Received February 25, 2018, accepted April 17, 2018, date of publication May 1, 2018, date of current version May 24, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2832099

Optimizing the Cost-Performance Tradeoff for Coflows Across Geo-Distributed Datacenters

XINPING XU[1], WENXIN LI[1], KEQIU LI[1], (Senior Member, IEEE), HENG QI[1], AND YINGWEI JIN[2]
[1]School of Computer Science and Technology, Dalian University of Technology, Dalian 116023, China
[2]School of Management, Dalian University of Technology, Dalian 116023, China

Corresponding author: Heng Qi (hengqi@dlut.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB1000205, in part by the State Key Program of National Natural Science of China under Grant 61432002, in part by NSFC under Grant 61672379, Grant 61772112, Grant 61425002, and Grant 61751203, in part by the Dalian High-Level Talent Innovation Program under Grant 2015R049, and in part by the National Natural Science Foundation for Outstanding Excellent Young Scholars of China under Grant 61422214.

ABSTRACT For data analytics jobs running across geographically distributed datacenters, *coflows* have to go through the inter-datacenter network over relatively low bandwidth and high cost links. In this case, optimizing *cost-performance* tradeoffs for such coflows becomes crucial. Ideally, decreasing the coflow completion time (CCT) can significantly improve the network performance, meanwhile, reducing the transmission cost introduced by these coflows is another fundamental goal for datacenter operators. Unfortunately, minimizing both the CCT and the transmission cost are *conflicting* objectives that cannot be achieved concurrently. Prior methods have significant limitations when exploring such tradeoffs, because they either merely decrease the average CCT or reduce the transmission cost independently. In this paper, we focus on a cost-performance tradeoff problem for coflows running across the inter-datacenter network. Specifically, we formulate an optimization problem, so as to minimize a combination of both the average CCT and the average transmission cost. This problem is inherently hard to solve due to the unknown information of future coflows. Therefore, we present *Lever*, an online coflow-aware optimization framework, to balance these two *conflicting* objectives. Without any prior knowledge of future coflows, *Lever* has been proved to have a non-trivial competitive ratio in solving this cost-performance tradeoff problem. Results from large-scale simulations demonstrate that *Lever* can significantly reduce the average transmission cost, and at the same time, speed up the completion of these coflows, compared with the state-of-the-art solutions.

INDEX TERMS Scheduling algorithms, optimal scheduling, performance analysis.

I. INTRODUCTION

Data-parallel jobs are increasingly running across geographically distributed datacenters as well as between public clouds [1]–[6], with the purpose of processing large volumes of data that are generated and stored all over the world. A key feature of these jobs is that a collection of flows, termed *coflow* [7], will be generated to transfer the intermediate data between successive computation stages (e.g., map and reduce). A coflow will not finish only until all its flows have completed.

In the context of geo-distributed settings, all flows of a coflow necessarily have to be transferred across the inter-datacenter network (as shown in Fig. 1), resulting in a *cost-performance* tradeoff problem. On the one hand, the inter-datacenter bandwidth is an expensive and scarce resource that can cost up to hundreds of millions of dollars annually [8], [9]. On the other hand, data volumes of the coflow could be enormous for jobs running across geo-distributed datacenters [1], and speeding up the completion of the coflow has significant impact on the performance of the corresponding jobs [10], [11].

However, optimizing such *cost-performance* tradeoff for coflows running across geo-distributed datacenters is inherently a hard task. Simply minimizing the coflow completion time (CCT) could lead to high inter-datacenter transmission cost. Even worse, simply optimizing the inter-datacenter transmission cost can arbitrarily increase CCT of coflows. This is mainly due to the fundamental difference between these two metrics, and the high variability in available bandwidth [1], [12], [13] as well as the transmission cost per unit

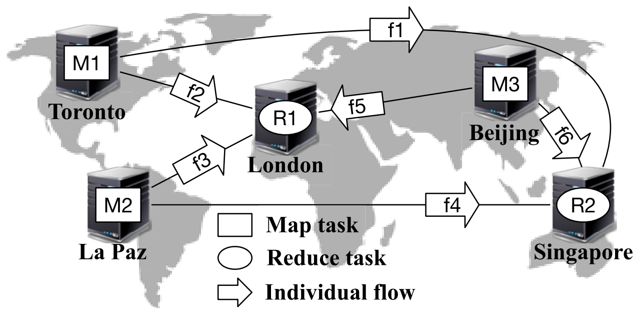

FIGURE 1. An illustrative example of a coflow consisting of 6 individual flows that necessarily have to traverse the inter-datacenter network.

data [14]–[16] of different inter-datacenter links. In this context, cost savings can typically be achieved by switching the traffic to low cost links, whereas CCT speedups are obtained by fully exploiting high-capacity links for data transmission.

To the best of our knowledge, no existing solutions are in place to solve such *cost-performance* tradeoff problem for coflows across geo-distributed datacenters. *First*, most existing work only considers optimizing CCT of coflows by efficiently scheduling (and routing) all flows within each coflow [10], [11], [17]–[22]. *Second*, although some existing methods investigate how to reduce the transmission cost introduced by inter-datacenter transfers [15], [16], [23], they do not take into account the flow dependency semantics and thus are all coflow-agnostic.

In this paper, we focus on the *cost-performance* tradeoff problem for coflows running across geo-distributed datacenters. Our primary focus is to *minimize the inter-datacenter transmission cost as well as CCT of coflows*. To this end, we blend the advantages of coflow routing and scheduling techniques to formulate an optimization problem, where the key objective is to minimize a combination of the average transmission cost and the average CCT across all coflows. This problem takes into account heterogeneous bandwidth capacities, diverse costs for transmitting one unit data on different links, and varying coflows arrival time. Unfortunately, due to the unknown information of future coflows, it is challenging to obtain an optimal solution for this optimization problem. To tackle this challenge, we present *Lever*, an online coflow-aware optimization framework. In *Lever*, we formulate a linear programming (LP) and handle it with the standard LP solver to derive routing and scheduling decisions for each coflow as soon as it arrives. Based on the optimal solution of the corresponding LP for each coflow, we then propose an efficient online algorithm to handle the multi-coflow scenario. The key idea of this online algorithm is to rescale the bandwidth allocation of each coflow based on a specific weight factor, while keeping the original routing decision of each coflow. Results from rigorous theoretical analysis prove that *Lever* can have a non-trivial competitive ratio when solving the original cost-performance tradeoff problem. To evaluate the performance of *Lever*, we conduct extensive simulations based on a real-world data trace provided by Facebook [18], [24]. Compared to the state-of-the-art solution RAPIER, *Lever* can reduce the average transmission cost by 49.72%. On the other hand, comparing with the Cost-only scheme, *Lever* can also decrease the average CCT of coflows.

