Partition-based Profit Optimisation for Multi-class Requests in Clusters of Servers

Ligang He, James Wenjun Xue and Stephen A. Jarvis
Department of Computer Science,
University of Warwick, Coventry, CV4 7AL, United Kingdom
liganghe@dcs.warwick.ac.uk

Abstract

This paper investigates profit optimisation by the partitioning of server pools. Different types of web requests are considered in this paper: best-effort requests and multi-class QoS-demanding requests. Each request is associated with a certain amount of revenue. The system earns associated revenue when completing a best effort request, while the revenue earned for serving a QoS-demanding request depends on the degree of satisfaction. The complete server pool is divided into two clusters of servers, each dedicated to serving one type of request. The total profits accrued when serving these requests are mathematically modelled in this paper. The optimisation equations for the pool partitioning have also been constructed so that the maximum profit can be achieved. An optimal server switching policy is also developed. The server switching policy is optimal in the sense that the maximum profit can be maintained by switching servers from one partition to the other. Costs involved in server switching are factored into the switching model. Supportive experimentation based on simulation have been conducted and the results verify the effectiveness of the pool partitioning policy and the server switching policy developed in this paper.

1. Introduction

It is common in e-business/commerce environments that different types of web requests are submitted for service. These requests have different requirements. Some requests do not have Quality-of-Service (QoS) requirements and are termed *best-effort* requests. Service providers attempt to supply effective and reliable service to this type of requests, but no guarantee is made as to the completion time. As long as a request is completed, a certain amount of revenue is earned.

Other requests are associated with QoS requirements under a Service-Level-Agreement (SLA) between the users and the service providers [11]. These requests are termed QoS-demanding request. If the request's QoS requirements are satisfied, the full revenue is earned by the service provider. Otherwise, depending on the SLA, only a discounted revenue (or no revenue) is gained. The QoS-demanding requests can be further divided into multiple service classes. The higher the service class of the requests is, the higher the revenue associated with the requests.

It is common practice in e-business/commerce environments that a service provider (often represented as a server or a cluster of servers) will only serve certain types of requests [12]. Therefore, different types of request are placed in separate queues, each of which is linked to a specific service provider.

The arrival rates of the different types of requests may also change over time. It is likely that the servers for one service provider are overused while the servers for another are underutilised. Therefore, it may be desired to adjust the server allocation accordingly to maintain the maximum profit. During the server re-allocation, the servers are switched from one service provider to another, which involves stopping the current service, leaving the current service provider, being reconfigured for the new service, and finally joining the server partition for another service provider. During this period, the servers being switched cannot process requests [12].

In this paper we: (i) investigate profit optimization of best-effort and QoS-demanding requests through the partitioning of clusters of servers and, (ii) calculate dynamically the conditions for switching servers between partitions according to the changes in the arrival rates of the requests.

Related research has been carried out to optimise the performance of web requests in cluster environments [2][13]. The work in [11] studies methods for maximizing profit for best-effort and QoS-demanding requests in a web-server farm. The difference between their work and that presented here is three fold. First, the application scenarios are different. [11] specifically addresses request allocation in a web-server farm. The method presented allocates the best-effort- and QoS-demanding requests across the whole server farm. In our work, these request types are derived from different applications; that is, two partitions of servers are configured with different services. One partition of servers can only serve either best-effort requests or QoS-demanding requests. Therefore, our focus

0-7695-3003-6/07 $25.00 © 2007 IEEE
DOI 10.1109/ICEBE.2007.71

is to achieve such a partition of the server pool that can gain maximum profit.

Second, the request allocation method developed in [11] assumes that the arrival rates of these requests are static. In our work, the arrival rates may change dynamically. Although the servers can be switched from one partition to the other, it is at the expense of spending time on reconfiguring the services. Therefore, an additional focus of our work is to investigate the optimised server switching policy to maintain the maximum profit when the arrival rates change.

Finally, in the workload model presented in [11], if the QoS demand of a request is missed, a fixed penalty is incurred. In this paper, we generalise the problem. When the QoS demand of a request is missed, the revenue to be gained diminishes over time following a certain function. The profits are modelled in this general setting.

The work presented in [12] also addresses the problem of partitioning a server pool to process different types of web request. It is also the case that the servers in one partition may switch to another dynamically. In [12] the holding cost for different types of request is assumed and the aim is to optimise the costs by switching servers dynamically. The work presented in this paper is different from [12] in several respects. First, this work adjusts the partition (by switching servers between the partitions) as changes occur in the arrival rate of requests, while [12] assumes that the arrival rates of the requests are fixed and adjustments are made to the partition for every incoming request. Second, [12] focuses on the holding costs of the requests and aims to optimise the total costs due to server switching, while this work takes into account the revenue and the distribution of the QoS demands, and aims to optimise the profits gained by the system.

The rest of the paper is organised as follows. Section 2 discusses the system and workload models. The server partitioning policy is derived in Section 3. In section 4 the server switching policy is proposed. Experimental results are presented in Section 5. This paper is concluded in Section 6.

