IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 4, 2020, accepted March 19, 2020, date of publication April 21, 2020, date of current version May 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2989303

Leveraging N-1 Queues to Improve the Energy Efficiency of Scalable Computing

CHENG HU[1], HUAN LUO[2], AND MINGDONG TANG[1]
[1]School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou 510006, China
[2]College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China

Corresponding authors: Huan Luo (hluo@fzu.edu.cn) and Mingdong Tang (mdtang@gdufs.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61976061, in part by the Science and Technology Planning Project of Guangzhou under Grant 202002030239, and in part by the Characteristics and Technological Innovation Project of Guangdong University of Foreign Studies under Grant 18TS21.

ABSTRACT The clusters in a blockchain computing system can be constructed to be elastic, thus supporting scalable computing and improving energy efficiency. To form an elastic cluster, the service nodes are dynamically divided into the working nodes and the reserved nodes. Specifically, the working nodes are active to meet the computing requirements of workloads, while the reserved nodes are switched to a low-power state for energy saving. Traditionally, workloads are distributed to working nodes in the mode of N-N service queues. But in this mode, the Quality of Service (QoS) of different working nodes may be diverse, because the requirements are various for the accumulated requests in different working nodes. As a result, the overall system capability is not sufficiently utilized, and the overall system QoS is dragged down. In this paper, we propose an N-1 queueing and on-demand resource provisioning method to process workloads in the mode of N-1 service queues. Different from N-N service queues, N-1 service queues prohibit the accumulation of requests in working nodes. Thereby, once there are idle working nodes, waiting requests can immediately be delivered to them. As a result, all the working nodes are sufficiently utilized, and the overall QoS is improved. Accordingly, after using the N-1 service queues, fewer working nodes are enough to meet the same Service Level Agreement (SLA) on same workloads. In addition, by using a resource demand monitor module, our method dynamically readjusts the number of working nodes to match workload demand. Finally, the energy efficiency of an elastic cluster can be measurably improved, due to that fewer working nodes are powered on while the same SLA can be met.

INDEX TERMS Blockchain computing systems, energy efficiency improvement, service queues, scalable computing.

I. INTRODUCTION

Owing to the trust free nature of blockchain [1], [2], the development of blockchain computing systems is greatly promoted. Blockchain computing systems are now widely applied in different research areas, such as establishing secure data transmission in Internet of Things (IoT) [3], [4], building conventional cloud exchange markets in cloud service [5], [6], etc. The emergence of a large number of such applications places a great demand on computing resources. [7]. Accordingly, the scales of blockchain computing systems expand quickly to meet the demands. Meanwhile, energy requirements are raised to sustain the large scale systems. The huge requirements of energy supply bring not only the financial problem but also an environmental pollution problem [8], [9].

To improve the system energy efficiency and reduce energy waste, elastic clusters are introduced to provide scalable computing for fluctuant workloads. An elastic cluster allows the number of working nodes to be dynamically adjusted. Thus, for saving energy in an elastic cluster, only partial nodes keep working, and the others are maintained at a low-power state. In general, a traditional elastic cluster system [10]–[12] can be considered as a two-layer structure, where the first layer is a manager (or several managers for large-scale systems), and the second layer is working nodes. Because the manager and working nodes can maintain their respective queues to accumulate the requests which form system workloads, the requests are generally queued and served in the mode of N-N service queues (or called N-N queues for short). In this

The associate editor coordinating the review of this manuscript and approving it for publication was Hong-Ning Dai.

mode, the requests can be accumulated in both of the two layers.

However, the arrivals and the required service time of requests are generally nonuniform. Therefore, even each working node serves a same number of requests in the N-N queues mode, the time needed to accomplish the requests is different. The working nodes, which receive relatively uniformly distributed requests, tend to obtain a lower response time than the ones with nonuniform distributed requests. In addition, the requests received by some working nodes probably require a low service time, thus these working nodes can achieve a high service rate. As a result, some working nodes have much idle time, while some other working nodes might struggle for services. Different from N-N queues, N-1 service queues mode (or called N-1 queues for short) only allow the requests to be accumulated in the first layer, and the requests can only be directly handled in the second layer. Once a working node is idle, a waiting request is immediately distributed by the managers to the node. Therefore, the assigned workloads to each working node are more proportional, when adopting N-1 queues instead of N-N queues. As a result, after using N-1 queues, the overall QoS of a cluster can be improved.

Therefore, this paper[1] proposes an N-1 queueing and On-Demand resource provisioning (N1OD) method for elastic clusters. N1OD makes the queues in the manager and working nodes follow the mode of N-1 queues. In contrast to the traditional N-N queues, the N-1 queues significantly improve the service efficiency of working nodes. Consequently, to maintain the same Service Level Agreements (SLA) under the same workloads, fewer working nodes are needed when adopting N-1 queues. In addition, leveraging some queueing theories related to N-1 queues, N1OD dynamically provides suitable resources for workloads. Thus, the energy usage wasted on supplying surplus working nodes can be reduced. Ultimately, at the same service level, our method can achieve notable improvement on the energy efficiency of elastic clusters.

The main contributions of this paper are as follows:

1) We explore the two-layer structure of a traditional elastic cluster which constitutes a blockchain computing system, and perform analyses to compare the system performances of the N-N queues and the N-1 queues.
2) We propose the N1OD method to replace the queues mode with the N-1queues mode. In addition, we realize a dynamical on-demand resource provision in N1OD, with the help of some related queueing theories to perceive the resource demand of workloads.
3) We perform extensive experiments to attest the effectiveness of our method. For comparison, our method along with other three state of the art methods are tested in the experiments. Finally, we carefully evaluate all these methods, with full discussions on the experimental results.

The rest of this paper is structured as follows. Section II discusses the related work. Section III introduces the architecture of a traditional elastic cluster, and Section IV proposes the models and related analysis for the elastic cluster. Section V introduces our proposed method N1OD in detail. Section VI carries out the experimental evaluation to attest the effectiveness of our method. Finally, Section VII concludes this paper.

II. RELATED WORK

Elastic clusters are proposed by many researchers to dynamically adjust the number of working nodes, thus cutting down the system energy cost and improving the system energy efficiency. Hameed *et al.* survey several resource provision/allocation studies in their literature [14]. They point out that, leveraging scalable designs to support on-demand resource provision/allocation, is an efficient way to save system energy cost and improve system energy efficiency. The scalable designs can be achieved by various methods, such as resource allocation adaption policy [15], objective function [16] and so on.

The related work is introduced in two categories. In the first category, the energy cost of system is cut down by providing service resources elastically. However, the provided service resources should meet the requirements of workloads, thus satisfying a premise that the system QoS should be guaranteed. If the system service efficiency can be improved, fewer service resources are required by a same workload to obtain an equivalent QoS. Therefore, in the second category, the involved studies further improve the service efficiency of system, thus the system energy efficiency can be further improved.

A. THE FIRST CATEGORY

With scalable computing resources in the form of virtual machine (VM) instances to run jobs, Xu *et al.* [17] propose a heterogeneity and interference-aware VM provisioning framework, Heifer. Heifer can predict the performance of MapReduce applications by a lightweight performance model which regards both the online-measured resource utilization and VM interference. Accordingly, Heifer gears the number of VM instances to provide a fitting capability for tenant applications. With extensive prototype experiments, they show that Heifer can guarantee the job performance while saving the job budget for tenants. In other words, the energy efficiency of computing resources is improved, due to the reduction on the energy cost for a same workload. Smart *et al.* [18] use a multi-objective goal attainment algorithm to improve the energy efficiency of the storage drives in a custom built storage cluster. The cluster contains multiple drives, and the goal of the algorithm is to optimally assign individual commands to drives thus minimizing the command energy usage. In this work, rather than working nodes, the service units are mainly regard as the

[1] A preliminary version of this paper [13] appears in Proceedings of the 2020 International Conference on Blockchain and Trustworthy Systems (BlockSys'2020).

storage drives. The power of idle storage drives is much lower than that of active ones. Thus, with a minimal number of drives to execute commands, the energy efficiency can be significantly improved.

