Computers, Materials & Continua
DOI: 10.32604/cmc.2023.035139
Article

Tech Science Press

ACO-Inspired Load Balancing Strategy for Cloud-Based Data Centre with Predictive Machine Learning Approach

Niladri Dey[1], T. Gunasekhar[1] and K. Purnachand[2,*]

[1]Department of CSE, K. L. University, Vaddeswaraam, 522302, Andhra Pradesh, India
[2]Department of CSE (Data Science), B. V. Raju Institute of Technology, TS, 502313, India
*Corresponding Author: K. Purnachand. Email: purnachandkollapudi2022@gmail.com
Received: 09 August 2022; Accepted: 15 November 2022

Abstract: Virtual Machines are the core of cloud computing and are utilized to get the benefits of cloud computing. Other essential features include portability, recovery after failure, and, most importantly, creating the core mechanism for load balancing. Several study results have been reported in enhancing load-balancing systems employing stochastic or biogenetic optimization methods. It examines the underlying issues with load balancing and the limitations of present load balance genetic optimization approaches. They are criticized for using higher-order probability distributions, more complicated solution search spaces, and adding factors to improve decision-making skills. Thus, this paper explores the possibility of summarizing load characteristics. Second, this study offers an improved prediction technique for pheromone level prediction over other typical genetic optimization methods during load balancing. It also uses web-based third-party cloud service providers to test and validate the principles provided in this study. It also reduces VM migrations, time complexity, and service level agreements compared to other parallel standard approaches.

Keywords: Predictive load estimation; load characteristics summarization; correlation-based; parametric reduction; corrective coefficient-based

1 Introduction

Cloud computing pools and shares computing, storage, and networking resources. These shared resources can be scaled based on resource pool usage. According to Rimal et al. [1], application developers and the industry accept the shared resource management of pooled resources method. Cloud computing allows remote program development, hosting, and management. Patidar et al. [2] say most network-based access to shared resources is paid. Cloud-based service providers use load balancing to control resource usage automatically. Unless told otherwise, pooled computers share network demand. Load-balancing solutions may help. Kaur et al. [3] say load balancing uses virtual machines and subsequent allocations or migrations. Virtualization allows multiple operating systems to run on a single physical resource or resource pool, according to Chandra et al. [4]. Virtualization lets data centers move applications, share resources, handle errors, and spread the load. Application owners may need portability since new hardware and software will likely use the same platform and hardware. Application and data center owners may save money by sharing load-balancing resources. Virtual machines use physical resources. This gives developers a virtual representation of shared physical resources. Resource competition may occur when many virtual machines use the same physical pool. Datacenter providers use hypervisors more. According to Mishra et al. [5], virtual machines have many benefits. Virtualization impacts hardware and software. Computed resources include processors, storage devices, and networks. According to Agarwal et al. [6], virtual machines' live migration resources are only loosely linked. They are spreading the load. During live migration, old sources are removed, and new ones are added. To move a virtual machine, reset it. Keep the virtual machine's memory.

Barzoki et al. [7] found a growing need for scalable, high-performance applications. Akbari et al. suggested researching biology-based algorithms [8]. Current research focuses on making algorithms to divide work without losing virtualization's benefits. This chapter demonstrates a natural way to balance loads. Next, the paper explains why more research is needed. Henceforth, it is natural to realize the demand for further optimizations. These algorithms are primarily driven by the rule engine and cannot encourage dynamic rule building, making them less dynamic and responsive to load balancing. These strategies are also less effective in terms of proper virtual machine utilization. Finally, it is discovered that these mechanisms are bottlenecked for reducing response time beyond a certain scale. Thus, this paper briefly aims to evaluate the possibility of summarizing load characteristics. This study also provides a better prediction technique for pheromone level prediction than other common genetic optimization methods used during load balancing. However, finding the best solution usually depends on how the problem is set up and, to a large extent, on the state of the data fed into the algorithms. So, optimization techniques are often used to make further improvements and find the best solution to a problem.

Further, this paper is organized such that, in Section–2, the foundational understanding of this domain of the research is furnished, the baseline methods for the load balancing are discussed to evaluate research gaps, in Section–3 as proposed algorithms, further this paper furnishes the obtained results and discuss the improvements compared with the parallel other research outcomes in Sections–4 and 5, the research conclusion is furnished.

2 Background of the Research

In this Section of the work, the foundation layout of the research is furnished. Bio-Inspired methods are widely popular in various fields of research. Every research in various fields proposes multiple algorithms to solve domain-specific problems, and the solutions by these algorithms lead to specific problem solutions. Nevertheless, algorithms often research to some extent where further improvements cannot be proposed, only relying on the core strategical solutions, and many of these algorithms provide multiple solutions, which are all timely and effective. However, finding the most optimal solution habitually relies on the problem environment, which can be significantly represented by the state of the data as input to the algorithms. Thus, optimization techniques are widely used for further improvements and to find the optimal solution for the problem. Apart from finding the best solution or the solution space, the optimization algorithms on top of the primary problem-solving algorithms can cater to a wide range of solutions. The Bio-Inspired optimization algorithms can be primarily categorized into two different categories:

2.1 Deterministic Methods

The first category is the deterministic methods. The deterministic method of the problem solution ensures the optimization solution in the complete solution space. The deterministic algorithms usually consider the features or the parameters which describe the problem in detail and thus guarantee to find the global solution to the same problem. Nevertheless, the deterministic methods are limited to solving the problem with less or no information from the outside and are generally treated as black box problems. Also, this method is difficult to solve in the case of the problem, which is highly unpredictable and frequently changes the pattern of the transitions. Further, these methods can also be difficult to apply to some problems, which deal with a higher order of the data and comparatively large dependencies, as demonstrated in work by Li et al. [9]. There are many prominent methods in this optimization category, such as Branch & Bound, Cutting Plane, Inner & Outer approximation, and convex methods.

