Power Management Schemes for Heterogeneous Clusters under Quality of Service Requirements

Jian-Jia Chen
Department of Informatics
Karlsruhe Institute of Technology, Germany
j.chen@kit.edu

Kai Huang
Computer Engineering Group
ETH Zurich, Switzerland
khuang@tik.ee.ethz.ch

Lothar Thiele
Computer Engineering Group
ETH Zurich, Switzerland
thiele@tik.ee.ethz.ch

ABSTRACT

For modern computer systems, both performance and power consumption must be considered to reduce the maintenance cost for quality of service guarantees. This paper proposes efficient and effective power management schemes for heterogeneous clusters. Distinct from existing heuristic approaches, we propose power management schemes with approximation factor guarantees, compared to the optimal power management. Our greedy power management schemes have 1.5-approximation or 2-approximation guarantees depending on the complexity. We also propose dynamic-programming approach which can trade the quality of the resulting solutions with different time/space complexity. Simulation results wrt different power consumption models show that the proposed schemes are effective for the minimization of the power consumption for large scale clusters.

Categories and Subject Descriptors

D.4.8 [**Operating Systems**]: Performance—*quality of service guarantees, heterogeneous clusters*

General Terms

power management, soft real-time systems

Keywords

dynamic voltage scaling

1. INTRODUCTION

In the recent years, power and energy consumption has become key concerns in server clusters or data centers. For example, a high-performance server with 300Watt power consumption consumes 2628 kiloWatt hours. Therefore, within one year, the annual power cost of the server is around $263, if the electricity cost is $0.1 per kiloWatt hour. Even without considering the cost of the power delivery subsystems and the cooling facility, for maintaining a cluster with hundreds of servers, the electricity cost is significant. Another fact is that the performance per watt remains roughly flat over time [1], although advanced hardware technology has improved the performance per hardware dollar. As a result, the electricity cost of server clusters will be more than the hardware cost and become a major fraction of the total cost of ownership.

To reduce the power consumption without sacrificing the performance, power-aware and energy-efficient scheduling has been extensively explored in the literature.For homogeneous server clusters with identical servers, Chase et al. [2] develop a load balancing framework to dynamically turn on/off servers, Xu et al. [10] propose algorithms to determine the number of servers to turn on by applying both DVS and DPM, and Wierman et al. [9] explore how to balance the mean energy consumption and the mean response time under processor sharing scheduling.

Considering the popularity of heterogeneous clusters, power management for heterogeneous server clusters under quality of service (QoS) guarantees has been recently explored in [6, 8, 4]. Specifically, Wang and Lu [8] develop a power management algorithm to order heterogeneous servers in a pre-defined order. After deciding the activation and deactivation of servers, Lagrange Multiplier Method is applied to decide the execution frequency. Similarly, for servers with discrete speeds, Rusu et al. [6] use two tables for deciding which servers to be activated and which frequency levels to be executed. Moreover, Guerra et al. [4] model the problem as an integer linear programming (ILP) problem by applying ILP solvers with high complexity to get a table for storing decisions for different workloads under pre-defined granularity.

For most commercial computing systems, the available frequency levels are fixed. As a result, the approaches in [8] might not be suitable, in which using a higher available frequency level might sacrifice the optimality. Because storing decisions in tables requires exponential space complexity in the worst case, researchers in [4, 6] discretize the possible amount of the to-be-served workload into pre-defined granularity and derive the scheduling tables based on the granularity. As a result, the quality of the derived solution heavily depends on the granularity. Moreover, if a server in the cluster is out of service due to some maintenance reasons, recomputing the scheduling tables by applying algorithms in [4, 6] might be time-consuming.

This paper explores the power management problem for heterogeneous clusters under QoS constraints. Distinct

SAC'11 March 21-25, 2011, TaiChung, Taiwan.

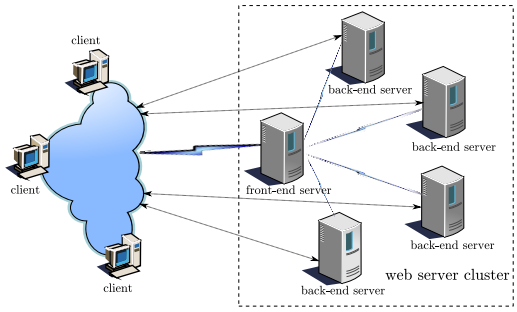


Figure 1: An example for a web server cluster

from the heuristic approaches in [4, 6, 8], we propose algorithms to provide different approximation guarantees for power consumption minimization under different time/space complexity. By considering systems with discrete frequency levels on servers, our schemes can be applied for general power consumption models and QoS models. Our greedy power management schemes have 1.5-approximation or 2-approximation guarantees depending on the complexity. Our dynamic-programming approach can trade the quality, in terms of power consumption, of the resulting solutions with the time/space complexity. As our approaches do not rely on global tables, they are more robust against failure of servers. If building a scheduling table is necessary for system designers, our proposed dynamic programming approach can be applied and extended to build tables with different granularity. Simulation results show that the proposed schemes are effective for minimizing the power consumption.

The rest of this paper is organized as follows: Section 2 provides system models and problem definition along with hardness analysis. Section 3 presents our greedy power management schemes with different approximation factor guarantees. Section 4 demonstrates how to use dynamic programming to trade the quality of the derived solution with the time/space complexity. Simulation results are presented in Section 5. Section 6 concludes this paper.

2. SYSTEM MODELS

2.1 System model

We consider a cluster with a front-end server, which arbitrarily distributes workload of requests to a cluster of back-end servers. The front-end server is assumed not to participate in processing any requests, but only to decide the power states of back-end servers and how to distribute the requests to those back-end servers that are activated. Figure 1 illustrates an example for a cluster of web servers. The cluster consists of M heterogeneous back-end servers, denoted by $m_1, m_2, \ldots, m_M$, providing CPU-bounded services. The heterogeneity comes from different hardware architectures, different manufacturing techniques, different vendors, etc. However, all these M back-end servers have the same functionality, i.e., a request can be served in any of these back-end servers. As a result, once a back-end server is activated (turned on), it can serve any assigned request.

To satisfy the performance requirement, the cluster has to provide its services under some quality of service (QoS) constraint. Due to heterogeneity of back-end servers, the performance of servers m_i and m_j at frequency f might be different. To have the same performance index under different servers, each server m_i is associated with a performance co-efficient α_i such that $\alpha_i f$ is the throughput (in terms of executed number of cycles or requests per time unit) of server m_i at frequency f.

