Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2014-004 March 24, 2014

Cicada: Predictive Guarantees for Cloud Network Bandwidth

Katrina LaCurts, Jeffrey C. Mogul, Hari Balakrishnan, and Yoshio Turner

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Cicada: Predictive Guarantees for Cloud Network Bandwidth

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, 30]. Most previous work on cloud-bandwidth guarantees has assumed that cloud tenants know what bandwidth guarantees they want. However, application bandwidth demands can be complex and time-varying, and many tenants might lack sufficient information to request a bandwidth guarantee that is well-matched to their needs. A tenant's lack of accurate knowledge about its *future bandwidth demands* can lead to over-provisioning (and thus reduced cost-efficiency) or under-provisioning (and thus poor user experience in latency-sensitive user-facing applications).

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. The key idea in the prediction algorithm is to treat a set of previously observed traffic matrices as "experts" and learn online the best weighted linear combination of these experts to make its prediction. 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 report introduces **predictive guarantees**, a new abstraction for bandwidth guarantees in cloud networks. A predictive guarantee improves application-performance predictability for network-intensive applications, in terms of expected throughput, transfer completion time, or packet latency. 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 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 network inter-node communication, such as in [1], [24], and [25], current cloud systems fail to offer even basic network performance guarantees; this inhibits cloud use by enterprises that must provide service-level agreements (SLAs).

Others have proposed mechanisms to support cloud bandwidth guarantees (see §2.2); 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. 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 report, we use network traces from HP Cloud Services[1] 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-varying 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-

[1] `http://hpcloud.com`

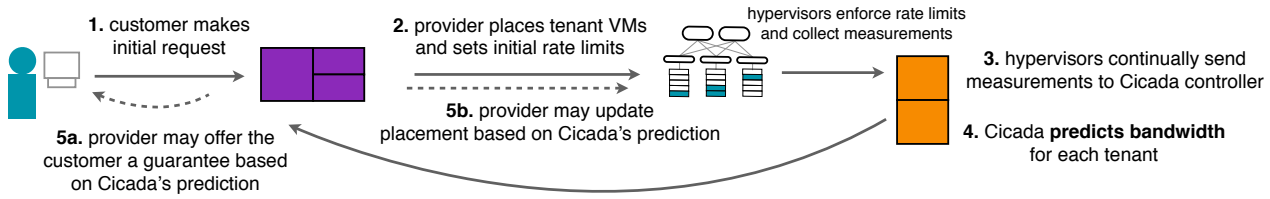


Figure 1: Cicada's architecture.

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 [30] 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**, 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. Figure 1 depicts Cicada's architecture, which we describe in detail in §4.

Our primary contributions are the introduction of predictive guarantees, and trace-based analyses of the motivation for, and utility of, our approach. We answer the following questions:

1. *Do real applications exhibit bandwidth variability?* We show that they do, using network traces from HP Cloud Services, and thus justify the need for predictions based on temporal and spatial variability in application traffic. While such variability is not surprising, it has not previously been quantified for cloud networks.

2. *How well does Cicada predict network demands?* We describe Cicada's prediction algorithm. The algorithm treats certain previously observed traffic matrices as "experts" and computes a weighted linear combination of the experts as the prediction. The weights are computed automatically using online learning. Using our network traces, we show that the median prediction errors of the algorithm are 90% lower than other methods. Moreover, we can often predict when the prediction is likely to be wrong (and avoid making guarantees in such cases). The algorithm can predict parameters for Oktopus' Virtual Oversubscribed Cluster (VOC) model [1] nearly as well as a perfect oracle.

3. *Can a provider use Cicada's predictions to better utilize its network infrastructure?* We present a greedy VM-placement heuristic, which uses these predictions. For well-provisioned networks, a provider using this heuristic can improve inter-rack bandwidth utilization by over 2× in certain datacenter topologies, compared to Oktopus.

2. Related Work

We begin by putting Cicada in context of other, related systems. We discuss related work on traffic prediction algorithms in §5.

2.1 Definitions

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.[2]: 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.

2.2 Cloud-Network Bandwidth Guarantees

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* [6]; 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. Cicada supports any of these models.

The Proteus system [30] 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

[2] `http://docs.openstack.org/trunk/openstack-compute/admin/content/users-and-projects.html`

phases, although each such phase can run at a different predicted bandwidth. Unlike Cicada, it does not generalize to a broad range of enterprise applications.

Kim et al. describe a system that measures the time-varying traffic for a tenant's VMs for a month, then uses a stable-marriage algorithm to place these VMs to reduce network oversubscription [14]. They evaluated this scheme using synthetic traffic, showing an improvement over random or first-fit VM placement. Other recent work has focused on making traffic predictions to produce short-term (10-minute) guarantees for video streaming applications [20]. Although this work considers VM-to-VM guarantees, it is not clear that the approach generalizes to long-term guarantees, or to applications beyond video streaming.

None of the papers discussed above, except for Proteus, address the problem of how the desired bandwidths are chosen. Hajjat et al. [10] 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. Though this issue would arise for a provider using Cicada, it can be addressed with numerous existing techniques. For example, SecondNet [9] supports either pipe-model or hose-model guarantees (their "type-0" and "type-1" services, respectively), and focuses on how to place VMs such that the guarantees are satisfied. Distributed Rate Limiting [23] supports a tenant-aggregate limit (similar to a hose model), and focuses on enforcing a limit at multiple sites, rather than within one cloud datacenter. GateKeeper [26] provides a pure hose-model guarantee, with the option of allowing additional best-effort bandwidth; its goal is to protect each VM's input-bandwidth guarantee against adversarial best-effort traffic. NetShare [16] focuses on how to provide enforcement mechanisms for cloud network guarantees; it may be viewed as a good way to achieve the "enforce rate limits" step of Figure 1.

