IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON INTERNET-OF-THINGS (IOT) BIG DATA TRUST MANAGEMENT

Received September 20, 2017, accepted October 21, 2017, date of publication October 31, 2017,
date of current version November 28, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2767703

On-Demand Capacity Provisioning in Storage Clusters Through Workload Pattern Modeling

CHENG HU[1], YUHUI DENG[1,2], AND LAURENCE T. YANG[3]

[1]Department of Computer Science, Jinan University, Guangzhou 510632, China
[2]State Key Laboratory of Computer Architecture, Institute of Computing, Chinese Academy of Sciences, Beijing 100190, China
[3]Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada

Corresponding author: Yuhui Deng (tyhdeng@jnu.edu.cn)

This work was supported in part by the NSFC under Grant 61572232 and Grant 61272073, in part by the Science and Technology Planning Project of Guangzhou under Grant 201604016100, in part by the Science and Technology Planning Project of Nansha under Grant 2016CX007, and in part by the Open Research Fund of Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences under Grant CARCH201401.

ABSTRACT Internet of Things (IoT), which is the inter-networking of a wide variety of physical devices, is widely used in our daily life. The exponential increase in the number of diverse devices has resulted in a significant increase in the volume, variety, velocity, and veracity of data (i.e., big data). These data present a large requirement on modern storage systems both for capacity and scale, and energy cost has become a critical problem. For storage clusters, much research effort has been invested in alleviating this problem by providing suitable resource capacity (i.e., on-demand providing). However, it is challenging to match the offered resource capacity with the real system workloads, thus resulting in a violation of service level agreement. By considering a storage cluster as a queueing system, this paper proposes a QoS-oriented capacity provisioning mechanism. Based on workload features, the mechanism models the pattern of current workloads as a suitable queueing model. In accordance with the model, our mechanism can well forecast the actual resource capacity demand without violating the service level agreement, and then offer the required resource capacity in terms of the real workloads. Experimental results demonstrate that the proposed mechanism significantly reduces the energy consumption of a typical storage cluster, while meeting the QoS requirements. It also significantly outperforms two classic and two state-of-the-art capacity provisioning mechanisms.

INDEX TERMS IoT, energy efficient, energy saving, storage cluster, capacity demand estimation, capacity provisioning, QoS, SLA.

I. INTRODUCTION

With the popularity of Information Technologies (ITs) in all walks of life, the next generation of computing and networking will involve a wide variety of devices. Anagnostopoulos *et al.* [1] point out that with the rapid advances in ITs such as global interconnection of heterogeneous information systems, tremendous structured and unstructured data have been generated from a wide range and multiple data sources. The global interconnection promotes the popularization of Internet of Things (IoT), and these tremendous data present a large requirement on modern storage systems both for capacity and scale. Thus, a rapid increase in the power consumption of modern storage systems is incurred. The energy consumption have become a critical problem for the operation of storage systems. For example, in 2013, an estimated up to 91 billion kilowatt-hours of electricity for a total cost of $12 billion are consumed by the data centers and computing clusters in US, and the electricity consumption is projected to increase to roughly 140 billion kilowatt-hours annually by 2020 [2], [3]. To this end, many researches have been proposed to balance the performance and power overhead in storage clusters. Among these researches, cutting down the power consumption of IT devices by converting the ones which are not actively working into a low power state is an effective method [4]. The reason behind this is that the available capacity of a system is normally much larger than the average demand of workloads, in order to handle the bursty workloads and achieve a good Quality of Service (QoS). As a result, significant energy is consumed by many servers which work at low utilization [5].

To save the energy cost of the superfluous resources, lots of research efforts have been invested with the idea of providing

resource capacity on demand. Most studies try to gear system capacity by considering one or a few isolated workload features [6], such as arrival rate of requests or system utilization. This is because the required capacity for real workloads normally increase as the average arrival rate of requests grows. In addition, system utilization partly reflects whether the offered capacity match real workloads.

However, the required capacity for real workloads is depend on various workload features such as coefficient of variation (CoV) for service time, service rate, and inter-arrival time of requests [7]. Only considering a few features or considering inappropriate features makes the capacity provision less valid for real workloads. Therefore, Gandhi *et al.* [6] present a capacity provisioning policy which considers both the arrival rate and the service time of workloads. This paper demonstrates that various workload features should be simultaneously considered when offering resource capacity for real system workloads. For example, even the average arrival rate of requests is maintained unchanged, due to the non-uniform inter-arrival time, the required capacity of a system for real workloads vary a lot as time goes by.

According to the above discussion, this paper proposes a capacity provisioning mechanism which simultaneously considers multiple workload features rather than one or a few isolated workload features, when offering the required resources for real workloads. This mechanism leverages multiple queueing models to model different workload patterns. Furthermore, based on the multiple workload features, the mechanism selects a suitable queueing model to represent the pattern of current workloads. Experimental results demonstrate that the proposed mechanism significantly reduces the energy consumption of a typical storage cluster, while meeting the QoS requirement. The main contributions of this paper are summarized as follows:

1) The correlations between workload features and system performance are analyzed to reveal that it's inappropriate to evaluate system performance by only considering one or a few isolated workload features. Multiple workload features should be considered simultaneously to provide required resource capacity for real workloads.
2) Based on considering multiple workload features with the help of queueing theory, multiple queueing models are used to model different workload patterns. For the energy efficiency of storage clusters, we present an Energy-efficient and QoS-oriented capacity Periodical provisioning mechanism (EQP), which can select a suitable queueing model to represent the pattern of current workloads. Thus, by analyzing the model under a specific mean waiting time SLA, EQP can obtain a good forecast for capacity demand and then provide a suitable capacity for real workloads.
3) Extensive experiments over real-world traces are performed to evaluate our mechanism against other two classic and two state-of-the-art mechanisms.

The remainder of the paper is organized as follows. Related work is introduced in Section II. Section III presents a motivational observation and the theoretical support. In Section IV, the detail implementation of EQP mechanism is described. Section V introduces the experiment environment. The experimental results and corresponding analysis are presented in Section VI. Finally, conclusions and the possible future work of this paper are provided in Section VII.

II. RELATED WORK

Clusters, which consist of loosely coupled nodes (or called servers) connected via fast networks [8], are often deployed in data centers to achieve high throughput and reliability. Dramatically increased energy cost for clusters have steered energy saving by using dynamic node scaling (or called dynamic capacity provisioning). According to the method used in providing capacity for real workloads, related researches can be classified into two categories. One is capacity provisioning with reactive methods, and the other one is capacity provisioning with predictive methods.

