Shaping Deadline Coflows to Accelerate Non-Deadline Coflows

Renhai Xu*, Wenxin Li†, Keqiu Li*, Xiaobo Zhou*
*Tianjin Key Laboratory of Advanced Networking, School of Computer Science and Technology, Tianjin University, China.
†School of Computer Science and Technology, Dalian University of Technology, China.
Xiaobo Zhou is the corresponding author: xiaobo.zhou@tju.edu.cn.

***Abstract*—Data-parallel applications generate a mix of coflows with and without deadlines. Deadline coflows are mission-critical and must be completed within deadlines, while non-deadline coflows desire to be completed as soon as possible. Scheduling such mix-coflows is an important problem in modern datacenters. However, existing solutions only focus on one of the two types of coflows: they either solely focus on meeting the deadlines of deadline-aware coflows or reducing the coflow completion times (CCTs) of non-deadline coflows. In this paper, we study the problem of optimizing deadline and non-deadline coflows simultaneously. To this end, we present a new optimization framework, *mixCoflow*, to schedule deadline coflows with the objective of minimizing and balancing their bandwidth footprint, such that non-deadline coflows can be scheduled as early as possible. Specifically, we develop the mathematical model and formulate the scheduling problem for deadline coflows as a lexicographical min-max integer linear programming (ILP) problem. Through rigorous theoretical analysis, this ILP problem has been proved to be equivalent to a linear programming (LP) problem that can be solved with standard LP solvers. By solving this LP, *mixCoflow* is able to balance the bandwidth footprint of deadline coflows while guaranteeing their deadlines. As a result, non-deadline coflows can be scheduled as soon as possible whenever they arrive. To demonstrate the effectiveness of our work, we have conducted extensive simulations based on a widely used Facebook data trace. The simulation results verify that *mixCoflow* can achieve significant improvement on the average CCT of non-deadline coflows, at no expense of increasing the deadline miss rates of deadline coflows, when compared to the state-of-art solutions.**

I. Introduction

It is routine for data-parallel applications (e.g., web search queries and MapReduce-like jobs) to run on datacenters to deal with the exponential growth of data [1–6]. A common feature of these applications is that they generate a set of parallel flows to transfer the intermediate data between successive computation stages—known as *cowflows* [7]. A succeeding computation stage cannot start until all its required inputs are in place, leading to an *all-of-nothing* feature for a coflow: all flows must be completed before a coflow is considered to be completed [6, 8].

In modern datacenters, coflows can be roughly divided into two categories: *deadline coflows* and *non-deadline coflows*. *Deadline coflows* are often generated by user-facing applications (e.g., web search, social network, advertisement systems), which have stringent latency requirements [7, 9]. Such deadline coflows are useful to the users if, and only if, they are completed within their deadlines. Otherwise, user experience will be hurt. This will in turn waste network bandwidth and incur revenue loss for the datacenter provider. On the other hand, *non-deadline coflows*, typically generated by cluster computing applications and data backups, have different performance requirements [6–8, 10]. More specifically, they impose no specific deadlines but generally desire to be completed as quickly as possible. When the two types of coflows coexist in a datacenter, an important problem is: how to schedule such a mix of coflows, with guaranteeing deadlines for deadline coflows and reducing completion times (CCTs) for non-deadline coflows.

While recognizing the importance of such mix-coflow scheduling problem, no existing solutions [6, 8, 10–14] are in place to optimize the deadline and non-deadline coflows simultaneously. The crux is that most of them only focus on optimizing one category of the coflows, which may hurt the performance of the other category of coflows. In other words, purely minimizing the CCTs of non-deadline coflows will cause high rates of deadline misses for the deadline coflows. Meanwhile, purely meeting deadlines of deadline coflows can arbitrarily prolong the CCTs of non-deadline coflows. It is worth noting that Varys [15] is perhaps the most related recent work which takes both deadline and non-deadline coflows into account. It separately designs two set of strategies, i.e., SEBF (Smallest-Effective-Bottleneck-First) and MADD (Minimum-Allocation-for-Desired-Duration), to minimize the average CCT and the number of late coflow. However, Varys essentially considers the two types of coflows independently rather than jointly, even though Varys strategies are applicable to both deadline/non-deadline coflows.

Bearing the above points in mind, one may wonder at this point that why not using Varys strategies to schedule the deadline and non-deadline coflows simultaneously. For example, one can schedule deadline coflows first with SEBF+MADD, and then similarly use SEBF+MADD to schedule non-deadline coflows with the residual network bandwidth. However, such trivially combination is problematic and can hurt the CCTs of non-deadline coflows while incurring minor or no improvement for the deadline miss rate. The main reason is that Varys strategies are unaware of the bandwidth footprint of deadline coflows, resulting in *heterogeneous* residual bandwidth in different time and links. In such a case, non-deadline coflows cannot fully use the residual bandwidth under the Varys strategies, and thus their CCTs will be increased.

