*Article*

# Energy-Aware Live VM Migration Using Ballooning in Cloud Data Center

Neha Gupta [1], Kamali Gupta [1,*], Abdulrahman M. Qahtani [2], Deepali Gupta [1], Fahd S. Alharithi [2], Aman Singh [3,4,5] and Nitin Goyal [6]

1   Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura 140401, India
2   Department of Computer Science, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia
3   Higher Polytechnic School, Universidad Europea del Atlántico, C/Isabel Torres 21, 39011 Santander, Spain
4   Department of Engineering, Universidad Internacional Iberoamericana, Arecibo, PR 00613, USA
5   Department of Project Management, Universidad Internacional Iberoamericana, Campeche, CP 24560, Mexico
6   Department of Computer Science and Engineering, Central University of Haryana, Mahendragarh 123031, India
*   Correspondence: kamali.singla@chitkara.edu.in

**Abstract:** The demand for digitization has inspired organizations to move towards cloud computing, which has increased the challenge for cloud service providers to provide quality service. One of the challenges is energy consumption, which can shoot up the cost of using computing resources and has raised the carbon footprint in the atmosphere; therefore, it is an issue that it is imperative to address. Virtualization, bin-packing, and live VM migration techniques are the key resolvers that have been found to be efficacious in presenting sound solutions. Thus, in this paper, a new live VM migration algorithm, live migration with efficient ballooning (LMEB), is proposed; LMEB focuses on decreasing the size of the data that need to be shifted from the source to the destination server so that the total energy consumption of migration can be reduced. A simulation was performed with a specific configuration of virtual machines and servers, and the results proved that the proposed algorithm could trim down energy usage by 18%, migration time by 20%, and downtime by 20% in comparison with the existing approach of live migration with ballooning (LMB).

## 1. Introduction

Cloud computing is trending among customers due to the tremendous advantages that it offers at very low rates [1]. Due to an increase in the number of customers, energy consumption has increased the expenditure of cloud service providers, thus increasing the burden on the pockets of customers. The primary solution for reducing energy usage is virtualization, which provides multiple virtual instances on limited physical machines/servers [2]. The request coming from the client is mapped on these virtual machines that are allocated on the available physical machines. After the allocation of all VMs to the servers, sometimes, there is a need to transfer the virtual machine from one server to another, which comes with the overhead of migration associated with components, which increases energy consumption. The reason for this shifting can be a failure in the server or VM, the overutilization of any server, etc. VM migration is depicted in Figure 1.

VM migration [3] is primarily of two types—(1) offline migration (service is halted for the complete migration time), and (2) online/live migration (service is halted during the time in which the VM gets transferred from source to destination, but the shifting of memory and storage state is performed in "power ON" mode). Among the two types, live VM migration fits the current scenario, as a customer cannot accept extended downtime of service.
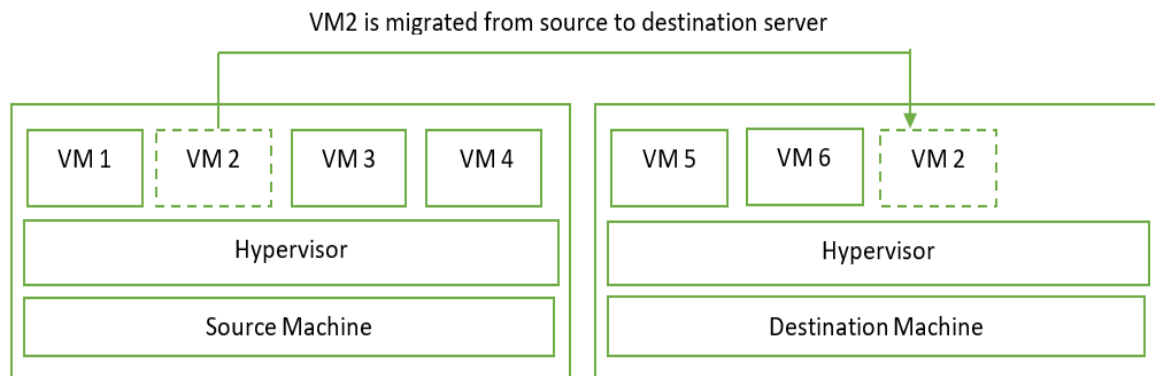
**Figure 1.** VM migration scenario of over-utilized and under-utilized servers.

Techniques used in memory data migration [4] are the following:

- **Pre-copy**—Firstly, the data are transferred while the VM serves the customer. After a certain point, the VM is turned off and transferred to the destination host/server.
- **Post-copy**—Firstly, the VM is turned off and transferred to the destination host/server. Then, the data are transferred while the VM serves the customer.
- **Hybrid**—Data are transferred both before and after the shifting of the VM.

Techniques used in storage data migration [4] are the following:

- **Pre-copy**—Disk blocks are transferred before memory pages.
- **Post-copy**—Disk blocks are transferred after memory pages.
- **Hybrid**—Disk blocks and memory pages are transferred simultaneously.

The energy used in one live migration should be optimized so that total energy of the data center can be controlled. The goal of our research was to minimize the energy of migration by reducing the migration time and downtime. The parameters that have significant importance in live VM migration were considered and evaluated as follows:

- **Energy consumption**—The overall energy usage in the cloud data centers can raise the cost to customers. Virtualization and live VM migration are the primary steps in this regard. Live VM migration, if performed frequently, can affect the system in the reverse direction.
- **Migration time**—It defines the time between the initiation of VM migration and the resumption of the VM on the destination server.
- **Downtime**—It is the time when the VM remains halted so that it can be finally shifted to the destination server.
- **Resource utilization**—It is the measurement of the server that indicates how efficiently the server is used in the data center.
- **Makespan**—It is the measurement of the server regarding the working time, that is, the submission of the first VM till the completion of the last VM.
- **Atomicity**—It is the property whereby migration is completed successfully without disturbing other VMs.
- **Convergence**—It is the point in time when the difference in the memory and storage state between the source and the destination server is almost nil. This signifies a successful copy of data.

The **contribution** of the paper in terms of the above-mentioned parameters is that the new proposed algorithm can reduce energy consumption, downtime, and migration time and can improve resource utilization significantly. The structure of the paper is as follows: Section 2 represents a thorough survey on existing live migration techniques. Section 3 presents the framework, flowchart, and algorithm form of the proposed approaches. Section 4 presents the experimental testbed for validating the proposed algorithm and its comparison with the existing algorithm. Section 5 illustrates the results and their discussion, showing a comparative detailed analysis of the existing and proposed

approaches. Section 6 discusses the workflow of a cloud data center after incorporating the proposed algorithm. Finally, Section 7 presents the concluding remarks.

