Cicada: Introducing Predictive Guarantees for Cloud Networks

Katrina LaCurts*, Jeffrey C. Mogul†, Hari Balakrishnan*, Yoshio Turner‡
*MIT CSAIL, †Google, Inc., ‡HP Labs

Abstract

In cloud-computing systems, network-bandwidth guarantees have been shown to improve predictability of application performance and cost [1, 28]. Most previous work on cloud-bandwidth guarantees has assumed that cloud tenants know what bandwidth guarantees they want [1, 17]. However, as we show in this work, application bandwidth demands can be complex and time-varying, and many tenants might lack sufficient information to request a guarantee that is well-matched to their needs, which can lead to over-provisioning (and thus reduced cost-efficiency) or under-provisioning (and thus poor user experience).

We analyze traffic traces gathered over six months from an HP Cloud Services datacenter, finding that application bandwidth consumption is both time-varying and spatially inhomogeneous. This variability makes it hard to predict requirements. To solve this problem, we develop a prediction algorithm usable by a cloud provider to suggest an appropriate bandwidth guarantee to a tenant. With tenant VM placement using these *predictive guarantees*, we find that the inter-rack network utilization in certain datacenter topologies can be more than doubled.

1 INTRODUCTION

This paper introduces **predictive guarantees**, a new abstraction for bandwidth guarantees in cloud networks. A provider of predictive guarantees observes traffic along the network paths between virtual machines (VMs), and uses those observations to *predict* the data rate requirements over a future time interval. The cloud provider can offer a tenant a guarantee based on this prediction.

Why should clouds offer bandwidth guarantees, and why should they use *predictive guarantees* in particular? Cloud computing infrastructures, especially public "Infrastructure-as-a-Service" (IaaS) clouds, such as those offered by Amazon, HP, Google, Microsoft, and others, are being used not just by small companies, but also by large enterprises. For distributed applications involving significant inter-node communication, such as in [1], [23], and [24], current cloud systems fail to offer even basic network performance guarantees; this inhibits cloud use by enterprises that must provide service-level agreements.

Others have proposed mechanisms to support cloud bandwidth guarantees (see §2); with few exceptions, these works assume that tenants know what guarantee they want, and require the tenants to explicitly specify bandwidth requirements, or to request a specific set of network resources [1, 17]. But do cloud customers really know their network requirements? Application bandwidth demands can be complex and time-varying, and not all application owners accurately know their bandwidth demands. A tenant's lack of accurate knowledge about its *future bandwidth demands* can lead to over- or under-provisioning.

For many (but not all) cloud applications, future bandwidth requirements are in fact predictable. In this paper, we use network traces from HP Cloud Services [11] to demonstrate that tenant bandwidth demands can be time-varying and spatially inhomogeneous, but can also be predicted, based on automated inference from their previous history.

We argue that predictive guarantees provide a better abstraction than prior approaches, for three reasons. First, the predictive guarantee abstraction is simpler for the tenant, because the provider automatically predicts a suitable guarantee and presents it to the tenant.

Second, the predictive guarantee abstraction supports time- and space-varying demands. Prior approaches typically offer bandwidth guarantees that are static in at least one of those respects, but these approaches do not capture general cloud applications. For example, we expect *temporal variation* for user-facing applications with diurnally-varying workloads, and *spatial variation* in VM-to-VM traffic for applications such as three-tier services.

Third, the predictive guarantee abstraction easily supports fine-grained guarantees. By generating guarantees automatically, rather than requiring the tenant to specify them, we can feasibly support a different guarantee on each VM-to-VM directed path, and for relatively short

time intervals. Fine-grained guarantees are potentially more efficient than coarser-grained guarantees, because they allow the provider to pack more tenants into the same infrastructure.

Recent research has addressed these issues in part. For example, Oktopus [1] supports a limited form of spatial variation; Proteus [28] supports a limited form of temporal-variation prediction. But no prior work, to our knowledge, has offered a comprehensive framework for cloud customers and providers to agree on efficient network-bandwidth guarantees, for applications with time-varying and space-varying traffic demands.

