sustainability

MDPI

Article

Designing an Efficient Cloud Management Architecture for Sustainable Online Lifelong Education

TaeYoung Kim [1] and JongBeom Lim [2,*]

[1] College of Education, Hankuk University of Foreign Studies, 107 Imun-ro, Dongdaemun-gu, Seoul 02450, Korea; ktyoung66@hanmail.net

[2] Department of Game & Multimedia Engineering, Korea Polytechnic University, Siheung-si, Gyeonggi-do 15073, Korea

* Correspondence: jblim@kpu.ac.kr

Received: 15 January 2019; Accepted: 8 March 2019; Published: 13 March 2019

Abstract: As online learning and e-learning are prevalent and widely used in education, it is important to design an efficient and reliable information system for storing learning data and providing on-demand learning services. In this paper, we design a cloud-based information system architecture for online lifelong education. Since a cloud system is based on virtualization technology, we propose a virtual resource management scheme—virtual machine allocation and monitoring nodes assignment. With the proposed cloud-based architecture, we can build and operate an e-learning information system for online lifelong education, which requires efficiency, reliability, and persistence. The evaluation results show that our proposed method can deal with more tasks for e-learning (requests for learning management system (LMS) navigations, text learning contents, text and media learning contents, and video learning contents) while introducing 48× fewer service level agreement (SLA) violations than the existing method.

Keywords: online learning; e-learning; lifelong education; information system

1. Introduction

Although education is strongly associated with children and youth, we are living in an age of technological innovation [1,2]. With the help of information technology, e-learning and online courses are available for adults and older people. Since the proportion of older people (over 60 years of age) continues to grow at an unprecedented rate [3], lifelong education has become an essential tool to improve self-efficacy, self-sustainability, and quality of life [4,5].

Among education methods, there are several advantages of online learning since it provides convenient and portable learning options [6]. More importantly, users can learn by online education at their own pace [7,8]. Therefore, users are able to develop skills and explore knowledge whenever and wherever there are.

Cloud computing is an Internet-based computing paradigm that provides an illusion of infinite computing resources in the form of pay-as-you-go and on-demand [9–11]. Because of the flexibility of cloud computing, it becomes an attractive way to provide the required computing and storage resources to sustainable online learning systems [12,13]. Another benefit of using cloud computing in online learning and lifelong education is an automation process for resource (virtual machine) provisioning [14–16]. Whenever a user (instructor or learner) requests a virtual machine instance, the requested virtual machine instance can be provisioned within a minute [17].

The request includes ownership information, tags, virtual hardware requirements, the operating system, and any customization of the request. Then, the request goes through an approval phase

and is executed. Educational institutions, as well as teachers and learners can benefit from the cloud computing model for cost reduction, elasticity, easy and wide accessibility, and high availability [18,19].

However, prior cloud based online lifelong education systems suffer from some limitations due to inefficient implementation or lack of cloud consolidation techniques. Thus, the objective of the proposed solution is to enhance the usability of online lifelong education systems without experiencing downtime or service disruption by providing an efficient cloud resource management scheme.

In this paper, we present an efficient cloud architecture for online lifelong education. Since there are many computing and storage nodes in the cloud computing system, efficient resource management is essential, especially for resource allocation and node monitoring. For resource allocation and node monitoring, we propose a complexity-efficient algorithm based on the n-queen problem.

The n-queen problem is a popular classic puzzle, where n queens were to be placed on an $n \times n$ chessboard such that no queen can attack any other queen. Although the n-queen does not directly relate to cloud resource management and consolidation, we exploit the structure of the problem and apply the algorithmic design to the virtual resource monitoring scheme to provide fault tolerance and eliminate the problem of single point of failures.

One of the benefits of the resource allocation is load balancing. Load balancing is an essential metric for cloud computing since it is related to the service level agreement (SLA). When a virtual machine is allocated in a heavily loaded physical machine, the allocated virtual machine does not perform well and this leads to SLA violations. On the other hand, when a virtual machine is allocated in a lightly loaded physical machine, the energy consumption will rise due to the physical machine. Therefore, balancing the load of physical machines is vital to achieve both SLA satisfaction and energy reduction.

As for node monitoring in cloud computing environments, the monitoring node may result in a single point of failure. In other words, when the monitoring node fails, the whole cloud computing system will stop working. If the monitoring node does not fail, it can be a bottleneck for monitoring a large number of nodes in the cloud computing system. This motivates us to develop an efficient monitoring scheme that does not exhibit a single point of failure or bottleneck problem. The proposed monitoring scheme is able to monitor nodes in the cloud computing system efficiently by designating a fixed number of monitoring nodes.

The aforementioned context of efficient cloud resource management directly affects online lifelong education in several ways: (1) it supports high availability of online learning services for both computing and storage, in other words, our solution eliminates the problems of a bottleneck and single point of failures when services are operating in cloud computing environments, (2) it minimizes SLA violations, which are of great concern to online learners in multitenancy service architectures, since our techniques focus on maintaining cloud consolidation while improving server utilization, and (3) it provides a pleasant learning experience through improved response time, fewer task failures for online learning services, and reduced power consumption.

The roadmap of the paper can be summarized as follows. Section 2 discusses our research motivation and describes the system model. Section 3 proposes our cloud resource management scheme that assigns monitoring nodes based on the classic n-queen problem and allocates virtual machines to physical hosts in cloud computing environments. Section 4 provides results of performance evaluations that ensure load balancing in terms of monitoring messages and low power consumption compared to a previous approach. Finally, Section 5 concludes the paper.

2. Motivation and System Model

Our research aims at an efficient cloud resource management on the premise of processing general cloud applications, including e-learning information systems based on virtual machines. In this section, we present our research motivation and the system model.

Cloud computing is a new computing paradigm that enables users to access virtualized resources (virtual machines) on-demand in a pay-per-use basis [20]. One of the reasons that cloud computing

has received much attention is that virtual resources can easily be scaled up and down for users' requests [21].

At the same time, cloud computing offers the required properties for e-learning services, which are computation-intensive and data-intensive [22]. The typical example of this scenario is massive open online courses (MOOCs) [23]. In the context of programming, instructors of courses can provision virtual machines that contain pre-installed programming-related software to provide laboratories and programming projects.

In [24], the authors investigated smart learning services by extending existing e-learning services through context-awareness to smart learning content based on cloud computing for personalized and customized learning services. However, this work does not introduce cloud resource management solutions, which are crucial to the use and maintenance of sustainable services such as e-learning.

To develop a smart learning management system (LMS) and improve collaborative learning in higher education, the authors of [25] integrated a multi-agents system for tracking of collaboration levels and productivity status based on a cloud computing architecture. However, it uses an existing cloud service platform and does not implement resource management modules inside the cloud computing environment.

