Computer Networks 91 (2015) 593–613

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Alleviating flow interference in data center networks through fine-grained switch queue management☆

Guo Chen, Youjian Zhao, Dan Pei*

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 21 October 2014
Revised 5 August 2015
Accepted 27 August 2015
Available online 9 September 2015

Keywords:
Data center networks
Switch queue management scheme
Flow interference

ABSTRACT

Modern data centers need to satisfy stringent low-latency for real-time interactive applications (*e.g.* search, web retail). However, short delay-sensitive flows generated from these applications often have to wait a long time for memory and link resource occupied by a few of long bandwidth-greedy flows because they share the same switch output queue (OQ). To address the above flow interference problem, this paper advocates more fine-grained flow separation in the switches than traditional OQ. We propose CQRD, a simple queue management scheme for data center switches, without change to the transport layer and requiring no coordination among switches. Through simulations, we show that CQRD can reduce the flow completion time (FCT) of short flows by more than 25% in a single switch and up to 50% in a multi-stage data center network, only at the cost of a minor goodput decrease of large flows. Additionally, just a 50% deployment of CQRD in top-of-rack (ToR) switches can lead to a ∼10–24% FCT reduction of short flows. Moreover, CQRD can improve short flows' FCT by ∼30–40% from OQ switches, using DCTCP (Alizadeh et al., 2010) [2] transport in DCN. Furthermore, we validate the feasibility of CQRD approach by implementing an 8 × 8 logical CQRD switch through simply changing the configurations of existing commodity switches. Also, we use a small testbed experiment to verify the implementation and the effectiveness of CQRD to alleviate flow interference in real environment.

1. Introduction

As people and business increasingly rely on the Internet in their daily life and work, the performance requirement on the data center networks (DCN), where most of the Internet applications are hosted, has become more stringent. However, recent studies have shown that short delay-sensitive flows from the real-time interactive applications (*e.g.* search, web retail), although contributing to majority of flows in DCNs [3], often have to wait a long time at switches for buffer and bandwidth resources occupied by a few of long bandwidth-greedy flows (*e.g.* backup, replication etc.). This causes a dramatic increase to the flow completion time (FCT) of most short flows, which can be more than 10 times higher [2].

As analyzed in many recent studies [2,4–6], the fundamental reason for the above mentioned performance degradation is that the commodity DCN switches' traditional and coarse (output queue, or OQ) queue management schemes are not suited well for the DCN traffic characteristics, causing

☆ This paper was previously presented in part [1] at LCN'14, Edmonton Canada, September 2014. Extensions to the conference version include detailed description to the CQRD approach and analysis to its impact on TCP performance. Also, we discuss how CQRD co-works with transport methods with adaptive rate control schemes. Moreover, a small-scale implementation and testbed experiments are added. Additionally, new experiments about incremental deployment, the impact of different buffer sizes, and transport methods with adaptive rate control schemes to CQRD's performance, are added in this paper.

* Corresponding author. Tel.: +86 (10) 62792837.
E-mail addresses: chen-g11@mails.tsinghua.edu.com (G. Chen), zhaoyoujian@tsinghua.edu.com (Y. Zhao), peidan@tsinghua.edu.com (D. Pei).

http://dx.doi.org/10.1016/j.comnet.2015.08.033
1389-1286/

unnecessary **flow interference**. We define that two flows **are interfered** by each other, when they pass through a switch while overlapping in time, and contend for some shared resources at switches, such as queue memory, or link capacity, *etc*. Many transport layer solutions [2,4,7,8] have been proposed to get around the coarse queue management problem by optimizing flows' rate assignment to keep the switch queues near empty. However, precise rate control is a great challenge due to the bursty traffic in DCN. Thus, only using these transport methods, flow interference can still happen in the coarse OQ due to flow burstiness. Therefore, a more fine-grained queue management scheme could be a good complement to these transport methods (more analysis in Section 4.3.3 and shown by experiments in Section 6.6). Moreover, all these transport approaches need to modify end host's protocol stack, which makes them facing some deployment difficulties. Another direction to solve the above performance degradation problem is flow scheduling [5,6]. These methods try to implement optimal flow scheduling to minimize the FCT of short flows. However, these solutions require significant changes on existing software or hardware of end hosts or switches, which are challenging to deploy. Furthermore, using these flow scheduling methods, small flows may still be interfered by long flows in original OQ switches, because of non-optimal scheduling [5] and long scheduling latency [6]. A fine-grained queue management scheme could also be complementary to these methods. More detailed discussion about related works is in Section 2.

