

Joint Coflow Optimization for Data Center Networks

ZHAOXI WU 

Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China
School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China
University of Chinese Academy of Science, Beijing 100049, China

e-mail: wuzhx@shanghaitech.edu.cn

ABSTRACT Data parallel applications in data centers generate, process, and store huge volumes of data. Coflow Completion time (CCT) is one of the major performance metrics to capture application-level semantics. This paper is the first one to study the joint consideration of task placement, coflow bandwidth scheduling, and path choice to minimize the average CCT in intra-data center. This paper proposes a joint online scheduling framework, which first develops a 2-approximation algorithm to reduce the CCT of a single coflow, and then follows the Shortest Remaining Time First (SRTF) principle to schedule multiple coflows. Extensive simulations based on practical trace demonstrate that the proposed framework has better performance than the state-of-the-art works.


INDEX TERMS Data center, coflow scheduling, task placement, path choice.

I. INTRODUCTION

Nowadays, large volumes of data from games, online video, data mining, scientific calculation, will be generated and processed in data-parallel frameworks such as MapReduce [1], Pregel [2], and Dryad [3]. A major feature of data parallel applications is that a collection of flows, termed coflow [4], will be generated to transfer the intermediate data between subsequent computation stages. A coflow in a data transfer phase will not finish only until all its flows have completed [5], [6].

However, a coflow's completion time (CCT) can cost more than 50 percent of job completion time [7]. As the difference between storage devices and computation speed are witnessed to get greater [8], flow transmission is more likely to become the performance bottleneck for a task. As in some previous works [9] and [10], this paper focuses on the data transfer phase of each job without considering the computation phase.

The existing works on minimizing the average CCT have focused on either task placement or coflow bandwidth scheduling. For example, in [11] and [12], in order to use the data locality and reduce the amount of data transfer, the scheduler places tasks close to their input data. The papers [9], [10] are another example, in which the scheduler

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Ching Ying .

changes priority and flow sending rate to minimize the average CCT with the endpoints of flows prefixed.

Fig. 1 shows the non-blocking model of data center networks used by most existing works. In this model, flows come from host incoming links and finally goes out from host outgoing links. The model ignores the inner links of data center networks and bandwidth limitation only occurs on ingress ports and egress ports. However, the bandwidth on the inner network are limited and can vary significantly across different paths [13]. Moreover, coflows in the network might concurrently compete for the transmission resources [14].

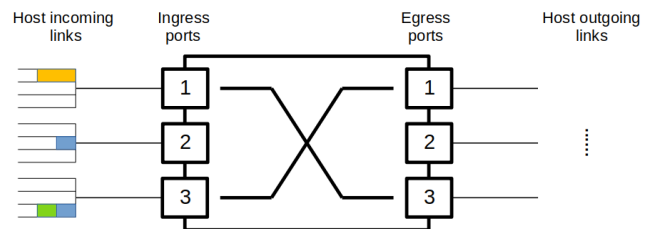


FIGURE 1. Non-blocking model of data center networks.

Thus, Fig. 2 shows the data center network model in this paper: the path in the inner network is aware, which is practical [13]. Thus in this model, the bandwidth limitation on each path should also be considered.

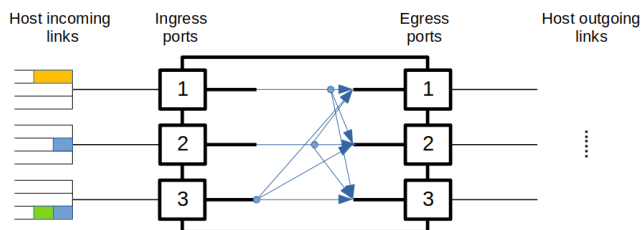


FIGURE 2. Inner network path-aware model in this paper.

As for the problem studied in this paper, Fig. 3 demonstrates the major part. When a coflow comes with a few of flows in it (In this figure, a coflow arrives with 4 flows inside), each flow should be scheduled to a corresponding task through the inner path-aware network shown in the above Fig. 2, and each task should be placed on a host. This model can be formulated as a large scale, integer, non-convex problem and proved to be NP-Hard [14], [15].

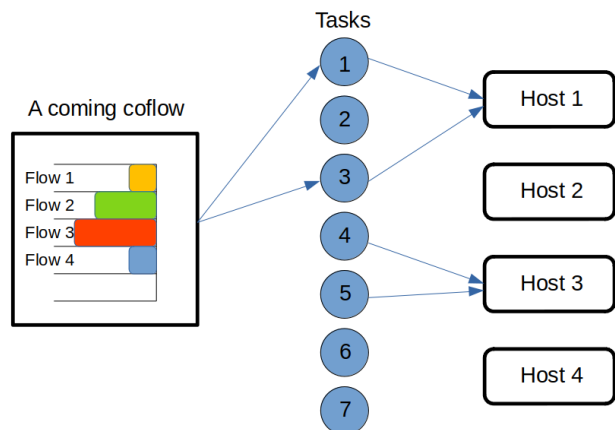


FIGURE 3. The major part of the problem studied in this paper.

This paper proposes a joint online task placement, coflow bandwidth scheduling, and path choice framework, to minimize the average CCT. The framework consists a 2-approximation algorithm for each single coflow and then follows the Shortest Remaining Time First (SRTF) principle to schedule multiple coflows by giving the highest priority to the coflow with the shortest remaining time. The idea to use SRTF principle as the scheduling mechanism is inspired by [6], [16].

In summary, the main contributions are as follows:

- This paper is the first one to study the problem of minimizing the average CCT via jointly considering task placement, coflow scheduling, and path choice in intra-data center networks.
- This paper proposes an online coflow-aware optimization framework to solve the problem.
- This paper presents theoretical analysis to demonstrate that the proposed algorithms can achieve a good competitive ratio.

- Extensive trace-driven simulations evaluate the performance of the proposed framework, in terms of average CCT, algorithm run time, and impacts of 4 coflow characteristics.

A. RELATED WORKS

There are a few of related works on coflow scheduling and task placement for intra-data centers.