2. System and Workload Models

The best-effort requests and QoS-demanding requests arrive at the system following Poisson processes. The arrival rates of best-effort requests and QoS-demanding requests are λ_b and λ_q, respectively. The mean service times of the best-effort requests and the QoS-demanding requests are $1/u_b$ and $1/u_q$, respectively.

Fig. 1 depicts the system model used in this work, which is often encountered in e-business scenarios. In the system, two types of requests are submitted through a common interface. However they are further placed in separate queues and served by different service providers. The pool of servers is divided into two partitions, one dedicated to serving best-effort requests (denoted as P_b) and the other to QoS-demanding requests (denoted as P_q).

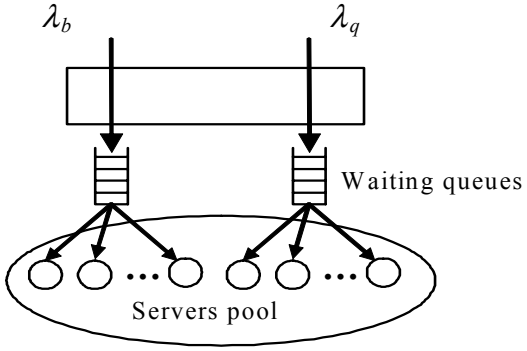


Fig. 1. The system model

When a best-effort request is completed, the revenue A_b is gained by the system. The Qualities-of-Service required by the QoS-demanding requests are expressed as the maximum waiting time that a request can tolerate before starting execution [1][11][15]. The full revenue of A (usually, $A > A_b$) is gained if the request's waiting time is less than its QoS requirement (maximum tolerable waiting time); otherwise, the revenue to be earned diminishes over time down to zero, following function $g(t)$, where t is the actual waiting time of the request. In Fig.2, the full revenue is gained until the waiting time reaches the QoS demand x, and then the revenue decreases linearly after x. When the waiting time exceeds the deadline by D_0, no revenue is earned. $g(t)$ in Fig. 2 can be formulated as Equation (1)

$$g(t) = -\frac{A}{D_0}t + \frac{A}{D_0}(x + D_0) \tag{1}$$

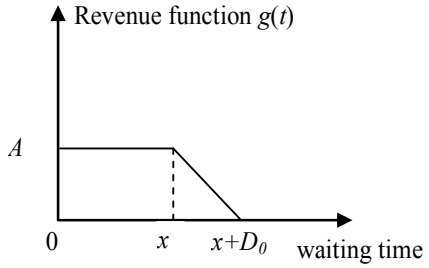


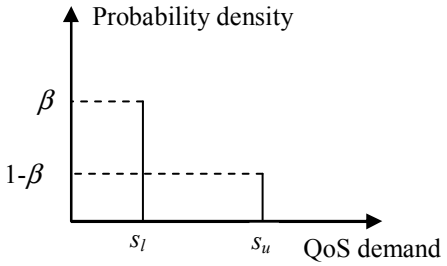
Fig.2. Illustration of the revenue function over time

There can be multiple discrete service classes in a typical e-business environment. The requests belonging to the same service class have the same QoS demands. It is shown in [7][8] that the QoS demands of the requests follow a q-spike distribution, where q is the number of service classes. Its probability density function $S(x)$ is formulated in Equation (2), where s_k is the QoS demand of the requests in the service class k ($1 \le k \le q$). A 2-spike distribution is illustrated in Fig. 3a.

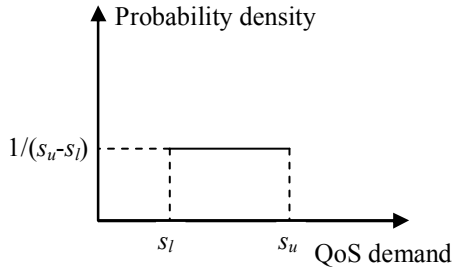
$$S(x)=\begin{cases}\beta_1 & x=s_1\\ \beta_2 & x=s_2\\ \vdots & \\ 1-\sum_{i=1}^{q-1}\beta_i & x=s_q\end{cases} \quad (2)$$

The discrete form of the service classes can be generalized into the continuous form, assuming that the QoS demand of the requests follow the continuous probability density function $S(x)$. For example, it is found in [7][8] that the QoS demands can follow a uniform distribution, as illustrated in Fig. 3b. Its probability density function $S(x)$ is given in Equation (3), where s_u and s_l are the upper and lower limits of the tolerable waiting time respectively.

$$S(x)=\frac{1}{s_u-s_l} \quad (3)$$



(a) 2-spike distribution



(b) Uniform distribution

Fig. 3. Distribution of the QoS demands (maximum tolerable waiting time of the requests)

The requests in the different service classes are usually associated with different full revenues. Therefore, the full revenue A in Fig. 2 can be expressed as a function of the QoS demands. $a(x)$ in Equation (4) is an example of such a function if the QoS demands of the requests follow the uniform distribution shown in Fig. 3. In $a(x)$, when the QoS demand of a request equals s_l (i.e. the request has the greatest demand), its full revenue is A_q ($A_q > A_b$). When a request has the least demand, s_u, its full revenue is

$$A_b+\frac{(A_q-A_b)}{e}$$

$$a(x)=A_b+(A_q-A_b)e^{-\frac{x-s_l}{s_u-s_l}} \quad (4)$$

During the server switching period, the switching servers cannot process requests. c_1 denotes the cost of a host switching from partition P_b to P_q. c_2 denotes the cost of a host switching from P_q to P_b.

