According to Eq (7), we obtain:

$$\begin{aligned} & LR\left(r_{vp}, r_{pm}, \gamma, \beta\right) \\ & = \sum_{pm:pm \in PM} \left\{ UT_{pm}\left(r_{pm}(t)\right) - \gamma_{pm} r_{pm}(t) \right\} \\ & \quad + \sum_{pm:pm \in PM} \sum_{vm:vm \in VM(PM)} r_{vp}(t)\left(\gamma_{pm} - \beta_{pm}\right) \\ & \quad + \sum_{pm:pm \in PM} \beta_{pm}\left(Capacity_{\ pm} - \varepsilon_{pm}^2 \right) \end{aligned} \tag{15}$$

Where $Capacity_{\ pm} - \varepsilon_{pm}^2$ represents the occupied resource of physical machine pm. To increase the PM utilisation and minimise server energy consumption, optimal packing is performed to place virtual machines then $\beta_{pm}\left(Capacity_{\ pm} - \varepsilon_{pm}^2 \right)$ can be considered as the gains of PM from packing.

To save energy any PMs that are not' in use should be turned off. The power consumption of active PMs is formulated as follows:

$$\sum_{j=1}^{k} PC\left(PM_{cpu_j}\right) = \begin{cases} PM_{idle} + (PM_{max} - PM_{idle}) \times PM_{cpu_j}, \text{ if } PM_{cpu_j} > 0 \\ 0, \ otherwise \end{cases} \tag{16}$$

where P_{idle} is the idle-state power of PM_j , P_{Max} is the maximum power of PM_j , and P_{cpuj} is the percentage value $\in$ [0, 1] that denotes the CPU utilisation.

Harris Hawk Optimisation for virtual machine placement

Heidari and Mirjalili et al. developed the Harris Hawks Optimisation Technique (HHO), a novel optimisation algorithm [47]. The algorithm mimics the natural behaviour and hunting strategy of Harris Hawks known as surprise pounce. The hawks collaborate to attack from many directions to startle the victim. Harris Hawks reveal a variety of pursuit methods dependent on the nature of the schemes and the victim's' evasive patterns. Exploration and exploitation tactics are proposed by the conventional HHO algorithm which is driven by exploring prey, surprise pounce, and Harris Haw's' particular attacking approach. The Harris Hawks are the candidate solutions and the targeted prey in each phase is the best candidate solution (nearly the optimal one). The exploration phase, transition from exploration to exploitation phase, and exploitation phase are the three phases of the HHO algorithm and are described below.

i) Exploration phase. All Harris Hawks are considered solutions during this phase denoted as a solution matrix H; is a $2 \times N$ matrix, where N is the number of VMs. Fig 2 depicts 4 feasible solutions where four hawk agents' solutions assign 5 VMs to 3 PMs. The four solutions are represented by the set $H = H_1, H_2, H_3, and\ H_4$. According to the number of VMs, all solutions are reviewed and ranked in ascending order of PMs that are utilised in this solution [12]. The order of sorting is H_3, H_1, H_4, and H_2. The best solution is H3 because it has the minimum number of PMs, which consumes less power.

The solution matrix of a single hawk H is defined in Eq 17. If there are *N* VMs to be allocated to *J* PMs, the value of the component of H of the matrix represents the PM index to host