1) COFLOW SCHEDULING

Early works are Orchestra [7], Baraat [17], and Varys [9], which use the concept of the coflow and propose heuristic methods to minimize the average CCT and meet deadlines in data centers. Qiu *et al.* [18] solves the problem of multiple coflow offline scheduling by designing a deterministic algorithm with a constant approximation ratio. RAPIER [6] simultaneously combining coflow routing and scheduling by a heuristic method, but lacks theoretical performance guarantees. Chowdhury and Stoica [10] uses a multi-level feedback queues (MLFQ) model and proposes an online algorithm to schedule coflows. After that Liu *et al.* [19] identifies the bottleneck flow in Alao using coflow information-agnostic model. DeepAalo [20] updates the thresholds of queues in Alao by deep reinforcement learning. Dogar *et al.* [17], Luo *et al.* [21], and D-CAS [22] study coflow-aware scheduling scheme in a distributed manner. Further, CODA [23] proposes an algorithm of automatically assigning flows to coflows before scheduling. BlindFlow [24] proposes an online algorithm without knowing flow demand volumes when they arrive. Chen *et al.* [25] and Wu and Fu [26] study the problem of achieving lexicographical max-min fairness among multiple jobs' utilities, which is a large-scale, nonlinear, integer and multi-objective problem. They take the data center network as a big non-blocking switch. All the above works assume that task placement has been fixed.

2) TASK PLACEMENT

To have a larger part of intermediate data and relatively high link bandwidth to reduce the job completion time, Iridium [27] and Flutter [28] place tasks close to data centers. They use linear objective functions which limits the scope of their algorithms. For each job, Sinbad [5] considers one computing stage at a time and places the output data flexibly. CLARINET [29] uses multiple iterations to schedule network flow and task placement separately. Corral [30], NEAT [31], and 2D-Placement [32] adopts joint optimization of input data and task placement to balance between the network contention and workload transmission. ShuffleWatcher [33] places both map and reduce tasks on the same set of racks to improve the locality and reduce cross-rack shuffling without considering bandwidth scheduling.

The most related works are [14], [34]–[38]. References [14], [34]–[36] jointly consider two of coflow scheduling, routing and endpoint placement in the environment of inter-data center networks to minimize the average CCT. They first propose an online scheduling

algorithm for a single coflow and prove its approximation ratio, then extend it to adopt multiple coflows situation. However, for example, SmartCoflow [35] is for inter-datacenter networks, the problem size can be smaller than this paper, which is for intra-datacenter networks. And SmartCoflow limits that each task can only be put on one datacenter, while this paper doesn't limit each task to be assigned to one host. Moreover, SmartCoflow proposes scheduling policy for single coflow and uses it in the multiple coflow situation. This paper sorts the coflows by remaining time and always deals with the shortest one. References [37], [38] jointly considers reducer placement and coflow bandwidth scheduling, which divides a task into mapper and reducer and focuses on reducer part. They regard data center network as a non-blocking switch, so that the network bottleneck exists only on the incoming and outgoing links. In this paper, a coflow can be assigned to multiple tasks and inner network bandwidth limitation is also assumed.

Compared with previous works, this paper involved three factors (coflow scheduling, task placement, and path choice) to optimize CCT, while prior works study one or two of these factors. It's argued that more factors do not lead to smarter algorithms, but may result in more efficient algorithms.

The rest of the paper is organized as follows: Section II describes the system model. Section III describes the algorithm details and proofs. Section IV describes the simulation setup and results. And Section V concludes the paper.

II. SYSTEM MODEL

To minimize the average CCT, this paper abstracts the network as a giant blocking switch that connects physical hosts by available paths, which means that the inner paths between coflow and tasks would experience congestion and it is different from the other classic non-blocking model [9], [10], [34], [39], [40]. Coflows arrive at the intra-data center network and send flows to tasks along some chosen path, while tasks need to be placed on some chosen host. Note that, physical hosts in a data center network can be heterogeneous and available bandwidth on each path can be different because some of the bandwidth can be occupied by other applications [14]. Flow size and flow length can be extracted from the log and meta-data files as coflow information [41], [42].

Suppose there are $|H|$ hosts and $|J|$ tasks in the data center network and each task can be placed on any one of the hosts. It is a general assumption which needs every host has plenty of resources to hold tasks. However, in practice for example, RPC [37] treats small coflow and latency-insensitive individual flows as background traffic, thus it makes sure that hosts have enough resources to orchestrate all the coflows they desire.

The set of unfinished coflows and the set of available path are denoted as C and P respectively. T_i represents the CCT of coflow i and K_i represents the set of flows belonging to coflow i . v_k^i indicates the amount of data that should be

transferred by the k^{th} flow of coflow i , and D_j^i indicates the set of flows in coflow i that needs to be fetched by j^{th} task.

Important notations are listed in Table 1.

TABLE 1. Notations and definitions.

Notation	Definition
C	the set of unfinished coflows
J	the set of tasks
H	the set of hosts
P	the set of available path
T^i	the CCT of coflow i
K_i	the set of flows belonging to coflow i
v_k^i	the amount of data that should be transferred by the k^{th} flow of coflow i
D_j^i	the set of flows in coflow i that needs to be fetched by j^{th} task
$r_{khl}^i(t)$	the rate of sending the k^{th} flow in coflow i to host h along path l at time t
$b_{hl}(t)$	the bandwidth limitation of path l to host h at time t
x_{jhl}^i	a binary variable to denote if task j^{th} is placed onto host h and coflow i sends data to this task along path l

III. ALGORITHM DETAILS

The overview of the proposed joint optimization framework is presented in Algorithm 1, where the available bandwidth set B is assumed to be known in advance [38] and [39].

At line 5 the algorithm calculates the minimum completion time and the corresponding task placement, bandwidth scheduling, and path choice for each single coflow, which is proved to be 2-approximation. The while loop in the algorithm uses Shortest Remaining Time First (SRTF) with the unfinished coflow set C , so that the algorithm is invoked either when a coflow comes or finishes.

