Quantifying the Benefits of Resource Multiplexing in On-Demand Data Centers *

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 host multiple applications on server farms by dynamically provisioning resources in response to workload variations. The efficiency of such dynamic provisioning on the required server farm capacity is dependent on several factors — the granularity and frequency of reallocation, the number of applications being hosted, the amount of resource overprovisioning and the accuracy of workload prediction. In this paper, we quantify the effect of these factors on the multiplexing benefits achievable in an on-demand data center. Using traces of real e-commerce workloads, we demonstrate that the ability to allocate fractional server resources at fine time-scales of tens of seconds to a few minutes can increase the multiplexing benefits by 162-188% over coarse-grained reallocation. Our results also show that these benefits increase in the presence of large number of hosted applications as a result of high level of multiplexing. In addition, we demonstrate that such fine-grained multiplexing is achievable even in the presence of real-world (inaccurate) workload predictors and allows overprovisioning slack of nearly 35-70% over coarse-grained multiplexing.

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, customers 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 [2]). Due to such large variations in loads, it is difficult to estimate workload requirements in advance, and hence worst-case resource provisioning is either infeasible or extremely inefficient.

In order to handle such workload variations, many data centers have started employing *self-managing techniques* for resource allocation [4, 10, 12]. These techniques dynamically reallocate resources among hosted applications based on their short-term demand estimates. The goal is to meet the application requirements *on demand* and adapt to their changing resource needs. The main advantages of on-demand resource allocation are that (i) it achieves better resource utilization by extracting multiplexing gains, and (ii) it makes the system more robust to unanticipated workload increases.

Several techniques have been proposed to dynamically provision resources to applications in on-demand data centers [4, 5, 8, 18, 20, 21]. 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, some techniques allocate full servers to applications at a time [4, 18], while others share server resources among multiple applications leading to finer control over server resources [16, 21]. Similarly, these techniques also differ in the time granularity at which they perform reallocation. For instance, security and privacy considerations may require OS re-installation and disk scrubbing on a server prior to its reallocation, resulting in allocation time-scales of tens of minutes [4]. Other schemes reduce the time-scale of reallocation by maintaining a ready pool of servers in energy-saving mode [14], or using techniques like remote booting [17] and fast reallocation [10].

These dynamic allocation schemes also need to estimate future workload requirements for reallocation. The accuracy of such prediction is critically dependent on the frequency of allocation as well as the variability of the workload itself. On-demand data centers also differ in several other aspects such as their size (the number of servers and resources available) and the number of customers or applications they support. In addition, many data centers perform overprovisioning of resources to handle unexpected overload conditions and to prevent the system from running at full capacity. Different data centers provide different degrees of over-allocation based on resource management costs and customer service guarantees.

Several questions need to be answered to understand how various design parameters impact the multiplexing benefits and capacity requirements in on-demand data centers.

1. Should data center resources be allocated to applications at a granularity of entire machines or is the ability to allocate

*This research was supported in part by NSF grants CCR-9984030 and EIA-0080119.

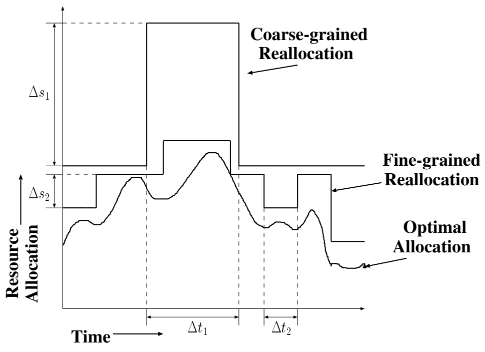


Figure 1: A metric for comparing optimal resource allocation to practical approaches. An optimal approach can reallocate resources infinitely often and in infinitesimally small amounts; a practical approach uses a finite time and space granularity, Δt and Δs, respectively.

fractional servers desirable?

2. Should resources be provisioned over time scales of seconds, minutes, or hours so as to extract the best multiplexing gains?

