PBAQ: Proactively Burst-aware Queue Level Load Balancing in DCN

Chengcheng Luo[†], Zhengyu Li[‡], Xiaoliang Wang[†], Sanglu Lu[†], Gaogang Xie[‡]

[†]National Key Laboratory for Novel Software Technology, Nanjing University
[‡]ICT-CAS
{luocc}@smail.nju.edu.cn,{waxili,sanglu}@nju.edu.cn,{zyli,xie}@ict.ac.cn

ABSTRACT

This poster presents PBAQ, a proactively burst-aware queue level load balancing mechanism in data-center networks. PBAQ is motivated by the facts that most of the load balancing mechanisms are mostly dependent on end-to-end (or tor-to-tor) path state feedback, and then select the best path based on the path states in the previous detection cycle. Nevertheless, E2E path detection is too expensive to ephemeral traffic change, and perhaps more importantly, it is unable to catch the real-time path utilization states, because of the utilization states changes that are induced by sub-RTT level traffic surges. PBAQ using emergency programmable switch to implement sketching based switch local surge detection, and combine it with next hop feedback to adjust shadow queue. Thus, it prevent heavily queue build up induced packet drop and keep a persistent queue length. Finally, PBAQ aims at eliminate tail latency which caused by burst induced tail drop.

CCS CONCEPTS

• **Networks** → Programmable networks;

KEYWORDS

proactive, load balancing, traffic surge, programmable switch

ACM Reference Format:
Chengcheng Luo[†], Zhengyu Li[‡], Xiaoliang Wang[†], Sanglu Lu[†], Gaogang Xie[‡]. 2018. PBAQ: Proactively Burst-aware Queue Level Load Balancing in DCN. In *Proceedings of APNet'18. BeiJing, China, August 02-03, 2018* , 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In practice, most DCNs (Data Center Networks) have been deployed with muti-rooted topology (fat-tree, leaf-spine), in order to guarantee full bisection bandwidth for a large number of end rack servers. While the muti-rooted topology offers plenty path diversities on path selection, the efficient user of link bandwidth is mainly decided by whether network traffic is evenly sprayed on each available path.

In production networks, ECMP has been the widely used data-plane load balancing algorithm. ECMP is a stateless path selection method that directly leverages 5-tuples flow hash digest as path selection key. Note that flow size distribution has a long-tail shape, and the large flow hash collision congestion and packet drop are both inevitable. In recent years, a large body of LB mechanisms focuses on how to leverage real-time network state for better data plane path selection and data traffic re-balance[1, 2, 5, 7–9].

A group of SDN based Load balancing mechanism, like Hedra, SWAN, B4 etc, which leverage centralized controller to periodically pull the data plane state for dynamically update path selection. Hedra[1] detect elephant flow in data plane, and only re-balance the elephant flow on special path that was prepared to large flow. SWAN[5] and B4[7] has common deploy circumstances—inter-DCN wan which has mostly stable traffic pattern compare to intra-DCN traffic pattern. Another kind of load balancing mechanism is significantly dependent on network state feedback in the data plane. Which is more responsive and robust, permitting it to make traffic re-balancing decision as low as 1RTT(100us). CONGA[2] is first congestion aware load balancing mechanism,which tor switch passively measure each tor-to-tor path state, and maintain state info on tor switch. However, production switch chip has limited memory resource, thus, which technique is only suit for small 2-layer clos topology. COLOVE[8] use edge vswitch overlay based active probe to detect path state which could be indicate by ECN or INT feedback, and it maintain server-to-server path state on vswitch. HULA[9] leverage emergency programmable switch chips to implement periodically active probe that send by each tor switch to every destination tor switch, each switch only maintain congestion state for next hop per destination.

In this work, we using emergency programmable switch and P4[3] to implement PBAQ—a fine grained adaptive re-routing capable Load Balancing mechanism. We have demonstrated that our design overview that only consuming small amount of on chip resource and will almost eliminate tail drop.

2 DESIGN OVERVIEW AND CHALLENGE

2.1 Design Challenge

subRTT level traffic surge(less than 50μs): According to newly measurement result, duration of traffic surge is usually as long as 50μs. However, tail drop are mainly caused by this subRTT level micro-burst that can't take by any reactively probe signal that consume at least RTT time. Timely detection and rapid response for local traffic sudden change is main obstacle to smooth burst traffic.

This work was done when C. Luo was a visiting PhD student at ICT-CAS..

APNet'18, August 02-03, 2018, BeiJing, China

ACM ISBN 978-1-4503-5244-4/18/08.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

large number of forwarding state and path state: Modern DCN have increasingly scale—which may be contained tens of thousands tor switch. Thus, if we use 3-bit to save per path congestion state in Fat-Tree topology with radix 10K, each tor switch will consume 600Mbit of local memory for congestion aware routing[9]. Besides,all the switches must save massive forwarding state to support tor-to-tor tunnel for each path it may be route packet over. Thus, there have limited on chip memory for traffic measurement.

fine grained near random traffic re-balance: Flow level load balancing is far from optimal, thus, flowlet is widely used in many research works. However, flowlet based routing that could not evenly spray continuous burst packet to each available path, will miss the last chance of traffic re-balancing. Much more fine grained LB need much more traffic state at smaller time granularity. but it is impractical to DCN which have tens of thousands switches. Thus, near random fine grained adaptive routing may be a practical solution.