For reactive methods, some indicators are used to indicate whether a node is required for service. According to this indication, a reactive method gears the number of nodes to match real workloads. Specifically, if nodes are over-provisioned, it turns some of them to a low power state for energy saving; if the provided nodes are insufficient, it resumes some other nodes form low power state for service. Chase *et al.* [9] use the utilization, request queue length and request throughput for each service as the indicators. They estimate resource demands based on continuous observations of these indicators. If these indictors indicate that the capacity of a cluster is over-provisioned, they concentrate workloads on a subset servers of the cluster, and reduce the energy consumption of the cluster by turning idle servers off. Meisner *et al.* [10] design an energy-conservation approach—PowerNap, to build a multi-service system which is energy efficient. The PowerNap can rapidly transition a server between a high-performance active state and a minimal-power nap state. In PowerNap, the indicator is the arrival of a request. When a request arrives at a nap server, the server is woken up by it's Network Interface Card (NIC) to serve this request, and after the server finishes all work on it, it is transitioned to the nap state again. Huang *et al.* [11] present a power-efficient scheme for erasure-coded storage clusters. In the scheme, nodes are divided into active ones and low-power ones, and data redundancies are maintained in active nodes. To conserve energy, writes to the low-power nodes are deferred. The buffer size of a node is used as the indicator in this scheme. To indicate whether capacity is enough, one lower bound and one upper bound are set for the buffer size of nodes. When the amount of data produced by writes are beyond the upper bound, the capacity is considered to be insufficient. In this case, they activate all the low-power nodes, and data writes to these nodes are synchronized. However, the scheme may causes undesirable energy waste, for the reason that when only a few data writes exceed the upper bound, all the low-power nodes have to be activated. Entrialgo *et al.* [12] propose two algorithms to realize

self-scaling in server clusters with the purpose of energy conservation. The two algorithm can scale the server capacity dynamically, by leveraging two utilization thresholds which are an upper limit and a lower limit. When the utilization of a cluster is above the upper limit, one of the algorithms turns on an off node; while, when the utilization is lower than the lower limit, the other algorithm offloads the most efficient service node and then turns it off. But their algorithms have two drawbacks. One is that when workloads sharply go up by a factor of 2 or even more, turning on one node at a time is not enough; and the other one is that sometimes utilization is not a good indicator for system performance (Section III-A).

Predictive methods predict the situation of coming workloads, and then determine whether the provided capacity would match the coming workloads. If current provided capacity would be insufficient for the coming workloads, they provide more service nodes; while, if current capacity would be over-provisioned, they decrease the number of service nodes for energy saving. Krioukov *et al.* [13] achieve an energy-saving cluster through a power-aware cluster manager—NapSAC. In the energy-saving cluster, the request arrival rate is predicted with the assume that the proportions of static and dynamic requests in the workloads are more or less constant over time. NapSAC will put some servers to a low-power sleep state when the arrival rate of the incoming requests would decrease. Conversely, if the arrival rate would increase, NapSAC will awaken some sleep servers for service. Vakilinia *et al.* [14] propose a platform for workload prediction and Virtual Machine (VM) placement in cloud computing Data Centers (DCs). Although it's not physical servers in the platform, the platform still can be regarded as a cluster which consists of several VMs. In the platform, an estimation module is introduced to predict the arrival rate of the new incoming workloads. The predicted arrival rate is then used to indicate how to distribute the workloads of VMs among the servers with the purpose of power consumption minimization. Gandhi *et al.* [6] point out that it's not advisable to provide resource capacity for real workloads by only considering one workload feature as many researchers did. So AutoScale, which uses the product of request arrival rate and service time to indicate the density of workloads, is proposed by them to manage capacity for clusters. AutoScale always predicts that the coming workloads may increase and result in capacity insufficient. Therefore, if current capacity is over-provisioned, current workloads are concentrated onto a small number of service nodes by using an index-packing routing scheme, and the other (unneeded) service nodes are not turned off immediately, they sit in the idle state before a fixed ''time out'' duration. However, to meet a response time SLA that a 95 percentile goal of 400 ms, they define the max capacity of a server as 60 requests per second according to their measurement. But because the corresponding capacity of a server suitable for a SLA vary as the workloads features change, the provided capacity may not well match the real workloads.

There are also some other ways to save cluster energy cost, such as energy-aware workload scheduling [15], [16] and thermal management [17], [18]. But these ways are not concerned in this paper, so we do not discuss them here.

III. MOTIVATION AND THEORETICAL SUPPORT

A. MOTIVATIONAL OBSERVATION OF SYSTEM BEHAVIOUR

As described above, many researches provide resource capacity with a few workload features. However, the required capacity for real workloads depend on various workload features, and some features can not well reflect whether the offered capacity match real workloads. So the reality is that their methods make the capacity provision less valid for real workloads. We give out an observation in our study to demonstrate this reality, and then provide heuristic analyses on this observation.

In one of our simulation experiments, we simulate a typical storage cluster, and the service node in the cluster can only handle one request at a time. A network file system trace—deasna2, whose request arrival rate is on average 475 per second (Section V-B), is replayed on this cluster. In the experiment, the mean service time of requests is about 2 milliseconds. Thus, if the inter-arrival time of requests is uniform, a service node is capable of handling $1/0.002 = 500$ requests per second. So, if only considering the request arrival rate, providing one service node in the cluster is enough. However, to avoid the negative effects incurred by the fluctuation of request arrival rate and service time, two service nodes are provided in practice.

To analyze the system behaviour of the cluster, we define resident requests as the received requests who haven't been finished in the cluster. Therefor, the more the resident requests are, the poorer the QoS of the cluster is. So, in order to observe whether workloads is out of the capacity of the two service nodes, the number of resident requests is recorded as time goes by.

Figure 1 shows the number of resident requests in a time length of one hour. As shown, at the period from the 1300-th second to the 1500-th second, the number rises sharply. In order to explore why the spikes are generated

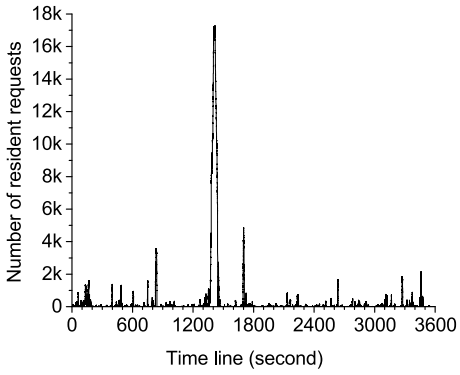

FIGURE 1. Number of resident requests.

in Figure 1, we divide a time period of one hour into equal durations with the same time length of 300 seconds. So the period from the 1300-th second to the 1500-th second is the fifth duration. First, to verify whether the spikes are caused by the fluctuation of request arrival rate, we give out the mean arrival rate of requests and the mean number of resident requests (calculated at the end of every duration) for each duration in Figure 2. As shown, the mean arrival rate does not fluctuate sharply, but the mean number of resident requests reaches up to 3429 at the fifth duration. Therefore, the spikes can not be considered as the result of the fluctuation of request arrival rate.

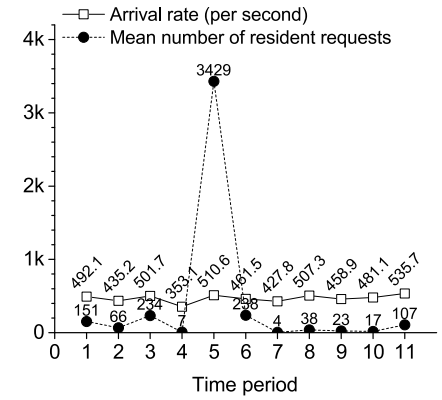

FIGURE 2. Resident VS arrival rate.

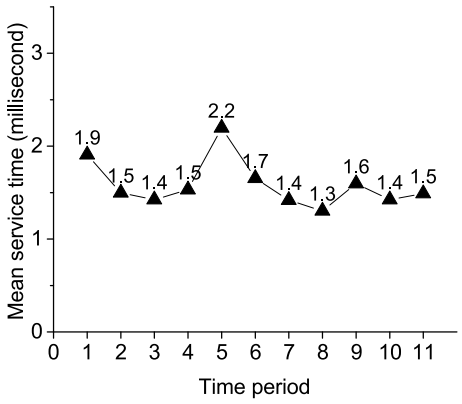

FIGURE 3. Mean service time.