We place predictive guarantees in the concrete context of **Cicada**[1], a system that implements predictive guarantees. Cicada observes a tenant's past network usage to predict its future network usage, and acts on its predictions by offering predictive guarantees to the customer, as well as by placing (or migrating) VMs to increase utilization and improve load balancing in the provider's network. Cicada is therefore beneficial to both the cloud provider and cloud tenants. More details about Cicada are available in [15].

2 Related Work

Throughout, we use the following definitions. A **provider** refers to an entity offering a public cloud (IaaS) service. A **customer** is an entity paying for use of the public cloud. We distinguish between customer and **tenant**, as tenant has a specific meaning in OpenStack [20]: operationally, a collection of VMs that communicate with each other. A single customer may be responsible for multiple tenants. Finally, **application** refers to software run by a tenant; a tenant may run multiple applications at once.

Recent research has proposed various forms of cloud network guarantees. Oktopus supports a two-stage "virtual oversubscribed cluster" (VOC) model [1], intended to match a typical application pattern in which clusters of VMs require high intra-cluster bandwidth and lower inter-cluster bandwidth. VOC is a hierarchical generalization of the *hose model* [5]; the standard hose model, as used in MPLS, specifies for each node its total ingress and egress bandwidths. The finer-grained *pipe model* specifies bandwidth values between each pair of VMs. CloudMirror allows tenants to specific a tenant abstraction graph, which reflects the structure of the application itself [17]. Cicada supports any of these models, and unlike these systems, does *not* require the tenant to specify network demands up front.

The Proteus system [28] profiles specific MapReduce jobs at a fine time scale, to exploit the predictable phased behavior of these jobs. It supports a "temporally interleaved virtual cluster" model, in which multiple MapReduce jobs are scheduled so that their network-intensive phases do not interfere with each other. Proteus assumes uniform all-to-all hose-model bandwidth requirements during network-intensive phases, although each such phase can run at a different predicted bandwidth. Proteus (and other related work [13, 19]) does not generalize to a broad range of enterprise applications as Cicada does.

Hajjat et al. [9] describe a technique to decide which application components to place in a cloud datacenter, for hybrid enterprises where some components remain in a private datacenter. Their technique tries to minimize the traffic between the private and cloud datacenters, and hence recognizes that inter-component traffic demands are spatially non-uniform. In contrast to Cicada, they do not consider time-varying traffic nor how to predict it, and they focus primarily on the consequences of wide-area traffic, rather than intra-datacenter traffic.

Cicada does not focus on the problem of enforcing guarantees [8, 16, 22, 25] nor the tradeoff between guarantees and fairness [21]. Though these issues would arise for a provider using Cicada, they can be addressed with any of the above techniques.

3 The Design of Cicada

The goal of Cicada is to free tenants from choosing between under-provisioning for peak periods, or overpaying for unused bandwidth. Predictive guarantees permit a provider and tenant to agree on a guarantee that varies in time and/or space. The tenant can get the network service it needs at a good price, while the provider can avoid allocating unneeded bandwidth and can amortize its infrastructure across more tenants.

3.1 An Overview of Cicada

Cicada has several components, corresponding to the steps in Figure 1. After determining whether to admit a tenant—taking CPU, memory, and network resources into account—and making an initial placement (steps 1 and 2), Cicada measures the tenant's traffic (step 3), and delivers a time series of traffic matrices to a logically centralized controller. The controller uses these measurements to predict future bandwidth requirements (step 4). In most cases, Cicada converts a bandwidth prediction into an offered guarantee for some future interval. Customers may choose to accept or reject Cicada's predictive guarantees (step 5). Because Cicada collects measurement data continually, it can make new predictions and offer new guarantees throughout the lifetime of the tenant.

Cicada interacts with other aspects of the provider's infrastructure and control system. The provider needs to rate-limit the tenant's traffic to ensure that no tenant undermines the guarantees sold to other customers. We distinguish between *guarantees* and *limits*. If the provider's limit is larger than the corresponding guarantee, tenants

[1] Thanks to Dave Levin, Cicada is an acronym: Cicada Infers Cloud Application Demands Automatically.