## 2. Related Work

The authors in [5] combined pre-paging with the post-copy technique of migration through dynamic self-ballooning implemented in Xen. The methodology reduced the network load by 21%. DSB releases free pages from the VM to the hypervisor regularly, considerably speeding up migration with minimal performance damage. The authors in [6] proposed the migration approach "Migration with Data Deduplication (MDD)", which uses data deduplication to increase the performance of migration. The proposed approach is able to minimize the data (memory and storage) transferred, migration time, and downtime. The authors of [7] reported live VM migration in VMware ESX with three approaches to storage migration, i.e., "Snapshotting (in ESX 3.5), Dirty block tracking (in ESX 4.0/4.1) and IO Mirroring (in ESX 5.0)".

The authors of [8] proposed "Server consolidation algorithm-Sercon", which can minimize the count of working servers as well as the count of VM migrations as compared with the "first-fit decreasing algorithm"; an experiment was performed on the .NET framework. The authors of [9] proposed two algorithms: "VM placement Optimization", which finds overloaded and underloaded hosts, and "Power Aware Best Fit Decreasing (PABFD)", which reduces the energy consumption in the migration process. The authors of [10] worked on reducing the power required for live VM migration and proposed two "power-capping" schemes for reducing sudden power shooting during migration. The authors of [11] proposed the "memory-compression-based VM migration approach (MECOM)" approach, which uses a memory compression technique to improve VM migration and was found to be effective in reducing migration time, downtime, and the data to be transferred.

The authors of [12] introduced ballooning to manage the memory on a server by deleting unused memory pages from time to time. They also proposed a "skewness" algorithm for load balancing. The authors of [13] proposed "introspection-based memory pruning" to improve live migration by minimizing unnecessary pages. This procedure reduces network traffic and migration time. The authors of [14] proposed the storage migration approach "LayerMover" using the data deduplication method for optimization. It was proved to be better than the previously used data deduplication approach. The authors of [15] proposed a "memory prediction technique" that chooses pages to migrate based on the dirty rate of the pages. In the proposed technique, the total migration time is reduced, but the downtime is not affected.

The authors of [16] discussed memory optimization techniques—"Virtual Swap Management Mechanism (VSMM) ", "ESX by VMware", "Transparent Page Sharing (TPS)", and "Database Optimization" and "Ballooning"—and their merits. The authors of [17] proposed the "Least recently used (LRU) stack distance-based delta compression" algorithm for effective live migration using the compression and prediction technique. The authors of [18] suggested strategies for multiple VM migrations such as "improved serial migration strategy" and "m mixed migration strategy". They also introduced queuing models such as "M/M/C/C" and "M/M/C" to measure the output. The authors of [19] presented a survey of all existing live VM migration techniques and discussed the objectives, base techniques, advantages, and performance metrics of VM migration. The authors of [20] introduced the concept of ballooning in live migration and showed the improvement of the maximum and minimum migration time.

The authors of [21] tested the working of migration in openstack in a situation of high system constraints and network issues. They performed a performance analysis of migration, which could help researchers to find better optimization methods. The authors of [22] presented a survey of existing live migration techniques, along with the name of the hypervisor, performance parameters, and basic technique (pre-copy, post-copy, and hybrid) used, and pros and cons, if any. The authors of [23] presented a taxonomy of live VM migration, which provides details of "planning and scheduling algorithms",

"performance and cost models", "migration generation in resource management rules", "management lifecycle and orchestration", and "assessment methodologies". The paper also mentioned and reviewed the current state of "migration scheduling types", "migration network awareness", "scheduling scope", "heterogeneous and homogeneous solutions", and "scheduling objectives". The authors of [24] proposed a security-based technique called "VMshield", which can protect virtual machines from outside attacks, and found that it can provide more security with less storage requirement than existing security methods. The author of [25] proposed a VM consolidation technique that is based on the "Ant Colony Optimization" method to provide a better VM placement technique. The authors of [26] proposed a "BoT scheduling algorithm" to maximize the profit of cloud service providers using the user-specified deadline for each request.

The authors of [27] proposed a game approach to VM migration for multi-tier software in the IaaS model and its fault detection technique. The authors of [28] proposed a resource management algorithm named "RU-VMM" using a VM migration technique. The paper also considered successful and dropped migration for calculating the threshold for resource utilization. The authors of [29] proposed a VM migration algorithm named "V2PQL" based on the Markov decision technique while maintaining the values of resource utilization and load balancing. Load balancing is defined as

$$\sigma_j = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left( \frac{e_j^{(i)}}{e_j} - 1 \right)} \tag{1}$$

Moreover, resource utilization is defined as

$$H^{(i)} = \sqrt{\sum_{j=1}^{k} \left( \frac{u_j^{(i)}}{u^{(i)}} - 1 \right)} \tag{2}$$

The authors of [30] worked on minimizing total migration time and proposed three algorithms—"Host Selection Migration Time (HSMT), VM Reallocation Migration Time (VMRMT), and VM Reallocation Bandwidth Usage (VMRBU)". The proposed algorithms decreased the migration time by a maximum of 50%, which can be calculated as

$$\text{Total time consumed} = \propto \times A(\text{HSMT}) + \beta \times B(\text{VMRMT}) + \gamma \times C(\text{VMRBU}) \tag{3}$$

The authors of [31] proposed "Enhanced artificial bee colony (PEA)" for a better migration of virtual machines in two phases. The total migrations were also reduced by 25%. The achieved energy reduction was 13%.

The next section presents the proposed algorithm and comparisons with the existing algorithm.

## 3. Proposed Algorithm

After an intense literature survey, many techniques were studied that can make live VM migration more energy efficient. One of the techniques is ballooning, which is a procedure used to delete unused data before migrating the virtual machine. It saves more time than other techniques, as it directly decreases the data that need to be transferred. It is achieved by checking the number of accesses made to the data (memory page or disk block) during the execution of the virtual machine, and all the data with low values are considered target data for deletion.

The steps of the existing ballooning algorithm [19] for live VM migration are the following:

- Before migration, execute ballooning, which is the process to delete unused data (memory pages and storage disk blocks) from the VM storage.
- Pre-copy to migrate the current state of memory pages and storage disk blocks.

- Power off the VM, transfer it to the target server, and restart.
- Post-copy to migrate the memory pages and disk blocks left.
- Calculate the migration time and downtime.

The drawback of the algorithm is that ballooning is not able to differentiate between a page created long ago having the least number of accesses and a page which has just been created and has zero accesses. Hence, in the proposed algorithm, the time for the generation of each data is also taken into account, so that just-created data that can be of significant importance are not considered unused.

The **novelty** of the proposed algorithm consists in the following:

- Ballooning is combined with the least recently used (LRU) page replacement technique.
- The pre-copy step is discarded to reduce the transmission time.
- The setup of the VM at the destination is performed parallelly with ballooning to save time by sending the configuration details of the VM from the source to the destination server beforehand.

