Impact of Space-Time Multiplexing Granularity on Provisioning in On-Demand Data Centers: Opportunities and Challenges

Abhishek Chandra, Pawan Goyal† and Prashant Shenoy

Department of Computer Science
University of Massachusetts Amherst
{abhishek,shenoy}@cs.umass.edu

†IBM Almaden Research Center
San Jose, CA
goyalp@us.ibm.com

Abstract

On-demand data centers efficiently host multiple applications on shared hardware by dynamically provisioning resources in response to workload variations. In this paper, we use real-world e-commerce workloads to demonstrate that the efficiency of such dynamic provisioning is critically dependent on the granularity of reallocation. Specifically, we show that the ability to allocate fractional server resources at time-scales of seconds can improve the multiplexing benefits by a factor of three. Using these results, we argue that existing techniques for dedicated hosting are highly sub-optimal. We argue for a new hosting platform architecture, based on virtual machine monitors, that can perform fine-grain resource multiplexing along the space and time dimensions while retaining the isolation and security benefits of dedicated hosting platforms.

1 Introduction

Internet data centers host multiple applications on a shared hardware platform, such as a server farm, and provide client applications with computing and storage resources. In such environments, applications pay for data center resources and in turn are provided guarantees on resource availability and performance. Since typical data center applications service Internet users, the workload seen by such applications can often vary in an unpredictable fashion (as exemplified by flash crowd scenarios [1]). A data center should handle these workload variations by dynamically provisioning resources for applications so as to meet their contracted performance needs.

Recently several techniques have been proposed to dynamically provision resources to applications in on-demand data centers [2, 3, 11, 14, 15, 17]. A common characteristic of these techniques is that they use past workload measurements to predict changes in an application's resource needs and reallocate resources based on these predictions. However, these techniques differ drastically in *how frequently* resources are reallocated and by *how much*. For instance, dedicated hosting platforms [2, 7, 11, 14] advocate reallocating entire servers to applications to accommodate workload variations. In such systems, security and privacy considerations may require OS re-installation and disk scrubbing on a server prior to its allocation, resulting in allocation time-scales of a few minutes or tens of minutes. The time-scale of reallocations can be reduced by maintaining a ready pool of servers in energy-saving mode [11].

In contrast, shared hosting platforms [13, 15] advocate allocating fractional servers to applications (by running multiple applications on a server). Performance isolation between applications is ensured by partitioning server resources among competing applications. Whereas some efforts have advocated partitioning based on worst-case needs of an application and over coarse time-scales of hours or days [15], others have proposed repartitioning resources over fine time-scales of tens of seconds [13], albeit at the expense of weaker performance isolation.

While these efforts represent various point solutions with different tradeoffs, the question of *what is the right granularity at which resources should be dynamically provisioned in computing-on-demand data centers* remains unanswered. That is, (1) should data center resources be allocated to applications at a granularity of entire machines or is the ability to allocate fractional servers desirable, and (2) should resource be provisioned over time scales of seconds, minutes, or hours so as to extract the best multiplexing gains?

In this paper, we conduct an experimental study to understand the impact of the granularity of resource provisioning—both in the space and time dimensions—on the effective capacity of such data centers. Using traces of real workloads, we demonstrate that the ability to allocate fractional server resources and at fine time-scales of seconds can improve the multiplexing benefits by a factor of 3. Using these results, we argue that existing techniques are highly sub-optimal in the amount of multiplexing benefits they can extract when dynamically provisioning resources. We argue for a new hosting platform architecture, based on virtual machine monitors, that can perform fine-grain resource multiplexing along the space and time dimensions while retaining the isolation and security benefits of dedicated hosting platforms. We conclude by describing some of the research challenges that arise in the design of such an architecture.

The rest of this paper is structured as follows. Section 2 quantifies the benefits of space-time resource multiplexing in on-demand data centers using real-world trace workloads. We examine the effectiveness of existing hosting solutions for on-demand data centers in Section 3. Finally, Section 4 lays out our case for a new hosting platform architecture that achieves fine-grain resource multiplexing.

