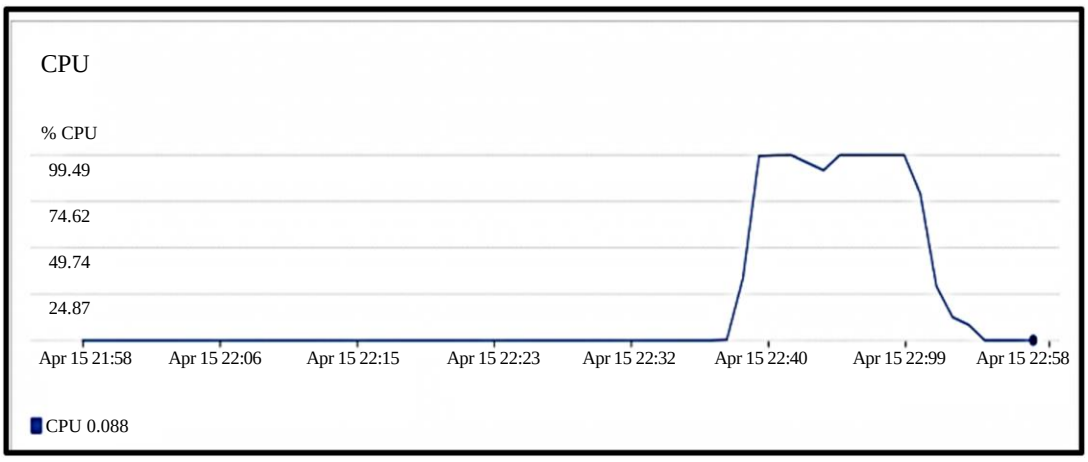

Fig. 7: CPU usage of all users as a coalition in Experiment 1

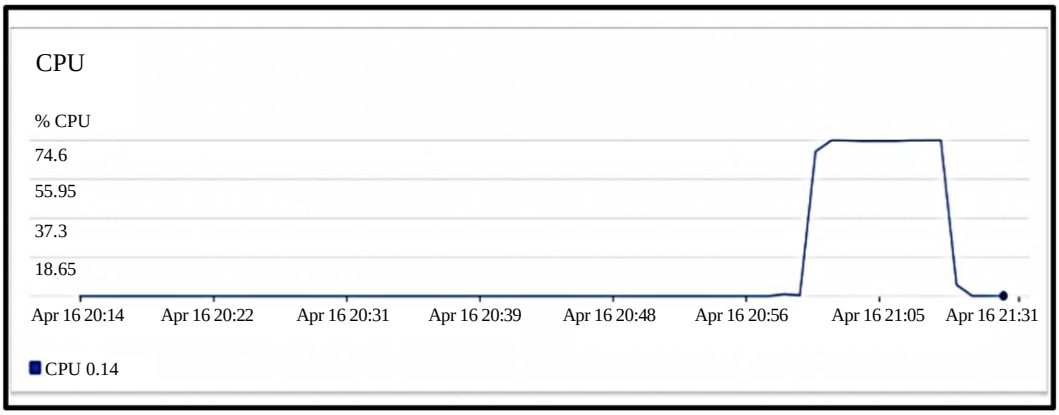

Fig. 8: CPU usage for U3 of our system in experiment 2

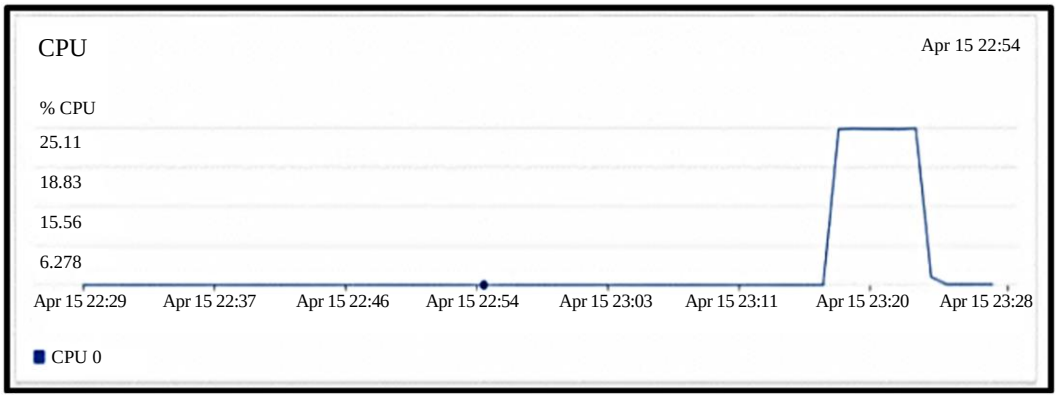

Fig. 9: CPU usages for U1 and U2 running and U3 does not run in our system – Worst case of Experiment 2