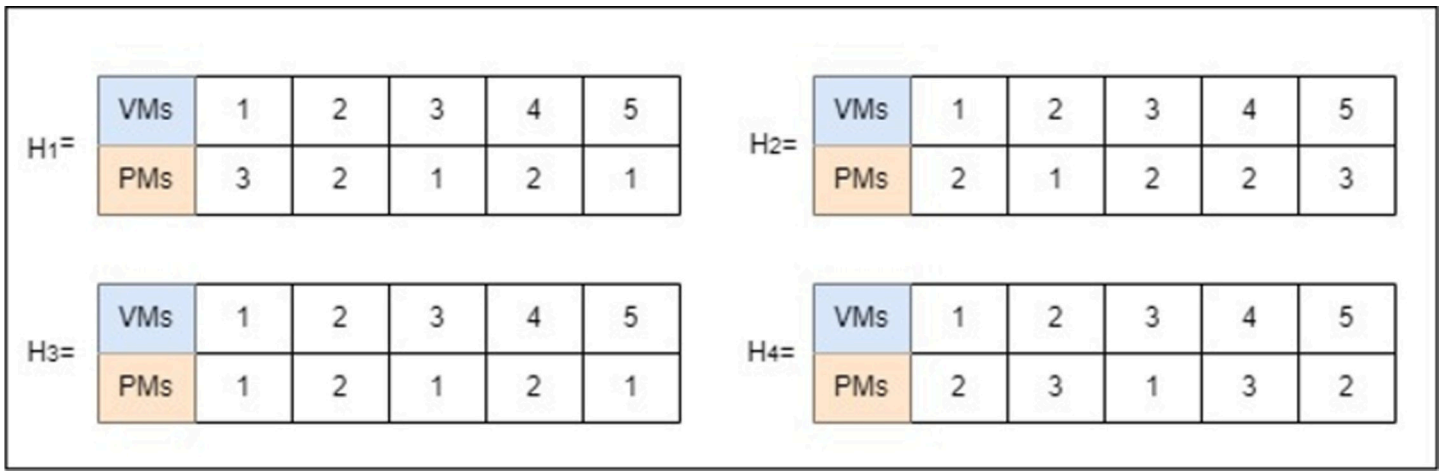

Fig 2. Example of four feasible applicable solutions.

https://doi.org/10.1371/journal.pone.0289156.g002

the VM number *n*.

$$H = \begin{bmatrix} y_1^i, y_2^i, y_{3,}^i, \ldots . y_N^i \\ y_{1,1}^{i,j}, y_{1,2}^{i,j}, y_{2,3}^{i,j}, \ldots . y_{1,N}^{i,j} \end{bmatrix} \tag{17}$$

The corresponding variable $y_{1,n}^{i,j}$ is the value j, $\forall\, j \in [1, J]$, if the virtual machine VM_i is assigned to $PM_j \in PM$. Variable y_n^i denotes the virtual machine $VM_i \in VM$.

The fitness value is determined for all these feasible solutions created on the desired prey in each iteration. To replicate Harris Hawk's' exploring abilities in the search space chosen and updating the solution matrix is according to two techniques as specified in Eq 18.

$$H(t+1) = \begin{cases} H_{rand}(t) - r_1 |H_{rand}(t) - 2r_2 H(t)| & p \geq 0.5 \\ \left(H_\rho(t) - H_s(t) - r_3 (LB + r_4 (UB - LB)\right) & p < 0.5 \end{cases} \tag{18}$$

Where $H(t+1)$ is the hawk's candidate solution/position in the second iteration t, $H_\rho(t)$ represents the best solution matrix/prey position and $H_{rand}(t)$ is the random solution selected in the present population.$H(t)$ represents the position vector of hawks in the present iteration t. r_1, r_2, r_3, r_4 and p are the random scaled factor within the range [0,1]. UB and LB are the upper bound and lower bounds of the variables, $H_s(t)$ denotes the average number of solutions. The index of the first row is updated according to the hawk agent solutions of the best and random solution as per Eq 18.

Hawk placements are generated because of this method inside UB and LB depending on 2 rules 1) Build solutions using a randomly chosen hawk from the present population as well as other hawks. 2) Construct solutions depending on the location of the prey, the average hawk position, and random scaled variables. While r3 is a scaling factor, if the value of r4 approaches 1 it will aid in increasing the rule's' unpredictability. An arbitrarily scaled measure length is added to LB in this rule. More diversification strategies to investigate other sections of the feature space are examined using a random scaled component. The average hawk position

(solutions) is formulated as follows:

$$H_s(t) = \frac{1}{M}\sum_{i=1}^{M} H_i(t) \tag{19}$$

Where, $H_s(t)$ is the current iteration's' average number of solutions. M denotes all possible solutions. $H_i(t)$ indicates the location of every solution in iteration t.

