Two-tier Resource Allocation for Slowdown Differentiation on Server Clusters

Xiaobo Zhou Yu Cai C. Edward Chow Marijke Augusteijn
Department of Computer Science
University of Colorado at Colorado Springs, CO 80933, USA
Corresponding author: Xiaobo Zhou, zbo@cs.uccs.edu

Abstract

Slowdown, defined as the ratio of a request's queueing delay to its service time, is accepted as an important quality of service metric of Internet servers. In this paper, we investigate the problem of providing proportional slowdown differentiation (PSD) services to various applications and clients on cluster-based Internet servers. We extend a closed-form expression of the expected slowdown of a popular Internet workload model with a typical heavy-tailed service time distribution from a single server mode to a server cluster mode. Based on the closed-form expression, we design a two-tier resource allocation approach, which integrates a dispatcher-based node partitioning scheme and a server-based dynamic process allocation scheme. We evaluate the two-tier resource allocation approach via extensive simulations and compare it with an one-tier node partitioning approach. Simulation results show that the two-tier approach can provide fine-grained PSD services on cluster-based Internet servers. We implement the two-tier approach on a cluster testbed. Experimental results further demonstrate the feasibility of the approach in practice.

1 Introduction

Due to the openness and dynamics of the Internet, the past few years have seen an increasing demand for provisioning of different levels of quality of service (QoS) to meet changing system configuration and resource availability and to satisfy different application and client requirements [3, 4, 10, 13, 16, 18, 19, 20]. A desirable property of an Internet server is that a request's queueing delay depends on its service time in a linear fashion [8]. For example, a request for a job twice as long as some others will spend time on the average twice as long in the server. Performance metric slowdown, defined as the ratio of a request's queueing delay to its service time, reflects this need. Both queueing delay and response time are major performance metrics on the server side. But they are not suitable to compare requests that have very different resource demands. Actually, clients are likely to anticipate short delays for "small" requests, and are willing to tolerate long delays for "large" requests [6]. A high slowdown also indicates the system is heavily loaded. Slowdown or its variant is being used as a fundamental performance metric in recently designed QoS-aware systems [2, 6]. For instance, in [6], the author proposed a task assignment mechanism for dispatching tasks with heavy-tailed size distribution to hosts of a cluster to minimize the mean slowdown of the jobs. Although the size-based QoS-aware resource management is able to ensure that small requests experience small slowdowns, none of the systems can guarantee quantitative quality spacings among request classes in terms of slowdown.

The proportional differentiation model [4] states that QoS metrics of certain classes of aggregated requests should be proportional to their differentiation parameters, independent of their workloads. It is accepted as an important relative Differentiated Services model [11] and is applied in the proportional delay and loss rate differentiation in packet forwarding and dropping [4, 7]. It is also adopted for server-side service differentiation [10, 18, 19]. We proposed a proportional slowdown differentiation (PSD) model on individual Internet servers [18]. Its objective is to maintain slowdown ratios between request classes according to their pre-specified differentiation parameters. In this paper, we investigate the problem of provisioning PSD services on cluster-based Internet servers.

The cluster-based network services are increasingly deployed on the Internet due to the inherent scalability and cost-effectiveness of cluster architectures. A server cluster also provides a new resource allocation granularity, server node, for service differentiation. We design a two-tier resource allocation approach for PSD services provisioning. Figure 1 depicts its infrastructure. Tier-1 scheme is to dynamically partition the server nodes into a number of sets (called sub-clusters). Differentiation priorities assigned to classes are observed by providing differentiated processing rates with node partitioning. One sub-cluster handles one request class in FCFS discipline. To provide fine-grained

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

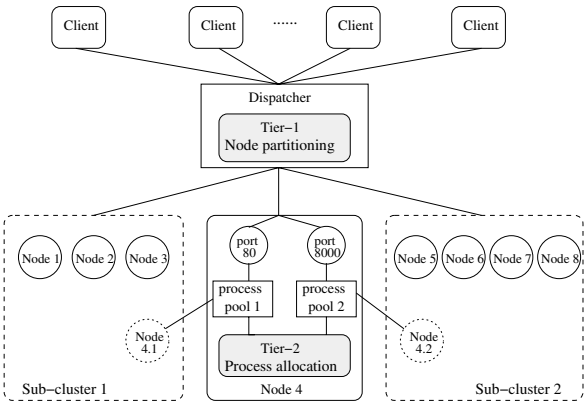

Figure 1. Two-tier resource allocation.

PSD services, one node or multiple nodes may have to be shared by different classes. Tier-2 scheme is to dynamically allocate resource units on an individual server, processes, to handle requests of different classes. Figure 1 shows a two-class differentiation scenario, where a node is shared between two classes by tier-2 resource allocation.