2 What is the Impact of Allocation Granularity on Resource Utilization?

To understand the impact of allocation granularity on resource utilization, consider Figure 1 which depicts a hypothetical resource allocation scenario in a data center. The lower curve in the figure shows the resource demand of an application. An *optimal* provisioning scheme will allocate resources exactly as demanded using infinitesimally small resource units and time quanta. Thus, the lower curve also represents the optimal resource allocation. In contrast, any practical provisioning scheme will allocate resources over a finite time period using finite resource units (e.g, one server); the resources allocated in any period should be sufficient to handle the peak requirements in that period. Figure 1 shows two such allocations, one coarse-grained and the other fine-grained. Observe that, depending on the granularity, there is some amount of over-allocation, since the allocation can be changed only once every Δt and the

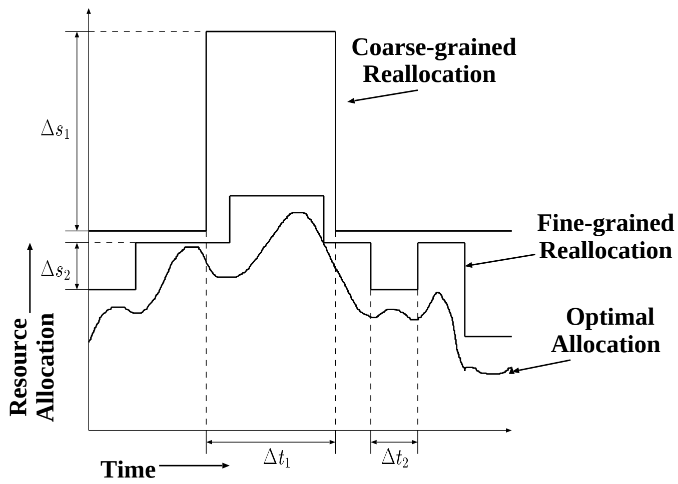


Figure 1: A metric for comparing optimal resource provisioning to practical approaches. An optimal approach can reallocate resources infinitely often and in infinitesimally small amounts; a practical approach uses a finite time and space granularity, Δt and Δs, respectively.

Workload	Duration	Number Requests	Avg Request size	Peak bit-rate
Ecommerce1	24 hrs	1,193,536	3.95 KB	458.1 KB/s
Ecommerce2	24 hrs	1,261,393	3.82 KB	1631.0 KB/s
Ecommerce3	24 hrs	247,705	7.23 KB	1346.9 KB/s

Table 1: Workload characteristics

allocation must always be a multiple of the allocation granularity Δs. Let $R^i_{opt}(t)$ and $R^i_{pract}(t)$ denote the amount of resources allocated to application i at time t using the optimal and a practical allocation scheme respectively. Then the maximum value of $\sum_{customers} R^i_{pract}(t)$ over all time t, denoted by R_{pract}, is the peak resource requirement of the practical scheme (this is essentially the total capacity required to host this set of applications). Similarly, the maximum value of $\sum_{customers} R^i_{opt}(t)$ over all time t, denoted by R_{opt}, is the peak capacity requirement of the optimal scheme. The *overprovisioning ratio* ρ is defined to be the percentage increase in resource requirement of the practical scheme when compared to the optimal:

$$\rho = \left(\frac{R_{pract}}{R_{opt}} - 1 \right) \cdot 100$$

Intuitively, ρ measures the percentage additional capacity required in a practical scheme to host the same set of applications when compared to the optimal.

To quantify the overprovisioning ratio for different spatial and temporal allocation granularities, we used web traces from three e-commerce sites hosted in a large commercial data center. The characteristics of these traces are summarized in Table 1. While these traces contain the arrival time and size of each request, the CPU processing time