The list of abbreviations mentioned in the algorithms is shown in Table 1.

**Table 1.** Abbreviation list.

| Abbreviation | Definition |
|---|---|
| $CPU_i$ | CPU capacity of server and VM |
| $POW_i$ | Power capacity of server |
| $TIME_i$ | Execution time of VM |
| $UTIL_i$ | Utilization factor of server |
| SR | Source server for VM migration |
| DS | Destination server for VM migration |
| $Num\_Pages_s$ | Number of memory pages used by VM selected for migration |
| $Num\_Blocks_s$ | Number of disk blocks used by VM selected for migration |
| $Trans_t$ | Total time needed to transfer memory page or disk block from one server to the other |
| $S_t$ | Time between halt of VM at source and resumption at destination server |
| $Set_t$ | Time required to set up VM at destination |
| TMT | Total migration time |
| P (F,t) [31] | Power capacity of the ith server in terms of function of placement "F" |
| $U_i$(F,t) [31] | Utilization factor of the ith server in terms of placement "F2 and time "t" |
| $Resutil_i$ [32] | Resource utilization of the ith server |
| Makespan [32] | Time from submission of the 1st VM to completion of the last VM |
| $CPU\_util_i$ | Utilized CPU value on the ith server |
| $Tvm_j$ | Execution time of VM j |
| $Tvm\_max_i$ | Maximum time of any VM on server i |

The existing and proposed algorithms are explained below:

- **Live migration with ballooning (LMB)**—As presented in Algorithm 1, initially the waiting VMs are allotted to the available server. If the server is overburdened, then a decision is taken to migrate the minimum-CPU-capacity VM from it to some other server that has sufficient space. Firstly, ballooning is performed to delete unused pages. Then, the migration of memory and disk storage state is performed in the pre-copy stage. After copying the data, the VM is turned off at the source server and resumed at the destination after setting up the configuration. The next step is again the migration of memory and storage state in the post-copy stage. Once the migration is over, energy consumption, total migration time, and downtime are computed. When no VMs are left, resource utilization and makespan are calculated. The framework is shown in Figure 2, and the flowchart is shown in Figure 3.
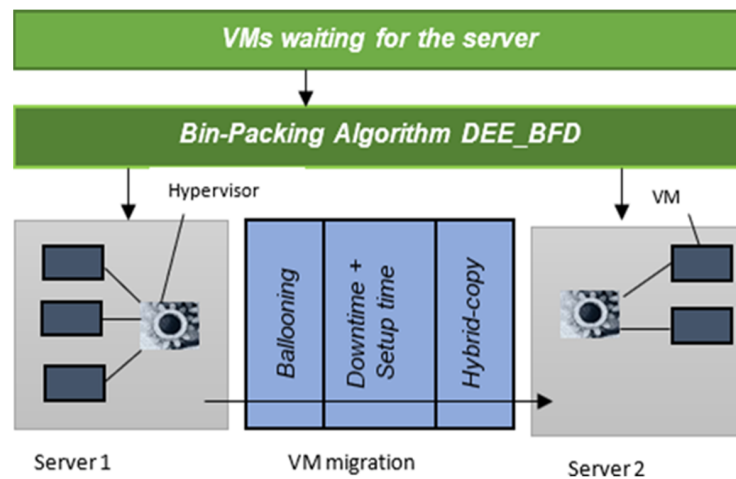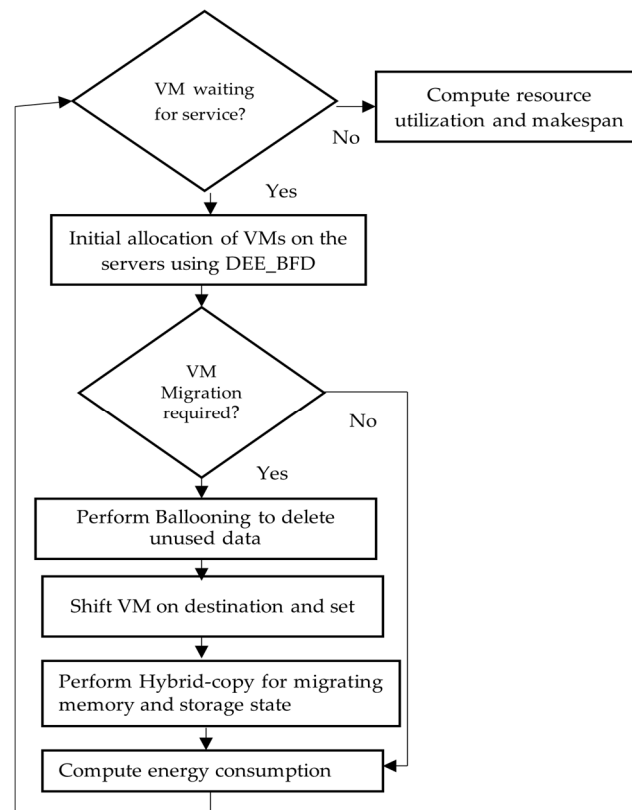
**Figure 2.** LMB framework.