The problem of provisioning PSD services on cluster-based Internet servers is important, because 1) the proportional model is a widely accepted differentiation model; 2) slowdown is a key QoS metric on the server side; and 3) cluster is a popular and scalable platform. It is challenging because in order to meet the need of quantitative differentiation, a closed-form expression of expected slowdown with respect to resource allocation on a server cluster is required. The tier-1 and tier-2 resource allocation schemes must be integrated to provide fine-grained PSD services. We are concerned with the scenario where the interarrivals meet Poisson distribution and the workload is heavy-tailed, in specific, meeting a bounded Pareto distribution as it is characteristic of many empirically measured Internet workloads [1, 6]. The workload is referred to as an $M/G_P/1$ model. We extend the expected slowdown expression in a closed analytic form from a single server mode to a server cluster mode. Based on the closed-form expression, we design the two-tier resource allocation approach for PSD provisioning on a server cluster. We build a simulation model to evaluate the differentiation predictability and controllability of the approach. We further implement the two-tier resource allocation approach on a cluster testbed to verify the feasibility of the approach in practice.

The structure of the paper is as follows. Section 2 reviews related works. Section 3 gives the slowdown expression for an $M/G_P/1$ workload on a server cluster. Section 4 presents the two-tier resource allocation approach. Section 5 focuses on the performance evaluation. Section 6 presents the design and implementation issues. Section 7 concludes the paper with remarks on future work.

2 Related Work

The proportional differentiation model has been extensively studied in packet scheduling with respect to packet delay, packet loss, and connection bandwidth; see [4, 7, 15] for representative approaches. The work in [10] demonstrated that some approaches developed for proportional delay differentiation (PDD) on networks can be tailored for PDD provisioning on Internet servers. However, those approaches are not applicable to PSD provisioning because slowdown is not only dependent on a job's queueing delay but also on its service time, which varies significantly depending on the requested services [18]. There are efforts on priority-based request scheduling with admission control for response time differentiation [3, 14]. Incoming requests were categorized into the appropriate queues and executed according to their strict priority levels [3] or adaptive priority levels [14]. The results showed that higher priority classes receive less response time than lower classes. However, the quality spacings between classes cannot be quantitatively guaranteed. In [18], we proposed a processing rate allocation strategy for PSD provisioning on individual Internet servers. We used virtual servers in performance simulation. In this paper, we extend the slowdown modeling from a single-server mode to a server-cluster mode. We further design and implement a two-tier application-level approaches for providing fine-grained PSD services.

In [20], stretch factor, a variant of slowdown, was adopted as the performance metric for differentiation provisioning in a server cluster. The work adopted an $M/M/1$ queueing model to guide dispatcher-based node partitioning optimization. The work implicitly applied processor sharing discipline for the modeling of stretch factor. However, in a single queue, a realistic scheduling discipline is FCFS. More importantly, recent Internet workload measurements indicate that for many Internet applications the exponential distribution is a poor model for service time distribution and that a heavy-tailed distribution is more accurate [1, 6].

In [13], the authors proposed and designed a sound integrated resource management framework that provides flexible service quality specification, efficient resource utilization, and service differentiation for cluster-based Internet services. The work introduced an interesting metric, quality-aware service yield, to combine the overall system efficiency and individual service response time in one model. As the work in [20], it chose exponentially distributed arrival intervals and service times for modeling and performance evaluation. In this paper, we investigate the problem of PSD provisioning with a popular heavy-tailed traffic pattern. Simulation aside, we also design and implement the two-tier resource allocation approach on a cluster testbed to verify its feasibility in practice.

3 Slowdown Modeling on a Server Cluster

Predictability and controllability are two basic requirements of service differentiation. Predictability requires that higher priority classes receive better or no worse service quality than lower priority classes, independent of the class load conditions. Controllability requires that a scheduler contain a number of controllable parameters that are adjustable for the control of quality spacings between classes. For quantitatively predictable and controllable differentiation, we need to have a closed form expression of slowdown with respect to resource allocation on cluster-based servers. We consider a popular heavy-tailed distribution, bounded Pareto distribution, for modeling service time distribution on Internet workload characteristics [1, 6]. The bounded Pareto distribution with respect to job size x is characterized by three parameters: α, the shape parameter; k, the shortest possible job; and p, the upper bound of jobs. As in [6], the probability density function is defined as:

$$f(x) = \frac{1}{1-(k/p)^\alpha}\alpha k^\alpha x^{-\alpha-1} \qquad \alpha, k > 0, k \le x \le p.$$

Since α, k, and p are parameters of the bounded Pareto distribution, we define a function $\mathcal{K}(\alpha,k,p) = \frac{\alpha k^\alpha}{1-(k/p)^\alpha}$. The probability density function $f(x)$ is rewritten as:

$$f(x) = x^{-\alpha-1}\mathcal{K}(\alpha,k,p) \qquad \alpha, k > 0, k \le x \le p. \quad (1)$$