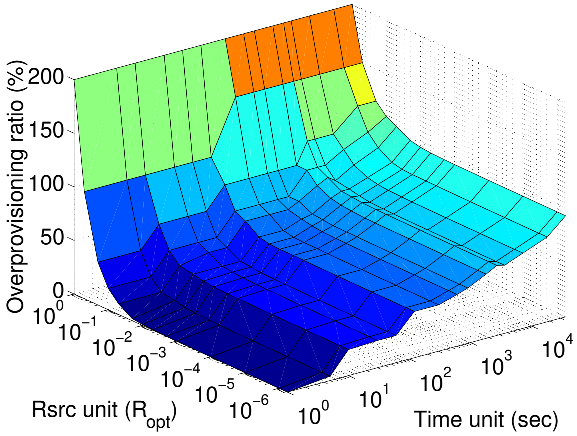
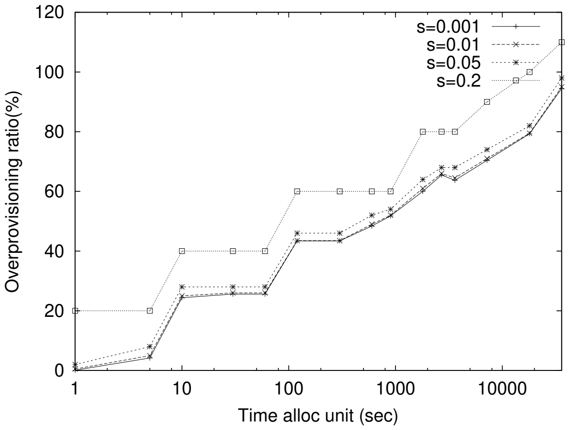
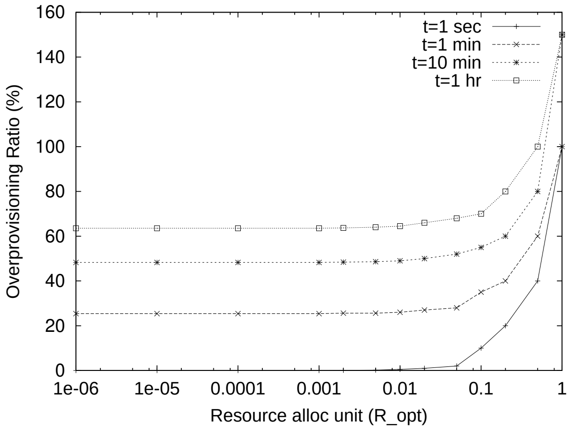


Figure 2: Effect of allocation granularity on resource utilization for a 3 customer system.

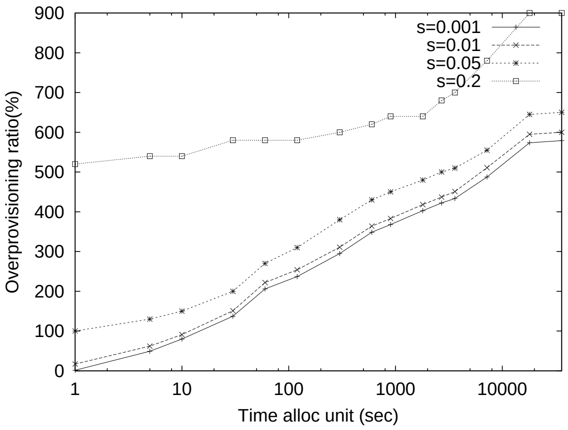


(a) Temporal overprovisioning ratio

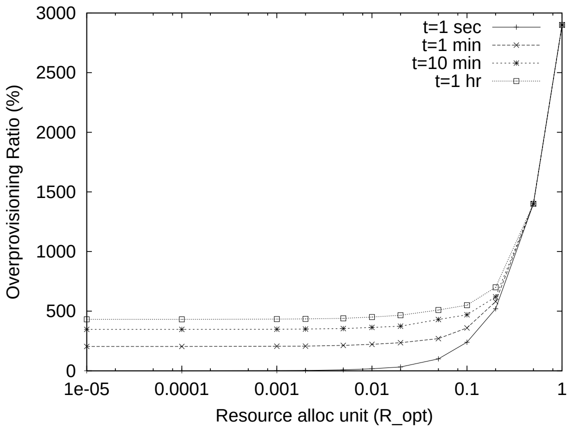


(b) Spatial overprovisioning ratio

Figure 3: Effect of varying time and spatial allocation granularities on overprovisioning ratio for a 3 customer system.



(a) Temporal overprovisioning



(b) Spatial overprovisioning

Figure 4: Effect of allocation granularity on resource utilization for a 30 customer system.

