Power-effiicent Resource Allocation in MapReduce Clusters

Kaiqi Xiong
College of Computing and Information Science
Rochester Institute of Technology
Rochester, NY 14623 USA

Yuxiong He
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA

***Abstract*—MapReduce has recently evolved in data-intensive parallel computing. It is a programming model for processing large data sets. The implementation of MapReduce typically runs on a large scale of cluster computing systems consisting of thousands of commodity machines. Such cluster computing systems are called MapReduce clusters. The high power consumption of MapReduce clusters has become a major concern since hundreds of MapReduce programs are implemented and thousands of MapReduce jobs are executed in such clusters like Amazon's Elastic MapReduce Clusters every day. Power management becomes one of the most important problems in MapReduce clusters. Furthermore, the availability of MapReduce clusters plays an essential role in the delivery of quality of services (QoS) for customer services. In this paper, we investigate the problem of resource allocation for power management in MapReduce clusters. Specifically, we propose resource allocation approaches to minimizing the mean end-to-end delay of customer jobs or services under the constraints of the energy consumption and the availability of MapReduce clusters and to minimizing the energy consumption of MapReduce clusters under the availability of MapReduce clusters and the mean end-to-end delay of customer jobs or services. Numerical experiments demonstrate that the proposed approaches are applicable and efficient to solve these resource allocation problems for power management in MapReduce clusters.**

I. Introduction

MapReduce introduced in Dean and Ghemawat [1] has become an important programming model and an associated implementation for the generation and processing of large datasets. Dean and Ghemawat [1] pointed out that more than ten thousand distinct MapReduce programs had been implemented at Google over four years. They have further indicated that a mean of one hundred thousand jobs are executed on MapReduce clusters at Google every day. The increased data-processing power of MapReduce clusters not only greatly enhances a data center s performance but also significantly increases power consumption, which is referred to as an energy and performance dilemma in Iyer et al. [2]. EPA estimated data center energy consumption to double during 2007-2012, costing about $7.4 billion in 2011 (see [3]). The high power consumption of MapReduce clusters has become a major concern since hundreds of MapReduce programs are implemented and thousands of MapReduce jobs are executed in such clusters like Amazon's Elastic MapReduce Clusters daily. Power management becomes one of the most important problems in MapReduce clusters. Furthermore, the availability of MapReduce clusters plays an essential role in the delivery of quality of services (QoS) for customer services.

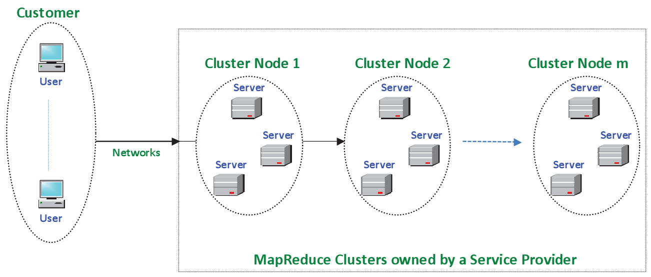

Fig. 1. Customer Service Requests in MapReduce Clusters

This research is concerned with the computing resources of MapReduce clusters owned by a service provider who offers on-demand data-intensive services and applications for a variety of business customers shown in Figure 1. In high-performance and high data-intensive MapReduce service computing environments, the processing of each customer service such as petascale data service applications running is often determined through a service level agreement (SLA). The SLA is often a combination of several QoS requirements such as performance and availability. In this paper, we consider an end-to-end delay that is one of the most important performance metrics. In Figure 1, a customer represents a group of business users that generate a stream of data-intensive service requests at a specified rate, which requires a certain level of QoS. Each service job/request is processed and sent to the customer service provider who owns multiple MapReduce clusters, each consisting of inter-connected cluster servers.

In addition, Chandrakasan et al. [4] and Zhai et al. [5], have showed that the power consumption of a cluster server is at least a cubic increase of its processor speed, namely, there are pros and cons among the performance of a cluster node, power consumption, and cluster availability in MapReduce cluster nodes. Thus, the objectives of this research are to investigate *the following two problems*:

(1) To minimize the mean end-to-end delay of each service request in MapReduce clusters under the constraint of power/energy consumption and the constraint of MapReduce cluster availability.
(2) To minimize the power/energy consumption that is used to process each service request in MapReduce clusters under the constraint of a mean end-to-end delay and the constraint of MapReduce cluster availability.

where the mean end-to-end delay that is the average time to process a service job or request among all the MapReduce computing cluster nodes as well as the energy consumption of each service job or request is a sum of the products of all power consumptions at each MapReduce cluster computing node and the mean time to process the service request at the node. Both of them are referred to as *the energy-aware resource allocation problems*.