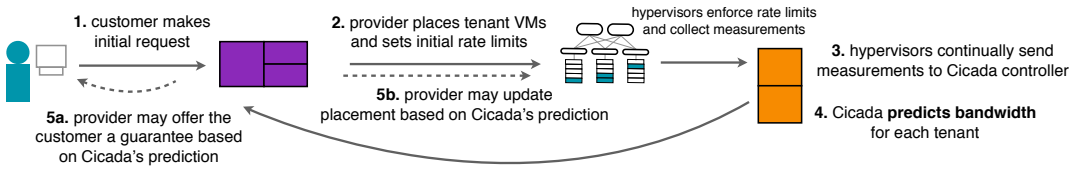

Figure 1: Cicada's architecture.

can exploit best-effort bandwidth beyond their guarantees.

A provider may wish to place VMs based on their associated bandwidth guarantees, to improve network utilization. We describe a VM placement method in §6. The provider could also migrate VMs to increase the number of guarantees that the network can support [6].

3.2 Assumptions

In order to make predictions, Cicada needs to gather sufficient data about a tenant's behavior. Based on our evaluation, Cicada may need at least an hour or two of data before it can offer useful predictions.

Any shared resource that provides guarantees must include an admission control mechanism, to avoid making infeasible guarantees. We assume that Cicada will incorporate network admission control using an existing mechanism [1, 12, 14]. We also assume that the cloud provider has a method to enforce guarantees, such as [21].

3.3 Measurement Collection

Cicada collects a time series of traffic matrices for each tenant, either passively, by collecting NetFlow or sFlow data at switches within the network, or by using an agent that runs in the hypervisor of each machine. We would like to base predictions on offered load, but when the provider imposes rate limits, we risk underestimating peak loads that exceed those limits, and thus under-predicting future traffic. We can detect when a VM's traffic is rate-limited via packet drops, so these underestimates can be detected, too, although not precisely quantified. Currently, Cicada does not account for this potential error.

3.4 Prediction Model

Some applications, such as backup or database ingestion, require bulk bandwidth, and need guarantees that the average bandwidth over a period of H hours will meet their needs (e.g., a guarantee that two VMs will be able to transfer 3.6Gbit in an hour, but not necessarily at a constant rate of 1Mbit/s). Other applications, such as user-facing systems, require guarantees for peak bandwidth over much shorter intervals (e.g., a guarantee that two VMs will be able to transfer at a rate of 1Mbit/s for the next hour).

Cicada's predictions describe the maximum bandwidth expected during any averaging interval δ during a given time interval H. If $\delta = H$, the prediction is for the bandwidth requirement averaged over H hours, but for an interactive application, δ might be just a few milliseconds.

Note that the predictive guarantees offered by Cicada are limited by any caps set by the provider; thus, a proposed guarantee might be lower than suggested by the prediction algorithm.

A predictive guarantee entails some risk of either under-provisioning or over-provisioning, and different tenants will have different tolerances for these risks, typically expressed as a percentile (e.g., the tenant wants sufficient bandwidth for 99.99% of the 10-second intervals). Cicada's prediction algorithm also recognizes conditions under which the prediction is unreliable, in which case Cicada does not propose a guarantee for a tenant.

3.5 Recovering from Faulty Predictions

Cicada's prediction algorithm may make faulty predictions because of inherent limitations or insufficient prior information. Because Cicada continually collects measurements, it can detect when its current guarantee is inappropriate for the tenant's current network load.

When Cicada detects a faulty prediction, it can take one of many actions: stick to the existing guarantee, propose a new guarantee, upgrade the tenant to a higher, more expensive guarantee, etc. How and whether to upgrade guarantees, as well as what to do if Cicada over-predicts, is a pricing-policy decision, and outside our scope. We note, however, that typical System Level Agreements (SLAs) include penalty clauses, in which the provider agrees to remit some or all of the customer's payment if the SLA is not met.