In [26], the authors constructed the digital learning environment by taking advantage of on-demand services of cloud computing. The learning environment promotes knowledge sharing, collaborative learning, and interactive communication. However, it does not provide techniques for learners to relieve the concerns for SLA, which is an important aspect in online learning services. On the other hand, one of our design goals is to develop an online learning platform that minimizes SLA violations through efficient cloud resource management.

As for developing an e-learning platform for resource sharing, the authors of [27] proposed an e-learning model based on private cloud computing. With the help of the virtualization technology, the private cloud based e-learning platform is capable of installing an e-learning system regardless of hardware requirements. Although it is designed to support multiplatform architectures, it does not support continuing interactions between the cloud server and online learners. However, we consider the sustainable online lifelong education with reduced service failures and improved fault tolerance.

Our work differs from previous work in that: (1) we design a cloud resource management framework that is transparent to the online learners and can be extended to other cloud based learning systems, (2) we consider a learning experience for minimizing SLA violations, service response time, and task failures, and (3) we propose an efficient cloud resource management implementation in the perspective of online lifelong education service providers by considering load balancing and power consumption.

The contribution of this work can be summarized as follows. We propose a cloud resource management scheme that not only provides a cloud monitoring mechanism with fault tolerance based on policy but also maintains virtual resources consolidation by examining the resource status and migrating virtual machines to reduce power consumption and improve utilization of cloud servers in the system. We model a monitoring nodes selection algorithm with efficient complexity by exploiting backtracking, and design a virtual resource allocation scheme that leverages the power of two choices [28].

For the system model, we consider 100 host machines in the cloud platform, with each host machine equipped with a Xeon X3470 2933 MHz (4 cores) central processing unit (CPU) and 8 GB random access memory (RAM). When allocating a virtual machine, a user specifies properties of the virtual machine such as the number of vCPU and the amount of RAM. For the experiments for the proposed cloud resource management scheme, we deem that the cloud platform is running for an hour, and therefore, several virtual machines reside in a number of hosts.

3. The Proposed Cloud Resource Management Algorithm

In this Section, we describe the proposed cloud resource management scheme that is designed to work with existing cloud frameworks as an extension. We explain in detail our architectural design for cloud resource management and how our cloud resource management schemes work with cloud managers.

3.1. Architectural Design

Figure 1 shows the architectural design of the proposed cloud resource management scheme, which is based on Xen hypervisor for managing virtual resources (computing, storage, and network). We consider Openstack for managing virtual machines for cloud computing and deploying infrastructure as a service (IaaS).

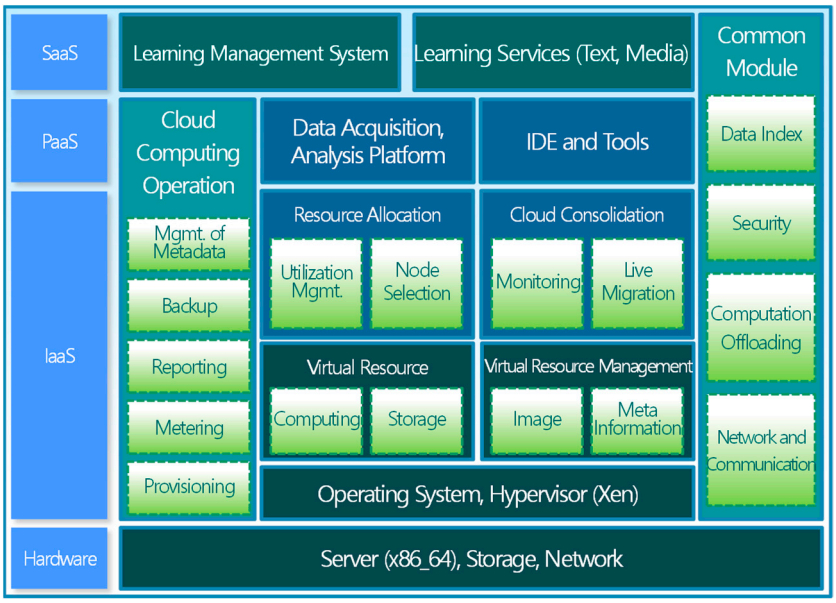


Figure 1. Architectural design of the proposed cloud resource management scheme. SaaS = software as a service; PaaS = platform as a service; IaaS = infrastructure as a service; IDE = integrated development environment.

The proposed monitoring scheme and live migration capabilities reside in the cloud consolidation part of the Figure and the proposed resource utilization management and node selection scheme reside in the resource allocation part of the Figure. The data acquisition, analysis platform, integrated development environments, and tools can be categorized as platform as a service (PaaS).

On top of PaaS, learning management systems, visualizing, and other services can be categorized as software as a service (SaaS). Across the boundaries, there are common modules such as data indexing, security, computation offloading, and networking. The cloud computing operation part of the Figure includes metadata management, backup, reporting, metering, and provisioning modules.

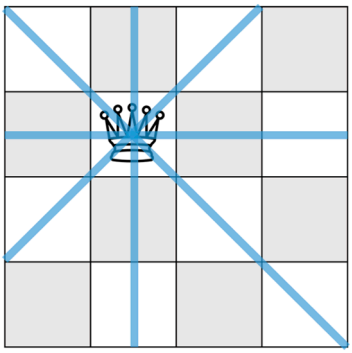
For the cloud resource monitoring scheme, the hypervisor interacts with hardware resources and the operating system. In order to retrieve the monitoring information of physical machines and virtual machines, the hypervisor injects queries to the host operating system and guest virtual machines, and the monitored information can be collected to the privileged virtual machine. Then, the collected monitoring information can be used for cloud consolidation and live migration.

In the meantime, our monitoring node selection algorithm based on the n-queen problem works with the resource allocation module based on the monitoring information of physical machines and virtual machines. Individual clients use an LMS and other learning services, which are compatible

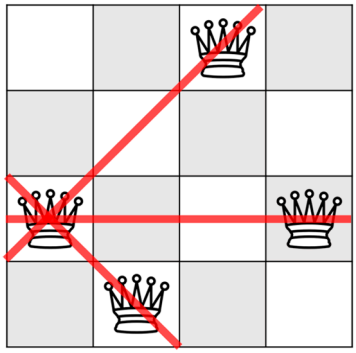
with both physical and virtual resources. Note that we assume that the LMS and learning services are running inside virtual machines.

3.2. The Monitoring Scheme

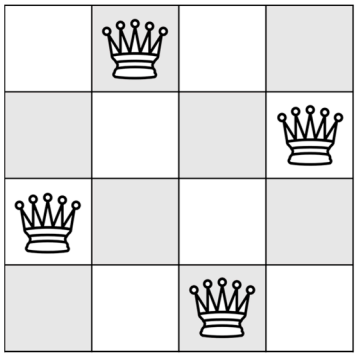
The proposed monitoring scheme is based on a classic n-queen problem as shown in Figure 2. In the example, there is an $n \times n$ board, where $n = 4$. In Figure 2a, a queen is placed on the (2, 2) location. The queen can attack other chess pieces vertically, horizontally, or diagonally based on the location. In this work, we consider a queen as a monitoring node that monitors other board locations (attack lines).