3. Do the achievable multiplexing gains increase with the number of hosted customers, and if so, by how much?

4. How do overprovisioning and workload prediction accuracy affect the resource allocation?

To answer these questions, in this paper, we conduct a study to understand the impact of these factors on the effective capacity of on-demand data centers. Using traces of real e-commerce workloads, we demonstrate that the ability to allocate fractional server resources and at fine time-scales of seconds to a few minutes can increase the multiplexing benefits by about 162–188% over coarse-grained reallocation. Our results also show that these benefits increase in the presence of large number of hosted applications as a result of high level of multiplexing. In addition, we demonstrate that such fine-grained multiplexing is more efficient even in the presence of inaccurate workload prediction, and allows overprovisioning slack of nearly 35-70% over coarse-grained multiplexing. A parallel effort [3] has also studied similar issues; our study differs from it in several respects and a detailed comparison of the two efforts is presented in Section 4.

The rest of this paper is structured as follows. In Section 2, we quantify the impact of resource multiplexing granularity in on-demand data centers using real e-commerce workloads. We characterize the resource usage efficiency of existing data center architectures in Section 3, and examine related work in Section 4. Finally, we present our conclusions and ongoing work in Section 5.

2. BENEFITS OF RESOURCE MULTIPLEXING

In this section, we present a study based on real web workloads that quantifies the potential multiplexing benefits of dynamic resource allocation based on various factors described above.

2.1 Optimal Allocation and Performance Metrics

Workload	Duration	Number Requests	Avg Request size	Peak bit-rate
Ecommerce1	24 hrs	1,194,137	3.95 KB	458.1 KB/s
Ecommerce2	24 hrs	1,674,672	3.85 KB	1631.0 KB/s
Ecommerce3	24 hrs	251,352	7.24 KB	1346.9 KB/s

Table 1: Workload characteristics

We first define the notion of optimal allocation and define a metric that quantifies the efficiency of an allocation scheme. Figure 1 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). This allocation should be such that the resources allocated in any period are 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 time units and the allocation must always be a multiple of the allocation granularity Δs. We refer to the units Δs and Δt as the *spatial* and *temporal* allocation granularity respectively.

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 total resource allocation in the system at time t for these schemes are $R_{opt}(t) = \sum_i R^i_{opt}(t)$ and $R_{pract}(t) = \sum_i R^i_{pract}(t)$ respectively. Using these quantities, we define the metric *capacity overhead* to quantify the resource usage efficiency of an allocation scheme. The capacity overhead ρ is defined to be the percentage increase in the resource requirement of a practical scheme when compared to the optimal.

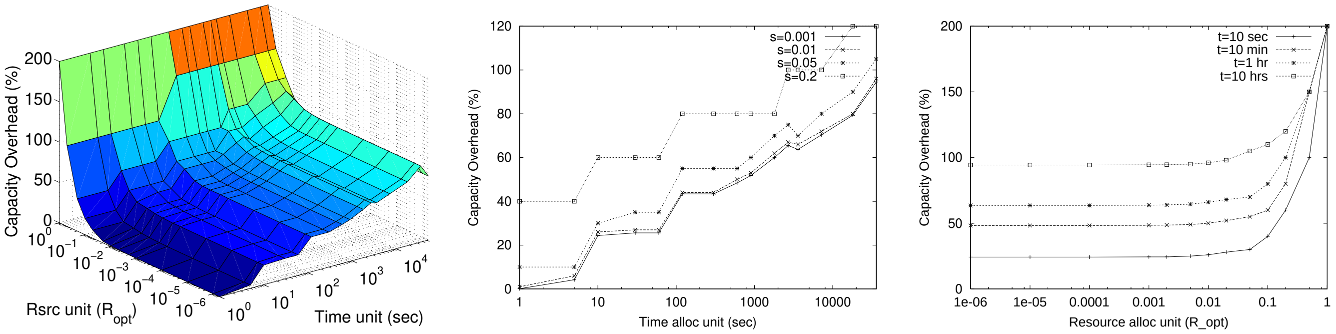
$$\rho = \left(\frac{R_{pract} - R_{opt}}{R_{opt}} \right) \cdot 100,$$

