Joint Optimization of Task Placement and Routing in Minimizing Inter-DC Coflow Completion Time

Yingya Guo*‡, Zhiliang Wang†‡, Han Zhang*‡, Xia Yin*‡, Xingang Shi†‡,and Jianping Wu*‡
*Department of Computer Science and Technology, Tsinghua University
†Institute for Network Sciences and Cyberspace, Tsinghua University
‡Tsinghua National Laboratory for Information Science and Technology (TNLIST)
Beijing, P.R. China
Email:{guoyingya, wzl, zhanghan, yxia}@csnet1.cs.tsinghua.edu.cn, {shixg, jianping}@cernet.edu.cn

***Abstract*—With the rapidly growing of geo-distributed applications in Internet, there are huge amount of data generated in geo-distributed datacenters to be processed and analyzed everyday. However, because of region privacy concerns and limitation of inter-DC WAN bandwidth, the traditional method to move all the geo-distributed data to a single centralized datacenter to process is not practical. Therefore, we intend to process the data where it generates by the big data applications (e.g., Hadoop) and optimize the coflow transfers in inter-DC WAN to improve the performance of the applications. Previous studies consider only one aspect: routing optimization or task placement optimization, which is insufficient. Both the placement of tasks and the routing of the coflow greatly influence coflow completion time.**

In this paper, we are the first to propose an $(1+\epsilon)$-approximation algorithm PRO (Placement and Routing Optimization) that jointly optimizes the placement of reduce tasks and the routing of a single coflow in the inter-DC WAN to reduce the coflow completion time. We also propose PROS (Placement and Routing Optimization of coflowS) to optimize the scheduling of multiple coflows offline. Then we transform the offline scheduling algorithm into online scheduling one. Compared with the methods that optimize the task placement only or routing only, PRO can reduce the single coflow completion time by 39.2%, 51.8% on average, respectively. Through extensive simulation, we can also observe that our proposed online scheduling algorithm performs better than First In First Out (FIFO) algorithm.

***Index Terms*—Inter-DC WAN; coflow completion time; task placement; routing**

I. Introduction

As with the rapid development of Internet applications, geo-distributed applications, such as online searching services from Google and Microsoft, generate a huge amount of data across different regions in the inter-Datacenter Wide Area Network (Inter-DC WAN). Big data computing frameworks (e.g., map-reduce) are deployed to analyze the large scale data in the geo-distributed inter-DC WAN. The traditional method to process the big data across multiple datacenters is to move all the data to a single datacenter and process them in a centralized manner with the big data processing applications. However, there are some limitations with this centralized manner. First, moving huge amount of data across inter-DC WAN consumes a lot of bandwidth, which is an expensive and limited resource for inter-DC WAN network. Second, because of privacy and security concerns [1], it is undesirable to move the data across regions, which may expose user information and pravicy. Third, the data is growing explosively, a single datacenter may not be able to accommodate huge amount of data in the future. Therefore, it is more efficient and secure to process the data locally, i.e, in a geo-distributed manner. The intermediate data which is processed locally is smaller in size and is efficient to be transferred, which can greatly reduce the response time of applications and improve user experience.

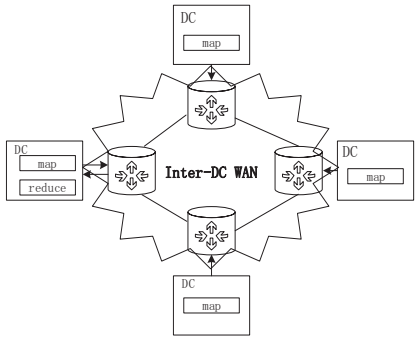


Fig. 1. Map-reduce framework in inter-DC WAN. In the shuffle phrase, the flows of a coflow transfer from all the map tasks to the reduce tasks. The placement of reduce tasks and routing of coflow both influence the coflow completion time.

Map-reduce is a programming model which consists of map, shuffle and reduce phases. Many open-source software platforms have implemented this programming model, such as Hadoop. In this paper, we mainly focus on the widely used big data computing frameworks: map-reduce. The parallel flows of the shuffle phase in the map-reduce model are defined as a coflow [2] and the completion time of the last flow determines the entire coflow completion time (CCT). Therefore, we intend to minimize the completion time of the last flow to reduce the completion time of the entire coflow. The scenario we study is shown in Fig.1. In our system, the map tasks are moved to geo-distributed datacenters. The intermediate data after the processing of the map tasks should be transferred to the reduce tasks across inter-DC WAN. Therefore, the placement of reduce tasks and the routing of the coflows in inter-DC WAN both effect the coflow completion time significantly. We need to consider both these two factors to minimize the coflow

completion time, so that we can improve the performance of the big data analytics. In the previous works of minimizing CCT in inter-DC WAN, they either optimize the reduce tasks placement [1] or the routing of the coflows [3], which is insufficient. The reason is that two factors both influence the coflow completion time and we should jointly optimize these two factors simultaneously to better reduce the coflow completion time.

In this paper, we are the first propose a framework that jointly optimizes the reduce task placement and routing of the coflows. We first formulate the problem of optimizing the completion time of single coflow as a math programming and propose an $(1+\epsilon)$-approximation algorithm PRO (Placement and Routing Optimization) to solve it. Then we propose offline and online scheduling algorithms to optimize the average completion time of multiple coflows. Through extensive experiments, the coflow completion time can be greatly reduced by PRO compared with the algorithms that only optimizes the reduce task placement or the routing. Moreover, our proposed online scheduling algorithms can outperform FIFO in minimizing the average CCT.

In summary, our contributions are summarized as follows:

- We are the first to jointly optimize the reduce task placement and routing of coflows to minimize the CCT of single coflow and average CCT of multiple coflolws.
- We propose an $(1+\epsilon)$ approximation algorithm PRO to reduce the single coflow completion time and offline and online scheduling algorithms to optimize average CCT of multiple coflows.
- Through extensive experiments, we can demonstrate that our algorithm PRO can reduce the single CCT by 39.2%, 51.8% compared to reduce tasks placement optimization only algorithm and routing optimization only algorithm, respectively. Our online scheduling algorithm can also outperform FIFO in minimizing average CCT of multiple coflows.

The rest of the paper is organized as follows. Section II is a motivating example and introduces the motivation of our work. Section III depicts the network model and formulates the problem of single coflow optimization. In Section IV, we propose an approximation algorithm PRO to solve the problem and analyze the approximation ratio of the algorithm. In Section V, we discuss the multiple coflows optimization. We propose offline and online multiple coflows scheduling algorithms. In Section VI, we provide the datasets of the topologies and evaluate our proposed algorithm on three network topologies. Section VII introduces the related work. Finally, we make conclusion in Section VIII.