For SRTF principle, at line 6-8, the algorithm schedules the coflows that wait longer than the preset threshold thd . And at line 9-11, the algorithm selects the coflow with the shortest remaining completion time. At line 14, the algorithm schedules the selected coflow and updates corresponding parameters.

Note that, the major part in Algorithm 1 is at line 5: the completion time of a single coflow should be minimized while jointly considering task placement, coflow bandwidth scheduling, and path choice. To deal with this challenge, the following of this section first demonstrates the problem as Problem 1, then reformulates it to an integer linear programming (ILP) as Problem 2. After that, Problem 2 is relaxed to Problem 3 in order to be solved timely, and then transformed into Problem 4 and 5 to derive a feasible solution. At last, a 2-approximation Algorithm 2 is proposed to get the solution due to Problem 4 is a classic unrelated parallel machine scheduling problem known as NP-hard [14]. The whole procedure of solving Problem 1 is summarized in Algorithm 3.

A. 2-APPROXIMATION FOR A SINGLE COFLOW

The problem of minimizing average CCT jointly considering task placement, coflow bandwidth scheduling, and path

Algorithm 1 The Proposed Framework in This Paper

Require: The set of unfinished coflows C , the set of available bandwidth B , the set of available path P , waiting time threshold thd

Ensure: Task placement, coflow scheduling, and path choice

- 1: Non-increasingly sort the coflows in C according to their waiting time
- 2: **while** $C \neq \emptyset$ **do**
- 3: Set the minimum completion time $T_{\min} = \infty$, the corresponding coflow $c_{\min} = \emptyset$
- 4: **for** $c \in C$ **do**
- 5: Compute the minimum completion time T_c for coflow c , and the corresponding task placement, bandwidth scheduling, and path choice scheme to achieve T_c
- 6: **if** wait time of $c > \text{thd}$ **then**
- 7: $T_{\min} = T_c, c_{\min} = c$, **break**
- 8: **end if**
- 9: **if** $T_c < T_{\min}$ **then**
- 10: $T_{\min} = T_c, c_{\min} = c$
- 11: **end if**
- 12: **end for**
- 13: $C = C \setminus c_{\min}$
- 14: Assign all the flows in coflow c_{\min} using task placement, bandwidth scheduling, and path choice scheme derived at line 5, and then update B and P .
- 15: **end while**

choice, given the information of coflow i , can be formulated as Problem 1:

Problem 1

$$\text{minimize } T^i \quad (1a)$$

$$\text{s.t.: } \sum_k r_{khl}^i(t) \leq b_{hl}(t), \quad \forall t \geq 0, l \in P, h \in H. \quad (1b)$$

$$\sum_l \sum_h \int_0^{T^i} r_{khl}^i(t) dt = v_k^i, \quad \forall k \in K_i. \quad (1c)$$

$$r_{khl}^i(t) \leq x_{jhl}^i b_{hl}(t), \quad \forall h \in H, t \geq 0, j \in J, l \in P, k \in D_j^i, \quad (1d)$$

$$\sum_h x_{jhl}^i = 1, \quad \forall j \in J, l \in P. \quad (1e)$$

$$x_{jhl}^i \in \{0, 1\}, \quad \forall j \in J, h \in H, l \in P. \quad (1f)$$

In Problem 1, the objective T^i is the CCT of coflow i . Constraint (1b) is used to limit the rate of flows sent to host h on path l , where $b_{hl}(t)$ is the bandwidth limitation of the path l to host h at time t , and $r_{khl}^i(t)$ is the rate of sending the k^{th} flow in coflow i to host h along the path l at time t . Constraint (1c) means that the data should be sent out before coflow i completes, where v_k^i is the amount of data that should be transferred by the k^{th} flow of coflow i and K_i is the set of flows belonging to coflow i . Constraint (1d) is to denote if the task j is placed on host h , where D_j^i is the set of flows in coflow i that needs to be fetched by the j^{th} task and x_{jhl}^i is

a binary variable to denote if task j^{th} is placed onto host h and coflow i sends data to this task along path l . According to [14], [15], this problem can be proved to be NP-hard.

In order to solve Problem 1, the following Problem 2 is constructed and Theorem 1 is proposed.

Problem 2

$$\text{maximize } f^i \quad (2a)$$

$$\text{s.t.: } \sum_k r_{khl}^i \leq b_{hl}, \quad \forall l \in P, h \in H. \quad (2b)$$

$$\sum_l \sum_h r_{khl}^i = v_k^i f^i, \quad \forall k \in K_i. \quad (2c)$$

$$r_{khl}^i \leq x_{jhl}^i b_{hl}, \quad \forall h \in H, j \in J, l \in P, k \in D_j^i. \quad (2d)$$

$$\sum_h x_{jhl}^i = 1, \quad \forall j \in J, l \in P. \quad (2e)$$

$$x_{jhl}^i \in \{0, 1\}, \quad \forall j \in J, h \in H, l \in P. \quad (2f)$$

In Problem 2, it is assumed that the bandwidth limitation of each path is constant, which is possible when the time interval is short and before the global information updates [37]. Under this assumption, due to the following Theorem 1, solving Problem 2 can get the same solution as solving Problem 1.

Theorem 1: When the bandwidth of each path are constant, suppose \hat{r}_{khl}^i and \hat{x}_{jhl}^i are the optimal solutions, and \hat{f}^i is the objective value of the Problem 2. Then, $T^i = \frac{1}{\hat{f}^i}$ is the optimal objective value of Problem 1.

$$r_{khl}^i(t) = \begin{cases} \hat{r}_{khl}^i, & t \in [0, \frac{1}{\hat{f}^i}], \\ 0, & t \in (\frac{1}{\hat{f}^i}, \infty). \end{cases} \quad (3)$$

and $x_{jhl}^i = \hat{x}_{jhl}^i$ are the solutions to achieve the optimal objective value.