2.2 Stochastic Method

The second category of them is the stochastic methods. These methods can find the solution in finite time and faster than the deterministic methods. Nonetheless, the stochastic methods cannot provide the optimal global solution for any problem solution space. The stochastic methods can only find the probabilistic solution in finite time, as showcased and proven by Zhou et al. [10]. The wide acceptance of the stochastic methods is driven by various advantages such as:

Firstly, less complex mathematical modeling can still be useful for finding the solution. Secondly, the case of large-scale data-driven problems, the recent problem trend, can be well handled by the stochastic methods as elaborated in work by Yan et al. [11].

Lastly, the time to find the optimal solution, the optional probabilistic solution, is less than the deterministic methods. The popular optimization algorithms under the stochastic methods are genetic algorithms, ant colonies, artificial bee colonies, particle swarms, and many more. This work also proposed a novel optimization method using bio-inspired strategies on the edge of the stochastic optimization method. Further, this work elaborates on the possibilities of applying bio-inspired optimization methods to solve cloud computing load-balancing problems. Nevertheless, these algorithms are criticized for the following reasons:

- These algorithms are primarily driven by the rule engine. They cannot encourage dynamic rule building, which makes these algorithms less dynamic and less responsive to load balancing.
- These strategies are also less effective in properly utilizing virtual machines.
- Finally, these mechanisms are found to be bottlenecked for reducing the response time beyond a certain scale.

Thus, these problems demand further research. In this chapter of the work, the problems encountered by the parallel research attempts are addressed with a novel bio-inspired mechanism for optimizations. The next Section of the work addresses the fundamental strategy for load balancing on cloud computing, and the principle mathematical model is analyzed.

3 Formulation of the Research Problem and Proposed Load Summarization Process

This portion of the paper discusses the basic technique for load balancing and summary. The underlying idea of load balancing is virtual machine migration. Thus, the load computation and migration methods must be well understood. According to Medina et al. [12], the commonly acknowledged procedure of migrating virtual machines from the source physical resource pool to the destination physical resource pool is termed live migration. The source virtual machines and services are not stopped, and the destination virtual machines and services are not restarted, but all maintenance activities may be done using kernel procedures. This technique improves availability and service level violations compared to other parallel solutions.

Regardless, integrating physical resource usage is the key to success for any load-balancing strategy. So, in this part, we suggest first the load summarization approach. Wen et al. [13] recently suggested a load formulation approach. This method's main flaw is the need for more consideration for diverse service requests. This Section explains the main problem with this method and sets up a different way to add up all the loads using service type load analysis in the mathematical lemma. The benefits of this load-balancing or virtual machine-moving strategy are explained [14–19]. Virtual machine bandwidth is hard to determine. This proposed change to workload summarization is helpful. Different types of services require different types of storage containers. Migration is based on more than capacity. With this change, we can see the real demand. Depending on when they need data, different services have different storage and replication needs. People's abilities don't explain migration. So, the proposed change to work summaries is helpful. Load balancing or virtual machine migration after a proposed method corrects workload summarization requires a basic understanding of optimization methods [20–26]. The recent outcomes also suggest that the summarization process of loads shall lead to the correct identification of the problem [27–29].

3.1 Formulation of the Research Problem

The flaws in the current methods have been exposed by the discovery of the fundamentals of load succinct summation, primitive optimization methods, and recent research on the primitive method. From now on, the issues that need to be addressed in this study will be outlined in this Section of the paper.

During this phase, two problems are recognized and formulated so that the solutions can be modeled in greater detail.

Primitive and enhanced ACO methods have both been criticized for being overly complex when applied to large search spaces, as discussed in the previous part of this document. Thus, to demonstrate the reduction potential of predictive methods, the following equation lemma is formulated:

Lemma–1: The prediction of the future load conditions shall lead to a significant reduction of the time complexity for any load-balancing algorithm.

Proof: The standard analogy compares the performance or the time complexity for two load balancing strategies, wherein in the first instance, the Algorithm calculates the load situations reactively, and in the second instance, the Algorithm calculates the load conditions proactively or using predictive analysis. However, the standard load analysis algorithms are furnished into four primary phases identification of the load condition (LC), identification of the instance capacity (IC), load optimization (LO), and finally, migration between the instances (MB). Here for reactive algorithms, assuming that these four phases take the time as t1, t2, t3, and t4, respectively with the total time as T. Thus, this can be realized as,

$$T = \langle t1 + t2 + t3 + t4 \rangle \tag{1}$$

Naturally, the first phase of identification of the load condition is highly repetitive and must be added as a service protocol to the data center architecture. Hence the time t1 is significantly higher compared to the other three phases' time complexity. As

$$t1 > \lfloor t2|t3|t4 \rfloor \tag{2}$$

Similarly, it is natural to realize that the optimization phase is also iterative and must be almost equal to the identification phase. Also, the migration phase may be flexible, but due to the larger size of the virtual machine, may take longer. Hence, the complete comparison between the time complexities in various phases can be identified as,

$$t_1 > t_4 > t_3 > t_2 \tag{3}$$

Alternatively, during the proactive or the predictive strategy for the same Algorithm, assuming that the standard load analysis algorithms are furnished into four primary phases identification of the load condition (LC), identification of the instance capacity (IC), load optimization (LO) and finally migration between the instances (MB). Here for reactive algorithms, assuming that these four phases take the time as t11, t22, t33, and t44, respectively with the total time as T1. Thus, this can be realized as

$$T1 = \langle t11 + t22 + t33 + t44 \rangle \tag{4}$$

Fundamentally, for the predictive strategy-driven algorithms, the iterative phases must be converted into a single-step process and can be computed during the previous step of the algorithms. As the identification of the load condition must be completed during the migration process of the previous phase. The separate time taken for identifying the load condition tasks is completed within the same time limit of the migration process. As

$$t11 \subseteq t44 \tag{5}$$

Henceforth, this understanding must also be incorporated with the previous assumptions, as stated in Eq. (2),

$$T1 = \langle \lfloor t11|t44 \rfloor + t22 + t33 \rangle \tag{6}$$

These enhancements to the Eq. (6) can be justified with the following relation,

$$\begin{aligned} &\textit{If } t11 \geq t44, \textit{then } T1 = \langle \lfloor t11 \rfloor + t22 + t33 \rangle \\ &\textit{Else}, \textit{ then time} \rightarrow T1 = \langle \lfloor t44 \rfloor + t22 + t33 \rangle \end{aligned} \tag{7}$$

Hence,

$$T >> T1 \tag{8}$$

Consequently, this mathematical formulation proves that load prediction can substantially decrease the computational complexity of the load-balancing Algorithm using predictive calculations.