In summary, the main contributions of this paper are as follows:

- We study a cost-performance tradeoff problem for coflows running across geo-distributed datacenters. Specifically, we formulate this problem as an optimization problem, so as to minimize a combination of the average CCT and the average transmission cost.
- We present a new coflow-aware optimization framework, *Lever*, to solve this cost-performance tradeoff problem. Without any prior knowledge of future coflows required, we show that *Lever* has a good competitive ratio in solving this cost-performance tradeoff problem.
- We conduct extensive trace-driven simulations to evaluate the performance of *Lever*. The results demonstrate that *Lever* can efficiently tradeoff the average CCT and average transmission cost, compared to state-of-the-art solutions.

The remainder of this paper is organized as follows. In Section II, we discuss the background and challenges of optimizing the cost-performance tradeoff for coflows across geo-distributed datacenters, also employ a motivating example to elaborate this problem. In Section III, we develop a mathematical model and formulate the cost-performance tradeoff problem. In Section IV, we propose an algorithm in *Lever* that seeks to minimize the average CCT and average transmission cost by routing and scheduling coflows efficiently. The experiment results are presented in Section V. Section VI discusses the related work and Section VII concludes this paper.

II. BACKGROUND, MOTIVATION AND CHALLENGES

In this section, we first present the background and employ an example to motivate our *cost-performance* tradeoff problem for coflows across geo-distributed datacenters. We then present the challenges of this problem.

A. BACKGROUND AND MOTIVATION

In today's inter-datacenter networks, there could be significant *heterogeneity* in both bandwidth capability and per unit transmission cost of different inter-datacenter links. According to the results of real-world connectivity measurement from Amazon EC2 in study [13], the available cross-datacenter bandwidth varies significantly among different inter-datacenter links: the highest bandwidth can be 100 Mbps larger than the lowest. On the other hand, different inter-datacenter links may be charged by different Internet Service Providers (ISPs) [15], [16] with different pricing models and strategies. In such a case, the cost of transmitting the same amount of data will also vary significantly across different inter-datacenter links.

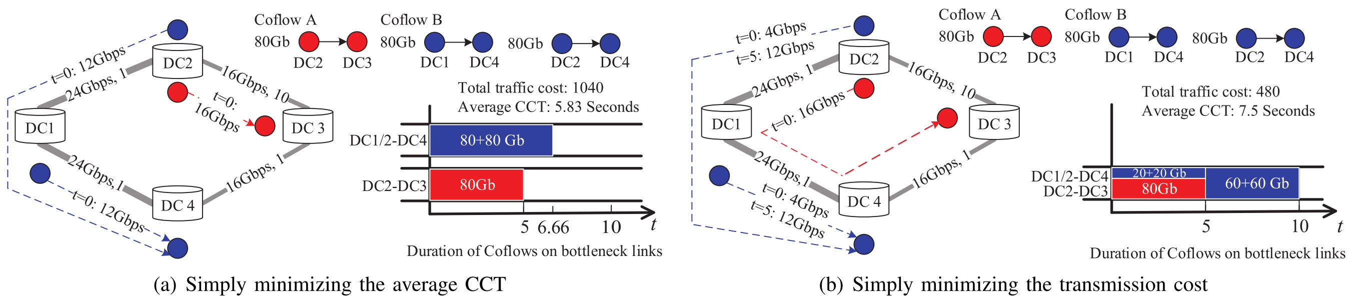

FIGURE 2. A motivating example, where (a) shows a scheme that simply minimizes the average CCT, while (b) shows a scheme that simply minimizes the transmission cost.

While recognizing the above *heterogeneities*, we employ a motivating example in Fig. 2 to exibit a better intuition of this problem. In our example, there is an inter-datacenter network consisting of 4 datacenters with 4 links. The bandwidth capacity and per unit transmission cost of each link is shown in the figure. Moreover, there are two coflows: Coflow A has one flow which requires transferring 80 Gb from DC2 to DC3; Coflow B has two flows, with one transferring 80 Gb from DC1 to DC4 and the other transferring 80 Gb from DC2 to DC4, respectively. As shown in Fig. 2(a), when simply minimizing the average CCT of these two coflows, the optimal strategy is as follows: from $t = 0$ to $t = 5$, routes the flow in coflow A along the direct path DC2→DC3 with 16 Gbps; from $t = 0$ to $t = 6.66$, routes the flow of DC1 to DC4 in coflow B along the direct path with 12 Gbps, while routes the other flow in coflow B along a detoured path DC2→DC1→DC4 with 12 Gbps. In such a case, CCTs of coflows A and B are 5s and 6.66s, respectively, and thus the average CCT is 5.83s. Though this scheme can minimize the average CCT, it introduces high transmission cost, i.e., $80 * 10 + 80 * 1 + 80 * (1 + 1) = 1040$. On the contrary, if switching the flow in coflow A to another detoured path DC2→DC1→DC4→DC3, the total transmission cost can then be reduced to 480, as shown in Fig. 2(b). Unfortunately the average CCT will be increased to 7.5s at the same time. This implies that there is a tradeoff point between the objectives of minimizing both the average CCT and the transmission cost.

B. CHALLENGES

The above example with simple settings looks straightforward. Nevertheless the general problem of jointly considering coflow scheduling and routing to simultaneously minimize the average CCT and the transmission cost can still be difficult due to the following challenges. *First*, the two goals of minimizing the average CCT and minimizing the transmission cost contradict with each other: simply achieving one goal can result in negative impact on the other goal. *Second*, the coflow scheduling decision will impact the routing decision, implying that coflow scheduling and coflow routing are deeply coupled with each other. *Third*, in most real-world situations, we can only know all the information about coflows that have arrived, making it even harder to obtain an optimal solution.

III. MODELING AND PROBLEM FORMULATION

In this section, we develop a mathematical model to study the optimization problem of jointly considering coflows scheduling and routing to simultaneously minimize the transmission cost and the average CCT of coflows.

A. MATHEMATICAL MODEL

We abstract the inter-datacenter network as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of datacenters and $\mathcal{L}$ is the set of links. Each link $l \in \mathcal{L}$ has a bandwidth capacity R_l. To indicate the cost diversity of different links, let c_l denote the cost of transmitting one unit of data on link l.

