2015 IEEE 35th International Conference on Distributed Computing Systems

EcoFlow: An Economical and Deadline-Driven Inter-Datacenter Video Flow Scheduling System

Yuhua Lin, Haiying Shen and Liuhua Chen
Department of Electrical and Computer Engineering
Clemson University, Clemson, South Carolina 29631
{yuhual, shenh, liuhuac}@clemson.edu

***Abstract*—As video streaming applications are deployed on the cloud, cloud providers are charged by ISPs for inter-datacenter transfers under the dominant percentile-based charging models. In order to minimize the payment costs, existing works aim to keep the traffic on each link under the charging volume (i.e., 95th percentile traffic volume from the beginning of a charging period up to current time). However, these methods cannot fully utilize each link's available bandwidth capacity, and may increase the charging volumes. To further reduce the bandwidth payment cost by fully utilizing link bandwidth, we propose an economical and deadline-driven video flow scheduling system, called EcoFlow. Considering different video flows have different transmission deadlines, EcoFlow transmits videos in the order of their deadline tightness and postpones the deliveries of later-deadline videos to later time slots so that the charging volume at current time interval will not increase. The flows that are expected to miss their deadlines are divided into subflows to be rerouted to other under-utilized links in order to meet their deadlines without increasing charging volumes. Experimental results on PlanetLab and EC2 show that compared to existing methods, EcoFlow achieves the least bandwidth costs for cloud providers.**

***Keywords*-Video streaming; Bandwidth cost; Inter-datacenter traffic; Percentile-based charging models**

I. Introduction

More and more streaming service providers (VSSPs), such as Netflix, begin to deploy their web applications on the cloud. Newly published videos and their replicas are allocated to distributed datacenters to serve users from different regions. When current video replicas are insufficient to provide a highly available and scalable streaming service, more video replications occur between different datacenters. Both server-to-customer video dissemination and video replication lead to a substantial amount of inter-datacenter traffic.

Cloud providers purchase transit bandwidth from ISPs based on certain pricing schemes, such as the 95th percentile charging model adopted by most ISPs [1]. In the 95th percentile charging model, the bandwidth cost is charged based on the 95th percentile value in all traffic volumes (data sizes) recorded in every 5-minute interval generated within a charging period (e.g., 1 month [1]). We refer the 95th percentile traffic volume from the beginning of the charging period up to current time as **charging volume**. Many previous studies focus on controlling the new traffic volume below the charging volume [1]–[4] in order to minimize the bandwidth payment cost on inter-datacenter video traffic to ISPs. We can mainly classify such previous studies into two groups: *store-and-forward* and *optimal routing path*.

The *store-and-forward* methods [2], [3] postpone the transmissions of the delay-tolerant data flow from peak hours to off-peak hours, so as to utilize the leftover traffic during off-peak hours. Such store-and-forward transfer systems predefine off-peak hours of each datacenter. Delaying the transmission of delay-tolerant videos from peak hours to off-peak hours is a coarse-grained scheduling strategy. It does not fully utilize the link's available bandwidth capacity when actual traffic load is light during peak hours. The optimal routing path [1], [4] identify routing paths for video flows with the objective of minimizing the bandwidth payment costs. However, these methods transmit each video immediately when the video transmission request arrives at the source datacenter regardless of their deadlines. Therefore, these methods can easily reach the charging volume of current link and create many reroute requests when a large number of video transfer requests arrive simultaneously, which many increase the charging volumes of some links.

II. Proposed method

To handle the problems in previous methods, we propose an economical and deadline-driven video flow scheduling system, called EcoFlow. It is based on the fact that different video flows have different deadlines. Different applications from cloud customers have different service-level agreements (SLAs) that specify data Get/Put bounded latency [5] or a service probability [6] by ensuring a certain number of replicas in different locations [7]. Thus, cloud providers would like to assign shorter transmission deadlines (deadline in short) to videos in applications with more stringent SLAs in order to minimize the SLA violation penalty to maximize their profits [4]. Different videos in one application also have different deadlines. For example, the flows for new video dissemination to a datacenter to serve user requests should have more stringent deadlines than the flows for video replication backups to boost availability. Based on the different deadlines of video flows, the basic idea of EcoFlow is to postpone the transfers of some delay-tolerant videos while still ensure their transmission within deadlines if the transmission of these videos will increase the current charging volume. The EcoFlow system includes three key steps.

Step 1: **available bandwidth capacity estimation.** We use T_p to denote the time window used to estimate the available bandwidth capacity on each link, and use T_r (T_r<T_p) to denote the time window to record traffic volume in current

1063-6927/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDCS.2015.84

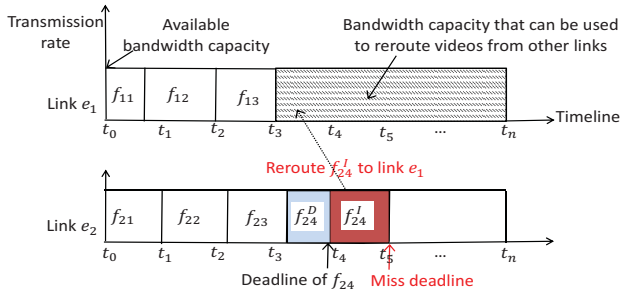

Fig. 1: An overview of EcoFlow.

