Improving the Throughput of ICTCP in Data Center Network

Palash M. Gourshettiwar, Prof. V. T. Gaikwad

Abstract— Transport Control Protocol (TCP) incast congestion happens when number of senders work in parallel with the same receiver 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 report is focusing on the TCP throughput, RTT, receive window. Our method is to 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 which is ToR.

Index Terms— Data-center networks, incast congestion, TCP, ICTCP, Congestion, RTT.

1. INTRODUCTION

In this report our discussion is mainly on the congestion problem that usually occurs in the Incast and the Incast is condition that is opposite to that of that of the broadcast. In broadcasting one node sends out messages to the multiple nodes but in Incast multiple nodes send messages to the same node. Network congestion is the situation in which an increase in data transmissions results in a proportionately smaller increase, or even a reduction, in throughput. Congestion degrades the performance of the network. Packets loss is the major reason for Congestion. 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. Focus is on avoiding packet loss before incast congestion, which is more appealing than recovery after loss. 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. Our aim is to design Incast congestion Control for TCP (ICTCP). However, 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 nonincast cases.

The technical novelties of this work are as follows: 1] To perform congestion control on the receiver side, we use the available bandwidth on the network interface as a quota to coordinate the receive window increase of all incoming connections. 2] Our per-flow congestion control is performed independently of the slotted time of the round-trip time (RTT) of each connection, which is also the control latency in its feedback loop. 3] Our receive window adjustment is based on the ratio of the difference between the measured and expected throughput over the expected. The live RTT is necessary for throughput estimation as observed that TCP RTT in a high-bandwidth low-latency network increases with throughput, even if link capacity is not reached.TCP incast has been identified and described by [5] in distributed storage clusters. In distributed file systems, the files are deliberately stored in multiple servers. However, TCP incast congestion occurs when multiple blocks of a file are fetched from multiple servers at the same time. Several application-specific solutions have been proposed in the context of parallel file systems.

TCP Incast Congestion

In fig1.2, a typical data-center network structure is there. There are three layers of switches/routers: 1.The ToR switch. 2.The Aggregate switch, and 3.The Aggregate router. A detailed case for a ToR connected to dozens of servers.

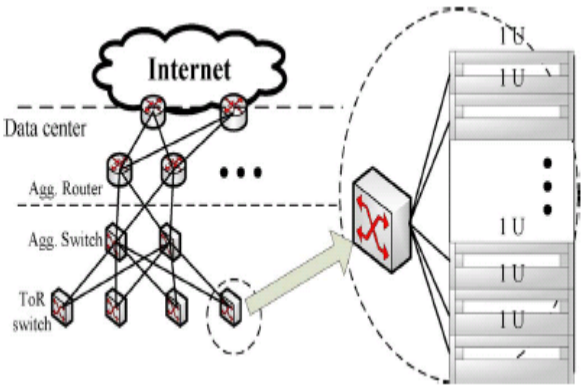

Fig. 1.2 Data-center networks and a detailed illustration of a ToR switch connected to multiple rack-mounted servers.

Incast congestion happens when multiple sending servers under the same ToR switch send data to one receiver server simultaneously, as shown in Fig. 1.3.

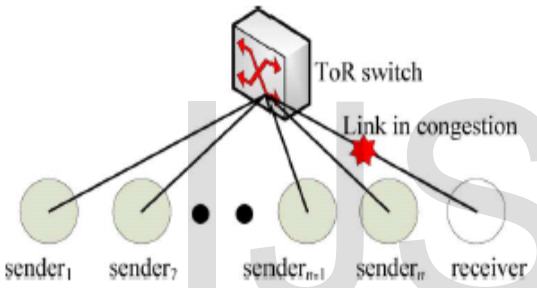

Fig. 1.3 Scenario of incast congestion in data-center networks, where multiple ( ) TCP senders transmit data to the same receiver under the same ToR switch.

1.3 Motivations

Incast is opposite to that of that of the broadcast. In broadcasting one node sends out messages to the multiple nodes. In Incast multiple nodes send messages to the same node. When number of server works parallel with single one then network congestion occurs. Network congestion is the situation in which an increase in data transmissions results in a proportionately smaller increase, or even a reduction, in throughput. Congestion degrades the performance of the network. Packets loss is the major reason for Congestion.

Packets are having Two numbers are associated with each packet

1. Sequence number-Indicates the ID of a packet.
2. Acknowledgment number-Indicates the expected ID of next packet, ack packets with same ack number indicates loss.

