Reliability-Aware Energy Efficiency in Web Service Provision and Placement

Ying Chen, Peng Zhang, Xiangzhen Kong, Chuang Lin
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
chenying12@mails.tsinghua.edu.cn pzhang@csnet1.cs.tsinghua.edu.cn
xiangzhen1985@gmail.com chlin@tsinghua.edu.cn

***Abstract*—Reliability is a critical concern in the provision and placement of web services. A breakdown of service would seriously reduce customers' satisfaction, and thus harm the revenue of service providers. To maintain a high reliability, the common approach is deploying multiple service instances across different physical servers. This would inevitably raise another concern of energy consumption. Thus, greening web services also becomes an important issue. In this paper, we study the fundamental tradeoff between reliability and energy consumption, and propose an optimization framework that considers both factors. In specific, we build a continuous-time Markov model to analyze the steady-state reliability and mean time to failure (MTTF) from a service-oriented perspective, and obtain the minimum number of service instances to meet the given reliability requirement. Then, we show that deploying these instances in the server cluster to minimize energy consumption is NP-hard. To this end, we propose a heuristic algorithm to approximate the result. The analytical and experimental results show the effectiveness, and the approximation ratio is less than 1.25 for 90% of the data sets we use.**

Keywords- Web service; reliability; energy; optimization

I. INTRODUCTION

Web services are gaining in popularity in both industry and academe. They can realize effective Business-to-Business (B2B) collaboration by automation of interoperation [1]. Typical examples of web services include booking an airline ticket, order procurement, finance, accounting, human resources, supply chain, and manufacturing [1], [2].

Reliability is a critical requirement in the design of web service systems, especially for applications such as account management, traffic control and military applications. It is an important attribute of Quality of Service (QoS) and is often included in the Service Level Agreement (SLA) [20], [22]. A breakdown of service would seriously reduce customers' satisfaction. An investigation shows that customer satisfaction is lower after service failure than in the case of error-free service, even given high-recovery performance [3]. There has been some work on the modeling and analysis of reliability [4], [5], [6]. However, the traditional study of reliability was focused on the component and physical hardware. Due to the complex service composition, resource sharing and large-scale physical machines, traditional study of reliability at the component and physical hardware level is not suitable for web service systems.

Besides, energy consumption is becoming a serious concern and energy efficiency has drawn more and more attention. It was estimated in [7] that the Google search consumes more than 32 million kWh resulting from the report that the energy consumed during an average Google search was 0.0003kWh. In [8], it is shown that service centers consume more than 1.5 percent of the power produced in the US, and the number is still increasing. According to industry analysis, enterprise data centers have doubled their energy usage in five years [9]. As the energy consumption is becoming a serious concern, it is the trend to consider energy efficiency in the design and implement of web service systems.

Considering the crucial reliability demand of web services and the importance of energy efficiency, it is necessary to combine these two factors together when studying the service provision and placement problem. Although there have been some pieces of work on combining reliability and energy efficiency, their focuses are other fields such as wireless networks or multiprocessor real-time systems [10], [11]**.** Reliable and energy-efficient service provision and placement problem in web service systems are largely unexplored.

In this paper, we study the reliability-aware energy efficiency problem (REEP) in web service provision and placement. We study the service-centric reliability and build a continuous-time Markov chain to model it. We use phase-type distribution to measure the reliability and mean time to failure (MTTF) of services. Then we develop an optimization model to minimize the energy consumption in web service systems. We prove the optimization for REEP to be NP-hard and propose an approximation algorithm to solve it. To verify the accuracy of the approximation algorithm, we compare the approximate solutions with optimal solutions obtained based on the LINGO software, which shows the effectiveness of the algorithm. The dynamics of services is also considered in this paper. The experiments show that the solution can reduce the energy consumption in the system while meeting the reliability demand. Also, we analyze the trade-off between reliability and energy consumption based on the experimental results.

Our main contribution is two-fold. First, we propose an optimization framework which jointly considers system reliability and energy consumption in web service systems.

Second, we prove this optimization problem to be NP-hard, propose an efficient algorithm to approximately solve it and conduct analysis to demonstrate its accuracy.

The remainder of this paper is organized as follows. Section 2 introduces the background of the service provision and placement problem in web service systems. In Section 3, we develop the reliability and energy efficiency models for service systems, and propose algorithms to solve REEP. In Section 4, we conduct experiments to analyze the results of our model and algorithms. Section 5 discusses related work and Section 6 concludes the paper.

II. BACKGROUND

Many web services run on middleware systems which are responsible for the management of resources. Some middleware systems use clustering technology to improve scalability and reliability, by integrating multiple instances of services, and presenting them to the users as a single service [12].

