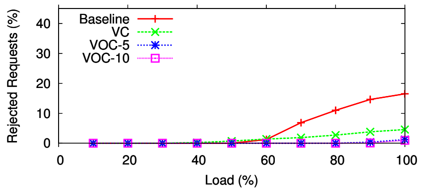
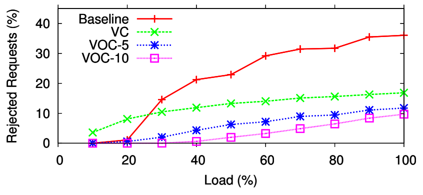
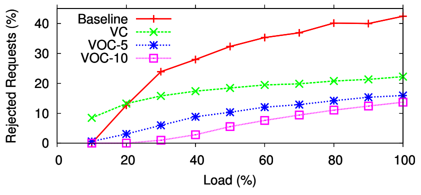


(a) Mean BW 100 Mbps



(b) Mean BW 500 Mbps



(c) Mean BW 900 Mbps

Figure 13: Percentage of rejected tenant requests with varying datacenter load and varying mean tenant bandwidth requirements. At load>20%, virtual networks allow more requests to be accepted.

scheme in which a request is rejected if it cannot be immediately allocated. For today's setup, requests are rejected if there are not enough available VMs when a request is submitted. For our virtual networks, instead, even if enough VMs are available, a request can still be rejected if the bandwidth constraints cannot be satisfied. We simulate the arrival and execution of 10,000 tenant requests with varying mean bandwidth requirements for the tenant jobs.

Rejected requests. Figure 13 shows that *only at very low loads, Baseline setup is comparable to virtual abstractions in terms of rejected requests, despite the fact that virtual abstractions explicitly reserve the bandwidth requested by tenants.* At low loads, requests arrive far apart in time and thus, they can always be allocated even though the Baseline setup prolongs job completion. As the load increases, Baseline rejects far more requests. For instance, at 70% load (Amazon EC2's operational load [3]) and bandwidth of 500 Mbps, 31% of Baseline requests are rejected as compared to 15% of VC requests and only 5% of VOC-10 requests.

Tenant costs and provider revenue. Today's cloud providers charge tenants based on the time they occupy their VMs. Assuming a price of k dollars per-VM per unit time, a tenant using N VMs for time T pays kNT dollars. This implies that *while intra-cloud network communication is not explicitly charged for, it is not free* since poor network performance can prolong tenant jobs and hence, increase their costs. Figure 12 shows the increase in tenant job completion times and the corresponding increase in tenant costs (upper X-axis) today. For all load values, many jobs finish later and cost more than expected– the cost for 25% tenants is more than 2.3 times their ideal cost had the network performance been sufficient (more than 9.2 times for 5% of the tenants).

The fraction of requests that are accepted and the costs for accepted requests govern the provider revenue. Figure 14 shows the provider revenue when tenants use virtual networks relative to Baseline revenue. At low load, the provider revenue is reduced since the use of virtual networks ensures that tenant jobs finish faster and they pay significantly less. However, as the load increases, the provider revenue increases since virtual network allow more requests to be accepted, even though individual tenants pay less than today. For efficiency, providers like Amazon operate their datacenters at an occupancy of 70-80% [3]. Hence, *for practical load values, virtual networks not only allow tenants to lower their costs, but also increase provider revenue!* Further, this estimation ignores the extra tenants that may be attracted by the guaranteed performance and reduced costs.

Charging for bandwidth. Providing tenants with virtual networks opens the door for explicitly charging for network bandwidth. This represents a more fair charging model

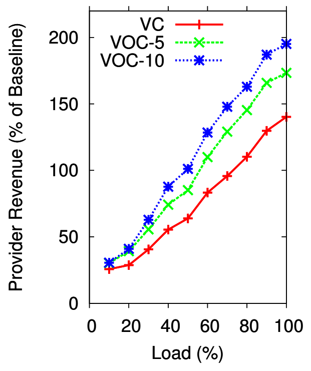


Figure 14: Provider revenue with virtual network abstractions. Mean BW = 500Mbps.

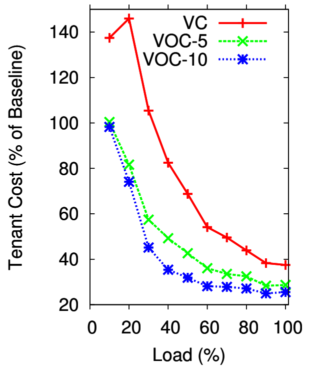


Figure 15: Relative tenant costs based on bandwidth charging model while maintaining provider revenue neutrality.

since a tenant should pay more for a *virtual cluster* with 500Mbps than one with 100Mbps. Given the lack of a reference charging model, we use a simple model to explore the economic benefits that the use of virtual networks would provide. Apart from paying for VM occupancy (k_v), tenants also pay a bandwidth charge of $k_b \frac{\$}{\text{bw*unit-time}}$. Hence, a tenant using a virtual cluster $<N, B>$ for time T pays $NT(k_v + k_b B)$.

