Research Article Impact Factor: 0.621 ISSN: 2319-507X
CK Raina, IJPRET, 2014; Volume 2 (8): 109-116 IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

AVOIDANCE OF THE CONGESTION IN DATA CENTER NETWORK

C.K. RAINA, PROF. M.Y .JOSHI

Department of Computer Science and Engineering, AIT Chandigarh, Gharuan

Accepted Date: 15/02/2014 ; Published Date: 01/04/2014

Abstract: Transport Control Protocol TCP) incast congestion happens when large numbers of senders work in parallel with the same server where the high bandwidth and low latency network problem occurs. For many data center network application such as search, the heavy traffic is present on such server. Incast congestion degrades the performance as the packets are lost at server side due to buffer overflow and hence the response time will be more. To improve the performance this paper is focusing on the TCP throughput, RTT, receive window and retransmission. In this paper our method is to adjust the TCP receive window proactively before Packet loss occurs. If the packets are loss After the increment of window then we will use the retransmission phenomenon which retransmits the lost packets and then it will do store and forward packet method to send the complete data.

Keywords: Data-center networks, in cast congestion, TCP, retransmission.

PAPER-QR CODE

Corresponding Author: Mr. C.K. RAINA

Access Online On:

www.ijpret.com

How to Cite This Article:

CK Raina, IJPRET, 2014; Volume 2 (8): 109-116

INTRODUCTION

The root cause of TCP incast collapse is that the highly burst traffic of multiple TCP connections overflows the Ethernet switch buffer in a short period of time, causing intense packet loss and thus TCP retransmission and timeouts. Previous solutions focused on either reducing the wait time for packet loss recovery with faster retransmissions [2], or controlling switch buffer occupation to avoid overflow by using ECN and modified TCP on both the sender and receiver sides [5]. This paper focuses on avoiding packet loss before incast congestion, which is more appealing than recovery after loss. Of course, recovery schemes can be complementary to congestion avoidance. The smaller the change we make to the existing system, the better. To this end, a solution that modifies only the TCP receiver is preferred over solutions that require switch and router support (such as ECN) and modifications on both the TCP sender and receiver sides. Our idea is to perform incast congestion avoidance at the receiver side by preventing incast congestion. The receiver side is a natural choice since it knows the throughput of all TCP connections and the available bandwidth. The receiver side can adjust the receive window size of each TCP connection, so the aggregate burstiness of all the synchronized senders are kept under control. We call our design Incast congestion Control for TCP (ICTCP). However, adequately controlling the receive window is challenging: The receive window should be small enough to avoid incast congestion, but also large enough for good performance and other non incast cases. A well-performing throttling rate for one incast scenario may not be a good fit for other scenarios due to the dynamics of the number of connections, traffic volume, network conditions, etc. This paper addresses the above challenges with a systematically designed ICTCP. We first perform congestion avoidance at the system level. We then use the per-flow state to finely tune the receive window of each connection on the receiver side**.**

A. TCP Incast Congestion

Show the data-center network structure. There are three layers of switches/routers: the top of rack(ToR) switch, the Aggregate switch, and the Aggregate router. Top of Rack connected to dozen of severs , the number of severs under the same ToR ranges from 44 to 48 , and the ToR switch is a 48- port Giga bits switch with one or multiple 10-Gb uplink. Incast congestion happen when multiple sending severs under The same ToR switch send data to one receive sever. each connection has same traffic amount with the number of senders increasing, which is used in another setup, in which the total traffic amount of all senders is a fixed one, so that the data volume per server per round decreases when the number of senders increases

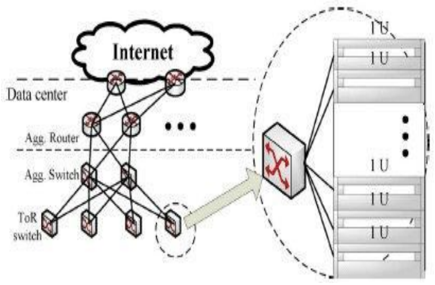

Fig.1 Data-center network and a detail of a ToR switch Connected to multiple rack- mounted sever

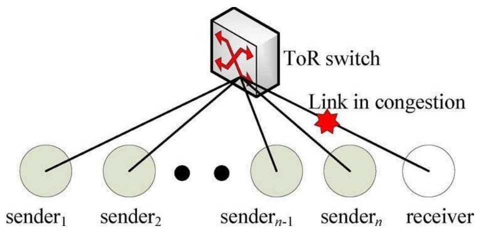

Fig .2 . Scenario of incast congestion in data-center networks, where multiple (n) TCP senders transmit data to the same receiver under the same ToR switch.