B. THE SECOND CATEGORY

To improve the system service efficiency, many researchers make a great deal of efforts. Lu *et al.* [19], [20] notice that the bursts of workloads greatly increase the system burden, so they propose a decomposing method to improve the system service efficiency on the bursts. In the method, the workloads in the bursts are first decomposed. Specifically, the I/O requests, which are out of the service limit, are extracted from the workloads. Then, the extracted requests are postponed, and finally recombined into the slack of later workloads. However, the method does not work on normal workloads. In other words, the service efficiency on normal workloads remains the same without improvement. Mardukhi *et al.* [21] use a genetic algorithm to select the optimal service for tasks. Their algorithm can decompose the global constraints into local constraints, and finally select an optimal service through a simple linear search. Because the tasks are optimally assigned to appropriate services, the service efficiency is improved to achieve a shorter computation time for workloads. Zhang *et al.* [22] propose a service curve-based QoS algorithm to support three types of applications in a same storage system. Specifically, three scheduling queues with different priorities are adopted to schedule the I/O requests respectively in latency guarantee applications, IOPS guarantee applications and best-effort applications. By using the service curve-based QoS algorithm, the scheduling can take account of the urgency status of I/O requests. Besides, to avoid failures on QoS targets, the algorithm can reschedule certain requests from the fairness queue, by migrating them to other appropriate queue.

In our earlier work [13], we notice that the service queues in traditional elastic clusters are with the mode of N-N queues. But the overall service capability is not sufficiently utilized under the N-N queues. Therefore, in the earlier work, we realize an N-1 queueing method to improve the service efficiency of elastic clusters. By comparison, this paper makes significant extensions on the earlier work. The extensions include: adding more details and analyses to elaborate the N-1 queues; proposing the new N1OD method rather than the preliminary method; improving the experimental evaluation with new elaborate experiments, etc. In our N1OD method, the N-1 queues are handy and with universal applicability to improve the system service efficiency on general workloads. In addition, N1OD can dynamically provide advisable resources for fluctuant workloads, leveraging queueing theorems to obtain the perception on the resource demand of workloads. What's more, in our experiments, real-world IO trace logs are replayed to produce approximate workload conditions as in blockchain computing systems.

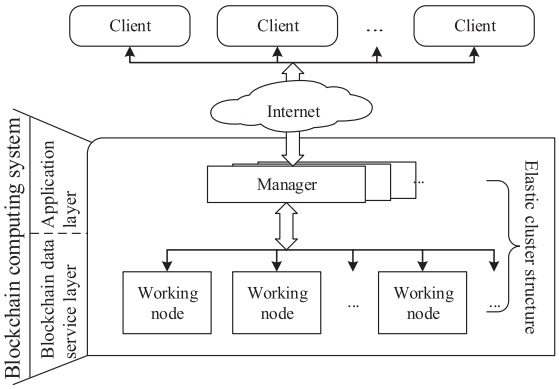


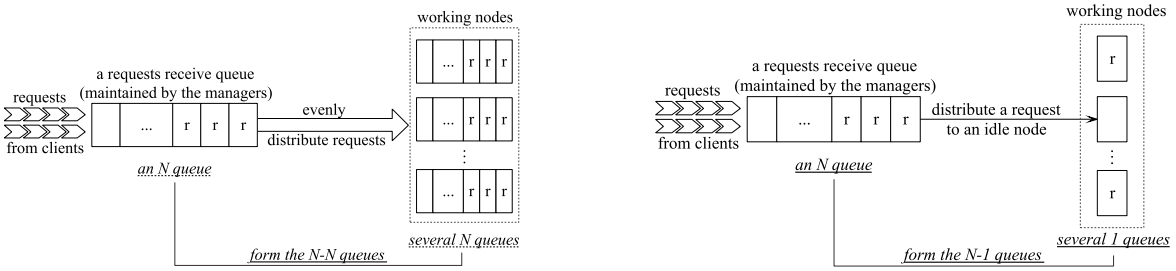
FIGURE 1. The structure of a blockchain computing system.

III. SYSTEM ARCHITECTURE

Figure 1 shows a blockchain computing system which supports scalable computing. As shown in Figure 1, the system can be divided into two layers. The first layer is the application layer which consists of managers, and the second layer is the blockchain data service layer which consists of working nodes. When clients send transactions via the Internet to the blockchain computing system, these transactions are first received by the managers. Then, the managers either update existing blockchain data or create new blockchain data, according to the requirements of the transactions. The blockchain data are maintained in the second layer, so the operations of updating existing blockchain data and creating new blockchain data are implemented in the second layer.

As shown in Figure 1, the blockchain computing system can be regarded as a traditional elastic cluster structure. The structure contains two layers including the application layer and the blockchain data service layer. In the first application layer, the transactions received are translated into several data manipulation requests. These requests are first accumulated in the managers, and then distributed by the managers to appropriate working nodes. Therefore, no matter what blockchain applications are deployed to the blockchain computing system, the system workloads can be regard as these requests.

In general, in the managers, the accumulated requests are queued in a First In First Out (FIFO) way. Therefore, no matter how many managers the system has, we can assume that only one queue is maintained by the managers. In addition, the requests are evenly distributed to the second layer with a specific manner, e.g., a round-robin manner. In the second layer, only the working nodes provide services for the requests. When requests are accumulated in a working node, these requests are queued and served by the working node in the FIFO way. The number of working nodes can either be decreased by switching some working nodes to a low-power state as reserved nodes, or be increased by resuming some reserved nodes from the low-power state as working nodes. Because reserved nodes should first be resumed as working nodes for service, they are not shown in Figure 1.



(a) The N-N queues mode

(b) The N-1 queues mode

FIGURE 2. Illustrations for the N-N and N-1 queues modes.

Because both the two layers can maintain their respective queues to accumulate the requests, such mode of handling requests is represented as the N-N queues mode. The first 'N' means that, when requests are first received by the cluster managers, multiple requests can be queued in the service queue maintained by the managers. The second 'N' means that, when requests are then delivered by the managers to appropriate working nodes, the service queues of the working nodes can also queue multiple requests. If the requests are only allowed to be accumulated in the first layer, and can only be directly handled in the second layer. The N-N queues mode is changed into the N-1 queues mode. In the N-1 queues mode, once a working node is idle, a waiting request in the manager is immediately distributed to the working node.

IV. SYSTEM MODEL AND THEORETICAL ANALYSIS

A. SYSTEM MODEL

According to queueing theory [23], a queueing system can work with different service queue modes. Among these service queue mode, the multiple single-server queues mode corresponds to the previously mentioned N-N queues mode. In addition, the multi-servers queue mode corresponds to the N-1 queues mode. Two subfigures are presented in Figure 2 to show the N-N queues mode and the N-1 queues mode. As shown in Figure 2a, each of the working nodes has its own single-server queue, thus there are multiple single-server queues. While, as shown in Figure 2b, the N queue maintained by the managers is a common multi-servers queue, which are served by all the working nodes.