Secondly, as discussed in the previous sections of this work, the ABC or BAT methods, both primitive and enhanced methods, are criticized for less stability due to incorporating the various parameters. Henceforth, in the following mathematical model using the correlation method, the stability of the proposed model is aimed to be increased.

Lemma–2: The correlation-based attribute reduction reduces the time complexity and model stability to a greater extent.

Proof: In this mathematical formulation, the non-correlation-based strategy is compared with the correlation-based strategy. Assuming that the non-correlation-based model is deployed with a set of n parameters as P [], and each parameter in the model is denoted as pi, then this relation can be formulated as,

$$P[] = \sum_{i=1}^{n} p_i \tag{9}$$

Also, if processing each parameter for the fitness function is t time, then for total time, T_1, for processing the fitness function can be formulated as,

$$T_1 = n * t \tag{10}$$

On the other hand, if n number of parameters can be reduced to m number of parameters using the correlation factors, then a set of m parameters as P1[] and each parameter in the model is denoted as $p1_i$, can be formulated as,

$$P1\,[] = \sum_{i=1}^{m} p1_i \tag{11}$$

where the n number of parameters can be converted into a lesser number of parameters, m, using the correlation method with X as correlation factors, then each parameter in the reduced set can be formulated as,

$$p1_i = \beta_1.p_i + \beta_2.p_{i+1} + \beta_3.p_{i+2} + \ldots.. \tag{12}$$

Again, if processing each parameter for the fitness function is t time, then for total time, T_2, for processing the fitness function can be formulated as,

$$T_2 = m * t \tag{13}$$

Here it is significant to realize that $m << n$ the following relation can be easily derived,

$$T_1 >> T_2 \tag{14}$$

Thus, with this mathematical formulation, it is conclusive that the correlation-based attribute reduction reduces the time complexity and model stability to a greater extent. The proposed predictive model is presented in the next Section of this work. Henceforth, based on the problem identifications and mathematical problem modeling, the proposed solutions with the mathematical models are furnished in the next Section of this work.

3.2 Mathematical Model for the Proposed Solutions

After understanding the load summarization and balancing process, primitive and recent methods for optimizations with the fundamental bottlenecks, and formulating the problems to be solved, the mathematical models for the proposed solutions are furnished in this Section of the work. This Section of the work discusses 2 major solutions, apart from the contribution of load summarization in the previous Section for predictive analysis of the load balancing mechanism with the correlation-based fitness function minimizations. Firstly, as demonstrated in Lemma–2, the incorporation of the predictive model for pheromone level is formulated.

Lemma–3: The corrective coefficient-based pheromone prediction analogy can improve the performance of the load-balancing strategy.

Proof: The corrective coefficient-based pheromone prediction strategy is formulated in this mathematical formulation. Assuming that the following is the network physical resource pool availability.

In this model example, there are a total of 4 physical resource pools configured as N1, N2, N3, and N4. Here N1 physical resource pool is categorized as the source or the overloaded physical resource pool, and other nodes are identified destinations or available physical resource pools for migration from the source. As this optimization process can be deployed seamlessly and the existing strategy is assumed to be already deployed with the ACO method, thus there are some initial pheromone levels exists on each path as traces of the previous virtual machine migrations. Thus, some pheromones exist for every path as PH1 to PH6, respectively. Also, assuming that any given virtual machine, V_i, is allocated the physical resource pools as n number of computes, C, m number of memory units, M, k number of storage units, S and finally, p number of network resources, N. Thus, the load, L, at a specified time point, t, can be presented as,

$$L(t) = \sum_{x=1}^{n} C_x + \sum_{x=1}^{m} M_x + \sum_{x=1}^{k} S_x + \sum_{x=1}^{p} N_x \tag{15}$$

Further, if the pheromone level at a given time t can be represented as PH(t) and at time t+1, the pheromone level can be presented as PH(t+1). The increase or the decrease of the pheromone level can be formulated as,

$$PH(t+1) = PH(t) \pm \Delta\tau \tag{16}$$

The factor $\Delta\tau$ defines the deposition of evaporation of the pheromone as the positive sign denotes the deposition and the negative sign denotes the evaporation. Here, this proposed method deploys the predictive analogy for pheromone level prediction as,

$$PH(t+1) = PH(t).e^{\pm KT} \tag{17}$$

The $e^{\pm KT}$ defines the decay or growth in the rate of pheromone deposition or evaporation, where $\pm K$ defines the rate of growth or decay, which is again controlled by the sign as positive or negative and T denotes the life span of the path or network connection span in this case. Also, the growth or decay rate is a time-dependent variable, which must be decided using the trend, TR(t), and the prediction depth, DP(t). Regardless to mention, these two new variables are also time-dependent. Hence, the growth of the decay rate can be formulated as,

$$K(t) = TR(t) + DP(t) \tag{18}$$

where the greater number of increases or decay in the pheromone levels can decide the trend,

$$TR(t) = \prod_{K\to +ve} K - \prod_{K\to -ve} K \tag{19}$$

Furthermore, the depth of the prediction is the number of historical data points considered. Regardless, as this proposed method relies on the pheromone level prediction, thus the correction during the prediction phases is highly important. Hence, this method deploys the correction strategy with the help of the mean square error α as the correction coefficient method as formulated here,

$$\alpha = \frac{\sum_{i=1}^{n} \left| k_i(t)^2 - k_i'(t)^2 \right|}{n} \tag{20}$$

