In-Network Computation is a Dumb Idea Whose Time Has Come

HotNets-XVI Dialogue

Jeff Mogul and Jitu Padhye

JM: A lot of people seem to be interested in exploiting – some might say "abusing" – P4 and similar reconfigurable switch hardware to do things besides packet-switching.

JP: The paper does start out by framing an important question: "what kinds of computation should be delegated to the network?" But does it answer that question, or does it just address a small aspect of it?

JM: They did motivate their approach with examples from machine learning and graph analytics, which are both increasingly heavy users of data-center networks.

JP: True, but we both know that real data-center networks are almost never operated at high utilizations except for brief periods, such as incast. It's not the applications themselves that I find unrealistic; it's that the techniques the authors introduce seem to be most useful for workloads where the network is heavily loaded, and I don't think the sorts of applications the authors use to motivate their work will load the network heavily for long periods. So in that sense, I don't buy the problem setting offered.

JM: Reducing network traffic *per se* isn't the only benefit. They point out that reducing the network traffic also reduces the load on the receiving host CPUs, which is potentially useful because it allows you to do (say) a MapReduce job with fewer reducer hosts. Are these big-data workloads really that sensitive to the amount of computing at the receiver?

JP: We have some experience with a variety of these workloads. Sometimes the receiver load matters, sometimes it doesnt.

JP: Another thing to consider is the question of resource consumption. The paper makes an analogy to using GPUs. However, GPUs are provably more cost-effective and power-efficient than CPUs for some workloads. P4 chips are really a lot more expensive than CPUs, and probably more power-hungry per computation step. Also, the ratio of CPUs to switch chips in any realistic data-center network is quite large, so you wouldn't expect to find a lot of P4 resources available.

JM: I also wondered whether the paper looked enough at the competition for switch resources, such as table space, between the hash tables used for aggregation functions, and the more network-specific uses of switch memory. Could this memory be more useful for supporting finer-grained forwarding rules, or for collecting more interesting statistics?

JM: Also, the paper does mention the use of in-network computation to support algorithms such as consensus, but then sort of drops that line. Perhaps – especially if you are worried more about application latency than network utilization – this is ultimately going to be a more exciting line to follow than simply reducing the data that flows through the network? Saving RTTs can be a bigger deal than saving bytes.

JP: Yes, that actually makes sense – if you can do certain logically-centralized operations inside the network, you're saving RTTs, and being on the path thats common to a lot of machines gives you an advantage for this kind of thing.

JM: We should also discuss whether we think its plausible that this stuff could be made tolerant to packet loss or switch failure, without losing all of the benefits. The Eris paper in SOSP 2017, which applies this approach to consensus, uses an "epoch number" mechanism to tolerate switch failures. But if the switches

are aggregating data from a graph analytics job, a failed switch probably means restarting some or all of the entire job.

JP: Encryption might also be an issue – it's not that a lot of people do encryption in the data center yet, but thats changing. We're doing that, among other reasons, because if a malicious actor can break into one switch, they can snoop on a lot of traffic – again, because the ratio of hosts to switches is large.

JM: So I think summing up, we like the overall goal of the paper, to figure out what kinds of in-network computation make sense. However, the paper kind of veered into addressing a class of problems where the challenges are intellectually interesting, but they didn't convince us that this actually makes sense in practice. We think there is room to explore a broader class of applications, including those that are RTT-sensitive, for in-network computing. We would also be really interested to see solutions for the fault-tolerance and encryption challenges.