4 Measurement Results

We designed Cicada under the assumption that the traffic from cloud tenants is predictable, but in ways that are not captured by existing models. Before building Cicada, we collected data from HP Cloud Services, to analyze the spatial and temporal variability of its tenants' traffic.

4.1 Data

We have collected sFlow [27] data from HP Cloud Services, which we refer to as the HPCS dataset. Our dataset consists of about six months of samples from 71 Top-of-Rack (ToR) switches. Each ToR switch connects to either 48 servers via 1GbE NICs, or 16 servers via 10GbE NICs. In total, the data represents about 1360 servers, spanning several availability zones. This dataset differs from those in previous work—such as [2, 3, 7]—in that it captures VM-to-VM traffic patterns.

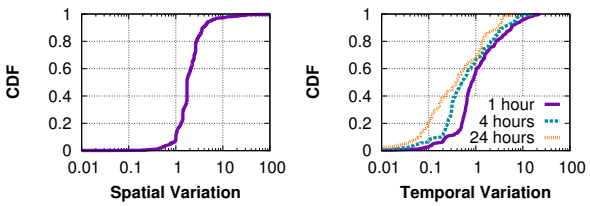

Figure 2: Variation in the HPCS dataset.

We aggregate the data over five-minute intervals, to produce datapoints of the form ⟨*timestamp, source, destination, number of bytes transferred from source to destination*⟩. We keep only the datapoints where both the source and destination are private IPs of virtual machines, and thus all our datapoints represent traffic *within* the datacenter. Making predictions about traffic traveling outside the datacenter or between datacenters is a challenging task, which we do not address in this work.

Under the agreement by which we obtained this data, we are unable to reveal information such as the total number of tenants, the number of VMs per tenant, or the growth rates for these values.

4.2 Spatial and Temporal Variability

To quantify spatial variability, we compare the observed tenants to a static, all-to-all tenant. This tenant is the easiest to make predictions for: every intra-tenant connection sends the same amount of data at a constant rate. Let F_{ij} be the fraction of this tenant's traffic sent from VM_i to VM_j. For this "ideal" all-to-all tenant, all of these values are equal; every VM sends the same amount of data to every other VM. As another example, consider a bimodal tenant with some pairs that never converse and others that converse equally. Let $k = n/2$. Each VM communicates with k VMs, and sends $1/k$ of its traffic; hence half of the F values are $1/k$, and the other half are zero.

For each tenant in the HPCS dataset, we calculate the distribution of its F values, and plot the coefficient of variation in Figure 2. The median *cov* value is 1.732, which suggests nontrivial overall spatial variation (for contrast, the *cov* of our ideal tenant is zero, and the *cov* of our bimodal tenant is one[2]).

To quantify temporal variability, we first pick a time interval H. For each consecutive H-hour interval, we calculate the sum of the total number of bytes sent between each pair p of VMs, which gives us a distribution T_p of bandwidth totals. We then compute the coefficient of variation for this distribution, cov_p. The temporal variation for a tenant is the weighted sum of these values. The weight for cov_p is the bandwidth used by pair p, to reflect the notion that tenants where only one small flow changes over time are less temporally-variable than those where one large flow changes over time.

[2]For the bimodal tenant, $\mu = 1/2k$ and $\sigma = \sqrt{1/n \sum_{i=1}^{n} (1/2k)^2} = 1/2k$.

For each tenant in the HPCS dataset, we calculate its temporal variation value, and plot the CDF in Figure 2. As with spatial variation, most tenants have high temporal variability. This variability decreases as we increase the time interval H, but we see variability at all time scales.

These results indicate that a rigid model is insufficient for making traffic predictions. Given the high spatial *cov* values in Figure 2, a less strict but not entirely general model such as VOC [1] may not be sufficient either. Furthermore, the high temporal variation values indicate that static models cannot accurately represent the traffic patterns of the tenants in the HPCS dataset.

5 Cicada's Traffic Prediction Method

Much work has been done on predicting traffic matrices from noisy measurements such as link-load data [26, 29]. In these scenarios, prediction approaches such as Kalman filters and Hidden Markov Models—which try to estimate true values from noisy samples—are appropriate. Cicada, however, knows the exact traffic matrices observed in past epochs; its problem is to predict a *future* traffic matrix.