To measure quality of servers, we might either apply analysis for soft real-time systems for the (average) percentage of requests that miss their timing constraints [6], or apply Queueing Theory to guarantee the average response time, e.g., M/M/1 model in [8] or M/G/1 PS model in [9]. For example, as shown in [8], if the quality of service is on the average response time under the M/M/1 queuing model, to serve workload with average request rate λ on server m_i, the average response time at frequency f is $\frac{1}{\alpha_i f - \lambda}$, where $\alpha_i f$ is the number of requests finished per time unit. For the rest of this paper, suppose that $q_i(f, \lambda)$ is the quality of service provided by server m_i at frequency f when the average request rate assigned on server m_i is with arrival request rate λ. Note that, the metric of $q_i(f, \lambda)$ depends on the definition of the QoS of a server.

The derivation of $q_i(f, \lambda)$ is not a focus of this paper. One might apply existing results in the literature, e.g., [6, 9, 8]. We only assume that $q_i(f, \lambda)$ is not worse than $q_i(f - \eta, \lambda)$ for any $\eta > 0$. In other words, this paper focuses on a more general setting, in which *for a fixed average request rate assigned to a server, the QoS provided by the server is not worse when the server is operated at a higher frequency.*

2.2 Power consumption and DVS models

We consider servers with discrete dynamic voltage scaling levels. The number of available frequency levels on server m_i is K_i. For brevity, we order the available frequency levels on server m_i from the lowest one to the highest. Let $f_{i,j}$ be the j-th lowest available frequency for server m_i, in which $f_{i,1} < f_{i,2} < \cdots < f_{i,K_i}$ when $K_i > 1$. The power consumption for server m_i on frequency $f_{i,j}$ is $P^{\dagger}_{i,j}$.

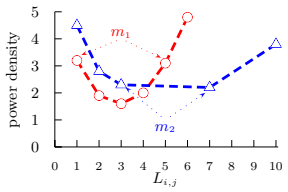
When a server m_i is activated, it must operate at least at frequency $f_{i,1}$. However, if it is not necessary to turn a server on, we can deactivated the server (if it is activated) to reduce the power consumption. If a server m_i is not activated, its power consumption is assumed to be a constant P^{δ}_i. If the power consumption at frequency $f_{i,j}$ is less than $f_{i,k}$ for some $k < j$, we can simply remove the power-inefficient frequency $f_{i,k}$. Therefore, we consider systems with $P^{\delta}_i < P^{\dagger}_{i,1} < P^{\dagger}_{i,2} < \cdots < P^{\dagger}_{i,K_i}$. As we cannot reduce the power consumption P^{δ}_i, it can be subtracted from the power consumption $P^{\dagger}_{i,j}$. For the rest of this paper, we will only focus on the *manageable power consumption* $P_{i,j}$ of server m_i at frequency $f_{i,j}$, in which $P_{i,j}$ is $P^{\dagger}_{i,j} - P^{\delta}_i$.

When a server is activated for execution, both efficiency and power consumption issues must be considered. For example, if a server has high power consumption, activating the server might consume too much power even though we might not have to activate the other servers. On the other hand, if a server has lower power consumption, activating the server might not be enough, and, hence, we might have to activate many servers to satisfy the QoS requirements.

As a result, we have to consider the *power density* of a server, which is defined as the power consumption of the server divided by the request rate it can serve under the QoS requirement, denoted by R, of the cluster. Note that as we focus on general settings of QoS requirement, R could be average response time, average waiting time, or worst-case response time, etc. Suppose that $L_{i,j}(R)$ is the (average)

j	$L_{1,j}$	$P_{1,j}$		$L_{2,j}$	$P_{2,j}$
1	1	3.2		1	4.5
2	2	3.8		2	5.6
3	3	4.8		3	6.9
4	4	8		7	15.4
5	5	15.5		10	38
6	6	28.8			

(a) power profiles



(b) power density

Figure 2: An example for power consumption and power density of servers.

request arrival rate that server m_i can serve at frequency $f_{i,j}$ with quality of service no worse than R. That is, $q_i(f_{i,j}, L_{i,j}(R))$ is not worse than R. For example, if the quality of service is the average response time in the M/M/1 queuing model, $q_i(f_{i,j}, L_{i,j}(R)) = \frac{1}{\alpha_i f_{i,j} - L_{i,j}(R)} \leq R$. For notational brevity, for the rest of this paper, we will define both $P_{i,0}$ and $L_{i,0}$ as 0. As R is assumed to be a fixed parameter, we will use $L_{i,j}$ to represent $L_{i,j}(R)$ for the rest of this paper. Figure 2 illustrates an example for the power consumption and the power density, in which the power consumption is an increasing function of the average request arrival rate but the power density is not.

Furthermore, we assume that the front-end server is responsible for estimating the average request rate for the next time interval for scheduling, and for distributing the requests to back-end servers such that the quality of service is satisfied and the power consumption is minimized. Throughout this paper, we assume that the workload prediction is known a priori, in which the average request rate of the cluster is Λ, and our task is to decide the activation and frequency levels of back-end servers to minimize the power consumption under the QoS constraint.

2.3 Problem definition

This work explores power management for a heterogeneous cluster under the quality of service requirement R. As the average request rate (workload) changes over time, the power management must be done dynamically to cope with dynamic rates. Suppose that the current average request rate of the cluster is Λ. Our objective is a) to activate/deactivate servers to distribute the average request rate to those servers that are activated for execution, and b) to decide the operation frequency of activated servers, such that the power consumption is minimized and the QoS requirement R is satisfied. For brevity, we denote the above problem as the *Power Management for hEterogeneous server Clusters* (PMEC) problem.

A solution S to the PMEC problem distributes Λ into $\lambda_1(S), \lambda_2(S), \ldots, \lambda_M(S)$ and decides the frequency levels $s_1(S)$, $s_2(S)$, ..., $s_M(S)$ of these M back-end servers. A solution S is said *feasible* for the PMEC problem if the rate distribution is no less than Λ, i.e., $\sum_{i=1}^{M} \lambda_i(S) \geq \Lambda$, and the frequency level on server m_i is sufficient to provide the quality of service requirement, i.e., $\lambda_i(S) \leq L_{i,s_i(S)}$. For brevity, we denote the power consumption of a solution S by $\Phi(S)$, in which $\Phi(S) = \sum_{i=1}^{M} P_{i,s_i(S)}$. A solution S is said *optimal* for the PMEC problem if its power consumption is the minimum among the feasible solutions. For a solution, if $\sum_{i=1}^{M} L_{i,s_i(S)}$ is no less than Λ, we can easily distribute $L_{i,s_i(S)}$ average request rate to server m_i when $s_i > 0$ without violating the QoS guarantee. Therefore, for the rest of this paper, we only focus our discussions on how to decide the frequency levels of servers to guarantee that the solution S satisfies $L_{i,s_i(S)} \geq \Lambda$ with adoption of the above request distribution strategy.