Therefore, a traditional elastic cluster can be modeled as a queueing system works with the multiple single-server queues mode. In this queueing system, requests are evenly distributed by the managers to servers. Therefore, each server theoretically receives the same number of requests. However, the arrivals and the required service time of requests are generally nonuniform. Therefore, although each server serves the same number of requests, the time needed to finish the requests is different. The servers, which receive relatively uniformly distributed requests, tend to obtain a lower response time than the ones with nonuniform distributed requests. In addition, the requests received by some servers probably require a low service time, thus these servers can achieve a high service rate. As a result, some servers have much idle time, while some other servers might struggle for services.

If requests are prohibited to be accumulated in working nodes, the traditional elastic cluster's N-N queues mode is changed into the N-1 queues mode. After changed into the N-1 queues mode, the elastic cluster can instead be modeled as a queueing system works with the multi-servers queue mode. In this queueing system, no requests can be accumulated in the servers. Once a server is idle, a waiting request is immediately distributed to the server.

Generally, request arrivals are seen as a Poisson process, and request service time is regarded as an exponential distribution [24]. According to Kendall's notation[2] [24], [25], an elastic cluster system with the N-N queues mode can be modeled as multiple M/M/1 queueing models, each model represents a working node. Besides, an elastic cluster system with the N-1 queues mode can be modeled as an M/M/n queueing model, where, n working nodes serve the common queue.

B. THEORETICAL ANALYSIS

Based on the existing research on queueing theory [23]–[25], two formulas can be used to analyze the performances of the multiple M/M/1 queueing models and the M/M/n queueing model. The related notations are summarized in Table 1. The first formula is for the multiple M/M/1 queueing models. For the models, because requests are evenly distributed to working node, each model has the same request arrival rate. Assume that the request arrival rate in each model is equal to λ, the mean waiting time of requests in each model is

$$W = \frac{\rho}{\mu - \lambda}. \tag{1}$$

The other formula is for the N-1 queues. For the N-1 queues, with the same assumption, if there are n working nodes, the whole arrival rate is $n\lambda$. In this case, the mean waiting

[2]With Kendall's notation, three factors written as the form of A/S/c are used to represent a queueing model. In that form, "A" refers to the distribution of inter-arrival time, "S" refers to the distribution of service time and "c" refers to the number of service servers. If the request arrivals follow a Poisson process (or called random process), or the service time required follow an exponential distribution, the notation 'M' is used.

TABLE 1. Notation descriptions.

Notation	Description
n	the number of working nodes
λ	request arrival rate; mean number of arrivals per second
μ	service rate; mean number of requests can be finished per second
S	mean service time for requests; amount of time being served, not counting time waiting in the queue ($S = n/\mu$)
ρ	utilization, also called traffic intensity ($\rho = \lambda/\mu$)
w	the waiting time of a requests, also called response time; time before service
W	the mean value of w for requests
T	mean time a request spends in system, i.e., residence time ($T = W + S$)
$C(n, \rho)$	Erlang C formula [26], [27], i.e. the probability that all servers are busy under M/M/n queueing model
$m_w(p)$	the maximum waiting time which is satisfied by $p\%$ requests
$M_w(p)$	the expected value of $m_w(p)$ set as the SLA
t_d	the time cost for resuming a node in low-power state
n_r	the number of working nodes at least required to satisfy the SLA

time of requests can also be calculated as follows,

$$W = \frac{C(n, \rho)}{n} \times \frac{S}{1 - \rho}. \tag{2}$$

To give an intuitive illustration, we assume that the whole arrival rate of requests is 400 per second, and the mean service time of requests S is 5 milliseconds. In addition, there are 4 working nodes (the λ of each node is $400/4$) in the system. With this assumption, if all the working node can be modeled as 4 M/M/1 queueing models, we can theoretically calculate the mean waiting time of requests, by leveraging (1),

$$\begin{aligned} W &= \frac{\rho}{\mu - \lambda} \\ &= \frac{\frac{400}{4} \times 0.005}{\frac{1}{0.005} - \frac{400}{4}} \\ &= 0.005 \ seconds. \end{aligned}$$

While, if the system can be modeled as an M/M/4 queueing model. We can also theoretically calculate the mean waiting time of requests through (2),

$$\begin{aligned} W &= \frac{C(n, \rho)}{n} \times \frac{S}{1 - \rho} \\ &= \frac{C(4, \frac{400}{4 \times \frac{1}{0.005}})}{4} \times \frac{0.005}{1 - \frac{400}{4 \times \frac{1}{0.005}}} \\ &\approx 0.00043 \ seconds. \end{aligned}$$

Even the number of working nodes is reduced to 3, the mean waiting time is $0.00\dot{2}$ seconds.

Therefore, if the actual and theoretical conditions are consistent, with the same hardware conditions and the same workloads, the service efficiency of N-1 queues is much higher than that of N-N queues. In other words, with N-1 queues, fewer working nodes are required to maintain the same SLA. However, the actual situations can be more complex. For example, if $S \geqslant 10ms$ in the above illustration, the value of ρ will be equal to or larger than 1, and zero or minus will appear in the denominators. This means that, working nodes are not adequate to handle requests, and the number of waiting requests will increase infinitely.

V. METHOD IMPLEMENTATION

In this paper, we propose the N1OD method to makes the queues of traditional elastic clusters follow the form of N-1 queues. Thus, the service efficiency of traditional elastic clusters can be improved to a great extent. In addition, N1OD maintains a Resource Demand Monitor Module (RDMM) which keeps tracking the current workload demand on the number of working nodes. Accordingly, owing to RDMM, N1OD is capable of providing suitable resources for workloads. Finally, with fewer but adequate working nodes, N1OD can significantly improve the energy efficiency of traditional elastic clusters.

A. FORM THE N-1 QUEUES

The key of transforming N-N queues to N-1 queues is to restrict the working nodes to directly execute requests without accumulating. Therefore, for an elastic cluster, N1OD method is implemented in managers. Specifically, as shown in Figure 2b, requests are first received by managers, and each time managers distribute a request to a working node which is idle (that is no request waiting or being executed in the node). If all the working nodes are not idle, managers accumulate new arrival requests and keep tracing the state of working nodes. Once any working node is detected to be idle, managers distribute the accumulated requests one by one. In this way, the traditional N-N queues of an elastic clusters are transformed to N-1 queues.

B. IMPLEMENT THE RDMM

To implement RDMM, we first propose an adaptive model. With this model, a good estimation on the resource demand of workloads can be made. According to the theoretical analysis in Section IV-B, N-1 queues can be modeled as M/M/n queueing model. In addition, we set an SLA that the waiting time of at least $p\%$ requests cannot be higher than an expected value $M_w(p)$, due to the reason that waiting time well reflect the QoS of a system [28]–[30]. By the way, distinguishing from $M_w(p)$, $m_W(p)$ is the maximum waiting time which is satisfied by $p\%$ requests.

Based on the existing research on queueing theory [23]–[25], the maximum waiting time which is satisfied by $p\%$ requests can be theoretically calculated for N-1 queues as follows:

$$m_w(p) = \frac{S}{n(1 - \rho)} \times ln(\frac{C(n, \rho)}{100 - p}). \tag{3}$$

Because $\rho = \lambda/\mu = \lambda S/n$, the above equation can further be transformed to,

$$m_w(p) = \frac{S}{n(1 - \lambda S/n)} \times ln(\frac{C(n, \lambda S/n)}{100 - p}). \tag{4}$$

Therefore, the number of working nodes required to satisfy the SLA, n_r, can be obtained, by substituting the actual values into (4) and solving out n. Notice that, (4) is for the M/M/n queueing model. Because the actual environments are changeable [15], to adapt to the actual situations, we add an accommodation coefficient θ into the formula,

$$m_w(p) = \theta[\frac{S}{n(1-\lambda S/n)} \times ln(\frac{C(n,\lambda S/n)}{100-p})]. \quad (5)$$

The actual situations of workloads might be variable, thus θ should be continuously updated.