Cicada's prediction algorithm adapts Herbster and Warmuth's "tracking the best expert" idea [10], which has been successfully adapted before in wireless power-saving and energy reduction contexts [4, 18]. To predict the traffic matrix for epoch $n+1$, we use all previously observed traffic matrices, $M_1, \ldots, M_n$ (below, we show that data from the distant past can be pruned away without affecting accuracy). In such a matrix, the entry in row i and column j specifies the number of bytes that VM i sent to VM j in the corresponding epoch (for predicting average bandwidth) or the maximum observed over a δ-length interval (for peak bandwidth).

The algorithm assigns each of the previous matrices a weight as part of a linear combination. The result of this combination is the prediction for M_{n+1}. Over time, these weights get updated; the higher the weight of a previous matrix, the more important that matrix is for prediction. In the HPCS dataset, we found that the 12 most recent hours of traffic are weighted heavily, and there is also a spike at 24 hours earlier. Weights are vanishingly small prior to this. Thus, at least in our dataset, one does not need weeks' worth of data to make reliable predictions.

We refer the reader to [15] for a full description of our prediction algorithm, including the method for updating the weights, as well as an evaluation against a VOC-style prediction algorithm [1]. To summarize those results, Cicada's prediction method decreases the median relative error by 90% compared to VOC when predicting either average or peak bandwidth. Cicada also makes predictions quickly. In the HPCS dataset, the mean prediction speed is under 10 msec in all but one case (under 25 msec in all cases).

Algorithm 1 Cicada's VM placement algorithm

```
P = set of VM pairs, in descending order of their bandwidth predictions
for (src, dst) ∈ P do
    if resources aren't available to place (src, dst) then
        revert reservation
        return False
    A = All available paths
    if src or dst has already been placed then
        restrict A to paths including the appropriate endpoint
    Place (src, dst) on the path in A with the most available bandwidth
```

6 Evaluation

Although Cicada's predictive guarantees allow cloud providers to decrease wasted bandwidth, it is not clear that these providers treat wasted intra-rack and inter-rack bandwidth equally. Inter-rack bandwidth may cost more, and even if intra-rack bandwidth is free, over-allocating network resources on one rack can prevent other tenants from being placed on the same rack.

To determine whether wasted bandwidth is intra- or inter-rack, we need a VM-placement algorithm. For VOC, we use the placement algorithm detailed in [1], which tries to place clusters on the smallest subtree that will contain them. For Cicada, we developed a greedy placement algorithm, detailed in Algorithm 1. This algorithm is similar in spirit to VOC's placement algorithm; Cicada's algorithm tries to place the most-used VM pairs on the highest-bandwidth paths, which in a typical datacenter corresponds to placing them on the same rack, and then the same subtree. However, since Cicada uses fine-grained, pipe-model predictions, it has the potential to allocate more flexibly; VMs that do not transfer data to one another need not be placed on the same subtree, even if they belong to the same tenant.

We compare Cicada's placement algorithm against VOC's on a simulated physical infrastructure with 71 racks with 16 servers each, 10 VM slots per server, and (10G/O_p)Gbps inter-rack links, where O_p is the physical oversubscription factor. For each algorithm, we select a random tenant, and use the *ground truth* data to determine this tenant's bandwidth needs for a random hour of its activity, and place its VMs. We repeat this process until 99% of the VM slots are filled. Using the ground-truth data allows us to compare the placement algorithms explicitly, without conflating this comparison with prediction errors. To get a sense of what would happen with more network-intensive tenants, we also evaluated scenarios where each VM-pair's relative bandwidth use was multiplied by a constant bandwidth factor (1×, 25×, or 250×).

Figure 3 shows how the available inter-rack bandwidth—what remains unallocated after the VMs are placed—varies with O_p, the physical oversubscription factor. In all cases, Cicada's placement algorithm leaves more inter-rack bandwidth available. When O_p is greater than two, both algorithms perform comparably, since this is a constrained environment with little bandwidth available overall. However, with lower over-subscription factors, Cicada's algorithm leaves more than twice as much bandwidth available, suggesting that it uses network resources more efficiently in this setting.