The updated indexes in the hawk solutions should be in the range of [1,J]. If the updated index is outside of the range then the algorithm recalculates it as follows:

$$y_{1,n}^{i,j}(t+1) = \begin{cases} y_{1,n}^{i,j}(t+1) mod\, J, & if\ y_{1,n}^{i,j}(t+1) \notin [1, J] \\ y_{1,n}^{i,j}(t+1), & otherwise \end{cases} \tag{20}$$

ii) The transition from exploration to exploitation. Based on the energy of the prey, this phase shows how HHO moves from exploration to exploitation. HHO posits that the energy of the prey is gradually depleted as a result of the fleeing activities. R_0 is the initial energy range between [−1,1] as expressed in Eq 21.

$$R = 2R_0\left(1 - \frac{t}{T}\right), R_0 \in [-1, 1] \tag{21}$$

Where t is the current iteration and T represents the maximum number of iterations.

iii) Exploitation phase. The exploitation phase is marked completed by utilising 4 methods/ways at parameter sets. These methods are created on the position that was discovered during the exploration phase. The prey, on the other hand, tries to flee often despite the hawk's 'efforts to track it down and trap it. HHO exploitation uses four different techniques to imitate the hawks' attacking style. Hard besiege, Soft besiege, soft besiege with progressive rapid dives, and hard besiege with progressive rapid dives are the four methods. These methods depend on 2 factors r and |R|. Where R represents the prey" fleeing energy and r is the likelihood of escaping with $r < 0.5$ indicating a better chance of the prey escaping successfully and $r \geq 0.5$ indicating an unsuccessful escape. The following is a summary of these approaches:

In the soft besiege approach, where $r \geq 0.5$ *and* $|R| \geq 0.5$, while the hawks gently round on the victim causing it to lose extra energy before completing the surprise pounce the prey still has some energy to flee. Soft besiege is mathematically formulated in Eq 22.

$$H(t + 1) = \Delta H(t) - R|JH_\rho(t) - \mathrm{H}(t)| \tag{22}$$

$$\Delta H(t) = H_\rho(t) - \mathrm{H}(t)$$

$$J = 2(1 - r_5),\ r_5 \in [0, 1]$$

Where $\Delta H(t)$ is the difference between the prey" position vector and the current location in iteration t, J is the ' prey" jump strength, and r_5 is a random variable.

In the hard besiege, where $r \geq 0.5$ *and* $|R| < 0.5$, the prey is exhausted and has a slight chance of escaping. In this situation, the hawk barely encompasses the target before launching the ultimate surprise pounce. Accordingly, the solution is updated using Eq 23.

$$H(t+1) = H_\rho(t) - R|\Delta H(t)| \tag{23}$$

In soft besiege with progressive rapid dives method with $r < 0.5$ *and* $|R| \geq 0.5$, the prey has enough energy to flee. The hawk manoeuvres deftly around the victim and descends tolerantly

before the attack. This is referred to as a clever soft besiege, in which the hawk's 'location is updated in two phases. In t^{he} 1st stage, the hawks approach the prey by calculating the prey" next move as in Eq 24.

$$K = H_{\rho}(t) - R\,|\,C\,H_{\rho}(t) - H(t)| \tag{24}$$

C represents the jump power of prey. The hawk then determines whether to dive in the second stage depending on a comparison of the prior dive and the likely outcome. If it is not,' the hawks will produce an uneven dive based on the Levy Flight (LF) notion as expressed in (25)

$$L = K + Q * LF(d) \tag{25}$$

Where d is the dimension of solutions, Q is the random vector of size 1*d. LF is the Levy Flight function designed using Eq 26

$$LF(d) = 0.01 * \frac{y * \sigma}{|z|^{\frac{1}{\beta}}},\ \sigma = \left(\frac{\tau(1+\beta) * \sin\left(\frac{\pi\beta}{2}\right)}{\tau\left(\frac{1+\beta}{2}\right) * \beta * 2^{\left(\frac{\beta-1}{2}\right)}}\right)^{1/\beta} \tag{26}$$

Where β is the default constant and y, z are the random variables between [0, 1]. As a result, a method for updating the Harris Hawk's 'locations with advanced quick dives may be devised as

$$H(t+1) = \begin{cases} K \text{ if } F(K) < F(H(t) \\ L \text{ if } F(Z) < F(H(t) \end{cases} \tag{27}$$

Here, K and L are performed using Eqs 24 and 25.