TCP throughput is severely degraded by incast congestion since one or more TCP connections can experience timeouts caused by packet drops. TCP variants sometimes improve performance, but cannot prevent incast congestion collapse since most of the timeouts are caused by full window losses due to Ethernet switch buffer overflow. The TCP incast scenario is common for data-center applications. we know that in a data center, traffic under the same ToR is actually a significant pattern known as work-seeks -bandwidth, as locality has been considered during job assignment. Considering that all traffic has to cross the ToR switches before reaching the destination server in the tree-like topology of a data-center network, we focus on the incast congestion on the last hop, i.e., output ports of ToR Ethernet switches as shown in.

II. ICTCP ALGORITHM

ICTCP provides a receive-window-based congestion control algorithm for TCP at the end-system. The receive windows of all low-RTT TCP connections are jointly adjusted to control throughput on incast congestion. ICTCP algorithm closely follows the design points made. It is described how to set the receiver window of a TCP connection.

A. Algorithm

1. Calculate available bandwidth. Increase when enough quota.

2. Divide window to two slots

3. In Ist sub-slot, receive window cannot be Increased but in second sub-slot, it could be increased up to the available bandwidth observed in Ist slot.

4. Measured throughput current application requirement over the TCP Connection-->max (observed Throughput, exponentially filtered Throughput)

5. Expected throughput: Throughput that is achieved if only constrained by receiver window --> max (measured throughput, receiver window/RTT**)**

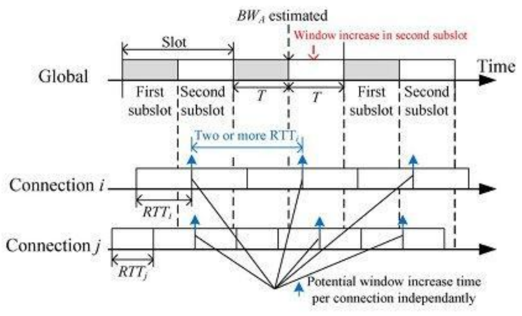

Fig.3 slotted time on global (all connection on that Interface) and arbitrary TCP connection i/j are independent

B. Control Trigger: Available Bandwidth

It is assumed there is one network interface on a receiver server, and define symbols corresponding to that interface. This algorithm can be applied to a scenario where the receiver has multiple interfaces, and the connections on each interface should perform this algorithm independently. Assume the link capacity of the interface on the receiver server is G. Define the bandwidth of the total incoming traffic observed on that interface as.

BW_T, which includes all types of packets, i.e., broadcast, multicast, unicast of UDP or TCP, etc. Then, define the available bandwidth on that bandwidth BW_A interface as

BW_A=max (0, α*C- BW_T)

Where α € |0, 1 is a parameter to absorb potential oversubscribed bandwidth during window adjustment. A larger α (closer to 1) indicates the need to more conservatively constrain the receive window and higher requirements for the switch buffer to avoid overflow; a lower α indicates the need to more aggressively constrain the receive window, but throughput could be unnecessarily throttled. A fixed setting of BW_A in ICTCP, an available bandwidth as the quota for all incoming connections to increase the receive window for higher throughput. Each flow should estimate the potential throughput increase before its receiving window is increased. Only when there is enough quota (BW_A) can the receive window be increased, and the corresponding quota is consumed to prevent bandwidth oversubscription.

*C. Per-Connection Control Interval: 2*RTT*

In ICTCP, each connection adjusts it's receive window only when an ACK is sending out on that connection. No additional pure TCP ACK packets are generated solely for receive window adjustment, so that no traffic is wasted. For a TCP connection, after an ACK is sent out, the data packet corresponding to that ACK arrives one RTT later. As a control system, the latency on the feedback loop is one RTT for each TCP connection, respectively. Meanwhile, to estimate the throughput of a TCP connection for a receive window adjustment; the shortest timescale is an RTT for that connection. Therefore, the control interval for a TCP connection is 2*RTT in ICTCP, and needed one RTT latency for the adjusted window to take effect and one additional RTT to measure the achieved throughput with the newly adjusted receive window.

D. Fairness Controller for Multiple Connections

When the receiver detects that the available bandwidth has become smaller than the threshold, ICTCP starts to decrease the receiver window of the selected connections to prevent congestion. Considering that multiple active TCP connections typically work on the same job at the same time in a data center, there is a method that can achieve fair sharing for all connections without sacrificing throughput. Note that ICTCP does not adjust the receive window for flows with an RTT larger than.

2ms, so fairness is only considered among low-latency flows.