In this paper, we study the mix-coflow scheduling problem, with the objective of meeting the deadlines of deadline coflows as well as reducing the CCTs of non-deadline coflows. To this end, we present *mixCoflow*, a new optimization framework that schedules the deadline coflows with *minimally* impact to the non-deadline coflows. More specifically, rather than directly considering one of the two types of coflows only, *mixCoflow* schedules deadline coflows first since they have higher priorities. But the thing is that when scheduling deadline coflows, *mixCoflow* attempts to minimize the impact on the non-deadline coflows by *minimizing the maximum bandwidth usage of deadline coflows across all time slots and all links*. We develop the mathematical model and formulate the deadline coflow scheduling problem as a lexicographical min-max integer linear programming (ILP) problem. Such a scheduling problem is inherently challenging to be solved, since it is NP-hard in general [16]. To tackle this challenge, we take an in-depth investigation of the structure of the ILP problem and surprisingly observe that the original ILP problem meets the following two conditions: *1)* a separable convex objective function and *2)* a totally unimodular constraint matrix. These conditions guarantee that the original ILP problem can be transformed into a linear programming (LP) problem, by applying the λ-technique [17] and linear relaxation. It has been proved that the transformed LP problem can be guaranteed to have the same solution to the original ILP problem. Moreover, the transformed LP can be efficiently and quickly solved with standard LP solvers. After the deadline coflows are scheduled, the remaining bandwidth can be allocated to the non-deadline coflows by using any existing methods such as FIFO and SEBF+MADD.

The reminder of this paper is organized as follows. In Section II, we present the mathematical model and problem formulation. In Section III, we show the design details of our *mixCoflow*. The extensive simulations are shown in Section IV. Finally, we discuss the related work in Section V and conclude this paper in Section VI.

II. Modeling and Problem Formulation

In this section, we develop a mathematical model to study the problem of minimizing and balancing the bandwidth footprint of deadline coflows, so as to minimize the impact on non-deadline coflows and thus accelerate non-deadline coflows.

A. Mathematical model

In our analysis, we abstract the datacenter network as a non-blocking switch interconnecting all the machines. In other words, coflow scheduling and bandwidth competition only takes place at the ingress/egress ports of this non-blocking switch, which corresponds to the incoming/outgoing links at each machine. Such a network abstraction is reasonable and has been widely used in many recent studies [10, 15].

In our mathematical model, we consider there are a set of machines in a datacenter network, which is denoted by $\mathcal{N} = \{1, 2, ..., N\}$. We consider a discrete time system, and consider that there are a set of time slots, denoted as $\mathcal{T} = \{1, 2, ..., T\}$. At each time slot $t \in \mathcal{T}$, each machine is capable of transmitting C units of data through its outgoing link, and receiving C units of data through its incoming link.

Since our objective is to minimize the impact to the non-deadline coflows when scheduling deadline coflows, we wish to always have free bandwidth after deadline coflows have been scheduled. So, in our mathematical model, we mainly consider the deadline coflows, and the non-deadline coflows can be scheduled later with existing methods. To indicate the deadline coflows, we denote $\mathcal{K} = \{1, 2, ..., K\}$ as the set of deadline coflows. Let $f_{i,j}^k$ denote a flow of deadline coflow $k \in \mathcal{K}$ that needs to transfer $D_{i,j}^k$ units data from machine i to j. For each deadline coflow k, we denote s_k and d_k as its start time and deadline, respectively. Similar to existing studies Varys [15], we assume that all flows in a coflow start at the same time and the information about all the flows can be known once the coflow has arrived at the network.

Decision variable: To indicate the coflow scheduling decision variable, we denote $x_{i,j}^{k,t}$ as the number of bandwidth units allocated to flow $f_{i,j}^k$ at time slot t. For simplicity, we assume that each unit of bandwidth is 1, and thus decision variable $x_{i,j}^{k,t}$ is integer:

$$x_{i,j}^{k,t} \in \mathbb{Z}^+,\ \forall i, j \in \mathcal{N},\ \forall k \in \mathcal{K},\ \forall t \in \mathcal{T} \tag{1}$$

Specifically, $x_{i,j}^{k,t} = 0$ means that the flow $f_{i,j}^k$ does not exist or this flow is waiting for transmission.