Resource allocation problems for various computing systems subject to SLAs such as end-to-end delay, availability, bandwidth, cluster utilization, and packet loss rate have been extensively studied in [6]–[19]. For example, He et al. [12] optimized and managed static workload under a mean delay and a mean miss rate in multi-cluster computing systems. A framework for resource provisioning under a mean delay in grid computing has been proposed in Menasce and Casalicchio [14]. Most of these studies do not consider energy management. There are many studies on power-aware computing and communication systems. But, most of them are concerned with a processor level rather than a system level. For instance, the comprehensive surveys of such studies are given in Benini et al. [20], Unsal and Koren [21], and Venkatachalam and Franz [22]. Most existing research considers power consumption in processor task scheduling with the variable voltages and speeds of processors. In Barnett [23], a mean execution time under a hard energy budget is minimized as well as the expected energy expenditure under a hard execution deadline is minimized for a single task with randomize execution requirements, respectively. Bansal et al. [24] proposed task scheduling approaches according to arrival times and deadlines given on a uniprocessor computer system with minimum energy consumption. Bunde [25] discussed job scheduling under the same requirements on all multiprocessors. A system value maximization problem based on the constraints of time and energy has been studied in Rusu et al. [26]. There are research studies on the energy management of MapReduce and Hadoop *physical* servers/clusters/data centers [27]–[30]. However, most of them are not related to resource allocation.

The above two energy-aware resource problems significantly extend our previous research through the following essential aspects: a) a novel customer service model that reflects real-world applications in MapReduce clusters, b) a consideration of MapReduce cluster availability that is part of QoS requirements defined in an SLA, c) each MapReduce cluster to be modeled as a *multiple* server queue that reflects data-intensive workloads in MapReduce clusters, and d) the algorithms to be developed are different in power and performance management significantly due to the uniqueness of the novel service request model. These differences result in the difficulty of solving the above two energy-aware resource problems numerically. The two energy-aware resource allocation problems are to find a trade-off among performance, energy consumption, and MapReduce cluster availability. All of them require us to give the calculation of an end-to-end delay, the calculation of energy consumption, and the calculation of cluster availability. Thus, in this research we first model the MapReduce clusters to process service requests as a queueing network shown in Figure 1, each cluster node with a multiple server queue. We then derive the calculation of an end-to-end delay of each service job or request and the calculation of its energy consumption. Thus, the two problems are constrained nonlinear optimization in which we have to develop numerical methods to solve them. Both solutions will be expressed as the minimal number of computing servers at each MapReduce cluster.

Moreover, to the best of our knowledge, this is the first queueing model to characterize MapReduce jobs. By using multiple-server queues given in Section III, we model MapReduce job processing. Such modeling for MapReduce jobs in a resource pool is new.

The rest of this paper is organized as follows. Section II provides the formal formulations of the above two energy-aware resource allocation problems. In Section III we propose the approaches and resulting algorithms for solving these problems. Experimental results demonstrate the applicability and efficiency of the proposed algorithms in Section IV. We conclude the paper and discuss our future work in Section V.

II. The Energy-aware Resource Allocation Problems in MapReduce Clusters

In MapReduce, *map* tasks process a block of input referred to as *a file chuck* at each cluster server at a time using default *FileInputFormat*. If the file chuck is very small and there are a lot of them, then many cluster servers are needed to process *map* tasks, each of which imposes extra power consumption and bookkeeping overhead. Thus, a typical block size used by Hadoop file systems (HDFS) is at least 64MB, which is required in the service request model of MapReduce clusters given in Figure 1. This model considers a stream of service requests. Without loss of generality, each request file is assumed to be at least 64 MB in this paper. Otherwise, an accumulation of several small requests is considered as a new request such that its size is over 64 MB. In this service model, the more number of computing servers in each MapReduce cluster station or node, the higher availability of the MapReduce cluster and the faster parallel processing of service requests but the higher energy consumption of each cluster node. The goal of solving the energy-aware resource allocation problems stated in Section I attempts to find an optimal number of cluster servers based on different QoS requirements. In order to formally present the formulation of the energy-aware resource allocation, we first give model assumptions and then mathematically define QoS requirements as follows.