Proof: Suppose T_{opt} is the optimal objective of Problem 1, and $r_{khl}^{\text{opt}}(t)$ is the corresponding solution. By setting

$$r_{khl}^i = \frac{\int_0^{T_{\text{opt}}} r_{khl}^{\text{opt}}(t) dt}{T_{\text{opt}}} \quad (4)$$

it turns out

$$\sum_k \int_0^{T_{\text{opt}}} r_{khl}^{\text{opt}}(t) dt = \sum_k r_{khl}^i T_{\text{opt}} \quad (5)$$

Since

$$\sum_k \int_0^{T_{\text{opt}}} r_{khl}^{\text{opt}}(t) dt = \int_0^{T_{\text{opt}}} \sum_k r_{khl}^{\text{opt}}(t) dt \quad (6a)$$

$$\leq \int_0^{T_{\text{opt}}} b_{hl} dt = T_{\text{opt}} b_{hl} \quad (6b)$$

The relation $\sum_k r_{khl}^i \leq b_{hl}$ is known. In the same way, r_{khl}^i satisfies constraints (2c) and (2e). For constraint (2d), it can be seen that

$$\sum_h \int_0^{T_{\text{opt}}} r_{khl}^{\text{opt}}(t) dt = \sum_h r_{khl}^i T_{\text{opt}} = v_k^i \quad (7)$$

Let $f^i = \frac{1}{T_{opt}}$, and get

$$\sum_h r_{khl}^i = \frac{v_k^i}{T_{opt}} = v_{kf}^i f^i \quad (8)$$

The above contents show that $r_{khl}^i = \frac{\int_0^{T_{opt}} r_{khl}^{opt}(t) dt}{T_{opt}}$ and $f^i = \frac{1}{T_{opt}}$ are feasible solution of Problem 2. Therefore

$$\hat{f}^i \geq \frac{1}{T_{opt}} \quad (9)$$

In addition, it can be easily verified that the variable settings claimed in Theorem 1 are feasible solutions of Problem 1. Therefore

$$\hat{f}^i \leq \frac{1}{T_{opt}} \quad (10)$$

Accordingly, $\hat{f}^i \leq \frac{1}{T_{opt}}$. □

Next, to deal with the binary variables in Problem 2, assume all the hosts are packed together into a single “huge” host. Then the following Problem 3 can be constructed, which is a linear programming problem.

Problem 3

maximize f^i (11a)

s.t.: $\sum_k r_{kl}^i \leq \sum_h b_{hl}, \quad \forall l \in P.$ (11b)

$\sum_l r_{kl}^i = v_{kf}^i f^i, \quad \forall k \in K_i.$ (11c)

Note that the solution of Problem 3 is an upper bound of Problem 2. Suppose the solution of Problem 3 can be got, then the solution needs to be scaled down in order to be feasible to Problem 2. Denote the scale down ratio to be α , then the following Problem 4 can be derived to solve α .

Problem 4

minimize α (12a)

s.t.: $\sum_j (\sum_{k \in D_j^i} r_{khl}^i) x_{jhl}^i \leq \alpha b_{hl}, \quad \forall h \in H, l \in P.$ (12b)

$\sum_h x_{jhl}^i = 1, \quad \forall j \in J, l \in P.$ (12c)

$x_{jhl}^i \in \{0, 1\}, \quad \forall j \in J, h \in H, l \in P.$ (12d)

Problem 4 is a classic NP-hard unrelated parallel machine scheduling problem [38], which, however, can be solved by a method based on relaxation and rounding technique [43].

By defining $e_{jhl}^i = \frac{\sum_{k \in D_j^i} r_{khl}^i}{b_{hl}}$, Problem 4 can be modified to be Problem 5 by relaxation.

Problem 5

minimize α (13a)

s.t.: $\sum_{\{j|e_{jhl}^i \leq \alpha\}} e_{jhl}^i x_{jhl}^i \leq \alpha, \quad \forall h \in H, l \in P.$ (13b)

$\sum_{\{h|e_{jhl}^i \leq \alpha\}} x_{jhl}^i = 1, \quad \forall j \in J, l \in P.$ (13c)

$x_{jhl}^i \geq 0, \quad \forall j \in J, h \in H, l \in P.$ (13d)

For a fixed α , Problem 5 is a linear programming. In order to get a feasible solution to Problem 4, the solution of Problem 5 needs to be handled by rounding, since the results can be fractional.

In the following, Lemma 1 and Lemma 2 are proposed for the aim of rounding the fractional solution.

Lemma 1: No more than $(|J| + |H|) \cdot |P|$ variables are non-zero in the optimal solution of Problem 5.

Proof: When Problem 5 is feasible, objective α is minimized. For a given α , denote u as the number of variables in problem 5, then the feasible region is a single point determined by u linearly independent rows of the constraints with the equality.

There are $|H| \cdot |P|$ constraints in (13b) and $|J| \cdot |P|$ constraints in (13c) with a total of $u + (|J| + |H|) \cdot |P|$ constraints in Problem 5. Then in (13d) there are at least $u - |J| \cdot |P| - |H| \cdot |P|$ constraints holding with the equality. Thus, in (13d) at most $(|J| + |H|) \cdot |P|$ constraints do not hold with equality, which means at most $(|J| + |H|) \cdot |P|$ variables are non-zeros. □

Lemma 2: A bigraph $G(x) = \{U, V, W\}$ can be constructed according to the solution of Problem 5, where $U = \{u_1, u_2, \dots, u_{|H| \cdot |P|}\}$ is the set of host nodes, and $V = \{v_1, v_2, \dots, v_{|J| \cdot |P|}\}$ is the set of task nodes. There exists an edge $w \in W$ between v_j and u_h if and only if $x_{jhl}^i > 0$. Under this circumstance, without increasing the scale down ratio, any connected component S in $G(x)$ can be constructed to be a tree with at most one more edge.

Proof: If Problem 5 is solved by only using the tasks, hosts and paths associated with S and denote the solution as x' , it is clearly that the scale down ratio is less or equal to that derived by using all the tasks, hosts, and paths. In the light of Lemma 1, the number of non-zero values in the solution is no more than the number of nodes in S . Thus, S can be constructed to be a tree with at most one more edge by altering the edges according to x' . □

Based on Lemma 1 and Lemma 2, Algorithm 2 is designed for task placement, where $N(S)$ is the set of nodes in S and $E(S)$ is the set of edges in S .

