Hypervisor-based Traffic Observation for Fair Bandwidth Allocations in Datacenter

Ilya Nikolaevskiy (ilya.nikolaevskiy@aalto.fi, Aalto University, Finland)

Markus Palonen (markus.palonen@aalto.fi, Aalto University, Finland)

Andrey Lukyanenko (andrey.lukyanenko@aalto.fi, Aalto University, Finland)

Andrei Gurtov (gurtov@hiit.fi, Helsinki Institute for Information Technology, Finland)

Abstract

Fair sharing of bandwidth among tenants in datacenters is important to guarantee prompt execution while providing isolation between different jobs. Many existing methods require tenants to provide explicit bandwidth demands which is not always possible. We demonstrate that these demands can be estimated on hypervisors by real-time observation of a traffic. With implementation we show feasibility of fair bandwidth allocation mechanism based on real time traffic measurement.

Introduction

Recently, there is a significant interest in datacenter network allocation. Allocations aim to improve datacenter network utilization, to provide bandwidth guarantees and to incentivize tenants to share the infrastructure.

Today, a tenant pays to cloud providers on per-VM basis. Recent studies show that network resources are not provided uniformly and fairly, many tenants experience high diversity in network bandwidth [1]. The tenants' requirement for network quantitative guarantees resulted in many allocation algorithms. These allocations aim to achieve a set of desired properties, namely, high-utilization, min-bandwidth guarantees, fairness.

Out of multiple allocation algorithms, we highlight two classes -- static and dynamic allocations. Dynamic allocations primary provide weights to the data transfer rates at which TCP congestion control operates. NetShare [2], FairCloud [3], and Seawall [4] are examples of dynamic allocation in the literature. On another hand, static allocations use pre-calculated at admission controller bandwidth reservations. Examples of such allocation methods are Oktopus [1] and SecondNet [5]. These reservations tenants request as a simplified model, at the same time provider places these models into the network. Two main forms of static allocation models are hose-model and matrix-model. Finally, some of the allocation methods attempt to utilize benefits of both classes and provide a hybrid class of dynamic and static allocation.

Among the benefits dynamic allocations provide – as any implementation of TCP protocol – is high-utilization: these allocations have work-conserving property. On another hand the isolation and, as a result, min-bandwidth guarantee are very hard to impossible to achieve with dynamic methods. Static allocations produce better guarantees for the tenants; the tenant which has static allocation will have hard bandwidth guarantees, which no any other tenant can break.

However, most proposed allocation methods suppose that users' demands for traffic are known a priori. This is rarely the case because any complex distributed application is hard to predict. We argue that correct solution for bandwidth allocation problem should estimate demands in real time.

STEM In a Nutshell

We propose a novel fair and strategy-proof bandwidth allocation method which doesn't require traffic demands from users to operate – Strategy-proof Task-Enforcement Mechanism (STEM). This mechanism is based around the notion of *tasks*. Task is a set of related flows, where user is interested only in completion time of slowest of them, but not in individual completion times. Such related flows often emerge in a scatter-and-gather workflow, where some data have to be partitioned between processing machines, and results have to be gathered back. In that situation application performance is bounded by straggler flows.

We consider that each tenant has only one job. Utility of each tenant is proportional to a minimal ratio of amount of allocated resource to amount of demanded among all resources. We call a set of tenant's demands for all resources — a demand vector. Because each tenant desires to get as much resources as possible, we are not interested in absolute values of tenants' demands. We have to consider only their relative values. Therefore, demand vectors of all tenants should be normalized. We suggest to divide all demands of a single tenant by the maximal demand of her. This results in all values of demand vectors to not exceed 1. As in DRF [6], we call the resource with the maximum demand dominant resource for each tenant, as it dominants her demands. To remove requirement of prior demands estimation by users, each demand is considered to be either 0 or 1, depending on if tenant needs that particular link for her flow.

To achieve strategy-proof allocation and exclude any possibility of misbehavior we base our allocation mechanism on DRF [6]: utilities of all tenants are forced to be equal, then maximum possible common utility is found subject to constraints of links bandwidth.

To make allocation work-conserving and increase datacenter utilization we do not rate-limit all flows to calculated rates, instead we use two classes of priority: part of each flow is marked as a high-priority traffic and all excess traffic is marked as a low-priority traffic. Therefore, if all switches in the datacenter have two separate outgoing queues, all priority traffic will never be dropped due to congestion, as rates are calculate in such a way. This way each tenant has a strategy-proof and fair minimum bandwidth guarantee, while low-priority traffic will occupy any remaining capacity without obstruction of calculated minimum bandwidth guarantees.