From (1), we have:

$$\begin{aligned} E[X] &= \int_k^p f(x)x\,dx \\ &= \begin{cases} \frac{\mathcal{K}(\alpha,k,p)}{\mathcal{K}(\alpha-1,k,p)} & \text{if} \quad \alpha \neq 1; \\ (\ln p - \ln k)\mathcal{K}(\alpha,k,p) & \text{if} \quad \alpha = 1. \end{cases} \quad (2) \\ E[X^2] &= \int_k^p f(x)x^2\,dx = \frac{\mathcal{K}(\alpha,k,p)}{\mathcal{K}(\alpha-2,k,p)}. \quad (3) \\ E[X^{-1}] &= \int_k^p f(x)x^{-1}\,dx = \frac{\mathcal{K}(\alpha,k,p)}{\mathcal{K}(\alpha+1,k,p)}. \quad (4) \end{aligned}$$

According to Pollaczek-Khinchin formula [8], we derived a closed-form expression of the expected slowdown in an $M/G_P/1$ workload on a single Internet server [18]. That is, let W be a job's queueing delay, and S be a job's slowdown, we have

$$E[S] = E[W]\cdot E[X^{-1}] = \frac{\lambda E[X^2]E[X^{-1}]}{2(1-\lambda E[X])}, \quad (5)$$

where the arrival process has rate λ and X denotes the bounded Pareto service time density distribution on the server. The slowdown formula follows from the fact that W and X are independent from a FCFS queue where requests of a class are processed by using FCFS discipline.

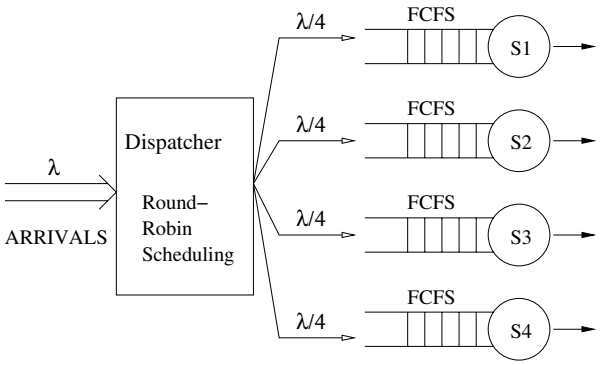

Figure 2. Round-Robin dispatching.

In the following, we further extend the closed-form expression of slowdown from the single-server mode to the server-cluster mode. Figure (2) illustrates a typical dispatching scheme, Round-Robin, which is one of the task assignment policies commonly proposed and deployed in clusters of servers [5, 9]. Jobs are assigned to hosts in a cyclical fashion with the kth job being assigned to Host $kmodm$, where m is the number of servers in the cluster. The policy equalizes the expected number of jobs at each host. There are other popular task dispatching disciplines, including Random, Shortest-Queue, and Least-Work-Remaining. We consider Round-Robin here because simplicity is an important property of any load-sharing scheme, as concluded by Eager *et al.* in [5] when they studied three approaches to load sharing in distributed servers. Relatively high overheads can outweigh the advantages of more complex schemes. Note that we do not want to compare different schemes for load sharing in this work. Like the Least-Work-Remaining task assignment scheme deployed in [6], Round-Robin scheme is also not applicable in the stateful e-Commerce applications while locality-aware request routing mechanisms support the session integrity [12]. As previous work in [6, 18], our work considers stateless jobs in the differentiation. When using Round-Robin dispatching policy, the arrival process at each server in the cluster has rate λ/m.

According to (5), we have

Lemma 1. *Given an $M/G_P/1$ queue on a cluster of m homogeneous FCFS servers, where λ denotes the arrival rate and X denotes the bounded Pareto service time density distribution on any of the servers. Let S_j be a job's slowdown on an individual server j ($1 \le j \le m$). By the use of Round-Robin task assignment on the dispatcher, the expected slowdown $E[S_j]$ is calculated as*

$$E[S_j] = \frac{\lambda E[X^2]E[X^{-1}]}{2(m-\lambda E[X])} \qquad 1 \le j \le m. \quad (6)$$

4 Two-tier Resource Allocation

Assuming incoming requests are classified into N classes, the proportional differentiation model aims to ensure the quality spacing between class i and class j to be proportional to certain pre-specified differentiation parameters δ_i and δ_j [4]; that is,

$$\frac{q_i}{q_j} = \frac{\delta_i}{\delta_j} \qquad 1 \le i, j \le N,$$

where q_i and q_j are the QoS factors of class i and class j, respectively. The PSD model aims to control the ratios of the average slowdown of classes based on their differentiation parameters $\{\delta_i, i = 1, \ldots, N\}$. Specifically, the PSD model requires that the ratio of average slowdown between class i and j is fixed to the ratio of the corresponding differentiation parameters

$$\frac{E[S_i]}{E[S_j]} = \frac{\delta_i}{\delta_j} \qquad 1 \le i, j \le N. \tag{7}$$

The differentiation predictability property requires that higher classes receive better service, i.e., lower slowdown. Without loss of generality, we assume class 1 is the "highest class" and set $0 < \delta_1 < \delta_2 < \ldots < \delta_N$.