A coflow is a set of correlated parallel flows, where a coflow can only be considered as finished after all its parallel flows have been completed. Let $\mathcal{K}$ denote the set of coflows. Each coflow arrives at time τ_k, with a set of flows $\mathcal{F}_k$. For each flow $i \in \mathcal{F}_k$, let s_i^k, e_i^k and d_i^k denote the source node, destination node and its volume, respectively. Let $\mathcal{P}_i^k$ denote the set of feasible paths for flow $i \in \mathcal{F}_k$.

To indicate the routing decision, we let $\alpha_{i,p}^k$ denote the percentage of volume d_i^k that is routed along path $p \in \mathcal{P}_i^k$. Furthermore, to indicate the scheduling decision, we let $b_{i,p}^k(\tau)$ denote the bandwidth allocated to flow $i \in \mathcal{F}_k$ on path $p \in \mathcal{P}_i^k$ at time τ. Note that $b_{i,p}^k(\tau)$ can be zero for some τ's, which means this flow is either waiting for transmission, or not routed along path p. The main notations used throughout this paper are listed in Table 1.

Remarks: In our mathematical model, a flow can potentially be routed along multiple paths, resulting in packet-level reordering. In fact, such problem can be effectively resolved with mature techniques like MPTCP [25]. Moreover, similar to existing studies [11], [18], [22], [26], all information about flows for a given coflow is assumed to be known as soon as this coflow arrives.

B. PROBLEM FORMULATION

We are now in a position to formulate the *cost-performance* tradeoff problem formally as an optimization problem that simultaneously minimizes both the average transmission cost

TABLE 1. **Notations and definitions.**

Symbol	Definition
$\mathcal{N}$	The set of datacenters
$\mathcal{L}$	The set of inter-datacenter links
R_l	The bandwidth capacity of link $l \in \mathcal{L}$
c_l	The transmission cost per unit data of link $l \in \mathcal{L}$
$\mathcal{K}$	The set of coflows
$\mathcal{F}_k$	The set of flows in coflow $k \in \mathcal{K}$
$\mathcal{P}_i^k$	The set of feasible paths for flow $i \in \mathcal{F}_k$
τ_k	The arrival time of coflow $k \in \mathcal{K}$
t_k	CCT of coflow $k \in \mathcal{K}$
s_i^k	The source node of flow $i \in \mathcal{F}_k$
e_i^k	The destination node of flow $i \in \mathcal{F}_k$
d_i^k	The data size of flow $i \in \mathcal{F}_k$
$\alpha_{i,p}^k$	The percentage of d_i^k that is routed on path $p \in \mathcal{P}_i^k$
$b_{i,p}^k(\tau)$	The amount of bandwidth allocated to flow $i \in \mathcal{F}_k$ on path $p \in \mathcal{P}_i^k$ at time τ

and the average CCT of coflows across geo-distributed datacenters, as shown in the following problem **P1**:

$$\min_{\alpha_{i,p}^k, b_{i,p}^k(\tau)} \frac{1}{K}\sum_{k\in\mathcal{K}} t_k + \frac{1}{K}\sum_{l\in\mathcal{L}} c_l \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{F}_k}\sum_{p\in\mathcal{P}_i^k} I_{(l\in p)} d_i^k \alpha_{i,p}^k \quad (1)$$

We'd not deep dive P1 but only a briefing here. Eq. (1) is the objective function that minimizes a combination of the average CCT and average transmission cost across all coflows. The term $\frac{1}{K}\sum_{k\in\mathcal{K}} t_k$ is the average CCT of coflows, and the term $\frac{1}{K}\sum_{l\in\mathcal{L}} c_l \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{F}_k}\sum_{p\in\mathcal{P}_i^k} I_{(l\in p)} d_i^k \alpha_{i,p}^k$ is the average transmission cost caused by all coflows, where $I_{(l\in p)}$ is equal to 1 if $l \in p$ and 0 otherwise. Note that by adding weight factors in front of both the average CCT and the average transmission cost, any desired tradeoff point between these 2 objectives can be achieved, e.g. if one is just willing to pay for a bit more performance. In addition, such preference between performance and cost may also vary along with the specific DCI setting, e.g. private WAN or public cloud. For simplicity, we assume the weight factor for each of conflicting objectives is set to 1.

Intuitively, the above problem **P1** is a LP problem, and it may be a step towards the right direction to design an offline optimal coflow scheduling and routing algorithm, such that both the average CCT and the average transmission cost can be minimized simultaneously. However, such offline algorithm inevitably relies on *a prior* knowledge of all the information about future coflows, including the source/destination nodes as well as the volumes. Such information can only be known when a coflow arrives, and can not be predicted without large-scale complex prediction techniques and systems. Hence, an online algorithm is more desired to solve **P1**.

IV. *LEVER* DESIGN

In this section, we propose an online control framework—*Lever*, to efficiently tradeoff the average CCT and total transmission cost for coflows running across geo-distributed datacenters, by coordinating both coflow scheduling and routing techniques. In this design, we will formulate a LP problem for each coflow as soon as it arrives. Based on the optimal solution of this LP, we rescale the bandwidth allocated to all existing coflows, meanwhile keeping the original routing decision for each coflow. In the following parts, we start by showing how to optimize the cost-performance tradeoff problem for a single coflow.

A. SINGLE COFLOW COST-PERFORMANCE TRADEOFF

We consider a special case, where there is only one coflow in the network. This coflow carries a set of individual flows $\mathcal{F}$, with the data volume of each flow $i \in \mathcal{F}$ being d_i^k. Let $\mathcal{P}_i$ denote the set of feasible paths for flow i. Denote $b_{i,p}$ and $\alpha_{i,p}$ as the scheduling and routing decisions, respectively. Now we have the following optimization problem, denoted as **P2**:

$$\min_{\alpha_{i,p}} \; t + \sum_{l\in\mathcal{L}} c_l \sum_{i\in\mathcal{F}}\sum_{p\in\mathcal{P}_i} I_{(l\in p)} d_i \alpha_{i,p} \quad (2)$$

$$\textbf{subject to:} \quad \sum_{i\in\mathcal{F}}\sum_{p\in\mathcal{P}_i} I_{(l\in p)} b_{i,p} \le R_l, \quad \forall l \in \mathcal{L} \quad (3)$$

$$\sum_{p\in\mathcal{P}_i} \alpha_{i,p} = 1, \quad \forall i \in \mathcal{F} \quad (4)$$

$$b_{i,p} \cdot t = \alpha_{i,p} d_i, \quad \forall i \in \mathcal{F}, \ \forall p \in \mathcal{P}_i \quad (5)$$

$$\alpha_{i,p} \ge 0, \quad \forall i \in \mathcal{F}, \ \forall p \in \mathcal{P}_i \quad (6)$$

The variable t represents CCT of this coflow, and the term $\sum_{l\in\mathcal{L}} c_l \sum_{i\in\mathcal{F}}\sum_{p\in\mathcal{P}_i} I_{(l\in p)} d_i \alpha_{i,p}$ calculates the transmission cost that would be produced by this coflow. Eq. (3) is the link capacity constraint, Eq. (4) implies that the summation of all flow splits must be exactly equal to the volume of each flow. It should be noted that the scheduling decision variable is a function of time in the previous mathematical model (Section III), but problem **P2** treat it as a constant value here. This implies that for a flow i routing along path p, its bandwidth should be equal to $b_{i,p}$ when the corresponding flow being transmitted, or zero when finished. As such, $b_{i,p}$ can directly be calculated via Eq. (5). Similarly, Eq. (6) implies that all the routing decision variables are non-negative.