b. Experiment 2

An interesting negative scenario is when three users having the same requirements listed in Table 4, however, U1 and U2 do not run at the same time as U3. The result as shown in Fig. 8 reveals a potential waste of around 25% of the CPU usage, as would happen with the baseline system.

One example of the worst case would be when U3 does not execute. As shown in Table 6 and Fig. 9, the result of running the coalition would waste 75% of CPU usage.

c. Experiment 3

Experiment settings: Suppose there are 2 users referred to as U1 and U2. Table 7 shows the actual requirement of each user. Two VMs will be created in the baseline system.

On the other hand, in our system, again the system will identify that both users belong to the same coalition and consequently, it will forward the requests to the VM reserved for this coalition.

Running the experiment: This experiment shows users running in an interchangeable mode. For ease of description, the workload for one hour will be described as divided into four equal timeslots, i.e., each slot is 15minutes long. For each timeslot, *only one* user will run their request. The resulting sequence of users' workloads will be as follows: U1-U2-U1-U2.

Table 5: CPU usage of Baseline system and coalition system in experiment 2

	CPU usage	
User	Baseline system	Coalition system
User 1	99.14%	74.72%
User 2	99.3%	
User 3	74.6%	

Table 6: Worst case in CPU usage of our system - Experiment 2

	CPU usage	
User	Baseline	Coalition two small user work
User 1	99.14 %	25. 11%
User 2	99.3 %	
User 3	74.72%	

In this experiment, the measured execution time, CPU usage and cost for both the baseline system and our system are measured.

Discussion: From the previous table, it is noticed that our system still has the advantage of starting execution earlier by a total of 58 sec (29 sec for each created VM), after deducting the time for identifying the group No. and the VM ID.

Regarding the CPU usage, it can be seen from Table 8, Fig. 10 and 11, that, with the baseline system,

utilization reaches approximately 66%, because each user becomes idle for 15 minutes in between execution. In our system, utilization reaches 99.36% as can be seen in Fig.12, as the coalition takes advantage of the idle timeslots to run the workload of the other user. 99.36% as can be seen in Fig. 12.

d. Experiment 4

This experiment uses the same requirements as described in Table 7; however, the two users work in parallel. This is an interesting negative scenario where forming coalitions are seen to perform worse than the baseline system. As it can be seen from Table 9, execution time nearly reached double the time of the baseline system. This is caused by the need for scheduling.

In addition, it is noticed that the CPU usage decreases from around 99% to approximately 98% as shown in Table 10 and Fig. 13, because of the CPU becoming idle while performing context switching from one user to another (Yuan and Liu, 2011).