Here, $k_i'(t)$ it defines the actual growth or decay and $k_i(t)$ the predicted growth or decay for n number of notes at time t. Finally, the prediction of the pheromone level for any given path can be formulated with the help of Eqs. (17)–(20) as,

$$PH\left(t+1\right)=PH\left(t\right).e^{\pm\prod_{K\to+ve}K(t)-\prod_{K\to-ve}K(t)+DP(t).T}\pm\alpha \tag{21}$$

The proof of the performance improvement is already established in the previous Section of this work. As demonstrated in Lemmas–2 and 3, incorporating correlation-based parametric reduction shall reduce the complexity.

Lemma–4: Correlation-based parameter reduction can improve the performance of the load-balancing strategy.

Proof: The correlation-based parameter-reduction strategy is formulated in this mathematical formulation. In this example, we assume that each virtual machine (Vi) has a variable number of processors, C, m multitude of memory units, M, and ultimately p number of social reserves, N, in its physical resource pool. In this way, the load at a given time, t, can be expressed as L,

$$L\left(t\right)=\sum_{x=1}^{n}C_x+\sum_{x=1}^{m}M_x+\sum_{x=1}^{k}S_x+\sum_{x=1}^{p}N_x \tag{22}$$

Also, assuming that any given destination resource pool, D_i, is allocated the physical resource pools as n number of computes, C, m′ number of memory units, M, k′ number of storage units, S and finally p′ number of network resources, N. Thus, the capacity, Cap_D, at a specified time point, t′, can be presented as,

$$Cap_D\left(t'\right)=\sum_{x=1}^{n'}C_x+\sum_{x=1}^{m'}M_x+\sum_{x=1}^{k'}S_x+\sum_{x=1}^{p'}N_x \tag{23}$$

Further, as demonstrated in Lemma–1, during the load summarization process, the generic equivalent must be defined and utilized for the load prediction. Assuming the new load is $L\left(t+1\right)$ at time t+1, it can be formulated as follows,

$$L\left(t+1\right)=\beta_c.\sum_{x=1}^{n}C_x\left(t\right)+\beta_M.\sum_{x=1}^{m}M_x\left(t\right)+\beta_S.\sum_{x=1}^{k}S_x\left(t\right)+\beta_N.\sum_{x=1}^{p}N_x\left(t\right) \tag{24}$$

Here, β_c, β_M, β_S, β_N are the compute, memory, storage, and network load coeffects, respectively, and can be formulated as,

$$\beta_c=\frac{CW}{\sum_{x=1}^{n}C_x\left(t\right)},\beta_M=\frac{MW}{\sum_{x=1}^{n}M_x\left(t\right)},\beta_S=\frac{SW}{\sum_{x=1}^{n}S_x\left(t\right)},\beta_N=\frac{NW}{\sum_{x=1}^{n}N_x\left(t\right)}, \tag{25}$$

CW, MW, SW, and NW are the compute, memory, storage, and network weight constants. The calculation of the weight constants is directly proportional to the number of elements in the service code running on the virtual machine. Korra et al. [23] discuss extracting the number of elements for these four types in work.

Finally, the fitness function can be presented as,

$$Fitness\left(\frac{D_i}{Source}\right)=\left\{\frac{Cap_{D_i}\left(t'\right)}{L\left(t+1\right)},PH\left(t\right).e^{\pm\prod_{K\to+ve}K(t)-\prod_{K\to+ve}K(t)+DP(t).T}\pm\alpha\right\} \tag{26}$$

Henceforth, the prediction of the final load can be predicted with Eq. (24) and the proof of the performance improvement is already established in the previous Section of this work. Also, the fitness function is expected to be maximum from Eq. (26). Thus, with the mathematical models of the proposed solutions discussed in this Section of the work, the proposed algorithm steps are furnished in the next Section.

3.3 Proposed Algorithms

After understanding the load summarization and balancing process, primitive and recent methods for optimizations with the fundamental bottlenecks, formulating the problems to be solved, and the mathematical models for the proposed solutions, in this Section of the work, the proposed algorithms based on the mathematical models are furnished. This work section elaborates on four algorithms for the complete load-balancing strategy. Firstly, the load summarization algorithm is discussed.

Algorithm 1: Load Summarization with Service Categorization (LSSC) Algorithm

```
Input:
Service list as SL [], VM List as VL [], Instance Processing Capacity as C [], Instance Memory Capacity
as M [], Instance Storage Capacity as S [], and Instance Bandwidth Capacity as N []
Output: Service-Specific Summarized Load
Step-1.
  For every element of SL [] as SL[i]
      a. For every element of VL [] as VL[j]
            i. Build the load of compute set as SP [] + = V[j].C []
            ii. Build the load of memory set as SM [] + = V[j].M []
            iii. Build a load of storage set as SS [] + = V[j]. S []
            iv. Build the network load set as SN [] + = V[j]. N []
Step-2.
  Produce sum (SP), sum (SM), sum (SS), sum (SN)
```

Secondly, the pheromone-level prediction algorithm is furnished.