Link capacity constraints: When scheduling coflows, both the outgoing and incoming link capacities should be satisfied. Thus, we have the following two constraints:

$$\sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{N}} x_{i,j}^{k,t} \leq C,\ \forall i \in \mathcal{N},\ \forall t \in \mathcal{T} \tag{2}$$

$$\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}} x_{i,j}^{k,t} \leq C,\ \forall j \in \mathcal{N},\ \forall t \in \mathcal{T} \tag{3}$$

Constraint (2) means that the total amount bandwidth allocated to all the flows on each outgoing links should be no more than the capacity of this link in any time slot. Similarly, for each incoming link, the summation of bandwidth allocated to all the flows in any time slot must not exceed the corresponding link capacity, as shown in constraint (3).

Deadline constraints: To meet the deadlines of deadline coflows, each flow in a coflow should be completed within the deadline. In our model, we consider that each flow can only be transferred within the deadline, because transmitting a flow after its deadline is unnecessary. Hence, we have the following two constraints:

$$\sum_{t=s_k}^{d_k} x_{i,j}^{k,t} = D_{i,j}^k,\ \forall i, j \in \mathcal{N},\ \forall k \in \mathcal{K} \tag{4}$$

$$x_{i,j}^{k,t} = 0,\ \forall i, j \in \mathcal{N},\ \forall k \in \mathcal{K},\ \forall t \in \mathcal{T} \setminus [s_k, d_k] \tag{5}$$

Here, the term $\sum_{t=s_k}^{d_k} x_{i,j}^{k,t}$ calculates the total amount of data that flow $f_{i,i}^k$ transmitted in the duration of $[s_k, d_k]$. Thus, constraint (4) means that each flow in a coflow should be fully transmitted within its deadline. Constraint (5) is used for eliminating the potential cases where a flow is still transmitting after its deadline.

B. Problem formulation

To formally formulate our problem of minimizing the maximum bandwidth usage incurred by deadline coflows across all time slots and all links, we define $\mathcal{Z}$ as the maximum bandwidth usage across all links and all time slots, which can be expressed as follows:

$$\mathcal{Z} \quad = \max_{i,j\in\mathcal{N},t\in\mathcal{T}} \sum_{k\in\mathcal{K}} x_{i,j}^{k,t} \tag{6}$$

With the above definition, we are now ready to formulate our optimal problem **P1** as follows:

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{Minimize}} \quad \mathcal{Z} \\ &\text{Subject to: Eqs. (1),(2),(3),(4),(5).} \end{aligned} \tag{7}$$

where the objective function (7) is to minimize the maximum bandwidth usage among all links in all time slots, which means that each link has nearly balanced bandwidth usage in each time slot. When the bandwidth usage can be well balanced, non-deadline coflows can have more potential to be accelerated.

We can easily check that this problem is an integer linear programming (ILP) problem, which has unique challenge that makes it difficult to solve this problem because that this problem is NP-hard in general [16]. However, we make a surprising observation that this ILP can be transformed into an equivalent linear programming (LP) problem which returns exactly the same optimal solution to the ILP, as we will show in the following section.

III. The Design of *mixCoflow*

In this section, we present the design of our *mixCoflow*. We start by showing how to transform the ILP **P1** into an equivalent LP problem. Generally, an integer programming problem can be transformed into a linear programming problem if the integer programming problem has a *separable convex objective* and *totally unimodular linear constraints*. After taking an in-depth of the structure of **P1**, we find that **P1** exactly has such property.

A. Separable convex objective

With the definitions of *lexicographically smaller* $\preceq$ and *lexicographical minimization* [18], we show that the optimal solution of problem **P1** can be obtained by solving the following *lexicographically minimization* problem **P2**:

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{lexmin}} \quad \boldsymbol{\xi} = (\xi_{1,1}^1, ..., \xi_{N,N}^1, ..., \xi_{N,N}^T) \\ &\text{Subject to: Eqs. } (1), (2), (3), (4), (5), \end{aligned} \tag{8}$$

where $\xi_{i,j}^t = \sum_{k\in\mathcal{K}} x_{i,j}^{k,t}$, $\forall i,j \in \mathcal{N}$, $\forall t \in \mathcal{T}$, and $\boldsymbol{\xi}$ is a vector with the dimension of $M = |\boldsymbol{\xi}| = TNN$. For this problem, the objective is to minimize the element in $\boldsymbol{\xi}$ which is the maximum bandwidth usage across all links and all time slots. Therefore, the optimal solution $\boldsymbol{x}^*$ that gives $\boldsymbol{\xi}^*$ is also the optimal solution for Problem **P1**, i.e., denoted as **P2** $\Rightarrow$ **P1**. To solve problem **P2**, Let $g(\boldsymbol{\xi})$ denote a function of $\boldsymbol{\xi}$:

$$g(\boldsymbol{\xi}) = \sum_{m=1}^{M} M^{\xi_m} = \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} M^{\xi_{i,j}^t} \tag{9}$$

where ξ_m is the m-th element of the vector $\boldsymbol{\xi}$. We can easily check that $g(\boldsymbol{\xi})$ is a summation of convex functions M^{ξ_m}, and accordingly $g(\boldsymbol{\xi})$ is also a convex function.