Different from these previous approaches, we address the DCN flow interference problem by directly tackling its root cause: coarse switch queue management schemes. Hence, we argue that the DCN flow interference, especially for the interference between large number of small delay-sensitive flows and a small number of giant flows, calls for a more fine-grained queue management than the current output queue (OQ) in the commodity DCN switches in order to alleviate the flow interference problem in DCNs. Toward this direction, in this paper we propose a simple[1] queue management scheme, crosspoint-queue with random-drop (CQRD). In CQRD, a separate queue is assigned to each pair of input and output port, and packets are randomly dropped upon the full of crosspoint-queue (or randomly marked with ECN [9] tag upon the queue length above the threshold). This paper presents the design, analysis, implementation and evaluation of CQRD, to alleviate flow interference in DCN. Our contributions can be summarized as follows:

- We revisit the mature crosspoint-queue and random-drop techniques, and combine them together into a simple fine-grained queue management scheme named CQRD, to solve flow interference problem in DCN. These two underlying techniques are widely used in current switching hardwares [10,11], which makes it simple to implement CQRD. Moreover, the proposed approach requires neither any coordination among switches, nor modification to end hosts, which makes it easy to deploy.
- For DCN environment which uses adaptive transport rate control based on ECN [9] (*e.g.* DCTCP [2]), we accordingly design CQRD with a random-mark scheme (see Section 4.3.3) besides random-drop for traditional TCP environment. A hybrid of CQRD and transport layer methods achieves even better performance.
- We implement an 8×8 logical CQRD switch through simply modifying the configurations of existing commodity switches, which validates the simplicity and feasibility of CQRD's approach.
- Through simulations, we show that CQRD significantly reduces the flow completion time of short flows by more than 25% in a single switch and up to ~50% in a multi-stage data center network, only potentially at the cost of a minor goodput decrease for large flows. Furthermore, CQRD can be incrementally deployed. Just a 50% deployment of CQRD in ToR switches leads to a ~10–24% FCT reduction of short flows. Moreover, we show that CQRD can further improve short flows' FCT by ~30–40% from OQ switches, using DCTCP transport in DCN. Also, we conduct a small testbed experiment to verify the CQRD implementation and the effectiveness of CQRD to alleviate flow interference in real environment.

The rest of the paper is organized as follows. We review the related works in Section 2. In Section 3, we use analysis and simulation to study flow interference and its performance impact in traditional OQ, HCF (state-of-the-art switch queue management for DCN) [12], and classic CQ [10]. In Section 4, we present the CQRD approach and theoretically analyze how it can alleviate flow interference in DCN. In Section 5, we introduce our implementation of a small scale CQRD switch and discuss the implementation of a large scale CQRD switch through application-specific integrated circuit chips (ASIC). In Section 6, we use simulation experiments to show that CQRD greatly improves the overall DCN performance, both at fully and partially deployment. Also, we study the impact of different buffer sizes to CQRD's performance. Additionally, we show that a hybrid of CQRD and transport layer methods could achieve even better performance. In Section 7, we evaluate CQRD in a small testbed. Finally we conclude in Section 8.

2. Related works

Long delay of short delay-sensitive flows due to flow interference is a well known problem in data center network. We describe several solutions below and illustrate the difference between CQRD and them.

2.1. Transport layer rate control

A major direction of prior work uses transport layer rate control to reduce flow completion time of short flows. DCTCP [2] and HULL [7] apply adaptive rate control schemes based on ECN [9] and packet pacing, to control the rate of giant flows. By keeping the queue size of switches near empty, they improve the overall FCT of short flows. D^2TCP [8] uses the deadline information for rate control, which allocates the rate of each flow according to their deadline information.

However, as also pointed out by their own authors [6], precise rate control is challenging due to the bursty traffic

[1] By "simplicity", in this paper, we mean that the CQRD design is easy to understand without much complexity, and the implementation is easy based on existing and mature underlying techniques.