Such a charging model presents an opportunity to redress the variability in provider revenue observed above. To this effect, we performed the following analysis. We used current Amazon EC2 prices to determine k_v and k_b for each virtual network abstraction so as to maintain provider revenue neutrality, i.e., the provider earns the same revenue as today.[4] We then determine the ratio of a tenant's cost with the new charging model to the status quo cost. The median tenant cost is shown in Figure 15. We find that except at low loads, *virtual networks can ensure that providers stay revenue neutral and tenants pay significantly less than Baseline while still getting guaranteed performance.* For instance, with a mean bandwidth demand of 500 Mbps, Figure 15 shows that tenants with virtual clusters pay 68% of Baseline at moderate load and 37% of Baseline at high load (31% and 25% respectively with VOC-10).

The charging model can be generalized from linear bandwidth costs to $NT(k_v + k_b f(B))$, where f is a bandwidth

[4]For Amazon EC2, small VMs cost 0.085\$/hr. Sample estimated prices in our experiments are at 0.04\$/hr for k_v, and 0.00016\$ /GB for k_b.

charging function. We repeated the analysis with other bandwidth functions ($B^{\frac{3}{2}}, B^2$), obtaining similar results.

5.4 Implementation and Deployment

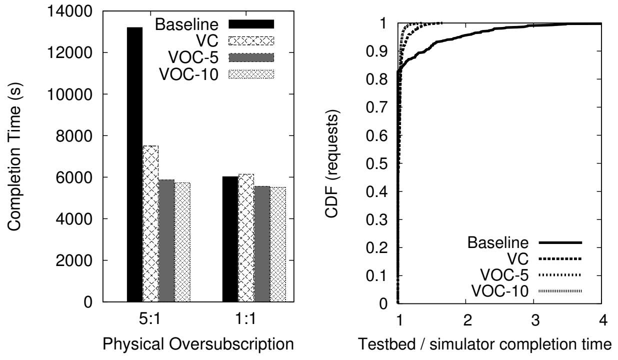
Our Oktopus implementation follows the description in Section 4. The NM maintains reservations across the network and allocates tenant requests in an on-line fashion. The enforcement module on individual physical machines implements the rate computation and rate limiting functionality (Section 4.3). For each tenant, one of the tenant's VMs (and the corresponding enforcement module) acts as a controller and calculates the rate limits. Enforcement modules then use the Windows Traffic Control API [4] to enforce local rate limits on individual machines.

Scalability. To evaluate the scalability of the NM, we measured the time to allocate tenant requests on a datacenter with 10^5 endhosts. We used a Dell Precision T3500 with a quad-core Intel Xeon 5520 2.27 GHz processor and 4 GB RAM. Over 10^5 requests, the median allocation time is 0.35ms with a 99^{th} percentile of 508ms. Note that this only needs to be run when a tenant is admitted, and hence, the NM can scale to large datacenters.

The rate computation overhead depends on the tenant's communication pattern. Even for a tenant with 1000 VMs (two orders of magnitude more than mean tenant size today [31]) and a *worst-case* scenario where all VMs communicate with all other VMs, the computation takes 395ms at the 99^{th} percentile. With a typical communication pattern [20], 99^{th} percentile computation time is 84ms. To balance the trade-off between accuracy and responsiveness of enforcement and the communication overhead, our implementation recomputes rates every 2 seconds. For a tenant with 1000 VMs and *worst-case* all-to-all communication between the VMs, the controller traffic is 12 Mbps ($\sim$1 Mbps with a typical communication pattern). Hence, the enforcement module imposes low overhead.

Deployment. We deployed Oktopus on a testbed with 25 endhosts arranged in five racks. Each rack has a Top-of-Rack (ToR) switch, which is connected to a root switch. Each interface is 1 Gbps. Hence, the testbed has a two-tier tree topology with a physical oversubscription of 5:1. All endhosts are Dell Precision T3500 servers with a quad core Intel Xeon 2.27GHz processor and 4GB RAM, running Windows Server 2008 R2. Given our focus on quantifying the benefits of Oktopus abstractions, instead of allocating VMs to tenants, we simply allow their jobs to run on the host OS. However, we retain the limit of 4 jobs per endhost, resulting in a total of 100 VM or job slots.