Similarly, Cicada does not focus on the tradeoff between guarantees and fairness. Cloud providers could address this issue by using an existing system such as FairCloud [22], which develops mechanisms that support various points in the tradeoff space.

2.3 Datacenter Network Measurement Studies

Despite the importance of datacenter networks, the research community has had scant access to measurements, because of the difficulty of gathering large-scale measurements, the privacy and security risks created by these data sets, and the proprietary value that providers place on understanding what goes on in their networks. We know of no published measurement studies on IaaS traffic matrices (except perhaps [2], discussed below).

Prior studies on similar networks have detected temporal and spatial variability. Benson et al. [3] analyzed link-level SNMP logs from 19 datacenters, finding "temporal and spatial variations in link loads and losses." These, we believe, were not cloud (IaaS) networks *per se* (they may have been SaaS/PaaS datacenters), although their applications may be similar to those of cloud tenants. Benson et al. [2] gathered SNMP statistics for ten datacenters and packet traces from a few switches in four datacenters. They describe several of these as "cloud data centers," but it is unclear whether they are actually IaaS networks. They report that "diurnal patterns [in link utilization] exist in all [ten] data centers," and that "time-of-day and day-of-week variation exists in many of the data centers," especially in the cores of these networks.

Greenberg et al. [8] report on SNMP and NetFlow data from a "large cloud service provider." We believe this, too, is not an IaaS provider. They report a distinct lack of short-term predictability for traffic matrices, but do not say whether this datacenter experiences diurnal variations. Kandula et al. [13] found considerable short-term spatial variation in a 1500-server data-mining cluster, but did not investigate whether this variation is predictable. Bodík et al. [4] also found spatial variation in inter-service traffic in a datacenter, as a result of the fact that machines responsible for different services did not typically communicate with one another.

2.4 Internet QoS

QoS on the Internet was a vibrant research area for many years, with architectures such as IntServ developed for end-to-end guarantees. End-to-end Internet QoS has seen little practical deployment, in part because most Internet paths involve multiple providers, making the economics and payment structure of any end-to-end guaranteed QoS scheme difficult. In contrast, a cloud network is run by a single operator, and inherently has a mechanism to bill its customers.

Another drawback of proposals such as IntServ is that they force the application, or end point, to make explicit reservations and to specify traffic characteristics. For all but the simplest of applications, this task is challenging, and many application developers or operators have little idea what their traffic looks like over time. Cicada resolves this issue through its use of predictions.

Cicada's predictive approach resembles ISP traffic engineering, which ISPs deploy in practice. Traffic engineering uses estimates of traffic to map flows or aggregates to paths in a network. The primary difference is that Cicada makes predictions about the future, and uses these to offer predictive guarantees to tenants, and/or to place VMs so as to shift traffic. Cicada does *not* deal with the problem of estimating traffic matrices from noisy link measurements, which many traffic engineering schemes address.

3. 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 (e.g., VOC-style allocations [1], static, all-to-all traffic matrices, etc.). Before building Cicada, we collected data from HP Cloud Services, to analyze the spatial and temporal variability of its tenants' traffic.

3.1 Data

We have collected sFlow [28] 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 (portions of a datacenter that are isolated from failures in other parts of the same datacenter). This dataset differs from those in previous work—such as [2, 3, 8]—in that it captures VM-to-VM traffic patterns. It does not include any information about what types of applications were running (e.g., MapReduce jobs, webservers, etc.), as that information is not available from packet headers.

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.

3.1.1 Dataset Limitations

During data collection, the physical network was generally over-provisioned. Hence, our measurements reflect the actual offered loads at the virtual interfaces of the tenant VMs. Some of the smaller VMs were output-rate-limited at their virtual interfaces by HPCS; we do not know these rate limits.

Because the HPCS dataset samples come from the ToR switches, they do not include any traffic between pairs of VMs when both are on the same server. We believe that we still get samples from most VM pairs, because, in the measured configuration (OpenStack Diablo), VMs are placed on servers uniformly at random. Assuming such placement on n servers, the probability of any two of a tenant's k VMs being on the same server is $1/n$; with $n = 1360$, the probability that we will miss any given VM-pair's traffic is less than .001%.[3]

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.

For privacy reasons, our dataset does not include information associating VMs with tenants or applications. Instead, we define tenants by using the connected components of the full traffic matrix, which is in line with the OpenStack definition of tenants.

[3]The expected number of unordered pairs of k VMs sharing n servers is $\frac{1}{n}\binom{k}{2}$. There are $\binom{k}{2}$ possible unordered pairs among k VMs, so the probability that any VM-pair shares the same server is $\frac{1}{n}\binom{k}{2}/\binom{k}{2}$.

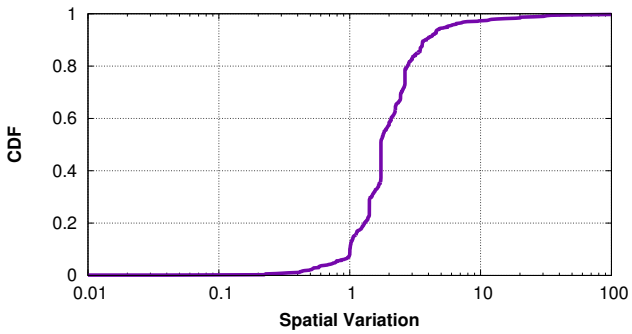


Figure 2: Spatial variation in the HPCS dataset.