Algorithm 2: Corrective Coefficient Based Pheromone Level Prediction (CCPLP) Algorithm

```
Input:
V: List of VMs, PH(t): Pheromone Level, K1: Rate of growth in PH, K2: Rate of decay in PH, T:
Simulation Duration, K: Set of deposition & evaporation events of PH, TR: Depth of Prediction
Output: PH(t+1) as predicted Pheromone Level
  Step-1.
      For each V[i]
          a. Initialize the parameters as
          b. TR = i
          c. If K[j] = "Growth"
                i. Increase K1
          d. Else
                i. Increase K2
          e. Calculate the rates as K11 = K1/T and K22 = K2/T
          f. Calculate the final rate as (K11 − K22) + TR
          g. Calculate the correction factor, CF = {(K11 − K22) + TR}/{K1 − K2}
          h. Generate PH(t+1) = {PH(t). pow (e, (K11 − K22) + TR)} − CF
  Step-2.
      Return PH(t+1) for each V
```

To better understand the advantages of the Algorithm, examples of real-world data are used. Analytical input is referred to as "input data," as shown in Table 1. Own Data from practical experimentation

Table 1: Initial data for evaluation

Instance type	Initial PH level	No. PH growth events	No. PH decay events	Duration (ms)
c5d.xlarge	13	4	5	22
c5d.2xlarge	19	4	5	58
c5d.4xlarge	18	5	5	25
c3.8xlarge	18	3	4	84
c5d.9xlarge	10	3	3	54
c5d.12xlarge	17	3	5	26
c5d.18xlarge	12	5	5	84
c5d.24xlarge	16	3	4	60

After the Initial PH level and the growth or decay events are identified, in the next phase, the growth rate and the decay rate are calculated as shown in Table 2.

Table 2: Growth and decay rate calculation

Instance type	Initial PH level	Rate of growth	Rate of decay	Duration (ms)
c5d.xlarge	13	0.18	0.23	22
c5d.2xlarge	19	0.07	0.09	58
c5d.4xlarge	18	0.20	0.20	25
c3.8xlarge	18	0.04	0.05	84
c5d.9xlarge	10	0.06	0.06	54
c5d.12xlarge	17	0.12	0.19	26
c5d.18xlarge	12	0.06	0.06	84
c5d.24xlarge	16	0.05	0.07	60

Further, based on the mentioned steps in the Algorithm, the predictive value for the PH levels is compared with the actual PH level values. In this process, the correction factors are also calculated, as shown in Table 3.

Table 3: Prediction of PH levels

Instance type	Initial PH level	Actual PH (Next Iteration) level	Predicted PH level	Correction factor
c5d.xlarge	13	12.95	13.61	0.654
c5d.2xlarge	19	18.98	19.13	0.152
c5d.4xlarge	18	18.00	18.65	0.652
c3.8xlarge	18	17.99	18.19	0.199
c5d.9xlarge	10	10.00	10.79	0.792
c5d.12xlarge	17	16.92	17.40	0.476
c5d.18xlarge	12	12.00	12.97	0.972

(Continued)

Table 3: Continued

Instance type	Initial PH level	Actual PH (Next Iteration) level	Predicted PH level	Correction factor
c5d.24xlarge	16	15.98	16.44	0.454

As this Algorithm is an iterative process, the correction factors shall become zero in further iterations, and the actual PH level shall equate with the predicted PH levels. Thirdly, the load prediction algorithm is elaborated.

Algorithm 3: Prediction of Computational Load using Correlation Method (PCLC) Algorithm

```
Input:
VM List as VL [], List of Constants of Weight for the Instance Capacity as CW [], List of Constants of Weight for the Instance Memory Capacity as MW [], List of Constants of Weight for the Storage Capacity as SW [], List of Constants of Weight for the Instance Bandwidth as BW [] and outputs from Algorithm – I as sum (SP), sum (SM), sum (SS), sum (SN)
Output: L(t+1) as predicted load
Step-1.
    For every element of V [] as V[i]
        a. Build a load of computing set as SP = VL[i]. CW[i]
        b. Build the load of memory set as SM = VL[i].MW[i]
        c. Build a load of storage set as SS = VL[i]. SW[i]
        d. Build the load of the network set as SN = VL[i].BW[i]
        e. Build the regression formulation coefficients as
        f. B1 = CW[i]/SP, B2 = MW[i]/SM, B3 = SW[i]/SS and B4 = BW[i]/SN
        g. Build the predicted load instance as L(t+1) = B1.VL[i]. SP + B2.VL[i].SM + B3.VL[i].
         SS + B4.VL[i].SN
Step-2.
    Return L(t+1) for each VL []
```

The benefits of the Algorithm are better understood with the help of sample data. As a starting point, this data is referred to as Table 4.

Table 4: Initial data for evaluation

Instance type	vCPUs	Memory (MiB)	Storage (GB)	Maximum number of network interfaces	CW	MW	SW	NW
c5d.xlarge	2	4096	50	3	15	15	16	2
c5d.2xlarge	4	8192	100	4	17	20	20	1
c5d.4xlarge	8	16384	200	4	17	16	18	2
c3.8xlarge	16	32768	400	8	17	16	15	1
c5d.9xlarge	32	61440	640	8	16	17	19	1
c5d.12xlarge	36	73728	900	8	15	15	18	1
c5d.18xlarge	48	98304	1800	8	15	16	17	2

(Continued)

Table 4: Continued

Instance type	vCPUs	Memory (MiB)	Storage (GB)	Maximum number of network interfaces	CW	MW	SW	NW
c5d.24xlarge	72	147456	1800	15	18	18	18	1

The parallel research results recommend deriving the weight constants from the source course. For this location, and under the assumption that the c5d.xlarge available physical pool is overburdened, a load prediction is performed, as shown in Table 5.

Table 5: Load prediction

Instance type	Predicted vCPUs	Predicted memory (MiB)	Predicted storage (GB)	Predicted maximum number of network interfaces
c5d.xlarge	30	61440	800	6

The strength calculations are performed in the next step of the Algorithm. Over or justified capacity is shown by the positive numbers, while under capacity is shown by the negative numbers in Table 6.

Table 6: Capacity calculations

Instance type	vCPUs	Memory (MiB)	Storage (GB)	Maximum number of network interfaces
c5d.2xlarge	−26	−53248	−700	−2
c5d.4xlarge	−22	−45056	−600	−2
c3.8xlarge	−14	−28672	−400	2
c5d.9xlarge	2	0	−160	2
c5d.12xlarge	6	12288	100	2
c5d.18xlarge	18	36864	1000	2
c5d.24xlarge	42	86016	1000	9

As a result, c5d.12xlarge, c5d.18xlarge, and c5d.24xlarge are the migration options that can be considered. The live vm migration methodology for task scheduling is presented as a final example.