a) *Model assumptions and notation:* In Figure 1, it is assumed that each MapReduce node consists of several MapReduce cluster servers. Let N_j be a total number of servers at cluster node j and n_j be a number of cluster servers allocated to ensure customer service guarantees where $0 \leq n_j \leq N_j$, where $j = 1, 2, \cdots, m$. To reduce the complexity of notation, we give an assumption in this paper: each MapReduce cluster node j consists of one type of servers. Each has the same cost c_j. Otherwise, we can decompose a cluster node consisting of more than one type of multiple servers into multiple sub-nodes so that each node only consists of a set of multiple servers with the same type and the same cost. Let μ_j be the service processing speed of a cluster server in MapReduce cluster node j.

b) *End-to-end energy consumption:* Denote by $\hat{T}$ the mean end-to-end delay of a customer service job or request and $\hat{E}$ the total of mean energy consumption for the customer service in the MapReduce cluster computing system as given in Figure 1. Clearly, $\hat{T}$ will be calculated based on the service processing speed of a cluster server in each MapReduce cluster node, μ_j as well as a number of MapReduce cluster servers denoted by n_j at each MapReduce cluster node. That is, $\hat{T} = \hat{T}(n_1, \mu_1; \cdots; n_m, \mu_m)$.

Furthermore, $\hat{E}$ is computed based on the service rate μ_j, the number of servers n_j, and the power consumption p_j where $j = 1, 2, \cdots, m$. In high-performance clusters, Zhai et al. [5] have shown that the clock frequency, f, in a MapReduce cluster server is proportional to the server's processor speed, μ as well as the server's power consumption, p, is further proportional to the server's clock frequency, f^β, where $\beta \geq 3$. This derives that the power consumption of p is proportional to the processor speed of μ^β in a cluster server. For discussion simplicity, p is assumed to be equal to μ^β. This means that the power energy of a MapReduce server is proportional to a cubic of its processing speed. Thus, there is a trade-off when we decide to maximize the performance of MapReduce clusters and minize their power consumption. Therefore,

$$\hat{E} = \hat{E}(n_1, \mu_1; \cdots; n_m, \mu_m)$$

Let T^D and E^D be the desired end-to-end delay and the desired energy consumption agreed and negotiated between a customer and its service provider according to a fee that has been paid by the customer. They are defined in the SLA between the customer and its service provider.

c) *The availability of MapReduce Clusters:* Availability plays a key role in cluster computing. It characterizes the variations of QoS in a cluster computing system over time. The availability of a computing system is defined by the number of failures that can be tolerated by the system in Brown and Patterson [31]. Each cluster consists of a number of MapReduce servers. Provisioning more servers than necessary prevents a MapReduce cluster from failure that improves the availability of a MapReduce cluster. In this paper, we use a traditional way to define the availability of a MapReduce server as the percentage of time that the MapReduce server is "up" or "down." Furthermore, the availability of MapReduce cluster j is defined as as the percentage of time that the MapReduce cluster has at least n_j MapReduce servers to be "up" at any time since n_j MapReduce servers are needed to ensure QoS requirements defined in an SLA. Denote $MTTF_j$ (Mean Time to Failure) by the mean time of a MapReduce servers failure, and $MTTR_j$ (Mean Time to Recover) by the mean time for recovering a server at cluster j. This means that each MapReduce server failure rate is $1/MTTF_j$ denoted by g_j, and recovering rate (i.e., the rate to put a MapReduce server back into operation) is $1/MTTR_j$ denoted by h_j $(j = 1, \cdots, m)$. Thus, $\eta_j = \frac{g_j}{g_j + h_j}$ is the unavailability rate of a MapReduce server at cluster node j. According to the above assumption, we require that at least n_j servers are up to ensure QoS defined in an SLA. Moreover, Based on the result of Xiong [32], n_j can be determined by using the following statement: MapReduce cluster j has at least n_j servers "up" (i.e., in a full operation) with the probability that

$$P_j(n_j, N_j) = \sum_{i=0}^{N_j - n_j} \frac{N_j!}{i!(N_j - i)!} \eta_j^i (1 - \eta_j)^{N_j - i}$$

Moreover, in MapReduce clusters, data replication is very important. We do not consider the data replication in this paper due to the page limit. But, it can be discussed in a similar fashion.