Let M be the number of servers in the cluster and m_i be the number of server nodes assigned to a sub-cluster for processing the requests of class i. Then, we have

$$\sum_{i=1}^{N} m_i = M \qquad 0 < m_i \le M\ . \tag{8}$$

For feasible resource allocation, we must ensure the system utilization constraint $\sum_{i=1}^{N} \lambda_i E[X] < M$. That is, the total processing requirement of the N classes of traffic is less than the processing capacity of the cluster.

According to Lemma 1, the set of (7), in combination with the constraint (8), lead to a linear equation system. It follows

$$m_i = \lambda_i E[X] + \frac{\tilde{\lambda}_i (M - E[X] \sum_{i=1}^{N} \lambda_i)}{\sum_{i=1}^{N} \tilde{\lambda}_i} \tag{9}$$

where $\tilde{\lambda}_i = \lambda_i / \delta_i$, the normalized arrival rate. The first term of (9) ensures that the sub-cluster allocated for handling requests of the corresponding class will not be overloaded. The second term means that the remaining capacity of the cluster is proportionally allocated to different classes according to their scaled arrival rates with respect to their differentiation parameters.

Lemma 1 assumes a round-robin task assignment among the homogeneous servers in a sub-cluster. According to (9), the number of servers in a sub-cluster is often not an integer. For example, in a two-class 8-node scenario, the calculated m_1 and m_2 by (9) could be 3.4 and 4.6, respectively. The tier-1 dispatching mechanism adopts a weighted round-robin (WRR) scheduling discipline to assign incoming requests of a class to servers of its corresponding sub-cluster. For instance, the weight to four servers of sub-cluster 1 is 1, 1, 1, and 0.4. The weight to five servers of sub-cluster 2 is 1, 1, 1, 1, and 0.6. The tier-2 resource allocation mechanism aims to achieve proportional resource sharing on the shared server. It dynamically changes the number of processes allocated to process pools for handling different request classes according to changing workloads while ensuring the ratio of resource allocations [17].

According to Lemma 1, the expected slowdown of class i, $E[S_i]$, is calculated as:

$$E[S_i] = \frac{\delta_i E[X^2] E[X^{-1}] \sum_{i=1}^{N} \tilde{\lambda}_i}{2(M - E[X] \sum_{i=1}^{N} \lambda_i)}. \tag{10}$$

5 Performance Evaluation

5.1 Simulation Model

In this section, we investigate the impact of the two-tier resource allocation approach on proportional slowdown differentiation in a server cluster. We built a simulator which consisted of a number of request generators, waiting queues, an arrival rate predictor, a request dispatcher, and a number of servers. Figure 3 outlines its structure. The arrival rate predictor estimated a class's workload every sampling period. A sampling period was set to the processing time of one thousand requests of mean job size. A moving window with window size of five sampling periods was adopted for workload prediction. The tier-1 node partitions and tier-2 process allocations were calculated according to (9). In the cluster, some servers were dedicated to handling requests from a specific request class, while some others were shared by multiple request classes. The dispatcher used WRR scheduling discipline to assign incoming requests to corresponding sub-clusters. In each of the shared servers, a number of task servers was simulated to handle requests of corresponding classes. As in [20], we compare the two-tier approach with an one-tier approach that relies on dynamic node partitioning but there is no tier-2 process allocation module. Another node partitioning approach is to assign a fixed number of servers to each class. The partitioning is static and obviously it cannot adapt to fluctuating workload conditions. Such kind of static node partitioning approaches cannot achieve proportional slowdown differentiation when workload is changing. Thus, it is omitted here.

The request generators produced requests with exponential interarrival distributions and bounded Pareto size distributions by using GNU scientific library. Each request was dispatched to a server (or a task server) and was processed

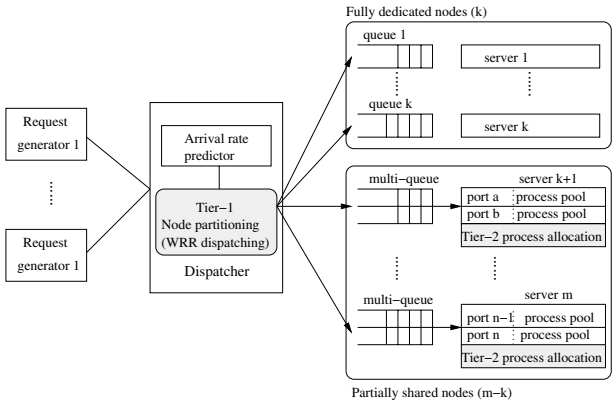

Figure 3. Structure of the simulation model.