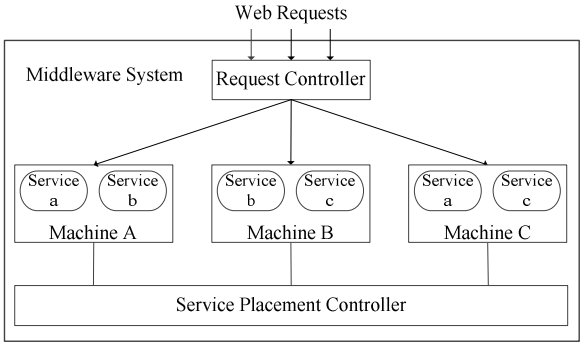

Figure 1. An example of a clustered web service system

Fig. 1 shows a typical example of a middleware system, which consists of one request controller, one placement controller and three machines. The request controller is responsible for flow control and admission control, etc. There are several services in the system, each of which may be replicated for high-reliability and performance. Here the replicas of services are called **instances** and are running on multiple physical machines. The placement controller decides how to deploy instances across physical machines, with the constraint that the total resource demand of instances on one machine should not exceed its resource capacity. Optimization considerations include reliability, responsiveness, throughput, energy consumption, etc. In this paper, we will consider two of them, i.e., reliability and energy consumption.

We are to answer the following question:

Given a set of services and a set of physical machines, how many instances should be provided for each service and which machines should them be deployed on, so as to meet reliability demand and be energy-efficient?

This problem can be formulated using an optimization framework. The goal is to minimize the total energy consumption in the cluster. Reliability demand is used as a constraint. Other conditions include resources constrains and overload protection.

The notations and definitions used in this paper are shown in Table 1.

TABLE I. NOTATIONS AND DEFINITIONS

notation	Definition
n	Number of services
c_i	Number of instances of service i
m	Number of physical machines
$P_{n\times m}$	The mapping matrix of services to machines
x_{ij}	The element in $P_{n\times m}$, denoting whether service i is deployed on machine j
y_j	The state of machine j
O_j	The set of instances that are on machine j
CPU_j	The CPU capacity of machine j
MEM_j	The memory capacity of machine j
cpu_i	The CPU demand of service i
mem_i	The memory demand of service i
ρ_u	The upper bound of utility of machines
r_{sev_i}	The reliability of service i
R_i	The reliability demand of service i

III. MODEL FOR REEP

This paper seeks to meet the reliability demand of services and minimize energy consumption in the cluster, subject to resource constraints. We will first discuss how to evaluate reliability and meet the demand.

A. Modeling Service-Centric Reliability

According to [6]，reliability is the ability of the system to offer service without interruption during a period of time. Formally speaking, the reliability of a system is the probability that the system functions properly and continuously during the time period $(t,t+\tau)$, given that the system is in normal state at the time t:

$$r(t,\tau)=\Pr(ready(t,t+\tau)\,|\,ready(t)) \quad (1)$$

In steady-state, the reliability can be analyzed by the probability that it can keep on service without any failure during the time interval $(0,\tau)$, assuming that it is in normal state at time 0. The above expression can be abbreviated as:

$$r(\tau)=\Pr(ready(0,\tau)\,|\,ready(0)) \quad (2)$$

Different from previous work focusing on the hardware [4], [5], we veer from the angle of view, and study the reliability from the service-oriented perspective by taking a top-down approach and characterize it using mean time to failure (MTTF). As for the reliability demand, there can be different levels and policies. For example, some services

may be guaranteed that they can work properly for one day at the probability of 0.99999, some may be assured to function without failure for 7 days at the probability of 0.9999.

We build a continuous-time Markov chain (CTMC) to model the lifecycle of a service. Assuming there are c_i instances of service i in the cluster, the corresponding CTMC can be described in Fig. 2.

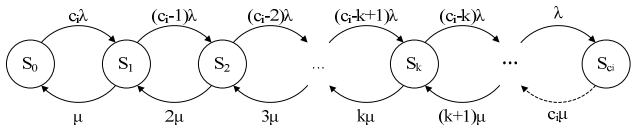

Figure 2. The Markov chain of the relialibility model

In this CTMC, there are (c_i+1) states $\{S_0, S_1, S_2, ..., S_{c_i}\}$, denoting the number of failed instances of service i. In order for the service to be functioning properly, there should be at least one instance that is in normal states. We care about the time it takes the service to transfer from normal states to abnormal state S_{c_i}. Once the service runs into the state S_{c_i}, the reliability of the service could not be satisfied any more. Therefore the state S_{c_i} is made to be an absorbing state in reliability analysis.

We define λ as the failure rate of one instance and μ as the repair rate. In state S_k, there are k failed instances, therefore the transmission rate of state S_k to state S_{k+1} is expressed as $(c_i-k)\lambda$, and the transition rate to state S_{k-1} is $k\mu$. Let $\boldsymbol{\pi}=(1,0,0,...0)$ be the initial state of the CTMC, assuming that at the very beginning, all the c_i instances are in normal states.