**Figure 3.** LMB flowchart.

- **Live migration with efficient ballooning (LMEB)**—As presented in Algorithm 2, firstly, the waiting VMs are allotted to the available server. If the server is overburdened, then a decision is taken to migrate the minimum-CPU-capacity VM from it to some other server that has sufficient space. Firstly, ballooning is performed to delete unused pages and disk blocks considering the time of generation to use the LRU (least recently used) technique. Then, the VM is turned off at the source server and resumed at the destination after setting up the configuration. The next step is the migration of memory and storage state in the post-copy stage. Once the migration is over, energy consumption, total migration time, and downtime are calculated. If there are no VMs left waiting, resource utilization and makespan are computed. The framework is shown in Figure 4, and the flowchart is shown in Figure 5.
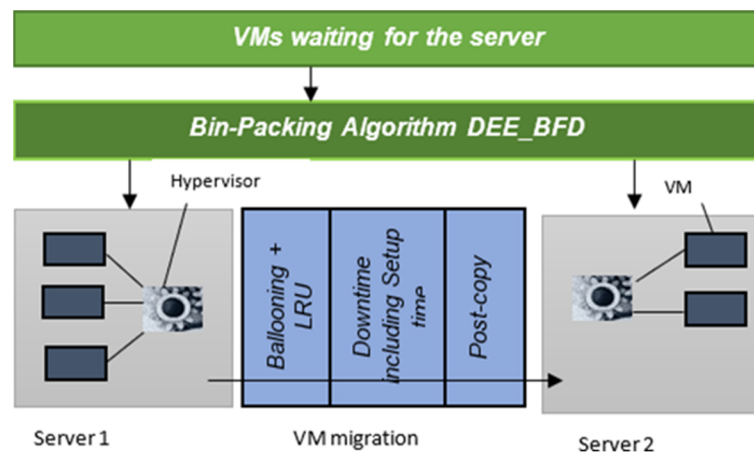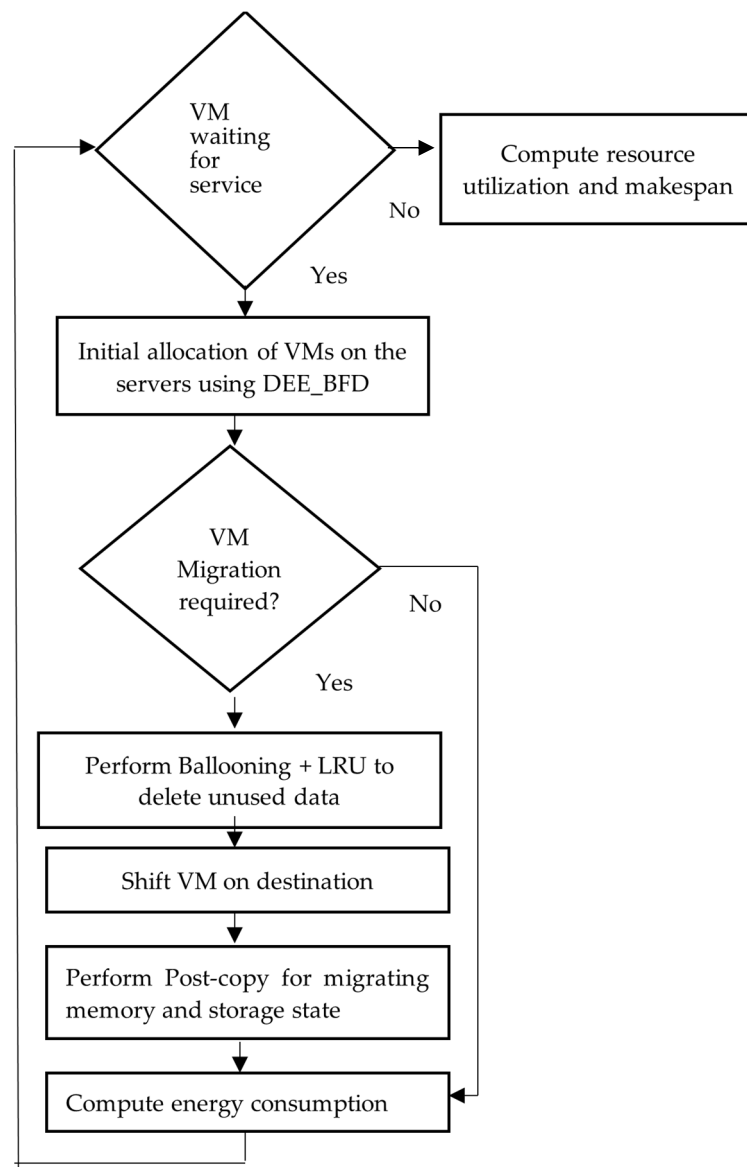
**Figure 4.** LMEB framework.



**Figure 5.** LMEB flowchart.

---

**Algorithm 1.** Live migration with ballooning (LMB)

---

1. Input $CPU_i$ and POWi of N (i = 1 to N) servers.
2. Input $CPU_j$ and TIMEj of M (j = 1 to M) VMs.
3. Initiate allocation of VMs on the servers using DEE_BFD [33].
4. Calculate $UTIL_i$ of N servers.
5. If UTILi = 0.90 to 1.00 then SR = $Server_i$
6. If UTILi = 0.20 to 0.50 then DS = $Server_i$
7. Select the minimum $CPU_j$ VM from SR.
8. $Num\_Pages_s$ and $Num\_Blocks_s$ = Random Number between 10 to 20.
9. Generate 1-d arrays—Memory [] and Blocks [] randomly with size $Num\_Pages_s$ and $Num\_Blocks_s$ respectively where $i^s$ index value = number of accesses
10. Perform ballooning by removing value $'0'$ from two arrays.
11. Assume: $Trans_t$ = 10 s 1 s $Trans_t$ consumes 10 kWh power $S_t$ = 10 s $Set_t$ = 5 s
12. Perform Pre-copy technique TMT = $(Num\_Pages_s + Num\_Blocks_s)$ * $Trans_t$.
13. Stop the VM on source and resume it on destination after completing the setup procedure.
14. Generate Memory [] and Blocks [] again for remaining accessed memory pages and disk blocks.
15. Perform Post-copy technique TMT = TMT + $(Num\_Pages_s + Num\_Blocks_s)$ * $Trans_t$.
16. Calculate final TMT, TMT = TMT +$S_t$ + $Set_t$
17. Calculate downtime DT = $S_t$ + $Set_t$
18. Calculate energy consumption [31] $P_i(F,t)$ = 0.7 * $P_i$ + (0.3 * $P_i$ * $U_i(F,t)$) for $i^{th}$ server $U_i(F,t)$ = $(CPU\_util_i)/(CPU_i)$
19. Calculate resource utilization and makespan [31] Res_util = $Res\_util_i$ for i = 1 to N, where $Res\_util_i = (\sum_{j=1}^{m} Tvm_j)/(makespan * M)$ and makespan = $\sum_{j=1}^{R} Tvm\_max_j$

---

---

**Algorithm 2.** Live migration with efficient ballooning (LMEB)

---

1. Input $CPU_i$ and $POW_i$ of N (i = 1 to N) servers.
2. Input $CPU_j$ and $TIME_j$ of M (j = 1 to M) VMs.
3. Initiate allocation of VMs on the servers using DEE_BFD [33].
4. Calculate $UTIL_i$ of N servers.
5. If $UTIL_i$ = 0.90 to 1.00 then SR = $Server_i$
6. If $UTIL_i$ = 0.20 to 0.50 then DS = $Server_i$
7. Select the minimum $CPU_j$ VM from SR.
8. $Num\_Pages_s$ and $Num\_Blocks_s$ = Random Number between 10 to 20.
9. Generate 2-d arrays—Memory [][2] and Blocks [][2] randomly with row size $Num\_Pages_s$ and $Num\_Blocks_s$ respectively where 1st column = number of accesses and 2nd column = time of generation.
10. Perform ballooning on source server by removing value $'0'$ from two arrays considering the time of generation and set up of VM on the destination server.
11. Assume: $Trans_t$ = 10 s 1 s $Trans_t$ consumes 10 kWh power $S_t$= 10 s
12. Stop the VM on the source and resume it on destination.
13. Perform Post-copy technique TMT = $(Num\_Pages_s + Num\_Blocks_s)$ * $Trans_t$.
14. Calculate final TMT, TMT = TMT +$S_t$
15. Calculate downtime DT = $S_t$
16. Calculate energy consumption [31] $P_i(F,t)$= 0.7 *$P_i$ + (0.3 * $P_i$ * $U_i(F,t)$) for ith server $U_i(F,t)$ = $(CPU\_util_i)/(CPU_i)$
17. Calculate resource utilization and makespan [31] Res_util = $Res\_util_i$ for i = 1 to N, where $Res\_util_i = (\sum_{j=1}^{m} Tvm_j)/(makespan * M)$ and
18. Makespan = $\sum_{j=1}^{R} Tvm\_max_j$

---

The next section discusses the comparative analysis of the algorithms in terms of parameters.