3. The server partitioning policy

The partitioning policy presented in this paper aims to optimise the profit gained by the server pool. In this subsection, we first model the profits in a time unit for a given partitioning of the server pool, and then construct the optimisation equations for achieving the optimal profits by taking the partitioning parameters as input.

Suppose the number of the servers in the server pool is m and the number of servers in partition P_b and P_q is m_b and m_q (m_q=m-m_b), respectively. The partitioning of the server pool is denoted as (m_b, m_q)

Since the revenue of A_b is awarded for completing a best-effort request, the profit gained in a time unit for completing the best-effort requests, denoted as p_b, can be calculated using Equation (5), where T_b is the throughput of the system for processing the best-effort requests. T_b can be calculated as in Equation (6) [10]

$$p_b=A_bT_b \quad (5)$$

$$T_b=\begin{cases}\lambda_b & \lambda_b<m_bu_b\\ m_bu_b & \lambda_b\geq m_bu_b\end{cases} \quad (6)$$

It is non-trivial to calculate the profits gained for serving the QoS-demanding requests. The full revenue associated with a request is gained if its waiting time is no more than its QoS demand; otherwise, the revenue is discounted following function $g(t)$, such as that shown in Fig. 2. The expression for calculating the profit of the QoS-demanding requests is derived as follows.

The partition (with m_q servers) used to serve the QoS-demanding requests is modeled as an M/M/m_q queue. As shown in [4][10], in an M/M/m_q queue, the cumulative probability distribution function of the job's waiting time, $P_w(x)$ (i.e., the probability that the job's waiting time is less than x is $P_w(x)$), can be calculated using Equation (7), where P_{m_q} is the probability that the system has no less than m_q requests. P_{m_q} can be calculated as in (8) [4][10].

$$P_w(x)=1-P_{m_q}e^{-(m_qu_q-\lambda_q)x} \quad (7)$$

$$P_{m_q}=\frac{m_qu_q(\frac{\lambda_q}{u_q})^{m_q}}{\left[m_q!\sum_{k=0}^{m_q-1}\frac{(\frac{\lambda_q}{u_q})^k}{k!}+\frac{(\frac{\lambda_q}{u_q})^{m_q}}{(1-(\frac{\lambda_q}{m_qu_q}))}\right](m_qu_q-\lambda_q)} \quad (8)$$

When the QoS demand of a request is x, the full revenue associated with it is $a(x)$, such as that seen in

Equation (4). Therefore, the probability that the full revenue of $a(x)$ will be gained is $P_w(x)$.

If the request's waiting time falls in the duration $[x, x+D_0]$, as shown in Fig. 2, the revenue to be earned follows the function $g(t)$. $w(t)$ is denoted as the probability density function of the request's waiting time in an M/M/m_q queuing model. According to queuing theory [10], $w(t)$ can be calculated as in Equation (9), where $\delta_0(t)$ is the unit impulse (Dirac delta) function.

$$w(t) = (1 - \frac{\lambda_q}{m_q u_q})\delta_0(t) - P_{m_q}(m_q u_q - \lambda_q)e^{-(m_q u_q - \lambda_q)t} \quad (9)$$

Therefore, the expected revenue to be gained for processing the requests with the QoS demand of x can be calculated as

$$a(x)P_w(x) + \int_x^{x+D_0} g(t)w(t)dt$$

Since the probability density function of the requests' QoS demands is $S(x)$ ($s_l \le x \le s_u$), the profit which can be gained (in a time unit) by the system for serving the QoS-demanding requests, denoted as p_q, can be calculated as in Equation (10); where T_q is the throughput of the system for processing the QoS-demanding requests. T_q can be computed similarly as in Equation (6).

$$p_q = T_q \times \int_{s_l}^{s_u} (a(x)P_w(x) + \int_x^{x+D_0} g(t)w(t)dt) \times S(x)dx \quad (10)$$

Applying Equation (1), (4), (3), (7), (8) and (6), the integral in Equation (10) can be reduced to Equation (11), where K_1, K_2, S and H are calculated in expression (12)

$$\begin{aligned} & K_1 D_0 - K_2 D_0 (e^{\frac{-s_u}{H}} - e^{\frac{-s_l}{H}}) \\ & + \frac{P_{mq}}{H}(\frac{e^{SD_0}}{S} - \frac{1}{S} - 2D_0)(\frac{K_1}{S}(e^{SSu} - e^{SSl}) \\ & + \frac{HK_2}{SH-1}(e^{(S-\frac{1}{H})Su} - e^{(S-\frac{1}{H})Sl})) \end{aligned} \quad (11)$$

$$\begin{cases} K_1 = \dfrac{A_{q1}}{D_0} \\ K_2 = \dfrac{(A_{q2} - A_{q1})}{D_0} e^{\frac{s_l}{s_u - s_l}} \\ S = -(m_q u_q - \lambda_q) \\ H = s_u - s_l \end{cases} \quad (12)$$