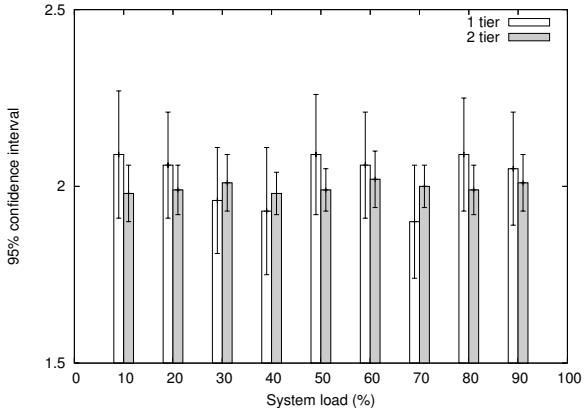

Figure 4. Two-class differentiation.

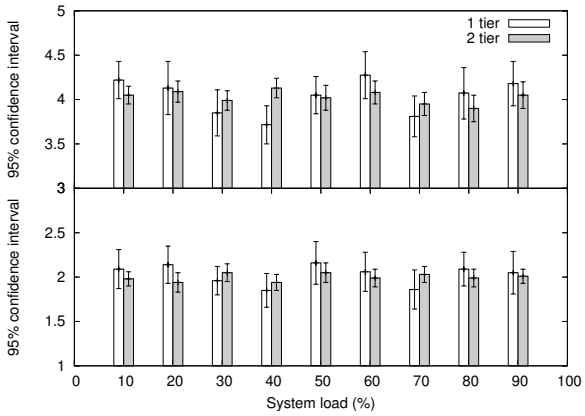

Figure 5. Three-class differentiation.

using FCFS discipline. In the simulation, different traffic classes may have different arrival rates. But they have the same job size distribution [18]. In the following, we first study the effectiveness of the two-tier resource allocation in achieving proportional slowdown differentiation in long term. Simulation parameters were set as follows. The shape parameter (α) of the bounded Pareto distribution was set to 1.5 as suggested in [4]. The lower bound (k) and upper bound (p) were set to 0.1 and 100, respectively [6]. The number of servers in the cluster was 8. We then conduct the sensitivity study of the approaches with respect to their short-term behaviors, differentiation variances, and the impact of the number of nodes in the cluster on differentiation performance.

5.2 Effectiveness of the Two-tier Approach

In the service differentiation context, the number of classes is limited, which is normally two or three [11, 20]. We first examine the slowdown differentiation due to two resource allocation approaches, *i.e.,* two-tier and one-tier, by the use of a two-class workload. Figure 4 depicts average slowdown ratios with the 95% confidence intervals at different workload conditions. The target slowdown ratios between two classes $\delta_b : \delta_a$ is 2 : 1. It shows that the achieved slowdown ratios due to two resource allocation approaches are always larger than 1, which means that the differentiated services are always predictable. That is, high priority classes receive better or "no worse" services than lower priority classes. The figure also shows that the two-tier approach achieves desirable proportionality of slowdown differentiation at various workload conditions. On the other hand, the one-tier approach based on node partitioning observes frequent and large slowdown ratio oscillations. This is because the node partitions calculated by (9) are rounded to integer numbers. The lack of tier-2 process allocation mechanism results in coarse-grained slowdown differentiation. Figure 4 also shows that the two-tier approach significantly outperforms the one-tier approach not only in terms of the mean slowdown ratio, but in terms of the size of the confidence intervals. This demonstrates the robustness of the two-tier approach.

We then investigate the differentiation problem by the use of a three-class workload. Figure 5 depicts the achieved slowdown ratios with the 95% confidence intervals at different system workload conditions. The target slowdown ratios among three classes $\delta_c : \delta_b : \delta_a$ is 4 : 2 : 1. In the case of three-class scenarios, a server may be shared by more than two classes and more than one server may be shared by multiple classes. From the figure, we can observe that the two-tier can achieve desirable proportionality of slowdown differentiation with respect to both the mean slowdown ratio and the size of confidence intervals.