Implementation

In our implementation hypervisors are observing outgoing traffic and notify central controller about all existing outgoing flows. Central controller calculates allocations and broadcasts them to all hypervisors (See Fig. 1). This broadcast is very light, as allocation is essentially a single value for all tenants. Then hypervisors are marking corresponding part of each flow as a high-priority traffic.

Our implementation is observing only a single parameter – existence of a flow. However, many other properties can be measured as well, for example: rate demand of a flow, class of a data, flow owner's payment and other parameters. Our allocation mechanism does not need them, however other allocation mechanisms may use these different properties to provide more sophisticated and precise bandwidth allocation.

To reduce load on a central controller, hypervisors are signaling about new or expired flows only several times per second. Before response from central controller is received about new flows they are entirely marked as low-priority traffic. In that way existing but not confirmed yet flows will not degrade minimum bandwidth guarantee of other flows.

Our implementation uses iptables NFQUEUE mechanism to run custom C code to process outgoing traffic. Our code observes and modify egress traffic, marking some part of it as a hi-priority packets. It notifies

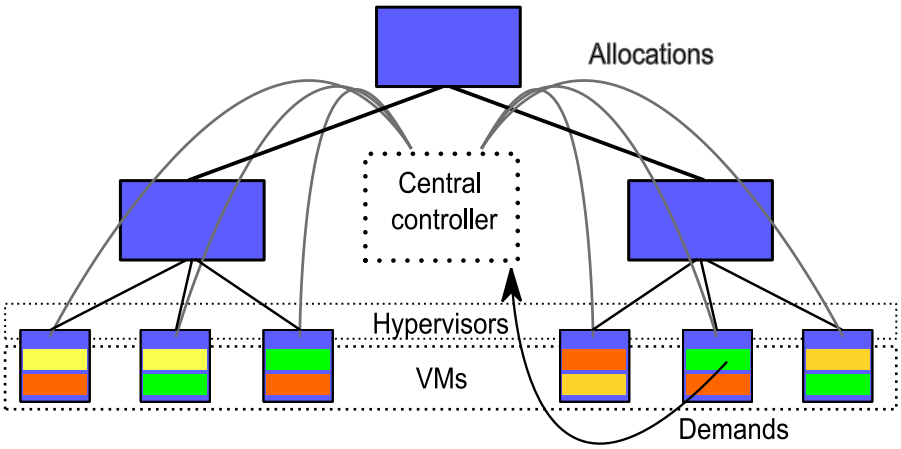

Figure 1: Sample architecture of an allocation system.

central controller, written in a python, about new and expired flows. Central controller, knowing topology and existing flows, calculates allocations and broadcasts them. We use mininet network emulator to simulate sample datacenter topology.

Conclusion

In this work we show that even without prior demands estimation from users it is possible to allocate bandwidth in a datacenter fairly in a strategy-proof way. Correct online measurement of traffic properties and informing most important parameters out of it to the controller makes the shaping of the traffic more sophisticated and optimized.

References

[1] Ballani, H., Costa, P., Karagiannis, T., and Rowstron, A. Towards predictable datacenter networks. In Proceedings of the ACM SIGCOMM 2011 conference (New York, NY, USA, 2011), SIGCOMM '11, ACM, pp. 242—253.

[2] V. T. Lam, S. Radhakrishnan, R. Pan, A. Vahdat, and G. Varghese. Netshare and stochastic netshare: Predictable bandwidth allocation for data centers. SIGCOMM Comput. Commun. Rev., 42(3):5—11, June 2012.

[3] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica. Faircloud: sharing the network in cloud computing. In Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, SIGCOMM '12, pages 187—198, New York, NY, USA, 2012. ACM.

[4] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha. Sharing the data center network. In Proceedings of the 8th USENIX conference on Networked systems design and implementation, NSDI'11, pages 23—23, Berkeley, CA, USA, 2011. USENIX Association.

[5] 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 Proceedings of the 6th International COnference, Co-NEXT'10, pages 15:1–15:12, New York, NY, USA, 2010. ACM.

[6] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, Dominant resource fairness: fair allocation of multiple resource types, in Proceedings of the 8th USENIX conference on Networked systems design and implementation, ser. NSDI'11. USENIX Association, 2011, pp. 24—24.