3.1.2 Measurement-Interval Granularity

Cicada's predictive guarantees are parameterized on two values: H, the amount of time the guarantee is good for, and δ, the peak-bandwidth measurement interval. During a period of H hours, Cicada's prediction reflects what the peak-bandwidth should be in any δ-second interval (e.g., if $H = 1$ and $\delta = 300$, Cicada's prediction reflects the maximum amount of bandwidth that it expects the application to use in any 300-second interval for the next hour). For latency-sensitive applications, predictions for $\delta << H$ are useful; for latency-insensitive applications, $\delta \approx H$.

Because our data collection samples samples over 5-minute intervals, for our datasets, $\delta \geq 300$ seconds. Real applications might require guarantees with $\delta < 1$ second; our implementation supports these small δ values (§8.1).

3.1.3 Over-sampling

Some of the paths in the HPCS dataset are over-sampled. If A and B are VMs on the same ToR switch S_1, traffic on $A \rightsquigarrow B$ will only be sampled at S_1. But if VM C resides on a different ToR S_2, traffic on $A \rightsquigarrow C$ will be sampled at both S_1 and S_2, twice as often as $A \rightsquigarrow B$.

We correct this problem by noting which switches we see on each path (sFlow samples are tagged with the ToR's own address). If we see two switches on a path, we know that this flow has been oversampled, so we scale those measurements down by a factor of two.

3.2 Spatial Variability

To quantify spatial variability, we compare the observed tenants to an ideal, static, all-to-all tenant (this tenant is ideal as it is the easiest to make predictions for: every intra-tenant connection uses the same amount of bandwidth all of the time). Let F_{ij} be the fraction of this tenant's traffic sent from VM_i to VM_j. For the "ideal" all-to-all tenant, the distribution of these F values has a standard deviation of zero, since every VM sends the same amount of data to every other VM.

For each tenant in the HPCS dataset, we calculate the distribution of its F values, and plot the coefficient of variation (standard deviation over mean) in Figure 2. The median *cov* value is 1.732, which suggests nontrivial overall spatial variation.

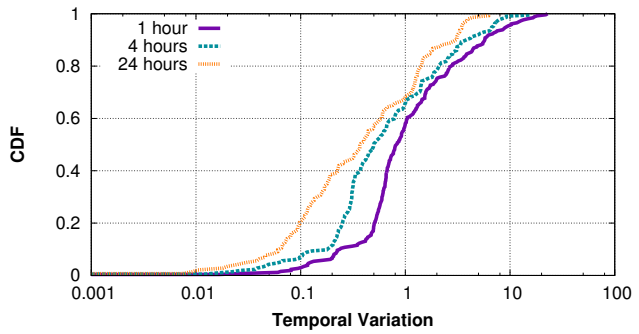


Figure 3: Temporal variation in the HPCS dataset.

Some VMs communicate much more than others: one tenant with $cov > 10$ has eight VMs, with a few pairs that send modest amounts of data, and all other pairs sending little to no data. These results suggest that a uniform, all-to-all model is insufficient for making traffic predictions; given the high *cov* values in Figure 2, a less strict but not entirely general model such as VOC [1] may not be sufficient either.

3.3 Temporal Variability

To quantify temporal variability, we first pick a time interval H. For each consecutive non-overlapping interval of H hours, we calculate the sum of the total number of bytes sent between each pair p of VMs. This 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, where the weight for cov_p is the bandwidth used by pair p. This scaling reflects 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.

For each tenant in the HPCS dataset, we calculate its temporal variation value, and plot the CDF in Figure 3. Like the spatial variation graph, Figure 3 shows that most tenants have high temporal variability. This variability decreases as we increase the time interval H, but we see variability at all time scales. Tenants with high temporal variation are typically ones that transfer little to no data for long stretches of time, interspersed with short bursts of activity.

The results so far indicate that typical cloud tenants may not fit a rigid model. In particular, Figure 3 shows that most tenants exhibit significant temporal variation. Thus, static models cannot accurately represent the traffic patterns of the tenants in the HPCS dataset.

4. The Design of Cicada

The goal of Cicada is to free tenants from choosing between under-provisioning for peak periods, or over-paying 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.

4.1 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). The customer might also propose its own guarantee, for which the provider can offer a price. 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 can exploit best-effort bandwidth beyond their guarantees.

A provider may wish to place and perhaps migrate VMs based on their associated bandwidth guarantees, to improve network utilization. We describe a VM migration method in §7.4. The provider could also migrate VMs to increase the number of guarantees that the network can support [7].

4.2 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, such as [1], [12], or [15]. We also assume that the cloud provider has a method to enforce guarantees, such as [22].

4.3 Measurement Collection

Cicada collects a time series of traffic matrices for each tenant. One could do this 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. Switch-based measurements create several challenges, including correctly ascribing VMs to the correct tenant. Our design collects VM-pair traffic measurements, using an agent that runs on each compute server (see §8.1), and periodically reports these to the controller.

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 underpredicting future traffic. We observe that we can detect when

a VM's traffic is rate-limited (see §8.1), so these underestimates can be detected, too, although not precisely quantified. Currently, Cicada does not account for this potential error.

4.4 Prediction Model

Some applications, such as backup or database ingestion, require bulk bandwidth—that is, they need guarantees that the average bandwidth over a period of H hours will meet their needs. Other applications, such as user-facing systems, require guarantees for peak bandwidth over much shorter intervals. Thus, 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. Our current implementation produces a prediction once per hour ($H = 1$).

Note, of course, 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 uses the results of the prediction to determine whether it can issue reliable predictive guarantees for a tenant; if not, it does not propose such a guarantee.

4.5 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 (and perhaps bear some fraction of the cost of the upgrade), 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.

We also note that a Cicada-based provider must maintain the trust of its customers: it cannot regularly under-predict bandwidth demands, or else tenants will have insufficient guarantees and their own revenues may suffer; it also cannot regularly over-predict demands, or else customers will be over-charged and take their business elsewhere. This motivates our focus in §7 on evaluating the quality of Cicada's predictions, including its decision whether a tenant's demands are in fact predictable.