Then, to verify whether the spikes are caused by the fluctuation of service time, we give out the mean service time of requests for each duration in Figure 3. As shown, the mean service time of requests also does not fluctuate sharply. Moreover, with a mean service time of 2 milliseconds, a node is capable of handling $1/0.002 = 500$ requests per second if the inter-arrival time is uniform. But as shown in Figure 2, although the mean arrival rate of requests is lower than 500 in most durations, there are many spikes which mean that the available capacity is not enough.

Further, we tested the the inter-arrival time of requests in the experiment, and find that the inter-arrival time in the fifth duration become very short. It is precisely because of the short inter-arrival time, the available resource capacity cannot catch up with the arrival requests, and this is the main reason causes the spikes in Figure 1. Therefore, various workload features should be simultaneously considered, so as to provide a suitable resource capacity for workloads. In addition, some isolated workload features such as arrival rate and utilization can not always well reflect the system performance.

B. QUEUING THEORY

A traditional storage cluster can be considered as a queuing system which queues requests according to their arrival time and serves them in a First Come First Serve (FCFS) way. For each node in the storage cluster, arrival requests are added to a First In First Out (FIFO) queue. Therefore, the system can be considered as multiple single-server queues. As Figure 4 shows, requests are dispatched to each single server, and this is often done by a manager node. In general, a manager node dispatches requests to service nodes in an uniform way. The uniform way means that if the arrival rate of requests to the system is λ, the arrival rate for each service node is λ/n in which n represents the total number of service nodes.

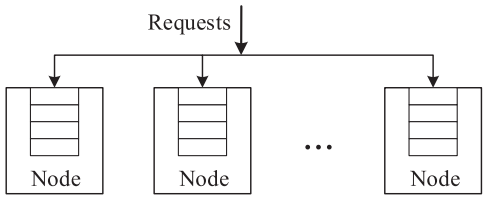

FIGURE 4. Multiple single-server queues.

Queuing theory provides us with the theoretical support for understanding the behaviors of queueing systems through workload features. We do not discuss the derivations and proofs of the conclusions related to queueing theory here, and the detail of them can be found in [19]. For each service node, it can be modeled as a single-server queue. According to the features of served workloads, there are mainly four queueing models which can be used to model a single-server queue. The queueing models can be described and classified with Kendall's notation [20] which uses three factors written as the form of $A/S/c$ to represent a queueing model. In that form, A refers to the distribution of inter-arrival time, S the distribution of service time and c the number of service nodes. Two common distributions are referred in this paper:

M: Poisson process (or called random) arrival process. / Exponential service time.

G: General (means arbitrary) distribution.

Generally, request arrivals are seen as a Poisson process and request service time is regarded as an exponential distribution [19], namely, $M/M/1$ queueing model. However, to be more accurate, general distribution is used when Poisson process cannot well match the real request arrivals. Consequently, four queueing models—$M/M/1$, $M/G/1$, $G/M/1$ and $G/G/1$ are involved in this paper. For a storage cluster,

the QoS of it can be regarded as the mean time that requests spend in the cluster (T_r). T_r contains two elements, one is mean waiting time (T_w) and the other is mean service time (T_s). T_s can be obtained by continuous observations, and for a data request it is a fixed value (*data size*/*read speed*). Thus, the optimal metric for the QoS of a cluster is T_w. With notations shown in Table 1, for each model, we summarize formulas for calculating T_w in Table 2. These formulas are obtained (or derived) from the studies of Stallings [7], Bhat [19], and Kingman [21].

TABLE 1. Notations for queueing models.

Notation	Description
λ	request arrival rate; mean number of arrivals per second
ρ	utilization (traffic intensity)
T_s	mean service time for each request; amount of time being served, not counting time waiting in the queue
T_w	mean waiting time for each request; amount of time before service
T_r	mean time a request spends in system (residence time)
μ	service rate (the reciprocal of T_s)
$A(t)$	the distribution function of inter-arrival time
$\phi(\theta)$	the LaplaceCStieltjes transform of $A(t)$
ζ	the least positive root of $z = \phi(\mu - \mu z)$
c_a	the coefficient of variation (CoV) for arrivals (that is the standard deviation of arrival times divided by the mean arrival time)
c_s	the CoV for service time
f	a scaling factor, $f = \frac{1}{2}(1 + c_s^2)$

TABLE 2. Formulas for calculating mean waiting time.

General	$\rho = \lambda T_s, \mu = \frac{1}{T_s}, T_r = T_w + T_s$		
M/G/1	$T_w = \frac{\rho T_s f}{(1-\rho)}$	**M/M/1**	$T_w = \frac{\lambda}{\mu(\mu-\lambda)}$
G/M/1	$T_w = \frac{\zeta}{\mu(\mu-\zeta)}$	**G/G/1**	$T_w \approx (\frac{\rho}{1-\rho})(\frac{c_a^2+c_s^2}{2})T_s$

IV. IMPLEMENTATION OF EQP

In this section, we first give out the overview of EQP, and point out the key issues in it. Then, the solutions for these issues are described respectively.

A. OVERVIEW OF EQP

EQP is aware of the status of each service node in a storage cluster. As shown in Figure 5, there are four steps in the implementation flow chart of EQP. In accordance with the orders labeled in the figure, the implementation of EQP is as follows:

① For each service node, the samples which are some observed values of past workloads (requests) are used to estimate the required features (parameters) for the coming workloads of it.

② According to the estimated features, a queueing model is selected to represent the pattern of the coming workloads in a service node.

③ By analyzing the selected queueing model for a service node, the theoretical value of the resource capacity which match the coming workloads in this node is calculated.

④ On the basis of all the theoretical values for all service nodes, EQP gears the resource capacity of the cluster to a proper value.

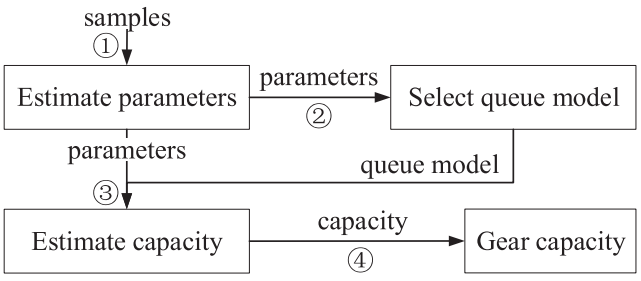

FIGURE 5. Flow chart of EQP.

For these steps, four key issues must be solved. The first one is how to take samples from workloads and estimate parameters from these samples. The second one is how to choose the proper queueing model for the coming workloads. The third one is how to estimate the required capacity. The last one is how to relieve the degrading effects on the system performance caused by bursty workloads. Next, with the QoS requirement of maintaining the mean waiting time of requests no larger than an expected value $E(T_w)$, the solutions are detailed in the following subsections.