II. A MOTIVATING EXAMPLE

In this section, we introduce the motivation of our work. We begin with a motivating example. As shown in Fig.2, it is a small scale inter-DC WAN network and is composed of three routers. The datacenters are connected through three routers. The numbers in the nodes are the indexes of the datacenters and the numbers in the parentheses are the intermediate data generated by the map tasks. The number on the links are the bandwidth. The OSPF link weights are set to 1.

Assume there is only one coflow and with one reduce task. If we choose datacenter 1 to place the reduce task, then the optimal routes for the intermediate data from datacenter 2 to 1 should be (2,3,1) and for the intermediate data from datacenter 3 to 1 should be (3,2,1). Therefore, we can obtain the CCT is 3s. Similarly, if we choose datacenter 2 to place the reduce task, the routes are (1,2) and (3,2), we can obtain the CCT is 3s. If we choose datacenter 3 to place the reduce task, the routes are (1,2,3) and (2,1,3), we can obtain the CCT is 2.5s. Therefore, by jointly optimizing the reduce task placement and routing, the CCT is 2.5s. If we only optimize the coflow routing and randomly choose datacenter 2 as the placement of reduce task, then CCT can be solved to be 3s according to the similar procedures as mentioned before. If we only optimize the reduce task placement and choose datacenter 3 to place the reduce task, but select the routes according to the shortest path routing protocol (e.g. OSPF), then the CCT that we obtain is 5s.

In this simple example, we can observe that by optimizing both the placement of reduce task and routing, we can reduce the coflow completion time by 50% and 16.7%, compared with only the optimization of placement of reduce task or only the optimization of routing.

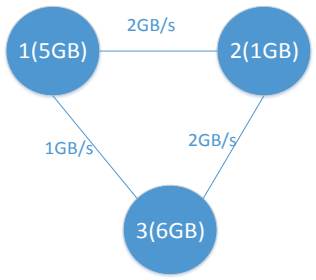


Fig. 2. An example of joint optimization. In this example, there is only one reduce task. The number on the links denotes the bandwidth of the link. The number in the parentheses is the intermediate data that should be transferred to the reduce tasks.

III. SINGLE COFLOW OPTIMIZATION

For simplicity, we first study the problem of minimizing the completion time of a single coflow in inter-DC WAN. In this section, we first formulate the optimization problem of minimizing the single coflow completion time in inter-DC WAN. Then we transform the formulated original Mixed Integer NonLinear Programming (MINLP) problem into a Linear Programming (LP) problem, which can be solved efficiently by LP solver.

A. Network Model

We first describe the inter-DC WAN scenario that we study. As shown in Fig.1, datacenters are connected through inter-DC WAN. The map tasks are placed geo-distributedly in

TABLE I
DEFINITION OF NOTATIONS

$G=(V,E)$	Undirected graph with nodes V and edges E
R_l	Capacity of the edge $l \in E$
D	Datacenter set
π	Reduce task set
C_{ij}	Time to receive all the intermediate data for reduce task i when placed at j datacenter
y_{ij}	A binary variable. $y_{ij}=1$ denotes that reduce task i is placed at datacenter j.
m_{kj}	The volume of the intermediate data transferred between datacenters k and j
b_{kj}	The assigned bandwidth for the flow between datacenters k and j.
x_{kj}^t	A binary variable. $x_{kj}^t=1$ denotes the flow between datacenters k and j is routed through path t.
S_i	The set of input datacenters that transfer flows for reduce task i.
a_j	The maximum number of reduce tasks that can be scheduled on datacenter j.
P_{kj}^t	The t-th path between datacenter k and j.

each datacenter and the intermediate data generated in map phrase is transferred to reduce tasks across the inter-DC WAN to be processed. Suppose that resources of the intra-DC network are sufficient and the computation time in each datacenter can be ignored. The placement of reduce task and the bandwidth across the inter-DC WAN are bottlenecks. We intend to optimize the placement of reduce tasks and routing of a single coflow in the shuffle phrase to minimize CCT.

Now we give a definition of the notations in TABLE I. The Inter-DC WAN can be modeled as a graph $G=(V,E)$, V denotes the backbone router set in inter-DC WAN and E denotes the link set among the routers. $R_l, (l \in E)$ represents the capacity of the link l. Each data center is connected to a router in the inter-DC WAN. We denote the datacenters as $D=\{1,2,..d\}$. Each datacenter is assigned map tasks and some datacenters are assigned reduce tasks. $\pi=\{1,2,..\gamma\}$ is the set of reduce task. We should determine the placement of reduce tasks and the routing of the single coflow among the Inter-DC WAN so that coflow completion time can be minimized. Our optimization problem can be formulated as follows:

$$\textbf{minimize}\{y_{ij} * C_{ij}\} \tag{1}$$

$$\sum_{j=1}^{d} y_{ij} = 1, \quad \forall i \in \pi \tag{1a}$$

$$\sum_{i=1}^{\pi} y_{ij} \le a_j, \quad \forall j \in D \tag{1b}$$

$$C_{ij} = \textbf{max}_{k \in S_i}(m_{kj}/b_{kj}), \quad \forall i \in \pi, \forall j \in D \tag{1c}$$

$$\sum_{k \in S_i} \sum_{j \in D} \sum_{t: l \in P_{kj}^t} b_{kj} x_{kj}^t \le R_l, \quad \forall l \in E \tag{1d}$$

$$\sum_{t} x_{kj}^t = 1, \quad \forall k \in S_i, j \in D \tag{1e}$$

$$x_{kj}^t, y_{ij} \in \{0,1\} \tag{1f}$$

C_{ij} is the transfer time to receive all the intermediate data for reduce task i when placed at j datacenter. y_{ij} is a binary variable. $y_{ij}=1$ denotes that the reduce task i is placed at datacenter j. Our goal is to minimize CCT. Equation (1a) requires reduce task i must be placed at one and only one datacenter. Equation (1b) denotes that the number of reduce tasks that are placed in datacenter i should not exceed the maximum number of reduce tasks that can be scheduled on datacenter i. a_j is the maximum number of reduce tasks that can be scheduled on datacenter j. m_{kj} is the volume of the intermediate data, which is transferred between datacenters k and j. b_{kj} is the assigned bandwidth of the flow between datacenters k and j. Equation (1c) computes the coflow completion time for the reduce task i placed at datacenter j. x_{kj}^t is also a binary variable. $x_{kj}^t=1$ denotes the flow between datacenters k and j is routed through path t, otherwise, $x_{kj}^t=0$. S_i is the set of input datacenters that transfer flows for the reduce task i. Equation (1d) constrains that the consumed bandwidth by the coflow should not exceed the link capacity. Equation (1e) represents that each flow chooses one and only one path between the source and destination. (1f) constrains that x_{kj}^t, y_{ij} are binary variables.