In what follows, we are going to formulate the two energy-aware resource allocation problems discussed in Section I for the service request model of MapReduce clusters given in Figure 1. Let $\mathcal{I}$ be a set consisting of all positive integers. As we recall, μ_j is the processing speed of each MapReduce cluster server and n_j is the number of servers in MapReduce cluster node j allocated to process MapReduce requests in the service model that is given in in Figure 1 where $j = 1, 2, \cdots, m$. Then, the two minimization problems discussed in Section I can be modeled and formulated as follows.

Problem I: *Minimizing the mean end-to-end delay subject to the constraints of energy consumption and MapReduce cluster availability:* When an arrival rate of MapReduce customer service jobs or requests is given, we need to seek for $n_1, n_2, \cdots$, and n_m to minimize the mean end-to-end delay of the service request expressed as:

$$\min_{\substack{n_j \in \mathcal{I},\, 0 \leq n_j \leq N_j \\ j=1,\cdots,m}} \hat{T}(n_1, \mu_1; \cdots; n_m, \mu_m)$$

subject to the constraint of energy consumption in processing the service request:

$$\hat{E}(n_1, \mu_1; \cdots; n_m, \mu_m) \leq E^D$$

and the constraints of MapReduce cluster availability:

$$P_j(n_j, N_j) \geq \delta_j\% \text{ and } P_{e2e} \geq \delta\%$$

where $n_1, n_2, \cdots$ and n_m are positive integers as well as P_{e2e} is the availability of MapReduce clusters. E^D is an energy constraint, and $\delta_j\%$ and $\delta\%$ are availability constraints in MapReduce cluster j and an end-to-end availability constrain in the MapReduce clusters. All of them are predefined in an SLA.

Problem II: *Minimizing energy consumption with the constraints of an end-to-end delay and MapReduce cluster availability:* When an arrival rate of customer service requests is given, we need to seek for $n_1, n_2, \cdots$, and n_m to minimize the mean energy consumption in processing a service request in the cluster computing system, that is,

$$\min_{\substack{n_j \in \mathcal{I},\, 0 \leq n_j \leq N_j \\ j=1,\cdots,m}} \hat{E}(n_1, \mu_1; \cdots; n_m, \mu_m)$$

subject to the constraint of the mean end-to-end delay for a service job or request:

$$\hat{T}(n_1, \mu_1; \cdots; n_m, \mu_m) \leq T^D$$

and the constraints of MapReduce cluster availability:

$$P_j(n_j, N_j) \geq \delta_j\% \text{ and } P_{e2e} \geq \delta\%$$

where $n_1, n_2, \cdots$ and n_m are positive integers as well as T^D is a predefined end-to-end delay that a customer agrees upon

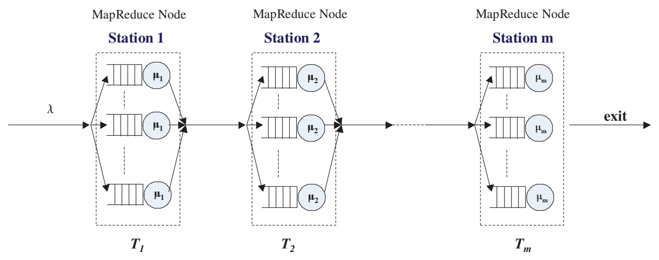

Fig. 2. A Service Request Model

with the service provider based a fee paid by the customer. P_{e2e}, δ_j, and δ are defined in the formulation of Problem II.

III. The Solutions of the Energy-aware Resource Allocation Problems

In this paper, we assume that there are m MapReduce operations such as "Map" and "Reduce." Each operation is executed in one MapReduce cluster (or called cluster node) consisting of "servers selected from a resource pool. We then model each MapReduce node in Figure 1 as a *multiple server* queue. Furthermore, "servers" in the first MapReduce cluster may be re-used in the j-th MapReduce cluster if they are available ($1 < j \leq m$). Thus, the path of a MapReduce service request given in the figure can be modeled as a queueing network model or handled in the pattern or chain of a tandem queue in Figure 2. In the MapReduce service model, there are m MapReduce cluster nodes (it is called MapReduce cluster station alternatively). They are sequentially numbered from 1 to m owned by the service provider depicted in Figure 1. A customer's service request job is first executed in the first MapReduce cluster. When it exits from the cluster node, the service job will continuously be processed at other $m-1$ MapReduce clusters. After the service request job at the m-th MapReduce cluster node is completed, the service request exits the queueing network shown in the figure.