where, $R_{pract} = \max_t R_{pract}(t)$ is the peak resource requirement of the practical scheme (this is essentially the total capacity required to host this set of applications). Similarly, $R_{opt} = \max_t R_{opt}(t)$ is the peak capacity requirement of the optimal scheme. For an allocation curve in Figure 1, R corresponds to the peak value of the curve.

Intuitively, ρ measures the additional capacity required by a practical scheme to host the same set of applications as the optimal scheme. For instance, a ρ value of 50 corresponds to a requirement of 150 servers when the optimal requirement is 100 servers. Thus, the *smaller* the value of ρ, the *more efficient* is a scheme in terms of its resource usage.

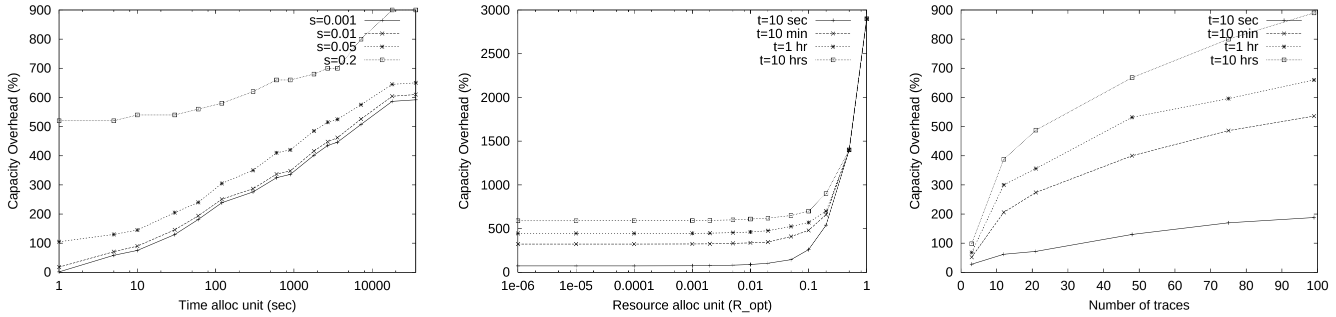
2.2 Effect of Allocation Granularity

To quantify the metric defined above for different allocation granularities, we conduct a study using 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 of a request was not available. Since for static web requests, CPU usage is highly correlated with the request size, we use request size as a proxy metric for CPU usage.



(a) Capacity overhead (b) Temporal capacity overhead (c) Spatial capacity overhead

Figure 2: Effect of allocation granularity on the capacity overhead for a 3 customer system.



(a) Temporal capacity overhead, 30 customers (b) Spatial capacity overhead, 30 customers (c) Impact of varying number of customers

Figure 3: Statistical multiplexing of resources in data centers with large number of customers.

We systematically vary the spatial (Δs) and temporal (Δt) allocation granularity for this workload mix and compute the value of ρ for each combination. We use a granularity of 1 second and 1 byte/sec to approximate the optimal allocation scheme. We express the spatial granularity Δs as a fraction of the peak requirement of the optimal scheme R_{opt}. Thus, if the optimal peak capacity requirement is 100 servers, then a spatial granularity of 0.01 indicates that resources are allocated 1 server at a time. Initially, we assume each resource allocation scheme to be *clairvoyant*, i.e., it allocates resources based on exact knowledge of future workload requirements. This assumption eliminates the impact of inaccuracies introduced by workload predictors. The impact of inaccuracies introduced by real-world workload predictors is studied in Section 2.4.