B. SAMPLING AND PARAMETER ESTIMATION

We divide time into periods with equal length, and capacity adjustment only can be performed at the end of each period. That's why our mechanism is so called periodical provisioning. For energy saving, the time length of a period must be long enough to avoid frequently resume a low-power node. The reason is that the energy spend on resuming a low-power node is a big budget. If a low-power node is resumed, only when the energy it spends on the low-power state (contains the energy spend on resuming) is lower than that it spends on the active state with the same duration, the energy can be saved. Namely, $P_l \cdot t_p + P_r \cdot \Delta T < P_a(t_p + \Delta T)$ where P_a is the power of a node when it is active, t_p is the time length of a period, ΔT is the time needed to resume a low-power node, P_l is the power of a node when it is in the low-power state and P_r is the power of a node when it is being resumed. For specified hardware, the parameters are fixed values except t_p, therefore we can figure out the least value of t_p. Consulting the system parameters measured by Zhang *et al.* [22] and the researches [23], [24], 30 seconds are set as the length of a period. At the end of each period, the corresponding workload features such as arrival rate, arrival time and service time are taken as a sample. With some of the latest samples, the required parameters are estimated, thus the selection of samples is crucial to make a good estimation.

Considering changes in user request patterns [25], we use a Linear Growth and Return to One (LGRO) way to select samples. Initially, the sample of last one period is used to

estimate the parameters of coming requests. Next, every time if the gap between real T_w and $E(T_w)$ is in a tolerable range (that is within $[(1-\delta)T_w, (1+\delta)T_w]$ where δ is an adjustable factor), the number of samples used for estimation grows linearly (adds one); otherwise, the number returns to one. In addition, the weight of a sample is set to $\frac{i}{\sum_{k=1}^{n} k}$ where i is the order of the sample and n is the total number of samples used for estimation. The intuition behind this is that when workload pattern changed, the past samples which represent the former pattern should be discarded, and newer samples should have a higher weight.

To illustrate LGRO, we use an example to show the sample selection for estimating λ. In this example, for simplicity, we assume that the distribution of inter-arrival time is even. As shown in Figure 6, the horizontal axes labeled "Period" shows the start of each period with a numerical tick, and the request arrival rate for each period is calculated at the end of this period (that is the beginning of next period). The upper graph shows the samples selected for estimation at the beginning of each period. For example, the samples used for estimating the λ for the 3-th period is p_1 and p_2. Here, the label p_1 represents the sample taken from the 1-th period, and it's similar for other labels. Also, the labels can represent the corresponding period. The lower graph shows the real arrival rate for each period.

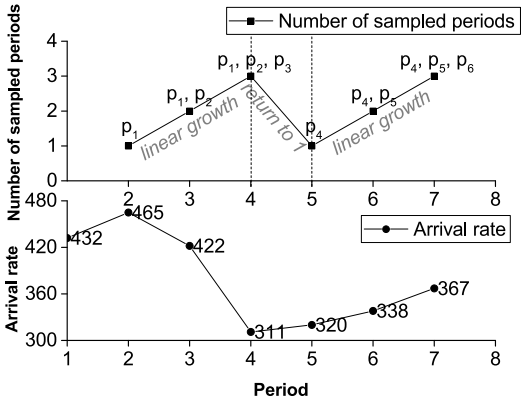

FIGURE 6. Illustration of LGRO.

Next, we explain the detail of this illustration. As time goes by:

1) At the beginning, the arrival rate of requests is 432 (per second) and this is used as the sample for period 2 (abbreviate p_2), so the estimated λ for p_2 is 432. The estimated λ for p_2 is close to the real λ for p_2. So, with this estimated value, resource capacity can be well provided for p_2 with the even distribution of inter-arrival time. As a result, the real T_w for p_2 will close to the expected value $E(T_w)$.
2) Now, the gap between the real T_w for p_2 and E(Tw) is in the tolerable range. Therefore, when it comes to the 3-th period, the number of samples used for estimation adds one. Thus, the samples p_1 and p_2 are used for the 3-th period, and the estimated λ for p_3 is $\frac{1}{1+2} \cdot 432 + \frac{2}{1+2} \cdot 465 = 454$.
3) For the same reason, the number of samples used for the 4-th period adds one, so the samples p_1, p_2 and p_3 are used for p_4. As a consequence, the estimated λ for p_4 is $\frac{1}{1+2+3} \cdot 432 + \frac{2}{1+2+3} \cdot 465 + \frac{3}{1+2+3} \cdot 422 = 438$.
4) However, the real arrival rate of p_4 is 311, so with the capacity suitable for the arrival rate of 438 per second, the real T_w will much lower than the $E(T_w)$. As a consequence of this, the number of samples used for estimation return to one. So, when it comes to the 5-th period, only the sample p_4 is used for p_5.
5) The arrival rates of the subsequent periods are close, accordingly, the number of samples used for estimation grow linearly (adds one every time).

Design an excellent parameter estimation method is a challenge. However, compared with the representative Moving Window Average (MWA) method [6], our method is more sensitive to the fluctuation of workloads. In addition, experimental results reveal that our method can achieve a good performance.

C. SELECTION OF QUEUEING MODEL

One way to select the best queueing model is to pick out the model whose mean prediction error is the lowest. However, due to the bursty nature of workloads, one queueing model can not well represent all the cases. Thus, performance degradation will be introduced now and then. What's more, unpredictable bursts are spread over system workloads, and the performance degradation is a big drag for DCs [26]. Therefore, the best way is to continually select a queueing model to represent current workloads, with the purpose of minimizing the prediction deviation.

Considering a storage cluster as multiple single-server queues, first we get the current workload features, then according to these features we select the best model. Specifically, in the process of sampling in each service node, we track the service time of each request. Then, by averaging the values, we get the mean value. Finally, the CoV of service time can be calculated. For $G/M/1$, due to the unknown distribution function of inter-arrival time, T_w can not be figured out. Therefore, we use the other three queueing models to estimate the performance of a service node. Referring to the online chapters–Chapter 20 of literature [7], the CoV of service time (c_s) is used to decide which model is adopted:

- if c_s is much less than 1, the arrivals of requests tend to be evenly spaced, and $G/G/1$ is adopted;
- if c_s is close to 1, the inter-arrival time tend to be exponential, and $M/M/1$ is adopted;
- if c_s is greater than 1, the arrivals of requests tend to congest, and $M/G/1$ is adopted.

D. CAPACITY ESTIMATION

In a storage cluster, now we can estimate the features of coming workloads. The main idea is to estimate the minimum number of available nodes, which can fulfill the SLA. In this paper, the SLA of a system is to guarantee the mean waiting time of requests no larger than a expected value—$E(T_w)$.

To find out how many available nodes can fulfill the QoS requirement (represented as a fixed mean waiting time SLA), we first denote the required number of available nodes by N. Thus, for each available node, the arrival rate of it is λ/N. Replacing T_w with $E(T_w)$ and λ with λ/N, then substituting the other parameters of coming workloads into the formulas in Table 2, N can be calculated out. Therefore, N is the number of available nodes with which the mean waiting time of requests is $E(T_w)$. If N is less than the number of current available nodes, the nodes which has the heaviest workloads are selected to be transitioned to the low-power standby state, and the number of available node in current new period is set to the rounded up integer value of N. This is because in this way, the remainder available nodes are more capable for the coming workloads, thus the cluster is more capable to tolerate workload fluctuations.