The intensity matrix of the CTMC can be given by $\widetilde{\mathbf{Q}}=[q_{j,k}]$, where,

$$\begin{cases} q_{j,j+1}=(c_i-j)\lambda & j=0,1,...c_i-1 \\ q_{j,j-1}=j\mu & j=1,2,...c_i\text{-}1 \\ q_{j,j}=-\left((c_i-j)\lambda+j\mu\right) & j=1,2,...c_i\text{-}1 \\ q_{j,k}=0 & \text{otherwise} \end{cases}$$

Let τ denote the time it takes the service to enter the absorbing state S_{c_i} for the first time. Hence the MTTF could be evaluated by the expectation value $E(\tau)$. The CTMC built in this paper is a chain with absorbing state and the state space is finite. In [13], it has been proven that τ conforms to a *phase-type* distribution, expressed as $\tau \sim PH(\boldsymbol{\pi}, \mathbf{Q})$, where $\boldsymbol{\pi}$ is the initial state, and $\mathbf{Q}$ is the intensity matrix of the transient states. The whole intensity matrix has the general form as

$$\widetilde{\mathbf{Q}}=\begin{pmatrix} \mathbf{Q} & \mathbf{q}_0 \\ \mathbf{0} & 0 \end{pmatrix}$$

According to [13], the distribution function of τ can be expressed as (3), where $\mathbf{e}$ is a $c_i \times 1$ vector with all elements being 1.

$$F(x)=\Pr(\tau \le x)=1-\boldsymbol{\pi}\exp(\mathbf{Q}x)\mathbf{e} \quad (3)$$

Therefore the reliability of the service can be expressed as

$$r(\tau)=\Pr(ready(0,\tau)\,|\,ready(0))=1-F(\tau) \quad (4)$$

The expectation of τ, which denotes the MTTF of the service is expressed as

$$MTTF=E(\tau)=-\boldsymbol{\pi}\mathbf{Q}^{-1}\mathbf{e} \quad (5)$$

With this model, one can obtain the optimal value of c_i with Algorithm 1 given the requirement of $R(\tau)$ and/or $E(\tau)$ and the system parameters λ and μ.

Algorithm 1: search algorithm for the optimal value of c_i

Input: Reliability demand $R(\tau)$ and mean time to failure demand $E(\tau)$, parameters λ, μ.

Output: c_i.

1. Initialize c_i=1;
2. obtain the intensity matrix $\mathbf{Q}$;
3. obtain reliability $r(\tau)$ and mean time to failure MTTF;
4. **if** ($r(\tau) < R(\tau)$) or (MTTF$< E(\tau)$) **then**
5. $c_i = c_i + 1$;
6. Return to step 3;
7. **end if**

B. Energy Efficiency

Then we discuss the energy efficiency, formulate it and build the mathematical model.

There are totally n services and m machines in the cluster. For each service i, the cluster should provide c_i corresponding instances, where c_i is obtained using the CTMC model proposed above to meet the reliability demand. Define the deployment of services on machines as 0-1 matrix $P=\{x_{ij}\}_{n\times m}$, where $x_{ij} \in \{0,1\}$ denotes whether service i is deployed on the physical server j.

As there are several instances of one service residing on different servers, there might exist several j such that $x_{ij}=1$, while satisfying $\sum_j x_{ij}=c_i$.

Let y_j be the state of machine j, y_j=1 expresses that machine j is in working state and y_j=0 means that it is in standby state. It is considered that if there is no instance on one machine, then it is in standby state.

The REEP discussed in the paper takes into account resource constraints and overload-protection. As for resource constraints, we consider CPU and memory, which are usually two most important factors to consider in resource

management problems. Let CPU_j denotes the CPU capacity of machine j and MEM_j be the memory capacity. cpu_i expresses the CPU demand of service i and mem_i be the memory demand. O_j is the set of instances that are on machine j. It should satisfy that $\sum_{i \in O_j} cpu_i \le CPU_j$.

The overload-protection is also considered to avoid one machine from being over-loaded. In this paper we choose a representative metric of the machine's utilization as the combination of CPU utilization and memory utilization. Define ρ_{ic} as the CPU utilization of the service i, ρ_{im} as the memory utilization. Therefore the utilization of a service can be expressed as $\rho_i = \alpha_1 * \rho_{ic} + \alpha_2 * \rho_{im}$, where α_1, α_2 are parameters representing the corresponding weights, satisfying that $\alpha_1 + \alpha_2 = 1$. Let ρ_u be the upper bound, which can be set according to the different quality of physical machines in different situations. Therefore the overload-protection policy should effectively manage the resource and place the services satisfying that $\sum_{i \in O_j} \rho_i \le \rho_u$.

Define E_j to be the energy consumption of the machine j. The problem seeks to minimize the energy consumption of all the machines in the cluster, i.e., $\sum E_j$.