## 4. Experimental Testbed

Simulation was performed in JAVA using the in-built classes to store the mapping of virtual machines to the server. Different scenarios were considered to check the validity of the proposed algorithm. These scenarios have the same number of servers, $s_i$ {i = 1, 2, 3, 4, 5}, and their configuration, $c_i$ {i = CPU, Power} but different numbers of virtual machines, $v_i$ {i = 1, 2, 3}, {i = 1, 2, 3, 4}, {i = 1, 2, 3, 4, 5}, {i = 1, 2, 3, 4, 5, 6}, {i = 1, 2, 3, 4, 5, 6, 7} and different configurations, $c1_i$ {i = CPU, Time}. The five scenarios were taken, and each scenario witnessed an incremental increase in the number of virtual machines. The configuration details of the server included CPU capacity in instructions per cycle and power capacity in kwh. The configuration details of the virtual machines included CPU capacity in instructions per cycle and execution time in seconds. The server configuration (CPU and power capacity) is presented in Table 2.

**Table 2.** Server configuration.

| Server No. | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| CPU capacity (instruction cycles) | 2500 | 2000 | 1500 | 3000 | 2700 |
| Power capacity (kWh) | 300 | 200 | 450 | 350 | 500 |

The virtual machine configurations (CPU and execution duration) in these five scenarios are presented in Tables 3–7.

**Table 3.** Configuration of 3 virtual machines for scenario 1.

| Virtual Machine No. | 1 | 2 | 3 |
|---|---|---|---|
| CPU capacity (instruction cycles) | 1200 | 800 | 1000 |
| Execution time (seconds) | 30 | 50 | 40 |

**Table 4.** Configuration of 4 virtual machines for scenario 2.

| Virtual Machine No. | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| CPU capacity (instruction cycles) | 1200 | 800 | 1000 | 2000 |
| Execution time (seconds) | 30 | 50 | 40 | 20 |

**Table 5.** Configuration of 5 virtual machines for scenario 3.

| Virtual Machine No. | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| CPU capacity (instruction cycles) | 1200 | 800 | 1000 | 1300 | 700 |
| Execution time (seconds) | 30 | 50 | 40 | 20 | 90 |

**Table 6.** Configuration of 6 virtual machines for scenario 4.

| Virtual Machine No. | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| CPU capacity (instruction cycles) | 1200 | 800 | 1000 | 1300 | 700 | 600 |
| Execution time (seconds) | 30 | 50 | 40 | 20 | 90 | 40 |

**Table 7.** Configuration of 7 virtual machines for scenario 5.

| Virtual Machine No. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| CPU capacity (instruction cycles) | 1200 | 800 | 1000 | 1300 | 900 | 750 | 2000 |
| Execution time (seconds) | 30 | 50 | 40 | 20 | 90 | 40 | 10 |

Random values were generated to compute the total number of pages and disk blocks used by the VMs. The number of accesses to memory pages and disk blocks were also computed randomly. Some assumptions were made for creating the same execution environment for both algorithms and to analyze the differences between them.

These assumptions were as follows:

- Maximum CPU capacity of server was 3000 instruction cycles.
- Maximum power capacity of server was 500 kWh.
- Maximum CPU capacity of VM was 1500 instruction cycles.
- Single memory page or disk block was transferred to the destination server through the network connection in 10 s.
- The energy consumed for one second in the transfer process was 10 kWh.
- The time between switching off the VM at the source server and resumption at the destination server was 10 s.
- The setup time of the VM with defined configuration (configuration of VM running on the source server) at the destination server was 5 s.

The next section presents the results of the execution and comparative analysis.

## 5. Results

The two algorithms were executed, and their results were compared in terms of energy consumption, migration time, downtime, resource utilization, and makespan. The energy consumption analysis is shown in Table 8 and Figure 6.

**Table 8.** Energy consumption for all scenarios in kWh.

| No. of Virtual Machines | LMB | LMEB |
|---|---|---|
| 3 | 4997.2 | 2847.2 |
| 4 | 5983.8 | 3233.8 |
| 5 | 5193.8 | 3043.8 |
| 6 | 5220.8 | 3770.8 |
| 7 | 5624 | 3474 |

The migration time analysis is shown in Table 9 and Figure 7.

**Table 9.** Migration time for all scenarios in seconds.

| No. of Virtual Machines | LMB | LMEB |
|---|---|---|
| 3 | 455 | 240 |
| 4 | 535 | 260 |
| 5 | 435 | 220 |
| 6 | 435 | 290 |
| 7 | 455 | 240 |

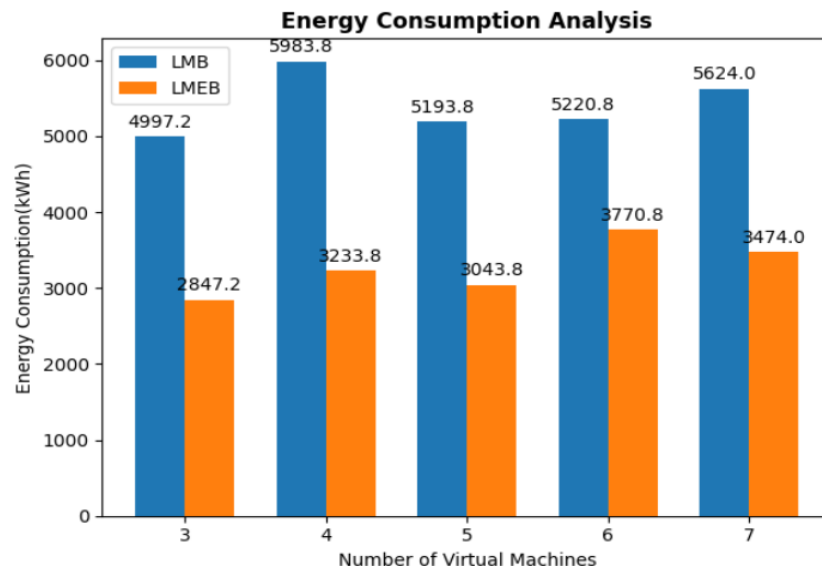The makespan analysis is shown in Table 10 and Figure 8.