In the last approach, hard besiege with progressive rapid dives where $r < 0.5$ *and* $|R| < 0.5$, the prey has no energy to flee, therefore the Harris Hawks try to approach the prey by diving quickly before making a surprise pounce to grab it. The hawk movement' in the situation is stated in Eq 27

Where K and L are as follows

$$K = H_{\rho}(t) - R\,|\,CH_{\rho}(t) - H_{s}(t)\,| \tag{28}$$

$$L = K + Q * LF(d) \tag{29}$$

The parameters used in this work are presented in Table 3.

Objective function. After the hawk agents' solutions have been updated the solutions are evaluated to choose the best one H_{ρ}. Only one HW (Hawk) solution is chosen as the best where HW denotes the number of hawk agents. The algorithm compares the solutions based on criteria e_1 *in* Eq 30, this shows the amount of energy consumed by this solution. The best solution is the H_{ρ} solution with the least power consumption and the fewest number of PMs

Table 3. Parameters of HHO.

Name of the parameter	Adopted Value
Number of search agents	50
Dimension	Number of VMs
Lower bound	1
Upper bound	800
Maximum Iterations	100

https://doi.org/10.1371/journal.pone.0289156.t003

used [12]. The objective function is formulated as:

$$min.\ e_1(H) = \sum_{j=1}^{J} PC\left(P_{CPU_j}\right) \tag{30}$$

Pseudocode 1: Harris Hawk Optimisation
Input: Set of virtual machines N and physical machines J
Output: $2 \times N$ allotment matrix mapping N virtual machines to J physical machines as H_ρ

1. **Initialisation**: HW: = 50, $ItrE$: = 100, it: = 0, LB: = 1, UB: = 800
2. **while** $it < ItrE$ **do**
3. Produce $HW \times 2 \times N$ hawk solution matrices
4. Assess the solutions and assign the best solution using the *best solution*: = H_ρ
5. initialise a: = 1 for each iteration of the search
6. **if** the solution $\neq H_\rho$ **then**
7. Update the present solution Ha using Eq (18)
8. **if** the number of hawks $a <$ HW **then**
9. set a: = a+1 and go to step 6
10. Assess the fitness of the HW solutions using Eq 30 and assign the best solution to H_ρ
11. **Return** $2 \times N$ allotment matrix H_ρ the best solution.

The pseudocode of HHO is depicted in Pseudocode 1 and the workflow process of the HHO is represented in Fig 3. The hawk agents start with randomly distributed indexes and then analyze their solution to determine the best H_ρ . The hawk agents then update their solutions based on the best option that has been chosen thus far. The optimal solution is then presented as matrix indices which map the VMs to the minimum number of PMs at the end of each iteration. The HHO algorithm has the advantages of simple operation, fewer adjustment parameters, ease of implementation and use of communication between hawks to improve the global search capability. But for higher dimensional problems it may have low converge performance.

Load balancing

To perform the load balancing in the data centre, host overload and host underload detection mechanisms are incorporated.

Host overload detection. Each host periodically executes an overload detection algorithm to de-consolidate VMs when needed to avoid performance degradation and SLA violation. In this work, we used the IQR (Interquartile Range) to detect the overloaded machines in the data centre and the Maximum Correlation (MC) policy [48] is applied to choose the VMs to be migrated from overloaded hosts to some other host. MC selects the VMs having a maximum correlation of the CPU consumption with other virtual machines.