In Algorithm 2, at line 2, tasks with only one host to place are handled. The rest of the task nodes have at least two node degrees. At line 4, one cycle exists in S and can be found and removed by depth first search, otherwise there is a forest of trees.

The following theorem shows that Algorithm 2 has an approximation ratio of 2.

Theorem 2: The approximation ratio of Algorithm (2) is 2.

Proof: A lower bound of the objective value of Problem 4 is denoted as α_{min} , which is also the optimal objective value of Problem 5. At line 2 of Algorithm (2), all the unsplit tasks are placed, the contribution of which to the scale down ratio is less than α_{min} . In Algorithm (2), line 3-13 make sure that every host holds at most one split task,

Algorithm 2 Get the Solution of Task Placement**Require:** The solution of Problem 5, $\{x_{jhl}^i\}$ **Ensure:** Task placement

- 1: Construct a bigraph BG according to $\{x_{jhl}^i\}$ as in Lemma 2
- 2: Remove all the tasks nodes with only one node degree and place these tasks onto the corresponding connecting hosts
- 3: **for** all connected components $S \in BG$ **do**
- 4: **if** $|N(S)| == |E(S)|$ **then**
- 5: Find the unique cycle in S with depth-first search
- 6: Arbitrarily orient the cycle in either of the two directions and place each task onto the host on the cycle subsequent to it
- 7: Remove this cycle from S , and the rest are a forest of trees, each of which contains at most one task leaf node
- 8: **for** the rest trees **do**
- 9: Rooting any task node or the unique task leaf node if there is one
- 10: Place each task onto its child host that holds the largest fraction of it
- 11: **end for**
- 12: **else**
- 13: Mark any task as the root to form a tree and place each task onto its child host that holds most of it
- 14: **end if**
- 15: **end for**

which increases the scale down ratio no more than α_{min} . Thus, the scale down ratio of Algorithm (2) is no more than $2\alpha_{min}$. \square

The whole procedure of solving Problem 1 is shown in Algorithm 3, in which Algorithm 2 only determines task placement, and Algorithm 3 uses Algorithm 2 to determine coflow bandwidth scheduling and path choice.

Algorithm 3 The Whole Procedure of Solving Problem 1**Require:** The size of each flow v_k^i , available bandwidth $\{b_{hl}\}$ **Ensure:** Task placement $\{x_{jhl}^i\}$ and flow transmission

- rate $\{r_{khl}^i\}$
- 1: Solve Problem 3 and get the maximum transmission rate r_{kl}^i
- 2: Use the solution of Problem 3, formulate Problem 4, get the result from Algorithm (2)
- 3: **for** all host h **do**
- 4: $r_{khl}^i = r_{kl}^i x_{jhl}^i, \forall k \in D_j^i, \alpha_{hl} = \frac{\sum_k r_{khl}^i}{b_{hl}}$
- 5: **if** $\alpha_{hl} > 1$ **then**
- 6: $r_{khl}^i = \frac{r_{khl}^i}{\alpha_{hl}}$
- 7: **end if**
- 8: **end for**

The following theorem shows that Algorithm 3 has an approximation ratio of 2.

Theorem 3: The approximation ratio of Algorithm 3 is 2.

Proof: Algorithm 3 introduces approximation only at line 2 via Algorithm 2. The bottleneck at each path can be got by solving Problem 3. Then at each path, Algorithm (3) scales down the coflow transmission rate in order to be feasible, which results in an approximation factor loss of 2. \square

IV. PERFORMANCE EVALUATION**A. SIMULATION SETUP**

To demonstrate the performance of the proposed framework, this paper compares it with two state-of-the-art works shown as follows. The reason to choose these two works is that they are the recent works after 2019 [14], [31], [35], [37] and represent two typical online algorithm designs.

- Baseline 1 [31]: considering task placement for new requests.
- Baseline 2 [37]: considering task placement and coflow scheduling only with incoming and outgoing links.

The range of evaluation consists the average CCT, algorithm run time, impacts of 4 coflow characteristics for Baseline 1, Baseline 2, and the proposed framework. And this paper develops a trace-driven simulator based on macOS Big Sur 11.0.1, with 32GB 2400MHz DDR4 and 2.3GHz 8 cores Intel i9.

The data used for simulation comes from a Hive/Map Reduce trace, which was collected on a 150-rack 3000-machine cluster with 10:1 oversubscription ratio [44] and was used in [9], [10], [14], [35]. As in the [45], the original coflow communication pattern is maintained.

As shown in Table 2, the non-zero coflows in the trace can be divided into 4 categories [9], [18]. The boundary between long coflow and short coflow is 5MB, and the boundary between narrow coflow and wide coflow is 50 flows inside.

TABLE 2. 4 coflows categories in the trace by length and width.

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

Use Linux Traffic Control [46], the path capacities and bandwidth limitation of each path can be set in the range of 100Mbps and 2Gbps randomly, in order to simulate the practical environments [45].

B. SIMULATION RESULTS**1) THE PERFORMANCE ON CCT**

Fig. 4 demonstrates the average CCT of coflows achieved by different schemes, in which the measurement unit on y-axis is $ms \times 10^4$. Across all types of coflows, the proposed framework can speed up the average CCT by 6% – 12% than Baseline 1 and Baseline 2. The reason is that the proposed framework jointly optimizes task placement, coflow bandwidth scheduling, and path choice, while the other two schemes ignore bandwidth scheduling and path choice.