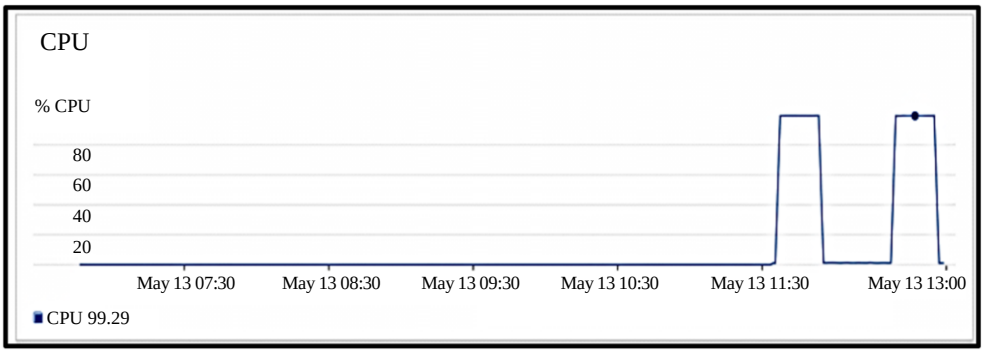

Fig. 10: CPU usage for U1 of Baseline system in Experiment 3

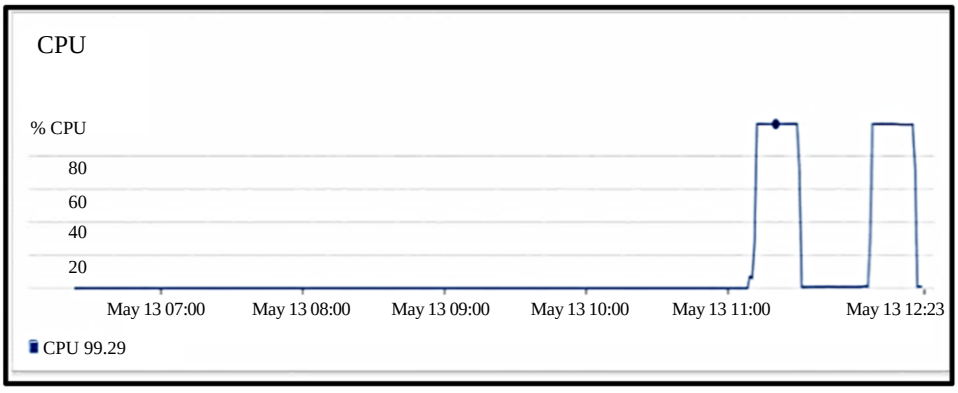

Fig. 11: CPU usage for U2 of Baseline System in Experiment 3

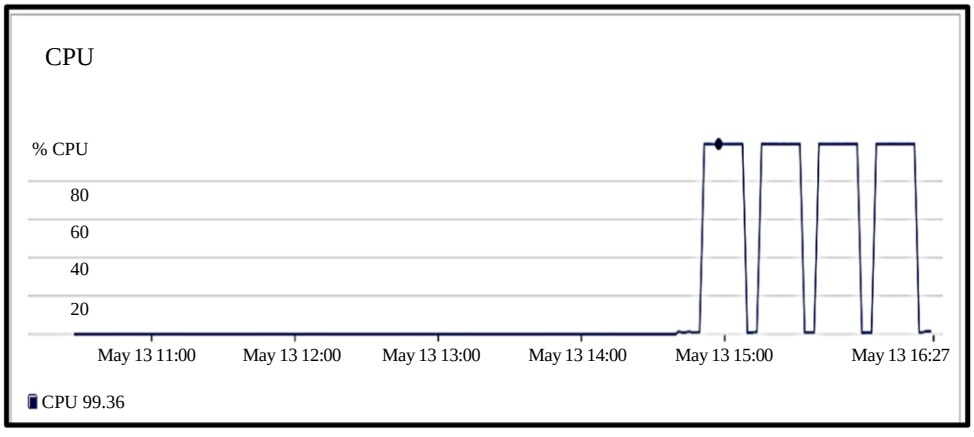

Fig. 12: CPU usage for U1 and U2 in Coalition system in Experiment 3

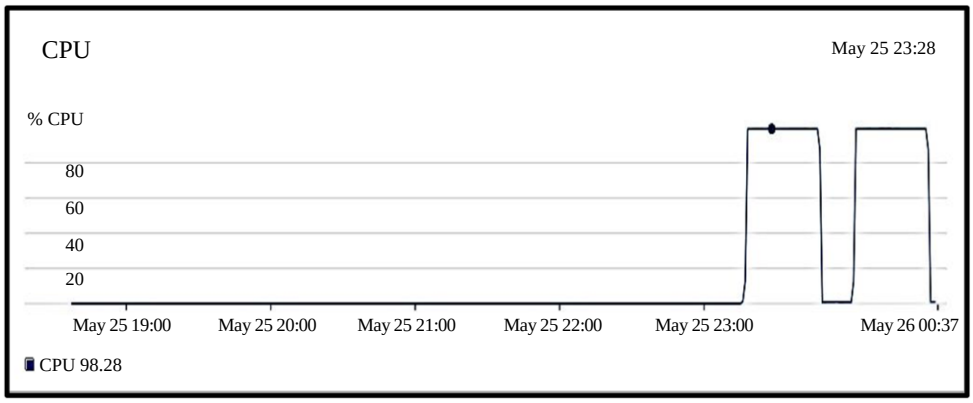

Fig. 13: PU usage - worst case (Experiment 4)