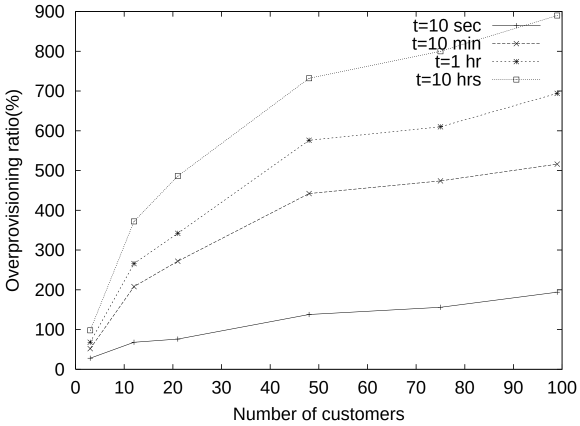


Figure 5: Statistical multiplexing of resources in data centers with large number of customers.

of a request was not available. Since for static requests, CPU usage is highly correlated with the request size, we use request size as a proxy metric for CPU usage. We systematically vary the spatial and temporal allocation granularity for this workload mix and compute ρ for each combination. To eliminate the impact of inaccuracies introduced by traffic predictors, we assume a clairvoyant resource allocation scheme that allocates resources based on exact knowledge of future workload requirements. Thus, our analysis produces theoretical best-case results for the workloads under study.

Figure 2 shows the values of overprovisioning ratio ρ for different temporal (Δt) and spatial (Δs) levels of reallocation granularity. We use a granularity of 1 second and 1 byte/sec to approximate the optimal allocation scheme. We express the spatial granularity as a fraction of the peak requirement of the optimal scheme R_{opt}. Thus, if peak capacity requirement is 10 servers, then a spatial granularity of 0.1 indicates resources are allocated 1 server at a time. Figure 2 shows that the coarser the spatial and temporal allocation granularity, the greater the overprovisioning ratio ρ (indicating larger overallocations at coarser allocation granularities). Next, we examine the effect of varying Δt and Δs in isolation on the overprovisioning ratio (see Figures 3(a) and 3(b)). Figure 3(a) shows that there is a monotonic increase in the value of ρ with Δt. ρ is relatively small for fine time allocations and increases with increasing Δt. Specifically, for $\Delta s = 0.01$, reallocating resources once every 1 sec, 1 min, 10 min and 1 hour yields ρ values of 1%, 27%, 50% and 66%, respectively. This result argues in favor of having as small a temporal allocation granularity as possible. In contrast, when the spatial allocation granularity is varied (see Figure 3(b)), we find that ρ is nearly constant until a certain point after which it increases steadily with increasing Δs. Specifically, for $\Delta t = 1$ minute, ρ is nearly constant at 25.4% until $\Delta s = 0.005$, and increases to 28%, 35%, 40% and 60% for Δs values of 0.05, 0.1, 0.2 and 0.5, respectively. Further, large spatial granularities yield very large overprovisioning ratios ρ regardless of the temporal granularity. This result suggests that while the spatial granularity need not be very fine-grained, the granularity should still be sufficiently small in order to extract high multiplexing gains.

The above results are for a data center with three customers. In practice, a data center will typically host applications from tens or hundreds of customers. To understand the impact of hosting a large number of applications on the overprovisioning ratio, we synthesize a larger number of traces from the three original traces by replication and time-shifting. For instance, to generate 30 traces, we replicate each of the original traces ten times and then time-shift each of the replicated traces by a random duration between ten minutes and three hours[1]. Figures 4(a) and (b) plot the overprovisioning ratio ρ obtained for different spatial and temporal allocation granularities in a 30 customer system. Like before, ρ increases with increasing spatial and temporal granularities. However, the magnitude of the overallocation is substantially larger when multiplexing a larger number of domains, indicating the need for finer allocation granularities to extract the potential multiplexing gains. This result is also depicted in Figure 5 which plots the overprovisioning ratio as the number of customers in the system is varied. Using a fixed spatial granularity of 0.02, the figure plots ρ for temporal granularities of seconds, minutes and hours for varying number of customers. The figure demonstrates that (i) for a given allocation granularity, the amount of overprovisioning grows with the number of customers, and (ii) the difference in ρ values of fine-grain and coarse-grain allocations (say 10 second and 1 hour) also grows with the number of customers. Thus, the benefits of fine-grain allocations are greater in larger data centers hosting a large number of customers.