Thus, the profit to be gained in a time unit by the system for serving both types of request, denoted as p, can be calculated as

$$p = A_b T_b + T_q \times \int_{s_l}^{s_u} (a(x)P_w(x) + \int_x^{x+D_0} g(t)w(t)dt)S(x)dx \quad (13)$$

The profit optimisation problem is now reduced to maximising Equation (14a), subject to the condition expressed in Equation (14b).

$$\begin{cases} p = A_b T_b + T_q \times \int_{s_l}^{s_u} (a(x)P_w(x) + \int_x^{x+D_0} g(t)w(t)dt)S(x)dx & (a) \\ m_b + m_q = m & (b) \end{cases} \quad (14)$$

The optimised partition for achieving maximum profit can be identified by searching all possible partitions, of which there are m. Therefore, the time complexity of achieving the optimized profits is O(m).

4. The server switching policy

When the arrival rates of the requests change, the current partition may not be able to deliver the maximum profit. It is therefore desirable to switch the servers from one partition to the other accordingly. However, it is not a cost-free procedure. During the switching period, the servers being switched cannot serve requests. Therefore, decisions have to be made as to whether it is worthwhile to switch servers, and if so, how much computing power should be switched.

Assume that the time spent switching a server from partition P_b to P_q is C_b and the time spent from P_q to P_b is C_q.

Assume t_1, t_2, …, t_n are a sequence of time points when the arrival rates of the requests change. If at time t_i ($1 \le i \le n$), the partition is (m_b, m-m_b), we label the partition state at t_i as m_b, denoted as $ST_i(m_b)$ ($0 < m_b < m$). The server switching problem can be modelled as finding the optimal path for the state transition of the partition from t_1 to t_n so that the total profits are maximised.

Assume that at time point t_i ($1 \le i \le n$) the arrival rates of the requests change to λ_b^i and λ_q^i, respectively. If the current partition state is $ST_i(m_b)$ and the partition state is updated to $ST_i(m'_b)$, the profit gained from t_i to t_{i+1}, denoted as $PL_{i,i+1}(m_b, m'_b)$, is derived in CASE 1) and CASE 2) as follows.

CASE 1) If $m_b = m'_b$, i.e. the current partition state $ST_i(m_b)$ remains unchanged, $PL_{i,i+1}(m_b, m'_b)$ can be computed by Equation (15). Here $p_o(\lambda_b^i, \lambda_q^i)$ is the optimal profit if adjusting the partition by taking the new arrival rates as input; $p_c(\lambda_b^i, \lambda_q^i)$ is the profit if the current partition remains unchanged, but with the new arrival rates as input and, t_{i+1} is the next time point when the arrival rates change.

$$PL_{i,i+1}(m_b, m_b^{'}) = p_c(\lambda_b^i, \lambda_q^i) \times (t_{i+1} - t_i) \quad (15)$$

$p_c(\lambda_b^i, \lambda_q^i)$ can be calculated by the application of Equation (13).

CASE 2) If $m_b \neq m'_b$, $|m'_b - m_b|$ is the number of servers that cannot serve requests during the switching period of C_b or C_q, depending on the switching direction. $PL_{i,i+1}(m_b, m'_b)$ can be calculated using Equation (16), where $p'(\lambda_b^i, \lambda_q^i)$ is the profit gained in a time unit during the server switching period, and $p_a(\lambda_b^i, \lambda_q^i)$ is the profit gained after the partition is adjusted up to the time point t_{i+1}. $p'(\lambda_b^i, \lambda_q^i)$ can be calculated using Equation (17), where the p_b and p_q functions are calculated by applying

Equations (5) and (10) respectively, with the parameters in the brackets as input. $p_a(\lambda_b^i, \lambda_q^i)$ can be calculated by applying Equation (13)

$$PL_{i,i+1}(m_b, m_b') = \begin{cases} p'(\lambda_b^i, \lambda_q^i) \times C_b + p_a(\lambda_b^i, \lambda_q^i) \times (t_{i+1} - t_i - C_b) \text{ if } m_b' < m_b \\ p'(\lambda_b^i, \lambda_q^i) \times C_q + p_a(\lambda_b^i, \lambda_q^i) \times (t_{i+1} - t_i - C_q) \text{ if } m_b' > m_b \end{cases} \quad (16)$$

$$p'(\lambda_b^i, \lambda_q^i) = \begin{cases} p_b(m_b', \lambda_b^i, \lambda_q^i) + p_q(m_q, \lambda_b^i, \lambda_q^i) \text{ if } m_b' < m_b \\ p_b(m_b, \lambda_b^i, \lambda_q^i) + p_q(m - m_b', \lambda_b^i, \lambda_q^i) \text{ if } m_b' > m_b \end{cases} \quad (17)$$

Denote $SPL_i(m_b)$ as the sum of the profit up to time point t_i when the partition state is $ST_i(m_b)$. Then

$$SPL_{i+1}(m'_b) = SPL_i(m_b) + PL_{i,i+1}(m_b, m'_b) \quad (18)$$

The server switching policy for achieving the maximum profit is outlined in Algorithm 1.