If activating all the back-end servers at the highest frequency level cannot satisfy the QoS requirement, one has to augment the back-end servers, and there does not exist any feasible solution for the PMEC problem. For the rest of this paper, we focus on the optimality issue by considering cases with $\sum_{i=1}^{M} L_{i,K_i} \geq \Lambda$. Obviously, the studied problem is $\mathcal{NP}$-hard, as it can be reduced to the Knapsack problem. Due to the $\mathcal{NP}$-hardness of the PMEC problem, this paper pursues polynomial-time approximation algorithms with worst-case guarantees on the quality of the derived solutions. A ρ-approximation algorithm for the PMEC problem (or, an algorithm with a ρ-approximation factor) guarantees to derive solutions with at most ρ times of the power consumption of the corresponding optimal solutions [7].

3. GREEDY SCHEMES

This section presents our proposed greedy power management schemes for the PMEC problem. We will first present the construction of a *decision tree* to decide whether we should turn on a server, or accelerate or decelerate the execution frequency of a server. Then, we will present our proposed greedy power management schemes with a 2-approximation factor, followed by improved schemes.

3.1 Constructing Decision Trees

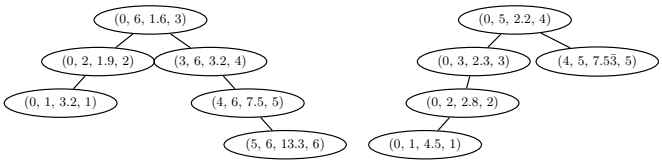
Before presenting our proposed greedy power management schemes, we will first describe the construction of a decision tree for a server m_i. Our proposed greedy power management schemes will decide whether we should activate a server for serving requests or accelerate a server for accommodating more request rate.

For the decision tree $\mathcal{T}_i$ of server m_i, suppose that each vertex v of the tree has the following fields: *density*, *index*, *start*, *end*, *right*, and $left$, in which *density* is the increased power density of this selection, *index* is the index of frequency level this vertex represents for, *start* (*end*, respectively) is the low bound of the frequency level (highest frequency level, respectively) used for the subtree rooted by v, $left$ (*right*, respectively) is the left-hand child (right-hand child, respectively) of vertex v. For clarity, we will use $density(v)$, $index(v)$, $start(v)$, $end(v)$, $right(v)$, and $left(v)$ to present the corresponding values.

We use Algorithm 1, i.e., calling Algorithm DT$(m_i, 0, K_i)$, to construct $\mathcal{T}_i$. In the specified range between frequency level indexes a and b given as part of input parameters of Algorithm DT, we find the index j^* such that the increased power density by operating at frequency f_{i,j^*} is the minimum, in which the increased power density at frequency $f_{i,j}$ is defined as $\frac{P_{i,j} - P_{i,a}}{L_{i,j} - L_{i,a}}$. Then, for the vertex (root of a sub-tree), we set $density(v)$ to $\frac{P_{i,j^*} - P_{i,a}}{L_{i,j^*} - L_{i,a}}$, $index(v)$ to j^*, $start(v)$ to a, $end(v)$ to b, the right child by calling DT(m_i, j^*, b) recursively, and the left child by calling DT$(m_i, a, j^* - 1)$ recursively. If Algorithm DT is called with $a \leq b$, this is a termination condition, in which we just simply return a null pointer. The construction of the decision tree $\mathcal{T}_i$ for server m_i takes $O(K_i^2)$ time complexity and has $O(K_i)$ space complexity. Note that the decision

Algorithm 1 DT

Input: (m_i, a, b);
Output: a decision tree in the feasible range $[a, b]$ for server m_i;
1: **if** $a \leq b$ **then**
2: return **null**;
3: **end if**
4: construct a vertex v;
5: $j^* \leftarrow \arg\min_{a<j\leq b} \frac{P_{i,j}-P_{i,a}}{L_{i,j}-L_{i,a}}$; (break ties arbitrarily)
6: $density(v) \leftarrow \frac{P_{i,j^*}-P_{i,a}}{L_{i,j^*}-L_{i,a}}$;
7: $index(v) \leftarrow j^*, start(v) \leftarrow a, end(v) \leftarrow b$;
8: $right(v) \leftarrow DT(m_i, j^*, b)$;
9: $left(v) \leftarrow DT(m_i, a, j^* - 1)$;
10: return v;



(a) $\mathcal{T}_1$ (b) $\mathcal{T}_2$

Figure 3: An example for the decision tree of the example in Figure 1, where the numbers on a vertex denote its fields $start, end, density$, and $index$ accordingly.

trees of servers are constructed only once in off line.

Figure 3 illustrates an example for the decision tree for the power consumption of a server presented in Figure 2. Note that, if the power consumption is a convex function of average request rate as the case in Figure 2, the resulting decision tree is skew. Based on the definition of the decision, we will have the following lemmas.

LEMMA 1. *Given a vertex v in the decision tree $\mathcal{T}_i$ of server m_i. For any index j with $start(v) < j \leq index(v)$,*

$$P_{i,j} + density(v)(L_{i,index(v)} - L_{i,j}) \geq P_{i,index(v)}.$$

PROOF. By the definition of the decision tree, we have $\frac{P_{i,index(v)}-P_{i,start(v)}}{L_{i,index(v)}-L_{i,start(v)}} = density(v)$ and $\frac{P_{i,j}-P_{i,start(v)}}{L_{i,j}-L_{i,start(v)}} \geq density(v)$. This implies $\frac{P_{i,index(v)}-P_{i,j}}{L_{i,index(v)}-L_{i,j}} \leq density(v)$. Therefore, $P_{i,index(v)} = P_{i,j} + P_{i,index(v)} - P_{i,j} \leq P_{i,j} + density(v)(L_{i,index(v)} - L_{i,j})$. □

LEMMA 2. *Given a vertex v in the decision tree $\mathcal{T}_i$ of server m_i. For any index j with $index(v) < j \leq end(v)$,*

$$\frac{P_{i,j} - P_{i,index(v)}}{L_{i,j} - L_{i,index(v)}} \geq \frac{P_{i,j} - P_{i,start(v)}}{L_{i,j} - L_{i,start(v)}}.$$

PROOF. Suppose that $\frac{P_{i,j}-P_{i,index(v)}}{L_{i,j}-L_{i,index(v)}} < \frac{P_{i,j}-P_{i,start(v)}}{L_{i,j}-L_{i,start(v)}}$ for contradiction. This will lead to the conclusion that $\frac{P_{i,index(v)}-P_{i,start(v)}}{L_{i,index(v)}-L_{i,start(v)}} > \frac{P_{i,j}-P_{i,start(v)}}{L_{i,j}-L_{i,start(v)}}$, which contradicts the construction of the decision tree. □

3.2 Algorithm Greedy

By adopting the decision tree, we propose a greedy algorithm, denoted as Algorithm Greedy, to decide the activation and operation frequencies of servers. The pseudo-code of Algorithm Greedy is presented in Algorithm 2. The basic idea is to try to accommodate more request rate with the smallest increased power density. We will start from the case that none of the servers is activated at the beginning,