To summarize, our results indicate that to efficiently multiplex data center resources, it is desirable to have a fine time granularity as well as a sufficiently small spatial granularity.

3 How Do Existing Architectures Perform?

We now consider several existing architectures for on-demand data centers, which are point solutions in the space-time granularity space, and examine their effectiveness.

Dedicated Hosting Architectures: As explained earlier, dedicated hosting architectures [2, 14, 11] multiplex a pool of servers among multiple customers. This is achieved by partitioning the server pool among applications and increasing or decreasing the number of servers assigned to an application based on its predicted workload. Allocation of a server may involve: (1) deallocation of the server from another customer, (2) disk scrubbing to prevent data leaks; (3) a fresh OS and application installation; and (4) application startup.

Performing these operations can take a thousand seconds or more [11, 14]. Observe from Figure 4(a) that, in a 30 customer system, time multiplexing at a granularity of thousand seconds results in a 368-640% overhead when compared to the optimal fine time-scale multiplexing. Thus, it is possible to extract significant additional gains by reducing the time-scale of server allocations in such architectures.

From the perspective of space multiplexing, these architectures allocate resources at a granularity of entire machines. Since our space multiplexing results use a normalized space metric, the implication of allocating entire machines depends of the peak requirements of the applications. Figure 4(b) indicates that, for a 30 customer system, the

[1]We use a time-shift range of 10 minutes to 3 hours to incorporate time-zone effects that might arise in real web workloads.

multiplexing benefits can be maximized when the allocation granularity is 0.01 (or 1%) of the peak requirement. This is essentially the knee of the curve in Figure 4(b). Assuming that the 30 customers collectively have a peak capacity requirement of 100 servers, allocating entire machines in such a data center is optimal since 1% of the peak requirement is 1 server (an entire machine). On the other hand, if the peak capacity requirement of the 30 applications is smaller than 100 servers, then the optimal allocation granularity (1% of the peak) is less than a server, indicating that fractional server allocation is necessary to maximize the multiplexing benefit. The smaller the peak capacity requirement, the greater the benefit of allocating fractional server resources (and vice versa). Note that, these benefits also depend on the total number of customers in the system—a larger customer base enhances the the benefits of fractional server allocation.

Fast Reinstallations and Reserve Pools: Recent efforts have recognized the need to reduce server allocation times in dedicated hosting platforms and have proposed several techniques to reduce the allocation overheads. For instance, the OS and application installation time can be reduced by using shared remote boot images or software reboots [5, 6]. If the servers use a storage area network and the storage systems support advanced storage functions, then the need for disk scrubbing could be eliminated by using copy-on-write disks. Further, a reserve pool of idle servers can be maintained in energy-saving mode for fast reallocations [11]. Note that if the reserve pool is empty due to a sudden increase in demand, then the overheads of deallocating a server from an existing application and reallocating it to the needy application must still be incurred. Observe also that none of these optimizations eliminate application startup delay. Thus, it appears that these optimizations can reduce the server allocation overheads to a few minutes (about 5-10 minutes). Despite this substantial reduction, Figure 4(a) indicates a 295-600% overhead when compared to optimal allocation and points to the additional gains that can accrue by reducing the multiplexing time-scale to seconds or tens of seconds.

Shared Hosting Platforms: Traditionally shared hosting platforms have been used to run applications whose resource requirements do not warrant an entire server[2]. Shared hosting platforms, by their very nature, allocate fractional server resources to applications. Further, they employ resource management mechanisms such as proportional share schedulers [12] or resource containers [4] to enforce these allocations at a fine time-scale. The allocation of an application can be modified by reconfiguring scheduler parameters such as weights. Whereas some techniques advocate doing so at a granularity of tens of seconds [13], others have focused on very coarse-grain allocations based on worst-case needs [15]. At a first glance, shared hosting platforms appear to provide the necessary flexibility for fine-grain time and space multiplexing of resources. A key limitation though is that they provide less isolation and security between applications when compared to dedicated hosting platforms (where isolation is automatic due to the absence of sharing).