B. Transform into an LP problem

There are integer variables x_{kj}^t, y_{ij} in (1) and they make the problem a mixed integer programming. There is also the product of two variables in the objective function, which makes the problem a nonlinear programming. Therefore, the problem formulated in (1) is a MINLP problem, which is proved to be NP-hard [4]. To make the problem easy to solve, we determine to transform the MINLP problem into an LP problem. We focus on many-to-one coflow communication patterns [5] in our paper, that is, there is only one reduce task for a coflow. We assume placement of reduce task i is determined ahead in datacenter j. Then our problem can be formulated as (2):

$$\textbf{minimize } C_{ij} \tag{2}$$

$$C_{ij} = \textbf{max}_{k \in S_i}(m_{kj}/b_{kj}) \tag{2a}$$

$$\sum_{k \in S_i} \sum_{t: l \in t} b_{kj} x_{kj}^t \le R_l, \quad l \in E, k \in S_i \tag{2b}$$

$$\sum_{t} x_{kj}^t = 1, \quad k \in S_i \tag{2c}$$

$$x_{kj}^t \in \{0,1\} \tag{2d}$$

However, in this formulation, our optimization constraints (2a) and (2d) are still nonlinear. Therefore, problem (2) is

still an MINLP problem, which is NP-hard. Therefore, we choose to transform the problem (2) into a linear programming problem, which can be easily solved. We observe that there are binary variables, so we relax the constraint (2d) to $x_{kj}^t \in [0,1]$. What's more, we notice that equation (2a) is discrete, so we substitute it with $(m_{kj}/b_{kj}) \leq C_{ij}, k \in S_i$. Then our problem can be transformed into the following form:

$$\textbf{minimize } C_{ij} \tag{3}$$

$$m_{kj}/b_{kj} \leq C_{ij}, \quad k \in S_i \tag{3a}$$

$$(2b) - (2c) \tag{3b}$$

$$x_{kj}^t \in [0,1] \tag{3c}$$

In (3), we set $C_{ij} = \frac{1}{\alpha_{ij}}$ and substitute it in (3a), we can obtain that $m_{kj} * \alpha_{ij} \leq b_{kj}$. Then we relax the (2b) and obtain $\sum_{k \in S_i} \sum_t m_{kj} * \alpha_{ij} * x_{kj}^t \leq R_l$. Because there is a product of two variables (α_{ij} and x_{kj}^t) in the inequation, it still makes the problem nonlinear and difficult to solve. Then we set $p_{kj}^t = \alpha_{ij} * x_{kj}^t$ and the problem (3) can be transformed to the following form:

$$\textbf{maximize } \alpha_{ij} \tag{4}$$

$$\sum_{k \in S_i} \sum_t m_{kj} * p_{kj}^t \leq R_l, \quad l \in E, k \in S_i \tag{4a}$$

$$\sum_t p_{kj}^t = \alpha_{ij}, \quad k \in S_i \tag{4b}$$

$$(3c) \tag{4c}$$

Finally, problem (4) that we obtain is a linear programming. We can use the LP solver (e.g., CPLEX, GUROBI) to solve the relaxed LP efficiently and obtain the optimal problem p_{kj}^t, α_{ij}. Because we relax the binary variables $x_{kj}^t \in \{0,1\}$ to $x_{kj}^t \in [0,1]$, the solutions x_{kj}^t we obtain may be fractional and may be infeasible for the original problem (1). Therefore, to make the solution feasible for the original problem, we propose a rounding algorithm. The rounding algorithm we design is as follows: we route the flow from k to j on the path $t^{'}$ (i.e, let $x_{kj}^{t^{'}} = 1$), which satisfies the constraints $p_{kj}^{t^{'}} = \textbf{max}_t p_{kj}^t$. Substituting p_{kj}^t with $\alpha_{ij} * x_{kj}^t$ in (4a), problem (4) turns to a linear programming only with variables α_{ij}. We can solve this problem with LP solver easily and obtain α_{ij}, which is the final feasible solution to our problem. Finally, we can calculate coflow completion time with equation $C_{ij} = 1/\alpha_{ij}$.

IV. Algorithm Description

In this section, we first introduce the deterministic approximate algorithm PRO that we propose to jointly optimize the coflow routing and task placement and explain the algorithm in details. Next, we present the analysis of the approximate algorithm and prove the approximation ratio is $(1+\epsilon)$.

A. Algorithm details

We propose an approximate algorithm PRO (Placement and Routing Optimization) to solve the optimization problem (1). In each iteration, we determine the placement of the reduce tasks that provide the minimum coflow completion time. Then we solve (4) to obtain the routing that minimizes CCT. Algorithm 1 shows the details of PRO.

1) PRO: In Algorithm 1, we first initial CCT (line 1). Then we determine the placement of reduce tasks (line 2-7). We start the algorithm by setting the reduce task in datacenter j (line 2). Then we fix the placement of reduce task and solve the transformed LP problem (4) (line 3). We round the solution to make it feasible for the original problem (line 4). If the C_{ij} computed is less than the current optimal CCT, then we store the position of the reduce task and update the optimal CCT (line 5-7). Therefore, after iteration, we can obtain the placement of the reduce task with the minimum CCT (line 8). Finally, we can obtain the routes of the coflow and CCT after placing the reduce task (line 9-10).

2) Rounding: In Algorithm 2, we introduce Rounding algorithm. We obtain the routes and CCT of the coflow in this algorithm. According to the analysis in Section III, we first round the solution and set the fractional $x_{kj}^{t^{'}}$ to 1 by choosing the t' that satisfies $p_{kj}^{t^{'}} = \textbf{max}_t p_{kj}^t$ (line 1). Then we can obtain the α_{ij} with $x_{kj}^{t^{'}}$ set to 1 (line 2). Finally, we can obtain $C_{ij} = 1/\alpha_{ij}$.