The benefits of the Algorithm are better understood with the help of sample data. As a starting point, this data is referred to in Table 7.

Based on the capacity, it is natural to realize that c5d.9xlarge, c5d.12xlarge, c5d.18xlarge, and c5d.24xlarge are the feasible migration options. Thus, based on the proposed optimization fitness function, the capacity must be optimal, and the pheromone level must be highest. Hence, c5d.9xlarge is selected for the migration destination. Hence, based on the complete analysis, it is observed that the most cost-effective physical resource pool is selected for the migration. The more relevant proof is generated in the next Section of this work.

Algorithm 4: Load Balancing by Predictive Corrective Coefficient and Correlative Prediction (LB-PCC-CP)

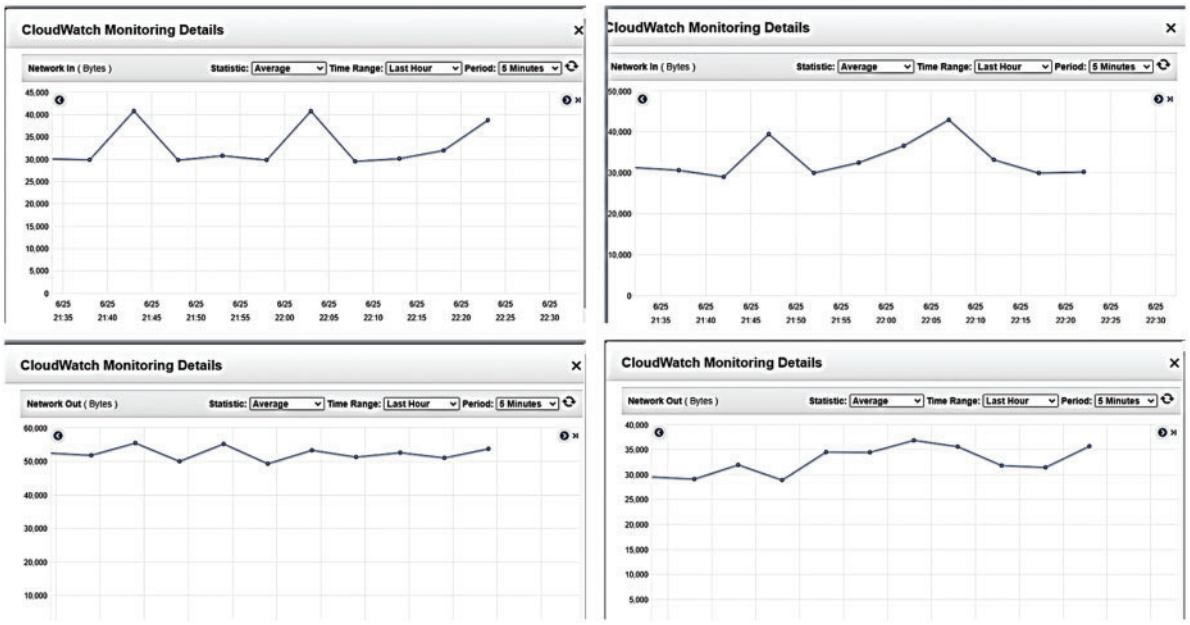
```
Input: V: List of VMs, sum (SP), sum (SM), sum (SS), sum (SN): From Algorithm 1, PH[t+1]: From Algorithm 2(CCPLP), L[t+1]: From Algorithm 3(CBLP)
Output: V(t+1) as Destination
   Step-1.
      For each V[i]
         a. Calculate the threshold, $TH = (SP + SM + SS + SN) - L(t)$
         b. If $L[t+1] > TH$
         c. Then,
               i. Calculate the fitness function, $FF = Max (Cap[t+1]), Max (PH[t+1])$
   Step-2.
      Sort the V [] based on FF
   Step-3.
      Select the optimal V[x] based on FF
   Step-4.
      Start Migration
```

Table 7: Pheromone level and optimal capacity

Instance type	Predicted vCPUs	Predicted memory (MiB)	Predicted storage (GB)	Predicted maximum number of network interfaces	Calculated PH	On-demand linux pricing (USD per Hour)
c5d.xlarge	−26	−53248	−700	−2	19.135	0.224
c5d.2xlarge	−22	−45056	−600	−2	18.652	0.448
c5d.4xlarge	−14	−28672	−400	2	18.187	0.896
c3.8xlarge	2	0	−160	2	10.792	2.117
c5d.9xlarge	6	12288	100	2	17.399	2.016
c5d.12xlarge	18	36864	1000	2	12.972	2.688
c5d.18xlarge	42	86016	1000	9	16.437	4.032
c5d.24xlarge	66	135168	2800	9	11.252	5.376

Fig. 1 shows the visualization of the continuous improvements.

Henceforth, after the detailed discussion of the proposed Algorithm, in the next Section of the work, the results are analyzed.