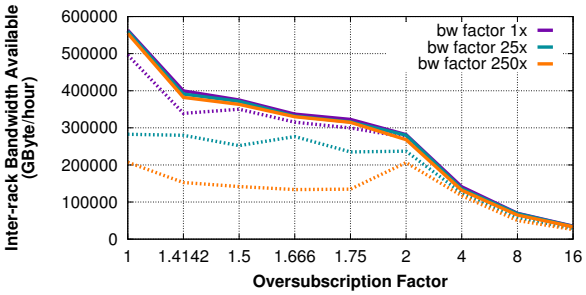

Figure 3: Inter-rack bandwidth available after placement.

Over-provisioning reduces the value of our improved placement, but it does not remove the need for better guarantees. Even on an over-provisioned network, a tenant whose guarantee is too low for its needs may suffer if its VM placement is unlucky. A "full bisection bandwidth" network is only that under optimal routing; bad routing decisions or bad placement can still waste bandwidth.

7 Conclusion

This paper described the rationale for predictive guarantees in cloud networks, and the design of Cicada, a system that provides this abstraction to tenants. Cicada provides fine-grained, temporally- and spatially-varying guarantees without requiring the clients to specify their demands explicitly. We outlined a prediction algorithm where the prediction for a future epoch is a weighted linear combination of past observations, with the weights updated and learned online in an automatic way. Using traces from HP Cloud Services, we showed that the fine-grained structure of predictive guarantees can be used by a cloud provider to improve network utilization in certain datacenter topologies.

Acknowledgments

We are indebted to a variety of people at HP Labs and HP Cloud for help with data collection and for discussions of this work. In particular, we thank Sujata Banerjee, Ken Burden, Phil Day, Eric Hagedorn, Richard Kaufmann, Jack McCann, Gavin Pratt, Henrique Rodrigues, and Renato Santos. This work was supported in part by the National Science Foundation under grant IIS-1065219 as well as an NSF Graduate Research Fellowship.

REFERENCES

[1] BALLANI, H., COSTA, P., KARAGIANNIS, T., AND ROWSTRON, A. Towards Predictable Datacenter Networks. In *SIGCOMM* (2011).

[2] BENSON, T., AKELLA, A., AND MALTZ, D. A. Network Traffic Characteristics of Data Centers in the Wild. In *IMC* (2010).

[3] BENSON, T., ANAND, A., AKELLA, A., AND ZHANG, M. Understanding Data Center Traffic Characteristics. In *WREN* (2009).

[4] DENG, S., AND BALAKRISHNAN, H. Traffic-aware Techniques to Reduce 3G/LTE Wireless Energy Consumption. In *CoNEXT* (2012).

[5] DUFFIELD, N. G., GOYAL, P., GREENBERG, A., MISHRA, P., RAMAKRISHNAN, K. K., AND VAN DER MERWE, J. E. A Flexible Model for Resource Management in Virtual Private Networks. In *SIGCOMM* (1999).

[6] ERICKSON, D., HELLER, B., YANG, S., CHU, J., ELLITHORPE, J., WHYTE, S., STUART, S., MCKEOWN, N., PARULKAR, G., AND ROSENBLUM, M. Optimizing a Virtualized Data Center. In *SIGCOMM Demos* (2011).

[7] GREENBERG, A., HAMILTON, J. R., JAIN, N., KANDULA, S., KIM, C., LAHIRI, P., MALTZ, D. A., PATEL, P., AND SENGUPTA, S. VL2: A Scalable and Flexible Data Center Network. In *SIGCOMM* (2009).

[8] GUI, C., LU, G., WANG, H. J., YANG, S., KONG, C., SUN, P., WU, W., AND ZHANG, Y. SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees. In *CoNEXT* (2010).

[9] HAJJAT, M., SUN, X., SUNG, Y.-W. E., MALTZ, D., RAO, S., SRIPANIDKULCHAI, K., AND TAWARMALANI, M. Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud. In *SIGCOMM* (2010).