In the simulation above, the arrival rate ratios of classes (A to B in the two-class scenario, and A to B to C in the three-class scenario) were set to be the same as their differentiation weight ratios. We conducted a wide ranger of

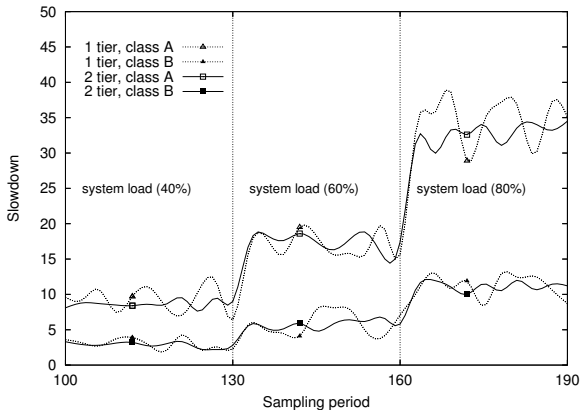

Figure 6. A short-term view of slowdown.

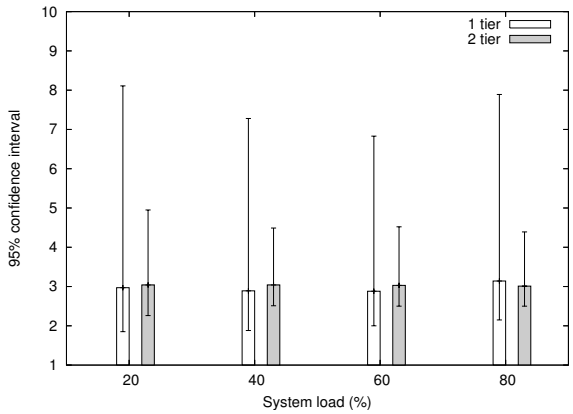

Figure 7. Variance of the proportionality.

sensitivity analyses. We varied the arrival rate ratios of the classes and the differentiation weight ratios of the classes. As it is expected, the proportional differentiation is independent of workload of classes. While we do not have space to present all of the results, note that we did not reach any significantly different conclusion regarding to the slowdown differentiation proportionality achieved by the two-tiered resource allocation approach.

5.3 Sensitivity Study of the Two-tier Approach

We have demonstrated that the two-tier resource allocation approach can achieve desirable proportional slowdown differentiation in long term with respect to the mean slowdown ratios. Now we want to investigate the sensitivity of the approach under different workload conditions. We conducted simulation using a two-class workload with target slowdown ratio $\delta_b : \delta_a = 3 : 1$. Figure 6 shows a short-term view of the slowdown of requests of the two classes in the sampling periods due to the two resource allocation approaches, when the system workload is low (40%), moderate (60%), and high (80%), respectively. The simulation was run for 100 sampling periods for warming up and then the data was collected for 30 sampling periods at each of three workload conditions. Obviously, we can observe that the two-tier approach achieves more consistent differentiation results during different sampling periods at various workload conditions.

Figure 7 further quantitatively depicts the variance of the differentiation proportionality due to the two approaches. At each of the four workload conditions (20%, 40%, 60%, 80%), we conducted simulations using a two-class workload with the target slowdown ratio $\delta_b : \delta_a = 3 : 1$. The upper line is the 95th percentile; the bar is the mean; and the lower line is the 5th percentile. We can observe that the two-tier approach can significantly reduce the variance generated by the one-tier approach. For example, when the workload is 40%, difference between the 95th and the 5th percentile is 1.98 and 5.4, and the mean is 3.04 and 2.89, due to the two-tier approach and the one-tier approach, respectively. At 80% workload condition, the difference between 5th and 95th is 1.89 and 5.74, and the mean is 3.01 and 3.14, due to the two-tier approach and the one-tier approach, respectively. The better proportionality in terms of the mean slowdown ratio has already been depicted by Figures (4) and (5). It is the small variance degree that further justifies the superiority of the two-tier resource allocation approach. We can conclude that the tier-2 process allocation mechanism deployed at the shared servers significantly contributes to the small variance degree of the proportional slowdown differentiation.

Figure 8 depicts the achieved mean slowdown ratio with the 95% confidence intervals due to two approaches when the number of servers in the cluster varies from 4 up to 20. Obviously, the performance of the two-teir approach is almost not affected by the number of nodes. On the other hand, the performance of the one-tier approach improves as the number of nodes in the cluster increases. This can be explained by the fact that as the number of nodes increases, the node partitioning itself can achieve finer granularity of resource allocation. It is obvious as the number of nodes goes up to infinity, there is no need to have the tier-2 process allocation mechanism.

6 Design and Implementation

We implemented the two-tier resource allocation approach on a Linux 2.6 system. The cluster testbed consists of four machines (PIII 600MHz, 256MB RAM) as request generators, one machine (PIII 1GHz, 516MB RAM) as the dispatcher of the cluster, and eight machines (PIII 600MHz, 256MB RAM) as cluster nodes. The machines are connected with 100Mbps Ethernet.