Algorithm 1. The optimal server switching policy

1. **for** each t_i ($1 \le i \le n$) **do**
2. **for** each m'_b ($0 < m'_b < m$) **do**
3. **for** each m_b ($0 < m_b < m$) **do**
4. **if** $m_b = m'_b$
5. apply Eq.8 to compute $PL_{i,i+1}(m_b, m'_b)$
6. **else**
7. apply Eq.9 to compute $PL_{i,i+1}(m_b, m'_b)$
8. **end if**
9. $tmp = SPL_i(m_b) + PL_{i,i+1}(m_b, m'_b)$
10. if $tmp > max\_profit$
11. $max\_profit = tmp$
12. $last\_path = m_b$
13. **end for**
14. $SPL_{i+1}(m'_b) = max\_profit$;
15. store the partial path of state transition $<m_b, m'_b>$
16. **end for**
17. **end for**
18. The maximum profit is $max\{SPL_n(m_b) | 0 < m_b < m\}$
19. Trace back from m_b computed in Step 18 to obtain the path for the partition state transition

The time complexity of Algorithm 1 is $O(m^2 n)$. Its effectiveness is proved in Theorem 1.

Theorem 1. Algorithm 1 can achieve the maximum profit.

Proof: We regard a partition state at time point t_i, i.e. $ST_i(m_b)$, as a vertex in a tree and the state transition from $ST_i(m_b)$ to $ST_{i+1}(m'_b)$ as an edge between vertices. Then, the weight of the edge, i.e. the transition cost, can be calculated using Equation (15) (if $m_b = m'_b$) or Equation (16) (if $m_b \neq m'_b$). Therefore, achieving the maximum profit in the stated scenario is equivalent to finding the longest path to the root in a tree. Algorithm 1 is equivalent to the algorithm of finding the longest path in a tree. Thus, Algorithm 1 can achieve the maximum profit.

5. Experimental Studies

Simulation experiments have been conducted to evaluate the performance of the pool partitioning and the server switching policy.

λ_b and λ_q are determined in the experiments as follows, where WL is the percentage of the saturated workload and PA_b is the proportion of the best-effort requests in the total workload.

$\lambda_b = mu_b \times WL \times PA_b$

$\lambda_q = mu_q \times WL \times (1\text{-}PA_b)$

When applying Equation (14) to search for the optimal partitioning, or reconfiguring the servers and switching them to the other partition, the arrival rates of the requests may be greater than the total service rates of a partition (oversaturated). When this happens to the best-effort partition, the profit gained in a time unit is calculated using Equations (5) and (6). If this situation occurs to the QoS-demanding partition, the profit gained in a time unit is set to 0.

The default experimental parameters are given in Table 1 unless otherwise stated.

Table 1. Parameters for simulation experiments

	Explanation	**Value**
$S(x)$	Distribution of QoS demands	Eq.3
$g(t)$	Revenue function of waiting time	Eq.1
$a(x)$	Full Revenue function of QoS demands	Eq.4
s_u	Upper limit of QoS demands	1.0
s_l	Lower limit of QoS demands	0
WL	Percentage of saturated workload	0.5
PA_b	Proportion of best-effort requests	0.5
PA_q	Proportion of QoS demanding requests	0.5
m	Total number of servers in the server pool	12
u_b	Service rate for best-effort requests	0.5
u_q	Service rate for QoS-demanding requests	1.0
D_0	Parameter in $g(t)$, See Eq.1	0.5
A_b	Revenue associated with best-effort requests	5.0
A_q	Parameter in $a(x)$, See Eq.4	10.0
C_b	Time spent switching from partition P_b to P_q	1.0
C_q	Time spent switching from partition P_q to P_b	2.0
TP	Average time period between two consecutive changes in arrival rates	3.0

The traditional policy for pool partitioning allocates computing power to different types of requests according to their proportions in the total workload [12][14]. The performance metrics evaluated in these experiments are *profit unit* (profit gained in a time unit) and *profit* (total profit during the specified time period)

5.1 Performance comparison with the traditional partitioning

Fig. 4 compares the performance between the partitioning policy presented in this paper and the traditional partitioning policies based on the proportions of these two types of requests.