Denote by λ the external arrival rate to the first MapReduce node in the queueing network due to MapReduce service requests, and λ_j the effective arrival rates to MapReduce node j, where $j = 1, 2, \cdots, m$. It is assumed that the service time of each server queue is exponentially distributed and the external arrival to the first MapReduce cluster occurs in a Poisson fashion. That is, MapReduce node j is modeled as a $M/M/n_j$ queue, where let us recall that $1 \leq n_j \leq N_j$ is the number of MapReduce servers allocated to process services in the node for $j = 1, 2, \cdots, m$. The service request model of this paper is different from the ones studied in our previous research such as Xiong and Perros [32]. This is because we consider a multiple server queue in this paper rather than a single one in the previous research. The change complicates our studies but reflects the realistic workload of MapReduce clusters depicted in Figure 2. Because a network propagation delay is minimal compared to the processing time of a service request or job at each MapReduce node in the energy-aware resource allocation problems, we do not include an infinity server in the queueing model. But, it is included in Xiong and Perros [32] and Xiong [33]. In this tandem mode, the end-to-end service availability of MapReduce clusters is: $P_{e2e} = \Pi_{j=1}^{m} P_j(n_j,\, N_j)$.

Next, we calculate both the end-to-end delay and the end-to-end energy consumption are needed in the problems given in Section I. Let $\hat{T}_j$ and $\hat{E}_j$ be the mean delay of a service request and its mean energy consumption at MapReduce node j. Then,

$$\hat{T}(n_1, \mu_1; \cdots; n_m, \mu_m) = \hat{T}_1 + \hat{T}_2 + \cdots + \hat{T}_m$$

According to Kleinrock [34], $\hat{T}_j$ is given by

$$\hat{T}_j = \frac{1}{\mu_j} + \frac{r_j^{n_j} p_j(0)}{n_j!(n_j\mu_j)(1-\rho_j)^2} \tag{1}$$

where $r_j = \frac{\lambda_j}{\mu_j}$, $\rho_j = \frac{\lambda_j}{n_j\mu_j}$, and $p_j(0)$ is computed by

$$p_j(0) = \left(\sum_{k=0}^{n_j-1} \frac{r_j^k}{k!} + \sum_{k=n_j}^{\infty} \frac{r_j^k}{n_j^{k-n_j} n_j!} \right)^{-1}$$

Similarly,

$$\hat{E}(n_1, \mu_1; \cdots; n_m, \mu_m) = \hat{E}_1 + \hat{E}_2 + \cdots + \hat{E}_m \tag{2}$$

where $\hat{E}_j = n_j p_j \hat{T}_j = n_j \mu_j^{\beta} \hat{T}_j$ for $j = 1, 2, \cdots, m$.

Additionally, let us recall that λ is the external arrival rate to the first MapReduce node. The traffic equations of the above queuing network are: $\lambda_j = \lambda$. Moreover, according to the above power consumption, p_j is a cubic relation with its service speed at MapReduce node j. Thus, $\mu_j = p_j^{\frac{1}{3}}$.

(A). The objective and constraint functions of Problems I and II are either increasing or decreasing as n_j. For this reason, we will investigate the property of these problems by decomposing Problems I and II into the subproblems without the constraints of service availability. That is,

Subproblem I: Find integers n_j to minimize the objective function: $\hat{T}(n_1, \mu_m; \cdots; n_m, \mu_m)$ under

$$\hat{E}(n_1, \mu_m; \cdots; n_m, \mu_m) \leq E^D$$

Subproblem II: Find integers n_j to minimize the objective function: $\hat{E}(n_1, \mu_1; \cdots; n_m, \mu_m)$ under

$$\hat{T}(n_1, \mu_1; \cdots; n_m, \mu_m) \leq T^D$$

Then, there exist the solution of the problems and they can be found in the boundaries of their constraints.

Lemma 1: The number of cluster servers $n_j = r_j + \alpha\sqrt{r_j}$ is required at cluster node j to ensure the guarantee of QoS requirements. Then, $\lim_{n_j \to \infty} C(n_j, r_j) = \alpha$, where $C(\cdot,\cdot)$ is the Erlang-C formula, $\alpha = 1 - W_j(0)$ is constant and $1 - W_j(0)$ represents the probability that an arriving customer has a nonzero wait in queue.

