Article Review

Swift: Delay is Simple and Effective for Congestion Control in the Datacenter

Summary

The author introduces a congestion control framework named Swift, which employs delay as the congestion signal for network congestion control in data centers. Swift divides delay into two types: fabric delay and end-host delay, and adapts the window size (fcwnd and ecwnd) with an Additive-Increase Multiplicative-Decrease (AIMD) algorithm for each delay type to achieve congestion control. Unlike other techniques that rely on switch feedback for adjustment, such as DCTCP, Swift can adapt dynamically to various software and hardware changes in large data centers, including topology or application modifications, making it more flexible in deployment. Based on the results from small-scale experiments and production experience, Swift outperforms methods that require switch feedback for congestion signals in all aspects, effectively managing large-scale incast, achieving lower latency (particularly in long tail), and better bandwidth utilization and fairness.

Strength and Weakness

Strength:

1. The paper analyzes delay at two levels, fabric and end-host, and offers tailored adjustments to the congestion window based on the source and attributes of delay.
2. The paper introduces a mode (congestion window size ¡ 1) that incorporates pacing to handle large-scale incast situations.
3. The proposed method has high feasibility and flexibility in deployment, as it only requires the timestamp of the NIC to measure delay and can be used in conjunction with other congestion control methods.
4. The paper's comprehensive experimental analysis, including small testbed experiments and actual deployment in data centers, demonstrates the effectiveness of the proposed method in achieving lower Latency (especially in the long tail), high IOPS, high throughput, and low Loss rate.

Weakness:

1. The paper lacks an analysis of an analytical model, which means there is no theoretical guarantee.
2. The experiment only tests in Google's data center and hides a lot of information, making it difficult to determine if Swift is suitable in other locations.
3. Tuning parameters before deploying Swift can take extra time and is one of the additional costs to achieve the best performance.

Questions

1. This method relies on the timestamp provided by the NIC to provide accurate timing. If there is suddenly a large error in the time measurement, it will cause a huge impact. How can this be resolved?
2. The paper has not mentioned how to adjust the end-host target delay. How should this be calculated?

Conclusion

The idea of using delay as a congestion signal has been discussed in previous literature.[1] However, this paper takes it a step further by dividing delay into the fabric and end-host components to calculate and adjust congestion windows. This innovative approach is the most notable contribution of this article.

[1] Mittal et al. "TIMELY: RTT-based congestion control for the datacenter." In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication.*