However if N is larger than the number of current available nodes, it means that the workloads increase. Thus the requests arrived in the prior period but not finished are left to current period. Therefore, to approximate the real situation, an additional value is added to the estimated arrival rate. The additional value (λ_a) can be calculated by dividing the number of requests beyond the capacity of the cluster by the time length of the new period. The former part consists two portions, one is the requests overloaded in the prior period, and the other portion is the requests beyond capacity caused by the time delay of resuming a standby node in the current period. Therefore, the total number of requests beyond capacity is $n(\lambda/n - \lambda/N) \times (t_p + t_r)$ where t_p is the time length of a period, t_r is the time needed to resume a standby node, and n is the number of current available nodes. Thus, dividing the total number by t_p we get $\lambda_a = n(\lambda/n - \lambda/N) \times (1 + t_r/t_p)$. Now, the approximate arrival rate ($A(\lambda)$) for current period is $\lambda+\lambda_a$. Similar to the calculation of N, the number of available nodes ($N^{'}$) with which the mean waiting time of requests is $E(T_w)$ can be calculate by replacing T_w with $E(T_w)$ and λ with $A(\lambda)/N^{'}$, then substituting the other parameters of coming workloads into the formulas in Table 2. Finally, the number of available nodes is geared to the rounded up integer value of $N^{'}$.

E. RELIEVE THE DEGRADING EFFECTS OF BURSTY WORKLOADS

Due to the bursty nature of workloads [27], a problem must be considered is how to relieve the negative effects of bursty workloads on system performance. As stated above, EQP only gears capacity at the end of each time period. When workloads burst in a time period, capacity shortage will last to the next period. In view of achieving a good QoS, EQP allows extra system capacity adjustment to fit the bursty workloads instantly when the QoS turns worse (in our experiments, when the mean waiting time exceed the configured SLA). In addition, when workloads decrease, to avoid a serious capacity shortage caused by workload rebounding, each time, at most half of current available nodes can be transitioned into the low-power state.

V. EXPERIMENT ENVIRONMENT

A. EVALUATION MODEL

Our mechanism is evaluated in a typical energy-saving storage cluster which is similar to the ones presented in numerous research studies [9]–[13]. The system model of this cluster is shown in Figre 7, where all homogeneous nodes are under the control of a manager. Requests are first received by the manager and then dispatched to the working node which owns the most remainder capacity. Moreover, the manager keep tracking the performance of all available nodes, and at the end of each period it gears the number of available nodes to maintain the provided resource capacity match the real workloads.

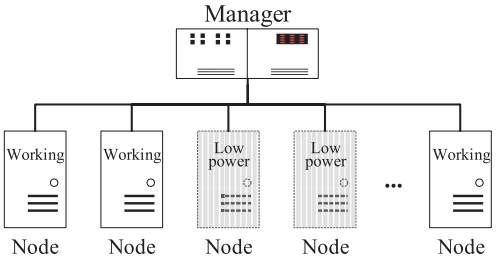

FIGURE 7. System model.

To save energy, some nodes are switched into a low-power state, and other ones keep working for service. The state transitions of cluster nodes are shown in Figure 8. When a working node is serving a request, it is in the active state; when a working node does not serve any request, it is in the idle state; the nodes which have been suspended into the standby state are the low-power nodes. In our experiments, the power of the three states are set to 60W, 40.2W and 4W respectively which are the values measured by Zhang *et al.* [22]. Also, according to their research, the energy cost of suspending a node and resuming a node are set to 4J and 519J, and the time cost of these two transitions are 1 second and 10 seconds. Besides, the cost for transitions between active and idle is ignorable.

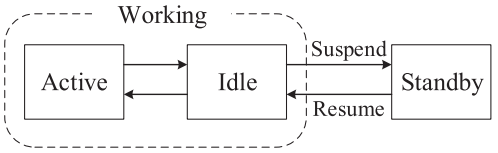

FIGURE 8. State transitions.

Remarkable, an available node is defined as the node which can receive and serve new requests. When the cluster decides to suspend a working node, the node will no more receive any new request. Before being suspended, the node has to wait until all the requests in it are finished. In this paper, a node in this waiting situation is called a leaving node. Although a leaving node is working, it is not an available node. So the resource capacity of the cluster is exactly the number of available nodes, and the goal of our mechanism is to provide suitable number of available nodes for real workloads.

When the cluster needs to add available nodes, the best choice is the leaving nodes which have the lightest workloads. This is because, these leaving nodes haven't been suspended into the standby state, and no extra overhead is needed to take them back for work. In the absence of definition of available node, the words ''working node'' appears in previous sections which mean the node can receive new requests should precisely be written as ''available node''.

B. EXPERIMENT SETUP

We evaluate EQP against five capacity provisioning mechanisms by simulation experiments which simulate the energy-saving storage cluster described above. Two metrics are used to compare the performance of all the mechanisms. One is the energy consumption of cluster and the other is the mean waiting time of requests. Specifically, a Reactive (Re) and a Predictive (Pr) mechanisms which are described in the study of Gandhi *et al.* [6] are classic mechanisms, EI [12] and AutoScale [6] (AS) are state-of-the-art mechanisms:

Re: The Reactive capacity provisioning mechanism (Re) reacts to the current request arrival rate, and attempt to keep exactly λ/λ_c servers available at time t, where λ_c is the max arrival rate of requests a server can handle under a specify SLA of mean waiting time. However, as we analyzed above, λ_c is not constant, thus it's hard to determine the most suitable capacity. In our implementation of Reactive, λ_c is set to the mean arrival rate a server can handle under the SLA. Every time, an arrival request is dispatched to the node whose resident requests are the fewest.

Pr: There are two representative predictive policies for Predictive capacity provisioning mechanism (Pr): one is Moving Window Average (MWA) and the other is Linear Regression (LR). Here, we use MWA because it has a better performance than LR [6]. The way Pr providing resource capacity is similar to Re, except that Pr uses a predicted value of arrival rate rather than current real λ. In our experiments, arrival rate is calculated each second in every period, and the window is set to 30 seconds which is the length of a period. WMA firstly predicts the request arrival rate at the later 40 second (10 seconds are needed to resume a node). Then it determines the number of available nodes needed to meet the SLA at the end of each period. The predicted value is the mean of former values, such as the predicted arrival rate at 31s is the mean value of former 30s, and the predicted arrival rate at 32s is the mean value of former 31s (including the predicted value for 31s), and so on.

EI: The Energy-efficient capacity Immediately provisioning mechanism (EI) gears resource capacity leveraging two utilization thresholds—U_{max} and U_{min}. When the utilization of cluster is under U_{min}, EI decreases an available node immediately; and when the utilization exceeds U_{max}, EI adds an available node immediately. U_{max} and U_{min} are set to 0.95 and 0.9 respectively according to the study [12]. Because some nodes may in the process of resuming and cannot serve immediately, the estimation of wether the provided capacity is suitable is made after these nodes finished that process. Every time, an arrival request is dispatched to the node whose utilization is the lowest.

AS: AutoScale capacity provisioning mechanism (AS) uses $\lambda_c \times T_s$ (denoted by C_{ref}) to represent the capacity of a node, rather than λ_c. To realize AS, we record both the service time and arrival rate for each available node when the SLA is satiated, and then get the mean values(namely λ_c and T_s respectively). When the provided capacity is not suitable for current real workloads, AS tries to scale the number of available nodes to $\lceil L_{sys}/C_{ref} \rceil$ where L_{sys} represents the current value of $\lambda \times T_s$. An available node is not suspended immediately in AS, and when it is idle, it still keeps available for 5 seconds which are half of the time needed to resume a node. Every time, an arrival request is dispatched to the node whose workloads are the lightest using an index-packing idea [6].