The Erlang-C formula represents the probability that an arriving job has a nonzero wait in the queue given by

$$\begin{aligned} C(n_j, r_j) &= 1 - Pr\{\text{no more than } n_j \text{ jobs in the queue}\} \\ &= 1 - p_j(0) \sum_{k=0}^{n_j-1} \frac{r_j^{n_j}}{n_j!} \end{aligned} \tag{3}$$

where Pr stands for the probability.

(B). Due to Lemma 1, it is easy to find a strictly feasible point $(n_1(0), \cdots, n_m(0))$ that is required to satisfy the constraints of Problems I and II since all the constraints are reduced to one dimension with respect to α. We can then apply the interior method to solve these problems.

Algorithms for Solving Problems I and II is scratched as follow. Let us first start with **Algorithm I** for Problem 1 by introducing the barrier function:

$$\Psi_T = \begin{cases} -\frac{1}{E^D - \hat{E}} - \frac{1}{P-\delta} - \sum_{j=1}^{m} \frac{1}{P_j - \delta_j}, & \text{for } (n_1, \cdots, n_m) \in \Omega \\ +\infty, & \text{otherwise} \end{cases}$$

where $\Omega = \{(n_1, \cdots, n_m) | \hat{E} < E_D, P_j > \delta_j$, and $P > \delta$, for $n_j \in \mathcal{I}$ for $j = 1, \cdots, m\}$. We choose the type of $\frac{1}{x}$ as a barrier function because the formulation of inverting the LST of $\hat{T}$ in the research that is much more complicated than the ones in [19] and [35] since $\hat{T}$ is more complicated in this project. The inversion of the LST of $\hat{T}$ is involved in the singularity of its integrand and has no closed-form solution. Moreover, Problem I is transformed into the unconstrained minimization problem as follows: $\min_{n_1, \cdots, n_m \in \mathcal{I}} \quad (\hat{T} + \tau \, \Psi_T)$. It follows from barrier Ψ_T that the above objective function "blows up" at the boundary that consists of $\hat{E} = E_D$, $P_j = \delta_j$, and $P_{e2e} = \delta$. This means that the minimization problem has a "barrier" to cross the boundary. We thus propose an algorithm to remove the barrier gradually through a reduction of τ toward zero.

Similarly, we can derive **Algorithm II** for solving Problem II by using the barrier function of Problem II.

IV. Numerical Experiment Results

In order for us to investigate the applicability and efficiency of the above two algorithms for solving energy-aware resource allocation problems, we have conducted many numerical experiments for a variety of scenarios. We select and present some of them due to the space limit. Moreover, as mentioned before energy consumption is proportional to a number of MapReduce servers allocated to server MapReduce services at each MapReduce node. Thus, an optimal resource allocation solution is not expressed as power consumption at each MapReduce node. Instead, it is expressed by the number of MapReduce cluster servers.

In our experiments, we use Arena (see Altiok and Melamed [36]) to simulate the proposed queueing network in this paper. Since the Arena simulation experiment exactly represents the queueing network in this research, the simulation results obtained in Arena are considered as "exact." The implementations of Algorithms 1 and 2 were done in Mathematica. Moreover, all parameter values are unitless in this section.

Due to the page limit, we present the experimental example that is used for both Algorithms 1 and 2. Let us select $m = 8$, $\lambda = 1162.8$, $T^D = 12$, $E^D = 35,000$, and $\beta = 3$. For $j = 1, \cdots, 8$, μ_j and η_j are given in Table I, $N_j = 45$, $\delta_j = 99.99$, and $\delta = 99.98$. From the traffic equation, we have that $\lambda_j = \lambda = 1162.8$.

By using Algorithm 1, we have found that the number of servers required to ensure power consumption and cluster availability is 12, 29, 32, 15, 16, 22, 19, and 23 for MapReduce nodes 1 to 8, respectively. Similarly, through Algorithm 2, we have obtained that the number of servers necessary to ensure end-to-end delay and cluster availability is 9, 24, 26, 12, 12, 16, 13, and 15 for MapReduce nodes 1 to 8, respectively. Furthermore, we also used using the brute-force approach to simulate the service model. It has been validated that these numbers of servers obtained through the proposed approximate method are indeed optimal. Moreover, we have conducted numerous experiments in cases of different m. For example, when only the first five MapReduce nodes are considered in the above experiment where all parameter values are the same except $E^D = 16,000$, the solutions are 11, 27, 30, 14, and 14 for Problem I, as well as 10, 25, 27, 12, and 12 for Problem II, respectively.