5. CICADA'S TRAFFIC PREDICTION METHOD

Much work has been done on predicting traffic matrices from noisy measurements such as link-load data [27, 31]. 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 (an epoch is H-hours long; typically one hour); its problem is to predict a *future* traffic matrix.

To the best of our knowledge, there is little work in this area, especially in the context of cloud computing. On ISP networks, COPE [29] describes a strategy that predicts the best *routing* over a space of traffic predictions, made up of the convex hull of previous traffic matrices. It is not a prediction scheme per se, but we draw inspiration from this work, and base our prediction algorithm around computing a "best" convex combination of previous traffic matrices as our future estimate.

5.1 Algorithm

The algorithm uses Herbster and Warmuth's "tracking the best expert" idea [11], which has been successfully adapted before in wireless power-saving and energy reduction contexts [5, 18]. To predict the traffic matrix for epoch $n+1$, we use all previously observed traffic matrices, $M_1, \ldots, M_n$ (later, we show that matrices from the distant past can be pruned away without affecting accuracy). In our context, the rows and columns of the matrix are each VMs, with the entry in row i and column j specifying 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).

Each of these previously observed matrices acts as an "expert," recommending that M_i is the best predictor, $\hat{M}_{n+1}$, for epoch $n+1$. The algorithm computes $\hat{M}_{n+1}$ as a weighted linear combination of these matrices:

$$\hat{M}_{n+1} \quad = \quad \sum_{i=1}^{n} w_i(n) \cdot M_i$$

where $w_i(n)$ denotes the weight given to M_i when making a prediction for epoch $n+1$, with $\sum_{i=1}^{n} w_i(n) = 1$.

The algorithm learns the weights online. At each step, it updates the weights according to the following rule:

$$w_i(n+1) = \frac{1}{Z_{n+1}} \cdot w_i(n) \cdot e^{-L(i,n)}$$

where $L(i,n)$ denotes the *loss of expert i in epoch n* and Z_{n+1} normalizes the distribution so that the weights sum to unity.

We use the relative Euclidean ℓ^2-norm between M_i and M_n as the loss function L. Treating both these matrices as vectors, $\vec{M}_i$ and $\vec{M}_n$ respectively, the relative ℓ^2-norm error is

$$L(i,n) = E_{\ell^2} = \|\vec{M}_i - \vec{M}_n\| / \|\vec{M}_n\|.$$

That is, the norm of the individual errors over the Euclidean norm of the observed data (the square-root of the sum of the squares of the components of the vector).

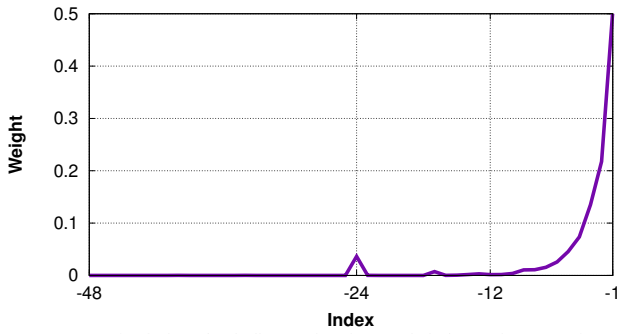


Figure 4: Average weights produced by Cicada's prediction algorithm. These values represent the average of the final weight values over each application in the HPCS dataset.

Note that this algorithm can predict both average bandwidth as well as peak bandwidth. The only difference is the input matrices: for average bandwidth, the matrices $M_1, \ldots, M_n$ represent the total amount of data in each epoch, while for peak bandwidth, they represent the maximum over a δ-length interval. It is also easy to extend this model to produce hose-model predictions rather than pipe-model predictions, simply by using input matrices that represent the hose bandwidth (in this case, each matrix is a $1 \times n$ matrix, and entry i corresponds to the number of bytes sent out of VM i).

5.1.1 Intuition

Our intuition when approaching the problem of predicting traffic is that the traffic matrix M_n should depend most heavily on the most recent traffic matrices (M_{n-1}, M_{n-2}, etc.), as well as on traffic matrices from similar time periods on previous days (M_{n-24}, M_{n-48}, etc.).

If our intuition were correct, Cicada's prediction algorithm will naturally result in higher weights for these matrices. After making the predictions for each application in our dataset, we took the weights for each application, and calculated the average weight values over our entire dataset. These average weights are plotted in Figure 4 (the x axis is limited to the two most recent days of data). 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 24 hours earlier. In particular, we looked for a spike at the 7-day offset, expecting that some user-facing applications have weekly variations, but found none. This result indicates that, at least in our dataset, one does not need weeks' worth of data to make reliable predictions; a much smaller amount of data suffices.

5.2 Alternate Prediction Algorithms

We tested Cicada's prediction algorithm against two other algorithms. First, we tried a machine learning algorithm based on linear regression. A prediction between VMs i and j was made by finding "relevant" historical data between i and j, finding the linear function f that best mapped a previous epoch's data to the next epoch's, and using f to make a prediction for the current time. The "relevant" historical data was data between i and j from similar times-of-day and days-of-week; similarity was determined via a Mann-Whitney U-Test, as in [17]. This algorithm was also based on our intuition that traffic from the same time-of-day would be more relevant than traffic from other times. In practice, we found that this algorithm performed comparably to the bank-of-experts-based algorithm on average bandwidth predictions, but was very unreliable for peak bandwidth predictions; for that reason, we do not present detailed results in §7.

We also tested Cicada's algorithm against a simple EWMA. This technique performed worse than Cicada's algorithm for both peak and average bandwidth prediction, and so we do not present detailed results in §7.