EQP: The Energy-efficient and QoS-oriented capacity Periodical provisioning mechanism (EQP) is the one we proposed, and it is described in Section IV. Compared with EI, it finds out a suitable capacity for real workloads by analyzing a corresponding queueing model. EQP directly gears the cluster to that capacity, rather than increase or decrease one available node every time as EI does. Every time, the manager dispatches an arrival request to the available node which has the most capacity remained, that is the one which has the minimum value of T_w.

The workloads for the cluster is produced from three real-world network file system traces: lair62b, home02 and deasna2. The details of these traces can be obtained from the web page: http://www.eecs.harvard.edu/sos/traces.html. The main characteristics of these traces are listed in Table 3. The weekday fragments of traces are arbitrarily selected for our experiments (only use daytime records) for the sake of generality, and the experiment results are the mean values for every test.

TABLE 3. Main characteristics of traces.

Trace name	Total size	Duration	Arrival rate (per second)
lair62b	11GB	936 hours	28
home02	48GB	2160 hours	398
deasna2	32GB	960 hours	475

The value of the δ can influence the reactivity of mechanisms. However, to simplify the experiment results, the value of δ (Section IV-B) is set to a fixed value—0.2. The QoS requirement for the cluster is set as an mean waiting time SLA, which requires the mean waiting time of requests is not larger than 0.03 seconds. Because of the bursty nature of workloads and the time delay introduced by resuming nodes,

in most cases the actual result of mean waiting time will larger than the expected value. Hence, to fulfill the SLA, the value of $E(T_w)$ is set to 0.02 seconds.

For each trace, we determine the number of nodes contained in the cluster. First, we test the mean waiting time for every trace with the number of available node start from 1. If the mean waiting time is lower than 0.03 seconds, the SLA is met. Then according to our test results, the average numbers of available nodes required to fulfill the SLA for lair62b, home02 and deasna2 are 7, 9 and 15 respectively. Finally, based on the truth pointed out by Entrialgo *et al.* [12] that the utilization of servers in data centers is typically under 50%, the cluster contains 14, 18 and 30 nodes respectively which are double of the required quantity. In addition, at the beginning, half nodes are available in the cluster. All the settings mentioned above for each trace are summarized in Table 4 where N represents the total number of nodes in the cluster and N_i represents the initial number of available nodes.

TABLE 4. System settings.

Trace	δ	SLA (QoS)	$E(T_w)$	N	N_i
lair62b	0.2	T_w<0.03s	0.02s	14	7
home02				18	9
deasna2				30	15

VI. EXPERIMENTAL RESULTS

In this section, we first present the performance of the cluster for each period (period performance), that is the mean waiting time for requests in each period and the energy consumption in each period. Both of them are calculated in the end of each period. Then, we give out the results of overall performance. The results contains two parts, one is the total SLA violation ratio, and the other is the total energy consumption. Finally, we present the results of capacity unsuitable degree for each mechanism, to show how suitable the provided capacity is for real workloads. All the above experiment results are produced by several experiments, which test arbitrarily selected segments from the traces (Section V-B).

A. PERIOD PERFORMANCE

At the end of each period, the mean waiting time for requests and the energy consumption of the cluster in current period are calculated. The results are shown respectively in Figure 9 and Figure 10. In each figure, the subfigures from left to right correspond to the results for lair62b, home02 and deasna2 respectively.

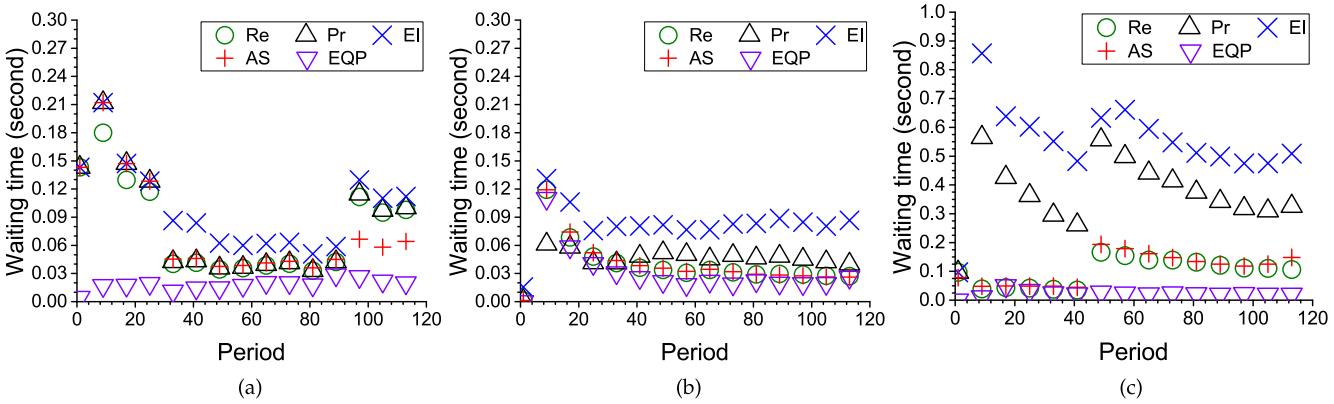

FIGURE 9. Mean waiting time for each period. (a) lair62b. (b) home02. (c) deasna2.

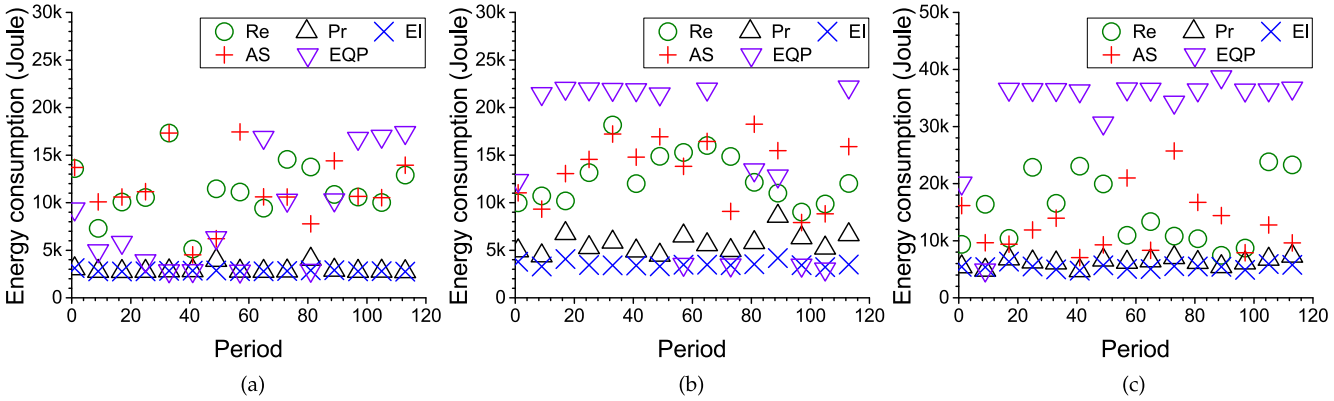

FIGURE 10. Energy consumption for each period. (a) lair62b. (b) home02. (c) deasna2.

