Solving TCP Incast in Cluster Storage Systems

Vijay Vasudevan*, Hiral Shah*, Amar Phanishayee*, Elie Krevat*
David Andersen, Greg Ganger, Garth Gibson
vrv@cs.cmu.edu, hsshah@andrew.cmu.edu, amarp@cs.cmu.edu, ekrevat@cs.cmu.edu
dga@cs.cmu.edu, ganger@ece.cmu.edu ,garth@cs.cmu.edu
Carnegie Mellon University

Abstract

Synchronized request workloads are becoming increasingly common in today's commodity clusters. Examples include parallel reads/writes in cluster filesystems such as Lustre, Panasas [6], or pNFS [5]; search queries sent to dozens of nodes, with results returned for sorting [1]; or parallel databases that harness multiple back-end nodes to process parts of queries [3].

Recent work has shown that when using TCP for these workloads, these networks experience Incast: a severe throughput collapse due to TCP timeouts triggered by severe losses at Ethernet switches [4]. One promising TCP-level technique is to reduce the minimum retransmission (*RTO*) duration by three orders of magnitude, from 200ms down to microseconds. Previous work suggested these benefits in simulation, but left unaddressed the more practical issues of safety, overhead, and generality posed by this solution.

We show that the standard TCP implementation does not provide fine-grained timers and cannot reduce the *RTO* enough to avoid Incast; standard TCP implementations do not keep track of time at microsecond granularity and thus cannot timeout on timescales appropriate to cluster storage environments.

In this work, we leverage high-resolution kernel timers [2] to provide fine-grained, microsecond resolution TCP round-trip-times and timeouts, demonstrating that TCP Incast can be avoided for up to 47 concurrent senders (Figure 1). When using the current "jiffy"-based TCP timers, the minimum retransmission timeout is bounded below by 5ms and results in suboptimal throughput as one increases the number of servers. As storage environments move towards 10Gbps Ethernet with smaller round trip times, fine-grained TCP timers are both necessary for and effective in avoiding Incast.

These techniques allow today's cluster storage solutions to continue to use TCP on commodity Ethernet hardware for increasing stripe widths, while simultaneously avoiding TCP Incast throughput collapse.

*Denotes Student Authors

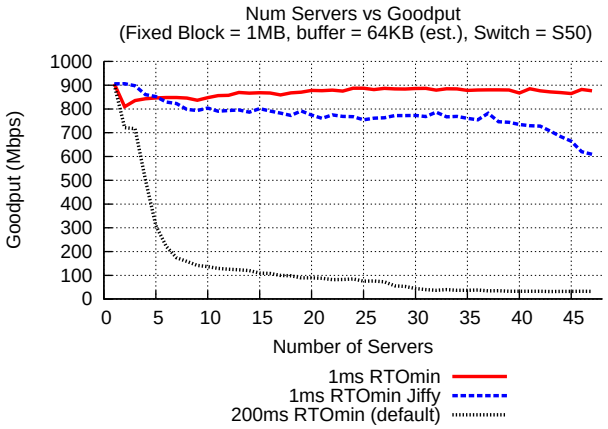

Figure 1: TCP Incast: A throughput collapse occurs with increased numbers of concurrent senders in a synchronized TCP request. Reducing *RTO* to microsecond granularity alleviates Incast for up to 47 concurrent senders. Measurements taken on a 48 Linux 2.6.28 nodes, all attached to a single switch.

References

[1] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proc. 6th USENIX OSDI*, Dec. 2004.

[2] hrtimers. High-resolution timer subsystem. `http://www.tglx.de/hrtimers.html`.

[3] L. Huston, R. Sukthankar, R. Wickremesinghe, M. Satyanarayanan, G. Ganger, E. Riedel, and A. Ailamaki. Diamond: A storage architecture for early discard in interactive search. In *Proc. 3rd USENIX Conference on File and Storage Technologies*, Mar. 2004.

[4] A. Phanishayee, E. Krevat, V. Vasudevan, D. G. Andersen, G. R. Ganger, G. A. Gibson, and S. Seshan. Measurement and analysis of TCP throughput collapse in cluster-based storage systems. In *Proc. USENIX Conference on File and Storage Technologies*, Feb. 2008.

[5] S. Shepler, M. Eisler, and D. Noveck. NFSv4 Minor Version 1 – Draft Standard. URL `http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-minorversion1-13.txt`.

[6] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Zelenka, and B. Zhou. Scalable Performance of the Panasas Parallel File System. In *Proc. USENIX Conference on File and Storage Technologies*, Feb. 2008.