So the problem can be formulated as follows.

$$\underset{x_{ij}}{\text{minimize}} \quad \sum E_j \tag{6}$$

Subject to

$$\sum_{i \in O_j} cpu_i \le CPU_j \quad \forall j \in \{1,2...m\} \tag{7}$$

$$\sum_{i \in O_j} mem_i \le MEM_j \quad \forall j \in \{1,2...m\} \tag{8}$$

$$\sum_{i \in O_j} \rho_i \le \rho_u \quad \forall j \in \{1,2...m\} \tag{9}$$

$$\sum_{j} x_{ij} = c_i \quad \forall i \in \{1,2...n\} \tag{10}$$

$$x_{ij} \in \{0,1\} \forall i \in \{1,2...n\} \forall j \in \{1,2...m\} \tag{11}$$

The objective function is to minimize the energy consumption in the cluster. Equation (7) and (8) show the resources constraints referring to CPU and memory. Equation (9) denotes that the utilization of machines should not exceed the upper bound. Equation (10) is used to guarantee that c_i instances are provided for service i. c_i is the number of instances to provide for service i obtained by the CTMC model to meet the reliability demand.

C. Model Specialization

Kephart et al. [14] have argued that one of the most effective ways to save energy is workload consolidation and powering off the spare physical machines. Much work has proven that the energy consumption of a machine in working state is greatly different from that in standby state. Therefore in this paper we care most about whether the machine is in working state or in standby state instead of the slight variation of energy consumption in working state in detail. The objective function to minimize energy consumption can be reduced to minimizing the number of working machines in the cluster and can be written as $\sum y_j$.

As for the resources constraints, take the CPU resource for example. Equation (7) is used to restrict that the load of CPU on machines should not exceed the capacity. The expression can be written as $\sum cpu_i * x_{ij} \le CPU_j * y_j$.

The problem can be specialized as

$$\underset{x_{ij}}{\text{minimize}} \quad \sum y_j \tag{12}$$

Subject to

$$\sum_i cpu_i * x_{ij} \le CPU_j * y_j \quad \forall j \in \{1,2...m\} \tag{13}$$

$$\sum_i mem_i * x_{ij} \le MEM_j * y_j \quad \forall j \in \{1,2...m\} \tag{14}$$

$$\sum_i \rho_i * x_{ij} \le \rho_u * y_j \quad \forall j \in \{1,2...m\} \tag{15}$$

$$\sum_j x_{ij} = c_i \quad \forall i \in \{1,2...n\} \tag{16}$$

$$x_{ij} \in \{0,1\}, y_j \in \{0,1\} \forall i \in \{1,2...n\} \ \forall j \in \{1,2...m\} \tag{17}$$

We can prove that the problem formulated by this model is NP-hard.

Theorem 1: The problem formulated by this model is NP-hard.

Proof: We prove the theorem by reduction from multi-dimensional bin-packing problem. So far much research has been done to study the multi-dimensional bin-packing problem which is well known to be NP-hard [15]. We first give the formal definition of it: assume there is a list of items $V = (v_1, v_2, v_3, ...v_n)$ to be packed; each of the items is a d-dimensional vector $(r_1, r_2, r_3, ...r_d)$ representing the demand. The bin can also be represented by d-dimensional vector $(R_1, R_2, R_3, ...R_d)$ and R_i denotes the capacity of the dimension i. The goal of multi-dimensional bin-packing problem is to pack the items into the bins so as to use the minimum number of bins.

As for REPP, suppose there are n services and c_i instances that should be provided for each service i. We need to deploy the instances on servers and seek to use the minimum number of servers without violating the capacities. Remind that it is a special case of this problem if $c_i = 1$ for all service i, which can be reduced from three-dimensional bin-packing problem: each service is viewed as an item with three-dimensional vector (cpu_i, mem_i, ρ_i) and the goal is to use minimum number of servers to pack those services. It is easy to prove the proposed problem is NP-hard by reduction to absurdity. Assume the problem is not NP-hard, and can be solved in polynomial time. Then the special case of it, which could be reduced from the three-dimensional bin-packing problem, can also be solved in polynomial time. However, it is contrary to the theory that the multi-dimensional bin-packing problem is NP-hard and

could not be solved in polynomial time. Therefore the problem proposed in this paper is NP-hard. ■

Since the problem is NP-hard, it is not easy to find the optimal solution. Much work has been devoted to finding the approximate solutions to the multi-dimensional bin-packing problem and many algorithms have been proposed. In the paper we use an approximation algorithm evolved from the FFD (first fit decreasing) algorithm [16], which is widely-used and easy to implement.

The algorithm in detail is shown as Algorithm 2. The input of the algorithm is the number of services and the corresponding number of instances to be provided. Step (1) initializes this problem's decision variable x_{ij} and y_j. Step (2) sorts the services according to their demand in non-increasing order. It is not unique as for how to compare the demand of different services since it includes both CPU and memory. Although there can be several methods, it introduces no changes to our model and algorithm, and here in this paper we choose a combination of CPU and memory demand $0.5 \cdot \frac{cpu_i}{CPU} + 0.5 \cdot \frac{mem_i}{MEM}$ as the metric. Step (3) to step (17) decides how to deploy those instances of services. Step (5) to step (13) finds a first-fit machine for the present instance. Step (6) judges whether any instance of the present service has been deployed on the machine; if so, the machine is passed since it is considered in this paper that two instances of the same service should be deployed on different servers. Once the first physical server that satisfies those constraints is found, x_{pk} is set to be 1. To consider the exceptional case, if the expected utilizations of all the servers are higher than ρ_u, the algorithm chooses the server with the lowest utilization to deploy the instance where the resource constrains could remain satisfied.