III. ANALYSIS OF PROBLEM.

The root cause of TCP incast collapse is that the highly burst traffic of multiple TCP connections overflows the Ethernet switch buffer in a short period of time, causing intense packet loss and thus TCP retransmission and timeouts. Previous solutions focused on either reducing the wait time for packet loss recovery with faster retransmissions or controlling switch buffer occupation to avoid overflow by using ECN and modified TCP on both the sender and receiver sides. This paper focuses on avoiding packet loss before incast congestion, which is more appealing than recovery after loss. Of course, recovery schemes can be complementary to congestion avoidance. The smaller the change we make to the existing system, the better. To this end, a solution that modifies only the TCP receiver is preferred over solutions that require switch and router support (such as ECN) and modifications on both the TCP sender and receiver sides. Our idea is to perform incast congestion avoidance at the receiver side by preventing incast congestion. The receiver side is a natural choice since it knows the throughput of all TCP connections and the available bandwidth. The receiver side can adjust the receive window size of each TCP connection, so the aggregate burstiness of all the synchronized senders are kept under control. We call our design Incast congestion Control for TCP (ICTCP).

In previous versions the senders are sending many packets to the main server, but if the senders increase then the load on receiver side increases. As the window size on receiving side is less before, now we are increasing the window size so that it can accommodate many retransmission acknowledgements.

IV. WORKING MODULES

Here we have the ToR switch which is Data center Network.

1. ToR switch is working as a Data center Network.

2. ToR switch managed all the traffic.

3. All the data/Packets from number of Server is collected at ToR and then Transfers to client

4. ToR manages all the traffic from number of Servers and then transfer to client and avoids packet lost.

V. EXPERIMENTAL RESULT

After the implementation of the ICTCP algorithm in the following platform. We got the result that Packet losses are reduced to large extend and thus the bandwidth is preserved to large extend. The ICTCP method is also used for the prevention of congestion in network

VI. APPLICATIONS

1. Buffer overflow does not possible.

2. Timeout problem is less possible.

3. Load does not increased on receiver side.

4. Only received window is needed to be adjusted

5. Bandwidth is preserved to large extend as avoidance of retransmission.

VII. CONCLUSION

Transport Control Protocol (TCP) incast Congestion happens when large number of senders work in parallel with the same server where the high bandwidth and low latency network problem occurs. Incast congestion degrades the performance as the packets are lost at server side due to buffer overflow and hence the response time will be more. To improve the performance the focus is on the TCP throughput, RTT, receive window. In this ICTCP method is implemented that adjust the TCP receive window proactively active before packet loss occurs. Our aim is to avoid the wastage of bandwidth by adjusting its size as per the number of packets. To avoid the packet loss the ICTCP algorithm has been implemented. The algorithm has been implemented in the data center network.

REFERENCES

1. Haitao Wu, Zhenqian Feng Chuanxiong Guo, yongguang zhang “incast congestion control for TCP in Data Center Networks”
2. V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. Andersen, G. Ganger, G. Gibson, and B.Mueller, “Safe and effective fine-grained TCP retransmissions for datacenter communication,” in *Proc. ACM SIGCOMM*, 2009, pp. 303– 314.

3. S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “The nature of data center traffic: Measurements & analysis,” in *Proc. IMC*, 2009, pp. 202–208.

4. J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. OSDI*, 2004, p. 10.

5. M. Alizadeh, A. Greenberg, D.Maltz, J. Padhye, P. Patel, B.Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *Proc. SIGCOMM*, 2010, pp. 63–74.

6. D. Nagle, D. Serenyi, and A. Matthews, "The Panasas ActiveScale storage cluster: Delivering scalable high bandwidth storage," in *Proc. SC*, 2004, p. 53. 14.

7. E. Krevat, V. Vasudevan, A. Phanishayee, D. Andersen, G. Ganger, G. Gibson, and S. Seshan, "On application-level approaches to avoiding TCP throughput collapse in cluster-based storage systems," in *Proc. Supercomput.*,2007, pp.1–4.

8. C. Guo, H. Wu, K.Tan, L. Shi, Y.Zhang, and S. Lu, "DCell:Ascalable and fault tolerant network structure for data centers," in *Proc. ACM SIGCOMM*, 2008, pp. 75–86.

9. M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACMSIGCOMM*, 2008, pp. 63–74.

10. C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM*, 2009, pp. 63–74.

11. L. Brakmo and L. Peterson, "TCP Vegas: End to end congestion avoidance on a global internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

12. R. Braden, "Requirements for internet hosts—Communication layers," RFC1122, Oct. 1989.