Algorithm 2 Greedy

Input: average request arrival rate Λ, M servers with decision trees $\mathcal{T}_1, \ldots, \mathcal{T}_M$, QoS requirement R;
Output: a feasible solution under QoS requirement R;
1: **if** $\Lambda > \sum_{i=1}^{M} L_{i,K_i}$ **then**
2: return "no feasible solution";
3: **end if**
4: let $S^\dagger$ be the solution by activating all servers at their highest frequency levels;
5: set s'_i to 0 and v_i to the root of tree $\mathcal{T}_i$;
6: $\ell \leftarrow 0$;
7: **while** there exists $v_i \neq$ null **do**
8: $i^* \leftarrow \arg\min_{1\leq i\leq M \text{ and } v_i\neq \text{null}} density(v_i)$;
9: $q_{i^*} \leftarrow L_{i^*,index(v_{i^*})} - L_{i^*,start(v_{i^*})}$;
10: **if** $\ell + q_{i^*} < \Lambda$ **then**
11: $\ell \leftarrow \ell + q_{i^*}$;
12: $s'_{i^*} \leftarrow index(v_{i^*})$;
13: $v_{i^*} \leftarrow right(v_{i^*})$;
14: **else**
15: let S' be the solution by setting server m_{i^*} at frequency level $index(v_{i^*})$ and the others servers m_i at frequency level s'_i;
16: **if** S' has less power consumption than $S^\dagger$ **then**
17: $S^\dagger \leftarrow S'$;
18: **end if**
19: $v_{i^*} \leftarrow left(v_{i^*})$;
20: **end if**
21: **end while**
22: return $S^\dagger$ as the solution;

and, in each step, we try to activate or accelerate a server at a power-efficient frequency level by using the given decision trees of servers.

Initially, the frequency level s'_i of server m_i is set to 0, and the vertex v_i of server m_i in the decision tree is set to the root of decision tree $\mathcal{T}_i$. The variable ℓ is used to record the amount of total request rate served by executing at frequency level s'_i on server m_i, i.e., ℓ is $\sum_{i=1}^{M} L_{i,s'_i}$. While there exists some non-null v_i, we enter the loop to activate or accelerate a server m_{i^*}, in which the increased power density is the smallest in Step 8 in Algorithm 2. Clearly, if we increase s'_{i^*} to $index(v_{i^*})$, we will set increase the served request rate by $L_{i^*,index(v_{i^*})} - L_{i^*,(v_{i^*})}$, abbreviated by q_{i^*}. If $\ell + q_{i^*}$ is less than Λ, we set ℓ to $\ell + q_{i^*}$, s'_{i^*} to $index(v_{i^*})$, and v_{i^*} to $right(v_{i^*})$. Otherwise, we know that the solution S' by setting server m_{i^*} at frequency level $index(v_{i^*})$ and the others servers m_i at frequency level s'_i is a feasible solution for the PMEC problem. If S' is better than the best solution $S^\dagger$ so far, we replace the best solution $S^\dagger$ with S'. Moreover, we also have to set v_{i^*} to $left(v_{i^*})$ for the case $\ell + q_{i^*} \geq \ell$. The solution $S^\dagger$ is then returned when all v_is are null for all servers m_is.

The time complexity of Algorithm Greedy is $O(M \sum_{i=1}^{M} K_i)$, since each iteration in the `while` loop in Algorithm 2 takes $O(M)$ time and there are at most $O(\sum_{i=1}^{M} K_i)$ iterations.

Take the input instance in Figure 1 for example. Suppose that Λ is 9. For the first iteration in the loop of Algorithm Greedy, we will greedily choose i^* as 1 by updating ℓ to 3 and s_1 to 3. The second iteration chooses i^* as 2, and then $\ell + q_2 = 10 > \Lambda$. Therefore, we have a solution S' with $(s_1, s_2) = (3, 4)$ and 20.2 power consumption by updating v_2 to the left child of the root of decision tree $\mathcal{T}_2$. In the third iteration, we will then choose i^* as 2 again by setting s_2^* to 3 and ℓ to 6. For the next iterations, the algorithm goes to the rightmost child of decision tree $\mathcal{T}_1$, and then $\ell + q_{i^*} = 9$,

where S' in this case is with power consumption 35.7. As a result, Algorithm Greedy will return the solution $S^\dagger$ with $(s_1, s_2) = (3, 4)$ for this example.

3.3 Analysis of Algorithm Greedy

Based on Algorithm Greedy, we have the following lemma for feasible solutions.

LEMMA 3. *For any feasible solution, there must be at least one activated server m_i with frequency level higher than s'_i.*

This lemma comes from the definition of s'_i in Algorithm Greedy where we can guarantee that $\sum_{i=1}^{M} L_{i,s'_i} < \Lambda$ at any moment.

For the optimal solution S^* (it exists but is unknown), suppose that s^*_i is the assigned frequency level of server m_i. Note that if server m_i is not activated for serving requests in S^*, then s^*_i is set as 0. By Lemma 3, there must be at least one faster server m_i in solution S^*, in which $s^*_i > s'_i$.

We now analyze the power consumption of the derived solution of Algorithm Greedy, compared to the power consumption of solution S^*. We first decompose optimal solution S^* by running Algorithm Greedy in the loop between Step 7 and Step 21 in Algorithm 2 as follows:

- If the condition $\ell + q_{i^*} < \Lambda$ in Step 10 in Algorithm 2 is false and $s^*_i \geq index(v_{i^*}) > s'_{i^*}$, let this server be m_{k^*} and break the loop before Step 19 in Algorithm 2.
- Let $s^\flat_i$ ($v^\flat_i$, respectively) be the frequency level s'_i (v_i, respectively) before breaking the loop.
- Let D^* be $density(v_{i^*})$, which is the increased power density when we break the loop.

Let $S^\flat$ be the solution by activating server m_i at frequency levels $s^\flat_i$ with average request rate $L_{i,s^\flat_i}$. Moreover, let $S^\sharp$ be the solution by activating server m_{k^*} at frequency level $index(v_{k^*})$ with average request rate $L_{k^*,index(v_{k^*})}$ and the other servers m_is at frequency levels $s^\flat_i$ with average request rate $L_{i,s^\flat_i}$. For brevity, let $s^\sharp_i$ ($s^\flat_i$, respectively) be the frequency level of server m_i in solution $S^\sharp$ ($S^\flat$, respectively).

By the definition of $S^\sharp$ and $S^\flat$, we know that

$$\Phi(S^\sharp) - \Phi(S^\flat) \leq \Phi(S^*). \tag{1}$$

We use the example in Figure 2 for demonstrating how to construct $S^\flat$ and $S^\sharp$. Suppose that S^* is with $(s^*_1, s^*_2) = (2, 4)$. For constructing $S^\sharp$ and $S^\flat$, we have the situation that $\ell + q_{i^*} \geq \Lambda$ in the second iteration of the loop, and then we know that $s^*_2 = 4 \geq index(v_{2^*}) = 4 \geq s'_2 = 4$. Therefore, solution $S^\sharp$ is with $(s^\sharp_1, s^\sharp_2) = (3, 4)$ and solution $S^\flat$ is with $(s^\flat_1, s^\flat_2) = (3, 0)$.