***Theorem** 1:* $g(\cdot)$ preserves the order of lexicographically no greater $\preceq$, *i.e.*, $\boldsymbol{\xi}^* \preceq \boldsymbol{\xi} \iff g(\boldsymbol{\xi}^*) \le g(\boldsymbol{\xi})$.

Proof: Here we first prove $\boldsymbol{\xi}^* \preceq \boldsymbol{\xi} \Longrightarrow g(\boldsymbol{\xi}^*) \le g(\boldsymbol{\xi})$. we assume $r(r \ge 1)$ is the index of the first non-zero element in $\boldsymbol{\xi}^* - \boldsymbol{\xi}$, which means that $\xi_i^* = \xi_i, \forall i < r, \xi_r^* < \xi_r$. Then, we have:

$$\begin{aligned} g(\boldsymbol{\xi}^*) - g(\boldsymbol{\xi}) &= \sum_{m=1}^{M} M^{\xi_m^*} - \sum_{m=1}^{M} M^{\xi_m} \\ &= M^{\xi_r^*} + \sum_{m=r+1}^{M} M^{\xi_m^*} - M^{\xi_r} - \sum_{m=r+1}^{M} M^{\xi_m} \\ &\le (M-r+1)M^{\xi_r^*} - M^{\xi_r} \\ &\le M^{\xi_r^*+1} - M^{\xi_r} \le 0 \end{aligned} \tag{10}$$

With the above inequalities, we get $\boldsymbol{\xi}^* \preceq \boldsymbol{\xi} \Longrightarrow g(\boldsymbol{\xi}^*) \le g(\boldsymbol{\xi})$. Now, we focus on the proof of $g(\boldsymbol{\xi}^*) \le g(\boldsymbol{\xi}) \Longrightarrow \boldsymbol{\xi}^* \preceq \boldsymbol{\xi}$ through its contrapositive: $\neg(\boldsymbol{\xi}^* \preceq \boldsymbol{\xi}) \Longrightarrow g(\boldsymbol{\xi}^*) > g(\boldsymbol{\xi})$, here $\neg(\boldsymbol{\xi}^* \preceq \boldsymbol{\xi}) \iff \boldsymbol{\xi} \prec \boldsymbol{\xi}^*$, therefore, we should prove $\boldsymbol{\xi} \prec \boldsymbol{\xi}^* \Longrightarrow g(\boldsymbol{\xi}) < g(\boldsymbol{\xi}^*)$, it can be easily proved by (10). Hereto, theorem is proved. ■

Based on the above Theorem 1, we now formulate the following problem **P3** that is equivalent to the problem **P2**, in terms of the optimal solution:

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{Minimize}} \quad g(\boldsymbol{\xi}) \\ &\text{Subject to: Eqs. } (1), (2), (3), (4), (5). \end{aligned} \tag{11}$$

Since $\boldsymbol{\xi}^*$ is lexicographically minimization, thus $\boldsymbol{\xi}^* \preceq \boldsymbol{\xi}$, $\forall \boldsymbol{\xi}$. Thanks to $\boldsymbol{\xi}^* \preceq \boldsymbol{\xi} \iff g(\boldsymbol{\xi}^*) \le g(\boldsymbol{\xi})$, we can get the minimum value of $g(\boldsymbol{\xi})$ is $g(\boldsymbol{\xi}^*)$. Therefore, problem **P3** has the same optimal solution as problem **P2**, denoted as **P3** $=$ **P2**.

B. Totally unimodular constraint matrix

In addition to the separable convex objective, the totally unimodular constraint matrix is an important factor that enforces a LP problem to have integral solutions. More precisely, denoting the feasible region of a LP problem as $\{x|A\boldsymbol{x} = b\}$ or $\{x|A\boldsymbol{x} \le b\}$, if the constraint matrix A is totally unimodular and b is integral, then such feasible region is an integral polyhedron and it only has integral extreme points. Typically, an m-by-n matrix is totally unimodular coefficient matrix, if it satisfies the following two conditions:

1. All elements of this matrix are in the range of $\{-1, 0, 1\}$;
2. For any subset $R \subset \{1, 2, ..., m\}$, it can be divided into two disjoint sets—R_1 and R_2, such that $|\sum_{i\in R_1} a_{ij} - \sum_{i\in R_2} a_{ij}| \le 1$, $\forall j \in \{1, 2, ..., n\}$.