Table 7: Execution time and CPU usage for baseline and coalition systems of experiment 3

User	Actual VM requirement	VM Request		Execution Time		CPU Usage	
		Baseline System	Coalition System	Baseline System	Coalition System	Baseline System	Coalition System
U1	VM of type standard with 1 CPU core and 3.75 GB RAM	VM 1 consists of cores *as requested*	Standard type VM that consists of 1 CPU core and 3.75 GB RAM	900 s +900 s + 30 s = 1830 s	900 s +900 s + 0.80 s = 1800.80 s	(99.29%+0 +99.29%)/ 3 = 66.19%	99.36%
U2	Same as U1	VM 2 consists of cores *as requested*		900 s + 900 s + 30 s = 1830 s	900 s + 900 s +0.76 s = 1800.76 s	(99.4%+0 +99.4%)/3 = 66.27%	

Table 8: Execution time for baseline and coalition systems in experiment 3

	Execution time	
User	Baseline system	Coalition system
User 1	900+900+30 s = 1830 s	900+900+0.80 s = 1800.80 s
User 2	900+900+30 s = 1830 s	900+900+0.76 s = 1800.76 s

Table 9: Execution time for baseline and coalition systems of experiment 4

	Execution time	
User	Baseline system	Coalition system
User 1	900+900+30 s = 1830 s	1864+1836+0.81 s = 3700.81 s
User 2	900+900+30 s = 1830 s	1743+1886+0.70 s = 3629.70 s

Table 10: Worst case in CPU usage experiment 4

	CPU usage	
User	Baseline system	Coalition system
User 1	99.29%	98.28%
User 2	99.4%	

Discussion

From the above experiments, several ideas are studied, which are deriving the type of user from their pervious behaviour, determining the size of the coalition and the settings of the cases that benefit from the coalition formation strategy. Firstly, determining the user's type as either standard, high-CPU or high-Memory, in this study requires analysing their behaviour on an hourly basis. Secondly, trying several coalition sizes, it turns out that having a coalition size of four members and below and allowing for variable sizes of coalitions to form yields better formation among the whole system. This setting provided a smaller number of users that do not belong to a coalition, thus, smaller number of VM of individual users are needed. Thirdly, different cases of cloud users with different requirements and behaviour are studied as described in Table 3. It turns out that the case where users work in parallel mode and they request on-template size of VMs provided saving of approximately 25% of CPU usage of one of the used machines. In the case where users work in interchangeable mode and request off-template VM sizes, there is an improvement of 34% in CPU Usage. The limitations of the strategy are shown in the results of experiment 2 and experiment 4. In experiment 2, both systems had the same performance, which causes the waste of 25% of CPU usage. In experiment 4, the coalition formation system took nearly double the time

that the base line system needed to run the tasks. This lack of improvement occurs when the expected behaviour does not happen.

Finally, comparing the coalition formation system performance is done against the original GCE system which we refer to as the baseline system and as can be seen in some scenarios the proposed system outperformed the GCE system.

Conclusion and Future Work

Resource allocation is a challenging task. Several techniques and methods have been proposed to ensure allocating resources to cloud users that fulfil their requirements and minimizes waste. However, there is still room for investigating new approaches to tackle the resource allocation problem. The contribution of this work is the novel use of the coalition formation strategy to tackle the resource allocation problem. It illustrated cases where using this strategy has been beneficial and also the negative cases. The promising cases offered improvement in the efficiency of the process, minimizing the number of used VM, consequently, contributing to green computing and minimizing the cost for service provider and users. It is vivid that the performance of the system is dependent on the accuracy of the system's prediction. The limitation of this strategy is that it is dependent on the user's behaviour pattern, in provisioning of the resource, however, in case one user or more changed the expected behaviour, this will lead to some complication using the derived coalitions. In addition, a key question is related to whether there is enough divergence or convergence among the requirements of the users to make coalitions successful. This point needs further investigation on the behaviour of users of existing clouds. There is also the concern related to serving coalition members within the same VM. Solutions to this concern were discussed in Section V. For future work, evaluating the coalition formation strategy on real data, such as Google trace log will give better insights regarding the practicality of the system. In addition, using more sophisticated prediction algorithms for user behaviour analysis may provide useful improvements on the performance of formed coalitions. Finally, the policy for forming coalitions used in this study join users that have similar requested VM types in the same coalition, it would be interesting to study forming coalitions based on requirements with heterogeneous VM types.