It is clear that problem **P2** is a LP problem with up to $|F| \times \max_i |\mathcal{P}_i|$ variables, where $|F|$ is the number of flows in this coflow and $\max_i |\mathcal{P}_i|$ is the maximum number of paths across all sets of $\mathcal{P}_i, \forall i$. Since **P2** is a LP, it can be easily solved by standard linear programming solvers e.g. MOSEK [27]. Therefore, once a coflow arrives, we can immediately obtain its routing and scheduling decisions by solving the relevant LP problem.

B. EXTENDING TO THE MULTI-COFLOW SCENARIO

When there are multiple coflows coexisting in the network, we treat the LP solution for each coflow as a black box, and design a competitive algorithm to solve the original problem **P2**. The key idea here is that when a new coflow arrives, we first formulate a relevant LP problem **P2**, and then solve this LP with standard LP solver to obtain its routing and scheduling decisions. Finally, we rescale the bandwidth allocation of all coflows but keep the original routing decision for each coflow.

The whole procedure to handle this multi-coflow scenario is shown in Algorithm 1. To avoid frequent calculations, it computes the routing decision for each coflow only once, by solving the relevant LP problem **P2** with the assumption that each coflow occupies the network exclusively. The routing and scheduling decisions computed for each coflow are then stored in $\{\{\alpha_{i,p}^k\}, \{b_{i,p}^k\}\}$ to avoid duplicated calculations in future. Since there are multiple coflows coexisting in the network, coflows can significantly interact with each other. In such a case, the stored solution for each coflow may become infeasible, as the network is shared by multiple coflows, rather than a single coflow. To make the solution for each coflow still be feasible, Algorithm 1 scales down the bandwidth allocation for each coflow k with a weight factor ϵ_k. The weight factor ϵ_k is computed based on the optimal objective of the LP problem **P2**, and is specific to each coflow k. Such weight factor can somehow guarantee the fairness among multiple competing coflows: more bandwidth will be allocated to large coflows, while short coflows be given relatively less bandwidth. Finally, after re-weight the scheduling decision, Algorithm 1 again scales the bandwidth allocation for all coflows by a same largest possible factor, so as to achieve the desired result of optimization.

Algorithm 1 An Online Cost-Performance Tradeoff Algorithm for the Multi-Coflow Scenario

```
Input: $\mathcal{F}_k, \tau_k, t_k, \forall k; s_i^k, e_i^k, d_i^k, \mathcal{P}_i^k, \forall k, \forall i \in \mathcal{F}_k$
Output: Routing and scheduling decisions for all coflows
1: while a new coflow arrives or an existing coflow finishes do
2:     Define $\Omega$ as the set of coflows that are not completed till the current time.
3:     for each coflow $k$ in $\Omega$ do
4:         Define $\epsilon_k := \frac{\sqrt{OPT_{P2}^k}}{\sum_{k'\in\Omega}\sqrt{OPT_{P2}^{k'}}}$, where $OPT_{P2}^k$ is the optimal objective value of the LP P2 for coflow $k$.
5:         Update the routing and scheduling solution of $k$ as $\{\{\alpha_{i,p}^k\}, \{\epsilon_k b_{i,p}^k\}\}$, where $\{\{\alpha_{i,p}^k\}, \{b_{i,p}^k\}\}$ is the optimal solution to the LP P2 of coflow $k$;
6:     end for
7:     Scale the bandwidth of all flows in $\mathcal{J}_\Omega$ with a same factor which is the largest possible value to make the routing and scheduling decisions still be valid, so as to achieve the desired result of optimization.
8: end while
```

C. PERFORMANCE ANALYSIS

Algorithm 1 essentially divides the original problem **P1** into multiple sub-problems, and then leverages optimal solutions of all sub-problems to construct one feasible solution of **P1**. So, one may question that what is the optimality of Algorithm 1 with respect to the original problem **P1**. To answer this question, we state the following lemma and theorem with proof to demonstrate that Algorithm 1 has a good competitive ratio for **P1**.

Lemma 1: For each coflow k, denote OPT_{P2}^k as the optimal objective value with respect to the LP problem **P2**, and let OPT_{alg}^k denote the objective value substituting for the routing and scheduling decisions achieved by Algorithm 1 in Eq. (2). As such, we have $OPT_{alg}^k \le \frac{1}{\epsilon} OPT_{P2}^k$ for all k, where $\epsilon := \min_{k\in\mathcal{K}} \epsilon_k$.

Proof: Given a coflow k, define $\{\{\dot{\alpha}_{i,p}^k\}, \{\dot{b}_{i,p}^k\}\}$ as the optimal solution with respect to problem **P2**. Thus, we have the following equality:

$$OPT_{P2}^k = \frac{\dot{\alpha}_{i,p} d_i}{\dot{b}_{i,p}} + \sum_{l\in\mathcal{L}} c_l \sum_{i\in\mathcal{F}} \sum_{p\in\mathcal{P}_i} I_{(l\in p)} d_i \dot{\alpha}_{i,p}$$

On the other hand, when multiple coflows coexist in the network, Algorithm 1 only rescales the bandwidth allocation for each coflow k with the factor ϵ_k, while always keeps its original routing decision. Thus, we have:

$$OPT_{alg}^k = \frac{\dot{\alpha}_{i,p} d_i}{\epsilon_k \dot{b}_{i,p}} + \sum_{l\in\mathcal{L}} c_l \sum_{i\in\mathcal{F}} \sum_{p\in\mathcal{P}_i} I_{(l\in p)} d_i \dot{\alpha}_{i,p}$$