It can be observed from Fig. 4 that the advantage of the optimal partitioning over the proportional partitioning becomes increasingly prominent as the workload level

increases. This is because when the workload level is low, the demands of the QoS-demanding requests can be easily satisfied. Therefore, even if the pool partition generated by the proportional policy is different from that generated by the optimal policy, the demands of the requests can still be met. When the workload level becomes high, the small difference in the partitions generated by these two policies could lead to a big variance in the profits.

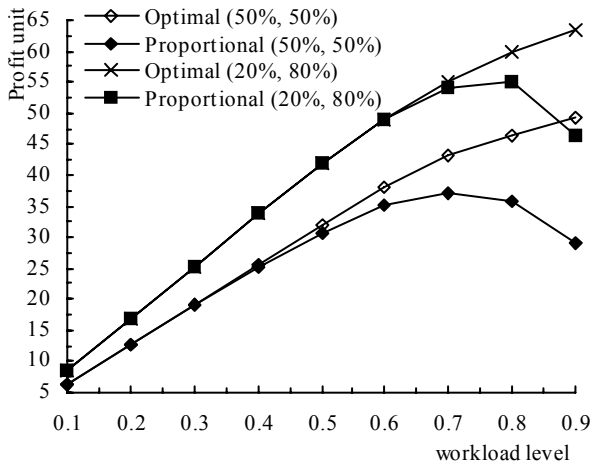


Fig. 4. Performance comparison among optimal partitioning and the proportional partitioning under different workload levels; the percentages in the legend are the proportions of these two types of requests.

5.2 Impact of partitioning

The experiments in Fig. 5 demonstrate the impact of the different partitioning on the profit, as well as the contributions that these two types of requests make. The number on the *x*-axis is the number of servers in partition P_b. The remaining servers in the pool are used to serve the QoS-demanding requests.

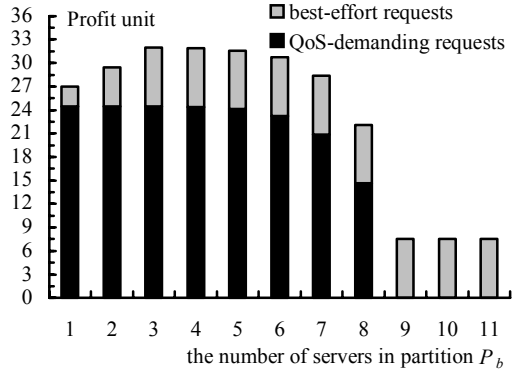


Fig. 5. The impact of partitioning on the profit; WL=0.5, PA_b=0.5, PA_q=0.5.

As can be observed from Fig. 5, the profit unit first increases and then decreases as fewer servers are allocated to serve the QoS-demanding requests. Further observations show that the profit earned from the QoS-demanding requests remains constant until the number of servers processing the best-effort requests reaches 5. This is because the QoS demands of all requests can be satisfied and therefore, the full revenue associated with the QoS-demanding requests can be gained.

5.3 Impact of the request proportions

Fig. 6 demonstrates the profit unit as a function of the proportions of these two types of requests. The contributions that these two types of requests make towards the profit are also demonstrated in this figure.

As shown in Fig. 6, the portion of the profit earned by the best-effort requests increases as the proportion of the best-effort requests increases. However, the profit unit decreases as the proportion of the best-effort requests increases. This is because the higher revenue is associated with the QoS-demanding requests. Therefore, when the proportion of the QoS-demanding requests is higher, more revenue is expected to be gained. This suggests that the QoS-demanding requests play a bigger role in determining the profit gained by the system.

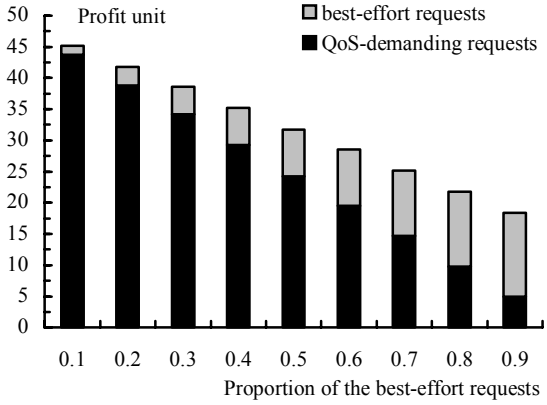


Fig. 6. Performance under different proportions of the best-effort requests and QoS demanding requests; WL is 0.5; other experimental parameters are the same as those found in Fig.1.

5.4 Impact of QoS demands on profits

Fig. 7 shows the impact of the QoS demands (i.e. s_u) on the unit of profit. The QoS demands are measured as the maximum waiting time that a request can tolerate, which follows a uniform distribution in $[s_l, s_u]$. If the system fails to meet the QoS demand of a request, the revenue gained decreases following the function defined in Equation (1).

As shown in Fig. 7, the results from the two workload levels show different trends. When the workload level is 0.5, the changes in the QoS demands have no obvious impact on the profit, while when the workload level is 0.8, the profit increases as s_u increases. This is because when

the workload is 0.5, the QoS demands of most requests can be satisfied under the system and workload settings in the experiments.