**Figure 6.** Energy consumption analysis.



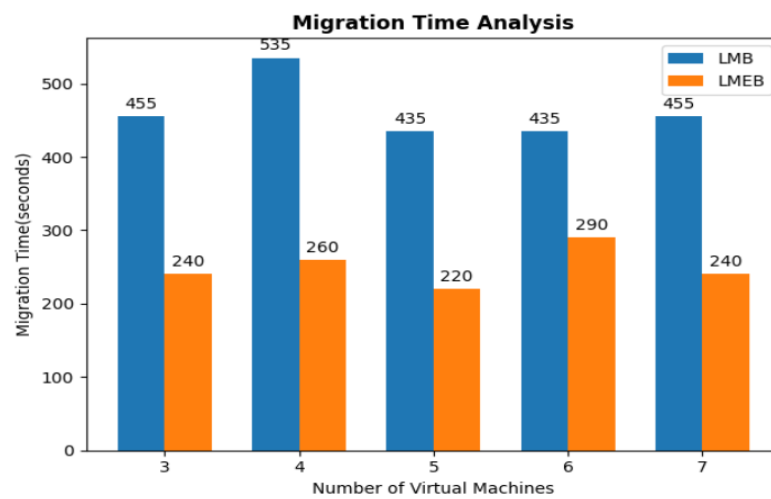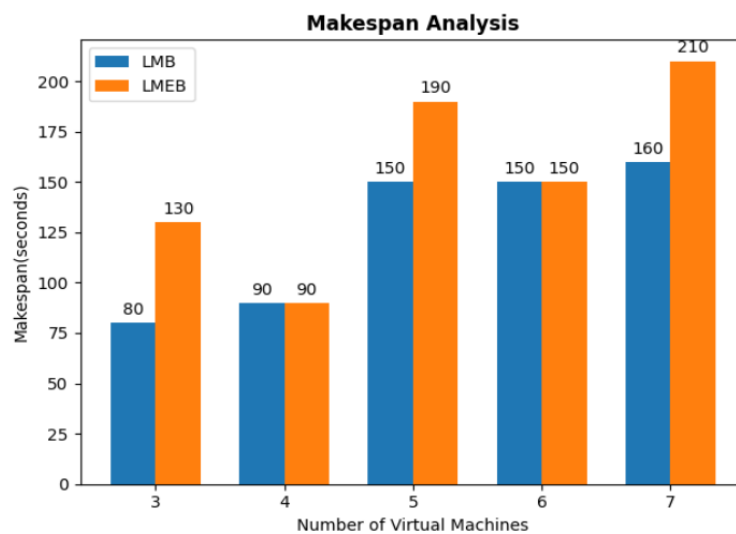**Figure 7.** Migration time analysis.



**Figure 8.** Makespan analysis.

**Table 10.** Makespan for all scenarios in seconds.

| No. of Virtual Machines | LMB | LMEB |
|:---:|:---:|:---:|
| 3 | 80 | 130 |
| 4 | 90 | 90 |
| 5 | 150 | 190 |
| 6 | 150 | 150 |
| 7 | 160 | 210 |

*5.1. Statistical Analysis*

The energy consumption values were analyzed in terms of standard deviation and variance to check the validity of the results.

The equation for standard deviation is

$$S = \sqrt[2]{\frac{1}{N-1}\sum_{i=1}^{N}(xi - \overline{x})} \tag{4}$$

And the equation for variance is
$$v = s^2 \tag{5}$$

Using the values in Table 8, the statistical analysis of energy consumption was performed, and the results are presented in Table 11.

**Table 11.** Results of statistical analysis of energy values.

| | LMB | LMEB |
|:---:|:---:|:---:|
| Standard deviation | 396.17 | 361.79 |
| Variance | 156,950 | 130,894 |

*5.2. Discussion*

The existing and proposed algorithms were analyzed in terms of the parameters explained in Section 1. The details are as follows:

- **Energy consumption**—The standard deviation and variance were computed, and the results are presented in Table 11. The variance indicates the consistency of the algorithm. The lower the value is, the greater is the consistency is; therefore, LMEB was proved to be more energy efficient than LMB. The average energy consumption of the existing algorithm (LMB) was 5403.92 kWh, and that of the proposed algorithm (LMEB) was 3273.92, so it was reduced by 39%.
- **Migration time**—According to Table 9, the migration time of LMEB was smaller than that of LMB. The average migration time of LMB was 463 s, and that of LMEB was 250 s, so it was reduced by 46%.
- **Downtime**—The downtime depends on the network properties between the source and destination servers and was constant for both algorithms in all cases. The downtime value for LMEB was 10 s, and that for LMB was 15 s. LMEB had a smaller downtime value as it does not include the setup time (5 s) of the virtual machine at the destination. The downtime of LMB was 15 s, and that of LMEB was 10 s, so it was reduced by 25%.
- **Resource utilization**—The values were found to be equal in both algorithms in all cases. So, there were no differences between LMB and LMEB in terms of resource utilization.
- **Atomicity and convergence**—LMEB was designed in such a way that it preserves atomicity and convergence, as there is only one transfer of data during the post-copy technique, so there are no differences in the data at the source and destination machines; the migration process was not affected by any other resources, so it was successfully completed.

- **Makespan**—According to Table 10, the makespan of LMEB was found to be larger than that of LMB in some cases. So, LMEB does not guarantee a lowering of the total working time of the servers.

*5.3. Complexity*

The migration of a virtual machine is not dependent on the number of virtual machines and servers, but it depends on the transmission speed of the network cable from the source to the destination machine; only in this way, the complexity of LMEB is O(t), where t Mbps is the transmission speed.

## 6. Workflow of Cloud Data Center

After the comparative study, the proposed migration algorithm, LMEB, was found to be more energy efficient and also potentially useful in other applications, such as the IoT, smart city architecture [34,35], the scheduling of various tasks [36,37], etc. The technique of the initial allocation of VMs to the physical server and live VM migration together can contribute to improving the performance of cloud data centers by consuming less energy. The proposed migration algorithm, LMEB, in combination with the bin-packing algorithm DEE-BFD [33], is shown in Figure 9.