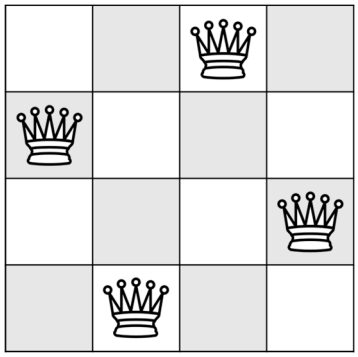
(**a**) Attack lines of a queen



(**b**) Wrong solution



(**c**) A possible solution



(**d**) Another solution

Figure 2. A classic n-queen problem, where $n = 4$.

The basic idea of our cloud monitoring scheme is that we locate queens on the board so that no two queens can attack each other. For example, Figure 2b shows a wrong solution for the n-queen problem, where $n = 4$, since there are conflicts between the following queens: (3, 1) and (1, 3), (3, 1) and (3, 4), and (3, 1) and (4, 2). Possible solutions for the n-queen problem are shown in Figure 2c,d.

The simple approach to the n-queen problem is to use an exhaustive search. However, the search space of this approach is huge. If we consider $n = 8$, for instance, there are 64^8 (= 2^{48} = 281,474,976,710,656) possible placements for eight queens on the 8×8 board. Then, we remove placements in mutually attacking locations. Hence, we use a more efficient algorithm for the n-queen problem based on a state space tree.

In a state space tree, a node is said to be promising if the partial solution is still feasible. Any time the partial node becomes the infeasible node, this non-promising branch will no longer be explored. To find a feasible solution, the algorithm backtracks to the previous promising node and examines the other branches of the state space tree.

Algorithm 1. The monitoring nodes selection algorithm using backtracking.

```
    Input: N = √(number of nodes in the system)
    Output: Q_1, Q_2, ... Q_N ∈ QSet
    Initialization: i ← N
1:  call Queens(N);
2:  function Queens(n)
3:  if n == 0 then
4:          return; // All monitoring nodes are selected.
5:      end if
6:      while (i <= N) do
7:          Q_n ← (i, n); // Place a queen on (i, n)
8:          QSet ← Qset ∪ Q_n;
9:          if (call checkOtherQueens(i, n) then // if Q_n does not check other queens
10:             Queens(n − 1); // Recursive call
11:         end if
12:         Q_n ← (null, null);
13:         QSet ← Qset − Q_n; // Backtrack
14:     end while
15: end function
16: function checkOtherQueens(i, j)
17:     for all Q_i ∈ QSet do
18:         (k, l) ← Q_i;
19:         if (|j − l| == |i − k|) then
20:             return false;
21:         else
22:             return true;
23:         end if
24:     end for
25: end function
```

Algorithm 1 shows the proposed monitoring nodes selection algorithm based on the n-queen problem using backtracking. The input of the algorithm is N ($\sqrt{number\ of\ nodes\ in\ the\ system}$) and the output is placements information on the board for N queens. For initialization, a local variable i is set to N. To begin with, it calls the *Queens*() function with N argument (line 1). In the *Queens*() function, when n is equal to 0, it returns and indicates all queens (monitoring nodes) are placed (assigned) on the board without conflicts (lines 3–5).

In the *while* statement, it iterates when i is less than or equal to N (line 6). First, it places a queen on (i, n) location and it is added to *QSet* (lines 7–8). Then, it checks other queens for conflicts by calling *checkOtherQueens*() (line 9). In this stage, the flow of the process execution is redirected to the *checkOtherQueens*() function (lines 16–25). To check conflicts between queens, it retrieves Q_i from *QSet*. For all Q_i, it compares locations of the two queens (Q_i and Q_n). Note that Q_n is a queen selected from line 7. Then, it checks whether $|j - l|$ is equal to $|i - k|$ (line 19). If the two values are equal, then it returns false (conflict). Otherwise, the function returns true (no conflict).

After returning the *checkOtherQueens*() function, the flow of the process execution is returned to line 9. If the result of the *checkOtherQueens*() function is true, then it calls the *Queens*() function recursively with $n - 1$ argument (line 10). Otherwise, it backtracks for Q_n, which is assigned from lines 7–8. For backtracking, the location of Q_n is nullified and it is removed from *QSet* (lines 12–13). When it backtracks, the algorithm does not go deep into the state space tree.

3.3. The Resource Allocation Scheme

Algorithm 2 shows the proposed cloud architecture management algorithm. There are three parts to the algorithms: (1) virtual machine allocation, (2) queen nodes assignment, and (3) resource monitoring. For the virtual machine allocation, it first checks and gets virtual machine information for a user's request. Then, our virtual machine allocation algorithm selects two host candidates at random. After selecting the two host candidates, it checks the requirement of a user's request. If a host candidate does not qualify a user's request, the procedure listed in lines 3–6 is repeated. Note that if one of the two host candidates does not qualify a user's request, it selects another random host in the system for the host candidate.

Afterward, it checks the utilization of the two host candidates. Of the two host candidates, our virtual machine allocation algorithm selects the one whose utilization is lower. The other one is selected as a nil host (lines 8–14). For optimizing cloud consolidation, it further checks the migration suitability for virtual machines of the nil host. If virtual machines on the nil host can be migrated to the target host, it performs the migration process. Each virtual machine on the nil host is scheduled for migration from the nil host to the target host (lines 16–19). Then, the nil host is scheduled to power off to reduce energy consumption.

The queen nodes assignment algorithm is for optimizing the monitoring overheads induced by the master–slave architecture of cloud computing environments. The basic idea of the queen nodes assignment algorithm is to assign $\sqrt{number\ of\ nodes\ in\ the\ system}$ monitoring nodes in the $n \times n$ board. The monitoring nodes are equivalent to queens in the traditional n-queen problem. The queen nodes are designated as the monitoring nodes in the cloud computing system. The designated monitoring nodes monitor at most $3 \times (n - 1)$ nodes (vertical, horizontal, and diagonal nodes) of $n \times n$ nodes in the system. Thus, it mitigates the problems of the single point of failure and bottleneck.

The procedure of the queen nodes assignment algorithm is listed in Algorithm 1 (lines 23–41). To calculate the board size, it first retrieves the number of hosts in the cloud computing system. Then, it calculates the number of queens (monitoring nodes) by computing square root and round functions. The number of chess pieces is set to the number of hosts in the system at this stage. Note that the logical numbers are also assigned to the hosts for encoding the n–queen problem.