6. Comparing Cicada to VOC

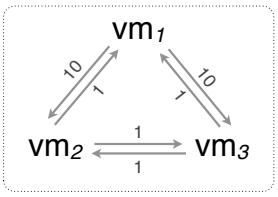
Because we believe that the VOC model [1] represents the state of the art in spatially-varying cloud-bandwidth reservations, in our trace-based evaluation of Cicada, we use a VOC-style system as the baseline. In [1], the authors use VOCs to make bandwidth reservations for tenants, allowing different levels of bandwidth between different groups of VMs. Although their system was *not* developed for making bandwidth predictions, we can interpret its bandwidth reservations as predictions. We compare a VOC-style system to Cicada, finding that Cicada can accurately predict the parameters for a VOC-style system, and that Cicada's pipe model results in less waste than VOC's hierarchical hose model.

6.1 VOC-style Guarantees

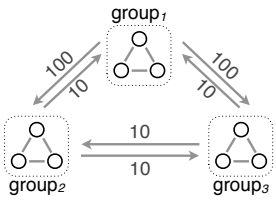
The Oktopus system introduced the Virtual Oversubscribed Cluster (VOC) model, to "capitalize on application structure to reduce the bandwidth needed from the underlying physical infrastructure" [1]. In VOC, the tenant's guarantee request is of the form $\langle N, B, S, O \rangle$, such that a virtual network of N VMs is divided into groups of size S. The VMs within each group are effectively connected to each other via links of bandwidth B through a non-oversubscribed virtual switch. Between any pair of groups, VOC provides bandwidth B/O; thus, O is the effective oversubscription factor for inter-group traffic. VOC is designed to reflect not only a typical application structure, but also a typical physical datacenter topology, where Top-of-Rack (ToR) switches have high capacity, but the aggregation layer that connects the ToRs is oversubscribed.

Oktopus itself is designed to place the VMs on the network such that their VOC requests are met. It assumes that the tenants are responsible for choosing the parameters $\langle B, S, O \rangle$. We designed a heuristic, detailed in the Appendix, to determine these parameters, and to allow variable cluster sizes.

To compare Cicada and VOC, we interpret the B and B/O hoses as predictions: VOC predicts that a VM in a particular group will use B bandwidth to send to VMs within its cluster, and groups will use B/O bandwidth to send to other groups. These predictions are not pipe-model predictions, as is our preferred model for Cicada, but we are still able to compare



(a) This traffic pattern will result in over-allocation within one VOC group.



(b) Similarly, this traffic pattern results in over-allocation across VOC groups.

Figure 5: Cases where VOC is inefficient.

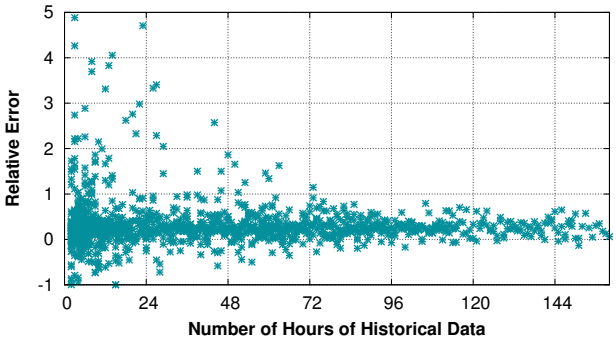


Figure 6: Relative error vs. history

the accuracy of the two systems on a tenant-by-tenant basis.

6.2 Inefficiencies of VOC

The Oktopus paper showed that the VOC model gives providers more flexibility than a clique abstraction, which provides a static, uniform bandwidth between all pairs of VMs. However, we give an example to illustrate how even the VOC model can limit provider flexibility for certain applications, due to over-allocating bandwidth both within groups and across groups.

Consider a group of three VMs as in Figure 5(a). Suppose that VM_1 sends 20 units total to VM_2 and VM_3. Because of this, B must be at least 20 for each VM in the group. However, if VM_2 and VM_3 send fewer than 20 units total, this value of B will over-allocate bandwidth. In practical terms, if each VM is on a distinct server, the VOC model requires allocating 20 units of each server's NIC output bandwidth to this tenant, even though VM_2 and VM_3 only need 2 NIC-output units. A similar pattern can exist across groups, as in Figure 5(b), where one group requires more total bandwidth than the others.

VOC's over-allocation of bandwidth in these scenarios stems from an assumption that VMs within a group behave similarly (sending the same amount of data to the rest of the group), as well as a corresponding assumption across groups.

7. EVALUATION

We evaluated our data on the HPCS dataset. We tested several hypotheses: first, that Cicada can determine when one of its predictions is reliable or not; second, that Cicada can accurately predict a tenant's future bandwidth requirements; and third, that predictions incorporating spatial and temporal variation can yield guarantees that waste less bandwidth than uniform guarantees or static. We define "wasted bandwidth" as the bandwidth that is guaranteed to a tenant but not used. Wasted bandwidth is a proxy for estimating how much money a customer would save—methods that waste less bandwidth will likely save the customers money—but allows us to avoid defining a particular cost model. Overall, we found that the reliability of Cicada's predictions was correlated with the size of a tenant and the frequency of under-predictions, that Cicada's guarantees were accurate for both average and peak predictions, and that Cicada's guarantees were more accurate than guarantees based on VOC, decreasing the relative per-tenant error by 90% in both the average-bandwidth case and the peak-bandwidth case.

We observed very little evidence of VM flexing in our dataset (where the number of a tenant's VMs changes over time). Flexing would typically manifest as over-prediction errors (making predictions for VM pairs that used to exist, but no longer do because of flexing). To eliminate this mistake, we eliminated any predictions where the ground truth data was zero, but found that it did not appreciably change the results. For this reason, we do not present separate results in this section, and simply report the results over all of the data.