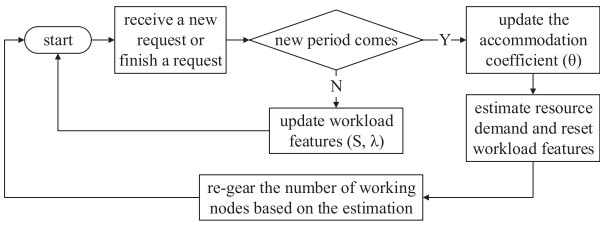


FIGURE 3. The workflow of RDMM.

We present a workflow in Figure 3 to clarify the process of RDMM. As shown:

1) when a new request arrives or a request in service is accomplished, RDMM checks whether a new time period comes. Here, all time periods (or just called period) are set with the same length of time.
2) If current time is still in the same period as before (the process labeled with letter 'N'), RDMM just updates the workload features S and λ in current period.
3) While, if current time is in a new period, RDMM calculates the latest accommodation coefficient θ with (5), by substituting the actual values of S, λ, n and $m_W(p)$ in the previous period.
4) Next, RDDM estimates the resource demand of current workload, n_r, by calculating n with (5) (setting $m_W(p) = M_w(p)$).
5) Finally, RDMM adjusts the number of working nodes base on the estimated value.

VI. EXPERIMENTAL EVALUATION

A. EXPERIMENT SETUP

In the evaluation, we conduct simulation experiments to demonstrate the effectiveness of our method. In the elastic cluster, there are 20 nodes. The involved simulation parameters for the cluster are set according to the parameters measured on the typical cluster nodes [31]. Concretely, these parameters include the power of a working node in different states, the time delay and energy consumption of switching a working node to a low power state (or vice versa). The details for these parameters are shown in our previous work [32].

The system workloads are generated from a real-word IO trace, where the service time of a request depend on the IO type and the length of the required data. Specifically, we use the "K5cloud" [33], [34] trace, which captures the IO access logs in the FUJITSU K5 cloud service. The reason we choose the K5cloud trace is because the FUJITSU K5 cloud service covers a large number of large-scale parallel systems, including lots of data centers and business systems. Such service scenario is in line with our research scenario. To produce quantities of requests, only the daytime records in weekdays are used.

Different SLAs are set for evaluation, that the value of $M_w(p)$ are respectively set to 10, 20, 40, 60, 80, 100 milliseconds and p is set to 98. Among these values of $M_w(p)$, 100 milliseconds are a widely used value [9], such as used for determining whether the network delay for playing online games is tolerable [35]. The high value of p, 98%, can help to reflect the maximum system service capability with different methods. Accordingly, the system QoS is poor, when the actual value of $m_w(p)$ is larger than $M_w(p)$, i.e., the SLA is not satisfied. Otherwise, if the actual value of $m_w(p)$ is lower than $M_w(p) \times 20\%$[3], surplus working nodes might be provided thus the SLA is excessively satisfied.

In our experiments, other three methods are hand-picked for comparison. Along with our N1OD method, all the methods are summarized as follows.

- **Baseline**: The baseline method works with the traditional N-N queues. To sufficiently utilize the capacity of working nodes, each time a request is distributed to the working node with the shortest queue length. In addition, when the SLA is not satisfied or excessively satisfied as we discussed in the previous paragraph, this method immediately adjusts the number of working nodes. In this way, one working node is added when the SLA is not satisfied and reserved when the SLA is excessively satisfied.
- **EN-N**: The Energy-saving N-N queueing (EN-N) method works like the baseline method, but the timing for adjusting the number of working nodes is different. In order to constrain the energy penalty brought by resource scaling [32], EN-N performs a resource scaling operation every 30 seconds [36], [37]. As a result, this method can save more energy than the baseline method. Considering the QoS in each time span of 30 seconds, EN-N uses the mean waiting time to determine whether the SLA is satisfied. Specifically, every 30 seconds, EN-N calculates the mean waiting time of the requests which are accomplished during the time span. If the mean waiting time is higher than $M_w(p)$ or lower than $M_w(p) \times 20\%$, this method adjusts the number of working nodes.
- **N-N**^{2}: In this method [19], [20], each working node maintains two-level N queues. Generally, when the waiting time of accumulated requests satisfy the SLA, only the high level queue is used. To maintain a good QoS, when the SLA is not satisfied, the upper limit of the high

[3]Due to the reason that the average server utilization in most data centers is 20% [30]

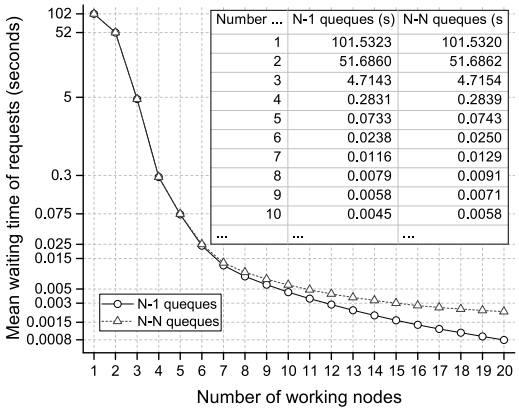
level queue length (ul) is limited to $\lfloor \frac{M_w(p)}{S} \rfloor$[4] [9]. If the length of the high level queue reaches the upper limit, the low level queue is used to receive the excess requests. Because there are two N queues in each working node, this method is referred as N-N^2. Compared with the baseline method, N-N^2 can mitigate the service congestion and maintain a good QoS for partial workloads. Therefore, N-N^2 can improve the system QoS under heavy workloads. Its timing and the way to adjust the number of working nodes are the same as those of the baseline method.

- **N1OD**: This method is the one we proposed in this paper. Due to the high service efficiency of N-1 queues to handle requests, N1OD can achieve a better efficiency than other methods. As a consequence, compared with other methods, N1OD tends to meet the same SLA with fewer working nodes. In addition, compared with other methods, N1OD can obtain a better QoS with the help of RDMM that continuously tracks the resource demand of workloads.

B. FUNDAMENTAL EVALUATION

As indicated in Section IV-B, the actual situations can be more complex than the theoretical scenario. When the system models' utilization ρ reaches 100% but still cannot satisfy the requirement, the mean waiting time W cannot be calculated out through the two equations anymore. The reason is that, the value of ρ should be less than 1 for theoretical analyses, but a value equals to or greater than 1 might obtained when calculating ρ in actual situations. So in this section, we first conduct a basic experiment to evaluate N-N queues and N-1 queues. In this experiment, the workloads are produced by arbitrarily replaying 15 minutes K5cloud weekday trace between 8:00 and 10:00. In addition, the evaluations are performed many times by increasing the number of working nodes from 1 to 20, and the values of W are recorded. We plot the curves that the values of W vary with the number of working nodes in Figure 4. As shown in Figure 4, the dots, which represent the mean waiting time of the requests of N-N and N-1 queues, are located very close when the number of working nodes ranges from 1 to 10. To show the actual values of these close dots clearly, we recorded their corresponding mean waiting time of requests in the table which is provided in Figure 4.