We repeat the experiments from Section 5.2 on the testbed and determine the completion time for a batch of 1000 tenant jobs (mean tenant size N is scaled down to 9). As before, each tenant job has a compute time (but no actual computation) and a set of TCP flows associated with it. Figure 16(a) shows that virtual clusters reduce completion time by 44% as compared to Baseline (57% for VOC-10). We repeated the experiment with all endhosts connected to one switch (hence, no physical oversubscription). The bars on the right in Figure 16(a) show that virtual clusters match the Baseline completion time while VOC-10 offers a 9% reduction. Since the scale of these experiments is smaller (smaller topology and tenants), virtual networks do not have much opportunity to improve performance and the reduction in completion time is less significant. However, tenant jobs still get guaranteed network performance and hence, predictable completion times.



(a) Completion Time (b) Cross Validation

Figure 16: Testbed experiments show that virtual networks provide performance gains and validate our simulator.

Cross-validation. We replayed the same job stream in our simulator and for each tenant request, we determined the ratio of the completion time on the testbed and the simulator. Figure 16(b) shows that for the vast majority of jobs, the completion time in the simulator matches that on the testbed. Some divergence results from the fact that network flows naturally last longer in the live testbed than in the simulator which optimally estimates the time flows take. We note that jobs that last longer in the testbed than the simulator occur more often with Baseline than with virtual networks. This is because the Baseline setup results in more network contention which, in turn, causes TCP to not fully utilize its fair share. Overall, the fact that the same workload yields similar performance in the testbed as in the simulator validates our simulation setup and strengthens our confidence in the results presented.

6. RELATED WORK

The increasing prominence of multi-tenant datacenters has prompted interest in network virtualization. Seawall [31] and NetShare [22] share the network bandwidth among tenants based on weights. The resulting proportional bandwidth distribution leads to efficient multiplexing of the underlying infrastructure; yet, in contrast to Oktopus, tenant performance still depends on other tenants. SecondNet [15] provides pairwise bandwidth guarantees where tenant requests can be characterized as $<N, [B_{ij}]_{N\times N}>$; $[B_{ij}]_{N\times N}$ reflects the complete pairwise bandwidth demand matrix between VMs. With Oktopus, we propose and evaluate more flexible virtual topologies that balance the trade-off between tenant demands and provider flexibility.

Duffield et al. [12] introduced the hose model for wide-area VPNs. The hose model is akin to the *virtual cluster* abstraction; however, the corresponding allocation problem is different since the physical machines are fixed in the VPN setting while we need to choose the machines. Other allocation techniques like simulated annealing and mixed integer programming have been explored as part of testbed mapping [29] and virtual network embedding [37]. These efforts focus on allocation of arbitrary (or, more general) virtual topologies on physical networks which hampers their scalability and restricts them to small physical networks (O(10^2) machines).

7. CONCLUDING REMARKS

This paper presents virtual network abstractions that allow tenants to expose their network requirements. This enables a *symbiotic* relationship between tenants and providers; tenants get a predictable environment in shared settings while the provider can efficiently match tenant demands to the underlying infrastructure without muddling their interface. Our experience with Oktopus shows that the abstractions are practical, can be efficiently implemented and provide significant benefits.

Our abstractions, while emulating the physical networks used in today's enterprises, focus on a specific metric– inter-VM network bandwidth. Tenants may be interested in other performance metrics, or even non-performance metrics like reliability. Examples include bandwidth to the storage service, latency between VMs and failure resiliency of the paths between VMs. In this context, virtual network abstractions can provide a succinct means of information exchange between tenants and providers.

Another interesting aspect of virtual networks is cloud pricing. Our experiments show how tenants can implicitly be charged for their internal traffic. By offering bounded network resources to tenants, we allow for *explicit and fairer* bandwidth charging. More generally, charging tenants based on the characteristics of their virtual networks eliminates hidden costs and removes a key hindrance to cloud adoption. This, in effect, could pave the way for multi-tenant datacenters where tenants can pick the trade-off between the performance of their applications and their cost.

8. REFERENCES

[1] Amazon EC2 Spot Instances. `http://aws.amazon.com/ec2/spot-instances/`.

[2] Amazon Cluster Compute, Jan. 2011. `http://aws.amazon.com/ec2/hpc-applications/`.

[3] Amazon's EC2 Generating 220M, Jan. 2011. `http://bit.ly/8rZdu`.

[4] Traffic Control API, Jan. 2011. `http://msdn.microsoft.com/en-us/library/aa374468%28v=VS.85%29.aspx`.

[5] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proc. of ACM SIGCOMM*, 2008.

[6] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *Proc. of USENIX NSDI*, 2010.

[7] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris. Reining in the Outliers in Map-Reduce Clusters using Mantri. In *Proc. of USENIX OSDI*, 2010.