Algorithm 2 obtains the matrix $[x_{ij}]_{n\times m}$, which denotes how the instances of services are deployed on the servers. In real systems, services can register or leave the system dynamically [18]. Accordingly, the algorithm can rerun after a period of time and produces a new solution to the service provision problem.

As for the worst-case time complexity of Algorithm 2, it is mainly composed of two parts: sorting process time and search process time. The sorting process ranks the n services and costs at least $O(n \cdot \log(n))$ time. There are totally $\sum_{i=1}^{n} c_i = N$ instances to be deployed. For each instance, it takes at most m times to find the first server that can hold it. Therefore the worst-case time complexity is $O(n \cdot \log(n) + N \cdot m)$. Actually the search process can be improved by using more efficient search method such as heap search. Then the worse-case time complexity can be improved to be $O(n \cdot \log(n) + N \cdot \log(m))$.

Algorithm 2: approximation algorithm for REEP

Input: number of services n, number of instances c_i of each service i, number of physical servers m

Output: service deployment matrix $[x_{ij}]_{n\times m}$, server state vector $\vec{y}$

1. Initialize the element x_{ij} to be 0, set y_j =0 for all servers
2. Collect the integration demand of each service, then sort them in non-increasing order
3. **for** service p = 1 to n **do**
4. **for** instance j = 1 to c_p of service p **do**
   //Find a server k to deploy the instance
5. **for** server k = 1 to m **do**
6. **while** (x_{pk} ==1) **do**
7. k ++; // instances of the same service // should be on different servers
8. **end while**
9. **if** it is satisfied that

$$\sum_i cpu_i * x_{ik} + cpu_p \le CPU_k$$

$$\sum_i mem_i * x_{ik} + mem_p \le MEM_k$$

$$\sum_i \rho_i * x_{ik} + \rho_p \le \rho_u$$

   then
10. Set x_{pk} =1, y_k =1;
11. Break;
12. **end if**
13. **end for**
14. j ++;
15. **end for**
16. p ++;
17. **end for**

IV. EXPERIMENTAL RESULTS

A. Service Reliability

In this section, we analyze and validate the model and algorithm proposed in this paper. We first evaluate reliability, which is determined synthetically by the failure rate, the repair rate, number of instances and period of time. We use the failure data set at LANL [17] to analyze and simulate our model and algorithm, which contains data collected over the past nine years from 1996 to 2005, covering systems including a total of 4,750 machines and 24,101 processors. All failures during the process of services at LANL are recorded in this dataset. The failures are categorized into 6 classes which are Facilities failure, Hardware failure, Human Error, Network failure, Software failure and Undetermined failure. In this work, we do not distinguish the category of failures. The sojourn time in normal state and failure state can be obtained by statistic methods from the trace in the dataset. Then the failure rate and repair rate can be obtained by calculating the reciprocals of the sojourn times in normal state and failure state, respectively.

Fig. 3 shows the log linear trend of reliability according to the number of instances. In our experiment, we use $\ln 0.9$, $\ln 0.99$, $\ln 0.999$, etc., as references to evaluate the reliability. We analyze the reliability during one day, four days and seven days as examples.

As it can be seen from Fig. 3, with the increase of the number of instances, the reliability of service also increases. In fact, more redundancies can improve service reliability. The reliability of a service that works continuously for one day is larger than that for two days and seven days since it is harder to guarantee that the service would not fail as time increases. Actually, if we let the period of time $\tau \to \infty$, then the reliability $R(\tau) \to 0$ for it is not possible to work properly and continuously forever. Given the reliability demand, we can obtain the number of instances that should be provided for each service.

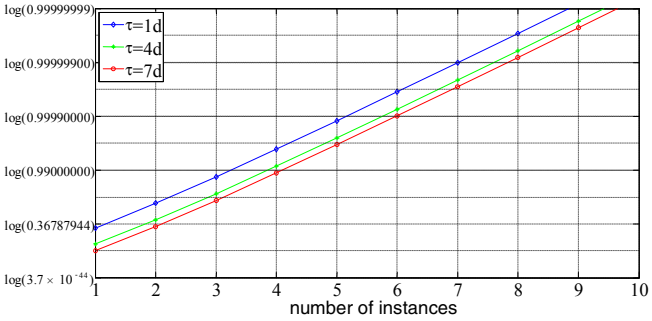

Figure 3. The relationship between reliability and instance number

B. Comparison with Optimal Solution

The evolved FFD algorithm is used to deploy the instances on the physical servers in the cluster to reduce the number of servers that are on. We compare the results of the proposed algorithm with the optimal solutions obtained by LINGO software, which finds the optimal solutions using exhaustive search. It has been illuminated that the proposed algorithm could get an approximate solution in polynomial time, while the optimal solution could not be obtained in polynomial time as the problem is proven to be NP-hard. Therefore, the proposed algorithm could reduce the operation time significantly especially for large scale systems and services.