As shown, the mean waiting time for Re, Pr, EI and AS always cannot fulfill the SLA (0.03 seconds). That's because the workload features they considered are not sufficient enough as we explained before. Among these four mechanisms, both Re and Pr only focuss on the changes of arrival rate in workloads, and EI just focusses on the changes of the utilization. However, as revealed in Section III-A, arrival rate or utilization sometimes can not well reflect the QoS of a cluster. To provide a good QoS, a cluster sometimes has to work with a large resource capacity, under a low mean arrival rate or a low utilization. Although both considering the arrival rate and service time, the parameter C_{ref} used in AS is still not a proper one to represent the capacity of a service node. As a result, the QoS of AS is also far from satisfactory, and is often close to Re. EQP, by contrast, can make a good estimation on resource capacity demond through the workloads features involved in queueing theory. Thus, it can provide a suitable capacity for real workloads on the fly, no matter how workloads change. Consequently, EQP can maintain the mean waiting time of requests under the SLA most of the times. Moreover, it can observed that the mean waiting time for each period is close to the SLA. This validates that EQP indeed can always provide a suitable capacity to fulfill the SLA, for the reason that the mean waiting time will be far below the SLA if capacity is over-provisioned and will violate the SLA if the provided capacity is insufficient.

Pr predicts the future arrival rate as the mean of historical values. In general, taking mean covers the burst nature of workloads. Thus, the capacity provided by Pr is relatively stable, and the energy consumption of the cluster with Pr change slightly during different periods. The energy consumption for EI also have the same situation as Pr, for EI maintains the utilization of the cluster within a fixed range and not care of the workload features. While, when it comes to the other three mechanisms, the energy consumption vary a lot during different periods. This reality reflects that workloads are fluctuant. Besides, as explained in the end of Section III-A, although sometimes the utilization or arrival rate is low, the required capacity is large to fulfill the SLA. Thus, as a result of providing a suitable resource capacity for real workloads, EQP consumes more energy than other mechanisms in many periods.

B. OVERALL PERFORMANCE

To show the QoS of the cluster with different mechanisms clearly, we first define a metric—SLA violation ratio. The SLA violation ratio is 0 if the mean waiting time is under the SLA, and the SLA violation ratio equivalents to the ratio of the exceeded value if the mean waiting time violate the SLA. That is, the value of SLA violation ratio is 0 when the SLA is not not violated, while, if the mean waiting time violate the SLA, the value is

$$(T_w - S(T_w))/S(T_w)$$

where T_w is the actual mean waiting time of requests and $S(T_w)$ is the maximum mean waiting time set by the SLA (0.03 seconds in our experiments). For example, if the real mean waiting time is 0.09 seconds which exceed the SLA by 0.06 seconds, thus the SLA violation ratio is 0.06/0.03 equals to 2. Obviously, the lower the SLA violation ratio of one mechanism achieves, the better the QoS of the mechanism is. The sum of SLA violation ratios for all periods in one hour is calculated to show the overall performance of the cluster. Accordingly, the total energy consumption of these periods are also calculated, and is normalized to the one-hour energy consumption of the cluster with all nodes serving (available).

As Figure 11 shows, EI has the highest sum of SLA violation ratios for each traces. This demonstrates that the utilization is not a good indicator for QoS, though EI maintains the utilization of the cluster within an appropriate range, it can not obtain a good QoS. The values for Pr and Re are lower than EI, and this reveals that providing capacity by considering the arrival rate can obtain a better performance than considering the utilization. Re and AS have similar performance, and the QoS of them outperform that of Pr. The reason behind this is that Re and AS are more aware of workload fluctuations, they immediately scale up the capacity when they detect that the workloads become heavier. EQP provides the best QoS among all mechanisms, and for lair62b it does not violate the SLA at all.

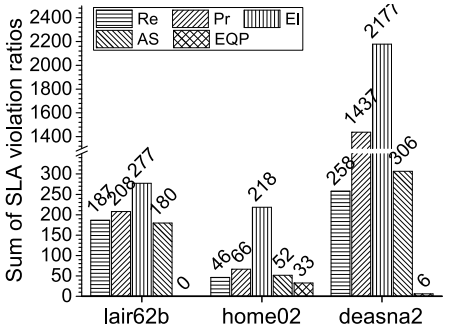

FIGURE 11. Sum of SLA violation ratios.

The total energy consumption for each mechanism is shown in Figure 12. As explained in above subsection, the capacity provided by Pr and EI is relatively stable,

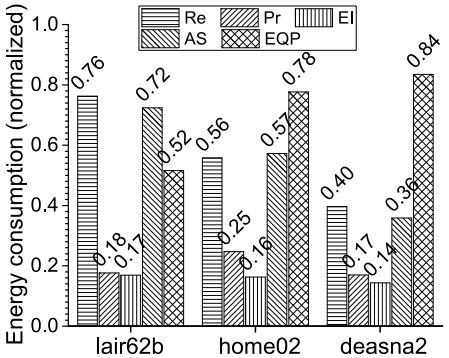

FIGURE 12. Total energy consumption.

accordingly the energy consumption of Pr and EI is maintained at a lower level. In contrast, the energy consumption for Re and AS is higher, for the reason that they scale up the system capacity when workloads just temporary increase thus more capacity is supplied. In the viewpoint of energy saving, Pr and EI are good choices, however, the QoS of them is poor. Compared with other mechanisms, although overall EQP consumes more energy, it provides a suitable resource capacity which can just fulfill the SLA almost all the time as stated in above subsection. The reason behind the low energy consumption of other mechanisms is that they often do not provide enough capacity for real workloads.

C. CAPACITY UNSUITABLE DEGREE

In this subsection, we separately describe the capacity unsuitable degree which indicates how suitable the provided capacity is for real workloads. The unsuitable degree is expressed as the sum of

$$|T_w - S(T_w)| / S(T_w)$$

for all periods. Obviously, the more suitable the provided capacity is, the smaller the value of the unsuitable degree is. Compared with the SLA violation ratio, it also care about whether the capacity is over-provisioned by calculating the absolute value. The most suitable case is that the actual mean waiting time equal to the value set by the SLA. In this case, $|T_w - S(T_w)| / S(T_w)$ equal 0. The unsuitable degrees for each mechanism are shown in Figure 13.

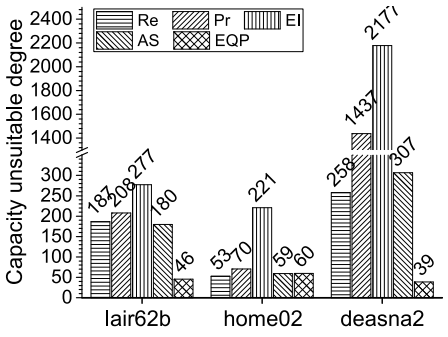

FIGURE 13. Capacity unsuitable degree.

As shown, the unsuitable degree of EQP is always very low for each trace, because the mean waiting time of EQP is always close to the SLA as Figure 9 shows. Although for home02, the unsuitable degree of EQP is a little bigger than Re and AS, Re and AS violate the SLA as Figure 9b shows. Besides, the capacity provisioned by EQP is always a suitable value under the SLA. Therefore, the experiment results of the unsuitable degrees for each mechanism also attest that the overall high energy cost of EQP is just the result of that EQP provides a suitable capacity for real workloads, whereas the low energy cost of other mechanisms is caused by insufficient capacity provision.