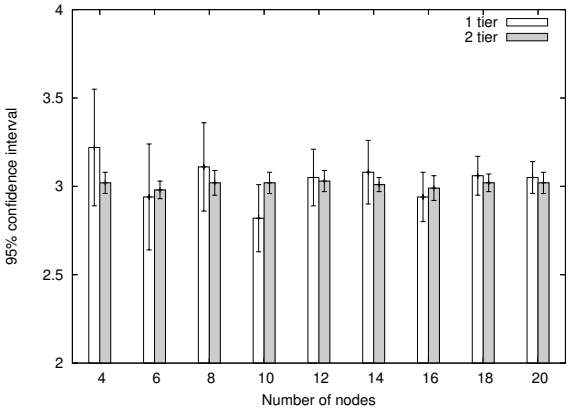

Figure 8. Impact of the number nodes on PSD.

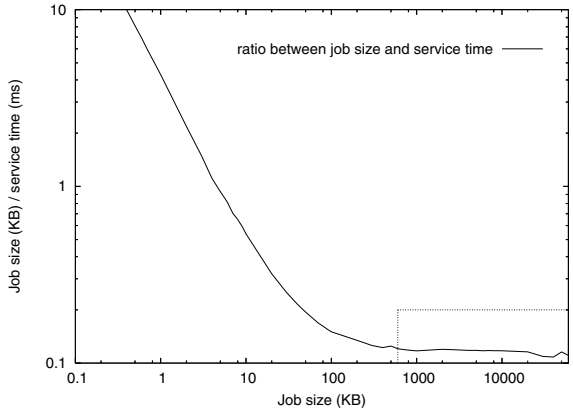

Figure 9. Service time vs. job size.

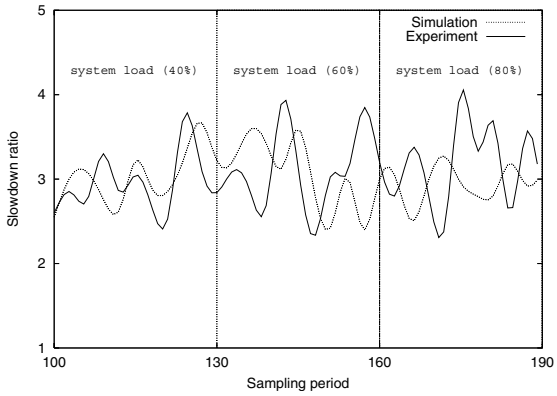

Figure 10. Experiment vs. simulation results.

We used enhanced Httperf to generate Http requests for a $M/G_P/1$ workload. We modified the Linux Virtual Server for the request dispatcher. In the dispatcher, a classifier module determines a request's class type according to its header information (e.g., port number). A node allocation module obtains the arrival rate of each class and calculates the node partitions according to (9). The dispatcher then assigns requests belonging to the same class to the cluster nodes in the corresponding sub-cluster using the weighted round-robin scheduling discipline. We deployed Apache Web Server 1.3.29 on each cluster node. The cluster nodes process the incoming Http requests, record their response time, and calculate the achieved slowdown. In a two-class scenario, one node may be shared by two classes. Note that every child process in Apache Web server is considered to be identical. We implemented an application-level dynamic process allocation approach on the shared node so as to achieve the fractional node partitions by controlling the number of child processes that a class is allocated. We also implemented several TCP/IP socket programs on the dispatcher and on the cluster nodes so that they can exchange information such as node partitions and achieved average slowdown on each node.

The implementation requires that the workload to meet the $M/G_P/1$ model, *i.e.*, the Poison distribution of arrival time and the bounded Pareto distribution of service time. In the simulation, service time of a request is often assumed to be proportional to the size of the requested job. However, in reality, the service time of a job is often not strictly linear to its size. Figure 9 depicts the relationship between the service time of a job on a cluster node and its size. We observed that when the job size is between 500 KB to 60 MB, the ratio between the job size and the service time is nearly linear. Therefore, we used the files in that size range to generate the $M/G_P/1$ workload in the experiment.

Figure 10 depicts the achieved slowdown ratio due to the two-tier resource allocation approach by simulation and by experiment based on the implementation, respectively. It shows the differentiation results of a two-class workload with target slowdown ratio ($\delta_b : \delta_a = 3 : 1$) when the system workload is low (40%), moderate (60%), and high (80%), respectively. Compared to the results due to the simulation, the experiment based on the implementation observes larger slowdown ratio oscillations and scales. This is explained by the fact that the generated workload in the experiment does not strictly meet the $M/G_P/1$ model. However, the experiment results overall are still around the target ratio. This demonstrates the feasibility of the two-tier resource allocation approach in achieving cluster-based PSD services in practice.