TCP Timeout-

1. Timeout happens when the sender does not receive ack for a long time period (retransmission or RTO).

2. When there is severe congestion in the network timeout happens.

- In distributed file systems, as the files are deliberately stored in multiple servers at the time of fetching congestion occur.
- With the recent progress in data-center networking, TCP incast problems in data-center networks have become a practical issue.
- 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 buffer overflow.

With the analyzing of the characteristics of the data center network, the communication pattern and the TCP congestion control algorithm, the reasons of the TCP Incast as follows:

- Since the top of rack switches are shallow buffered, highly busty, fast and simultaneous data transmissions overfill the switch buffer to make packet losses.
- Mass packet losses will lead to TCP congestion control, making the sending window half and reducing the sending rate.
- Intense packet loss results in TCP timeouts. The TCP timeouts last 100's milliseconds, but the round trip time of data center network is around 100's microsecond. Corse grained RTOs reduce the application throughput 90%.

2. Analysis of Problem

The root cause of TCP incast collapse is that the highly busty 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

Transport Control Protocol (TCP) incast congestion happens when number 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 report is focusing on the TCP throughput, RTT, receive window.

In distributed file systems, as the files are deliberately stored in multiple servers at the time of fetching congestion occur. With the recent progress in data-center networking, TCP incast problems in data-center networks have become a practical issue. 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 buffer overflow.

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.

5. Working Modules

Here we have the ToR Switch which is Data Center Network

- ToR Switch is acting as data centre Network.
- All traffic is managed at ToR Switch.
- All the data/Packets from number of servers is collected at ToR and then transfers to client.
- ToR manages all the traffic from number of Servers and then transfer to client and avoids packet lost.

6. Experimental Results and Discussion

After the implementation of the ICTCP algorithm in the following platform. The results are as follows-

- Simulation Duration of the packet on the server which is uniform to all.

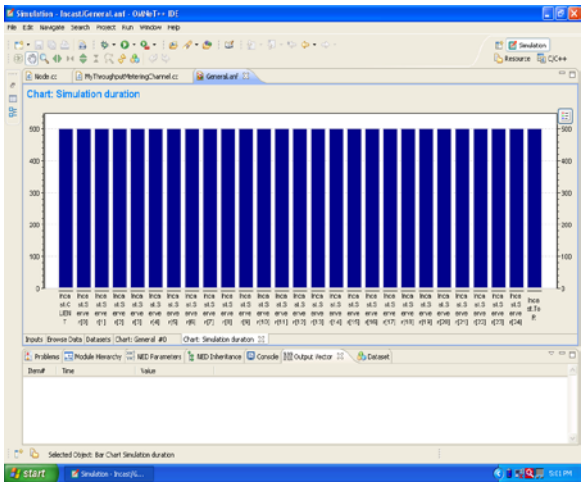

Fig.6.1 Simulation Duration

- Interarrival times: Count on the servers of the packets showing complete preservation of bandwidth.

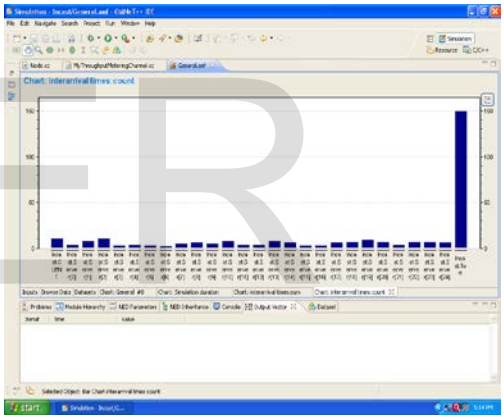

Fig. 6.3 Interarrival times: Count

7. Applications

Sr. No.	RETRANSMISSION Algorithm	ICTCP Algorithm
1	Buffer Overflow problem.	Buffer Overflow does not possible.
2	Timeout problem.	Timeout problem less possible.
3	Load increased on receiver side.	Load does not increased on receiver side
4	Both windows are needed to be adjusted.	Only receiver window is needed to be adjusted.
5	Wait time reduced by retransmission.	Not in ICTCP Algorithm.
6	Bandwidth is not preserved.	Bandwidth is preserved to large extend as avoidance of retransmission.

Table 7 Showing efficiency of ICTCP algorithm over retransmission algorithm

CONCLUSION

Transport Control Protocol (TCP) incast congestion happens when 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 which is Top of Rank (ToR).

From the above graph 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.

REFERENCES

[1] 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.

[2] 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.

[3] A. Phanishayee, E. Krevat, V. Vasudevan, D. Andersen, G. Ganger, G.Gibson, and S. Seshan, "Measurement and analysis of TCP throughput collapse in cluster-based storage systems," in *Proc. USENIX FAST*, 2008, Article no. 12.

[4] 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.

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

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

[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