7.1 Quantifying Prediction Accuracy

To quantify the accuracy of a prediction, we compare the predicted values to the ground truth values, using the relative ℓ^2-norm error described in §5.1. For Cicada, the prediction and ground-truth vectors are of length $N^2 - N$, because Cicada makes predictions between each pair of distinct VMs. In a VOC-style system, there are fewer predictions: one prediction for each of the N VMs to the other VMs in its group, and one prediction for each group to the other groups. Regardless of the number of total predictions, we get one error value per tenant, per time interval.

In addition to the relative ℓ^2-norm error, we also present the per-tenant relative error. This metric is simply the sum of the prediction errors divided by the total amount of ground-truth data. Unlike the relative ℓ^2-norm error, this metric discriminates between over-prediction and under-prediction, since the latter can be more disruptive to application performance. Because it does not use vector norms, the per-tenant relative error makes it easier for us to see if either system has substantially under-predicted for a tenant. However, it is possible that under- and over-prediction errors for different VM pairs could cancel out in the relative error; this type of cancellation does not happen in the relative ℓ^2-norm error.

7.2 Determining Whether Predictions Are Reliable

In our dataset, we found that Cicada could not reliably make a correct prediction for tenants with few VMs. In all of the results that follow, we eliminate tenants with fewer than five VMs. This elimination is in line with the fact that

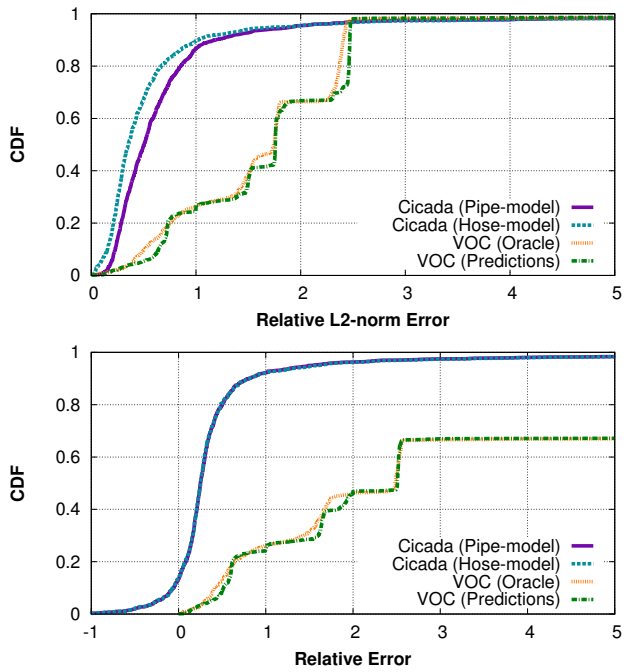


Figure 7: Prediction errors for average bandwidth.[4]

Cicada is meant for tenants with network-heavy workloads. It is possible that on other datasets, the precise number of tenants below which Cicada cannot reliably make predictions will differ. Additionally, for tenants that started out with a series of under-predictions, Cicada's prediction algorithm rarely recovered. For this reason, we also consider tenants with more than fifteen under-predictions to be unreliable, and do not continue making predictions for them (we do, however, include results from all predictions for that tenant up to that point).

Interestingly, we found that Cicada could often make accurate predictions even with very little historical data. Figure 6 shows the relative per-tenant error as a function of the amount of historical data. Though there is a clear correlation between relative error and the amount of historical data, it is not necessary to have many hours of data in order to make accurate predictions.

7.3 Prediction Errors for Individual Tenants

To compare the accuracy of Cicada and a VOC-style model, we make predictions for one-hour intervals. We allow both types of guarantees to change over time; that is, the VOC configuration for a particular hour need not be the same as the configuration in the next hour. This is an extension from the original Oktopus paper [1], and improves VOC's performance in our comparison.

We evaluate VOC-style guarantees using both predicted parameters and "oracle-generated" parameters (i.e., with perfect hindsight). For the oracle parameters, we determine the VOC clusters for a prediction interval using the ground-truth data from that interval. This method allows us to select the absolute best values for B and O for that interval, as well as the best configuration. Thus, any "error" in the VOC-oracle results comes from the constraints of the model itself, rather than from an error in the predictions.

7.3.1 *Average Bandwidth Guarantees*

Figure 7 shows the results for average bandwidth prediction. Both Cicada models have lower error than either VOC model; Cicada's pipe model decreases the error by 90% compared to the VOC oracle model (comparison against the predictive VOC model, as well as between VOC and Cicada's hose model, yields a similar improvement). The ℓ_2-norm error decreases by 71%. The VOC model using predictions also closely tracks the VOC-oracle model, indicating that Cicada's prediction algorithm can generate accurate predictions for a system using VOC.

In terms of per-tenant relative error, Cicada occasionally under-predicts, whereas neither VOC model does. Under-prediction could be worse than over-prediction, as it means that an application's performance could be reduced. The effect of under-provisioning can be lessened by scaling predictions by an additive or multiplicative factor, though this risks over-prediction. In the results presented here, we have scaled the predictions by $1.25\times$. In addition to lessening the effects of under-provisioning, this scaling also allows the bank-of-experts algorithm to make a prediction that is *greater* than any of the previous matrices. We were unable to determine a systematic way to remove the remaining under-predictions, but speculate that altering the loss function to penalize under-prediction more heavily than over-predictions may help; we leave this to future work.

7.3.2 *Peak Bandwidth Guarantees*

Figure 8 compares peak predictions for $\delta = 5min$. As with the average-bandwidth predictions, Cicada's prediction errors are generally lower, decreasing the median error again by 90% from the VOC oracle model (the median ℓ_2-norm error decreases by 80%). As before, Cicada does under-predict more frequently than either VOC model, but overall, the results for peak-bandwidth guarantees show that Cicada performs well even for non-average-case traffic.