IQR is a method for setting an adaptive CPU utilisation threshold based on robust statistics. In descriptive statistics, the IQR, also called the midspread or middle fifty, is a measure of statistical dispersion. It is equal to the difference between the third and first quartiles: IQR = Q3 —Q1. Unlike the total range, the interquartile range is a robust statistic having a breakdown point of 25% and thus is often preferred to the total range. Using IQR the CPU utilisation

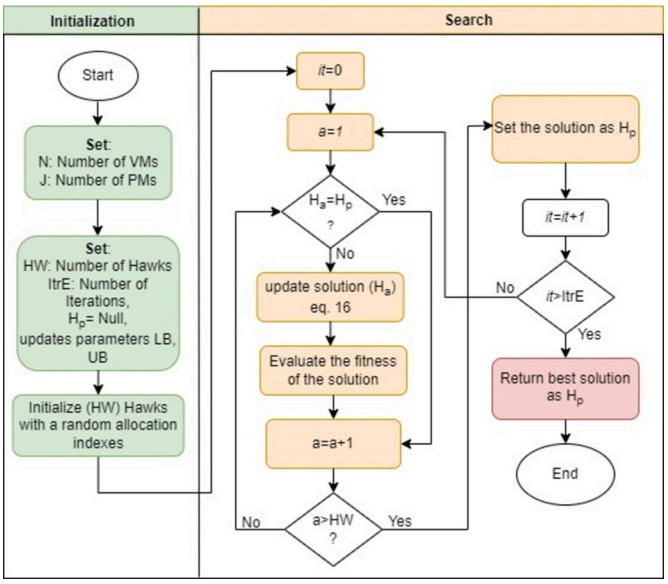

Fig 3. Workflow process of HHO algorithm.

https://doi.org/10.1371/journal.pone.0289156.g003

threshold is defined in Eq 31.

$$T_u = 1 - s * IQR \tag{31}$$

where s is a parameter of the method defining the safety of the method.

Host underload detection. First, all overloaded hosts are identified using the overload detection technique and the VMs that will be migrated are assigned to the destination hosts. The system then attempts to deploy all the VMs from this host onto other hosts with minimal utilisation relative to the other hosts while ensuring that they are not overloaded. The VMs are configured for migration to the determined destination hosts if such a placement is possible. To save energy the source host is put to sleep mode once the migrations are done. The source host is maintained operational if all the VMs from the source host cannot be moved to other hosts. For all non-overloaded hosts, this step is done repeatedly.

Complexity analysis of the proposed method

In further discussion V and P denote the number of VMs and PMs respectively. Each iteration consists of two steps. The first step is to update the solution. The first step includes applying Eq 19 to each column of all solution matrices. As a result of modifying the VM's resource utilisation the PM's CPU and RAM utilisation may rise or decrease. Thus, the time complexity of this step is O(V).

Second, to balance the load across the data centre, overloaded PMs are collected as discussed in the load balancing section. In the worst case when the PM is overloaded, selected VMs are migrated to other PMs. If the underloaded PM is found the PM is switched to sleep mode by migrating the VMs to some other PM. The time complexity of this task is $O(V \times P)$. The PM's CPU and RAM usage increases and decreases with each VM migration from one PM to another according to Eq 30. Thus, the fitness function can be computed by summing up the power consumption of each PM in the time complexity of O(P).

Table 4. Characteristics of servers.

Host Type	Depiction
HP ProLiant G4	1860 MIPS, 2 GB network bandwidth, 4GB RAM and 1.5 GB storage
HP ProLiant G5	2660 MIPS, 4 GB network bandwidth, 4GB RAM, and 2.5 GB storage

https://doi.org/10.1371/journal.pone.0289156.t004

So, the worst-case total time complexity of the 2 steps and the fitness computation of each iteration is $O(V + (V \times P) + P)$. As algorithm 1 has ItrE iterations, the worst time complexity is $O(HW \times \text{ItrE} \times \text{V} \times \text{P})$ for HW hawks.

Experiment and comparisons

This section covers the experimental setup, performance measurements, and experimental outcomes.