The following theorem verifies that the coefficient matrixes for all constraints in the original ILP problem exactly form a totally unimodular matrix.

***Theorem** 2:* The coefficient matrix of linear constraints (2), (3), (4) and (5) form a totally unimodular matrix.

Proof: We observe that both the constraints (2) and (3) have TN inequations, while the constraint (4) has KNN equations and the constraint (5) has $NN\sum_{k=1}^{K}(S_k - 1 + T - D_k)$ equations. Denote $A_{m\times n}$ as the coefficient of all of the inequations and equations, where $m = 2TN + KNN + NN\sum_{k=1}^{K}(S_k - 1 + T - D_k)$, and $n = KNNT$. It should be noted that n is the dimension of the variable $\boldsymbol{x}$. We can easily check that any element of $A_{m\times n}$ is either 0 or 1, which means that the condition 1 can be satisfied. For any subset $R \subset \{1, 2, ..., m\}$, we can select all the elements that belong to $\{1, 2, ..., 2TN\}$ to compose the set R_1, and let the rest of the elements compose the set R_2. It is easy to check that the summation of all rows of R_1, is a $1 \times n$ vector with all elements equal to 2. Similarly, the summation of all rows of R_2, is a $1 \times n$ vector with elements equal to 1. Hence, we have $\sum_{i\in R_1} a_{i,j} = 2$ and $\sum_{i\in R_2} a_{i,j} = 1$, Eventually, $|\sum_{i\in R_1} a_{ij} - \sum_{i\in R_2} a_{ij}| \leq 1,\ \forall j \in \{1, 2, ..., n\}$, implying that the condition 2 is satisfied.

In summary, we have proved that both conditions for total unimodularity are satisfied, thus the theorem is proved. ■

C. Transform to LP problem

Now, we transform problem **P3** to a LP problem. Because **P3** is a convex problem and its coefficient matrix of linear constraints (2), (3), (4) and (5) form a totally unimodular matrix, with λ-representation technique [17], coincidentally, we can transform problem **P3** to a LP problem **P4** as follows:

$$\begin{aligned}
\min_{\boldsymbol{\lambda},\boldsymbol{x}} \quad & \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{s\in\mathcal{S}} M^s \lambda_{i,j}^{t,s} \qquad (12)\\
\text{s.t.} \quad & \sum_{s\in\mathcal{S}} \lambda_{i,j}^{t,s} = 1,\ \forall i,j \in \mathcal{N},\ \forall t \in \mathcal{T},\ \mathcal{S} = [0, C] \cap \mathbb{Z}\\
& \sum_{s\in\mathcal{S}} s\lambda_{i,j}^{t,s} = \xi_{i,j}^{t} = \sum_{k\in\mathcal{K}} x_{i,j}^{k,t},\ \forall i,j \in \mathcal{N},\ \forall t \in \mathcal{T}\\
& \lambda_{i,j}^{t,s}, x_{i,j}^{t,s} \in \mathbb{R}^+,\ \forall i,j \in \mathcal{N},\ \forall t \in \mathcal{T},\ \forall s \in \mathcal{S}\\
& \text{Constraints (2), (3), (4), (5).}
\end{aligned}$$

where, problem **P4** and **P3** have the same optimal solution, denoted as **P4** = **P3**.

***Theorem** 3:* Problem **P4** has the same optimal solution with problem **P1**.

Proof: we can get **P4** = **P3** and **P3** = **P2** from equations (12) and (11), respectively. We also have **P2** $\Rightarrow$ **P1** due to equation (8). Hence, we have:

$$\mathbf{P4} = \mathbf{P3} = \mathbf{P2} \Rightarrow \mathbf{P1}, \qquad (13)$$

where **P4** $\Rightarrow$ **P1**, it means that the optimal assignment variables $\boldsymbol{x}^*$ that gives **P4** is also the optimal solution for Problem **P1**. Therefore, theorem is proved. ■

Given the above LP problem, *mixCoflow* can then schedule the deadline and non-deadline coflows with the following steps. First, whenever an existing deadline coflow completes or a new deadline coflow arrives, our *mixCoflow* will solve the LP problem. As such, the amount of bandwidth that should be allocated to all the deadline coflows over all time slots and all links can be obtained. Second, *mixCoflow* will allocate the remaining bandwidth resource to non-deadline coflows with any existing method, such as FIFO and SEBF+MADD.