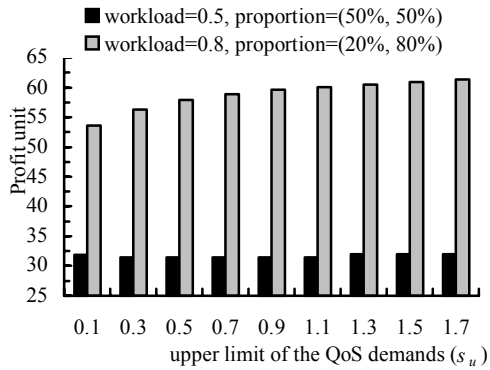


Fig. 7. Impact of QoS demands on profits; The QoS demands (waiting time) follows the uniform distribution in time duration [s_l, s_u].

It is therefore the case that further reducing the QoS demands (by increasing s_u) will not affect the profit. Under the higher workload level of 0.8 however, the QoS demands of more requests can be met by reducing the QoS demands, which leads to an increase in the profit.

5.5 Performance comparison with other server switching policies

Experiments have been conducted to compare the performance achieved by the optimal server switching policy presented in this work, with the proportional switching policy and a non-switching policy. The proportional switching policy is defined as follows: the servers are switched so that after switching, the computing power in each partition is proportional to the percentage of these two types of requests.

Table 2. Performance comparison under different server switching policies; Cb=1, Cq=2; the initial workload level is 0.1; the average time duration between two consecutive changes in the workload level is 3; PAb=0.5, PAq=0.5; other parameters are the same as those found in Fig. 4.

		Workload level							
	policies	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Profit	The optimal policy	56.6	114.1	190.9	286.5	400.4	529.8	669.3	814.5
	No server switching	56.0	107.6	173.9	254.8	350.4	460.6	585.2	723.8
	Proportional switching	56.5	113.9	189.6	281.8	386.98	498.6	606.2	693.7

Table 3. Profits as the calculation of the optimal server switching policy progresses; the experimental parameters are the same as those in Table 2. In the brackets, the first figure is the profit up to the current time point and the second figure is the partition state at the last time point when the workload level changes.

		Workload level								
	m_b	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Profit	0	(0.0, 0)	(48.6, 1)	(100.6, 2)	(172.9, 2)	(264.3, 3)	(374.7, 3)	(502.1, 4)	(644.5, 4)	(795.5, 4)
	1	(19.2, 0)	(56.1, 1)	(108.1, 2)	(180.4, 2)	(271.8, 3)	(382.2, 3)	(509.5, 4)	(651.9, 4)	(802.6, 4)
	2	(0.0, 0)	(56.6, 1)	(114.1, 2)	(187.9, 2)	(279.3, 3)	(389.7, 3)	(516.8, 4)	(658.9, 4)	(808.9, 4)
	3	(0.0, 0)	(56.6, 1)	(114.1, 2)	(190.9, 3)	(286.7, 3)	(397.0, 3)	(523.8, 4)	(665.0, 4)	(813.5, 4)
	4	(0.0, 0)	(56.6, 1)	(114.1, 2)	(190.8, 3)	(286.5, 3)	(400.4, 4)	(529.8, 4)	(669.3, 4)	(814.5, 4)
	5	(0.0, 0)	(56.6, 1)	(114.1, 2)	(190.6, 3)	(285.6, 3)	(398.0, 4)	(524.7, 4)	(659.7, 4)	(795.1, 4)
	6	(0.0, 0)	(56.6, 1)	(113.9, 2)	(189.8, 3)	(283.1, 3)	(391.6, 4)	(511.0, 4)	(633.4, 4)	(749.8, 4)
	7	(0.0, 0)	(56.5, 1)	(113.3, 2)	(187.2, 3)	(275.9, 3)	(375.6, 4)	(479.6, 4)	(578.0, 4)	(702.8, 4)
	8	(0.0, 0)	(56.0, 1)	(110.7, 2)	(179.3, 3)	(257.2, 3)	(338.2, 4)	(430.9, 4)	(561.8, 4)	(702.8, 4)
	9	(0.0, 0)	(53.8, 1)	(102.3, 2)	(157.8, 3)	(213.4, 3)	(313.5, 4)	(430.9, 4)	(561.8, 4)	(702.8, 4)
	10	(0.0, 0)	(45.6, 1)	(78.2, 2)	(132.1, 3)	(213.4, 3)	(313.5, 4)	(430.9, 4)	(561.8, 4)	(702.8, 4)
	11	(0.0, 0)	(27.2, 1)	(70.1, 2)	(132.1, 3)	(213.4, 3)	(313.5, 4)	(430.9, 4)	(561.8, 4)	(702.8, 4)

Under the non-switching policy, the initial optimal partition is calculated using the pool partitioning policy. Then the partition remains unchanged throughout the sequence of changes in the workload level. Table 2 only shows the experimental results when the workload level changes, while the proportions of the two types of requests remain unchanged. The results for changing both workload level and request proportion demonstrate the greater advantages of the optimal policy over the other two policies.