CloudWatch Monitoring Details

Network In (Bytes) Statistic: Average Time Range: Last Hour Period: 5 Minutes

CloudWatch Monitoring Details

Network In (Bytes) Statistic: Average Time Range: Last Hour Period: 5 Minutes

CloudWatch Monitoring Details

Network Out (Bytes) Statistic: Average Time Range: Last Hour Period: 5 Minutes

CloudWatch Monitoring Details

Network Out (Bytes) Statistic: Average Time Range: Last Hour Period: 5 Minutes

Figure 1: Visualization of the continuous improvements

4 Comparative Analysis

After understanding the load summarization and balancing process, primitive and recent methods for optimizations with the fundamental bottlenecks, formulating the problems to be solved, the mathematical models for the proposed solutions, and proposed algorithms based on the mathematical models, in this Section of the work, the obtained results are furnished. The results are simulated in controlled hardware and tested on the amazon web service platform for better results. The simulation is carried out firstly on CloudSim and AWS EC2 Services. The results obtained from CloudSim are furnished here. During this simulation, the performance measure parameters such as Energy consumption, Number of VM migrations, Violation of SLA, Number of host shutdowns, and total Execution time are considered. The algorithms are simulated over 10 PlanetLab [26] datasets and compared with the primitive algorithms such as “iqr_mmt, iqr_mu, iqr_rs, lrr_mc, lrr_mmt, lrr_mu, lrr_rs, lr_mc, lr_mmt, lr_mu, lr_rs, mad_mc, mad_mmt, mad_mu, mad_rs, thr_mc, thr_mmt, thr_mu and thr_rs”. Finally, in this Section of the work, the proposed method is compared with the other virtual machine migration method with the average value for each parameter discussed. The analysis is furnished here in Table 8.

Table 8: Comparative analysis

Algorithms	Energy consumption (kWh)	Number of VM migrations	Violation of SLA (%)	Number of host shutdowns	Total execution time (sec)
IQR_MMT	45.35	5113	0.22	1499	0.0032
IQR_MU	47.25	5593	0.25	1618	0.0034
IQR_RS	44.55	4806	0.26	1458	0.0028

(Continued)

Table 8: Continued

Algorithms	Energy consumption (kWh)	Number of VM migrations	Violation of SLA (%)	Number of host shutdowns	Total execution time (sec)
LRR_MC	34.35	2203	0.14	685	0.0027
LRR_MMT	35.37	2872	0.13	806	0.0027
LRR_MU	35.38	2808	0.13	816	0.0026
LRR_RS	34.07	2222	0.13	666	0.0022
LR_MC	34.35	2203	0.14	685	0.0024
LR_MMT	35.37	2872	0.13	806	0.0024
LR_MU	35.38	2808	0.13	816	0.0022
LR_RS	34.18	2206	0.14	683	0.0019
MAD_MC	39.31	4124	0.26	1294	0.0033
MAD_MMT	40.60	4669	0.23	1378	0.0033
MAD_MU	42.52	4965	0.26	1476	0.0037
MAD_RS	39.74	4229	0.26	1338	0.0030
THR_MC	27.99	2140	0.24	646	0.0016
THR_MMT	29.02	3006	0.21	761	0.0014
THR_MU	31.34	4073	0.30	989	0.0017
THR_RS	27.62	2007	0.22	599	0.0012
Li et al. [9], 2020	35.37	5593	0.37	816	0.0034
Zatout et al. [22], 2021	47.25	2872	0.31	828	0.0037
Korra et al. [23], 2019	27.99	4229	0.30	874	0.0031
LB-PCC-CP (Proposed)	34.15	2250	0.13	674	0.0019

From the analysis, it is observed that for Energy consumption as low as 34.15 (kWh) as better validation, the proposed method stands in 6th position. For the Number of VM migrations as low as 2250 as better validation, the proposed method stands at 7th position. For Violation of SLA as low as 0.13(%) as better validation, the proposed method stands at 1st position. For the Number of host shutdowns as low as 674 as better validation, the proposed method stands at 4th position. For total Execution time as low as 0.0019 (sec) as better validation, the proposed method stands at the 5th position, ensuring the performance in the upper half during the total analysis. Hence, this is conclusive that the proposed model is a highly stable performance-improved method for load balancing strategy.

5 Conclusion

The current load-balancing systems have nearly explored genetic optimization strategies. As a result, load-balancing solutions can only improve response times up to a degree. These optimization approaches are criticized for being less dynamic, rule-based, and less effective on virtualized resources. To optimize load balance, this study presents a unique approach for predictive load estimate, reduction, or summarization, combining correlation-based parametric reduction and correction coefficient-based pheromone prediction. This work shows the solution of long-standing problems using standard optimization methods, such as highly complex probability distributions, higher complexity of the solution search space for ACO using predictive analysis of the pheromone level, non-coordinated search space problem, which cannot be solved by PSO, and increasing complexity problem while in ACO. The findings show a considerable decrease in response time or time complexity for optimization resulting in fewer SLA violations and a significant reduction in virtual machine migrations compared to other standard benchmarked techniques. From the analysis, we can see that for Energy consumption as low as better validation, the proposed method ranks 6th. For the Number of VM migrations as low as better validation, it ranks 7th. For Violation of SLA as low as better validation, it ranks 1st. For the Number of host shutdowns as low as better validation, it ranks 4th, and for total Execution time as low as better validation, it ranks 1st. So, the proposed model is an improved load-balancing method that is stable and works well.

Acknowledgement: The authors would like to thank the R&D departments of K. L. University, Vaddeswaraam, and B. V. Raju Institute of Technology for supporting this work.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] B. P. Rimal, E. Choi and I. Lumb, "A taxonomy and survey of cloud computing systems," in *2009 Fifth Int. Joint Conf. on INC, IMS and IDC*, Seoul, Korea (South), pp. 44–51, 2009.

[2] S. Patidar, D. Rane and P. Jain, "A survey paper on cloud computing," in *2012 Second Int. Conf. on Advanced Computing & Communication Technologies*, Rohtak, India, pp. 394–398, 2012.

[3] J. Kaur, M. Kaur and S. Vashist, "Virtual machine migration in cloud data centers," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 8, pp. 190–193, 2014.

[4] D. G. Chandra and D. B. Malaya, "A study on cloud os," in *Int. Conf. on Advanced Computing and Communication Technologies*, Rajkot, Gujarat, India, pp. 692–697, 2012.

[5] M. Mishra, A. Das, P. Kulkarni and A. Sahoo, "Dynamic resource management using virtual machine migrations," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 34–40, 2012.