IV. Performance Evaluation

In this section, we evaluate *mixCoflow* by large-scale simulations based on a real-world data trace collected from Facebook [19].

Comparing solutions: We compare the following schemes with *mixCoflow* in our simulations. It should be noted that each of the following scheme is only used to schedule the deadline coflows rather than non-deadline coflows. For completeness, after the deadline coflows are scheduled by each scheme, we use SEBF heuristic to schedule the non-deadline coflows.

- **FIFO (First-In-First-Out)**: schedules deadline coflows based on their arriving times [8]. This scheme aggressively takes all the bandwidth when scheduling each deadline coflow, which may seriously impact the CCTs of non-deadline coflows.
- **EDF (Earliest-Deadline-First)**: all the deadline coflows are scheduled in an ascending order of their deadlines [20]. This scheme strictly prioritizes deadline coflows and could complete a coflow far before its deadline, which is actually unnecessary and may increase the CCTs of non-deadline coflows.
- **Varys**: schedules deadline coflows with the Shortest-Effective-Bottlence-First (SEBF) first, and then leverages Minimum-Allocation-for-Desired-Duration (MADD) to allocate bandwidth for each flow in a deadline coflow [15]. This scheme uses the exactly right bandwidth resources to guarantee the deadlines of deadline coflows. However, it makes no attempt to balance footprint, and thus may impact the non-deadline coflows.

Performance metrics: For deadline coflows, the primary metric is deadline miss rate, which is the percentage of deadline coflows that miss their deadlines. For non-deadline coflows, we define the *factor of improvement* in the average CCT (coflow completion time) as the primary metric. More specifically, the factor of improvement of scheme 1 compared to scheme 2 can be calculated as

$$\text{Factor of Improvement} = \frac{CCT_2}{CCT_1} \qquad (14)$$

where CCT_1 and CCT_2 are the average CCTs achieved by scheme 1 and scheme 2, respectively.

Simulation setup: We simulate a datacenter network with 150 machines. The incoming/outgoing link of each machine is uniformly set to be 800Mbps, which is a common setting in production datacenter [15].

Data trace: We use the Hive/MapReduce trace provided by Facebook as the workload in our simulations, which is a widely adopted trace in the existing studie [15]. We can easily check that each coflow in the original trace only contains the whole data size that each reducer needs to fetch. Hence, we uniformly sample the size for each flow within it and accordingly obtain the information of all the flows.

We divide all the 526 coflows into two categories of coflows, i.e., deadline coflows and non-deadline coflows, by using a *ratio*. For example, a ratio of 3:1 means that 75% of the 526 coflows are deadline coflows and the rest are non-deadline coflows. Specifically, the deadline of each deadline coflow is set to be its minimum completion time in an empty network multiplied by $(1 + U(0, x))$, where $U(0, x)$ is a uniformly random number in the range $(0, x)$. Unless otherwise specified, $x = 1$. Such deadline settings are similar to the study [15].

Simulation results: For deadline coflows, we mainly show the results on the deadline miss rate. For non-deadline coflows, we mainly present the results of the factor of improvement on the average CCT. Detailed simulation results are shown as follows:

1) Deadline miss rate: As aforementioned, deadline coflows are mission-critical and are only useful to the applications when they are completed before their deadlines. By varying the percentage of deadline coflows from 10% to 100%, we show the deadline miss rates achieved by different methods in Fig. 1. It is easy to find that our *mixCoflow* incurs a lower deadline miss rate than the FIFO method, under all the settings of the percentages of deadline coflows. The root reason is that FIFO is unaware of the deadlines of coflows. On the other hand, we can further observe that *mixCoflow* incurs more or less deadline miss rate, compared to the EDF scheme. This is because that EDF aggressively takes all the bandwidth to complete deadline coflows. And that's why the CCTs of non-deadline coflows will be hurt after the deadline coflows are scheduled with EDF (we will show this point later). As for the Varys scheme, our *mixCoflow* can enjoy a little benefit on the deadline miss rate. The root reason is that Varys embraces a complicated admission control mechanism and heuristically allocates bandwidth to deadline coflows based on current remaining bandwidth resource, which can easily overlook the optimal allocation strategy.