7 Conclusion

Slowdown is an important performance metric on Internet servers because it takes into account both the delay and the service time of a request simultaneously. There are few

research work for proportional slowdown differentiation (PSD) services on the server side, while proportional delay differentiation has been studied extensively both on the network side and on the server side. In this paper, based on a popular Internet traffic model with a typical heavy-tailed service time distribution, we extended a closed form expression of the expected slowdown from a single server mode to a server cluster mode using a simple and scalable weighted round-robin scheduling discipline on the dispatcher. we designed a two-tier resource allocation approach to provide PSD services in a server cluster. Simulation results have shown that the approach can achieve the PSD services on cluster-based servers. We also implemented the approach in a cluster testbed. Experimental results showed that the approach is effective and practical.

As in [6, 18], this work assumed that the jobs assigned to server clusters were stateless. Our future work will be on providing PSD services to stateful applications like e-Commerce on server clusters, where both Round-Robin and Least-Remaining-Work dispatching disciplines are not applicable to support session integrity.

Acknowledgement

This material is based on research sponsored by the Air Force Research Laboratory, under agreement numbers FA9550-04-1-0239 and F49620-03-1-0207. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

The authors would also like to thank Network Information and Space Security Center (NISSC) for providing support in the work.

References

[1] M. Arlitt, D. Krishnamurthy, and J. Rolia. Characterizing the scalability of a large Web-based shopping system. *ACM Trans. on Internet Technology*, 1(1):44–69, 2001.

[2] C. Chekuri, A. Goel, S. Khanna, and A. Kumar. Multi-processor scheduling to minimize flow time with ϵ resource augmentation. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 363–372, 2004.

[3] X. Chen and P. Mohapatra. Performance evaluation of service differentiating Internet servers. *IEEE Trans. on Computers*, 51(11):1,368–1,375, 2002.

[4] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proc. ACM SIGCOMM*, 1999.

[5] D. Eager, E. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Trans. on Software Engineering*, 12(5):662–675, 1986.

[6] M. Harchol-Balter. Task assignment with unknown duration. *Journal of ACM*, 29(2):260–288, 2002.

[7] Y. Huang and R. Gu. A simple fifo-based scheme for differentiated loss guarantees. In *Proc. of the Int'l Workshop on Quality of Service (IWQoS)*, 2004.

[8] L. Kleinrock. *Queueing Systems, Volume II*. John Wiley and Sons, 1976.

[9] T. T. Kwan, R. E. McGrath, and D. A. Reed. Ncsa's world wide web server: Design and performance. *IEEE Computer*, 28(11):68–74, 1995.

[10] S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau. A proportional-delay diffserv-enabled Web server: admission control and dynamic adaptation. *IEEE Trans. on Parallel and Distributed Systems*, 15(5):385–400, 2004.

[11] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the internet. *IETF RFC 2638*, 1999.

[12] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Drusche, W. Zwaenepoel, and E. Nahum. Locality-aware request distribution in cluster-based network servers. In *Proc. 8th Int'l Conf. on Architectural support for programming languages and operating systems*, 1998.

[13] K. Shen, H. Tang, T. Yang, and L. Chu. Integrated resource management for cluster-based Internet services. In *Proc. of USENIX OSDI*, pages 225–238, December 2002.

[14] M. M. Teixeira, M. J. Santana, and R. H. C. Santana. Using adaptive priority scheduling for service differentiation QoS-aware Web servers. In *Proc. IEEE 23rd Int'l Conf. on Performance, Computing, and Communications (IPCCC)*, pages 279–285, 2004.

[15] J. Wei, C.-Z. Xu, and X. Zhou. A robust packet scheduling algorithm for proportional delay differentiation services. In *Proc. of IEEE Globecom, Vol.2*, pages 697–701, 2004.

[16] Q. Zhang, E. Smirni, and G. Ciardo. Profit-driven service differentiation in transient environments. In *Proc. 11th IEEE/ACM Int'l Symp. on Modeling, Analysis and Simulation of Computer Telecommunications Systems*, 2003.

[17] X. Zhou, Y. Cai, G. K. Godavari, and C. E. Chow. An adaptive process allocation strategy for proportional responsiveness differentiation on Web servers. In *Proc. IEEE 2nd Int'l Conf. on Web Services (ICWS)*, pages 142–149, July 2004.

[18] X. Zhou, J. Wei, and C.-Z. Xu. Processing rate allocation for proportional slowdown differentiation on Internet servers. In *Proc. IEEE 18th Int'l Parallel and Distributed Processing Symp. (IPDPS)*, pages 88–97, April 2004.

[19] X. Zhou and C.-Z. Xu. Harmonic proportional bandwidth allocation and scheduling for service differentiation on streaming servers. *IEEE Trans. on Parallel and Distributed Systems*, 15(9):835–848, 2004.

[20] H. Zhu, H. Tang, and T. Yang. Demand-driven service differentiation for cluster-based network servers. In *Proc. IEEE INFOCOM*, pages 679–688, 2001.