Acknowledgment

Our team wishes to thank Mr. Osama Younis for providing essential technical support. We would also like to thank Prof. Fathy Eassa -the head of our research team- for his continuous guidance. In addition, we would like to thank Dr. Lamiaa Elrfaei, Prof. Hanan Elazhari and Dr. Etimad Fadel for their valuable input.

Author's Contributions

Hend Fakhri Noureldin: Provided the results of the experiments and laid out the first draft of the paper.

Mai Fadel: Supervised the design of the experiments, improved the literature review part and re-written some parts of the papers to be clear.

Both authors revised the final version of the document.

Ethics

This is original work resulting from a Master thesis done by Hend and supervised by Mai.

References

Alsadie, D., Tari, Z., Alzahrani, E. J., & Zomaya, A. Y. (2017, October). Energy-efficient tailoring of VM size and tasks in cloud data centers. In 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA) (pp. 1-5). IEEE. https://doi.org/10.1109/NCA.2017.8171339

Alsadie, D., Tari, Z., Alzahrani, E. J., & Zomaya, A. Y. (2018, January). Dynamic resource allocation for an energy efficient vm architecture for cloud computing. In Proceedings of the Australasian Computer Science Week Multiconference (pp. 1-8). https://doi.org/10.1145/3167918.3167952

Amiri, M., & Mohammad-Khanli, L. (2017). Survey on prediction models of applications for resources provisioning in cloud. Journal of Network and Computer Applications, 82, 93-113. https://doi.org/10.1016/j.jnca.2017.01.016

Bairagi, A. K., Alam, M. G. R., Talukder, A., Nguyen, T. H., & Hong, C. S. (2016, January). An overlapping coalition formation approach to maximize payoffs in cloud computing environment. In 2016 International Conference on Information Networking (ICOIN) (pp. 324-329). IEEE. https://doi.org/10.1109/ICOIN.2016.7427124

Barán, B., & López-Pires, F. (2017). Resource Allocation for Cloud Infrastructures: Taxonomies and Research Challenges. In Research Advances in Cloud Computing (pp. 263-289). Springer, Singapore. https://doi.org/10.1007/978-981-10-5026-8_11

Bi, J., Yuan, H., Tan, W., Zhou, M., Fan, Y., Zhang, J., & Li, J. (2015). Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center. IEEE Transactions on Automation Science and Engineering, 14(2), 1172-1184. https://doi.org/10.1109/TASE.2015.2503325

Chen, J. (2018, August). A cloud resource allocation method supporting sudden and urgent demands. In 2018 Sixth International Conference on Advanced Cloud and Big Data (CBD) (pp. 66-70). IEEE. https://doi.org/10.1109/CBD.2018.00021

Deng, K., Song, J., Ren, K., & Iosup, A. (2013, November). Exploring portfolio scheduling for long-term execution of scientific workloads in IaaS clouds. In SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (pp. 1-12). IEEE. https://doi.org/10.1145/2503210.2503244

Dewangan, B. K., Agarwal, A., Choudhury, T., Pasricha, A., & Chandra Satapathy, S. (2020). Extensive review of cloud resource management techniques in industry 4.0: Issue and challenges. Software: Practice and Experience. 1–20. https://doi.org/10.1002/spe.2810

Dezhabad, N., Ganti, S., & Shoja, G. (2019, November). Cloud Workload Characterization and Profiling for Resource Allocation. In 2019 IEEE 8th International Conference on Cloud Networking (CloudNet) (pp. 1-4). IEEE. https://doi.org/10.1109/CloudNet47604.2019.9064138

GCE. (2014). Google Compute Engine. https://cloud.google.com/compute/docs/disks

Gong, S., Yin, B., Zheng, Z., & Cai, K. Y. (2019). Adaptive multivariable control for multiple resource allocation of service-based systems in cloud computing. IEEE Access, 7, 13817-13831. https://doi.org/10.1109/ACCESS.2019.2894188