Algorithm 1: PRO

Input: $G = (V, E)$, R_l, D
Output: y_{ij}, C_{ij}, x_{kj}^t

1 $CCT = max\_int$;
2 **foreach** $j \in D$ **do**
3 $\quad p_{kj}^t$ = solve the problem (4);
4 $\quad (C_{ij}, x_{kj}^t)$ = Rounding (p_{kj}^t) ;
5 $\quad$ **if** $C_{ij} < CCT$ **then**
6 $\quad\quad pos = j$;
7 $\quad\quad CCT = C_{ij}$;
8 $y_{ipos} = 1$;
9 p_{kj}^t = solve the problem (4);
10 (C_{ij}, x_{kj}^t) = Rounding (p_{kj}^t) ;
11 return $(y_{ij}, C_{ij}, x_{kj}^t)$;

Algorithm 2: Rounding

Input: p_{kj}^t
Output: (C_{ij}, x_{kj}^t)

1 round the solution $x_{kj}^{t'} = 1$, with $t', p_{kj}^{t^{'}} = \textbf{max}_t p_{kj}^t$;
2 obtain α_{ij} with $x_{kj}^{t^{'}} = 1$ in (4) ;
3 $C_{ij} = 1/\alpha_{ij}$;
4 return (C_{ij}, x_{kj}^t) ;

To introduce our algorithm concretely, we can employ the proposed algorithm to solve the aforementioned example in

Section II. We choose datacenter 3 to place the reduce task and then we optimize the routing of coflow which is composed of flows from 1 to 3 and from 2 to 3. The paths from 1 to 3 is (1,3) and (1,2,3), which are labeled by x_{13}^1, x_{13}^2. The paths from 2 to 3 is (2,1,3) and (2,3), which are labeled by x_{23}^1, x_{23}^2. We have the following equations and inequations:

$$\textbf{maximize } \alpha_j \tag{5}$$

$$5 * p_{13}^1 + p_{23}^1 \leq 1 \tag{6}$$

$$5 * p_{13}^2 \leq 2 \tag{7}$$

$$p_{23}^1 \leq 2 \tag{8}$$

$$5 * p_{13}^2 + p_{23}^2 \leq 2 \tag{9}$$

$$p_{13}^1 + p_{13}^2 = \alpha_j \tag{10}$$

$$p_{23}^1 + p_{23}^2 = \alpha_j \tag{11}$$

We can solve this linear programming and obtain the optimal solution $p_{13}^1 = 0.1, p_{13}^2 = 0.4, p_{23}^1 = 0.5, p_{23}^2 = 0$. We then round the solution: $p_{13}^2 = \textbf{max}\{p_{13}^1, p_{13}^2\}$, therefore, we can obtain $x_{13}^1 = 0, x_{13}^2 = 1$. Similarly, we can obtain $x_{23}^1 = 1, x_{23}^2 = 0$. Then we can substitute the value of X into problem (4) and obtain $\alpha_{13} = 0.4$, $b_{13} = 0.4 * 5 = 2, b_{23} = 0.4 * 1 = 0.4$, then $C_{13} = 1/\alpha_{13} = 2.5$s.

B. Approximation ratio analysis

C_R denotes the coflow completion time of our proposed algorithm PRO and C_O denotes the coflow completion time of the original problem (1). Then we give an analysis of approximation ratio of our algorithm in Theorem 1 and 2 .

Theorem 1. $C_O \leq C_R$.

Proof. In our algorithm, we relax the original problem and then round the solution. So our solution is only a feasible solution for the original problem. Therefore, $C_O \leq C_R$. □

Theorem 2. $C_R \leq (1+\epsilon) * C_O, \forall \epsilon > 0$.

Proof. We assume the objective value of problem (4) is α_{upper}. Because problem (4) is a relaxation of problem (1), we can easily conclude that:

$$\alpha_{upper} \geq \alpha_O \tag{12}$$

In Algorithm 2, we route the flow on the path k with maximum $p_{kj}^t = \alpha_{upper} * x_{kj}^t$, $\forall \epsilon > 0$, therefore, we can obtain:

$$p_{kj}^t > \frac{\alpha_{upper}}{1+\epsilon} * x_{kj}^t \tag{13}$$

Substituting (13) with p_{kj}^t in (4a), we can have:

$$\sum_{k \in S_i} \sum_j \sum_t m_{kj} * \frac{\alpha_{upper}}{1+\epsilon} * x_{kj}^t < \sum_{k \in S_i} \sum_j \sum_t m_{kj} * p_{kj}^t \leq R_l \tag{14}$$

Therefore, $\frac{\alpha_{upper}}{1+\epsilon}$ is a feasible solution to problem (4) with given rounded x_{kj}^t. α_R is the optimal solution to problem (4) with given rounded x_{kj}^t. We can obtain:

$$\alpha_R \geq \frac{\alpha_{upper}}{1+\epsilon} \geq \frac{\alpha_O}{1+\epsilon} \tag{15}$$

Finally, we can obtain $\forall \epsilon > 0$:

$$C_R \leq (1+\epsilon) * C_O \tag{16}$$

□

By previous analysis, the approximation ratio of our proposed algorithm is $(1+\epsilon)$. Therefore, PRO is a $(1+\epsilon)$-approximation algorithm.

V. Multiple Coflows optimization

In this section, we consider to optimize reduce tasks placement and routing for multiple coflows $F = \{f_i, i \in [1, r]\}, |F| = r$. We need to determine the routing and reduce task placement of each coflow and the scheduling between the inter-coflows to minimize the average CCT of multiple coflows. We first present an offline algorithm for multiple coflows optimization. Then we transform the offline scheduling algorithm into an online scheduling algorithm.

A. Offline multiple coflows optimization

Given that the shortest-first policy (SFP) is optimal [6], we would like to schedule the coflows with the shortest completion time first. Therefore, we propose the following PROS (Placement and Routing Optimization of coflowS) algorithm (Algorithm 3), which is optimal, to minimize the average CCT.

Algorithm 3: PROS

Input: $G = (V, E)$, R, D, N
Output: y_{ij}, x_{kj}^t
1 **foreach** $i \in r$ **do**
2 Obtain CCT C_{f_i} of f_i by Algorithm PRO ;
3 Sort the coflows in increasing order of C_{f_i} and obtain F^*;
4 **foreach** $f_i \in F*$ **do**
5 Obtain x_{kj}^t and y_{ij} of f_i by Algorithm PRO ;
6 schedule coflow f_i ;

In this algorithm, we first calculate the CCT with Algorithm 1 (line 1-2) by optimizing the reduce tasks placement and routing. Then we sort the coflows in increasing order of CCT and obtain the sorted coflows F^* (line 3). Finally, we schedule the coflows according to the order of coflows in the sorted F^* to minimize the average CCT (line 4-6). According to shortest coflow first policy, our scheduling algorithm among multiple coflows is optimal.