2.2 Design Overview

PBAQ take advantage of switch local buffer queue length variance as implicit congestion signal, and combine next hop feedback for queue level traffic measurement. In order to make reasonable tradeoff among timeliness, deployability and resource consumption we have been made three important design choice. First, we mainly use switch local measurement to infer path state, rather than end-to-end measurement. Second, in order to timely re-balance traffic surge, we make path switching decision at per discontinuous flow fragment granularity which is more fine grained than per flowlet re-route and more agile than per packet reroute. Third, to make it easy to deploy on production network, we abstract a shadow queue, a queue sharing group with n queue that belongs to different switch port.

slide windows based streaming measurement model: Let's define stream in time windows which contain several sub-flow fragment, and each sub-flow fragment may have a few of Discontinuous flow fragment that each of this is a continuous sequence of packets belong to same flow. Furthermore, cumulatively, the primary component of traffic change in each windows was gathered by high-pass filtering.

queue level proactive traffic surge detection: Because of DC network have huge amounts of concurrent flow, to detect traffic at flow level would require considerable memory resource and stateful memory resource. Also, compared with large flow transmission, mice flow transmission was much more rapid and would be finished in only a few RTT periods. Besides, the power-law characteristics of flow size distribution for DC traffic indicate that flow level detection is trivial Which single mice flow have limited impact on queue build up and packet drop. However, if there have massive mice flow that aggregate to single output port instantaneously, it also will be induce serious queue build up and tail drop. Thus, by comparison, traffic surge detection at more coarser grain is non-trivial.

We use switch queue metadata that are available on PISA architecture switch chip such as RMT[4] and Intel FlexPipe[6] to track queue level traffic sudden change. For example, Tofino provide seven queuing related metadata(contain port queue id) which could programmed in ingress pipeline to determine the egress queue id. Our traffic detection framework is heavily depend Sketching algorithm. Generally, let's define cardinality estimate function $CE\colon \{identifier\_set\colon identifier \in \mathbb{H}\} \mapsto Cardinality$, and hot item estimation function which estimate to item that occurrence frequency is lager than a fixed proportion of entire packet stream in time windows $HE\colon \{identifier\_set\colon identifier \in \mathbb{H}\} \mapsto Hot\_item\_set$. thus,

$$n_{pq} = CE(egress\_port \cup egress\_queue\_id \cup arrival\_time) \quad (1)$$

$$heavy\_hitter\_queue = HE(egress\_port \cup egress\_queue\_id) \quad (2)$$

where n_{pq} is quantity of packet that received by queue q^{th} of port p. $heavy\_hitter\_queue$ is the queue that receive most packet at last measurement cycles.

shadow queue based traffic re-balance: In each control loop, most of existing work are doing obvious routing decision. Our main idea is that to leverage local traffic surge detection result and downstream feedback as traffic re-balance metric to make adaptive routing decision. Specifically, we divide network traffic into two type according to traffic patterns—up-link traffic and down-link traffic. Each pattern have significant difference impact on traffic control. Up-link side switch queue will be build up and tail drop since end host burst send mode or traffic aggregation on any up-link switch. Down-link switch queue build up reason is because there have a series of aggregation flow send to same output port. Thus, we schedule packet in certain DF on shadow queue which make up with several queue of each belong to different switch port. Switch local detection result and next-hop feedback determine shadow queue parameters.

REFERENCES

[1] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. 2010. Hedera: dynamic flow scheduling for data center networks.. In *Nsdi*, Vol. 10. 89–92.

[2] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Francis Matus, Rong Pan, Navindra Yadav, George Varghese, et al. 2014. CONGA: Distributed congestion-aware load balancing for datacenters. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 503–514.

[3] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. 2014. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 87–95.

[4] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. 2013. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. ACM, 99–110.

[5] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. 2013. Achieving high utilization with software-driven WAN. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. ACM, 15–26.

[6] Intel 2017. "Intel FlexPipe". http://tinyurl.com/nzbqtr3

[7] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. 2013. B4: Experience with a globally-deployed software defined WAN. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. ACM, 3–14.

[8] Naga Katta, Mukesh Hira, Aditi Ghag, Changhoon Kim, Isaac Keslassy, and Jennifer Rexford. 2016. CLOVE: How I learned to stop worrying about the core and love the edge. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. ACM, 155–161.

[9] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. 2016. Hula: Scalable load balancing using programmable data planes. In *Proceedings of the Symposium on SDN Research*. ACM, 10.