LEMMA 4. *Solution $S^\sharp$ is a feasible solution for the PMEC problem, and the power consumption $\Phi(S^\sharp)$ is no less than the power consumption $\Phi(S^\dagger)$ of the solution $S^\dagger$ derived from Algorithm Greedy.*

PROOF. By definition, we know that $\sum_{i=1}^{M} L_{i,s^\flat_i} < \Lambda$ and $L_{k^*,index(v_{k^*})} + \sum_{i=1,\ldots,M \text{ and } i\neq k^*} L_{i,s^\sharp_i} \geq \Lambda$. Moreover, solution $S^\sharp$ is the same as solution S' in Step 15 in Algorithm 2, *if we do not break the loop* (which is the case of Algorithm Greedy). Therefore, we also know that $\Phi(S^\dagger) \leq \Phi(S^\sharp)$. □

Based on Lemma 4, to show the 2-approximation factor of Algorithm Greedy, we will simply show

$$\Phi(S^\sharp) \leq 2\Phi(S^*). \tag{2}$$

By the feasibility of solution S^* and infeasibility of solution $S^\flat$, we have the following lemma.

LEMMA 5. $\sum_{i=1}^{M} L_{i,s^*_i} \geq \Lambda > \sum_{i=1}^{M} L_{i,s^\flat_i}$.

PROOF. By the feasibility condition of solution S^*, we know that $\sum_{i=1}^{M} L_{i,s^*_i} \geq \Lambda$. As solution $S^\flat$ is infeasible, we also know that $\Lambda > \sum_{i=1}^{M} L_{i,s^\flat_i}$. □

By comparing solutions $S^\flat$ and S^*, we divide these M back-end servers into two sets K_1 and K_2, in which

$$K_1 \leftarrow \left\{ m_i \mid s^*_i < s^\flat_i \right\}, \tag{3a}$$

$$K_2 \leftarrow \left\{ m_i \mid s^\flat_i \leq s^*_i \right\}. \tag{3b}$$

For sets K_1 and K_2, the following lemmas show important properties resulting from the decision trees.

LEMMA 6. *For any server m_i in set K_1, we have*

$$P_{i,s^\flat_i} \leq P_{i,s^*_i} + D^* \left(L_{i,s^\flat_i} - L_{i,s^*_i} \right).$$

PROOF. Suppose that $v^\flat_i$ (v^*_i, respectively) is the vertex in the decision tree $\mathcal{T}_i$ in which $index(v^\flat_i)$ ($index(v^*_i)$, respectively) is $s^\flat_i$ (s^*_i, respectively). By the definition of solution $S^\flat$, we have $D^* \geq density(v^\flat_i)$. Then we have three cases:

- v^*_i is in the subtree of $v^\flat_i$: By Lemma 1 and $D^* \geq density(v^\flat_i)$, we know that the statement stands.
- $v^\flat_i$ is in the subtree of v^*_i: By Lemma 2 and $D^* \geq density(v^\flat_i)$, we know that $D^* \geq density(v^\flat_i) \geq \frac{P_{i,s^\flat_i} - P_{i,s^*_i}}{L_{i,s^\flat_i} - L_{i,s^*_i}}$, which proves the statement.
- v^*_i and $v^\flat_i$ are in the left and right subtree of some vertex V in $\mathcal{T}_i$: By combining the above two cases, we know that $P_{i,s^\flat_i} \leq P_{i,index(v)} + D^*(L_{i,s^\flat_i} - L_{i,index(v)})$ and $P_{i,index(v)} \leq P_{i,s^*_i} + D^*(L_{i,index(v)} - L_{i,s^*_i})$. Hence, the statement also holds for this case.

□

LEMMA 7. *For any server m_i in set K_2, we have*

$$P_{i,s^*_i} \geq P_{i,s^\flat_i} + D^* \left(L_{i,s^*_i} - L_{i,s^\flat_i} \right).$$

PROOF. We use the same notations used in the proof of Lemma 6. By the definition of solution $S^\flat$, we only have the case, where v^*_i is in the subtree of $v^\flat_i$. Moreover, we know that $D^* \leq \frac{P_{i,j} - P_{i,s^\flat_i}}{L_{i,j} - L_{i,s^\flat_i}}$ for any index j with $index(v^\flat_i) < j \leq end(v^\flat_i)$. As v^*_i is in the subtree of $v^\flat_i$, we only have $index(v^\flat_i) = s^\flat_i \leq s^*_i \leq end(v^\flat_i)$. Clearly, the statement holds for both $s^\flat_i = s^*_i$ and $s^\flat_i \leq s^*_i$. □

Based on the above lemmas, we show the approximation factor of Algorithm Greedy in the following theorem.

THEOREM 1. *Algorithm Greedy is a polynomial-time 2-approximation algorithm for the PMEC problem, provided that all $L_{i,j}$s on server m_i at frequency $f_{i,j}$ are given.*

PROOF. We will first prove $\Phi(S^\flat) \leq \Phi(S^*)$. By Lemma 6, we know that

$$\sum_{m_i \in K_1} P_{i,s^\flat_i} \leq \sum_{m_i \in K_1} P_{i,s^*_i} + D^*(L_{i,s^\flat_i} - L_{i,s^*_i}) \tag{4}$$

Based on Lemma 5, we have

$$\begin{aligned} \sum_{m_i \in K_1} L_{i,s^\flat_i} - L_{i,s^*_i} &< \Lambda - \sum_{m_i \in K_2} L_{i,s^\flat_i} - \sum_{m_i \in K_1} L_{i,s^*_i} \\ &\leq \sum_{m_i \in K_2} L_{i,s^*_i} - \sum_{m_i \in K_2} L_{i,s^\flat_i}. \end{aligned} \tag{5}$$

Algorithm 3 E-Greedy

Input: average request arrival rate Λ, M servers with decision trees $\mathcal{T}_1, \ldots, \mathcal{T}_M$, QoS requirement R;
Output: a feasible solution under QoS requirement R;

let $\hat{S}$ be the solution derived from Greedy;
for $i \leftarrow 1; i \leq M; i \leftarrow i+1$ **do**
  for $j \leftarrow 1; j \leq K_i; j \leftarrow j+1$ **do**
    let S' be the solution by activating server m_i at frequency $f_{i,j}$ and the other servers by calling Algorithm Greedy with arrival rates $\Lambda - L_{i,j}$;
    if S' is feasible and $\Phi(S') < \Phi(\hat{S})$ **then**
      $\hat{S} \leftarrow S'$;
    end if
  end for
end for
return $\hat{S}$ as the solution;

As a result, by Lemma 7, we know

$$\Phi(S^\flat) < \sum_{m_i \in K_1} P_{i,s_i^*} + \sum_{m_i \in K_2} (P_{i,s_i^\flat} + D^*(L_{i,s_i^*} - L_{i,s_i^\flat})) \leq \sum_{m_i \in K_1} P_{i,s_i^*} + \sum_{m_i \in K_2} P_{i,s_i^*} = \Phi(S^*). \quad (6)$$

Therefore, based on Equation (6) and Equation (1), we know that $\Phi(S^\sharp) \leq 2\Phi(S^*)$, which proves the 2-approximation factor of Algorithm Greedy.