As shown in Figure 4, when the number of working nodes is less than 7, the mean waiting time with N-1 queues is very close but a little lower than that with N-N queues. The reason is that the difference between N-1 queues and N-N queues are quite small, when the number of working nodes is few. Especially when there is only 1 working node, for both the N-1 and N-N queues, the requests are served in a First Come First Service (FCFS) way. In this case, there is no difference to serve requests for both queues, except that waiting requests may accumulated in different queues. In addition, the high values of the mean waiting time imply that, the workloads are heavy for such numbers of working nodes. Accordingly, the working nodes are already tired of handling the workloads, thus the improvement can be achieved by N-1 queues is greatly limited. However, the difference is remarkable when the number of working nodes is more than 10. In this case, as the number of working nodes grows, the mean waiting time of N-1 queues is increasingly less than that of the N-N queues, and the reduction can be more than 50%. In conclusion, the service efficiency of N-1 queues is higher than that of the N-N queues, when there are multiple working nodes.



Number ...	N-1 queues (s)	N-N queues (s
1	101.5323	101.5320
2	51.6860	51.6862
3	4.7143	4.7154
4	0.2831	0.2839
5	0.0733	0.0743
6	0.0238	0.0250
7	0.0116	0.0129
8	0.0079	0.0091
9	0.0058	0.0071
10	0.0045	0.0058
...	...	...

FIGURE 4. The results of practical tests.

C. COMPREHENSIVE EVALUATION

Further, to attest the effectiveness of our method, we make a comprehensive evaluation with extensive experiments. To conduct experiments efficiently, the comparison methods along with our method, N1OD, are concurrently evaluated, using an arbitrarily sliced one-hour trace segment. Ten seconds are used as the time length of a period. In addition, each experiment is repeated 10 times, and each time another arbitrarily sliced one-hour trace segment is used. Finally, a comprehensive experimental result data are recorded. With full discussions on the experimental results, this subsection carefully evaluates our method along with all the comparison methods.

1) SERVICE EFFICIENCY

To evaluate the service efficiency of different methods, under the condition that the SLA is satisfied with the least working nodes, we record the throughput under different methods. With the least number of working nodes to satisfy the SLA (100%), the throughput (tp_m) is calculated as $\frac{1}{n_r} \times 100\%$. Accordingly, the fewer working nodes required to satisfy the SLA, the higher throughput tp_m a working node can achieve.

The experimental results are shown in Figure 5. As shown, the results of Baseline and ENN are always the same,

[4]According to the Theorem 1 and Corollary 2 in our study, with this upper limit ul, the response time of all the requests in the high level queue can theoretically be less than $M_w(p)$, besides, the actual value of p can be $\frac{ul/td+1/S}{\lambda/n}$.

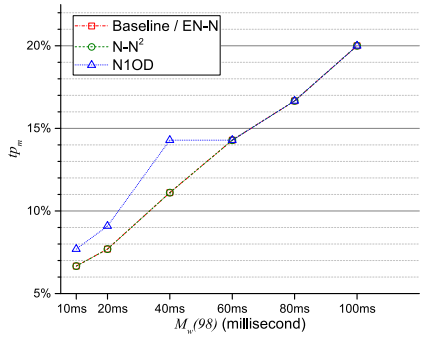


FIGURE 5. tp_m.

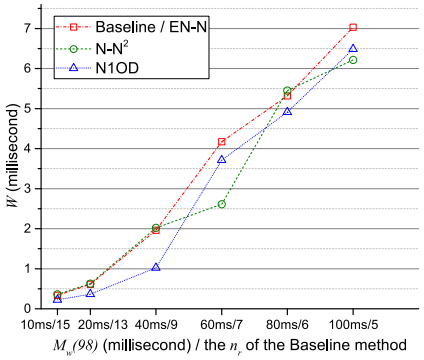


FIGURE 7. W.

therefore their results are presented use a same curve with the legend "Baseline / EN-N". The reason is that Baseline and ENN work with the same N-N queues and no resource scaling is performed. In addition, we can observe that the values of tp_m for N-N are the same as that of Baseline and EN-N. This is because although N-N^2 can mitigate the workload jam [19], [20] and maintain a good QoS for partial requests, 98% requests is an exacting proportion for maintaining their good QoS. As a result, the working nodes required by N-N^2 to satisfy the SLA are still the same as that required by Baseline and EN-N. By contrast, N1OD always achieves the highest values of tp_m. Especially when the SLA is strict, that the $M_W(98)$ is set less than 60 milliseconds, the values of tp_m of N1OD are far higher than that of the other three methods. This result is in accordance with the analysis made in Section IV-B, that the service efficiency of N-1 queues is higher than that of N-N queues.

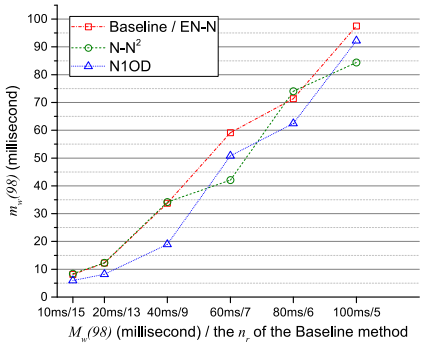


FIGURE 6. $m_W(p)$.

In addition, using the n_r of the Baseline method as the benchmark, the values of $m_W(98)$ and W under different methods are evaluated to further reveal the methods' differences in service efficiency. The experimental results are shown in Figure 6 and Figure 7. The two figures reveal the similar situation. Firstly, benefited from the high service efficiency of N-1 queues, N1OD always obtains lower $m_W(98)$ and W than Baseline and EN-N. Secondly, N-N^2 sometimes obtains lower $m_W(98)$ and W than other three methods. This is because service congestion can be mitigated by the low level queue used by N-N^2 in working nodes, and such mitigation makes some contributions to the improvement of QoS. While, such contributions are insufficient under a strict SLA, and N1OD obtains lower $m_W(98)$ and W than N-N^2 when $M_W(98)$ is set less than 60 milliseconds.

2) PERIODIC PERFORMANCE

To further explore the performance of each method in different periods, we perform experiments to record the $m_w(98)$ during each period. Because each experiment is performed with one-hour trace segment, there are 360 periods in each experiment. At the beginning of each experiment, all the 20 nodes are working. The former 90 periods are used as the initialization phase, thus enough time is provided to each method for initially adjusting the number of working nodes. We draw the curves that periodic performance varies with different values of $M_w(98)$ in Figure 8. In each subfigure, the horizontal x-axis shows the period number varies from 90 to 360. As shown, the values of $m_w(98)$ for EN-N are the highest. This is because EN-N scales the number of working nodes every 30 seconds (3 times the length of a period). Moreover, EN-N uses mean waiting time to detect whether current working nodes are adequate. The curves of Baseline, N-N^2 and N1OD are close, and are centered around the corresponding values of $M_w(98)$. This indicates that, compared with EN-N, the other methods are all more aware of the SLA. For each method, there are several time periods in which the SLA is not satisfied. The reason is that, a time delay is needed to scale resources [32], and the resource demand of requests sometimes cannot be timely met. The situation of EN-N is the worst, for mean waiting time is used by EN-N to reflect QoS, and the time span of re-scaling resources is long.