However, if the number of chess pieces and (the number of queens)^2 are not equal, the solution of the n–queen problem cannot be applied (e.g., assigning a queen to the (n, n) location.). Therefore, we resolve this problem by assigning duplicated signatures to existing chess pieces. The procedure for this problem is listed in lines 27–29. After assigning the logical chess pieces on the $n \times n$ board, it solves the n-queen problem and assigns chess pieces according to the calculated solution.

The resource monitoring procedure is based on the assigned chess pieces. The queen nodes monitor vertical, horizontal, and diagonal nodes in the cloud computing system. The other nodes (except queens) provide their local information to the queen node when requested from the queen node. Since the queens cannot attack each other in the $n \times n$ board, system information of a node is forwarded to only one queen in the system.

Algorithm 2. The proposed cloud resource management algorithm.

```
begin virtual machine allocation
vm_info ← get_vminfo(user_request);
do
host_candidate1 ← get_randomhost(seed);
host_candidate2 ← get_randomhost(seed);
while check_requirement(vm_info, host_candidate1) && check_requirement(vm_info,
host_candidate2)
if check_utilization(host_candidate1) < check_utilization(host_candidate2) then
target_host ← host_candidate1;
nil_host ← host_candidate2;
else
target_host ← host_candidate2;
nil_host ← host_candidate1;
end if
if check_migration_suitability(nil_host, target_host) then
while retrieve_vm(nil_host) do
source_vm ← retrieve_vm(nil_host);
schedule_migration(source_vm, target_host);
end while
schedule_shutdown(nil_host);
end if
end
begin queen nodes assignment
num_host ← count_running_host();
num_queen ← round(sqrt(num_host));
num_chessman ← num_host;
while num_chessman < num_queen * num_queen do
tmp_chessman ← get_randomhost(seed);
assign_chessman(tmp_chessman, num_chessman + 1);
end while
solution ← calculate_nqueen(num_queen);
num ← 0;
while num < num_queen * num_queen do
node ← get_chessman(num);
assign_queen(node, solution);
num ← num + 1;
end while
end
begin resource monitoring
for all queen_node do
monitor vertical, horizontal, diagonal nodes according to predefined policy;
end for
end
```

4. Performance Evaluation

In this section, we present evaluation results of the proposed cloud management schemes for monitoring and resource allocation. We demonstrate that the proposed cloud management schemes reduce SLA violations for LMS and learning services, and improve load balancing, resource efficiency, and power consumption.

4.1. Experiments for LMS and Learning Services

We consider an LMS and learning services that provide basic LMS functions and learning environments. The client requests one of the following functions or tasks: (1) LMS navigations, (2) text learning contents, (3) text and media (images) learning contents, or (4) video learning contents. We simulate the multiple clients' requests based on a virtualized environment (Xen hypervisor and Openstack). We measure the number of failed tasks by increasing the number of clients from 50 to 550. The number of host machines is 15 and a host machine can be configured to hold up to 2 virtual machines.

For comparison, the default configuration of the cloud computing system is used. The proposed method is implemented on top of the cloud computing system and deployed with pre-configured settings (monitoring and resource allocation). The numbers of tasks are 3627, 9292, 15,756, 22,171, 28,505, and 33,649 when the numbers of clients are 50, 150, 250, 350, 450, and 550, respectively. The average number of requests for a client is about 64. Note that when a request does not respond within the pre-defined time, it can be considered as a failed task.

As shown in Figure 3, when the numbers of clients are 50 and 150, the results of the existing method and the proposed method are comparable. However, as the number of clients increases, the performance gap between the two methods is noticeable. When the number of clients is 250, the proposed method has 4× fewer failed tasks.

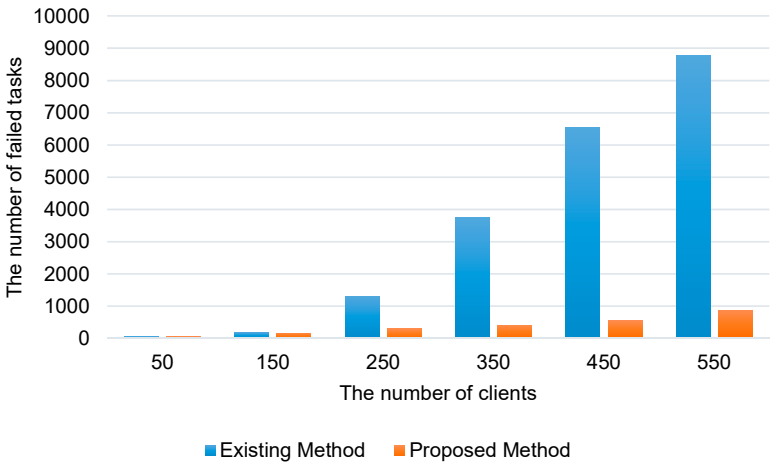


Figure 3. The number of failed tasks.

When the number of clients is 550, however, the proposed method has 10x fewer failed tasks. The main reason of task failures is due to the heavy load of virtual machines. Since the default configuration runs with the next fit policy and has no consideration of efficient resource management such as load balancing, our proposed method has much fewer failed tasks.

Figure 4 shows the number of SLA violations as the number of clients increases. The SLA violation is defined as a slow response for the requests and the SLA violation occurs when more tasks are allocated to overloaded virtual machines. Similar to the previous experiment, our proposed method has much fewer SLA violations compared with the existing method (about a 48× improvement when the number of clients is 550).

Figure 5 shows the average server utilization for host machines. Because the proposed method is capable of managing more tasks from clients (our method has fewer failed tasks than the existing method), the average server utilization for our proposed method is higher than that for the existing method. The percentages of the average server utilization for the existing method are 4.75%, 12.22%, 24.30%, 33.96%, 44.32%, and 50.58%, while those for the proposed method are 6.75%, 20.38%, 33.80%, 49.17%, 63.64%, and 74.64%. The implication of this result is that our method can deal with more

tasks while introducing fewer SLA violations, and our proposed cloud resource management scheme is effective.

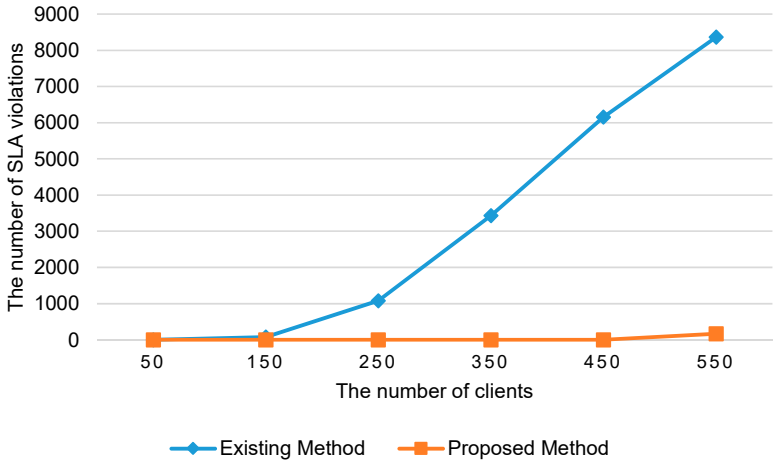


Figure 4. The number of service level agreement (SLA) violations.