7.3.3 *Discussion*

Though Cicada's prediction algorithm performs well on our dataset, we make no claim that it is the *best* prediction algorithm. Rather, we have shown that accurate traffic prediction is possible and worthwhile in the context of cloud computing.

7.4 Placement for Reducing Intra-rack Bandwidth

In the previous sections, we reported wasted bandwidth as a total; however, it is not clear that cloud providers treat wasted intra-rack and inter-rack bandwidth equally. Inter-rack bandwidth may cost more, and even if intra-rack bandwidth

[4]For average bandwidths, straightforward math shows that Cicada's hose-model and pipe-model relative errors will always be identical. The peak-bandwidth errors, shown in Figure 8, can diverge.

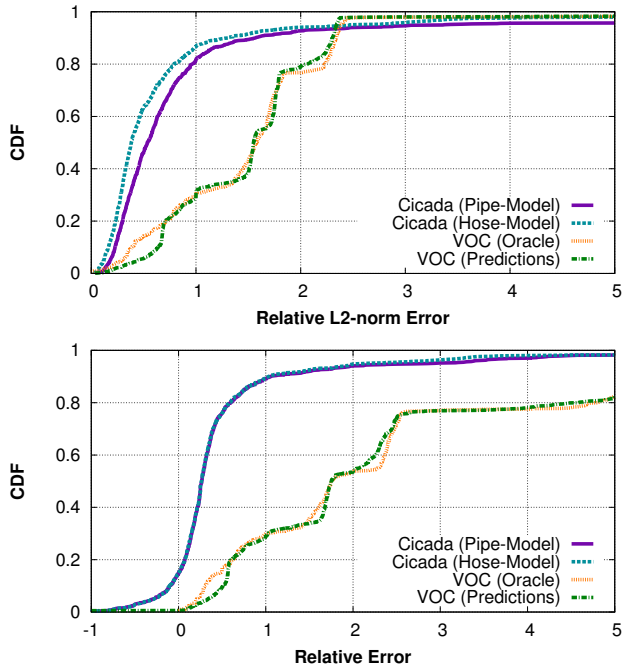


Figure 8: Prediction errors for peak bandwidth.