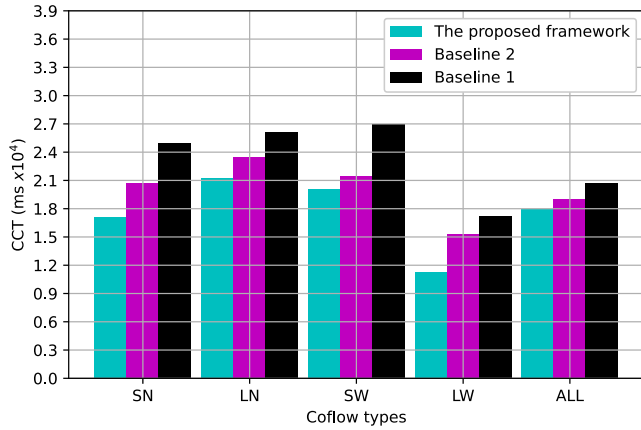


FIGURE 4. The average CCT of coflows, achieved by the proposed framework, Baseline 2, and Baseline 1.

To measure CCT of coflows at a microscopic level, Fig. 5 further plots CDF (Cumulative Distribution Function) of the completion time of three coflows schemes. Obviously, the curve of the proposed framework lies on the top of the other two. The percentages of coflows completed within 15×10^3 ms are 70.1%, 46.3%, 42.1% by the proposed framework, Baseline 2, and Baseline 1, respectively. And all coflows can be completed within 264020ms, 320320ms, 35144ms respectively.

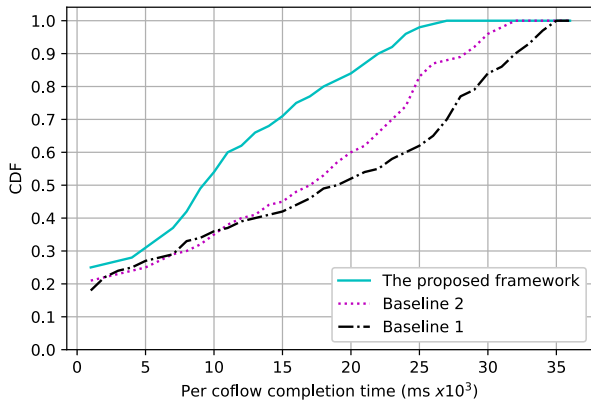


FIGURE 5. CDFs of per coflow CCT of the proposed framework, Baseline 2, and Baseline 1.

Fig. 6 illustrates the algorithm run time for three schemes. Note that the measurement unit on y-axis is now $ms \times 10$. It is easily seen that the run time of the proposed framework is much higher (2 – 3 times) than it of the other two schemes. The reason may be that the proposed framework jointly considers 3 factors (task placement, coflow bandwidth scheduling, and path choice) which needs more computation than Baseline 2 (task placement, and coflow bandwidth scheduling) and Baseline 1 (task placement). However, compared with the unit on y-axis of average CCT ($ms \times 10^4$) in Fig. 4, the effect of algorithm run time ($ms \times 10$) is relatively small.

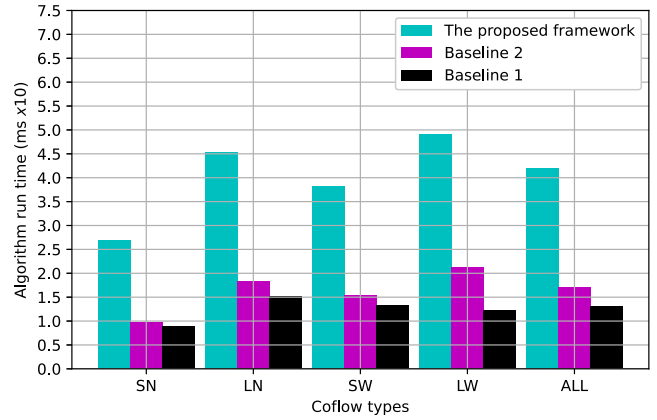


FIGURE 6. The algorithm run time, spent by the proposed framework, Baseline 2, and Baseline 1.

2) IMPACTS OF 4 COFLOW PARAMETERS

As in the reference [14], this subsection studies the impacts of 4 coflow parameters on average CCT: the total coflow number, the coflow width, the coflow size, and the inter-coflow arrival interval. In the following figures, the comparison baseline is the scheduling optimization with random task placement and path choice. It can be seen that the proposed framework has better performance than Baseline 2 and Baseline 1 under different scenarios.

a: COFLOW NUMBER

The coflow width, the coflow size and the mean inter-coflow arrival interval are set as 100, 500MB and 100ms, respectively [14], [35]. Fig. 7 shows that the performance of the average CCT increases with the growth of the coflow number for three schemes. The reason may be that the bandwidth scheduling and path choice strategy will get goodness when there exists a severe competition of network resource originated from more coflows. And the proposed framework has at least 15.4% more augmentation than the other two.

b: COFLOW WIDTH

The coflow number, the size and the mean inter-coflow arrival interval are set as 100, 500MB and 100ms, respectively [14], [35]. Fig. 8 shows that more improvement of average CCT is gained by all three schemes, due to the similar reason as above. And the proposed framework outperforms the other two schemes by at least 11.5%.

c: COFLOW SIZE

The coflow number, the width and the mean inter-coflow arrival interval are set as 100, 100 and 100ms, respectively [14], [35]. Fig. 9 shows that the performance improvement caused by three schemes are decreasing when coflow sizes increase. The reason may be that bigger coflow size means greater number of flows and bigger size of flows, which need more time for computation and transportation.

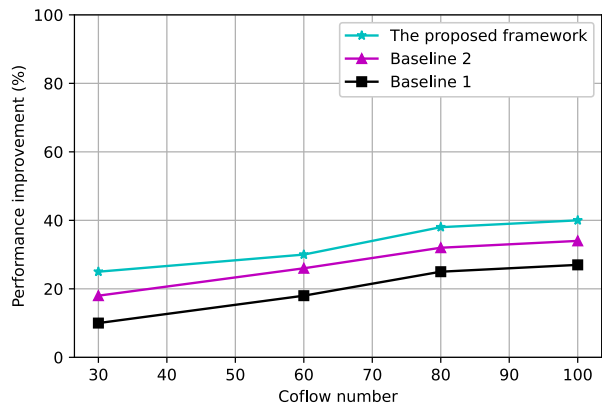


FIGURE 7. The impact of number of coflows on the average CCT of the proposed framework, Baseline 2, and Baseline 1 compared to the random case.