charging model. Based on historical data, we estimate the total volume of video traffic needed to be transmitted on each link during time interval $[t_0, t_n)$, $t_n - t_0 = T_p$, denoted by $\tilde{v}(t_0, t_n)$. Assume link e_1's charging volume at time t_0 is $\hat{v}_1(t_0)$, it then can transfer a volume of $\hat{v}_1(t_0) \times T_p/T_r$ video during time interval $[t_0, t_n)$. We define **a link's available bandwidth capacity** as the maximum transmission rate that can be used to transfer videos without increasing the current charging volume during a certain time interval. We then calculate the available bandwidth capacity $\Delta c_1(t_0, t_n)$ on link e_1 during time interval $[t_0, t_n)$:

$$\Delta c_1(t_0, t_n) = \hat{v}_1(t_0)/T_r - \tilde{v}_1(t_0, t_n)/T_p. \quad (1)$$

Step 2: **deadline-driven flow scheduling.** On each link, the pending video flows are scheduled on an earliest-deadline-first base. When the traffic capacity is fully occupied at the current interval, we postpone the transfer of flows with later deadlines to later time interval but still guarantee their deliveries by deadlines). On link e_2, the transmission of flow f_{21} fully utilizes the available bandwidth capacity on link e_2 in time interval $[t_0, t_1)$, so f_{22} with a later deadline than f_{21} will be sent after f_{21} finishes transmission. However, when f_{24} is scheduled after f_{23}, its expected transmission time is at t_5, which is later than its deadline. We divide f_{24} into two subflows: f_{24}^D and f_{24}^I. On link e_1, all pending videos are scheduled to finish transmission before t_3, its available capacity during $[t_3, t_n)$ is not utilized (highlighted in dashed fill). We call the available bandwidth capacity that are not utilized during $[t_3, t_n)$ **extra bandwidth capacity** ($\delta c_1(t_3, t_n)$), $\delta c_1(t_3, t_n) = \Delta c_1(t_0, t_n)$. We define the links with extra bandwidth capacity during a time interval as the under-utilized links. The extra bandwidth capacity on link e_1 can be utilized to reroute subflow f_{24}^I from e_2 by its deadline.

Step 3: **routing path identification.** For the video rerouting, we aim to identify an alternating path that has extra bandwidth capacity to transmit the video by its deadline. To this end, we reply on the Dijkstra's algorithm [8] and propose a path identification method.

III. Preliminary Results

We conducted experiments on Amazon EC2 platform [9] to evaluate the performance of EcoFlow in comparison with other systems. We compare the performance of EcoFlow with three datacenter traffic scheduling strategies: 1) Direct transfer (denoted as Direct), which directly transfers video flows to the destination whenever the video transfer requests are initiated by the cloud provider without considering each link's charging volume; 2) JetWay [4] and 3) NetSticher [3]. We defined two types of videos: Standard Definition (SD) videos with sizes randomly selected in [500, 800] MB, and High Definition (HD) videos with sizes randomly selected in [2, 4] GB [4]. A datacenter transfers x and y videos per hour (including both SD and HD videos) to all other datacenters during its peak hours and off-peak hours, respectively, where x and y were randomly selected from [2, 5] and [0, 1], respectively. The transfer request of each video is initiated at a random time during the selected hours, and its deadline is chosen in [30, 120] minutes after the transfer request's initiated time. We assumed a video with maximum tolerable transfer time longer than 60 minutes to be a delay-tolerant. There are a total of 7 datacenters on EC2, the capacity and cost per traffic unit of each link are set according to the studies in [4].

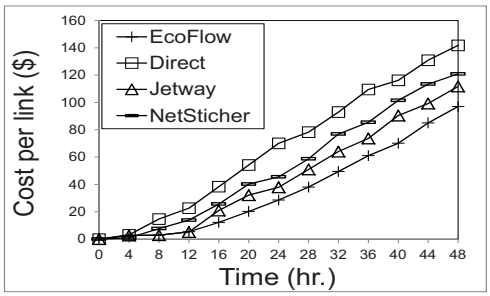

Fig. 2: Average bandwidth cost on EC2.

We calculated the bandwidth costs under the 95th percentile charging model, and defined a metric of *bandwidth cost per link* as the sum of bandwidth payment cost on all links divided by the total number of links in the network. Figure 2 shows the average cost per link at different time intervals for the 95th percentile charging models. We see that the per link cost follows: EcoFlow<JetWay<NetSticher<Direct.

Acknowledgements

This research was supported in part by U.S. NSF grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, and Microsoft Research Faculty Fellowship 8300751.

References

[1] D. Goldenberg, L. Qiu, H. Xie, Y. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. In *Proc. of SIGCOMM*, 2004.
[2] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram. Delay tolerant bulk data transfers on the internet. In *Proc. of SIGMETRICS*, 2009.
[3] L. Nikolaos, S. Michael, X. Yang, and R. Pablo. Inter-datacenter bulk transfers with netstitcher. In *Proc. of SIGCOMM*, 2011.
[4] Y. Feng, B. Li, and B. Li. Jetway: Minimizing costs on inter-datacenter video traffic. In *Proc. of Multimedia*, 2012.
[5] A. Hussam, P. Lonnie, and W. Hakim. Racs: A case for cloud storage diversity. In *Proc. of SoCC*, 2010.
[6] Service Level Agreements. http://azure.microsoft.com/en-us/support/legal/sla/, [Accessed in Sep. 2014].
[7] Amazon S3. http://aws.amazon.com/s3/, [Accessed in Sep. 2014].
[8] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.
[9] Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/, [Accessed in Sep. 2014].