The workflow starts with taking the configuration of the server in the data center as the input. Thereafter, the process of allocation of VMs to the server is presented through the bin-packing algorithm. After successful allocation, the servers are analyzed to know whether VM migration is required or not. If the answer is yes, then the ballooning technique is implemented along with the LRU algorithm to delete unused memory pages. LRU is the least recently used algorithm for page replacement, which considers the pages that refer to the past as target to be replaced by new pages. This procedure is followed by post-copy and the calculation of the energy value. The whole practice repeats until no VMs are waiting for the service. When there are no more VMs in the waiting list, resource utilization and makespan are calculated to check the functioning of the entire system, so that timely action can be taken for further improvement. The proposed workflow can be used for implementing virtual machine placement with minimum energy [38–42].
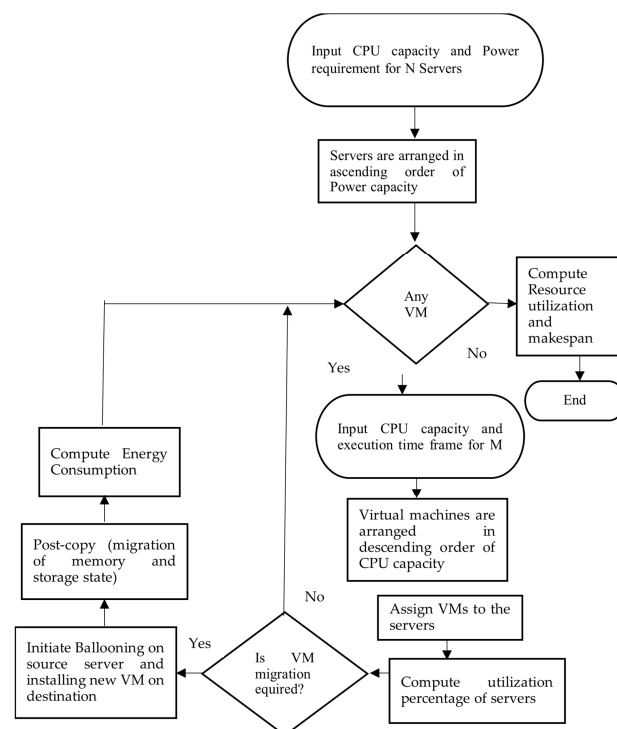


**Figure 9.** Overall workflow of cloud data center.

The next section presents the concluding remarks.

## 7. Conclusions

Live VM migration is an important approach to reduce energy consumption along with the bin-packing technique. In this paper, a new algorithm, LMEB, is proposed; LMEB aims at reducing the percentage of energy consumption, downtime, and migration time and maintaining the resource utilization value. The algorithm is also able to achieve the atomicity and convergence properties of migration. By making live VM migration more energy aware, the overall energy requirement and thus the performance of data centers can be upgraded.

As future work, the proposed algorithm could be upgraded with the concept of multiple live VM migrations and in terms of the makespan parameter. The algorithm could also be simulated on any cloud simulator so that other researchers can use it for further extension.

**Author Contributions:** Conceptualization, N.G. (Neha Gupta) and K.G.; Data curation, D.G.; Formal analysis, D.G. and N.G. (Nitin Goyal); Funding acquisition, A.M.Q.; Project administration, A.M.Q. and F.S.A.; Resources, F.S.A. and N.G. (Nitin Goyal); Software, F.S.A.; Supervision, N.G. (Nitin Goyal); Writing—original draft, N.G.; Writing—review and editing, K.G and A.S. All authors have read and agreed to the published version of the manuscript.

**Data Availability Statement:** Not applicable.

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

## References

1. Savu, L. Cloud Computing: Deployment Models, Delivery Models, Risks and Research Challanges. In Proceedings of the 2011 International Conference on Computer and Management, CAMAN 201, Wuhan, China, 19–21 May 2011. [CrossRef]
2. Xing, Y.; Zhan, Y. Virtualization and cloud computing. *Lect. Notes Electr. Eng.* **2012**, *143*, 305–312. [CrossRef]
3. Kapil, D.; Pilli, E.S.; Joshi, R.C. Live virtual machine migration techniques: Survey and research challenges. In Proceedings of the 2013 3rd IEEE International Advance Computing Conference, IACC, Ghaziabad, India, 22–23 February 2013; pp. 963–969. [CrossRef]
4. Zhang, F.; Liu, G.; Fu, X.; Yahyapour, R. A Survey on Virtual Machine Migration: Challenges, Techniques, and Open Issues. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1206–1243. [CrossRef]
5. Michael, R.; Hines, K.G. Post-Copy Based Live Virtual Machine Migration Using Adaptive Pre-Paging and Dynamic Self-Ballooning. In Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE'09), Washington, DC, USA, 11–13 March 2009; pp. 51–60.
6. Zhang, X.; Huo, Z.; Ma, J.; Meng, D. Exploiting data deduplication to accelerate live virtual machine migration. In Proceedings of the Proceedings—IEEE International Conference on Cluster Computing, ICCC 2010, Heraklion, Greece, 20–24 September 2010; pp. 88–96. [CrossRef]
7. Mashtizadeh, A.; Celebi, E.; Garfinkel, T.; Cai, M. The design and evolution of live storage migration in Vmware Esx. In Proceedings of the 2011 USENIX Annual Technical Conference, USENIX ATC 2011, Portland, OR, USA, 15–17 June 2011; pp. 187–200.
8. Murtazaev, A.; Oh, S. Sercon: Server consolidation algorithm using live migration of virtual machines for green computing. *IETE Tech. Rev.* **2011**, *28*, 212–231. [CrossRef]
9. Adhianto, L.; Banerjee, S.; Fagan, M.; Krentel, M.; Marin, G.; Mellor-Crummey, J.; Tallent, N.R. HPCTOOLKIT: Tools for performance analysis of optimized parallel programs. *Concurr. Comput. Pract. Exp.* **2010**, *22*, 685–701. [CrossRef]
10. Jeong, J.; Kim, S.H.; Kim, H.; Lee, J.; Seo, E. Analysis of virtual machine live-migration as a method for power-capping. *J. Supercomput.* **2013**, *66*, 1629–1655. [CrossRef]
11. Jin, H.; Deng, L.; Wu, S.; Shi, X.; Chen, H.; Pan, X. MECOM: Live migration of virtual machines by adaptively compressing memory pages. *Future Gener. Comput. Syst.* **2014**, *38*, 23–35. [CrossRef]
12. Jenitha, V.H.A.; Veeramani, R. Dynamic memory Allocation using ballooning and virtualization in cloud computing. *IOSR J. Comput. Eng.* **2014**, *16*, 19–23. [CrossRef]

13. Wang, C.; Hao, Z.; Cui, L.; Zhang, X.; Yun, X. Introspection-Based Memory Pruning for Live VM Migration. *Int. J. Parallel Program.* **2017**, *45*, 1298–1309. [CrossRef]

14. Zhang, F.; Fu, X.; Yahyapour, R. LayerMover: Storage migration of virtual machine across data centers based on three-layer image structure. In Proceedings of the 2016 IEEE 24th International, Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS 2016, London, UK, 19–21 September 2016; pp. 400–405. [CrossRef]

15. Wu, T.Y.; Guizani, N.; Huang, J.S. Live migration improvements by related dirty memory prediction in cloud computing. *J. Netw. Comput. Appl.* **2017**, *90*, 83–89. [CrossRef]