[10] HERBSTER, M., AND WARMUTH, M. K. Tracking the Best Expert. *Machine Learning 32* (1998).

[11] HP Cloud. `http://hpcloud.com`.

[12] JAMIN, S., SHENKER, S. J., AND DANZIG, P. B. Comparison of Measurement-based Admission Control Algorithms for Controlled-load Service. In *Infocom* (1997).

[13] KIM, G., PARK, H., YU, J., AND LEE, W. Virtual Machines Placement for Network Isolation in Clouds. In *RACS* (2012).

[14] KNIGHTLY, E. W., AND SHROFF, N. B. Admission Control for Statistical QoS: Theory and Practice. *IEEE Network 13*, 2 (1999).

[15] LACURTS, K., MOGUL, J. C., BALAKRISHNAN, H., AND TURNER, Y. Cicada: Predictive Guarantees for Cloud Network Bandwidth. Tech. Rep. MIT-CSAIL-TR-004, Massachusetts Institute of Technology, 2014.

[16] LAM, T. V., RADHAKRISHNAN, S., PAN, R., VAHDAT, A., AND VARGHESE, G. NetShare and Stochastic NetShare: Predictable Bandwidth Allocation for Data Centers. *SIGCOMM CCR 42*, 3 (2012).

[17] LEE, J., LEE, M., POPA, L., TURNER, Y., BANERJEE, S., SHARMA, P., AND STEPHENSON, B. CloudMirror: Application-Aware Bandwidth Reservations in the Cloud. In *HotCloud* (2013).

[18] MONTELEONI, C., BALAKRISHNAN, H., FEAMSTER, N., AND JAAKKOLA, T. Managing the 802.11 Energy/Performance Tradeoff with Machine Learning. Tech. Rep. MIT-LCS-TR-971, Massachusetts Institute of Technology, 2004.

[19] NIU, D., FENG, C., AND LI, B. Pricing Cloud Bandwidth Reservations Under Demand Uncertainty. In *Sigmetrics* (2012).

[20] OpenStack Operations Guide - Managing Projects and Users. `http://docs.openstack.org/trunk/openstack-ops/content/projects_users.html`.

[21] POPA, L., KUMAR, G., CHOWDHURY, M., KRISHNAMURTHY, A., RATNASAMY, S., AND STOICA, I. FairCloud: Sharing the Network in Cloud Computing. In *SIGCOMM* (2012).

[22] RAGHAVAN, B., VISHWANATH, K., RAMABHADRAN, S., YOCUM, K., AND SNOEREN, A. C. Cloud Control with Distributed Rate Limiting. In *SIGCOMM* (2007).

[23] RASMUSSEN, A., CONLEY, M., KAPOOR, R., LAM, V. T., PORTER, G., AND VAHDAT, A. Themis: An I/O-Efficient MapReduce. In *SOCC* (2012).

[24] RASMUSSEN, A., PORTER, G., CONLEY, M., MADHYASTHA, H., MYSORE, R. N., PUCHER, A., AND VAHDAT, A. TritonSort: A Balanced Large-Scale Sorting System. In *NSDI* (2011).

[25] RODRIGUES, H., SANTOS, J. R., TURNER, Y., SOARES, P., AND GUEDES, D. Gatekeeper: Supporting Bandwidth Guarantees for Multi-tenant Datacenter Networks. In *WIOV* (2011).

[26] ROUGHAN, M., THORUP, M., AND ZHANG, Y. Traffic Engineering with Estimated Traffic Matrices. In *IMC* (2003).

[27] sFlow. `http://sflow.org`.

[28] XIE, D., DING, N., HU, Y. C., AND KOMPELLA, R. The Only Constant is Change: Incorporating Time-Varying Network Reservations in Data Centers. In *SIGCOMM* (2012).

[29] ZHANG, Y., ROUGHAN, M., DUFFIELD, N., AND GREENBERG, A. Fast Accurate Computation of Large-Scale IP Traffic Matrices from Link Loads. In *Sigmetrics* (2003).