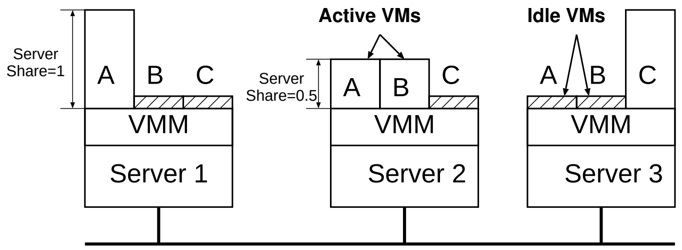


Figure 6: A data center architecture employing virtual machines. The figure shows three applications A, B and C requiring 1.5 servers, 0.5 servers and 1 server, respectively. The applications run in virtual machines on multiple physical servers and the VMs are allocated appropriate resource shares. Each server has some active and some idle VMs; resource shares of VMs are modified dynamically to handle workload variations.

4 A Case for a New Data Center Architecture

Shared hosting systems achieve high utilization of resource at the expense of isolation between applications. Current architectures for dedicated hosting, on the other hand, have high isolation but do not multiplex resources at fine time and space granularity. Thus, an on-demand data center architecture that employs fine-grain space-time multiplexing of resources while retaining the isolation and security advantages of the dedicated hosting model is desirable. We believe that a new hosting platform architecture based on a server partitioning technology such as *virtual machine monitors* [8, 16] or *virtual private servers* [9] can provide these benefits. For simplicity of exposition, we focus on virtual machine monitors in the rest of this section; however our arguments are applicable to virtual private servers as well.

A virtual machine monitor (VMM) virtualizes the hardware on a machine and enables the existence of multiple independent virtual machines (VM). By employing predictable and flexible resource management mechanisms, the virtual machine monitor can allocate different resource shares to different virtual machines [10, 16, 18]. Although virtual machine monitors support multiple applications each requiring less than one server, our architecture employs virtual machine monitors *to support applications that span multiple servers.* In such a system, a set of virtual machines serves a customer; multiple virtual machines, each belonging to a different customer, are hosted on the same server (see Figure 6). Resources allocated to the virtual machines of a customer are adjusted dynamically to handle the variations in the application's workload. Doing so ensures fine-grain space-time resource multiplexing: (i) fractional server allocation is possible by allocating non-zero shares to applications running on the same server, (ii) *implicit* fine-grain time multiplexing is enabled by employing proportional-share schedulers in the virtual machine monitor (which can reallocate unused machine resources at millisecond time-scales), and (iii) *explicit* fine-grain time multiplexing can be achieved by modifying scheduler parameters to specify an application's resource share. Further, virtual machines can provide isolation and security benefits that are comparable to dedicated hosting architectures [18].

[2] This is not a technical limitation, rather it is how shared hosting platforms (e.g., shared web hosting) are being used commercially. It is certainly possible to use shared hosting platforms to run more demanding applications that require multiple machines.

Note that although multiple virtual machines may be mapped onto a physical server, *not all VMs need to be active at the same time.* For instance, it is possible to emulate a dedicated hosting model by simply having one virtual machine active on each machine and keeping the other VMs in an idle state (this is achieved by allocating very small resource shares to the idle VMs and allocating most of the machine resources to the active VM—see Figure 6). An active VM can be deallocated and an idle VM can be "brought online" by simply reducing the resource share of the former and increasing that of the latter. Thus, the "server reallocation" overhead can be reduced to hundreds of milliseconds. No additional application startup delay is incurred since the application is already idling in the VM. Furthermore, when necessary, the benefits of the shared hosting architectures can be accrued by having multiple active VMs on a machine, each using a certain fraction of the server resources (and keeping the remaining VMs in an idle state).

Several research challenges arise when designing such an architecture. The benefits of a virtual machines-based architecture are expected to be high when a large number of virtual machines can be co-located on a machine. At any given time, a small fraction of these virtual machines will be active and the rest will be idle. Hence, scalable virtual machines whose resource shares can be quickly ramped up or down are desirable [18]. Second, to derive high levels of statistical multiplexing gains, an effective placement algorithm that dynamically places virtual machines with *complementary needs* on a shared server are desirable. That is, multiplexing benefits can be maximized by co-locating applications whose peak requirements are non-overlapping in time. Finally, pro-active reallocation of resources requires accurate workload prediction techniques. Doing so at a fine-time scale is especially challenging, since fine time-scale reallocations may result in oscillations or cause the system to react to "noise". Thus, the benefits of fine time-scale provisioning can be extracted only if stable workload prediction techniques are available.

