Silo: Predictable Message Latency in the Cloud – Public Review

Teemu Koponen
Styra, Inc.
koponen@styra.com

Application user experience benefits from predictable infrastructure performance. Whether it is a compute, storage, or network resource, any variance in performance is eventually to be exposed to users, unless applications are designed to tolerate the variance. Applications hosted in private datacenters have this design goal less frequently, however; in these environments over-provisioning of the resources is still an economical approach to guarantee the performance – contrary to the cloud in which high resource utilization is a paramount objective. As a result, predictable infrastructure performance emerges as a key factor to reduce the pressure to redesign applications for the cloud.

In multi-tenant clouds networking guarantees have proven particularly challenging. While the community has produced several incrementally deployable proposals to provide bandwidth guarantees in multi-tenant datacenters, the story with managing queueing delays is different: either the proposals lack in isolation between tenants, they assume cooperation among tenants (through use of a single transport protocol), or they require new queueing disciplines at network switches altogether.

In this paper, the authors tackle this problem and describe a system called Silo that provides per tenant network guarantees for bandwidth, delay, *and* burst allowance, all implemented in software at virtual switches, without assumptions about tenants or their network stacks. Silo builds on careful placement of VMs using a placement algorithm that maps tenants' guarantees to physical network switch queue capacities and delays. The algorithm admits VMs only if their network guarantees can be met. It is two insights that make Silo feasible. First, per network calculus, by controlling packet sending rates (of each tenant at the network edge), the algorithm can bound the datacenter network switch queues. Second, establishing tight queue bounds translates to a requirement of accurate pacing of packets at line speed. To accomplish this, Silo uses an elegant trick: virtual switches fill the gaps between VM transmitted packets with invalid *void* packets. The resulting line rate packet streams are sent to first-hop switches which then drop the void packets leaving the valid packets accurately paced.

The design of Silo has its limitations. In particular, while carefully optimized, Silo packet pacer consumes CPU resources which would be better used for tenant workloads, and it also requires disabling NIC TSO which further reduces CPU time available for the workloads. Hardware offloading of pacing to NICs seems still more practical. In the end, these limitations do not shadow the architectural takeaway of the paper: the last words on the division of functional responsibilities between datacenter network switches, NICs, and host software are still to be written.