Hadjres, S., Kara, N., El Barachi, M., & Belqasmi, F. (2018). An SLA-aware cloud coalition formation approach for virtualized networks. IEEE Transactions on Cloud Computing. https://doi.org/10.1109/TCC.2018.2865737

Islam, S., Keung, J., Lee, K., & Liu, A. (2012). Empirical prediction models for adaptive resource provisioning in the cloud. Future Generation Computer Systems, 28(1), 155-162. https://doi.org/10.1016/j.future.2011.05.027

Kaur, G., Bala, A., & Chana, I. (2019). An intelligent regressive ensemble approach for predicting resource usage in cloud computing. Journal of Parallel and Distributed Computing, 123, 1-12. https://doi.org/10.1016/j.jpdc.2018.08.008

Kholidy, H. A. (2020). An intelligent swarm based prediction approach for predicting cloud computing user resource needs. Computer Communications, 151, 133-144. https://doi.org/10.1016/j.comcom.2019.12.028

Li, C., Li, J., Li, Y., & Han, Z. (2019). Bayesian coalition formation game for virtual 5G core network functions. IEEE Access, 7, 29805-29817. https://doi.org/10.1109/ACCESS.2019.2902419

Maurer, M., Brandic, I., & Sakellariou, R. (2013). Adaptive resource configuration for cloud infrastructure management. Future Generation Computer Systems, 29(2), 472-487. https://doi.org/10.1016/j.future.2012.07.004

Moreno, I. S., Garraghan, P., Townend, P., & Xu, J. (2014). Analysis, modeling and simulation of workload patterns in a large-scale utility cloud. IEEE Transactions on Cloud Computing, 2(2), 208-221. https://doi.org/10.1109/TCC.2014.2314661

Ravi, K., Khandelwal, Y., Krishna, B. S., & Ravi, V. (2018). Analytics in/for cloud-an interdependence: A review. Journal of Network and Computer Applications, 102, 17-37. https://doi.org/10.1016/j.jnca.2017.11.006.

Saidi, K., Hioual, O., & Siam, A. (2019, November). Resources allocation in cloud computing: a survey. In International Conference in Artificial Intelligence in Renewable Energetic Systems (pp. 356-364). Springer, Cham. https://doi.org/10.1007/978-3-030-37207-1_37

Singh, S., & Chana, I. (2014). Metrics based workload analysis technique for IaaS. In: International Conference on Next Generation Computing and Communication Technologies, (pp. 1–6). http://arxiv.org/abs/1411.6753%5Cnhttp://dblp.uni-trier.de/rec/bib/journals/corr/SinghC14a

Singh, S., & Chana, I. (2015). Q-aware: Quality of service based cloud resource provisioning. Computers & Electrical Engineering, 47, 138-160. https://doi.org/10.1016/j.compeleceng.2015.02.003

Sunyaev, A. (2020). Cloud computing. In Internet Computing (pp. 195-236). Springer, Cham. https://doi.org/10.1007/978-3-030-34957-8_7

Thakur, N., & Han, C. Y. (2021). Multimodal Approaches for Indoor Localization for Ambient Assisted Living in Smart Homes. Information, 12(3), 114. https://doi.org/10.3390/info12030114

Vashistha, A., & Verma, P. (2020, January). A Literature Review and Taxonomy on Workload Prediction in Cloud Data Center. In 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 415-420). IEEE. https://doi.org/10.1109/Confluence47617.2020.9057938

Wei, L., Foh, C. H., He, B., & Cai, J. (2015). Towards efficient resource allocation for heterogeneous workloads in IaaS clouds. IEEE Transactions on Cloud Computing, 6(1), 264-275. https://doi.org/10.1109/TCC.2015.2481400

Xiao, Z., Song, W., & Chen, Q. (2012). Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Transactions on Parallel and Distributed Systems, 24(6), 1107-1117. https://doi.org/10.1109/TPDS.2012.283

Yuan, Y., & Liu, W. C. (2011, October). Efficient resource management for cloud computing. In 2011 International Conference on System science, Engineering design and Manufacturing informatization (Vol. 2, pp. 233-236). IEEE. https://doi.org/10.1109/ICSSEM.2011.6081285