TABLE II. The Execution Times of Running Algorithms 1 and 2

m	4	8	16	32	64
Algorithm 1	1.55	5.92	21.38	45.42	196.85
Algortihm 2	1.69	1.48	24.19	49.72	215.98

In this research, we have also experimentally investigated the cases of much more number of MapReduce cluster nodes, for example, m is chosen as 16, 32, and 64, respectively. Table II gives the execution times of running Algorithms 1 and 2 in seconds when the numbers of stations are 4, 8, 16, 32, and 64.

As we notice, the execution times of running Algorithms 1 and 2 do not increase dramatically as the number of MapReduce cluster nodes increases. As a matter of fact, it is observed that they approximately increase with the number of cluster nodes, which is consistent with the results given in Section III. Therefore, our algorithms are applicable to deal with the case of a large number of MapReduce cluster nodes. We have also analyzed the performance of our algorithms via other system parameters. However, we do not present their experimental results in detail due to space limit.

V. Conclusions and Future Work

In this research, we have systematically investigated the performance, energy consumption, and service availability of MapReduce clusters for MapReduce service request. We have further modeled their relationship as the following two minimization problems: (1) minimizing the mean of an end-to-end delay under the constraints of mean energy consumption and service availability, and (2) minimizing the total mean energy consumption under the constraints of a mean end-to-end delay and service availability.

By investigating an intrinsic property of the two problems, we have first developed two algorithms for solving them. Our numerical experiments have showed that the proposed algorithms are applicable and efficient to solve these two energy-aware resource allocation problems in MapReduce clusters. That is, we have proposed the resource allocation algorithms for high-performance and energy-efficient MapReduce clusters presented in this paper. The proposed research can be extended to discuss other MapReduce service models.

TABLE I. THE VALUES OF PARAMETERS μ_j AND η_j

j	1	2	3	4	5	6	7	8
μ_j	174.42	58.14	51.68	130.815	122.4	93.024	123.12	116.28
η_j	0.39	0.25	0.31	0.22	0.24	0.2	0.35	0.28

ACKNOWLEDGMENT

K. Xiong's work is supported in part by the U.S. National Science Foundation (NSF) under CNS 1065665 and NSF/BBN under #1895 through CNS 1128122, the Air Force Research Laboratory (AFRL), and the Office of Naval Research (ONR). The contents of this paper do no necessarily reflect the position or the policies of the U.S. Government. K. Xiong's work is also supported in part by Amazon AWS, and the College of Computing and Information Sciences under the SEED and FEAD grants at Rochester Institute of Technology.

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: simplified data progrssing on large clusters," *Communications of the ACM*, vol. 51, pp. 107–113, 2008.

[2] B. Iyer, J. Beu, and T. Conte, "Length adaptive processors: A solution for energy/performance dilemma in embedded systems," in *Proceedings of the 12th IFIP conference on Human-Computer interaction (INTER-ACT)*, 2009.

[3] EPA, "EPA reports: significant energy efficiency opportunities for u.s. servers and data centers," in *http://yosemite.epa.gov/opa/admpress.nsf/0de87f2b4bcbe56e852572a000651fde/4be8c9799fbceb028525732c0053e1d5!*, 2007.

[4] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low power CMOS digital design," *IEEE Journal of Solid State Circuits*, vol. 27, 1992.

[5] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," *Proceedings of 41st Design Automation Conference (DAC'04)*, pp. 868–873, 2004.

[6] D. Ardagna, M. Trubian, and L. Zhang, "SLA based resource allocation policies in autonomic environments," *Journal of Parallel and Distributed Computing*, vol. 67, no. 3, pp. 259–270, 2007.

[7] J. Bolot, "End-to-end packet delay and loss behavior in the Internet," in *Proceedings of the ACM SIGCOMM*, 1993.

[8] E. Bouillet, D. Mitra, and K. Ramakrishnan, "The structure and management of service level agreements in networks," *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 691–699, 2002.