Because the time complexity of Algorithm Greedy is $O(M\sum_{i=1}^{M} K_i)$, we reach the conclusion that Algorithm Greedy is a polynomial-time 2-approximation algorithm for the PMEC problem. □

3.4 Algorithm E-Greedy

Based on the 2-approximation of Algorithm Greedy, we are going to present an improved greedy algorithm, called Algorithm E-Greedy. The approach is to force a server m_i to run at a specified frequency $f_{i,j}$, and then the rest $M-1$ servers are used to serve the rest $\Lambda - L_{i,j}$ request rate. Among all (at most $\sum_{i=1}^{M} K_i$ feasible) solutions under the above restriction, we return the best one. The algorithm is illustrated in Algorithm 3.

THEOREM 2. *Algorithm E-Greedy is a polynomial-time* 1.5*-approximation algorithm for the PMEC problem, provided that all $L_{i,j}$s on server m_i at frequency $f_{i,j}$ are given.*

PROOF. The time complexity is $O(M(\sum_{i=1}^{M} K_i)^2)$. We now focus on the approximation factor. Again, let solutions $S^\dagger$, S^*, $S^\flat$, and $S^\sharp$ as defined in Section 3.3. Consider the solution S' by restricting server m_{k^*} running at frequency level $s^\sharp_{k^*}$, where k^* is defined while constructing solution $S^\sharp$. Let π be $P_{k^*,s^\sharp_{k^*}}$. Here are two cases:

- $\pi < \frac{1}{2}\Phi(S^*)$: Along with Equation (6) and $\pi \geq \Phi(S^\sharp) - \Phi(S^\flat)$, we have
$$\Phi(S^\dagger) \leq \Phi(S^\flat) + \pi < 1.5\Phi(S^*).$$
- $\pi \geq \frac{1}{2}\Phi(S^*)$: We know that $\Phi(S^*) - \pi$ is the optimal power consumption for using the $M-1$ servers except m_{k^*} for serving $\Lambda - L_{k^\sharp,s^\sharp_i}$. Hence, due to the 2-approximation factor of Algorithm Greedy, we have
$$\Phi(S') \leq \pi + 2(\Phi(S^*) - \pi) \leq 1.5\Phi(S^*).$$

Since $\Phi(\hat{S})$ is less than or equal to the above two cases, the theorem is proved. □

Algorithm 4 DP

Input: ϵ, average request arrival rate Λ, M servers, QoS requirement R, solution of Algorithm Greedy $\Psi(S^\dagger)$;
Output: a feasible solution under QoS requirement R;

$P^\flat_{i,j} \leftarrow \left\lfloor \frac{2MP_{i,j}}{\epsilon\Phi(S^\dagger)} \right\rfloor \ \forall 1 \leq i \leq M, 1 \leq j \leq K_i$;
for $p \leftarrow 0; ; p \leftarrow p+1$ **do**
  for $i \leftarrow 1; i \leq M; i \leftarrow i+1$ **do**
    derive $\Psi_i(p)$ by Equation (9) and Equation (10);
  end for
  if $\Psi_M(p) \geq \Lambda$ **then**
    $P' \leftarrow p$;
    back-track the dynamic programming entries from $\Psi_M(P')$ to find the solution S^ϵ contributing to$\Psi_M(P')$;
    return solution S^ϵ;
  end if
end for

4. DYNAMIC PROGRAMMING

This section provides a fully polynomial-time approximation scheme (FPTAS) for the PMEC problem by applying dynamic programming. An FPTAS for the PMEC problem is a $(1+\epsilon)$-approximation algorithm with polynomial-time complexity by treating $\frac{1}{\epsilon}$ as an input parameter for any positive ϵ. Unless $\mathcal{NP} = \mathcal{P}$, fully polynomial-time approximation schemes are the best in terms of polynomial-time approximation algorithms with worst-case guarantees.

Suppose that $\Phi(S^\dagger)$ is power consumption of the solution derived by applying Algorithm Greedy in Section 3. To derive (more precise) approximated solution, we first derived the rounded power consumption $P^\flat_{i,j}$ as follows:

$$P^\flat_{i,j} = \left\lfloor \frac{2MP_{i,j}}{\epsilon\Phi(S^\dagger)} \right\rfloor, \quad (7)$$

where ϵ is a user-specified parameter for the tolerable approximation factor. Then, we perform dynamic programming based on the rounded power consumption. Suppose that $\Psi_i(p)$ is the maximum average request rate that can be served by using only servers $m_1, m_2, \ldots, m_i$ with rounded power consumption no more than p. Hence, for brevity, for $1 \leq i \leq M$, we define

$$\Psi_i(p) = -\infty \text{ when } p < 0. \quad (8)$$

Suppose that j_p is the frequency level j with $P^\flat_{1,j} \leq p < P^\flat_{1,j+1}$ for $j < K_1$. Furthermore, when p is no less than K_1, let j_p be K_1. The boundary condition of $\Psi_1(p)$ for $p \geq 0$ is:

$$\Psi_1(p) = L_{i,j_p}. \quad (9)$$

Then, for $i \geq 2$, the value of $\Psi_k(p)$ can be calculated by the following recursive function:

$$\Psi_i(p) = \max_{j=0}^{K_i} \left\{ \Psi_{i-1}(p - P^\flat_{i,j}) + L_{i,j} \right\}. \quad (10)$$

Suppose that P' is the minimum value with $\Psi_M(P') \geq \Lambda$. By back-tracking the dynamic programming table, we can derive a solution S^ϵ with $\sum_{i=1}^{M} P^\flat_{i,s^\epsilon_i} = P'$ and $\sum_{i=1}^{M} L_{i,s^\epsilon_i} \geq \Lambda$, in which the frequency level on server m_i in the solution is s^ϵ_i. Algorithm 4 presents the dynamic programming, denoted by Algorithm DP, in which the detail for back-tracking is omitted due to space limitation.

The following theorem shows that the quality of the derived solution S^ϵ from the above dynamic programming is not too far away from the optimum, even in the worse case.