B. Online multiple coflows optimization

In section V-A, we propose an offline multiple coflow optimization framework. Then in this part, we consider to transform the offline coflows scheduling to online coflows scheduling. Because of non-negligible signaling overhead and performance loss of preemptive scheduling [7], we assume the scheduling of the coflows is non-preemptive in our approach. In non-preemptive scheduling, the bandwidth and routes for coflows stay unchanged when being transferred. The coflow once starts transferring cannot be preempted until finishes. To avoid starvation, we schedule the coflow which waits for more than a threshold Γ when a current coflow finishes. The online algorithms are depicted in details in Algorithm 4:

Algorithm 4: Online non-preemptive scheduling

```
1  while a new coflow arrives or a current coflow finishes do
2      if a new coflow f_n arrives then
3          Compute the CCT of f_n through PRO ;
4          Insert f_n into unscheduled coflow set I ;
5      if the current coflow finishes then
6          waited_time_expire = false ;
7          foreach f_u ∈ I do
8              if waiting time of f_u > Γ then
9                  schedule f_u ;
10                 wait_time_expire = true ;
11                 break;
12         if wait_time_expire == false then
13             schedule the coflow with the minimum CCT
               in set I ;
```

In Algorithm 4, we wake up our scheduling when a new coflow arrives or a current coflow finishes (line 1). If a new coflow arrives, then we minimize the CCT through algorithm PRO and insert the coflow into the unscheduled coflow set I (line 2-4). If the current coflow finishes, we check if there is a coflow with waiting time expiring a threshold Γ (line 8). If exists, then we schedule this coflow immediately (line 9-11), else we schedule the coflow with the minimum CCT in unscheduled coflow set I (line 12-13).

Theorem 3. *The online non-preemptive multiple coflows optimization algorithm has a competitive ratio of* $(n+\epsilon)$, $\forall \epsilon > 0, n$ *is coflow numbers.*

Proof. Suppose the process time of each coflow is $t_1, t_2, ..., t_n$, which is in increasing order. In offline optimal scheduling, the shortest coflow first policy is optimal. Therefore, the average CCT in offline scheduling C_{off} is computed as follows:

$$\begin{aligned} C_{off} &= [t_1 + (t_1 + t_2) + ... + (t_1 + t_2 + ... + t_n)]/n \\ &= [n * t_1 + (n-1) * t_2 + ... + t_n]/n \end{aligned} \tag{17}$$

While, in our online scheduling, the worst case is that coflows arrive in decreasing order of CCT and the interval of coflow arrival time is little larger than the CCT of coflows. Then there are no coflows in unscheduled list and the coflow is processed as soon as it arrives. The average CCT in online scheduling C_{on} is computed as follows:

$$\begin{aligned} C_{on} &= [t_n + (t_n + t_{n-1}) + ... + (t_n + ... + t_1)]/n \\ &= [n * t_n + (n-1) * t_{n-1} + ... + t_1]/n \end{aligned} \tag{18}$$

$$\frac{C_{on}}{C_{off}} = \frac{[n * t_n + (n-1) * t_{n-1} + ... + t_1]}{[n * t_1 + (n-1) * t_2 + ... + t_n]} \tag{19}$$

When t_n is very large, then

$$\lim_{t_n \to \infty} \frac{C_{on}}{C_{off}} = \frac{n * t_n}{t_n} = n \tag{20}$$

Therefore, $\forall \epsilon > 0$,

$$\frac{C_{on}}{C_{off}} \leq (n + \epsilon) \tag{21}$$

Then we can conclude the competitive ratio of the online coflow optimization is $(n+\epsilon), \forall \epsilon > 0$.

□

VI. Evaluation

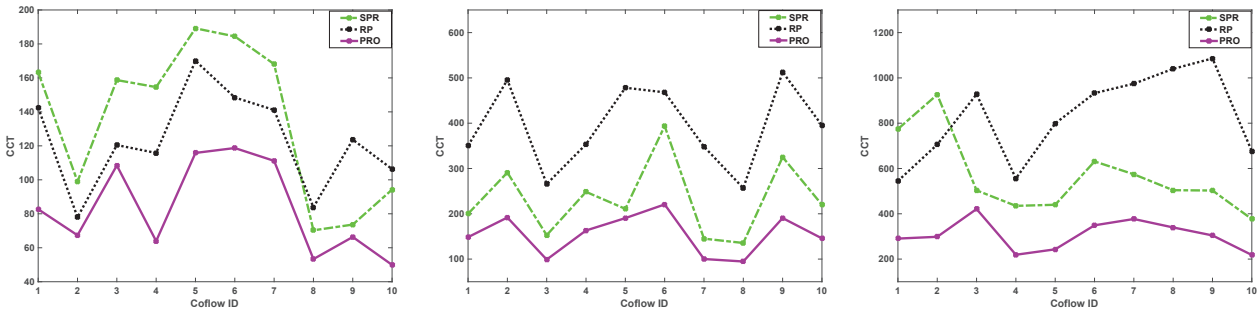
In this section, we carry on the experiments evaluation. Through simulation with different topologies, we can obtain that the single coflow completion time can be better reduced with the joint optimization of task placement and routing, which performs better than placement optimization only (SPR) and routing optimization only (RP) algorithms. The average CCT can be better reduced with online scheduling algorithm than FIFO.

A. Evaluation setting

In the simulation, we develop a simulator written in C++. We conduct our experiments on one production inter-DC WAN: an inter-DC WAN testbed with 5 DCs and 14 inter-DC links [8] and two synthetic topologies, which are generated by BRITE. The detailed information of the topologies is shown in TABLE II. We employ CPLEX to solve the transformed LP problem in our experiments. The intermediate data sizes that should be transferred are randomly generated. The bandwidths of all the inter-DC WAN links are set to 100Mbps [1]. The experiments runs on a personal computer with CPU i5-3320M 2.6GHz and 4GB of RAM. We compare our algorithm with the following algorithms.