Due to the low capability provision, EN-N consumes the lowest energy than other methods. We plot Figure 9 to show the energy consumption of all the methods during each period. As shown in Figure 9, the total energy consumption of EN-N is much lower than other methods. While, the $m_w(98)$ of EN-N is unsatisfactory as indicated previously. The total energy consumption of N1OD is lower than that of Baseline and N-N^2, especially when $M_W(98)$ is small. When $M_W(98) = 10$ (Figure 9a), $M_W(98) = 20$

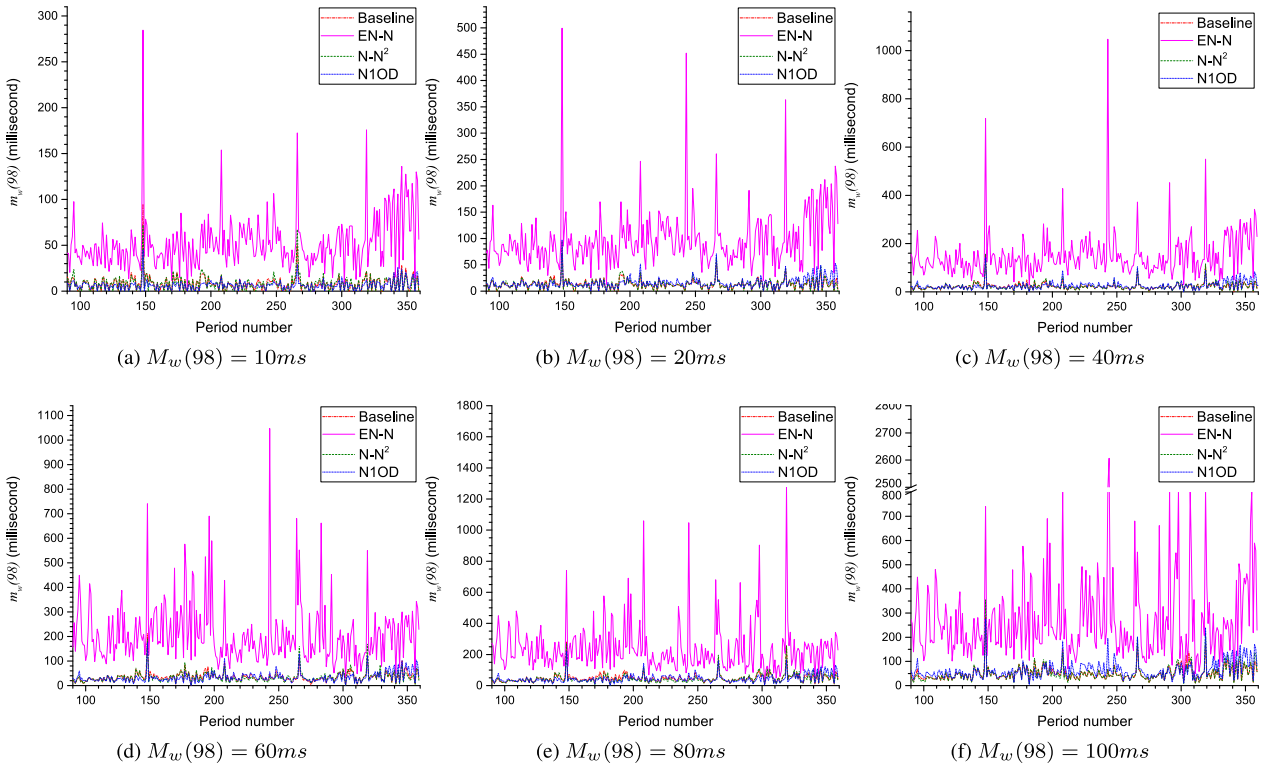


FIGURE 8. Periodic performance for $m_W(98)$ under different values of $M_W(98)$.

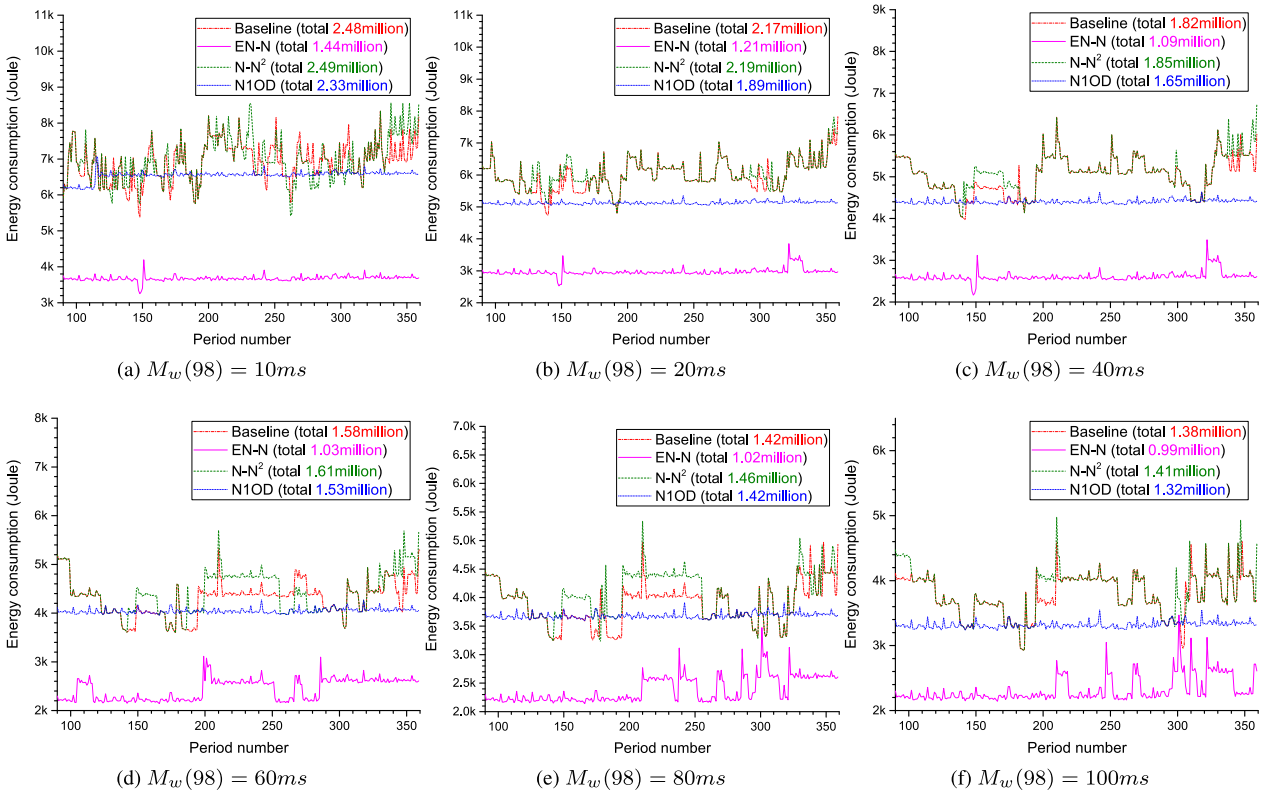


FIGURE 9. Periodic and overall system energy consumption under different values of $M_W(98)$.