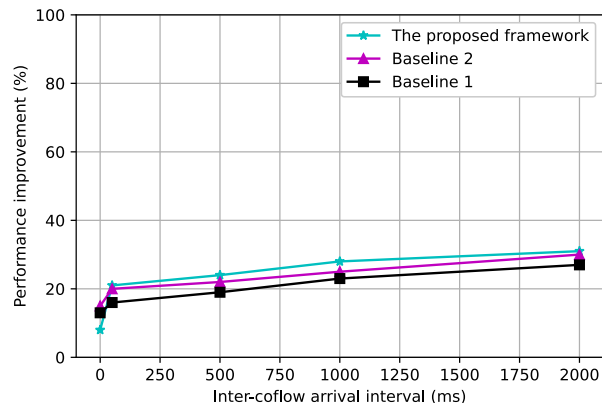


FIGURE 10. The impact of inter-coflow arrival interval on the average CCT of the proposed framework, Baseline 2, and Baseline 1 compared to the random case.

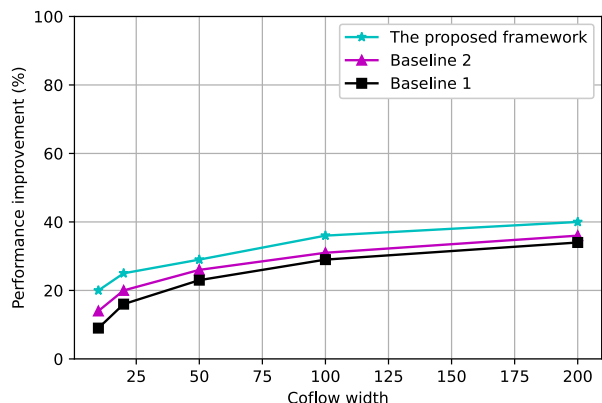


FIGURE 8. The impact of coflow width on the average CCT of the proposed framework, Baseline 2, and Baseline 1 compared to the random case.

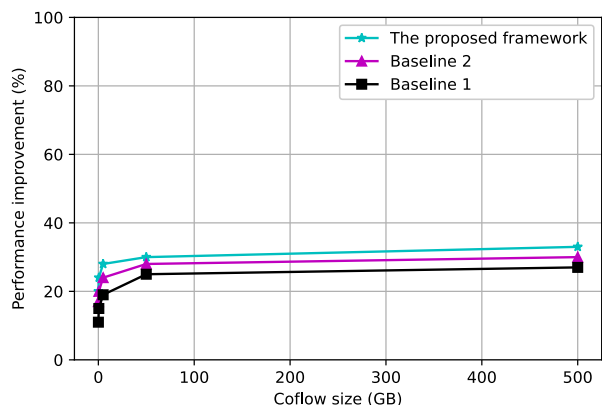


FIGURE 9. The impact of coflow size on the average CCT of the proposed framework, Baseline 2, and Baseline 1 compared to the random case.

The proposed framework has 7% more augmentation than the other two schemes.

d: INTER-COFLOW ARRIVAL INTERVAL

The other 3 parameters are prefixed as the previous parts [14], [35]. Fig. 10 shows that as the intervals increase, the slopes of improvement is greater at first and then become smaller.

And it can be observed that when the intervals are small, the curve of the proposed framework is relatively lower. The reason may be that the algorithm run time can not be ignored compared with the average CCT at this time. However, as the arrival intervals increase, the proposed framework has higher improvement on average CCT over the other two schemes.

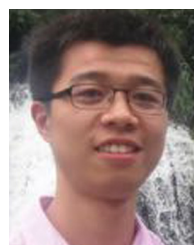
V. CONCLUSION

This paper is the first one to jointly consider minimizing the average CCT of task placement, coflow bandwidth scheduling, and path choice in intra-data centers. The proposed framework consists a 2-approximation algorithm for single coflow and follows the SRTF principle for multiple coflows. Extensive trace-driven evaluations have shown that the proposed framework is better than the state-of-the-art works. Jointly considering coflow optimization of transmission cost and routing policy might be a promising future direction, since the trade-off between cost and bandwidth consuming is a critical metric.

REFERENCES

- [1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. OSDI*, 2004.
- [2] G. Malewicz, M. H. Austern, A. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2010, pp. 135–146.
- [3] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed data-parallel programs from sequential building blocks," in *Proc. EuroSys*, 2007, pp. 59–72.
- [4] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proc. 11th ACM Workshop Hot Topics Netw.*, 2012, pp. 31–36.
- [5] M. Chowdhury, S. Kandula, and I. Stoica, "Leveraging endpoint flexibility in data-intensive clusters," in *Proc. ACM SIGCOMM Conf.*, Aug. 2013.
- [6] Y. Zhao, K. Chen, W. Bai, M. Yu, C. Tian, Y. Geng, Y. Zhang, D. Li, and S. Wang, "Rapier: Integrating routing and scheduling for coflow-aware data center networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 424–432.
- [7] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 98–109, Oct. 2011.
- [8] L. Luo, J. Nelson, L. Ceze, A. Phanishayee, and A. Krishnamurthy, "Parameter hub: A rack-scale parameter server for distributed deep neural network training," in *Proc. ACM Symp. Cloud Comput.*, 2018, pp. 41–54.