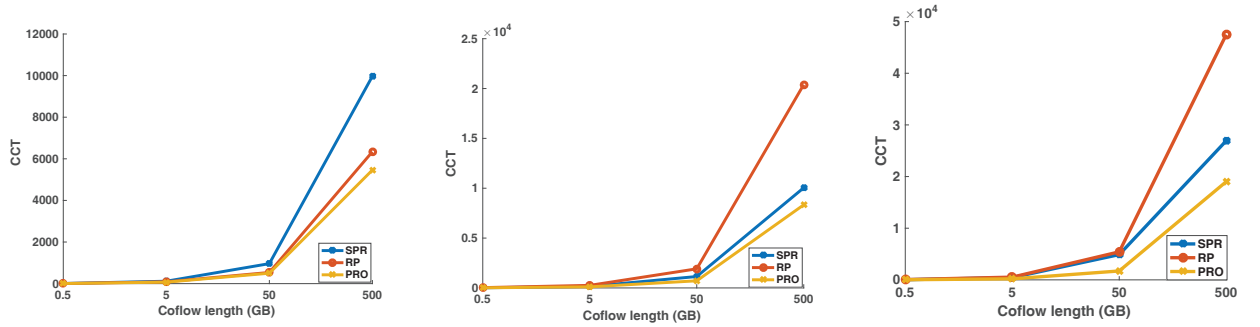
Random Placement (RP): The reduce task is located at a random datacenter and the routing is computed by the algorithm in [9]. In this method, we only optimize the coflow routing.

Shortest Path Routing (SPR): The reduce task is selected by approximation algorithm in Algorithm 1, while the routing of the coflow is based on k-shortest paths routing. In this method, we only optimize the task placement.



(a) IDN (b) Synthetic 1 (c) Synthetic 2

Fig. 3. CCT for different single coflow. We can obtain that the CCT can be reduced greatly by a joint optimization of task placement and routing in different topologies.



(a) IDN (b) Synthetic 1 (c) Synthetic 2

Fig. 4. CCT under various coflow lengths. We can obtain that under different coflow length, our proposed PRO can obtain a lower CCT compared to RP and SPR.

TABLE II
THREE NETWORK TOPOLOGIES

Topologies	Nodes	Links
IDN	5	14
Synthetic 1	10	11
Synthetic 2	20	21

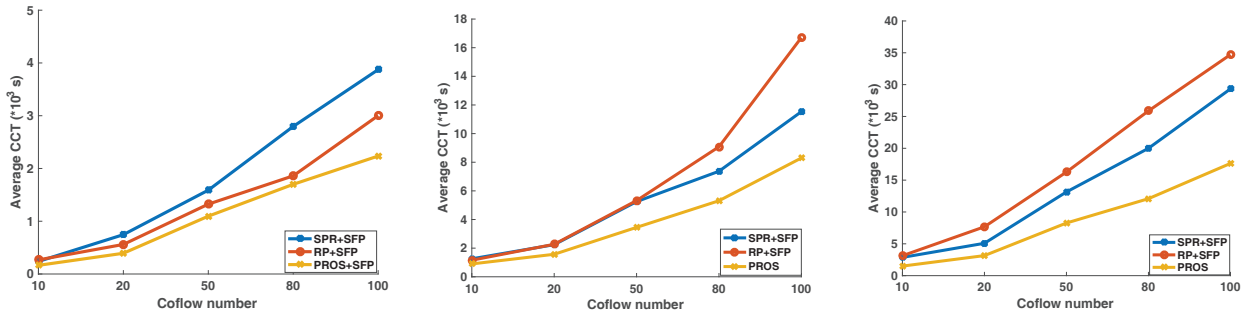
TABLE III
AVERAGE PERFORMANCE UNDER THREE ALGORITHMS

Topologies	SPR (s)	RP (s)	PRO (s)
IDN	135.55	123.02	83.77 (38.2%,31.9%)
Synthetic 1	232.28	392.47	154.44 (33.5%,60.6%)
Synthetic 2	566.80	824.28	306.05 (46.0%,62.9%)

B. Simulation

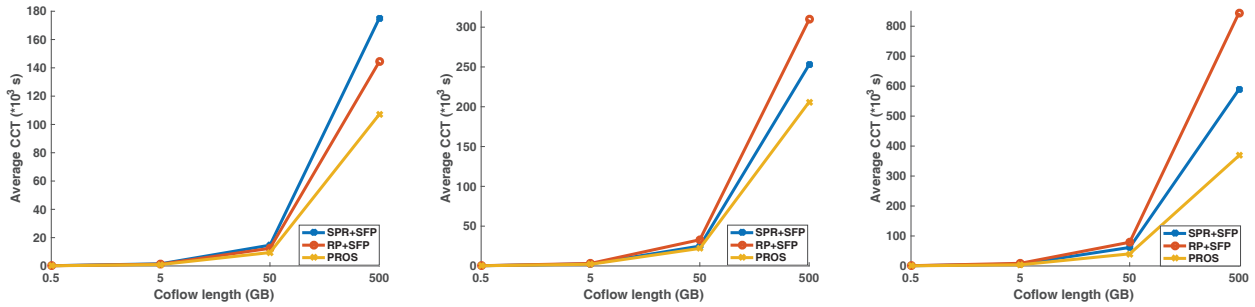
1) Single coflow: We first exploit PRO algorithm to jointly optimize the task placement and routing of a single coflow to reduce CCT. We plot that CCT varies with different coflows under different algorithms in three inter-DC WAN topologies. As shown in Fig.3, we can find that under various intermediate data sizes, our proposed algorithm PRO obtains a lower CCT by jointly optimize the routing and placement of reduce tasks compared with RP and SRP. We can also observe that in Fig.3a, RP outperforms SPR; in Fig.3b and Fig.3c, SPR outperforms RP. Because in IDN, there are five DCs and the scale of the inter-DC WAN is small. Therefore, there is higher possibility to choose the optimal DC for the reduce task in random algorithm. Therefore, in Fig.3a, RP outperforms SPR.

To make more concrete about improvement ratio of CCT in our algorithm, we compute the improvement ratio of average CCT in three different algorithms in TABLE III. As shown in TABLE III, we can obtain that the CCT can be reduced by 38.2% and 31.9% compared to SPR and RP in IDN, by 33.5% and 60.6% compared to SPR and RP in synthetic 1, by 46.0% and 62.9% compared to SPR and RP in synthetic 2. The CCT can be better minimized with the optimization of task placement only than coflow routing only. We can conclude that task placement contributes more to reducing CCT than coflow routing.



(a) IDN (b) Synthetic 1 (c) Synthetic 2

Fig. 5. Average CCT under various number of coflows. We can conclude that our proposed PROS can obtain a lower average CCT compared to RP with shortest first policy inter-coflow scheduling (RP+SFP) and SPR with shortest first policy inter-coflow scheduling (SPR+SFP).



(a) IDN (b) Synthetic 1 (c) Synthetic 2

Fig. 6. Average CCT under various lengths of coflows.

Then we evaluate CCT under various coflow lengths. The coflow length is the size of the largest flow in this coflow. We set the coflow length to 0.5GB, 5GB, 50GB, 500GB, respectively and draw the curves that CCT varies with coflow lengths in Fig.4. As shown in Fig.4, we can observe that our proposed algorithm PRO can obtain a lower CCT compared to SPR and RP.