As $\epsilon_k := \frac{\sqrt{OPT_{P2}^k}}{\sum_{k'\in\Omega}\sqrt{OPT_{P2}^{k'}}}$, we have $\frac{1}{\epsilon_k} \ge 1$ for each coflow k. Then, we have:

$$\begin{aligned} OPT_{alg}^k &= \frac{\dot{\alpha}_{i,p} d_i}{\epsilon_k \dot{b}_{i,p}} + \sum_{l\in\mathcal{L}} c_l \sum_{i\in\mathcal{F}} \sum_{p\in\mathcal{P}_i} I_{(l\in p)} d_i \dot{\alpha}_{i,p} \\ &\le \frac{1}{\epsilon_k}\left(\frac{\dot{\alpha}_{i,p} d_i}{\dot{b}_{i,p}} + \sum_{l\in\mathcal{L}} c_l \sum_{i\in\mathcal{F}} \sum_{p\in\mathcal{P}_i} I_{(l\in p)} d_i \dot{\alpha}_{i,p}\right) \\ &= \frac{1}{\epsilon_k} OPT_{P2}^k \end{aligned}$$

Defining $\epsilon := \min_{k\in\mathcal{K}} \epsilon_k$, the lemma can thus be inferred. ■

Theorem 1: Algorithm 1 is $\frac{K}{\epsilon}$-competitive for the original problem **P1**, where K is the total number of coflows in $\mathcal{K}$.

Proof: Let OPT_{P1} denote the optimal objective value for the original problem **P1**. Since each coflow k can contribute to OPT_{P1} with no less than OPT_{P2}^k when it monopolizes the network, we have $OPT_{P1} \ge \frac{1}{K}\sum_{k=1}^K OPT_{P2}^k$. Define OPT_{alg} as the objective value substituting for the routing and scheduling decisions achieved by Algorithm 1 in Eq. (1). In the following parts, we focus on the proof process of $OPT_{alg} \le \frac{K}{\epsilon} \cdot OPT_{P1}$.

Consider the following optimization problem for the subset Ω of $\{1, \ldots, K\}$:

$$\text{Minimize} \sum_{k\in\Omega} \frac{OPT_{P2}^k}{z_k}$$

$$\text{Subject to:} \sum_{k\in\Omega} z_k \le 1,$$

where z_k is a non-negative value. By applying the Cauchy-Schwarz inequality, we have:

$$\sum_{k\in\Omega}\frac{OPT_{P2}^{k}}{z_k} \geq (\sum_{k\in\Omega}\frac{OPT_{P2}^{k}}{z_k})\cdot(\sum_{k\in\Omega}z_k) \geq (\sum_{k\in\Omega}\sqrt{OPT_{P2}^{k}})^2.$$

This implies that when the optimal solution for the above optimization problem is equal to $z_k = \sqrt{OPT_{P2}^{k}}/\sum_{k\in\Omega}\sqrt{OPT_{P2}^{k}}, \forall k \in \Omega$, it can be optimally solved. Thus, for each Ω in every iteration of Algorithm 1, weighted factor ϵ_k's are optimally picked with respect to Ω. More specifically, ϵ_k's have least impact on the optimal objective function OPT_{P1} when rescaling the bandwidth of each coflow.

Define $\epsilon_k^{(K)} := \sqrt{OPT_{P2}^{k}}/\sum_{k=1}^{K}\sqrt{OPT_{P2}^{k}}$. Since Ω might be a subset of $\{1,\ldots,K\}$, we have the following inequality for any k:

$$\epsilon_k \geq \frac{\sqrt{OPT_{P2}^{k}}}{\sum_{k=1}^{K}\sqrt{OPT_{P2}^{k}}} = \epsilon_k^{(K)}.$$

Combining with Lemma 1, we have the following inequality:

$$\frac{OPT_{alg}^{k}}{\epsilon_k} \leq \frac{OPT_{alg}^{k}}{\epsilon_k^{(K)}} \leq \frac{1}{\epsilon}\cdot\frac{OPT_{P2}^{k}}{\epsilon_k^{(K)}} = \frac{1}{\epsilon}\sqrt{OPT_{P2}^{k}}\sum_{k=1}^{K}\sqrt{OPT_{P2}^{k}}.$$

By applying the Cauchy-Schwarz inequality again, we get

$$\begin{aligned} OPT_{alg} &= \frac{1}{K}\sum_{k=1}^{K}\frac{OPT_{alg}^{k}}{\epsilon^{k}} \\ &\leq \frac{1}{K}\sum_{k=1}^{K}\frac{OPT_{alg}^{k}}{\epsilon_k^{(K)}} \\ &\leq \frac{1}{K\epsilon}(\sum_{k=1}^{K}\sqrt{OPT_{P2}^{k}})^2 \\ &\leq \frac{1}{\epsilon}\sum_{k=1}^{K}OPT_{P2}^{k} \\ &\leq \frac{K}{\epsilon}OPT_{P1}. \end{aligned}$$

Thus, proved. ■

V. PERFORMANCE EVALUATION

In this section, we conduct real-world trace-driven simulations to realistically evaluate the performance of the algorithm we proposed. Specifically, we examine CCT and the corresponding cost for each coflow, as well as quantify the performance of our algorithm regarding to the average CCT and average transmission cost. In the following parts, we start by presenting the simulation setup, then showing the simulation results.

TABLE 2. Coflows categorized by length and width.

Coflow types:	SN	LN	SW	LW
Length:	Short	Long	Short	Long
Width:	Narrow	Narrow	Wide	Wide
% Coflows:	71%	6%	9%	14%
% Bytes:	0.03%	0.04%	0.31%	99.62%

A. SIMULATION SETUP

We simulate an inter-datacenter network with 20 datacenters and 380 ($= 20 \times 19$) links. To mimic the heterogeneous bandwidth environments, the capacity of each inter-datacenter link is randomly chosen between 100Mbps and 2Gbps. Such heterogeneous link capacities can practically be archived by Linux Traffic Control [28]. In our simulation, we adjust the data transmission cost per unit bandwidth for each link within the range [0.0003, 0.0012]\$/Mb. This range is also utilized by Amazon to price data transfers in the U.S. east region [29].

We conduct our simulations based on a Hive/MapReduce trace provided by Facebook [18], [24]. This trace is widely used in existing literatures that focuses on coflow studies, which contains 526 coflows collected from a 3000-machines 150-racks cluster. Accordingly, we scale down the above coflows to match this 20-datacenters inter-datacenter network in our deployment. During the scaling process, we preserve the original communication pattern of the coflows. It should be noted that the original trace only contains the amount of traffic which each reducer needs to fetch. Hence, we distribute them to the mappers in a unified manner, and accordingly obtain the size of each flow between any datacenter-pair.