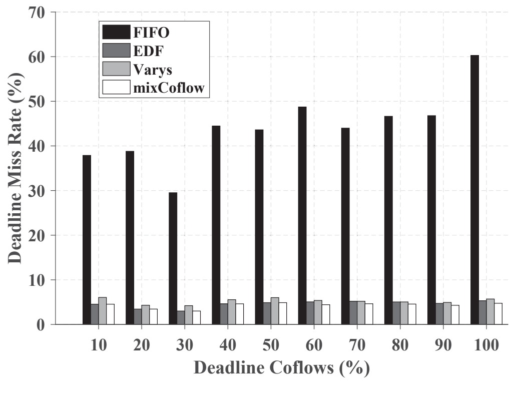

Fig. 1. The deadline miss rate under different percentages of deadline coflows.

2) Factor of Improvement: After deadline coflows are scheduled, the remaining bandwidth resource can be used for the non-deadline coflows. As aforementioned, the scheduling of deadline coflows will impact the CCTs of non-deadline coflows. We therefore use the factor of improvement in the average CCT of non-deadline coflows to show that our *mixCoflow* can efficiently reduce such impact. We show the factor of the improvement in the average CCT in Fig. 2, with varying percentages of deadline coflows. It is clear that our *mixCoflow* can reduce average CCT of non-deadline coflows, compared to all the other comparsion methods. Especially, when the percentage of deadline coflows is 80%, our *mixCoflow* can improve the average CCT by up to 18.18×, 20.24×, 9.35×, when compared to FIFO, EDF and Varys schemes, respectively. The reason for large improvement in the average CCT is that *mixCoflow* can minimize the maximum bandwidth usage caused by deadline coflows, over all time slots and all links. In such a case, the non-deadline coflows can be minimally impacted, and thus the average CCT can be significantly reduced.

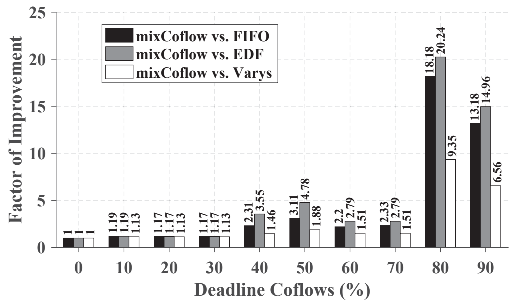

Fig. 2. The factor of improvement in the average CCT of non-deadline coflows.

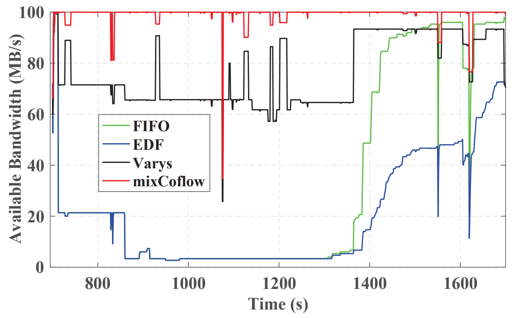

Fig. 3. The rest available bandwidth on all links and all time slots.

3) Remaining bandwidth: It is important to keep in mind that the key idea of this paper is to schedule deadline coflows with minimally impact on non-deadline coflows by balancing and minimizing the bandwidth footprint of deadline coflows over all time slots and all links. To completely understand this point, we record the remaining bandwidth on each link at each time slot when the deadline coflows have been scheduled. To ease the presentation, we mainly present the average remaining bandwidth across all links in a scenario where the percentage of deadline coflows is 75%, as shown in Fig. 3. From this figure, we can observe that *mixCoflow* has more remaining bandwidth than FIFO, EDF and Varys schemes at most of the time. Moreover, the remaining bandwidth incurred by *mixCoflow* is more balanced than the other comparison methods. This is why *mixCoflow* can achieve low average CCT of coflows.

V. Related Work

mixCoflow focuses on jointly scheduling deadline and non-deadline coflows in a datacenter, with the objective of meeting deadlines of deadline coflows while reducing the CCTs of non-deadline coflows. There is a large body of recent work that focuses on either guaranteeing deadlines for deadline coflows or reducing CCTs for non-deadline coflows. In this section, we only discuss some closely related ones.

Guaranteeing deadlines for deadline coflows: Existing work mainly focuses on decreasing coflow deadline miss rate [14, 15]. For example, Varys [15] first leverages an admission control mechanism to reject coflows whose minimum possible CCT exceeds their deadlines. Then, Varys separately design two set of strategies (i.e., SEBF and MADD) to schedule the admitted coflows. Taking one step further, Chronos [14] combines priority-based scheduling and limited multiplexing techniques to allocate bandwidth for coflows to just finish on time. However, these works are unaware of the bandwidth footprint of deadline coflows over time, and thus they will hurt the performance of non-deadline coflows.

