Efficient Coflow Scheduling Without Prior Knowledge — Public Review

Hitesh Ballani
Microsoft Research
hitesh.ballani@microsoft.com

A lot of blood, sweat and tears have been shed in the quest to improve network performance, mostly in terms of flow completion times. But this race to the top has meant that we have been guilty of forgetting what really matters — application performance. Applications have different notions of the utility they derive from flow completion, so determining the right network metric to optimize is a tricky proposition. In 2012, Chowdhury and Stoica proposed the Coflow abstraction to concisely capture application-level performance semantics at the network layer. A "coflow" refers to a set of flows with a collective goal. For example, the all-or-nothing property of data-parallel jobs entails that coflow completion is what a job really cares about, not when individual flows within a coflow finish. This is a very neat observation!

The coflow abstraction leads to the coflow scheduling problem, i.e., how should network traffic be scheduled to minimize coflow completion times? This is an NP-hard problem. Two solutions were proposed at Sigcomm 2014, Varys and Baraat. Varys reduces coflow completion times by scheduling them in order of their total size. This relies on a tightly-coupled centralized design and assumes flow sizes are known a priori. Instead, Baraat adopts a FIFO-like scheduling policy that lends itself to decentralized implementation but at the expense of performance.

This paper addresses the shortcomings of these past efforts. It presents a non-clairvoyant coflow scheduling policy which does not require knowledge of flow sizes, while not giving up on any performance to avoid head-of-line blocking. The main idea is to approximate the least-attained service (LAS) scheduling. Coflows start in the highest priority queue but are gradually demoted to lower priority as they send more bytes. Determining a coflow's current size necessitates a coordination point. However, the priority levels are discrete which means the coordination can be done loosely. The authors also describe Aalo, a system that implements such coflow scheduling. The evaluation, performed with EC2 experiments and simulations, shows that Aalo's non-clairvoyant scheduling performs close to schedulers with complete information about coflows.

The reviewers agreed that Aalo represents the natural next step in coflow scheduling. They appreciated that the design is simple yet it accounts for several practical details, and it is thoroughly evaluated. By not requiring any a priori information about coflows, Aalo achieves efficient scheduling in spite of operational dynamics due to multi-stage jobs, multi-wave scheduling, failures and speculative execution. Figure 9 in the paper illustrates this very well — for multi-wave scheduling, Aalo can even improve upon a clairvoyant approach like Varys. This may sound counter intuitive at first but the paper explains why Aalo is better positioned to accommodate any runtime dynamics.

Aalo's impressive performance comes at a cost though. A key concern is the need for (loose) coordination. Scaling to large clusters will require a high coordination period. This restricts the class of applications that Aalo can accommodate. For example, data center traffic resulting from online search queries can be modeled as coflows but would be very hard for Aalo to schedule.

On the analytical front, several questions remain. First, least-attained service scheduling is known to be optimal amongst non-clairvoyant flow scheduling policies for heavy tailed distributions. Is this true across coflows too? Second, while the PIAS work at NSDI 2015 showed how optimal priority thresholds can be determined when scheduling flows, doing the same for coflows is an open question. Finally, while the evaluation shows impressive results across a blocking fabric, the analysis models the data center as one non-blocking switch. Coflow scheduling in settings where this assumption does not hold seems harder.

In summary, Aalo takes coflow scheduling a significant step closer to reality by addressing serious concerns afflicting existing solutions. The open questions raised by the paper means this will be not be last word we hear on the topic, so stay tuned.