2) Multiple coflows: Then we evaluate the average CCT under various number and lengths of multiple coflows in offline scheduling PROS. In PROS, multiple coflows arrive at the inter-DC WAN simultaneously. We employ different algorithms: PROS, RP with shortest first policy inter-coflow scheduling RP+SFP, SPR with shortest first policy inter-coflow scheduling SPR+SFP to minimize the average CCT.

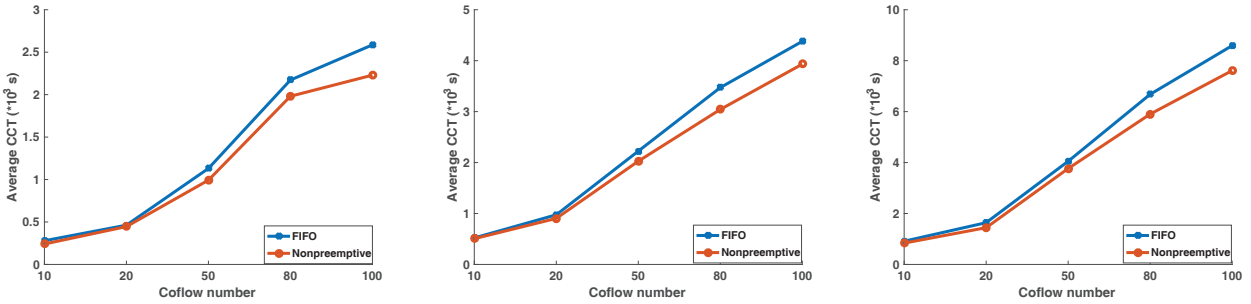
We draw the curves that average CCT varies with the number of coflows in Fig.5. We can observe that with the increasing of coflow number, average CCT increases. PROS can obtain a lower average CCT compared to SPR+SFP and RP+SFP. Moreover, with the increasing of number of coflows, the improvement ratio of PROS compared with other algorithms is increasing. Therefore, our algorithm PROS outperforms SPR+SFP and RP+SFP with multiple coflows scheduling.

Then in Fig.6, we show the curves for the evaluation of average CCT on various coflow lengths. We can observe that with the increasing of coflow length, average CCT increases. PROS can obtain a lower average CCT compared to SPR+SFP and RP+SFP.

From the previous simulation and analysis, we can conclude that by joint optimization of routing and task placement, our proposed algorithm can outperform the routing only and task placement only algorithms in offline scheduling algorithm.

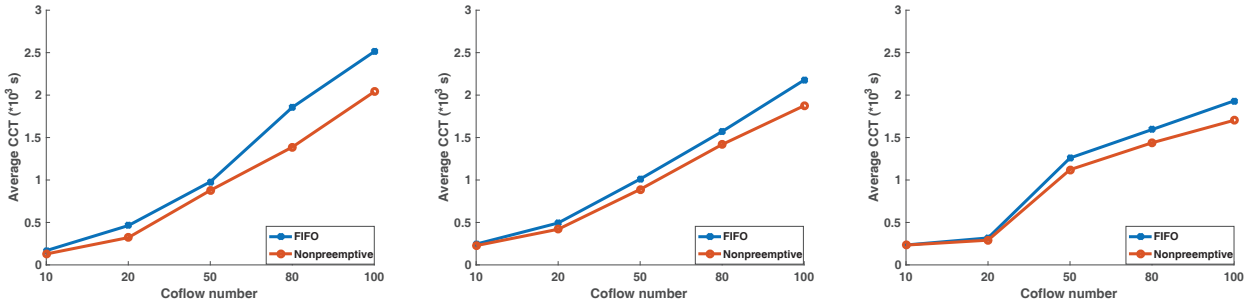
Moreover, we compare our proposed online scheduling of multiple coflows with First In First Out (FIFO) scheduling. We set that the coflows arrive every 10 seconds and are distributed evenly. The coflow length is set to 5GB. We plot the curves that average CCT varies with different number of coflows in Fig.7. As shown in Fig.7, we can observe that average CCT is increasing with the increasing of coflow numbers. Then in TABLE IV, we compute the improvement ratio of our online scheduling algorithm under different coflow number. We can obtain that our algorithm can reduce the average CCT by $11.86\%, 9.22\%, 11.24\%$ on average under different topologies. Our non-preemptive online scheduling of multiple coflows performs better than FIFO scheduling in minimizing the average CCT.

In Fig.8, we observe the average CCT when coflow arrival interval is set to 1s, 5s and 20s, respectively. Our online scheduling algorithm performs better than FIFO with different coflow arrival intervals. When the interval becomes larger,



(a) IDN (b) Synthetic 1 (c) Synthetic 2

Fig. 7. Online scheduling of multiple coflows. Coflows is evenly distributed and coflow interval is set to 10s.



(a) Interval=1s (b) Interval=5s (c) Interval=20s

Fig. 8. Online scheduling of multiple coflows in IDN with different coflow arrival intervals.

TABLE IV
AVERAGE CCT IMPROVEMENT UNDER DIFFERENT COFLOW NUMBER

Coflow Number	10	20	50	80	100	Average
IDN	15.9%	3.8%	13.9%	9.7%	16%	11.86%
Synthetic 1	3.0%	7.8%	9.8%	14.1%	11.4%	9.22%
Synthetic 2	8.9%	13.6%	7.6%	13.2%	12.9%	11.24%

the gap between the two algorithms becomes smaller. For if the interval is very large, there will leave no coflows in the unscheduled list in our algorithm and every coflows will be scheduled after its arrival, which approaches the performance of FIFO.

VII. RELATED WORK

There are numerous works on coflows transfer optimization. In this section, we briefly present the most related work on coflow scheduling, coflow routing and tasks placement, respectively.

Coflow scheduling: Chowdhury et.al first introduce the concept of coflow and propose an architecture Orchestra to optimize the scheduling of coflow in datacenter network in [2]. They also propose Varys [10] and Aalo [11] to minimize the CCT and guarantee the coflow deadlines with efficient coflow scheduling algorithms. In [12], Qiu et.al minimize the total weighted completion time of coflows in datacenter networks. They are the first to propose polynomial deterministic algorithm for the coflow scheduling problem with an approximation ratio of $\frac{67}{3}$. In [13], to provide differential treatment to coflows with different degrees of sensitivity, Chen et.al introduce a new utility optimal scheduler among competing coflows and satisfy the max-min fairness among these coflows. In [14], Fahad R. Dogar et.al design and implement a decentralized coflow scheduling system Baraat, which schedules the coflows in a FIFO order but dynamically changes the order of scheduling when heavy tasks are encountered. In [15], Zhang et.al first propose CODA to automatically identify coflows with machine learning techniques and propose an error-tolerant scheduler that can tolerate occasional identification errors.