Reducing CCTs for non-deadline coflows: There are also many works focusing on reducing CCTs of non-deadline coflows. The typical research works (e.g., [10, 13, 15]) mainly apply simple heuristics, such as FIFO, EDF, MRTF (Minimum-Remaining-Time-First), *x-Approximation* and SEBF, to schedule non-deadline coflows. While these works are efficient in reducing the CCTs of non-deadline coflows, they do not consider deadline coflow scheduling. The main difference between our *mixCoflow* and the above existing works lies in that *mixCoflow* jointly consider the deadline and non-deadline coflows, yet is able to reduce the impact on the non-deadline coflows when scheduling deadline coflows.

VI. Conclusions

In this paper, we present *mixCoflow*, a new coflow-aware optimization framework that jointly schedules a mix of coflows with and without deadlines. When scheduling deadline coflows, *mixCoflow* attempts to minimize and balance the bandwidth footprint across time slots and all links, so as to leave more bandwidth for non-deadline coflows and reduce the impact on the CCTs of non-deadline coflows. Specifically, in *mixCoflow*, we formulate a lexicographical min-max ILP problem for scheduling deadline coflows. This ILP is NP-hard in general. Fortunately, with several steps of non-trivial transformations, we prove that the optimal solution to this ILP can be obtained by solving an equivalent LP problem. In such a case, *mixCoflow* can schedule the deadline coflows by solving the relevant LP problem and leave the remaining bandwidth for non-deadline coflows which can be scheduled with any existing methods. Extensive trace-driven simulations demonstrate that our *mixCoflow* can significantly reduce the CCTs of non-deadline coflows without incurring increasing on the deadline miss rate for the deadline coflows, when compared to the prevailing solutions.

VII. Acknowledgment

This work is supported in part by the National Key Research and Development Program of China No. 2016YFB1000205, in part by the State Key Program of National Natural Science of China under Grant No. 61432002, in part by the Joint Funds of the National Natural Science Foundation of China under Grant No. U1701263, in part by the National Natural Science Foundation of China under Grant No. 61702365, 61672379 and 61772112, in part by the Natural Science Foundation of Tianjin under Grant No. 17JCQNJC00700 and 17JCYBJC15500, and also in part by the Special Program of Artificial Intelligence of Tianjin Municipal Science and Technology Commission under Grant No. 17ZXRGGX00150.

References

[1] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," in *Proc. of ACM SIGCOMM*, 2015.

[2] R. Viswanathan, G. Ananthanarayanan, and A. Akella, "Clarinet: Wan-aware optimization for analytics queries," in *Proc. of USENIX OSDI*, 2016.

[3] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[4] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS operating systems review*, vol. 41, no. 3. ACM, 2007, pp. 59–72.

[5] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. of Usenix HotCloud*, 2010.

[6] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *Proc. of ACM SIGCOMM*, Toronto, Canada, 2011.

[7] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proc. of ACM Workshop on Hot Topics in Networks*, 2012.

[8] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, "Decentralized task-aware scheduling for data center networks," in *Proc. of ACM SIGCOMM*, 2014.

[9] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *Proc. of ACM SIGCOMM*, 2011.

[10] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," in *Proc. of ACM SIGCOMM*, 2015.

[11] H. Zhang, L. Chen, B. Yi, K. Chen, M. Chowdhury, and Y. Geng, "Coda: Toward automatically identifying and scheduling coflows in the dark," in *Proc. of ACM SIGCOMM*, 2016.

[12] W. Wang, S. Ma, B. Li, and B. Li, "Coflex: Navigating the fairness-efficiency tradeoff for coflow scheduling," in *Proc. of IEEE INFOCOM*, 2017.

[13] L. Chen, W. Cui, B. Li, and B. Li, "Optimizing coflow completion times with utility max-min fairness," in *Proc. of IEEE INFOCOM*, 2016.

[14] S. Ma, J. Jiang, B. Li, and B. Li, "Chronos: Meeting coflow deadlines in data center networks," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[15] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in *Proc. of ACM SIGCOMM*, Chicago, IL, USA, 2014.

[16] J. K. Karlof, *Integer programming: theory and practice*. CRC Press, 2005.

[17] R. Meyer, "A class of nonlinear integer programs solvable by a single linear program," *SIAM Journal on Control and Optimization*, vol. 15, no. 6, pp. 935–946, 1977.

[18] L. Chen, S. Liu, B. Li, and B. Li, "Scheduling jobs across geo-distributed datacenters with max-min fairness," in *Proc. of IEEE INFOCOM*, 2017.

[19] "Facebook hive/mapreduce trace." [Online]. Available: https://github.com/coflow/coflow-benchmark

[20] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.