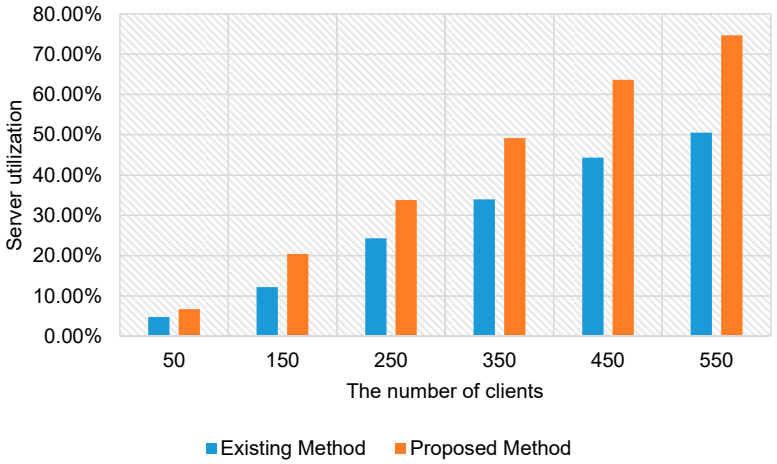


Figure 5. Server utilization for host machines.

4.2. Load Balancing

The performance of the proposed cloud management scheme is directly reflected in load balancing, since communication messages for cloud resource monitoring can be distributed from a single node to several nodes. With this in mind, the proposed scheme also supports fault tolerance and avoids the single point of failures. To show the performance improvement of the proposed cloud management scheme, we first focus on measuring load balancing of resources in cloud computing environments. To do so, we use the following balance metric:

$$(\mathrm{Max}_{\#\mathrm{ofMessages}} - \mathrm{Min}_{\#\mathrm{ofMessages}}) / \frac{\sum_{i=1}^{n} \#of\,Messages_i}{n}, \quad (1)$$

where n is the number of resources.

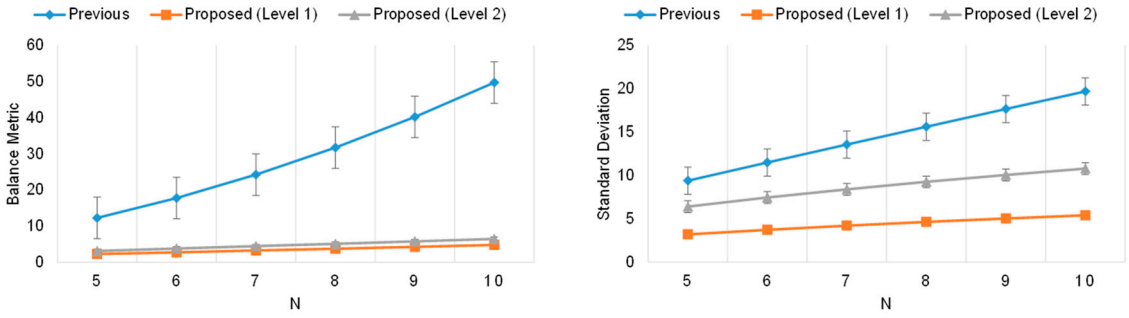
The load balancing refers to the distribution of message communication of the system. Thus, the performance metric is specifically designed to measure the difference between the maximum value and the minimum value. With the balance metric, we can easily observe the dispersion of message traffic for monitoring methods.

We consider two levels of fault tolerance for monitoring nodes in the system. Taking Figure 2c as an example, when the queens monitor other nodes in either the vertical or horizontal direction,

we can monitor all the nodes in the system. In addition, when the queens monitor other nodes in both vertical and horizontal directions, nodes in the system get monitoring request messages twice, which can provide fault tolerance. We denote the former level as Level 1 and the latter one as Level 2.

For instance, when a queen (from Figure 2c) on (3, 1) fails, nodes on (1, 1), (2, 1), (4, 1), (3, 2), (3, 3), and (3, 4) can be monitored by other queens on (1, 2), (2, 4), (4, 3), (1, 2), (4, 3), and (2, 4), respectively. Note that the failed queen can be noticed by the central cloud in the system since the central cloud manager regularly interacts with the designated queen nodes.

Figure 6 depicts the balance metric and standard deviation for the previous method (master–slave architecture) and the two proposed methods (Level 1 and Level 2) in terms of monitoring messages. Note that when N is 5, 6, 7, 8, 9, and 10, the number of resources is 25, 36, 49, 64, 81, and 100, respectively. As seen from the Figure, the balance metric increases as N increases. However, many increments of the balance metric can be seen from the previous method.



(**a**) Balance metric (lower is better) (**b**) Standard deviation

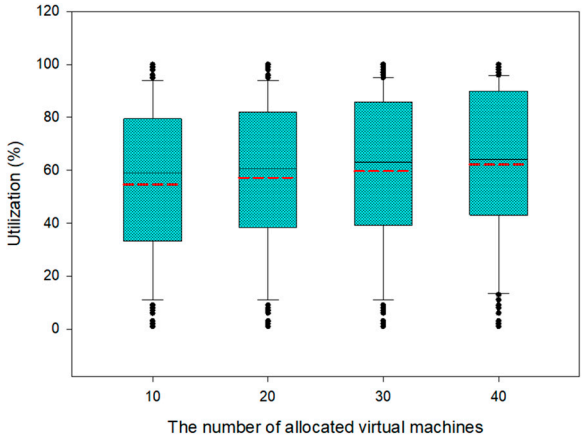
Figure 6. The balance metric and standard deviation for monitoring messages.

The values of the balance metric are about 12.24, 2.22, and 3.07 when N is 5, and 49.74, 4,73, and 6.42 when N is 10. The proposed method has a fault tolerance that is 10× and 7× less than the previous method, for Levels 1 and 2, respectively. This indicates the proposed cloud monitoring scheme improves load balancing, while providing fault tolerance in the presence of failures. Figure 6b shows the standard deviation of monitoring messages for cloud resources in the system. Considering the standard deviation, the proposed method has less dispersion of a dataset relative to its mean.