Let r_1 be the number of working-state physical servers obtained by the evolved FFD algorithm and r_2 be the optimal solution obtained by the LINGO software. Therefore, the approximation ratio can be written as r_1 / r_2. The number of instances in the cluster is varied from 10 to 21, and for each case we run more than 10 experiments by varying the resource constraints of CPU and memory to get the approximation ratio. We do not compare the results for instances number larger than 21, because in such cases there are hundreds of constraint inequalities and it takes more than one hour to get the optimal solution by LINGO software using the machine with a dual-core CPU at 3.3GHz frequency.

Fig. 4 shows the cumulative distribution function (CDF) of the approximation ratio. It is shown that the solutions of the proposed algorithm are larger than the optimal solutions. The approximation ratio lies between 1 and 1.5 and is less than 1.25 for 90% of the data sets we use, which validates the effectiveness of our proposed algorithm. Besides, with the number of instances increasing, the increasing rate of time complexity of the approximation algorithm is greatly smaller than that of using LINGO software to get the optimal solution. Therefore, the proposed algorithm is both precise and efficient.

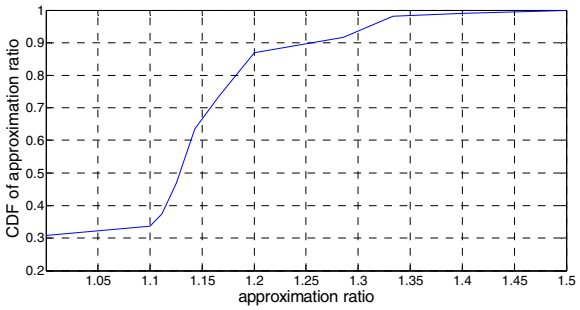

Figure 4. CDF of approximation ratio

C. Energy Efficiengy

We use the power consumption data collected in our lab to simulate our model and algorithm. We measure the real-time power consumption of the Dell PowerEdge R710 servers for 25 minutes by power-measuring equipment. We start with the standby state, then wake on the servers, record the power consumption and obtain the average power consumption of servers at standby state and working state. The average power consumption in working state is 150.0 W, while in standby state it is 7.7 W. Comparing the difference of power consumption between standby state and working state, it can be concluded that the two-value assumption of our model makes sense.

We show how the model and algorithm can reduce energy under certain reliability demand. We conduct two simulation experiments, the first one is before energy optimization and with load-balanced mechanism, where there are fixed number of instances on each physical machine. The second one is with energy-efficient policy. To meet reliability demand, more than one instance may be provided for each service. In our experiments, we vary the reliability demand for one day from 0.9 to 0.99999, and compare the power consumption of the two experiments.

In Fig. 5, the blue bars represent the power consumption after optimization, and the green bars are the power consumption saved by optimization. Fig. 5 shows that with the energy-efficient policy, the power consumption can be greatly reduced. Besides, with the increase of reliability demand, the power consumption grows in both of the two experiments. In the real systems, in order to improve reliability, more redundancy should be provided, thus improving the energy consumption of the whole systems. Therefore, a trade-off between the reliability demand and energy consumption is practical in the design and implementation of web service systems.

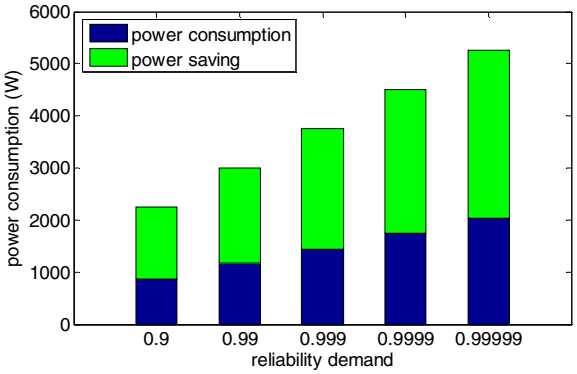

Figure 5. Power consumpiton and power saving using energy-efficient policy under different reliability demand

Considering the dynamics of service systems, a service can be registered and released dynamically [18]. The placement controller can well handle this situation, which is validated by Fig. 6.

At the very beginning, there are 3 services in the system with reliability demand 0.9, 0.99, 0.999 and the total power consumption after energy optimization is about 1200 W. After two days, a new service with the reliability demand 0.999 registers and joins the system. The proposed algorithm reruns and adjusts the placement. Fig. 6 shows that the total power consumption after optimization increases to about 1350 W. This runs for three days until the 6th day when a service with reliability demand 0.999 leaves the cluster. Then the algorithm reduces power consumption to about 1200 W. Two days later, a new service with reliability demand 0.9999 registers in the system and the power consumption increases. It can also be seen from Fig. 6 that the power consumption can be greatly reduced after optimization. Furthermore, with the number of services increasing, more power can be saved by using energy-efficient policy.