A coflow is a collection of parallel flows which can be characterized by two key parameters. The first one is the coflow *length*, which is defined as the size of the largest flow in bytes, and the second one is the coflow *width*—the number of parallel flows of this coflow. Similar to [18] and [26], we divide all the coflows used in our simulations into 4 categories based on their characterizes, as shown in Table 2. More specifically, a coflow is considered to be *short* if its largest flow is less than 50Mb and *narrow* if it has no more than 20 flows.

In our simulations, we compare the following schemes with *Lever*:

- **RAPEIR**: seamlessly combines routing and scheduling techniques to optimize the average CCT of coflows [11]. This scheme corresponds to a performance-only scheme that simply ignores the transmission cost.
- **Cost-only**: purely minimizes the average transmission cost for coflows running over the inter-datacenter network, which is conceptually equivalent to the core method in the literature [15].

B. SIMULATION RESULTS

1) THE PERFORMANCE ON CCT

Fig. 3 first illustrates the average CCT of coflows achieved by different schemes. We have the following observations from this figure: 1) For most of coflow types, our proposed *Lever* can maintain an average CCT between RAPIER and Cost-only schemes. This is reasonable because our proposed *Lever* targets on trading off the average CCT and average cost. One question here is why *Lever* results in a very high average CCT for LN type of coflows. The root reason may be that *Lever* always routes LN coflows along a same set of paths. Fortunately, since the percentage of LN is quite small, the average CCT of LN will not affect the overall average CCT across the entire set of coflows. 2) Though RAPIER is a performance-only scheme, it shows higher CCT than the Cost-only scheme for LW type of coflows. This is because LW coflows are long and wide, leave RAPIER little space to optimize the corresponding CCT. 3) Across all types of coflows, our proposed *Lever* can speed up the average CCT by 1.1×, comparing to the Cost-only scheme. Such results demonstrate that *Lever* can efficiently decrease the average CCT of coflows in general.

To understand CCT of coflows at a microscopic level, we further plot CDF (Cumulative Distribution Function) of the completion time across all coflows schemes of RAPIER, *Lever* and Cost-only in Fig. 4. Note that the X-axes are in logarithmic scale. We can clearly see that the performance of *Lever* on CCT is between that of RAPIER and Cost-only, as the *Lever* curve lies between the curves of RAPIER and Cost-only. We can further check that the percentages of coflows completed within 1000ms are 37.64%, 32.32% and 20.34% by RAPIER, *Lever*, and Cost-only schemes, respectively. Under these three schemes, all coflows can be completed within 1370850ms, 1426500ms and 1372950ms, respectively.

2) THE PERFORMANCE ON THE TRANSMISSION COST

Lever aims at optimizing the tradeoffs between the average CCT and average transmission cost. Therefore, another important metric is the transmission cost introduced by coflows running across geo-distributed datacenters. To quantify this metric, we record the corresponding costs produced by each coflow, and plot the average cost by RAPIER, *Lever*, and Cost-only schemes in Fig 5. Note that the Y-axes are in logarithmic scale. It is clear that *Lever* can achieve an average transmission cost that is between RAPIER and Cost-only schemes, across all coflow types. Combining the results of Fig.3 and Fig. 4, we can conclude that *Lever* can efficiently trade off the average CCT and the average transmission cost. Furthermore, throughout all coflows, the average transmission cost introduced by RAPIER, *Lever*, and Cost-only schemes are 126.0509$, 63.3836$ and 39.1150$, respectively. This implies that comparing to RAPIER, our proposed *Lever* can actually reduce the average cost by 49.72%.

To clearly illustrate the underlying reason for such cost reduction, we further plot CDF of the transmission cost

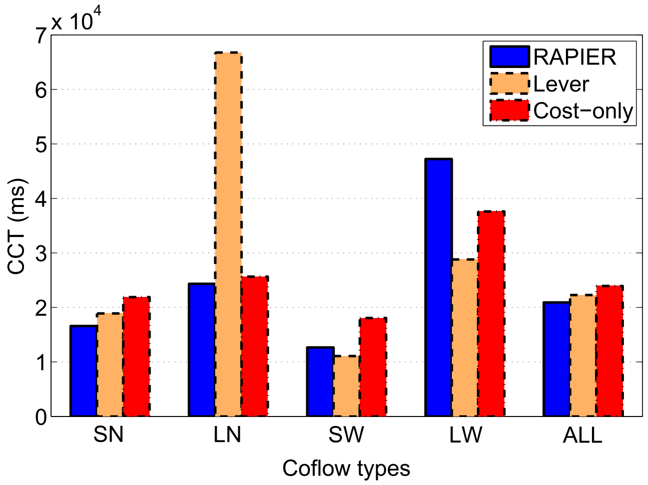

FIGURE 3. The average CCT of coflows, achieved by RAPIER, *Lever* and Cost-only schemes.

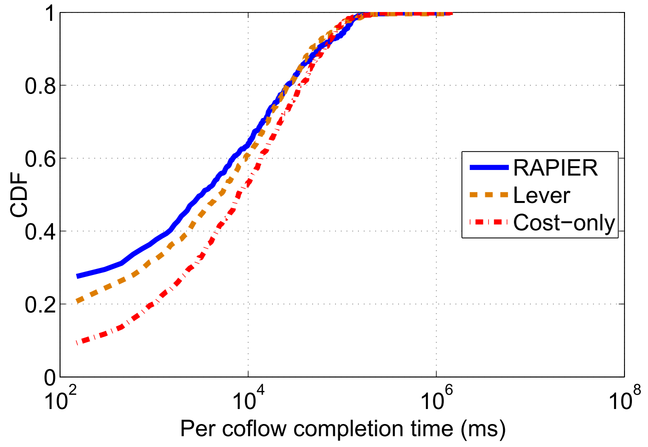

FIGURE 4. CDFs of per coflow CCT by RAPIER, *Lever* and Cost-only schemes.

across all coflows by RAPIER, *Lever*, and Cost-only schemes in Fig. 6. It can be easily checked that the *Lever* curve is between the curves of RAPIER and Cost-only, implying the performance of *Lever* on the transmission cost is between RAPIER and Cost-only schemes as well. We can further observe that 98% of coflows produce a cost less than 1140$ under the *Lever* scheme, while regarding to RAPIER and Cost-schemes, the corresponding values are 2260$ and 700$, respectively.

Remarks: The above results verify the key idea of *Lever*: optimizing tradeoffs between the average CCT and average transmission cost of coflows running across geo-distributed datacenters.

VI. RELATED WORK

In this section, we only review some closely related work, including minimizing the average CCT of coflows and minimizing the transmission cost of inter-datacenter traffic.