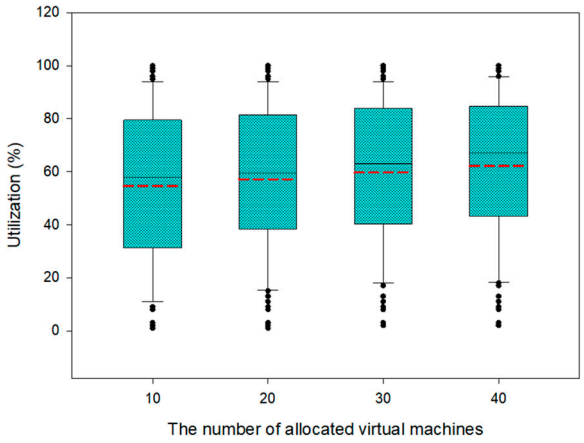
4.3. Resource Efficiency

Improvement of cloud consolidation for cloud computing environments is crucial. Cloud consolidation aims to make efficient use of virtualized cloud resources and prevent cloud servers from being under-utilized by packing more resources in physical hosts in the system. To show the efficient use of the proposed cloud resource management scheme, we measure the average, median, 10th, 25th, 75th, and 90th percentiles of CPU utilization by allocating the varying number of virtual machines from 10 to 40 as shown in Figure 7.

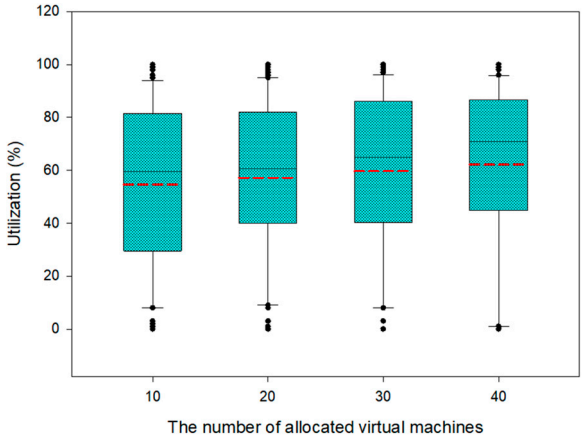
Comparing Figure 7a,b, we observe that those hosts running the same virtual machines have a noticeable difference than the previous method. The section between the 25th and 75th percentiles for the proposed method is shorter than that of the previous method. At the same time, the median value for the proposed method is higher than that for the previous method. Comparing the previous method and the proposed method with migration, the average utilization is about 62.1% and 68.3%, respectively. Note that the average value of Figure 7c includes powered off hosts, which are calculated as 0, and the number of powered off hosts is about 10% in this experiment.



(**a**) Previous method



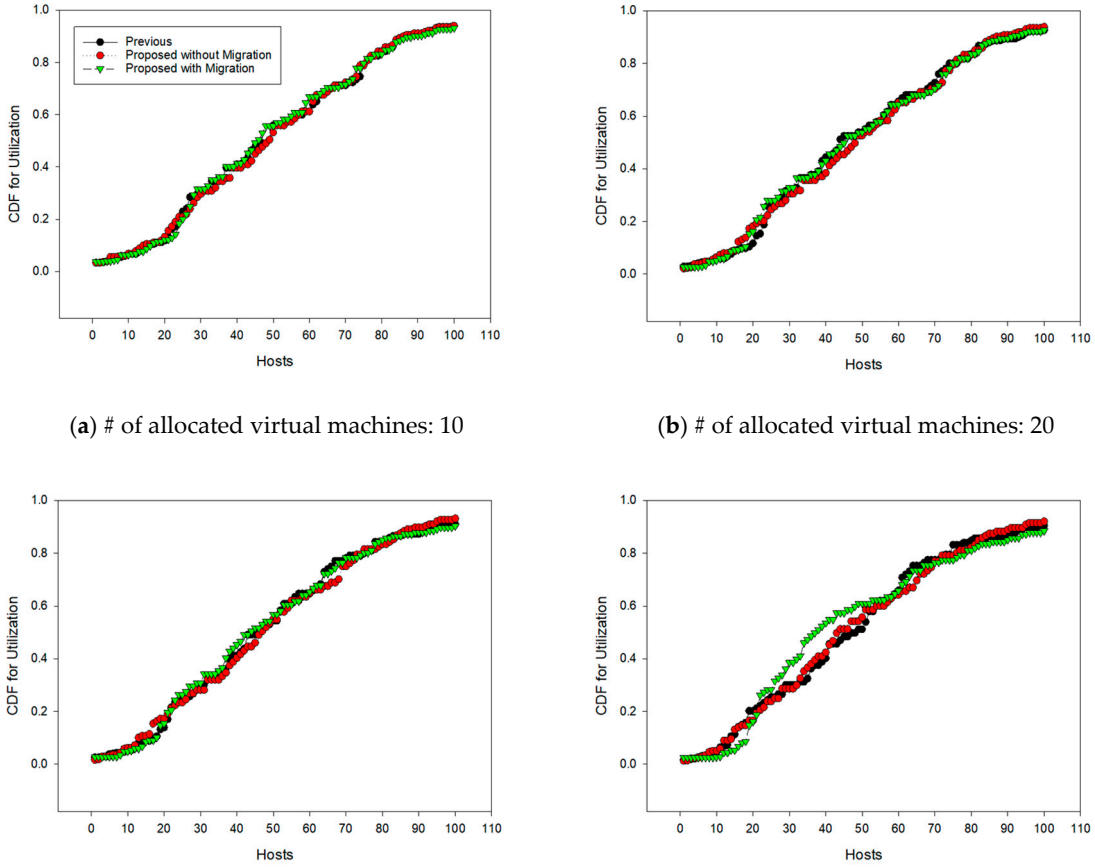
(**b**) Proposed method without migration



(**c**) Proposed method with migration

Figure 7. The average, median, 10th, 25th, 75th, and 90th percentiles of CPU utilization. The average and median values are marked as dashed lines and straight lines, respectively.

Figure 8 shows cumulative distribution functions (CDFs) for CPU utilization when the numbers of allocated virtual machines are 10, 20, 30, and 40. When the number of allocated virtual machines is 10, the three methods are comparable, where the differences in the standard deviation are about ± 1.0. However, when the number of allocated virtual machines is 40, the differences in the standard deviation are about ± 5.0. This is because the proposed cloud resource management scheme performs virtual machine migrations and shutdown processes when available.



(**a**) # of allocated virtual machines: 10

(**b**) # of allocated virtual machines: 20

(**c**) # of allocated virtual machines: 30

(**d**) # of allocated virtual machines: 40

Figure 8. Cumulative distribution functions (CDFs) for CPU utilization.