[9] C. Chassot, F. Garcia, G. Auriol, A. Lozes, E. Lochin, and P. Anelli, "Performance analysis for an IP differentiated services network," in *Proceedings of IEEE Intnernational Conference on Communication (ICC)*, June 2002.

[10] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation based congestion control for unicast applications," in *Proceedings of the ACM SIGCOMM*, 2000.

[11] Y. Fu and A. Vahdat, "Service level agreement based distributed resource allocation for streaming hosting system," in *Proceedings of the 7th International Workshop on Web Content Caching and Distribution (WCW)*, August 2002.

[12] L. He, S. Jarvis, D. Spooner, and G. Nudd, "Optimising static workload allocation in multiclusters," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS)*, 2004.

[13] J. Martin and A. Nilsson, "On service level agreements for IP networks," in *Proceedings of the IEEE INFOCOM*, June 2002.

[14] D. Menasce and E. Casalicchio, "A framework for resource allocation in grid computing," in *Proceedings of the 12th IEEE MASCOTS*. IEEE, October 2004, pp. 259–267.

[15] J. Nabrzysbi, J. Schopf, and J. Weglarz, *Grid Resource Management*. Kluwer Academic Publication, 2004.

[16] D. Nowak, P. Perry, and J. Murphy, "Bandwidth allocation for service level agreement aware Ethernet passive optical networks," in *Proceedings of the IEEE GLOBECOM*, 2004.

[17] S. Penmatsa and A. Chronopoulos, "Price-based user-optimal job allocation scheme for grid systems," in *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[18] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A resource allocation model for QoS management," in *Proceedings of the 18th RTSS*, December 1997.

[19] K. Xiong, "Energy-aware resource provisioning in cluster computing," in *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2010.

[20] L. Benini, A. Bogliolo, , and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 8, pp. 299–316, 2000.

[21] O. Unsal and I. Koren, "System-level power-aware design techniques in real-time systems," *Proc. IEEE*, vol. 91, pp. 1055–1069, 2003.

[22] V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Computing Surveys*, vol. 37, pp. 195–237, 2005.

[23] J. Barnett, "Dynamic task-level voltage scheduling optimizations," *IEEE Trans. Computers*, vol. 54, 2005.

[24] N. Bansal, T. Kimbrel, and K. Pruhs, "Dynamic speed scaling to manage energy and temperature," in *Proceedings of the 45th Annual IEEE Symposium Foundations of Computer Science (FOCS'04)*, 2004.

[25] D. Bunde, "Power-aware scheduling for makespan and flow," in *Proceedings of the 18th ACM Symposium Parallelism in Algorithms and Architectures (SPAA'06)*, 2006.

[26] C. Rusu, R. Melhem, and D. Mosse, "Maximizing the system value while satisfying time and energy constraints," in *Proceedings of the 23rd IEEE Real-time System Symposium (RTSS'02)*, 2002.

[27] Y. Chen, A. Ganapathi, A. Fox, R. Katz, and A. Patterson, "Statistical workloads for energy efficient MapReduce," in *Technical report at University of California at Berkeley, http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-6.pdf*, 20010.

[28] Y. Chen, A. Ganapathi, and R. Katz, "To compress or not to compress - compute vs. I/O tradeoffs for MapReduce energy efficiency," in *Technical report at University of California at Berkeley, http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-36.pdf*, 20010.

[29] Y. Chen, L. Keys, and R. Katz, "Towards energy efficient MapReduce," in *Technical report at University of California at Berkeley, http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-109.pdf*, 2009.

[30] J. Leverich and C. Kozyrakis, "On the energy (in)efficiency of Hadoop clusters," *ACM SIGOPS Operating Systems Review*, vol. 44, 2010.

[31] A. Brown and D. Peterson, "Towards availability benchmarks: A case study of software raid systems," in *Proceedings of the USENIX Annual Technical Conference*, 2000.

[32] K. Xiong and H. Perros, "SLA-based resource allocation in cluster computing systems," in *Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2008.

[33] K. Xiong, "Multiple priority customer service guarantees in cluster computing," in *Proceedings of the 23rd International Parallel and Distributed Processing Symposium (IPDPS)*, May 2009.

[34] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. Wiley, New York, 1975.

[35] K. Xiong, "Power and performance management in priority-type cluster computing systems," in *Proceedings of the 25th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2011.

[36] T. Altiok and B. Melamed, *Simulation Modeling and Analysis with Arena*. Cyber Research, Inc., 2001.