(Figure 9b) and $M_W(98) = 40$ (Figure 9c), the total energy consumption of Baseline is 106.44%, 114.81% and 110.3% of that of N1OD respectively. In addition, the total energy consumption of N-N^2 is 106.87%, 115.87% and 112.12% of that of N1OD respectively. The reason is that, the service efficiency of N1OD is higher than that of Baseline and N-N^2,

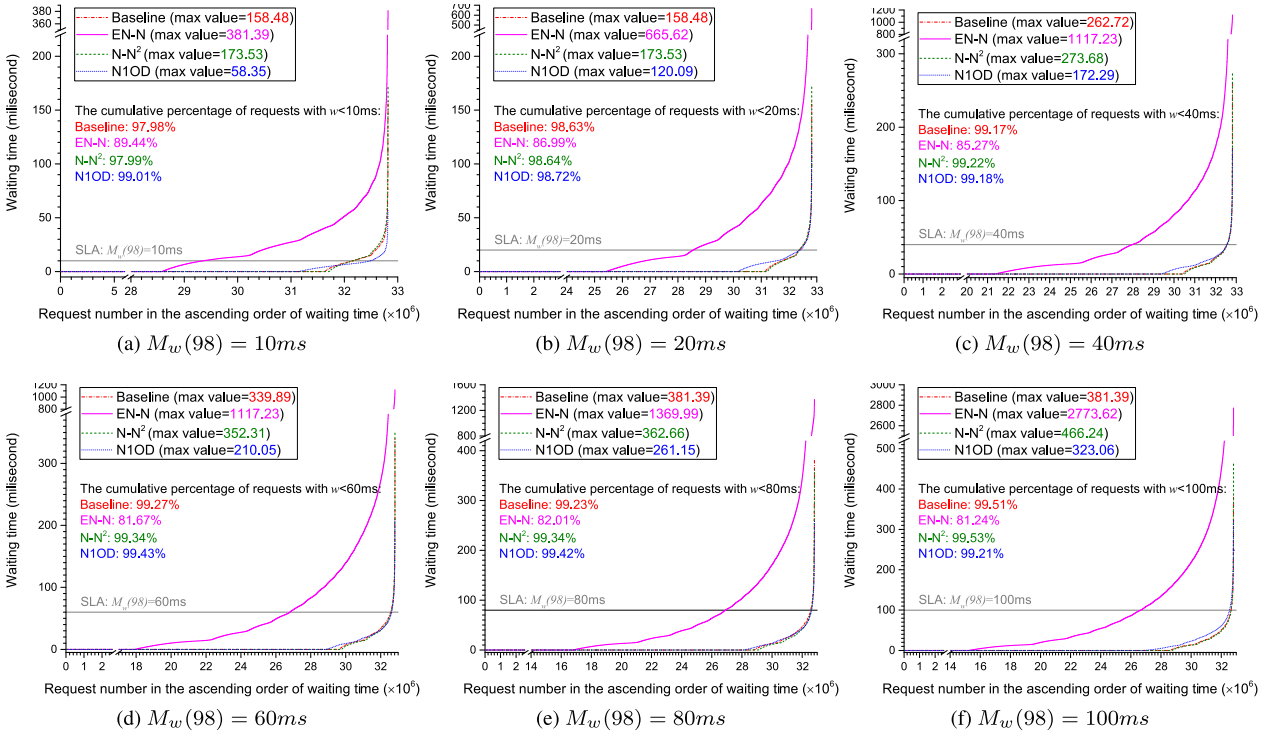


FIGURE 10. The waiting time of each request under different values of $M_W(98)$.

especially when $M_W(98)$ is small, i.e., under a strict SLA. Furthermore, attribute to the using of RDMM, advisable number of working nodes are provided to satisfy the SLA. As a result, fewer working nodes are required by N1OD to satisfy a same SLA. Accordingly, the quantity of working nodes need to be readjusted is small. That's the reason that the energy consumption during the periods is steadier for N1OD than other methods, and this is also the reason for the low energy consumption of N1OD. Moreover, the total energy consumption of N-N^2 is slightly higher than that of Baseline. This is because two-level N queues are contained in each working node for the N-N^2 methods, and the timing for a working node entering the low-power state is deferred by the unaccomplished requests in the low level N queue.

3) OVERALL PERFORMANCE

To evaluate the system QoS under different methods, the waiting time w of all accomplished requests is evaluated and recorded. We sort the requests in the ascending order of w and the results are shown in Figure 10. The figure shows the maximum values of w and the cumulative percentage of requests whose w is lower than the corresponding value of $M_w(98)$. Here, the value set for $M_w(98)$ is used as the SLA, e.g., $M_w(98) = 10$ means the SLA is set as that there should be more than 98% requests whose w is lower than 10 milliseconds. As shown in each subfigure, the values of w for EN-N grow faster than that for other methods. As a result, the cumulative percentage for EN-N in each subfigure is lower than that for other methods, and is lower than 98%. In other words, the system QoS under EN-N cannot meet the SLA. While, except in the case that $M_w(98) = 10ms$, the SLA can be satisfied by Baseline and N-N^2. In contrast, the SLA can always be satisfied by N1OD in all the cases, that the cumulative percentages for N1OD are always higher than 98%.

More concretely, the EN-N provides the fewest working nodes than other methods, thus it saves the most energy cost while obtains the worst QoS. The energy consumption and QoS of Baseline and N-N^2 are similar. The two-level N queues in N-N^2 can improve the service efficiency on busty workloads [19], [20], thus the cumulative percentages of N-N^2 are slightly higher than that of Baseline. However, since the two-level N queues delay the scheduling of some requests, the maximum values of w for N-N^2 are higher than that of Baseline. Due to the high service efficiency of N-1 queues, the maximum values of w for N1OD are always the lowest, and the cumulative percentages of N1OD are nearly always the highest.

VII. CONCLUSION

Considering the energy efficiency of blockchain computing systems, elastic clusters can be adopted as the infrastructures to provide scalable computing. In this paper, we reveal that the service efficiency of traditional N-N service queues is lower than that of N-1 service queues. Therefore, if traditional N-N service queues in an elastic cluster are transformed to N-1 service queues, fewer working nodes will be needed to meet the same SLA. Thus, the energy consumption of an

elastic cluster can be reduced, while still maintaining a similar service level. Therefore, to implement this transformation and further improve the energy efficiency of elastic clusters, we propose the N1OD method. In addition, N1OD realizes the RDMM module which can dynamically readjusts the number of working nodes to match the workload demand. To demonstrate the effectiveness of our method, we conduct extensive experiments under different SLAs with different values of $M_w(98)$, on a simulator which simulates a traditional elastic cluster. Experimental results indicate that, our method not only satisfies the overall SLAs, but also achieves a low energy consumption.

REFERENCES

[1] T. Aste, P. Tasca, and T. Di Matteo, "Blockchain technologies: The foreseeable impact on society and industry," *Computer*, vol. 50, no. 9, pp. 18–28, 2017.

[2] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Gener. Comput. Syst.*, vol. 105, pp. 475–491, Apr. 2020.

[3] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8076–8094, Oct. 2019.

[4] W. Liang, M. Tang, J. Long, X. Peng, J. Xu, and K.-C. Li, "A secure FaBric blockchain-based data transmission technique for industrial Internet-of-Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3582–3592, Jun. 2019.

[5] S. Xie, Z. Zheng, W. Chen, J. Wu, H.-N. Dai, and M. Imran, "Blockchain for cloud exchange: A survey," *Comput. Electr. Eng.*, vol. 81, Jan. 2020, Art. no. 106526.

[6] Z. Li, Z. Yang, and S. Xie, "Computing resource trading for Edge-Cloud-Assisted Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3661–3669, Jun. 2019.

[7] A. Zhou, Q. Sun, and J. Li, "Bcedge: Blockchain-based resource management in d2d-assisted mobile edge computing," *Softw., Pract. Exper.*, Oct. 2019, doi: 10.1002/spe.2758.