Regarding minimizing the average CCT of coflows, existing work can be divided into three categories based on the techniques employed. The *first* category is scheduling. For instance, Varys [18] first applies an SEBF

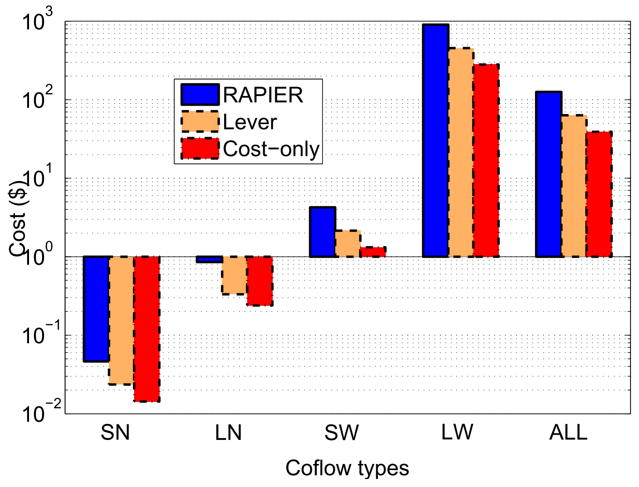

FIGURE 5. The average cost of coflows by RAPIER, *Lever* and Cost-only.

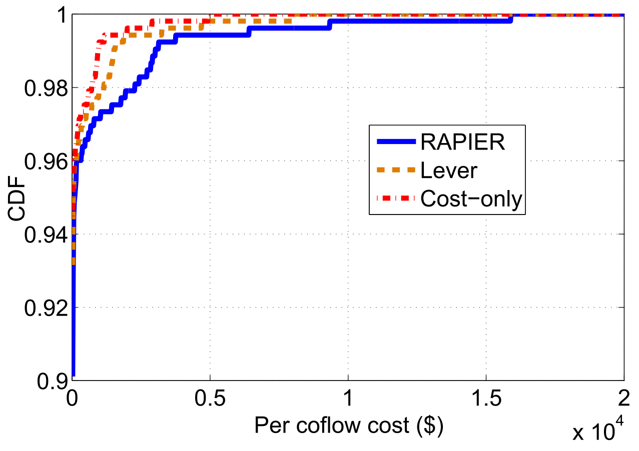

FIGURE 6. CDFs of per coflow cost by RAPIER, *Lever* and Cost-only schemes.

(Shortest-Effective-Bottleneck-First) heuristic to determine the schedule order of multiple concurrent coflows, and then utilizes an MADD (Minimum-Allocation-for-Desired-Duration) algorithm to allocate rates to individual flows. While Varys is efficient, it relies on the complete prior information of coflows. Motivated by this point, Aalo [10] applies multi-level feedback queues (MLFQ) to schedule coflows, without requiring any prior knowledge of coflows. Taking one step further, CODA [19] employs an algorithm to automatically cluster network flows into different coflows before scheduling, such that no modification is needed for applications to extract coflows. The *second* category focuses on combining scheduling and routing to minimize the average CCT of coflows. RAPIER [11] is perhaps the first work that seamlessly integrates routing and scheduling to speed up the completion of coflows in datacenter networks. But RAPIER's solution can not provide a theoretical upper bound of the average CCT. Therefore Li *et al.* [26] propose OMCoflow, an efficient online algorithm, to route and schedule coflows simultaneously. OMCoflow has been proved to provide a theoretical guarantee of the average CCT. The *third* category of work is to leverage techniques of endpoint (or reduce tasks) placement when scheduling coflows in the network. For instance, CLARINET [6] first assigns locations to reduce tasks to determine the endpoint placement of flows firstly, and then schedules the resulting network flows. Nevertheless, it considers endpoint placement and scheduling independently rather than jointly. Motivated by this work, Li *et al.* [22] propose SmartCoflow, which considers the endpoint placement and coflow scheduling simultaneously. Although the above methods are efficient in minimizing the average CCT of coflows, they all neglect an indispensable metric—transmission cost that would be caused by the coflows when running over the inter-datacenter network. In addition to above three categories of related work, some focus on achieving fairness among coflows [20], [21], which also do not consider the transmission cost.

Regarding minimizing the transmission cost of inter-datacenter traffic, NetStitcher [30] applies a store-and-forward method, which temporally stores the traffic at intermediate datacenters which to be forwarded to the destination once there is available bandwidth. Nevertheless, NetStitcher can only reduce the transmission cost for a single bulk transfer, which is far from realistic. Motivated by this work, Feng *et al.* [16] present JetWay, which considers the co-existence of multiple video flows when minimizing the transmission cost. Jalaparti *et al.* [14] present Pretium, which leverages a dynamic pricing to regulate the traffic among inter-datacenter links. This can somehow maintain a low cost for end users, but cannot minimize the transmission cost for network operators. Most recently, Li *et al.* [23] propose to utilize "free" time slots through the percentile pricing model to schedule the inter-datacenter traffic, such that the transmission cost can have a chance to be significantly reduced or even minimized. Above methods are efficient in minimizing the cost for inter-datacenter traffic, which, however, only focus on individual flows. Actually, there are also plenty of researches on minimizing the transmission cost introduced by jobs running across geo-distributed datacenters. More specifically, Vulimiri *et al.* [2] focus on optimizing the execution plan and leveraging techniques of caching to reduce the cost caused by query jobs running over the inter-datacenter network. Pixida [3] utilizes the method of graph partition to minimize the transmission cost. As an extension of Pixida, WANalytics [4] further employs the cache mechanism, which, however, can cause overhead when caching and computing within a datacenter. On the other hand, JetStream [31] focuses on aggregating and degrading streaming data before being transmitted over the inter-datacenter network. Though reducing the transmission cost, these solutions are actually flow-level which do not consider the flow dependency semantics, i.e. all coflow-agnostic.

VII. CONCLUSION

In this paper, we jointly consider the objectives of both minimizing cost and maximizing performance (i.e., decreasing the completion time of coflows) when transmitting coflows across geo-distributed datacenters. To characterize the combination of these two objectives, we develop a rigorous mathematical model and formulate an optimization problem to minimize the average CCT and average transmission cost concurrently. Then, we propose a coflow-aware framework—*Lever*, to solve this problem in an online manner. *Lever* first applies linear programming (LP) to obtain a routing and scheduling solution for one single coflow, and then extend this LP to handle multiple, dynamically arrived coflows. The theoretical analysis shows that *Lever* has a non-trivial competitive ratio in solving this cost-performance tradeoff problem. The experimental results further demonstrate that *Lever* can speed up the completion of coflows and reduce the transmission cost substantially.