Experimental setup

To test the proposed method we used the CloudSim 3.0 toolkit simulator. Cloudsim offers a variety of virtual machine provisioning methodologies and Virtualised resources. We used real workload traces from PlanetLab to conduct the experiment. PlanetLab is a component of the CoMon project, which collects CPU utilisation from over 1000 virtual computers running on various hosts in over 500 locations across the world. We employed four distinct types of virtual computers in our test setup: Micro, Small, Medium, and Extra-Large instances. A total of 600 HP ProLiant G4 and HP ProLiant G5 heterogeneous hosts have been deployed. Table 4 lists the characteristics of these servers.

Performance natrix and results

To assess the proposed approach against other algorithms the workloads depicted in Table 5 were used. It shows the workload number and the number of VMs in each of the workloads. Each of the workload files contains CPU utilisation values measured every 5 minutes in PlanetLab's VMs [49]. These trace files contain traces of CoDeeN, the Coral Content Distribution Network, and Open DHT. The experiment was carried out using the workloads specified and the average result of these workloads was used to evaluate different algorithms based on the five metrics described below.

Experiments were carried out in two scenarios. Firstly, simulation is carried out using the workloads specified and the average result of these workloads is used to evaluate different algorithms based on the metrics under various workloads and task utilisation. For the second scenario, the performance of the proposed algorithm is done with scaling resources to study the performance in underloaded and overloaded conditions.

a) Energy consumption. Energy Consumption represents the total amount of energy/power consumed by all the data centre's PMs. Fig 4 depicts the energy consumption of the

Table 5. Traces of workload from PlanetLab.

Workload		No. of VMs
20110306	W1	898
20110309	W2	1061
20110325	W3	1078
20110412	W4	1054
20110420	W5	1033

https://doi.org/10.1371/journal.pone.0289156.t005

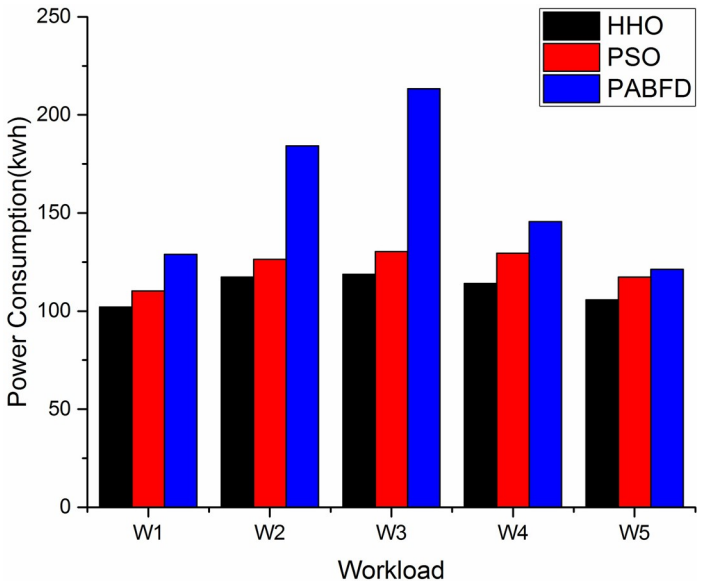

Fig 4. Energy consumption of data centre using different techniques using Scenario 1.

https://doi.org/10.1371/journal.pone.0289156.g004

algorithms using scenario 1. As the instance size grows, the power consumption gradually increases. The result shows that HHO decreases average power consumption by 9% and 27% compared to PSO (Particle Swarm Optimisation) and PABFD (Best Fit Decreasing) respectively. Fig 5 shows the study with scaling resources using scenario 2 where the proposed algorithm shows the least energy consumption. The simulation is supported by two machines HP ProLiant ML110 G4 (Intel Xeon 3,040, 2 cores, 1,860 MHz, 4 GB) and HP ProLiant ML110 G5