THEOREM 3. *Deriving S^ϵ takes $O\left(\frac{MK_{\max}}{\epsilon} + MK_{\max}\right)$ time complexity and $O\left(\frac{M}{\epsilon} + M\right)$ space complexity, where $K_{\max}$ is $\max_{i=1,2,\ldots,M} K_i$. For any input instance with feasible*

solution S^*,

$$\Phi(S^\epsilon) \leq (1+\epsilon)\Phi(S^*).$$

PROOF. By the optimality of the dynamic programming, it is not difficult to see that

$$\sum_{i=1}^{M} P^\flat_{i,s^\epsilon_i} \leq \sum_{i=1}^{M} P^\flat_{i,s^*_i}. \quad (11)$$

Then, by Equation (7) and Equation (11), we have

$$\sum_{i=1}^{M} \frac{2MP_{i,s^\flat_i}}{\epsilon\Psi(S^\dagger)} \leq M + \sum_{i=1}^{M} \frac{2MP_{i,s^*_u}}{\epsilon\Psi(S^\dagger)} \quad (12)$$

$$\Rightarrow \Phi(S^\epsilon) \leq \frac{\epsilon\Phi(S^\dagger)}{2} + \Phi(S^*) \leq_1 (1+\epsilon)\Phi(S^*), \quad (13)$$

where $\leq_1$ comes from the fact $\Phi(S^\dagger) \leq 2\Phi(S^*)$.

We now prove the complexity. Since the dynamic programming algorithm returns a solution with $P' = \sum_{i=1}^{M} P^\flat_{i,s^\epsilon_i}$. The space complexity is hence

$$O(MP') = O\left(\frac{2M\Phi(S^\epsilon)}{\epsilon\Phi(S^*)} + M\right) = O\left(\frac{M}{\epsilon} + M\right). \quad (14)$$

Similarly, the time complexity is $O(\frac{MK_{\max}}{\epsilon} + MK_{\max})$ since the time complexity for deriving an entry $\Psi_i(p)$ is $O(K_i)$. □

5. PERFORMANCE EVALUATION

This section provides performance evaluation for the proposed power management schemes, including Greedy, E-Greedy, and DP. To demonstrate the generality of our approach, three QoS models are applied, i.e., the M/M/1 [8] queuing model, the M/G/1 PS [9] queuing model, and a soft real-time model similar to[6]. All results in this experiment are mean values of 10 different runs on an Intel Xeon CPU with 3.06 *GHz*.

5.1 Simulation Setting

To evaluate how heterogeneity affects the power consumption, a 4-tuple ($f_{i,max}$, c_i, α_i, β_i) is used to compute the power consumption. Variables $f_{i,\max}$, c_i, α_i and β_i are random variables within range $[1, 4]$, $[20, 80]$, $[200, 400]$, and $[2, 5]$, denoting the maximum speed, the constant power consumption, the CPU performance coefficient, and the frequency coefficient of server m_i, respectively. The power consumption of a deactivated server is assumed to 0. The operating frequencies are discretized into 10 scaling levels by uniform distribution within range (0, $f_{i,max}$). The power consumption for server m_i at frequency f is $P_i(f) = c_i + \beta_i \cdot f^3$, as adopted in [8, 3, 5] as well. For evaluation, we evaluate cases with 100 and 200 back-end servers, considering three QoS models as follows.

M/M/1 Queuing Model [8]: In this model, the average response time is used as the QoS constraint. As a result, $L_{i,j}(R) = f_{i,j} \cdot \alpha_i - \frac{1}{R}$ where R is the average response time given for the QoS control. Since R only introduces constant offset, the setting of R only has minor effect. Therefore, we set R as 1 in our experiment. For comparison, we also simulate an algorithm extended from the TP-CP-OP algorithm developed in [8] which assumes continuous frequencies. To find a feasible solution for discrete frequencies, the closest upper frequency on each server is used,, denoted as R-TP-CP-OP.

M/G/1 PS Queuing Model [9]: In this model, job arrivals to the servers follow a Poisson distribution. The QoS constraint is the mean response time $E[R] = 0.38$ sec. The resulting $L_{i,j}(R) = (\mu \cdot r - \frac{1}{E[R]})\frac{1}{f_{i,j}}$ where $1/\mu = 38\,ms$ is the mean job-execution time and $r = \frac{f_{i,j}}{f_{i,max}}$ is the speed ratio of the execution speed to the maximum speed of server m_i. We do not compare with the approaches in [9] since they focus on homogeneous servers.

Soft Real-Time Request (SRR) Model: The SRR model is similar to the one in [6] and considers only dynamic requests. The execution time of a request follows a normal distribution with mean $\lambda = 24.5\,ms$ and deviation $\delta = 60\,ms$. The deadline of a request is $D = 200\,ms$. The QoS constraint is that the probability of all requests that will not miss their deadlines is $R = 95\%$. The $L_{i,j}(R)$ is thus defined as the maximal $L_{i,j}$ such that the probability of $L_{i,j} \cdot \lambda < f_{i,j} \cdot D$ is 0.95. In this experiment, we use the inverse cumulative distribution function of the normal distribution $(L_{i,j} \cdot \lambda, \sqrt{L_{i,j} \cdot \delta^2})$ coupled with a binary search to find $L_{i,j}(R)$.

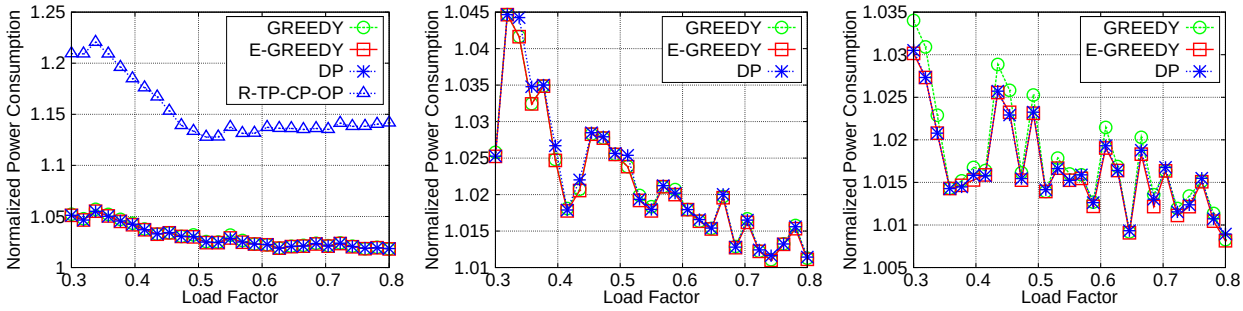
To vary the average request rate, we first compute the maximum tolerable request rate $\Lambda_{\max}$ of the cluster, $\sum_{i=1}^{M} L_{i,K_i}(R)$. For an input average request rate Λ, the load ratio is defined as $\frac{\Lambda}{\Lambda_{\max}}$. A lower bound of the optimal solution is computed as the baseline, which is obtained by adding $density(v_{i^*})\ell\frac{\Lambda-\ell}{q_{i^*}}$ to the solution when Algorithm 2 hitting the condition $\ell + q_{i^*} \geq \Lambda$. For comparison, all power consumption reported are normalized with respect to the computed lower bounds.

