Performance-Aware Cluster Scheduling

Diana Andreea Popescu
diana.popescu@cl.cam.ac.uk
University of Cambridge, UK

Andrew W. Moore
andrew.moore@cl.cam.ac.uk
University of Cambridge, UK

Given the network latency variability observed in data centers, applications' performance is also determined by their placement within the data centre. We present NoMora, a cluster scheduling architecture whose core is represented by a latency-driven, application performance-aware cluster scheduling policy. The policy places the tasks of an application taking into account the expected performance based on the measured network latency between pairs of hosts in the data center. Furthermore, if a tenant's application experiences increased network latency, for example due to network congestion, and thus lower application performance, their virtual machines may be migrated to a better placement. We show that our policy improves the overall average application performance compared to other policies.

Different mechanisms to provide predictable network bandwidth and latency for tenants to ensure predictable performance for applications have been proposed over the years [5–7]. In this work, we change the viewpoint: if the tenant wants a certain performance for their application, what network conditions does the application need?

Architecture. We combine three elements in the NoMora cluster scheduling architecture. *1. Functions that predict application performance dependent upon network latency.* Applications react differently to network latency [2, 8]. We modeled the relationship between application performance and network latency through functions that describe this relationship, using polynomial fitting on the data obtained through experiments run on a custom testbed. Such a performance function can be described as $f(network\_latency) = normalised\_application\_performance$ where f is a polynomial function. While we determined these dependencies through experiments for particular applications and experimental settings, we expect that the tenants could specify them, or these functions could be inferred automatically through recent machine learning techniques.

2. Network latency measurement system. Data center network latency measurement systems like Pingmesh [1], or PTPmesh [3] can provide the most recently measured network latency between hosts.

3. Latency-driven, application performance-aware, policy. We propose a new latency-driven, application performance-aware policy, whose goal is to place distributed applications in a data centre in a manner that gives them improved application performance. The cluster scheduling problem was modeled as a minimum-cost maximum-flow problem in previous work [4], and we describe briefly how this mapping takes place. A submitted task $T_{i,j}$, representing task j of job J_i, is represented by a vertex in the graph (flow network), and it generates one unit of flow. A sink S drains the flow generated by the submitted tasks. A task vertex sends a unit of flow along a path composed of directed arcs in the graph to the sink S. The path can pass through a vertex that correponds to a machine (host) M_m, meaning the task is scheduled to run on that machine, or it can pass through a special vertex for the unscheduled tasks of that job U_i, meaning that the task is not scheduled. The cost of an arc between a task and a machine is assigned according to a scheduling policy. The min-cost max-flow solver operates on the graph, the tasks being scheduled on machines according to the cost on the arcs by routing flow through the graph to the sink.

Next, we give a high level overview of our scheduling policy. When a job is submitted, the root task $T_{i,0}$ of the application (the server or the master) is scheduled immediately on any available machine. After being scheduled, the root task has an arc in the graph to the machine it is running on. The other jobs' tasks are waiting for the root to be scheduled first, and they do not have any arcs towards resources initially. After the root task is scheduled, each task $T_{i,j}, j > 0$ (the clients or the workers) will be scheduled. Assuming the root task is running on machine M_{root}, for a task $T_{i,j}$ that can be scheduled on machine M_m, the cost of the arc from $T_{i,j}$ to M_m is computed as $c(T_{i,j}, M_m) = 1/f(max(latency(M_{root}, M_m)))$, where the f value is the expected application performance for the measured network latency between machine M_{root} and machine M_m. We invert the f value, since if the performance is higher, the cost on the arc should be lower, meaning the task will be scheduled using that arc. Since there are multiple paths between two machines in data centers, due to Equal-Cost Multi-Path (ECMP), we cannot know which of the available paths between VMs will be taken by the application's flows. Thus, to be conservative, we use the maximum network latency value measured between the two machines.

Results. We compute the average application performance for the job's total runtime as the application performance determined by the measured network latency in every measurement interval divided by the maximum application performance that could be achieved in every measurement interval. Using the Google [9] cluster workload trace with 12,500 machines, augmented with the application performance predctions dependent upon network latency derived from our experiments, the NoMora policy improves overall average application performance by up to 13%, and up to 42.4% when VM migration is enabled, compared to a random placement policy.

References

[1] C. Guo *et al.* 2015. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In *ACM SIGCOMM 2015*.

[2] D. A. Popescu *et al.* 2017. *Characterizing the impact of network latency on cloud-based applications' performance.* Technical Report UCAM-CL-TR-914. University of Cambridge, Computer Laboratory.

[3] D. A. Popescu *et al.* 2017. PTPmesh: Data Center Network Latency Measurements Using PTP. In *IEEE MASCOTS*. IEEE.

[4] I. Gog *et al.* 2016. Firmament: Fast, Centralized Cluster Scheduling at Scale. In *OSDI 16*. 99–115.

[5] K. Jang *et al.* 2015. Silo: Predictable Message Latency in the Cloud. In *SIGCOMM 2015 (SIGCOMM '15)*. ACM, New York, NY, USA, 435–448.

[6] Kumar, Praveen *et al.* 2019. PicNIC: Predictable Virtualized NIC. In *Proceedings of the ACM SIGCOMM (SIGCOMM '19)*. ACM, New York, NY, USA, 351–366.

[7] Mogul, Jeffrey C. *et al.* 2012. What We Talk About when We Talk About Cloud Network Performance. *SIGCOMM Comput. Commun. Rev.* 42, 5 (Sept. 2012), 44–48.

[8] N. Zilberman *et al.* 2017. Where Has My Time Gone?. In *Passive and Active Measurement (PAM)*. Springer.

[9] John Wilkes. 2011. More Google cluster data. Google research blog. (Nov. 2011). Posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.