Figure 2 shows the values of capacity overhead ρ for different Δt and Δs values. Figure 2(a) shows that the coarser the spatial and temporal allocation granularity, the greater the capacity overhead ρ (indicating larger over-allocations at coarser allocation granularities). Next, we examine the effect of varying Δt and Δs in isolation on the capacity overhead (see Figures 2(b) and 2(c)).

Figure 2(b) 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. For instance, with $\Delta s = 0.02$, reallocating resources once every 10 sec, 10 min, 1 hour and 10 hours yields ρ values of 28%, 52%, 68% and 98% respectively. In addition, we find that there are ranges of Δt values within which the ρ values are nearly constant. These ranges are 10 sec-1 min, 2-5 mins, 10-15 mins and 30-120 mins respectively. These results argue *in favor of having a small Δt value in the range of a few seconds to a few minutes*.

In contrast, when the Δs value is varied (see Figure 2(c)), we find that ρ is nearly constant until a certain granularity after which it increases steadily with increasing Δs. For instance, with $\Delta t = 1$ minute, ρ is nearly constant at 26% until $\Delta s = 0.005$, and increases to 35%, 40%, 60% and 100% with Δs values of 0.05, 0.1, 0.2 and 0.5, respectively. Further, Δs values close to the total resource requirement yield very large capacity overheads regardless of the Δt value. This result implies that *while the spatial granularity need not be very fine-grained, it should still be sufficiently small in order to extract high multiplexing gains*.

2.3 Effect of Number of Customers

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 capacity overhead, 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 3(a) and (b) plot the capacity overhead ρ

[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.

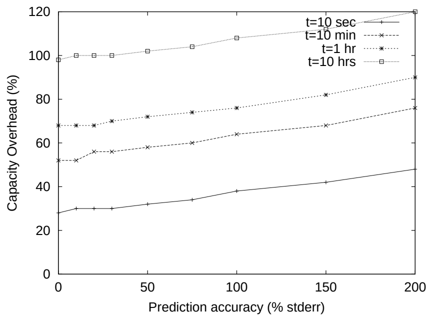


Figure 4: Effect of prediction inaccuracies on resource multiplexing

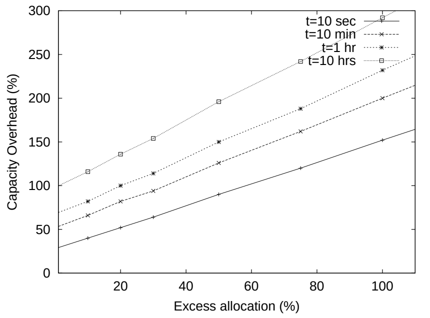


Figure 5: Effect of overprovisioning on resource multiplexing

obtained for different spatial and temporal allocation granularities in a 30-customer system. Like before, ρ increases with increasing Δs and Δt values. 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 3(c) which plots the capacity overhead as the number of customers in the system is varied. Using a fixed Δs value of 0.02, the figure plots ρ for Δt values of seconds, minutes and hours with varying number of customers. The figure demonstrates that (i) for a given allocation granularity, the amount of consumption overhead grows with the number of customers, and (ii) the difference between 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 magnified in data centers hosting a large number of customers*.

2.4 Effect of Prediction Inaccuracy and Overprovisioning

Thus far, our study has made two idealized assumptions: (i) resource allocators are assumed to be clairvoyant, i.e, they can predict the exact resource requirement for the next allocation interval, and (ii) the allocation is exact, leaving for no "headroom". These assumptions do not hold in real systems. Real workload predictors will be inaccurate, and since even good predictors will be unable to predict sudden, unanticipated workload variations, data centers overprovision resources to leave some "headroom" for such events. Consequently, we study the effects of prediction inaccuracies and overprovisioning.