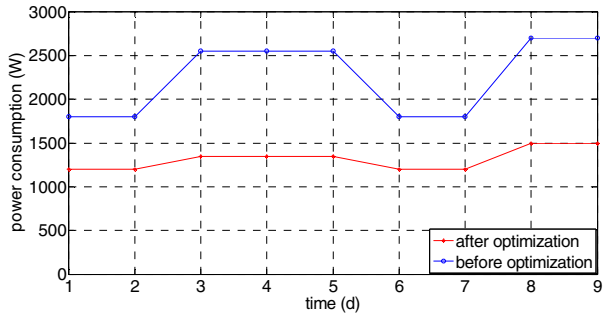

Figure 6. Power consumption with service registing and leaving

V. Related Work

The service provision and placement problem has been studied in literature. Tang et al. [12] proposed a placement controller to decide how to provide resources for web applications and where to place them. They consider resource constrains and load-balanced policy while striving to maximize total satisfied demand and resource utilization. In [19], the web service placement problem from the perspective of fairness was studied so as to treat them fairly; the authors used utility functions to model the satisfaction and sought to equalize the utility among different services while satisfying resource constraints. Li et al. [20] discussed how to deploy and manage the applications in the cluster by using performance model and treat it as a constraint so as to obtain the maximum profit out of the minimum resources. Their research studied how to maximize resource utilization, to obtain the maximal profit or how to fairly treat applications and services. Some pieces of work also studied performance, but were mainly focused on response time only [20], [24], [25].

Reliability is a critical attribute of Quality of Service (QoS) and is important in web service provision and placement. There has been some traditional research on reliability which studied reliability at the component and physical hardware level [4], [5]. Some of the methods were based on the structure of the system which consists of components that are series or in parallel, such as Reliability Block Diagrams (RBD) [4]. Other methods were based on State-based stochastic model which could deal with the dynamics of the system. In addition, there has been some study on the user-centric or service-oriented reliability [26]. Also, we have studied the service-oriented reliability by proposing a generic model from a top-down perspective in our previous work [27].

Besides, as energy consumption is becoming a serious concern, it receives much attention recently. Bartalos et al. [7] studied the modeling and estimating power consumption of web services and provided software services in context of locations to maximize energy efficiency. Ardagna et al. [23] studied how to deploy advanced active energy-aware business process applications by recognizing the multi-layer feedback nature of business process systems. They also discussed the problem of making trade-off between energy efficiency and performance. Steinder et al. [21] considered the energy consumption of machines as a function of CPU and present a combined power and performance management system to achieve significant power savings without unacceptable loss of performance. However, they only studied the response time as the metric of performance, and only a single bottleneck resource was considered. Guenter et al. [22] considered the multiple power states of servers and those power states could switch to each other. They proposed tradeoffs between cost, performance and reliability and studied when and how many servers to transition to each power state to meet demand while minimizing energy and reliability costs. But the reliability they studied was at the component and hardware level, and the reliability cost of components was due to duty-cycling which may have bad influence on the hardware.

VI. Conclusion

In this paper, we study the reliability-aware energy efficiency problem in service provision and placement, which is an important issue in the design and management of web service systems. We propose an optimization

framework which strives to minimize the energy consumption in web service systems while meeting the reliability demand. A continuous-time Markov chain is used to model the service-centric reliability. An approximation algorithm is proposed to minimize the energy consumption in the system. We compare the approximation solutions to the optimal solutions obtained from LINGO software to verify the accuracy of the algorithm. Experiment results show the effectiveness of our model and algorithm. This work is expected to offer a valuable reference for the deployment and management of web service systems.

There are several interesting directions for future work. Firstly, considering that there are several factors related to the energy consumption of a physical machine, one interesting problem is to build a more detailed energy consumption model according to different system states. The energy consumption could be evaluated more precisely. A second direction is to use multiple types of failures to model reliability, and different reliability probabilities of instances can also be considered. Thirdly, the overhead brought by adjusting the deployment settings can be studied, which would capture some other performance metrics. Finally, another interesting direction is to apply the models and algorithms in real web service systems and study some efficient parameter measurements to handle the high dynamics of SOA. The data set in real systems could provide a better understanding of the models and policies, which can serve as a guidance for system design and optimization.

ACKNOWLEDGMENT

This work is supported by the National Grand Fundamental Research 973 Program of China (No. 2010CB328105, No. 2009CB320504), and the National Natural Science Foundation of China (No. 60932003, No. 61020106002).

REFERENCES

[1] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," IEEE Transactions on Software Engineering, vol.30, no.5, May. 2004, pp. 311-327.

[2] B. Benatallah, M. Dumas, Q. Z. Sheng, A. H. H. Ngu, "Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services," Proc. 18th International Conference on Data Engineering, 2002, pp. 297-308.