The higher multiplexing gains in our virtual machine architecture come with one disadvantage. In the conventional dedicated hosting architectures, a hardware fault on one server affects a single customer. However, in our architecture, a server failure can effect multiple customers.

The design of such an on-demand hosting platform architecture based on a server partitioning technology is the subject of ongoing research.

Acknowledgements

We would like to thank Bill Tetzlaff for his comments that helped us improve this paper.

References

[1] S. Adler. The Slashdot Effect, An Analysis of Three Internet Publications. Linux Gazette, March 1999.

[2] K. Appleby, S. Fakhouri, L. Fong, M. K. G. Goldszmidt, S. Krishnakumar, D. Pazel, J. Pershing, and B. Rochwerger. Oceano - SLA-based Management of a Computing Utility. In *Proceedings of the IFIP/IEEE Symposium on Integrated Network Management*, May 2001.

[3] M. Aron, P. Druschel, and W. Zwaenepoel. Cluster Reserves: A Mechanism for Resource Management in Cluster-based Network Servers. In *Proceedings of the ACM SIGMETRICS Conference, Santa Clara, CA*, June 2000.

[4] G. Banga, P. Druschel, and J. Mogul. Resource Containers: A New Facility for Resource Management in Server Systems. In *Proceedings of the third Symposium on Operating System Design and Implementation (OSDI'99), New Orleans*, pages 45–58, February 1999.

[5] Boot-NIC. `http://www.vci.com/products/network_centric/bnic.asp`.

[6] G. Candea and A. Fox. Recursive Restartability: Turning the Reboot Sledgehammer into a Scalpel. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, May 2001.

[7] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle. Managing Energy and Server Resources in Hosting Centers. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP)*, pages 103–116, October 2001.

[8] J. Dike. User Mode Linux. In *Proceedings of the 5th Annual Linux Showcase and Conference*, November 2001.

[9] Ensim Linux Private Server. `http://www.ensim.com/solutions/linux/linux.html#PrivateServerTech`.

[10] K. Govil, D. Teodosiu, Y. Huang, and M. Rosenblum. Cellular Disco: Resource Management using Virtual Clusters on Shared-memory Multiprocessors. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP'99)*, pages 154–169, December 1999.

[11] J. Moore and J. Chase. Cluster on Demand. Technical Report CS-2002-07, Department of Computer Science, Duke University, 2002.

[12] A. Parekh and R. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks – The Single Node Case. In *Proceedings of IEEE INFOCOM '92*, pages 915–924, May 1992.

[13] P. Pradhan, R. Tewari, S. Sahu, A. Chandra, and P. Shenoy. An Observation-based Approach Towards Self-Managing Web Servers. In *Proceedings of the Tenth International Workshop on Quality of Service (IWQoS 2002)*, May 2002.

[14] S. Ranjan, J. Rolia, H. Fu, and E. Knightly. QoS-Driven Server Migration for Internet Data Centers. In *Proceedings of the Tenth International Workshop on Quality of Service (IWQoS 2002)*, May 2002.

[15] B. Urgaonkar, P. Shenoy, and T. Roscoe. Resource Overbooking and Application Profiling in Shared Hosting Platforms. In *Proceedings of the Fifth Symposium on Operating System Design and Implementation (OSDI'02)*, December 2002.

[16] C. Waldspurger. Memory Resource Management in VMWare ESX Server. In *Proceedings of the Fifth Symposium on Operating System Design and Implementation (OSDI'02)*, December 2002.

[17] M. Welsh, D. Culler, and E. Brewer. SEDA: An Architecture for Well-Conditioned, Scalable Internet Services. In *Proceedings of the ACM Symposium on Operating Systems Principles*, October 2001.

[18] A. Whitaker, M. Shaw, and S. D. Gribble. Scale and Performance in the Denali Isolation Kernel. In *Proceedings of the Fifth Symposium on Operating System Design and Implementation (OSDI'02)*, December 2002.