16. Deshmukh, P.P.; Amdani, S.Y. Virtual Memory Optimization Techniques in Cloud Computing. In Proceedings of the 2018 3rd IEEE International Conference on Research in Intelligent and Computing in Engineering, RICE 2018, San Salvador, El Salvador, 22–24 August 2018. [CrossRef]

17. Patel, M.; Chaudhary, S.; Garg, S. Improved pre-copy algorithm using statistical prediction and compression model for efficient live memory migration. *Int. J. High Perform. Comput. Netw.* **2018**, *11*, 55–65. [CrossRef]

18. Sun, G.; Liao, D.; Anand, V.; Zhao, D.; Yu, H. A new technique for efficient live migration of multiple virtual machines. *Future Gener. Comput. Syst.* **2016**, *55*, 74–86. [CrossRef]

19. Kaur, J.; Chana, I. Review of Live Virtual Machine Migration Techniques in Cloud Computing. In Proceedings of the 2018 International Conference on Circuits and Systems in Digital Enterprise Technology, ICCSDET, 2018, Kottayam, India, 21–22 December 2018; pp. 1–6. [CrossRef]

20. Chashoo, S.F.; Malhotra, D. VM-Mig-framework: Virtual machine migration with and without ballooning. In Proceedings of the PDGC 2018—2018 5th International Conference on Parallel, Distributed and Grid Computing 2018, Solan, India, 20–22 December 2018; pp. 368–373. [CrossRef]

21. Silva, D.D.; Eds, L.j.Z.; Hutchison, D. *Cloud Computing—CLOUD 2019*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; Volume 11513, pp. 99–113. [CrossRef]

22. Mazrekaj, A.; Nuza, S.; Zatriqi, M.; Alimehaj, V. An overview of virtual machine live migration techniques. *Int. J. Electr. Comput. Eng.* **2019**, *9*, 4433–4440. [CrossRef]

23. He, T.; Buyya, R. A Taxonomy of Live Migration Management in Cloud Computing. *arXiv* **2021**, arXiv:2112.02593.

24. Mishra, P.; Aggarwal, P.; Vidyarthi, A.; Singh, P.; Khan, B.; Alhelou, H.H.; Siano, P. 343 VMShield: Memory Introspection-Based Malware Detection to Secure Cloud-Based Ser- 344 vices against Stealthy Attacks. *IEEE Trans. Ind. Inform.* **2021**, *17*, 6754–6764. [CrossRef]

25. Karmakar, K.; Das, R.K.; Khatua, S. An ACO-based multi-objective optimization for cooperating VM placement in cloud data center. *J. Supercomput.* **2022**, *78*, 3093–3121. [CrossRef]

26. Lu Yin, J.Z.; Sun, J. A stochastic algorithm for scheduling bag-of-tasks applications on hybrid clouds under task duration variations. *J. Syst. Softw.* **2022**, *184*, 111123. [CrossRef]

27. Bui, K.T.; Ho, H.D.; Pham, T.V.; Tran, H.C. Virtual machines migration game approach for multi–tier application in infrastructure as a service cloud computing. *IET Netw.* **2020**, *9*, 326–337. [CrossRef]

28. Pande, S.K.; Panda, S.K.; Das, S.; Sahoo, K.S.; Luhach, A.K.; Jhanjhi, N.Z.; Alroobaea, R.; Sivanesan, S. A resource management algorithm for virtual machine migration in vehicular cloud computing 2021. *Comput. Mater. Contin.* **2021**, *67*, 2647–2663. [CrossRef]

29. Tran, C.H.; Bui, T.K.; Pham, T.V. Virtual machine migration policy for multi-tier application in cloud computing based on Q-learning algorithm. *Comput.* **2022**, *104*, 1285–1306. [CrossRef]

30. Gupta, A.; Namasudra, S. A novel technique for accelerating live migration in cloud computing. *Autom. Softw. Eng.* **2022**, *29*, 34. [CrossRef]

31. Tuli, K.; Kaur, A.; Malhotra, M. Efficient virtual machine migration algorithms for data centers in cloud computing. In Proceedings of the International Conference on Innovative Computing and Communications, Delhi, India, 25–26 March 2022; p. 473. [CrossRef]

32. Gupta, K.; Katiyar, V. Energy aware virtual machine migration techniques for cloud environment. *Int. J. Comput. Appl.* **2016**, *141*, 11–16. [CrossRef]

33. Kalra, M.; Singh, S. A review of metaheuristic scheduling techniques in cloud computing. *Egypt. Inform. J.* **2015**, *16*, 275–295. [CrossRef]

34. Gupta, N.; Gupta, K.; Gupta, D.; Juneja, S.; Turabieh, H.; Dhiman, G.; Kautish, S.; Viriyasitavat, W. Enhanced virtualization-based dynamic bin-packing optimized energy management solution for heterogeneous clouds. *Math. Probl. Eng.* **2022**, *2022*, 8734198. [CrossRef]

35. Datta, P.; Sharma, B. A survey on IoT architectures, protocols, security and smart city based applications. In Proceedings of the 8th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2017, Delhi, India, 3–5 July 2017. [CrossRef]

36. Bali, M.S.; Gupta, K.; Koundal, D.; Zaguia, A.; Mahajan, S.; Pandit, A.K. Smart architectural framework for symmetrical data offloading in IOT. *Symmetry* **2021**, *13*, 1889. [CrossRef]

37. Mohammadzadeh, A.; Masdari, M.; Gharehchopogh, F.S.; Jafarian, A. A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling. *Clust. Comput.* **2021**, *24*, 1479–1503. [CrossRef]

38. Mohammadzadeh, A.; Masdari, M.; Gharehchopogh, F. An improved grey wolves optimization algorithm for workflow scheduling in cloud computing environment. *J. Soft Comput. Inf. Technol.* **2020**, *8*, 17–29.

39. Gharehpasha, S.; Masdari, M.; Jafarian, A. Power efficient virtual machine placement in cloud data centers with a discrete and chaotic hybrid optimization algorithm. *Clust. Comput.* **2021**, *24*, 1293–1315. [CrossRef]

40. Gharehpasha, S.; Masdari, M.; Jafarian, A. Virtual machine placement in cloud data centers using a hybrid multi-verse optimization algorithm. *Artif. Intell. Rev.* **2021**, *54*, 2221–2257. [CrossRef]

41. Masdari, M.; Gharehpasha, S.; Jafarian, A. An optimal vm placement in cloud data centers based on discrete chaotic whale optimization algorithm. *J. Adv. Comput. Eng. Technol.* **2020**, *6*, 201–212.

42. Gharehpasha, S.; Masdari, M.; Jafarian, A. The placement of virtual machines under optimal conditions in cloud datacenter. *Inf. Technol. Control.* **2019**, *48*, 545–556. [CrossRef]