5.2 Simulation Results

Figure 4 illustrates the normalized power consumption of a 100-server cluster for the aforementioned three models. As shown in the figure, our schemes reasonably approximate the lower bounds for all cases. In general, better results are achieved when the load ratio increases. Especially for cases of load ratio larger than 0.5, our schemes derive solutions that consume less than 3% additional power consumption for all three models, compared to the lower bounds. The second observation is that since Algorithm R-TP-CP-OP uses a fixed order of servers according to high workload (80% of the maximal average request rate on servers), the decision for activating servers might be only sub-optimal, as depicted in Figure 4a. Note that, in Figures. 4b and 4c, we only compare our results with the computed lower bounds, because the approaches presented in [9, 6] apply exhaustive search and exact method to compute the optimum, respectively, the complexity of which constrains these approaches to clusters with small scales.

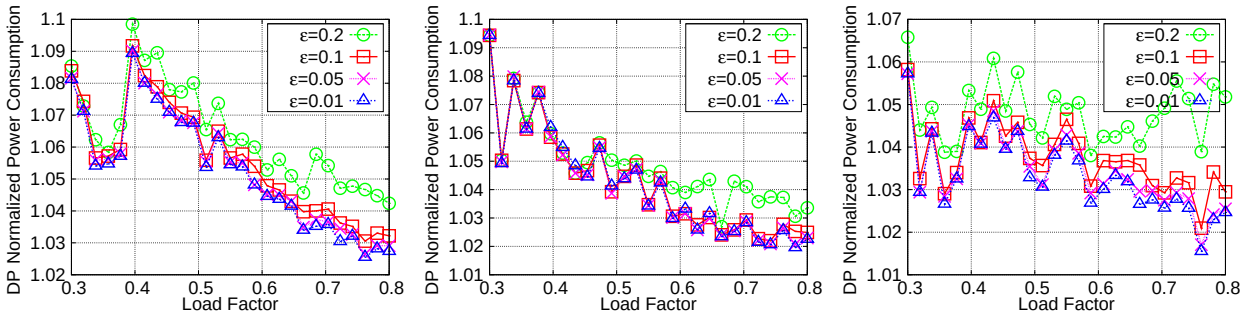
We also present the impact of the ϵ to Algorithm DP for all three QoS models in Figure 5 for a cluster with 200 servers. As expected, the smaller ϵ, the better approximation is obtained, at the cost of longer computation time. One observation is that that impact of varying the ϵ becomes more significant as the load ratio increases. The reason is that with a larger load, the exploration space is larger, and a higher ϵ would result in more errors for rounding down the power consumption in (7). From the figure, we can conclude that 0.05 is a proper value for ϵ. Further smaller values are not necessary.

Figure 6 depicts the computation time of our algorithms for a 200-server cluster. As shown in the figure, the time to compute a solution for this cluster is reasonably fast for all three algorithms. Algorithm GREEDY takes only a few milliseconds while the slowest one, i.e. Algorithm



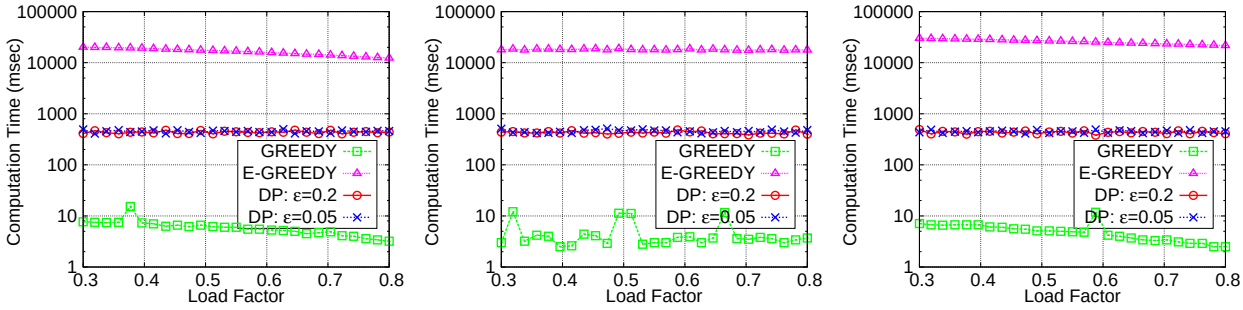
(a) M/M/1 (b) M/G/1 PS (c) SRR

Figure 4: Normalized power consumption of a 100-server cluster for the three QoS models with $\epsilon = 0.05$ for Algorithm DP.



(a) M/M/1 model (b) M/G/1 PS model (c) SRR model

Figure 5: Varying the ϵ for all three models for a 200-server cluster.



(a) M/M/1 model (b) M/G/1 PS model (c) SRR model

Figure 6: Computation time for all three models for a 200-server cluster.

E-GREEDY, is still in the range of seconds. Note that the computation time for Algorithm R-TP-CP-OP is not included, because it takes hours even for the 100-server case. From this figure, we can conclude that our algorithms are also suitable for time-critical large-scale clusters.

6. CONCLUSION

This paper explores the power management problem for a heterogeneous cluster to minimize the power consumption while guaranteeing quality of service constraints. We propose approximation algorithms to provide tradeoffs of approximation guarantees in power consumption minimization with time/space complexity. Simulation results show that the proposed schemes are effective for minimizing the power consumption.

7. REFERENCES

[1] L. A. Barroso. The price of performance. *Queue*, 3(7):48–53, 2005.

[2] J. S. Chase, D. C. Anderson, P. N. Thakar, A. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centres. In *Symposium on Operating Systems Principles*, pages 103–116. ACM Press, 2001.

[3] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo. Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems. In *IEEE RTAS*, pages 408–417, 2006.

[4] R. Guerra, J. Leite, and G. Fohler. Attaining soft real-time constraint and energy-efficiency in web servers. In *ACM SAC*, pages 2085–2089, 2008.

[5] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *ACM/EDAC/IEEE DAC*, pages 275–280, 2004.

[6] C. Rusu, A. Ferreira, C. Scordino, and A. Watson. Energy-efficient real-time heterogeneous server clusters. In *IEEE RTAS*, pages 418–428, 2006.

[7] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[8] L. Wang and Y. Lu. Efficient power management of heterogeneous soft real-time clusters. In *IEEE RTSS*, pages 323–332, Washington, DC, USA, 2008. IEEE Computer Society.

[9] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. In *Proc. IEEE INFOCOM*, Rio de Janeiro, 20-25 Apr 2009.

[10] R. Xu, D. Zhu, C. Rusu, R. Melhem, and D. Mossé. Energy-efficient policies for embedded clusters. In *ACM SIGPLAN/SIGBED LCTES*, pages 1–10, 2005.