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
```

is free, over-allocating network resources on one rack can prevent other tenants from being placed on the same rack (due to a presumed lack of network resources).

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,

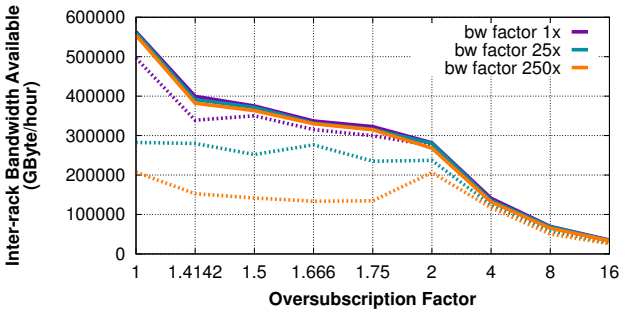


Figure 9: Inter-rack bandwidth available after placing applications.

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 9 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.

Over-provisioning reduces the value of our improved placement, but it does not necessarily 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.

8. Implementation

We have designed an implementation of Cicada as part of the OpenStack [21] framework. The implementation has two components: rate-limiters, which run in the hypervisor of every physical machine, and a controller that runs somewhere in the datacenter. We envision that Cicada extends the OpenStack API to allow the tenant's application to automatically negotiate its network guarantees, rather than requiring human interaction. A typical setting might be "Accept all

of Cicada's guarantees as long as my bill does not exceed D dollars," or "Accept all of Cicada's guarantees, but add a buffer of 10Mb/s to each VM-pair's allocated bandwidth" (if the customer anticipates some amount of unpredictable traffic).

8.1 Server Agents

On each server, Cicada uses an agent process to collect traffic measurements and to manage the enforcement of rate limits. The agent is a Python script, which manages the Linux `tc` module in the hypervisor. Cicada's agent uses `tc` both to count packets and bytes for each VM-pair with a sender on the server, and, via the the HTB queuing discipline, to limit the traffic sent on each pair in accordance with the current guarantee-based allocation. This type of collection is more fine-grained than sFlow, and `tc` also allows us to detect, but not precisely quantify, traffic demands in excess of the rate limit, by counting packet drops. The agent receives messages from the controller that provide a list of rate limits, for the $\langle src, dst \rangle$ pairs where src resides on that server. It also aggregates counter values for those pairs, and periodically reports them to the controller. To avoid overloading the controller, the reporting period P is larger than the peak-measurement period δ.

8.2 Centralized Controller

The centralized controller is divided into two components. The prediction engine receives and stores the data about the tenants from each server agent. Once a prediction is made and approved by the tenant, the rate controller uses the OpenStack API to determine which VMs reside on which physical server, and communicates the relevant rates to each rate limiter. At this point, Cicada could also use its placement algorithm (Algorithm 1) to determine whether any VMs should be migrated, and migrate them via the OpenStack API. Since Cicada makes long-term traffic predictions, this migration would be done at long timescales, mitigating the overhead of migrating VMs.

8.3 Scalability

While our current controller design is centralized, it does not need to be. Since Cicada makes predictions about applications individually, the prediction engine can be spread across multiple controllers, as long as all of the data for a particular application is accessible by the controller assigned to that application.

Cicada's use of pipe-model rate limiting—that is, one rate-limiter for each VM-pair, at the source hypervisor—could potentially create scaling problems. Some prior work has also used pipe-model rate limiting; for example, while Oktopus implements hose-model guarantees, it enforces these using "per-destination-VM rate limiters" [1]. However, we have found no published study of software rate-limiter scaling.

We tested the scalability of the `tc` rate-limiters used in our implementation on a pair of 12-core Xeon X5650 (2.67GHz) servers running Linux 3.2.0-23, with 10Gbps NICs. We used `netperf` to measure sender-side CPU utilization for

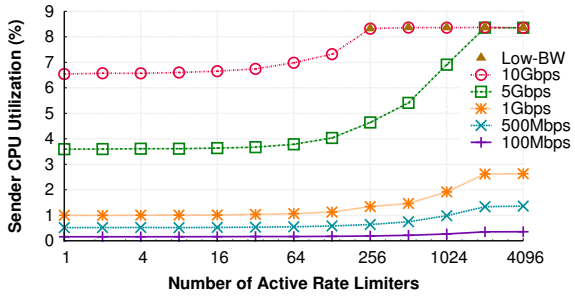


Figure 10: Effect of rate limiters on CPU utilization.

60-second TCP transfers under various rate limits, while varying the number of sender-side rate-limiters. (Only one TCP stream was active at any time.)

Fig. 10 shows mean results for at least 11 trials. Each curve corresponds to a rate limit; we marked cases where the achieved bandwidth was under 90% of the target limit. On this hardware, we could run many low-bandwidth limiters without much effect on CPU performance. When we used more than about 128 high-bandwidth limiters, both CPU utilization and TCP throughput suffered, but it is unlikely that a real system would allow both a 10Gbps flow and lots of smaller flows on the same server. We conclude that pipe-model rate limiting does indeed scale; [19] also supports this claim.

9. Conclusion

This report described the rationale for predictive guarantees in cloud networks, and the design of Cicada, a system that provides this abstraction to tenants. Cicada is able to provide fine-grained, temporally- and spatially-varying guarantees without requiring the clients to specify their demands explicitly. We presented a prediction algorithm using the "best expert" model, 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 Cicada accurately predicts tenant bandwidth needs. We also 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. We also thank John Dickerson for his help in refining the prediction algorithm in §5.1. This work was supported in part by the National Science Foundation under grant IIS-1065219 as well as an NSF Graduate Research Fellowship.

APPENDIX: VOC PARAMETER-SELECTION

The authors of [1] note that VOC can easily be extended to handle groups of multiple sizes (Section 3.2 of [1]), but give no method for automatically determining what the groups should be, nor for choosing the best values of B and O.

To address this issue, we developed a heuristic for determining the VOC groups as well as B and O. The heuristic allows groups of multiple sizes, and relies on two assumptions. First, that connections through the aggregate layer are physically oversubscribed by some factor $O_p > 1$. Second, that any guarantee should be sufficient to meet the tenant's offered network load. The alternative would be to carefully under-provision certain virtual network paths, which would increase a job's completion time but might also reduce the total cost for running some jobs. Such cost-reduction would require in-depth knowledge about the tenant's application and utility functions, and is beyond the scope of this work.

Our heuristic works by starting with an initial configuration of groups, with each VM is in its own group. The heuristic merges the groups that have the highest bandwidth between them for a new configuration, and iterates on this configuration in the same manner, until the new configuration wastes more bandwidth than the previous configuration.

Finding B and O for a particular configuration is easy, since Cicada collects measurement data. Our heuristic selects the B and O that minimize wasted bandwidth while never under-allocating a path. B, then, is the maximum of any hose within one group, and O is the maximum of any hose across groups, given the previous historical data.

Algorithm 2 VOC parameter selection.

```
groups_p = list of groups, initialized to one group per VM
G_v = groups_p // last valid configuration
while True do
    G' = mergeLargestClusters(groups_p)
    if G' = groups_p then
        break
    B, O, w = getParamsAndWastedBandwidth(G)
    if O < 1 // invalid configuration then
        groups_p = G'
        continue
    B_v, O_v, w_v = getParamsAndWastedBandwidth(G_v)
    if w > w_v then
        break
    else
        groups_p = G_v = G'
B, O = findParameters(G_v)
```

Output: B, O, G_v

10. 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] BODÍK, P., MENACHE, I., CHOWDHURY, M., MANI, P., MALTZ, D. A., AND STOICA, I. Surviving Failures in Bandwidth-Constrained Datacenters. In *SIGCOMM* (2012).

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

[6] 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).

[7] 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).

[8] 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).

[9] 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).

[10] 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).

[11] HERBSTER, M., AND WARMUTH, M. K. Tracking the Best Expert. *Machine Learning 32* (1998), 151–178.

[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] KANDULA, S., SENGUPTA, S., GREENBERG, A., PATEL, P., AND CHAIKEN, R. The Nature of Datacenter Traffic: Measurements & Analysis. In *IMC* (2009).

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

[15] KNIGHTLY, E. W., AND SHROFF, N. B. Admission Control for Statistical QoS: Theory and Practice. *IEEE Network 13*, 2 (1999), 20–29.

[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), 5–11.

[17] MALALUR, P. Traffic Delay Prediction from Historical Data. Master's thesis, Massachusetts Institute of Technology, 2010.

[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] MYSORE, R. N., PORTER, G., AND VAHDAT, A. FasTrack: Enabling Express Lanes in Multi-Tenant Data Center. In *CoNEXT* (2013).

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

[21] OpenStack. `http://openstack.org`.

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

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

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

[25] 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).

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

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

[28] sFlow. `http://sflow.org/about/index.php`.

[29] WANG, H., XIE, H., QIU, L., YANG, Y. R., ZHANG, Y., AND GREENBERG, A. COPE: Traffic Engineering in Dynamic Networks. In *SIGCOMM* (2006).

[30] 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).

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