[3] M. A. McCollough, L. L. Berry and M. S. Yadav, "Service failure and recovery in electronic retailing: An investigation of product-oriented and service-oriented transactions," Information Resources Management Journal, vol.22, no.3, 2009, pp. 1-15.

[4] K. S. Trivedi, "Probability and Statistics with Reliability, Queuing, and Computer Science Applications," second ed, New York: John Wiley and Sons, 2001.

[5] J. K. Muppala, M. Malhotra, and K. S. Trivedi, "Markov Dependability Models of Complex Systems: Analysis Techniques," Reliability and Maintenance of Complex Systems, S. Ozekici, ed., Germany: Springer, 1996, pp 442-486.

[6] A. Birolini, Reliability Engineering Theory and Practice, Fifth edition. Springer, 2007, pp. 2-10.

[7] P. Bartalos and M. B. Blake, "Green web services: Modeling and Estimating Power Consumption of Web Services," Proc. IEEE 19th International Conference on Web Services (ICWS 2012), Jun. 2012, pp. 178-185.

[8] W. Feng, X. Feng, and R. Ge, "Green Supercomputing Comes of Age," IT Professional, vol. 10, no. 1, 2008, pp. 17-23.

[9] J. G. Koomey, "Estimating regional power consumption by servers: A technical note", AMD Technical Study, 2007.

[10] J. M. Reason and J. M. Rabaey, "A study of energy consumption and reliability in a multi-hop sensor network," ACM SIGMOBILE Mobile Computing and Communications Review, vol.8, no.1, Jan. 2004, pp. 84 – 97.

[11] X. Qi, D. Zhu, and H. Aydin, "Global Reliability-Aware Power Management for Multiprocessor Real-Time Systems," Real-Time Systems, vol.47, no.2, Mar. 2011, pp. 109-142.

[12] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A Scalable Application Placement Controller for Enterprise Data Centers," 16th International World Wide Web Conference, 2007, pp. 331-340.

[13] M. Bladt, "a review on phase-type distribution and their use in risk theory," ASTIN Bulletin, vol.35, no.1, May. 2005, pp. 145-161.

[14] J. O. Kephart, H. Chan, R. Das, D. W. Levine, G. Tesauro, and F. R. an C. Lefurgy, "Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs," in IEEE Intl. Conf. on Autonomic Computing, Jun. 2006, pp. 145–154.

[15] C. Chekuri and S. Khanna. "On multi-dimensional packing problems". Annual ACM-SIAM Symposium on Discrete Algorithms, 1999, pp. 185-194.

[16] E.Coffman, J. Csirik, G. Woeginger, "Approximate Solutions to Bin-Packing Problems", in Handbook of Applied Optimization, P. Pardalos and M Resende, eds, Oxford University Press, 2002.

[17] Los Alamos National Laboratory, "All Systems Failure/Interrupt Data 1996-2005," http://institute.lanl.gov/data/fdata/.

[18] L. Zhang, J. Zhang, and H. Cai, "Services Computing," Springer, 2007.

[19] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguad´, "Utility-based Placement of Dynamic Web Applications with Fairness Goals" Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS 2008), 2008, pp. 9-16.

[20] J. Li, M. Woodside, J. Chinneck, M. Litoiu, and M. Litoiu, "Performance model driven QoS guarantees and optimization in clouds," Proc. 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD 2009), 2009, pp. 15-22.

[21] M. Steinder, I. Whalley, J. E. Hanson, and J. O. Kephart, "Coordinated Management of Power Usage and Runtime Performance," Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS 2008),2008, pp. 387-394.

[22] B. Guenter, N. Jain, and C. Williams, "Managing Cost, Performance, and Reliability Tradeoffs for Energy-Aware Server Provisioning," Proc. 2011 IEEE INFOCOM, 2011, pp. 1332-1340.

[23] D. Ardagna, C. Cappiello, M. Lovera, B. Pernici,and M. Tanelli, "Active Energy-Aware Management of Business-Process Based Applications Position Paper", Lecture Notes in Computer Science, vol. 5377, 2008, pp. 183-195.

[24] Y. Zhang, Z. Wang, B. Gao, C. Guo, W. Sun, and X. Li, "An Effective Heuristic for On-line Tenant Placement Problem in SaaS," Proc. IEEE 8th International Conference on Web Services (ICWS 2010),2010, pp. 425-432.

[25] A. M. V. Neves, C. A. L. Oliveira, R. M. F. Lima, and C. L. Sabat, "Computing Strategic Trade-Offs in Web Service Deployment and Selection," Proc. IEEE 19th International Conference on Web Services (ICWS 2012), Jun. 2012, pp. 210-217.

[26] A. Heddaya and A. Heldal, "Reliability, Availability, Dependability and Performability: A User-Centered View," Technical Report. Boston University. Computer Science Department, 1996.

[27] J. Huang, C. Lin, X. Kong, and Y. Zhu, "Modeling and Analysis of Dependability Attributes of Service Computing Systems," Proc. 2011 IEEE International Conference on Services Computing (SCC 2011), Jul. 2011, pp. 184-191.