In general, any dynamic allocation scheme estimates future workload requirements using a prediction algorithm. The prediction algorithm can introduce inaccuracies in its estimates, which in turn impacts the achievable resource utilization. Note that the need for prediction at very fine time-scales (in the order of seconds) can be avoided by employing work-conserving schedulers in the underlying OS (which automatically allocate unused resources to needy applications). Consequently, workload prediction is necessary only for coarser time-scales ranging from minutes to hours, for which reasonably accurate predictors exist [22]. Nevertheless, we still consider the effect of prediction inaccuracies across *all* time-scales.

First, we look at the effect that prediction inaccuracies can have on resource allocation. Instead of using specific prediction algorithms for our study, we characterize a generic prediction algorithm by its *prediction accuracy*. We define a predictor to have a prediction accuracy δ if its 95-percentile prediction error is bounded by $(\pm\delta \cdot \sigma_x)$, where σ_x is the standard deviation of the workload being predicted. By characterizing the prediction accuracy as a ratio of the standard deviation of the workload, we intend to capture the effect of the workload variability on the prediction errors. The intuition behind doing this is that a predictor is expected to be less accurate for a more variable and bursty workload, so that its prediction accuracy would depend on the variability of the workload itself.

For a predictor with an accuracy δ, the worst-case allocation would happen if it always predicted the maximum resource requirement within its accuracy, so that it would always allocate $(\delta \cdot \sigma_x)$ more resources than the actual requirement. In Figure 4, we show the effect of the prediction accuracy using such a worst-case allocation on the multiplexing benefits for different time-scale granularities with three customers. The interesting observation from the figure is that *even using a very inaccurate predictor at fine time scales gives better resource multiplexing benefits than using an accurate predictor at coarse time-scales*.

Even under the assumption of a clairvoyant predictor, most data centers overprovision their resources, i.e., they allocate resources in excess of the estimated requirement. This is done in order to handle unforeseen loads and to absorb prediction errors. In addition, overprovisioning is also done to prevent the system from running close to full capacity, and to provide some "head room" for overload protection. In such a scenario, a dynamic resource allocation scheme would allocate a certain amount of extra resource over the estimated requirement. Figure 5 plots the effect of varying the amount of over-allocation on the capacity overhead ρ. This figure indicates that the capacity overhead increases drastically as we increase the excess allocated capacity or "head room". But the key observation from the figure is that for the same multiplexing gains, allocation at fine time-scales of about 10 sec allows nearly 35-70% more head room than that allowed by coarse-grained allocation at the granularity of 1-10 hours.

These results show that it is possible to extract the multiplexing benefits at fine granularities despite prediction inaccuracies and resource overprovisioning.

3. PERFORMANCE OF DATA CENTER ARCHITECTURES

In this section, we quantify the benefits of some common on-demand architectures, which are essentially point solutions in the design space considered in our study.

Data Center Configuration	Optimal reqmt (Num servers)	Num customers	Dedicated Architecture			Fast Reallocation			Shared Architecture		
			Δs (R_{opt})	Δt (sec)	Num servers	Δs (R_{opt})	Δt (sec)	Num servers	Δs (R_{opt})	Δt (sec)	Num servers
Small	20	3	0.05	1800	34	0.05	300	31	0.005	10	25
Medium	100	15	0.01	1800	388	0.01	300	304	0.001	10	148
Large	1000	30	0.001	1800	5017	0.001	300	3759	0.0001	10	1739

Table 2: Resource requirements of data center architectures and reallocation techniques for different data center configurations.

Dedicated Architecture: A dedicated architecture [4, 18] multiplexes a pool of servers among multiple customers. This is achieved by partitioning the server pool among customers and increasing or decreasing the number of servers assigned to a customer based on its predicted workload. This architecture allocates resources at a granularity of entire machines. Thus, the spatial granularity Δs for these architectures is inversely proportional to the total resource requirement of the hosted customers. For instance, a dedicated architecture would have a Δs value of 0.01 when the optimal capacity requirement is 100 servers. Reallocation of a server in this architecture 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 time on the order of several minutes. Hence, the temporal granularity Δt of such an architecture could be considered to be about 30 minutes.