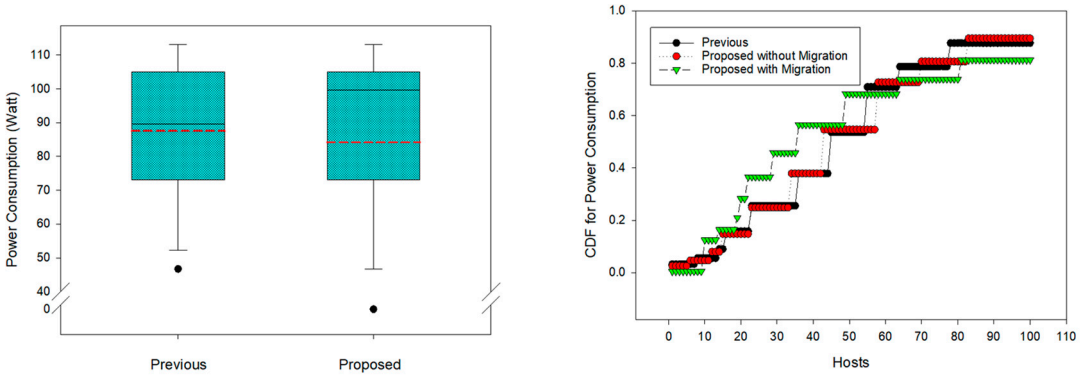
Taking Figure 8d as an example, CDF for migration of the proposed method with migration slowly grows and ranges from 0 to 20, since about 10% of hosts are powered off. In the middle ranges from 20 to 50, CDF for utilization of the proposed method with migration rises sharply in comparison with the other two methods. The reason for this is due to the cloud consolidation capabilities of the proposed method. More specifically, when performing the virtual machine allocation algorithm, it compares two resources and checks whether all virtual machines of a host can be migrated to another host. If it is available, all the virtual machines of a host are scheduled for migration, then the host can be powered off for power reduction.

4.4. Power Consumption

One of the reasons for cloud resource consolidation is to reduce power consumption and save energy. The proposed cloud resource management scheme not only improves resource efficiency but also reduces power consumption in cloud computing environments. To measure power consumption in cloud computing environments, we use the power model of an IBM server x3250 (Xeon X3470

2933 MHz (4 cores) CPU and 8GB RAM), which provides average active power data in 10% scale (http://www.spec.org/power_ssj2008/results/res2009q4/power_ssj2008-20091104-00213.html).

Figure 9 draws the percentiles and CDF for power consumption. As seen from Figure 9a, the median value of the proposed method is higher than that of the previous method, while the average value of the proposed method is lower than that of the previous one. The main reason for this is that the proposed method eliminates unnecessary hosts from running in the system, which ensures efficient resource management. The CDF for power consumption is similar to Figure 8d, and it confirms that about 10% of the hosts are powered off, and thus, saves unnecessary power consumption.



(**a**) (**b**)

Figure 9. The percentiles and CDF for power consumption. (**a**) The average, median, 10th, 25th, 75th, and 90th percentiles of power consumption; (**b**) CDF for power consumption when the number of allocated virtual machines is 40.

To calculate how much power consumption is saved, we measure the total power consumption and power reduction in Watts as shown in Figure 10. Note that we also use the power model of an IBM server x3250, and the power reduction is measured relative to the previous method. The total amount of power consumption for the previous method is about 8750 Watts, while that for the proposed method without and with migration is about 8710 and 8415 Watts, respectively. It translates into a power reduction of about 40 and 334 Watts for the proposed method without and with migration, respectively. The improvement comes from the fact that our proposed cloud resource management scheme migrates virtual machines to other hosts when allocating virtual machines and schedules hosts to be powered off whenever possible.

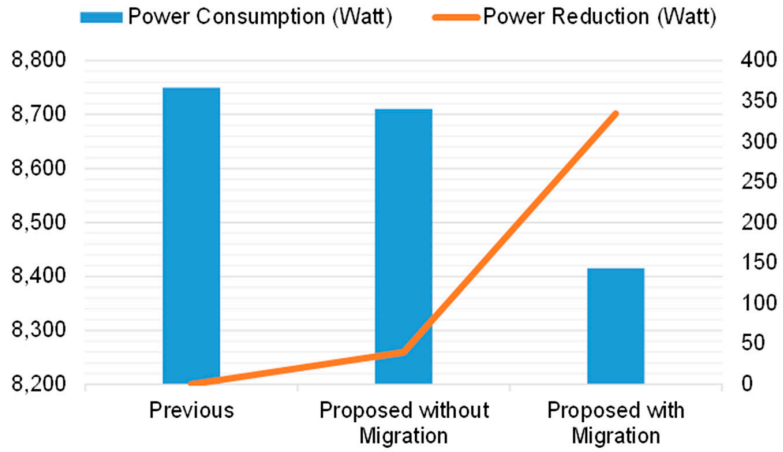


Figure 10. Power consumption and reduction.

4.5. Summary of Results and Discussion

Regarding the experiments for LMS and learning services, we observe a much lower number of failed tasks as the number of clients increases. This indicates that the existing method does not overcome the failed tasks in a scalable system. To mitigate the catastrophe for this scenario, we can add more physical machines in cloud computing environments. Nevertheless, the effect of adding physical machines applies to the proposed method as well. In this regard, our method is more beneficial for sustainable online lifelong education systems.

For the load balancing experiments, we use our own balance metric to measure the distribution of message communication. Although the balance metric is not a standard metric for load balancing, we observe that the balance metric is more effective at limiting the bottleneck than the standard deviation. Even though our method provides some degree of fault tolerance (Level 2 in Figure 6), the numbers show that our monitoring algorithm based on the n-queen problem effectively distributes the communication load.

As for resource efficiency, the proposed method with migration provides a more efficient use of physical machines than the previous method in cloud computing environments. The results depict that the proposed method has lower utilization than the previous method because we let the cloud resource manager shutdown unnecessary hosts. Note that the results are averaged for all the host machines in the system, while the results of Figure 5 are calculated only for active host machines. Since we focus on the use of online lifelong education services, we leave more sophisticated cloud consolidation techniques as future work.

Power consumption in datacenters is a global problem, and datacenters consume more than 2% of the world's electricity. The experiments in Section 4.4 consider only CPU in Watts, and do not include cooling-related power consumption. Although our proposed method with migration reduces power consumption with respect to the previous method, we conjecture that the real implementation of our proposed method in datacenters can further reduce the total power consumption since its cooling related power consumption also can be reduced. Note that the proposed method with migration features host shutdown.

5. Conclusions

As online lifelong education has become popular, cloud computing systems have received a lot of attention as they can provide computing resources and persistent storage. To offer more reliable and efficient cloud services for online lifelong education, we proposed an efficient cloud architecture management algorithm. The proposed virtual machine allocation reduces the complexity of the resource allocation process in virtualized cloud computing environments, and the monitoring nodes assignment algorithm based on the traditional n-queen problem resolves the potential problems in existing cloud computing systems (i.e., single point of failure and bottleneck). Through our proposed cloud resource management, we expect that online learners of the system will experience more pleasant learning activities since our experimental results show that our proposed method has 10× fewer failed tasks for e-learning requests and 10× improvement of load balancing while reducing SLA violations and power consumption. Future work includes deployments of various online learning applications and services to the cloud computing environments and evaluation on serverless computing environments.