[8] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards Predictable Datacenter Networks. Technical Report MSR-TR-2011-72, Microsoft Research, May 2011.

[9] R. Black, A. Donnelly, and C. Fournet. Ethernet Topology Discovery without Network Assistance. In *Proc. of ICNP*, 2004.

[10] B. Craybrook. Comparing cloud risks and virtualization risks for data center apps, 2011. `http://bit.ly/fKjwzW`.

[11] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An Engineering Approach*. Elsevier, 2003.

[12] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merive. A flexible model for resource management in virtual private networks. In *Proc. of ACM SIGCOMM*, 1999.

[13] A. Giurgiu. Network performance in virtual infrastrucures, Feb. 2010. `http://staff.science.uva.nl/~delaat/sne-2009-2010/p29/presentation.pdf`.

[14] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. In *Proc. of ACM SIGCOMM*, 2009.

[15] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees. In *Proc. of ACM CoNext*, 2010.

[16] M. Hajjat, X. Sun, Y.-W. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani. Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud. In *Proc. of SIGCOMM*, 2010.

[17] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn. Case study for running HPC applications in public clouds. In *Proc. of ACM Symposium on High Performance Distributed Computing*, 2010.

[18] A. Iosup, N. Yigitbasi, and D. Epema. On the Performance Variability of Production Cloud Services. Technical Report PDS-2010-002, Delft University of Technology, Jan. 2010.

[19] S. Kandula, J. Padhye, and P. Bahl. Flyways To Decongest Data Center Networks. In *Proc. of HotNets*, 2005.

[20] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The Nature of Data Center Traffic: Measurements & Analysis. In *Proc. of ACM IMC*, 2009.

[21] D. Kossmann, T. Kraska, and S. Loesing. An Evaluation of Alternative Architectures for Transaction Processing in the Cloud. In *Proc. of international conference on Management of data (SIGMOD)*, 2010.

[22] T. Lam, S. Radhakrishnan, A. Vahdat, and G. Varghese. NetShare: Virtualizing Data Center Networks across Services. Technical Report CS2010-0957, University of California, San Deigo, May 2010.

[23] A. Li, X. Yang, S. Kandula, and M. Zhang. CloudCmp: comparing public cloud providers. In *Proc. of conference on Internet measurement (IMC)*, 2010.

[24] D. Mangot. Measuring EC2 system performance, May 2009. `http://bit.ly/48Wui`.

[25] X. Meng, V. Pappas, and L. Zhang. Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. In *Proc. of Infocom*, 2010.

[26] Michael Armburst et al. Above the Clouds: A Berkeley View of Cloud Computing. Technical report, University of California, Berkeley, 2009.

[27] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. Mogul. SPAIN: COTS Data-Center Ethernet for Multipathing over Arbitrary Topologies. In *Proc of NSDI*, 2010.

[28] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren. Cloud control with distributed rate limiting. In *Proc. of ACM SIGCOMM*, 2007.

[29] R. Ricci, C. Alfeld, and J. Lepreau. A Solver for the Network Testbed Mapping problem. *SIGCOMM CCR*, 33, 2003.

[30] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz. Runtime measurements in the cloud: observing, analyzing, and reducing variance. In *Proc. of VLDB*, 2010.

[31] A. Shieh, S. Kandula, A. Greenberg, and C. Kim. Sharing the Datacenter Network. In *Proc. of USENIX NSDI*, 2011.

[32] P. Soares, J. Santos, N. Tolia, and D. Guedes. Gatekeeper: Distributed Rate Control for Virtualized Datacenters. Technical Report HP-2010-151, HP Labs, 2010.

[33] E. Walker. Benchmarking Amazon EC2 for high-performance scientific computing. *Usenix Login*, 2008.

[34] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan. c-Through: Part-time Optics in Data Centers. In *Proc. of ACM SIGCOMM*, 2010.

[35] G. Wang and T. S. E. Ng. The Impact of Virtualization on Network Performance of Amazon EC2 Data Center. In *Proc. of IEEE Infocom*, 2010.

[36] H. Wang, Q. Jing, S. Jiao, R. Chen, B. He, Z. Qian, and L. Zhou. Distributed Systems Meet Economics: Pricing in the Cloud. In *Proc. of USENIX HotCloud*, 2010.

[37] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking Virtual Network Embedding: substrate support for path splitting and migration. *SIGCOMM CCR*, 38, 2008.

[38] M. Zaharia, A. Konwinski, A. D. Joseph, Y. Katz, and I. Stoica. Improving MapReduce Performance in Heterogeneous Environments. In *Proc. of OSDI*, 2008.