Fast Reinstallations and Reserve Pools: Recent efforts have recognized the need to reduce server allocation times in the dedicated architecture and have proposed several techniques to reduce these reallocation overheads. For instance, the OS and application installation time can be reduced by using remote boot images [17], fast application switching [10], and by maintaining a reserve pool of idle servers in energy-saving mode [14]. While these techniques have the same spatial granularity as above, they reduce the Δt values to an order of about 5 minutes.

Shared Architecture: A shared architecture hosts multiple applications on each server and multiplexes the server resources among these applications [21]. A shared architecture, by its very nature, allocates fractional server resources to applications. Thus, it uses small Δs values corresponding to about 10% of a server capacity. Further, it employs resource management mechanisms such as proportional share schedulers [15] or resource containers [6] to enforce these allocations at fine time-scales. The allocation of an application can be modified by reconfiguring scheduler parameters such as weights or shares, and the work-conserving nature of the underlying scheduler can also be exploited to achieve resource reallocation at time-scales of a few seconds [16]. The main limitation of a shared architecture is its low degree of security and isolation as compared to the dedicated architecture.

Having described some of the data center architectures and reallocation mechanisms commonly deployed, we now quantify the potential capacity requirements of these architectures on data centers with different configurations. In Table 2, we describe three data center configurations corresponding to a small, a medium-size and a large data center. These configurations correspond to different number of customers and total optimal resource requirements of these customers. For each of these configurations, the table shows the allocation granularity for the various architectures and allocation techniques described above. Finally, the table shows the number of servers that would be required by each architecture corresponding to these configurations.

The results in Table 2 indicate that the excess capacity requirement is low for a small data center configuration for all the architectures. For the medium and large configurations, even though the requirement of a shared architecture is much greater than the optimal requirement, it is still respectively about 162% and 188% less compared to that for a dedicated architecture. Fast reallocation techniques reduce this overhead to about 105% and 116% respectively. Thus, these results show that sharing fractional resources between applications at the time-granularity of a few seconds to a few minutes is desirable.

4. RELATED WORK

A concurrent study [3] has examined similar questions about the potential gains in resource usage in utility computing models. While this study compares different utility computing environments such as a large multiprocessor server and a cluster of small servers, we compare different data center architectures with varying degree of allocation granularity applied within the same data center environment. Although this study uses a larger data set compared to ours, its results agree with our results in showing that existing data centers are grossly under-utilized and some utility models can achieve high multiplexing gains. The two studies also examine a different set of issues. For instance, while this study has considered the effect of workload affinity on allocation, we have examined factors such as the number of hosted customers, workload prediction accuracy and the amount of overprovisioning.

In the previous section, we broadly classified data center architectures as dedicated and shared. However, other architectures have been proposed that have aspects of both these architectures. Moore et al. [14] have proposes an hierarchical architecture that allocates virtual clusters to a group of applications. Servers can be reallocated among these clusters, and the applications within each cluster could be managed by a shared resource manager such as MUSE [8]. Another architecture is described in [19] that makes use of virtual clusters to allocate resources across globally distributed data centers. As these architectures incorporate some aspects of both dedicated and shared architectures, we believe their multiplexing gains would lie somewhere between those achievable by purely dedicated and shared architectures.

Prediction accuracy could be important for the dynamic resource allocation techniques as well. Prediction and workload characterization techniques have been proposed that work well for online prediction at coarse time-granularities of several minutes to hours [11, 22]. With shared architectures, it is possible to exploit the work-conserving nature of underlying schedulers to preclude the need for accurate prediction at time-scales of a few seconds.

Recently, several dynamic resource allocation techniques have been proposed for data centers that use modeling techniques to achieve resource guarantees [1, 7, 9, 13]. These techniques use measurement and prediction techniques to reallocate resources among applications based on their varying workload.