[6] A. Agarwal and S. Raina, "Live migration of virtual machines in the cloud," *International Journal of Scientific and Research Publication*, vol. 2, pp. 1–5, 2012.

[7] M. R. Barzoki and S. R. Hejazi, "Pseudo-polynomial dynamic programming for an integrated due date assignment, resource allocation, production, and distribution scheduling model in supply chain scheduling," *Applied Mathematical Modelling*, vol. 39, no. 12, pp. 3280–3289, 2015.

[8] M. Akbari and H. Rashidi, "A multi-objectives scheduling algorithm based on cuckoo optimization for task allocation problem at compile time in heterogeneous systems," *Expert Systems with Applications*, vol. 60, pp. 234–248, 2016.

[9] Y. Li, Y. Wen, D. Tao and K. Guan, "Transforming cooling optimization for green data center via deep reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 50, no. 5, pp. 2002–2013, 2020.

[10] J. Zhou, Y. Zhang, L. Sun, S. Zhuang, C. Tang *et al.,* "Stochastic virtual machine placement for cloud data centers under resource requirement variations," *IEEE Access*, vol. 7, pp. 174412–174424, 2019.

[11] S. Yan, Y. Zhang, S. Tao, X. Li and J. Sun, "A stochastic virtual machine placement algorithm for energy-efficient cyber-physical cloud systems," in *2019 Int. Conf. on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Atlanta, GA, USA, pp. 587–594, 2019.

[12] V. Medina and J. M. García, "A survey of migration mechanisms of virtual machines," *ACM Computer Surveys*, vol. 46, no. 3, pp. 1–30, 2014.

[13] W. Wen, C. Wang, D. Wu and Y. Xie, "An act-based scheduling strategy on load balancing in a cloud computing environment," in *2015 Ninth Int. Conf. on Frontier of Computer Science and Technology*, Dalian, pp. 364–369, 2015.

[14] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei *et al.,* "Minimizing cost and makes pan for workflow scheduling in the cloud using fuzzy dominance sort based HEFT," *Future Generation Computer Systems*, vol. 93, no. 5, pp. 278–289, 2019.

[15] X. Zhang, T. Wu, M. Chen, T. Wei, J. Zhou *et al.,* "Energy-aware virtual machine allocation for the cloud with resource reservation," *Journal of Systems and Software*, vol. 147, no. 5, pp. 147–161, 2019.

[16] V. Selvi and R. Umarani, "Comparative analysis of ant colony and particle swarm optimization techniques," *International Journal of Computer Applications*, vol. 5, no. 4, 2010.

[17] M. Juneja and S. K. Nagar, "Particle swarm optimization algorithm and its parameters: A review," in *2016 Int. Conf. on Control, Computing, Communication and Materials (ICCCCM)*, Allahabad, pp. 1–5, 2016.

[18] Y. Xu, P. Fan and L. Yuan, "A simple and efficient artificial bee colony algorithm," *Mathematical Problems in Engineering*, vol. 2013, no. 14, pp. 1–9, 2013.

[19] M. H. Kashan, N. Nahavandi and A. H. Kashan, "DisABC: A new artificial bee colony algorithm for binary optimization," *Applied Soft Computing*, vol. 12, no. 1, pp. 342–352, 2012.

[20] Y. Zhang and J. Sun, "Novel efficient particle swarm optimization algorithms for solving qos-demanded bag-of-tasks scheduling problems with profit maximization on hybrid clouds," *Concurrency Computation Practice Experience*, vol. 29, no. 21, pp. e4249, 2017.

[21] S. Sharma, D. S. Verma, D. K. Jyoti and D. Kavita, "Hybrid bat algorithm for balancing load in cloud computing," *International Journal of Engineering & Technology*, vol. 7, no. 4, pp. 26, 2018.

[22] M. S. Zatout, A. Rezoug, A. Rezoug and K. Baizid, "Optimization of fuzzy logic quadrotor attitude controller—particle swarm, cuckoo search and bat algorithms," *International Journal of Systems Science*, vol. 52, no. 4, pp. 883–908, 2021.

[23] S. Korra, D. Vasumathi and A. Vinayababu, "A framework for software component reusability analysis using flexible software components extraction," *International Journal of Recent Technology and Engineering*, vol. 8, no. 2S3, pp. 1359–1367, 2019.

[24] M. Labbadi, Y. Boukal and M. Cherkaoui, "Path following control of quadrotor uav with continuous fractional-order super twisting sliding mode," *Journal of Intelligent & Robotic Systems*, vol. 100, no. 3, pp. 1429–1451, 2020.

[25] M. K. A. Sharman, B. J. Emran, M. A. Jaradat, H. Najjaran, R. A. Husari *et al.,* "Precision landing using an adaptive fuzzy multi-sensor data fusion architecture," *Applied Soft Computing*, vol. 69, no. 4, pp. 149–164, 2018.

[26] P. Kollapudi, S. Alghamdi, N. Veeraiah, Y. Alotaibi, S. Thotakura *et al.,* "A new method for scene classification from the remote sensing images," *Computers, Materials & Continua*, vol. 72, no. 1, pp. 1339–1355, 2022.

[27] W. Jun and S. Xiaowei, "Research on sdn load balancing of ant colony optimization algorithm based on computer big data technology," in *2022 IEEE Int. Conf. on Advances in Electrical Engineering and Computer Applications (AEECA)*, Dalian, China, pp. 935–938, 2022.

[28] A. Yadav, S. J. Goyal, R. S. Jadon and R. Goyal, "Energy efficient load balancing algorithm through metahuristics approaches for cloud-computing-environment," in *2022 Int. Mobile and Embedded Technology Conf. (MECON)*, Noida, India, pp. 130–135, 2022.

[29] A. Kaushik and H. S. A. Raweshidy, "A hybrid latency- and power-aware approach for beyond fifth-generation internet-of-things edge systems," *IEEE Access*, vol. 10, pp. 87974–87989, 2022.