[8] L. Yang, Y. Deng, L. T. Yang, and R. Lin, "Reducing the cooling power of data centers by intelligently assigning tasks," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1667–1678, Jun. 2018.

[9] C. Hu, Y. Deng, G. Min, P. Huang, and X. Qin, "QoS promotion in energy-efficient datacenters through peak load scheduling," *IEEE Trans. Cloud Comput.*, early access, Dec. 12, 2018, doi: 10.1109/TCC.2018.2886187.

[10] Y. Deng, Y. Hu, X. Meng, Y. Zhu, Z. Zhang, and J. Han, "Predictively booting nodes to minimize performance degradation of a power-aware Web cluster," *Cluster Comput.*, vol. 17, no. 4, pp. 1309–1322, Dec. 2014.

[11] I. Anagnostopoulos, S. Zeadally, and E. Exposito, "Handling big data: Research challenges and future directions," *J. Supercomput.*, vol. 72, no. 4, pp. 1494–1516, Apr. 2016.

[12] A. Detti, L. Bracciale, P. Loreti, G. Rossi, and N. B. Melazzi, "A cluster-based scalable router for information centric networks," *Comput. Netw.*, vol. 142, pp. 24–32, Sep. 2018.

[13] C. Hu and M. Tang, "Reduce the energy cost of elastic clusters by queueing workloads with N-1 queues," in *Proc. Int. Conf. Blockchain Trustworthy Syst. BlockSys*, Z. Zheng, H.-N. Dai, M. Tang, and X. Chen, eds. Singapore: Springer, 2020, pp. 275–287.

[14] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu, S. U. Khan, and A. Zomaya, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, no. 7, pp. 751–774, Jul. 2016.

[15] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8050–8062, Aug. 2019.

[16] A. Zhou, S. Wang, X. Ma, and S. S. Yau, "Towards service composition aware virtual machine migration approach in the cloud," *IEEE Trans. Services Comput.*, early access, Dec. 24, 2020, doi: 10.1109/TSC.2019.2962128.

[17] F. Xu, F. Liu, and H. Jin, "Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2470–2483, Aug. 2016.

[18] E. Smart, D. Brown, K. Toumi Borges, and N. Granger-Brown, "Reducing energy usage in drive storage clusters through intelligent allocation of incoming commands," *Appl. Soft Comput.*, vol. 52, pp. 673–686, Mar. 2017.

[19] L. Lu, P. Varman, and K. Doshi, "Graduated QoS by decomposing bursts: Don't let the tail wag your server," in *Proc. 29th IEEE Int. Conf. Distrib. Comput. Syst.*, Washington, DC, USA, Jun. 2009, pp. 12–21.

[20] L. Lu, P. J. Varman, and K. Doshi, "Decomposing workload bursts for efficient storage resource management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 860–873, May 2011.

[21] F. Mardukhi, N. NematBakhsh, K. Zamanifar, and A. Barati, "QoS decomposition for service composition using genetic algorithm," *Appl. Soft Comput.*, vol. 13, no. 7, pp. 3409–3421, Jul. 2013.

[22] Y. Zhang, Q. Wei, C. Chen, M. Xue, X. Yuan, and C. Wang, "Dynamic scheduling with service curve for QoS guarantee of large-scale cloud storage," *IEEE Trans. Comput.*, vol. 67, no. 4, pp. 457–468, Apr. 2018.

[23] W. Stallings, *Operating Systems: Internals and Design Principles*, 9th ed. Upper Saddle River, NJ, USA: Pearson, 2017.

[24] C. Hu, Y. Deng, and L. T. Yang, "On-demand capacity provisioning in storage clusters through workload pattern modeling," *IEEE Access*, vol. 5, pp. 24830–24841, 2017.

[25] D. G. Kendall, "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain," *Ann. Math. Statist.*, vol. 24, no. 3, pp. 338–354, Sep. 1953.

[26] L. Schrage, "Queueing systems, Volume I: Theory," *Proc. IEEE*, vol. 65, no. 6, pp. 990–991, Jun. 1977.

[27] Wikipedia. *Erlang C Formula*. Accessed: Oct. 25, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Erlang_(unit)#Erlang_C_formula

[28] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. Katz, "NapSAC: Design and implementation of a power-proportional Web cluster," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 102–108, Jan. 2011.

[29] A. Biondi, M. D. Natale, and G. Buttazzo, "Response-time analysis of engine control applications under fixed-priority scheduling," *IEEE Trans. Comput.*, vol. 67, no. 5, pp. 687–703, May 2018.

[30] C. Hu and Y. Deng, "Fast resource scaling in elastic clusters with an agile method for demand estimation," *Sustain. Comput., Informat. Syst.*, vol. 19, pp. 165–173, Sep. 2018.

[31] L. Zhang, Y. Deng, W. Zhu, J. Zhou, and F. Wang, "Skewly replicating hot data to construct a power-efficient storage cluster," *J. Netw. Comput. Appl.*, vol. 50, pp. 168–179, Apr. 2015.

[32] C. Hu and Y. Deng, "Aggregating correlated cold data to minimize the performance degradation and power consumption of cold storage nodes," *J. Supercomput.*, vol. 75, no. 2, pp. 662–687, Feb. 2019.

[33] SNIA. *SNIA IOTTA Repository*. Accessed: Oct. 25, 2019. [Online]. Available: http://iotta.snia.org/tracetypes/4

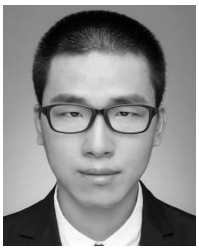
[34] K. OE, K. Ogihara, and T. Honda, "Analysis of commercial cloud workload and study on how to apply cache methods," *IEICE Tech. Rep.*, vol. 118, no. 165, pp. 7–12, 2018. [Online]. Available: https://www.ieice.org/ken/paper/20180730o1Fe/eng/

[35] F. Messaoudi, A. Ksentini, G. Simon, and P. Bertin, "Performance analysis of game engines on mobile and fixed devices," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 13, no. 4, pp. 1–28, Oct. 2017.

[36] M. Iritani and H. Yokota, "Effects on performance and energy reduction by file relocation based on file-access correlations," in *Proc. Joint EDBT/ICDT Workshops EDBT-ICDT*, 2012, pp. 79–86.

[37] J. Entrialgo, R. Medrano, D. F. García, and J. García, "Autonomic power management with self-healing in server clusters under QoS constraints," *Computing*, vol. 98, no. 9, pp. 871–894, Sep. 2016.

CHENG HU received the B.E. degree in software engineering from the School of Software, Nanchang University, and the Ph.D. degree in computer application technology from the Computer Science Department, Jinan University. He is currently a Teacher with the School of Information Science and Technology, Guangdong University of Foreign Studies. His current research interests include parallel and distributed computing, data center architecture, green computing, and cloud storage.



HUAN LUO received the B.Sc. degree in software engineering from Nanchang University, Nanchang, China, in 2009, and the Ph.D. degree in computer science from Xiamen University, Xiamen, China, in 2017. He is currently a Faculty Member with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. His research interests include point clouds processing, computer vision, and machine learning.

MINGDONG TANG received the B.S. degree in electrical engineering from Tianjin University, Tianjin, China, in 2000, the M.S. degree in control engineering from Shanghai University, Shanghai, China, in 2003, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2010. He is currently a Professor with the School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou, China. He is also with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China. His research interests include service-oriented computing, software engineering, and data mining. In addition, he is a member of the China Computer Federation and the ACM.