VII. CONCLUSION

The global interconnection of heterogeneous information systems promotes the popularization of IoT, and present a large requirement on modern storage systems both for capacity and scale. To alleviate the huge energy consumption of modern storage clusters, extensive researches cut the power of the components which are not actively working in clusters. However, the issue of matching the offered resource capacity with the real system workloads is a big challenge. Many researchers gear resource capacity base on some isolate workload features, such as the arrival rate of requests or the utilization of cluster. Although from the viewpoint of energy saving, they have a good performance, most of the time they do not provide a suitable capacity for real workloads. Thus, we propose an energy-efficient and QoS-oriented capacity provisioning mechanism, EQP, which uses a suitable queueing model to represent current workload pattern. Then, leveraging the model, EQP can obtain a good forecast for resource capacity demand under a SLA. Finally, with this forecast, EQP can provide a suitable capacity for real workloads. Experiment results show that EQP significantly outperforms the other mechanisms. Specifically, EQP on average saves more than 28.6% energy compared with the case that all nodes serving. Meanwhile, it almost does not violate and just fulfill the SLA which is set as the QoS requirement.

We present a rough way in Section IV-E to relieve the negative effects taken by bursty workloads. For future work, we plan to propose a more intelligent approach to tackle the negative effects, thus strictly fulfilling QoS requirements even the real workloads are complicated by numerous irregular fluctuations.

ACKNOWLEDGMENT

This paper was presented at the 15th IEEE International Symposium on Parallel and Distributed Processing with Applications (IEEE ISPA 2017) [28].

REFERENCES

[1] I. Anagnostopoulos, S. Zeadally, and E. Exposito, "Handling big data: Research challenges and future directions," *J. Supercomput.*, vol. 72, no. 4, pp. 1494–1516, 2016.
[2] R. Entezari-Maleki, L. Sousa, and A. Movaghar, "Performance and power modeling and evaluation of virtualized servers in IaaS clouds," *Inf. Sci.*, vols. 394–395, pp. 106–122, Jul. 2017.
[3] X. Zhan and S. Reda, "Power budgeting techniques for data centers," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2267–2278, Aug. 2015.
[4] H. Okamura, S. Miyata, and T. Dohi, "A Markov decision process approach to dynamic power management in a cluster system," *IEEE Access*, vol. 3, pp. 3039–3047, 2015.
[5] D. Wong and M. Annavaram, "KnightShift: Scaling the energy proportionality wall through server-level heterogeneity," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2012, pp. 119–130.
[6] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. A. Kozuch, "AutoScale: Dynamic, robust capacity management for multi-tier data centers," *ACM Trans. Comput. Syst.*, vol. 30, no. 4, pp. 14:1–14:26, Nov. 2012.
[7] W. Stallings, *Operating Systems: Internals and Design Principles*, 6th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2008.
[8] C. L. Seitz, "Recent advances in cluster networks," in *Proc. 42nd IEEE Symp. Found. Comput. Sci.*, Oct. 2001, p. 365.
[9] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," *ACM SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 103–116, Oct. 2001.

[10] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating server idle power," *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 1, pp. 205–216, Mar. 2009.
[11] J. Huang, F. Zhang, X. Qin, and C. Xie, "Exploiting redundancies and deferred writes to conserve energy in erasure-coded storage clusters," *ACM Trans. Storage*, vol. 9, no. 2, pp. 4:1–4:29, Jul. 2013.
[12] 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, 2016.
[13] 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.*, vol. 41, no. 1, pp. 102–108, 2011.
[14] S. Vakilinia, B. Heidarpour, and M. Cheriet, "Energy efficient resource allocation in cloud computing environments," *IEEE Access*, vol. 4, pp. 8544–8557, 2016.
[15] J.-J. Chen, M.-J. Kao, D. T. Lee, I. Rutter, and D. Wagner, "Online dynamic power management with hard real-time guarantees," *Theor. Comput. Sci.*, vol. 595, pp. 46–64, Aug. 2015.
[16] K. Li, "Scheduling parallel tasks with energy and time constraints on multiple manycore processors in a cloud computing environment," *Future Generat. Comput. Syst.*, to be published.
[17] E. Pakbaznia, M. Ghasemazar, and M. Pedram, "Temperature-aware dynamic resource provisioning in a power-optimized datacenter," in *Proc. Conf. Design, Autom. Test Eur. (DATE)*, Leuven, Belgium, 2010, pp. 124–129.
[18] A. Pahlavan, M. Momtazpour, and M. Goudarzi, "Power reduction in HPC data centers: A joint server placement and chassis consolidation approach," *J. Supercomput.*, vol. 70, no. 2, pp. 845–879, 2014.
[19] U. N. Bhat, *An Introduction to Queueing Theory: Modeling and Analysis in Applications*. Cambridge, MA, USA: Birkhäuser, 2015.
[20] 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, 1953.
[21] J. F. C. Kingman, "The single server queue in heavy traffic," *Math. Proc. Cambridge Philos. Soc.*, vol. 57, no. 4, pp. 902–904, 1961.
[22] 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.
[23] 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)*, New York, NY, USA, 2012, pp. 79–86.
[24] C.-A. Chen, M. Won, R. Stoleru, and G. G. Xie, "Energy-efficient fault-tolerant data storage and processing in mobile cloud," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 28–41, Jan./Mar. 2015.
[25] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *J. Netw. Syst. Manage.*, vol. 23, no. 3, pp. 567–619, 2015.
[26] L. Lu, P. Varman, and K. Doshi, "Graduated QoS by decomposing bursts: Don't let the tail wag your server," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Jun. 2009, pp. 12–21.
[27] Y. Wu, G. Min, K. Li, and B. Javadi, "Modeling and analysis of communication networks in multicluster systems under spatio-temporal bursty traffic," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 5, pp. 902–912, May 2012.
[28] C. Hu and Y. Deng, "QoS-oriented capacity provisioning in storage clusters by modeling workload patterns," in *Proc. 15th IEEE Int. Symp. Parallel Distrib. Process. Appl.*, 2017.

CHENG HU received the B.E. degree in software engineering from the School of Software, Nanchang University, in 2009. He is currently pursuing the Ph.D. degree with the Computer Science Department, Jinan University. His current research interests include parallel and distributed computing, data center architecture, green computing, and cloud storage.

YUHUI DENG received the Ph.D. degree in computer science from the Huazhong University of Science and Technology in 2004. He is a Professor with the Computer Science Department, Jinan University. Before joining Jinan University, he was with EMC Corporation as a Senior Research Scientist from 2008 to 2009. He was a Research Officer with Cranfield University, U.K., from 2005 to 2008. His research interests cover green computing, cloud computing, information storage, computer architecture, and performance evaluation.

LAURENCE T. YANG received the B.E. degree in computer science and technology from Tsinghua University, China, and the Ph.D. degree in computer science from the University of Victoria, Canada. He is a Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, China, as well as with the Department of Computer Science, St. Francis Xavier University, Canada. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, and big data. He has authored or co-authored over 200 papers in various refereed journals, many in IEEE/ACM Transactions/Journals. His research has been supported by the National Sciences and Engineering Research Council of Canada and the Canada Foundation for Innovation.