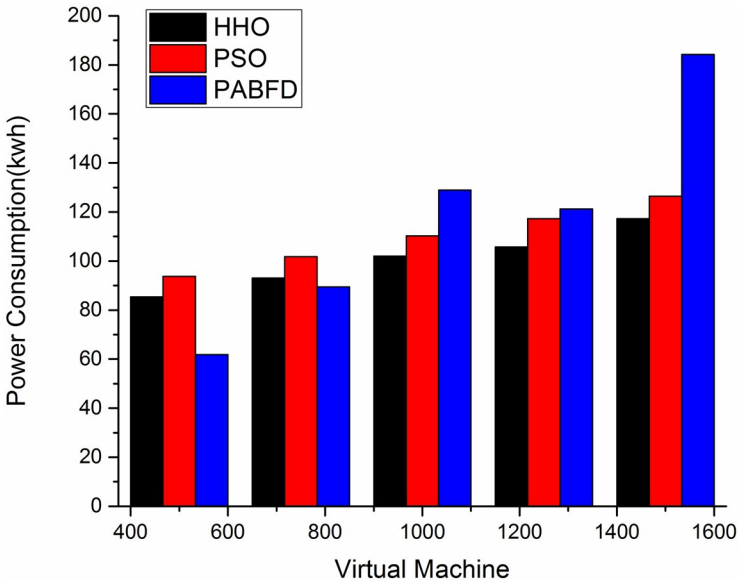

Fig 5. Energy consumption of data centre using different algorithms using Scenario 2.

https://doi.org/10.1371/journal.pone.0289156.g005

Table 6. Energy and utilisation of machine power model.

Server Utilisation	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

https://doi.org/10.1371/journal.pone.0289156.t006

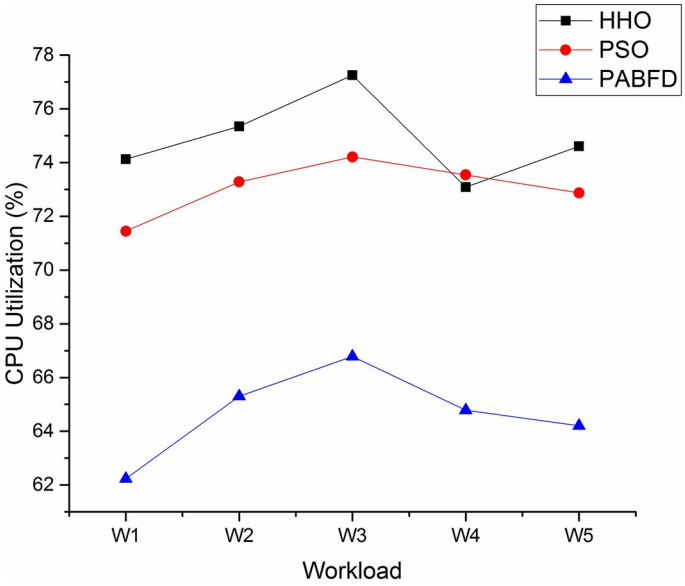

Fig 6. CPU utilisation in the heterogeneous environment using Scenario 1.

https://doi.org/10.1371/journal.pone.0289156.g006

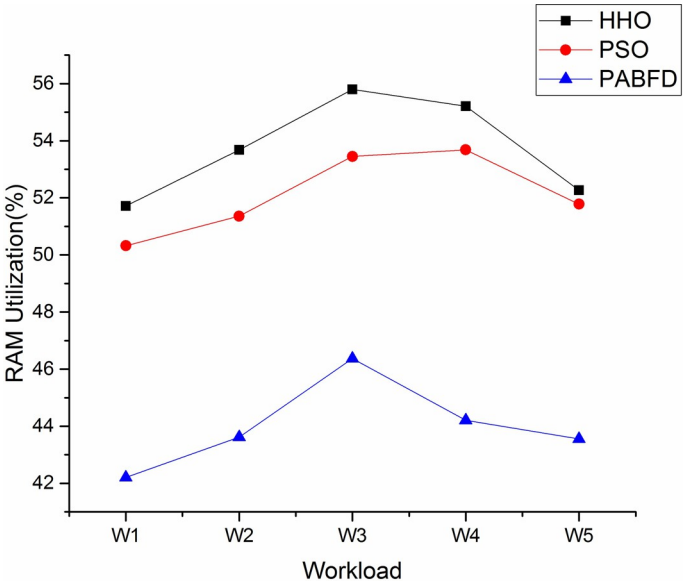

Fig 7. RAM utilisation in the heterogeneous environment using Scenario 1.