5. CONCLUSIONS AND ONGOING WORK

In this paper, we quantified the multiplexing benefits achievable in on-demand data centers. We used real web workloads to study the effect of various factors on these benefits. These factors include the granularity and frequency of reallocation, the number of customers being hosted, the amount of resource overprovisioning and workload prediction accuracy. Our results demonstrated that fine-grained multiplexing at short time-scales of the order of seconds to a few minutes combined with fractional server allocation leads to substantial multiplexing gains over coarse-grained reallocation. Our results also demonstrated that these gains increase with increasing number of hosted applications as a result of high level of multiplexing. In addition, we demonstrated that such fine-grained multiplexing is more efficient even in the presence of inaccurate workload prediction, and allows overprovisioning slack of nearly 35-70% over coarse-grained multiplexing for similar multiplexing gains.

As part of ongoing work, we are exploring several other issues related to resource multiplexing in data centers. These issues include the effect of application affinity on server allocation and application placement. For instance, certain application components may need to be placed together or different applications might need to be placed separately due to security concerns, leading to constraints on allocation granularity. Another important issue is the impact of additional resource allocation on application performance. We have assumed a linear relation between the amount of resource allocation and the performance of an application. Such an assumption may not hold because adding more resources could lead to performance penalties such as more communication costs for distributed applications, loss of cache affinity, load balancing overhead, etc. Finally, the cost of reallocation includes not only the cost of performing the actual allocation, but also that of *computing* the allocations. This computation cost can add to the reallocation cost and constrain the time granularity of allocation. We intend to address some of these issues in addition to refining our existing results by incorporating more sophisticated models of prediction accuracies and reallocation overhead.

6. REFERENCES

[1] T. Abdelzaher, K. G. Shin, and N. Bhatti. Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach. *IEEE Transactions on Parallel and Distributed Systems*, 13(1), Jan. 2002.

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

[3] A. Andrzejak, M. Arlitt, and J. Rolia. Bounding the Resource Savings of Utility Computing Models. Technical Report HPL-2002-339, HP Labs, Dec. 2002.

[4] 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.

[5] 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.

[6] 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.

[7] A. Chandra, W. Gong, and P. Shenoy. Dynamic Resource Allocation for Shared Data Centers Using Online Measurements. In *Proceedings of IWQoS'03. To appear*, June 2003.

[8] 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.

[9] R. Doyle, J. Chase, O. Asad, W. Jin, and A. Vahdat. Model-Based Resource Provisioning in a Web Service Utility. In *Proceedings of USITS'03*, Mar. 2003.

[10] Ejasent upscale data center. `http://www.ejasent.com/platform.shtml`.

[11] J. Hellerstein, F. Zhang, and P. Shahabuddin. A Statistical Approach to Predictive Detection. *Computer Networks*, Jan. 2000.

[12] HP Utility Data Center. `http://h30046.www3.hp.com/solutions/utilitydata.html`.

[13] Z. Liu, M. Squillante, and J. Wolf. On Maximizing Service-Level-Agreement Profits. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, 2001.

[14] J. Moore, D. Irwin, L. Grit, S. Sprenkle, and J. Chase. Managing Mixed-Use Clusters with Cluster-on-Demand. Technical report, Department of Computer Science, Duke University, Nov. 2002.

[15] 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.

[16] 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.

[17] Intel Preboot Execution Environment (PXE). `http://www.intel.com/labs/manage/wfm/tools/pxesdk20/index.htm`.

[18] 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.

[19] J. Rolia, S. Singhal, and R. Friedrich. Adaptive internet data centers. In *Proceedings of SSGRR 2000*, July 2000.

[20] K. Shen, H. Tang, T. Yang, and L. Chu. Integrated Resource Management for Cluster-based Internet Services. In *Proceedings of the Fifth Symposium on Operating System Design and Implementation (OSDI'02)*, Dec. 2002.

[21] 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)*, Dec. 2002.

[22] F. Zhang and J. L. Hellerstein. An approach to on-line predictive detection. In *Proceedings of MASCOTS 2000*, Aug. 2000.