It can be observed from Table 2 that the optimal server switching policy outperforms other two policies in all cases. This suggests that even if the request proportions remain unchanged, it may still be advantageous to adjust the partitioning when the workload level changes.

5.6 Operations of the optimal server switching policy

Table 3 shows the progress of the optimal server switching policy at the sequence of time points when the workload level changes (i.e. the operations of Algorithm 1). The initial workload level is 0.1. It can be observed in this table that at any time point and for a given partition, the server switching policy calculates the state transition path from the current partition to a given partition to achieve the maximum sum of the profit. When the workload level is 0.9, in all possible partitions, the maximum profit is achieved when m_b is 4. Tracing back the transition path, we can obtain the complete path of the partition state transition <1, 2, 2, 3, 3, 4, 4, 4>. This indicates that the server switching policy can judiciously adjust the partitioning and, as a result, the maximum profit is maintained.

6. Conclusion

This paper addresses profit optimisation for different types of requests in server pools. Both best-effort requests and QoS-demanding requests are considered in this paper. The QoS-demanding requests can be further classified into different service classes according to the urgency of their QoS demands under Service-Level-Agreements. The QoS demands are measured by the maximum waiting time that a request can tolerate before starting execution.

An optimal server partitioning policy is developed in this paper. In the policy, the profit gained in a time unit for serving the requests is mathematically modelled. The model respects the reality that the revenue gained for serving QoS-demanding requests depends on the satisfaction level. A general expression for calculating the profit is derived. Based on this expression, an optimisation equation is constructed to find the optimal partition of the server pool. The partition is optimal in the sense that the server pool can deliver maximum profit.

A server switching policy is also developed in this paper to adjust the server partitioning when the arrival rates of requests change. The server switching cost is taken into account in the switching model. The server switching policy can determine the optimal path of partition state transition so that the maximum profit can be maintained when the arrival rates of the requests change.

Acknowledgements

This work is funded in part by the UK Engineering and Physical Sciences Research Council, project reference EP/C538277/1.

References

[1] B. Adelberg, H. Garcia-Molina, B. Kao, “Emulating soft real-time scheduling using traditional operating system schedulers,” *Proc of IEEE 1994 Real-time Systems Symposium*, pp.292-298.

[2] S. A. Banawan, N. M. Zeidat. “A comparative study of load sharing in heterogeneous multicomputer systems.” *Proceedings of the 25th Annual Simulation Symposium*, 1992, pp. 22–31.

[3] M. Barreto, R. Avila, P. Navaux, “The MultiCluster model to the integrated use of multiple workstation clusters,” *Proc. of the 3rd Workshop on Personal Computerbased Networks of Workstations*, 2000, pp. 71–80.

[4] G. Bolch, Performance Modeling of Computer Systems, 2002.

[5] A. Bucur, D. Epema, “The maximal utilization of processor co-allocation in multicluster Systems,” *Int'l Parallel and Distributed Processing Symp* (IPDPS 2003), 2003, pp. 60-69.

[6] M. Fan, S. Mahlke, “Region-based hierarchical operation partitioning for multicluster processors,” *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, 2003, pp. 300-311.

[7] B. Kao, H. Garcia-Molina, “Scheduling soft real-time jobs over dual non-real-time servers,” *IEEE Trans. on Parallel and Distributed Systems*, 7(1): 56-68, 1996.

[8] L. He, S. A. Jarvis, D. P. Spooner, H. Jiang, D. N. Dillenberger, G. R. Nudd, "Allocating Non-real-time and Soft Real-time Jobs in Multiclusters", *IEEE Transactions on Parallel and Distributed Systems*, 17(2):99-112, 2006

[9] L. He, S. Jarvis, D. Spooner, G. Nudd, "Optimising static workload allocation in multiclusters", *18th IEEE International Parallel and Distributed Processing Symposium* (IPDPS'04), 2004

[10] L. Kleinrock, Queueing system, John Wiley & Sons, 1975.

[11] Z. Liu, M. S. Squillante, J. L. Wolf, “On maximizing service-level-agreement profits,” *Proceedings of the 3rd ACM conference on Electronic Commerce*, 213 – 223, 2001

[12] I. Mitrani, J. Palmer, "Optimal Server Allocation in Reconfigurable Clusters with Multiple Job Types", *Procs, International Conference of Computational Science and its Applications*, Part II, pp 76-86, 2004

[13] N. G. Shivaratri, P. Krueger, M. Singhal. “Load distribution for locally distributed systems.” *IEEE Computer,* 8(12):33–44, Dec. 1992.

[14] X.Y. Tang, S.T. Chanson, “Optimizing static job scheduling in a network of heterogeneous computers,” *the 29th International Conference on Parallel Processing*, 2000.

[15] W. Zhu, B. Fleisch, “Performance evaluation of soft real-time scheduling on a multicomputer cluster,” *The 20th International Conference on Distributed Computing Systems* (ICDCS 2000), 2000.