REFERENCES

[1] Q. Pu *et al.*, "Low latency geo-distributed data analytics," in *Proc. ACM SIGCOMM*, 2015, pp. 421–434.

[2] A. Vulimiri, C. Curino, P. B. Godfrey, T. Jungblut, J. Padhye, and G. Varghese, "Global analytics in the face of bandwidth and regulatory constraints," in *Proc. USENIX NSDI*, 2015, pp. 323–336.

[3] K. Kloudas. M. Mamede, N. Preguiça, and R. Rodrigues, "Pixida: Optimizing data parallel jobs in wide-area data analytics," in *Proc. VLDB Endowment*, 2015, pp. 72–83.

[4] A. Vulimiri, C. Curino, B. Godfrey, K. Karanasos, and G. Varghese, "Wanalytics: Analytics for a geo-distributed data-intensive world," in *Proc. CIDR*, 2015, pp. 1087–1092.

[5] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in *Proc. ACM SoCC*, 2015, pp. 111–124.

[6] R. Viswanathan, G. Ananthanarayanan, and A. Akella, "Clarinet: Wan-aware optimization for analytics queries," in *Proc. USENIX OSDI*, 2016, pp. 435–450.

[7] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proc. ACM Workshop Hot Topics Netw.*, 2012, pp. 31–36.

[8] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven wan," in *Proc. ACM SIGCOMM*, 2013, pp. 15–26.

[9] H. Zhang *et al.*, "Guaranteeing deadlines for inter-datacenter transfers," in *Proc. 10th Eur. Conf. Comput. Syst.*, 2015, Art. no. 10.

[10] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," in *Proc. ACM SIGCOMM*, 2015, pp. 393–406.

[11] Y. Zhao *et al.*, "Rapier: Integrating routing and scheduling for coflow-aware data center networks," in *Proc. IEEE INFOCOM*, Hong Kong, Apr. 2015, pp. 424–432.

[12] K. Hsieh *et al.*, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *Proc. NSDI*, 2017, pp. 629–647.

[13] Z. Hu, B. Li, and J. Luo, "Flutter: Scheduling tasks closer to data across geo-distributed datacenters," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[14] V. Jalaparti, I. Bliznets, S. Kandula, B. Lucier, and I. Menache, "Dynamic pricing and traffic engineering for timely inter-datacenter transfers," in *Proc. ACM SIGCOMM*, 2016, pp. 73–86.

[15] W. Li, K. Li, D. Guo, G. Min, H. Qi, and J. Zhang, "Cost-minimizing bandwidth guarantee for inter-datacenter traffic," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2016.2629506.

[16] Y. Feng, B. Li, and B. Li, "Jetway: Minimizing costs on inter-datacenter video traffic," in *Proc. ACM Multimedia*, Nara, Japan, 2012, pp. 259–268.

[17] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *Proc. ACM SIGCOMM*, Toronto, ON, Canada, 2011, pp. 98–109.

[18] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in *Proc. ACM SIGCOMM*, Chicago, IL, USA, 2014, pp. 443–454.

[19] 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. ACM SIGCOMM*, 2016, pp. 160–173.

[20] W. Wang, S. Ma, B. Li, and B. Li, "Coflex: Navigating the fairness-efficiency tradeoff for coflow scheduling," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

[21] L. Chen, W. Cui, B. Li, and B. Li, "Optimizing coflow completion times with utility max-min fairness," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[22] W. Li, X. Yuan, K. Li, H. Qi, and X. Zhou, "Leveraging endpoint flexibility when scheduling coflows across geo-distributed datacenters," in *Proc. IEEE INFOCOM*, Apr. 2018.

[23] W. Li, X. Zhou, K. Li, H. Qi, and D. Guo, "More peak, less differentiation: Towards a pricing-aware Online control framework for inter-datacenter transfers," in *Proc. IEEE ICDCS*, Jun. 2017, pp. 2105–2110.

[24] *Synthesized data From Real-World Traces of Data-Intensive Applications for Coflow Benchmarking*. Accessed: Feb. 12, 2018. [Online]. Available: https://github.com/coflow/coflow-benchmark

[25] C. Raiciu *et al.*, "How hard can it be? Designing and implementing a deployable multipath TCP," in *Proc. USENIX NSDI*, 2012, p. 29.

[26] Y. Li *et al.*, "Efficient Online coflow routing and scheduling," in *Proc. ACM MobiHoc*, 2016, pp. 161–170.

[27] E. D. Andersen and K. D. Andersen, "The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm," in *High Performance Optimization*. Boston, MA, USA: Springer, 2000, pp. 197–232.

[28] *Linux Traffic Control*. Accessed: Feb. 12, 2018. [Online]. Available: http://lartc.org/manpages/tc.txt

[29] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 854–862.

[30] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," in *Proc. ACM SIGCOMM*, 2011, pp. 74–85.

[31] A. Rabkin, M. Arye, S. Sen, V. S. Pai, and M. J. Freedman, "Aggregation and degradation in jetstream: Streaming analytics in the wide area," in *Proc. USENIX NSDI*, 2014, pp. 275–288.

XINPING XU is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Dalian University of Technology, China. His research interests include datacenter networks and cloud computing.

WENXIN LI received the B.E. degree from the School of Computer Science and Technology, Dalian University of Technology, China, in 2012, where he is currently pursuing the Ph.D. degree. His research interests include datacenter networks and cloud computing.

KEQIU LI (SM'12) received the bachelor's and master's degrees from the Department of Applied Mathematics, Dalian University of Technology, China, in 1994 and 1997, respectively, and the Ph.D. degree from the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, in 2005. He held a post-doctoral position at The University of Tokyo, Japan, for two years. He is currently a Professor with the School of Computer Science and Technology, Dalian University of Technology. He has authored or co-authored over 100 technical papers, such as the IEEE TPDS, ACM TOIT, and ACM TOMCCAP. His research interests include internet technology, data center networks, cloud computing, and wireless networks. He is an Associate Editor of the IEEE TPDS and the IEEE TC.

HENG QI received the B.S. degree from Hunan University in 2004, and the M.E. and Ph.D. degrees from the Dalian University of Technology, China, in 2006 and 2012, respectively. From 2016 to 2017, he was a JSPS Overseas Research Fellow with the Graduate School of Information Science, Nagoya University, Japan. He is currently an Associate Professor with the School of Computer Science and Technology, Dalian University of Technology. His research interests include computer network and multimedia computing.

YINGWEI JIN received the Ph.D. degree from the Dalian University of Science and Technology in 2005. He is currently a Professor with the School of Management, Dalian University of Technology, China. His research interests include computer network and security, Internet technology, and artificial intelligence.