https://doi.org/10.1371/journal.pone.0289156.g007

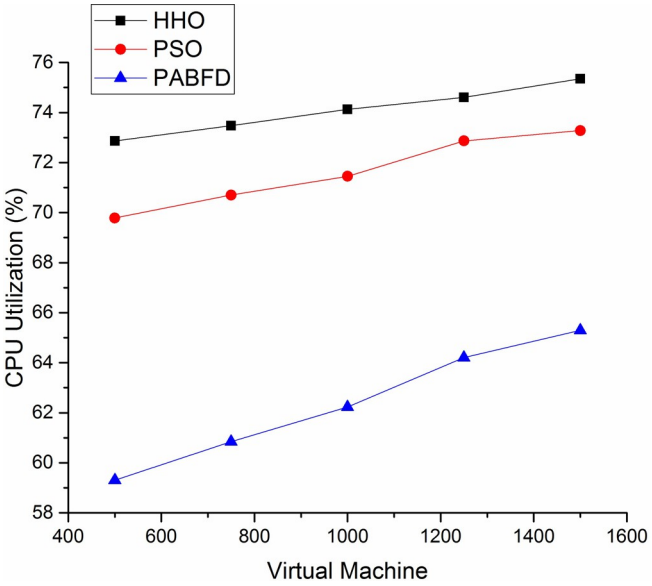

Fig 8. CPU utilisation using Scenario 2.

https://doi.org/10.1371/journal.pone.0289156.g008

(Intel Xeon 3,075, (2 cores, 2,660 MHz, 4 GB) [49] as defined in Cloudsim3.0. Table 6 shows the power consumption by a machine under different utilisation levels.

b) Resource utilisation. Figs 6 and 7 shows the resource utilisation of CPU and memory for the PMs to host VMs using scenario 1. It is observed from the result that the average CPU utilisation of HHO is higher by 3% and 12% compared to PSO and PABFD respectively. Similarly, the memory utilisation of HHO is 4% and 17% higher than PSO and PABFD. Figs 8 and 9 shows the performance study of CPU and RAM utilisation with increasing virtual machines

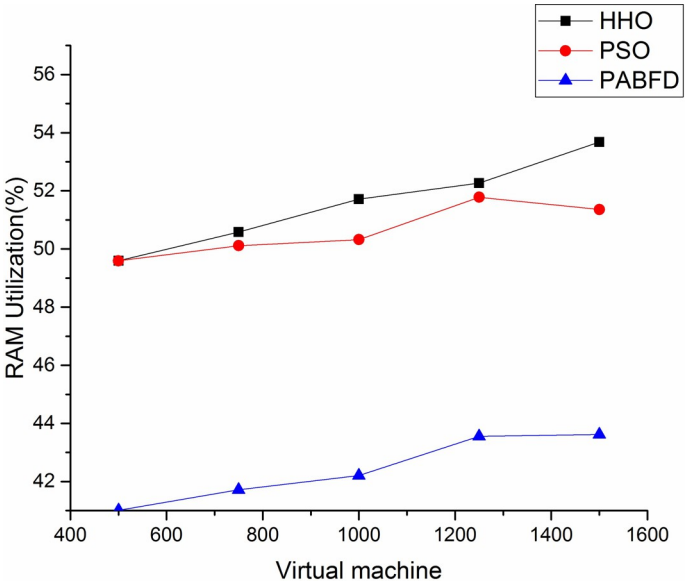

Fig 9. RAM utilisation using Scenario 2.

https://doi.org/10.1371/journal.pone.0289156.g009