Coflow routing: In [9], Zhao et.al optimize the average CCT in intra-DC network by integrating routing and scheduling. In coflow scheduling, they choose to schedule the coflow with Minimum Remaining Time First (MRTF) to minimize the average CCT. In coflow routing, they choose to formulate the path selection problem into an integer multi-commodity flow problem and propose a heuristic algorithm to solve it. In [16], Li et.al propose an online algorithm OMCoflow to optimize the routing and scheduling of coflows, which avoids

the problem of frequently rerouting in [9]. The theoretical analysis shows that OMCoflow can obtain a good competitive ratio in minimizing the average CCT. In [3], Zhang et.al optimize the transfer routing and scheduling to guarantee the transfer deadline. As to the routing of coflows, they employ the K-shortest paths routing.

Task placement: [17] is the first paper to achieve low query response times in inter-DC WAN by optimizing the data and reduce tasks placement. They assume the bottlenecks are the bandwidth of uplinks and downlinks between the geo-distributed datacenter sites and inter-DC WAN core. They employ an efficient linear formulation to solve the problem of task placement. In [1], Hu et.al are the first to optimize the reduce tasks placement to minimize the coflow completion time in inter-DC WAN. They propose Flutter, an online network-aware and stage-aware scheduling algorithm, to determine the placement of reduce tasks in the inter-DC WAN. The problem is formulated as an integer linear programming (ILP) and transformed to a linear programming (LP) problem, which can be solved efficiently with LP solvers. The CCT can be reduced by up to 25% in Flutter. Li et.al [18] propose to minimize the traffic generated by geo-distributed big data analytics transferred among the inter-DC WAN by jointly optimizing input data movement and task placement. They employ chance-constrained optimization techniques to guarantee the job completion time with high probability.

The routing and reduce task placement both influence the coflow completion time. However, there are no previous works that jointly optimize the coflow routing and task placement in inter-DC WAN. Therefore, in this paper, we consider the joint optimization of these two factors simultaneously.

VIII. Conclusion

The coflow completion time greatly influences the performance of big data analytics. Previous studies optimize either only the routing or only the reduce task placement. In this paper, we first propose an algorithm PRO to reduce a single coflow completion time by jointly optimizing the task placement and coflow routing. In our proposed algorithm, we first greedily search a placement of reduce task, then we solve the relaxed LP problem to determine the routing of a coflow. Compared with the algorithms that only optimize the task placement and routing, our proposed algorithm PRO can reduce the CCT by 39.2%, 51.8% on average, respectively. Then we propose an online non-preemptive scheduling algorithm to optimize multiple coflows. Our algorithm can reduce the average CCT by $11.86\%, 9.22\%, 11.24\%$ on average compared with FIFO under different topologies.

In future research, we consider to implement our proposed algorithm in a real map-reduce framework and consider the weighted CCT as our optimization objective.

Acknowledgment

This work is partially supported by the National High Technology Research and Development Program of China (863 Program) No. 2015AA016105 and 2015AA015603, the National Natural Science Foundation of China (Grant No. 61202357), and the Project for 2012 Next Generation Internet technology research and development, industrialization, and large scale commercial application of China (No. 2012 1763).

References

[1] Z. Hu, B. Li, and J. Luo, "Flutter: Scheduling tasks closer to data across geo-distributed datacenters," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[2] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," *Acm Sigcomm Computer Communication Review*, vol. 41, no. 4, pp. 98–109, 2011.

[3] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang, "Guaranteeing deadlines for inter-datacenter transfers," in *Proceedings of the Tenth European Conference on Computer Systems*, 2016, pp. 1–14.

[4] T. T. Cormen, C. E. Leiserson, and R. L. Rivest, "Introduction to algorithms," *Resonance*, vol. 1, no. 9, pp. 14–24, 2009.

[5] H. Susanto, H. Jin, and K. Chen, "Stream: Decentralized opportunistic inter-coflow scheduling for datacenter networks," in *Network Protocols (ICNP), 2016 IEEE 24th International Conference on*. IEEE, 2016, pp. 1–10.

[6] C.-Y. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 127–138, 2012.

[7] R. Yu, G. Xue, X. Zhang, and J. Tang, "Non-preemptive coflow scheduling and routing," in *Global Communications Conference (GLOBECOM), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[8] C. Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," *Computer Communication Review*, vol. 43, no. 4, pp. 15–26, 2013.

[9] Y. Zhao, K. Chen, W. Bai, and M. Yu, "Rapier: Integrating routing and scheduling for coflow-aware data center networks," in *IEEE Conference on Computer Communications*, 2015, pp. 424–432.

[10] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in *ACM Conference on SIGCOMM*, 2014, pp. 443–454.

[11] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," *Acm Sigcomm Computer Communication Review*, vol. 45, no. 5, pp. 393–406, 2015.

[12] Z. Qiu, C. Stein, and Y. Zhong, "Minimizing the total weighted completion time of coflows in datacenter networks," in *Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures*. ACM, 2015, pp. 294–303.

[13] L. Chen, W. Cui, B. Li, and B. Li, "Optimizing coflow completion times with utility max-min fairness," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[14] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, "Decentralized task-aware scheduling for data center networks," *Acm Sigcomm Computer Communication Review*, vol. 44, no. 4, pp. 431–442, 2014.

[15] H. Zhang, L. Chen, B. Yi, K. Chen, M. Chowdhury, and Y. Geng, "Coda: Toward automatically identifying and scheduling coflows in the dark," in *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 2016, pp. 160–173.

[16] Y. Li, H. C. Jiang, H. Tan, C. Zhang, G. Chen, J. Zhou, and F. C. M. Lau, "Efficient online coflow routing and scheduling," in *ACM International Symposium on Mobile Ad Hoc NETWORKING and Computing*, 2016, pp. 161–170.

[17] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," *Acm Sigcomm Computer Communication Review*, vol. 45, no. 5, pp. 421–434, 2015.

[18] P. Li, S. Guo, T. Miyazaki, X. Liao, H. Jin, A. Zomaya, and K. Wang, "Traffic-aware geo-distributed big data analytics with predictable job completion time," *IEEE Transactions on Parallel and Distributed Systems*, 2016.