Author Contributions: Methodology and writing-original draft preparation, T.K.; conceptualization and writing-review, J.L. as the corresponding author. All authors read and approved the final manuscript.

Funding: This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2017-S1A5B4055939 and NRF-2018R1D1A1B07045838) and by Hankuk University of Foreign Studies Research Fund of 2019.

Acknowledgments: We thank the anonymous reviewers for their careful reading and insightful suggestions to help improve the manuscript.

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

References

1. Tchamyou, V.S. Education, lifelong learning, inequality and financial access: Evidence from african countries. *Contemp. Soc. Sci.* **2018**, *2018*, 1–19. [CrossRef]
2. Agudo-Peregrina, Á.F.; Hernández-García, Á.; Pascual-Miguel, F.J. Behavioral intention, use behavior and the acceptance of electronic learning systems: Differences between higher education and lifelong learning. *Comput. Hum. Behav.* **2014**, *34*, 301–314. [CrossRef]
3. Luo, H.; Andersson, B.; Tang, J.Y.M.; Wong, G.H.Y. Applying item response theory analysis to the montreal cognitive assessment in a low-education older population. *Assessment* **2019**. [CrossRef] [PubMed]
4. Zhang, D. *Community-Based Lifelong Learning and Adult Education: Situations of Community Learning Centres in 7 Asian Countries*; Routledge: London, UK, 2018; pp. 1–3.
5. Alastalo, M. Eurostat: Making europe commensurate and comparable. In *Policy Design in the European Union: An Empire of Shopkeepers in the Making?* Heiskala, R., Aro, J., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 87–110.
6. Shea, P.; Bidjerano, T. Online learning in the 30 community colleges of the state university of new york: Differences in outcomes between classroom and online coursework. In *EdMedia + Innovate Learning 2017*; Johnston, J.P., Ed.; Association for the Advancement of Computing in Education (AACE): Washington, DC, USA, 2017; pp. 1192–1198.
7. Deming, D.J.; Goldin, C.; Katz, L.F.; Yuchtman, N. Can online learning bend the higher education cost curve? *Am. Econ. Rev.* **2015**, *105*, 496–501. [CrossRef]
8. You, J.W. Identifying significant indicators using lms data to predict course achievement in online learning. *Internet High. Educ.* **2016**, *29*, 23–30. [CrossRef]
9. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A.D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; et al. A view of cloud computing. *Commun. ACM* **2010**, *53*, 50–58. [CrossRef]
10. Buyya, R.; Yeo, C.S.; Venugopal, S.; Broberg, J.; Brandic, I. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.* **2009**, *25*, 599–616. [CrossRef]
11. Foster, I.; Zhao, Y.; Raicu, I.; Lu, S. Cloud computing and grid computing 360-degree compared. In Proceedings of the 2008 Grid Computing Environments Workshop, Austin, TX, USA, 12–16 November 2008; pp. 1–10.
12. Zhang, Q.; Cheng, L.; Boutaba, R. Cloud computing: State-of-the-art and research challenges. *J. Internet Serv. Appl.* **2010**, *1*, 7–18. [CrossRef]
13. Buyya, R.; Yeo, C.S.; Venugopal, S. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications, Dalian, China, 25–27 September 2008; pp. 5–13.
14. Shila, D.M.; Shen, W.; Cheng, Y.; Tian, X.; Shen, X.S. Amcloud: Toward a secure autonomic mobile ad hoc cloud computing system. *IEEE Wirel. Commun.* **2017**, *24*, 74–81. [CrossRef]
15. Al-Shara, Z.; Alvares, F.; Bruneliere, H.; Lejeune, J.; Prud'Homme, C.; Ledoux, T. Come4acloud: An end-to-end framework for autonomic cloud systems. *Future Gener. Comput. Syst.* **2018**, *86*, 339–354. [CrossRef]
16. Gill, S.S.; Chana, I.; Singh, M.; Buyya, R. Chopper: An intelligent qos-aware autonomic resource management approach for cloud computing. *Clust. Comput.* **2017**, *21*, 1203–1241. [CrossRef]
17. Dustdar, S.; Gambi, A.; Krenn, W.; Nickovic, D. A pattern-based formalization of cloud-based elastic systems. In Proceedings of the Seventh International Workshop on Principles of Engineering Service-Oriented and Cloud Systems, Florence, Italy, 23–23 May 2015; pp. 31–37.
18. González-Martínez, J.A.; Bote-Lorenzo, M.L.; Gómez-Sánchez, E.; Cano-Parra, R. Cloud computing and education: A state-of-the-art survey. *Comput. Educ.* **2015**, *80*, 132–151. [CrossRef]
19. Zhao, Q. Application study of online education platform based on cloud computing. In Proceedings of the 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), Yichang, China, 21–23 April 2012; pp. 908–911.

20. Khmelevsky, Y.; Voytenko, V. Cloud computing infrastructure prototype for university education and research. In Proceedings of the 15th Western Canadian Conference on Computing Education, Kelowna, BC, Canada, 7–8 May 2010; pp. 1–5.
21. Alabbadi, M.M. Cloud computing for education and learning: Education and learning as a service (elaas). In Proceedings of the 2011 14th International Conference on Interactive Collaborative Learning, Piestany, Slovakia, 21–23 September 2011; pp. 589–594.
22. Jararweh, Y.; Alshara, Z.; Jarrah, M.; Kharbutli, M.; Alsaleh, M.N. Teachcloud: A cloud computing educational toolkit. *Int. J. Cloud Comput.* **2013**, *2*, 237–257. [CrossRef]
23. Sonwalkar, N. The first adaptive mooc: A case study on pedagogy framework and scalable cloud architecture—Part I. *MOOCs FORUM* **2013**, *1*, 22–29. [CrossRef]
24. Kim, S.; Song, S.-M.; Yoon, Y.-I. Smart learning services based on smart cloud computing. *Sensors* **2011**, *11*, 7835. [CrossRef] [PubMed]
25. Mhouti, A.E.; Erradi, M. Towards a smart learning management system (smart-lms) to improve collaborative learning in higher education. In Proceedings of the 3rd International Conference on Smart City Applications, Tetouan, Morocco, 10–11 October 2018; pp. 1–9.
26. Ding, J.; Xiong, C.; Liu, H. Construction of a digital learning environment based on cloud computing. *Br. J. Educ. Technol.* **2015**, *46*, 1367–1377. [CrossRef]
27. Jayasena, K.P.N.; Song, H. Private cloud with e-learning for resources sharing in university environment. In *E-Learning, E-Education, and Online Training*; Vincenti, G., Bucciero, A., Helfert, M., Glowatz, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 169–180.
28. Mitzenmacher, M. The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distrib. Syst.* **2001**, *12*, 1094–1104. [CrossRef]