- [9] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varies," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 443–454.
- [10] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015.
- [11] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in *Proc. EuroSys*, 2010, pp. 265–278.
- [12] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair scheduling for distributed computing clusters," in *Proc. SOSP*, 2009, pp. 261–276.
- [13] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *Proc. NSDI*, 2017, pp. 629–647.
- [14] H. Tan, S. H.-C. Jiang, Y. Li, X.-Y. Li, C. Zhang, Z. Han, and F. C. M. Lau, "Joint online coflow routing and scheduling in data center networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 1771–1786, Oct. 2019.
- [15] C. Banino-Rokkones, O. Beaumont, and H. Rejeb, "Scheduling techniques for effective system reconfiguration in distributed storage systems," in *Proc. 14th IEEE Int. Conf. Parallel Distrib. Syst.*, Dec. 2008, pp. 80–87.
- [16] C.-Y. Hong, M. Caesar, and P. Brighten Godfrey, "Finishing flows quickly with preemptive scheduling," 2012, *arXiv:1206.2057*. [Online]. Available: <http://arxiv.org/abs/1206.2057>
- [17] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, "Decentralized task-aware scheduling for data center networks," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014.
- [18] Z. Qiu, C. Stein, and Y. Zhong, "Minimizing the total weighted completion time of coflows in datacenter networks," in *Proc. 27th ACM Symp. Parallelism Algorithms Archit.*, Jun. 2015, pp. 294–303.
- [19] L. Liu, H. Xu, C. Gao, and P. Wang, "Bottleneck-aware coflow scheduling without prior knowledge," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Jul. 2020, pp. 50–55.
- [20] S. Wang, S. Wang, R. Huo, T. Huang, J. Liu, and Y. Liu, "Deepaalo: Auto-adjusting demotion thresholds for information-agnostic coflow scheduling," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Jul. 2020, pp. 1123–1128.
- [21] S. Luo, H. Yu, Y. Zhao, B. Wu, S. Wang, and L. M. Li, "Minimizing average coflow completion time with decentralized scheduling," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 307–312.
- [22] S. Luo, H. Yu, Y. Zhao, S. Wang, S. Yu, and L. Li, "Towards practical and near-optimal coflow scheduling for data center networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 11, pp. 3366–3380, Nov. 2016.
- [23] 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 Conf.*, Aug. 2016, pp. 160–173.
- [24] A. Bhimaraju, D. Nayak, and R. Vaze, "Non-clairvoyant scheduling of coflows," in *Proc. 18th Int. Symp. Model. Optim. Mobile, AdHoc, Wireless Netw. (WiOPT)*, Jun. 2020, pp. 1–8.
- [25] L. Chen, S. Liu, B. Li, and B. Li, "Scheduling jobs across geo-distributed datacenters with max-min fairness," *IEEE Trans. Netw. Sci. Eng.*, vol. 6, no. 3, pp. 488–500, Jul. 2019.
- [26] Z. Wu and L. Fu, "Optimizing job completion time with fairness in large-scale data centers," *Future Gener. Comput. Syst.*, vol. 114, pp. 563–573, Jan. 2021.
- [27] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015.
- [28] Z. Hu, B. Li, and J. Luo, "Flutter: Scheduling tasks closer to data across geo-distributed datacenters," in *Proc. INFOCOM*, vol. 2016, pp. 1–9.
- [29] R. Viswanathan, G. Ananthanarayanan, and A. Akella, "CLARINET: Wan-aware optimization for analytics queries," in *Proc. OSDI*, 2016, pp. 435–450.
- [30] V. Jalaparti, P. Bodik, I. Menache, S. Rao, K. Makarychev, and M. Caesar, "Network-aware scheduling for data-parallel jobs: Plan when you can," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015.
- [31] A. Munir, T. He, R. Raghavendra, F. Le, and A. X. Liu, "Network scheduling and compute resource aware task placement in datacenters," *IEEE/ACM Trans. Netw.*, vol. 28, no. 6, pp. 2435–2448, Dec. 2020.
- [32] X. S. Huang and T. S. E. Ng, "Exploiting inter-flow relationship for coflow placement in datacenters," in *Proc. 1st Asia-Pacific Workshop Netw.*, Aug. 2017, pp. 113–119.
- [33] F. Ahmad, S. Chakradhar, A. Raghunathan, and T. Vijaykumar, "Shuffle-watcher: Shuffle-aware scheduling in multi-tenant mapreduce clusters," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 1–3.
- [34] Y. Li, S. H. Jiang, H. Tan, C. Zhang, G. Chen, J. Zhou, and F. C. Lau, "Efficient online coflow routing and scheduling," in *Proc. MobiHoc*, 2016, pp. 161–170.
- [35] W. Li, X. Yuan, K. Li, H. Qi, X. Zhou, and R. Xu, "Endpoint-flexible coflow scheduling across geo-distributed datacenters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 10, pp. 2466–2481, Oct. 2020.
- [36] Z. Wu, "Joint online coflow optimization across geo-distributed datacenters," *IEEE Access*, vol. 8, pp. 213602–213610, 2020.
- [37] Y. Zhao, C. Tian, J. Fan, T. Guan, X. Zhang, and C. Qiao, "Joint reducer placement and coflow bandwidth scheduling for computing clusters," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 438–451, Feb. 2021.
- [38] Y. Zhao, C. Tian, J. Fan, T. Guan, and C. Qiao, "RPC: Joint online reducer placement and coflow bandwidth scheduling for clusters," in *Proc. IEEE 26th Int. Conf. Netw. Protocols (ICNP)*, Sep. 2018, pp. 187–197.
- [39] W. Wang, S. Ma, B. Li, and B. Li, "Coflex: Navigating the fairness-efficiency tradeoff for coflow scheduling," in *Proc. IEEE Conf. Comput. Commun.*, May 2017, pp. 1–9.
- [40] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 503–514.
- [41] Y. Peng, K. Chen, G. Wang, W. Bai, Y. Zhao, H. Wang, Y. Geng, Z. Ma, and L. Gu, "Towards comprehensive traffic forecasting in cloud computing: Design and application," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2210–2222, Aug. 2016.
- [42] H. Wang, L. Chen, K. Chen, Z. Li, Y. Zhang, H. Guan, Z. Qi, D. Li, and Y. Geng, "FLOWPROPHET: Generic and accurate traffic prediction for data-parallel cluster computing," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst.*, Jun. 2015, pp. 349–358.
- [43] J. K. Lenstra, D. B. Shmoys, and É. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Math. Program.*, vol. 46, nos. 1–3, pp. 259–271, 1990.
- [44] *Facebook Hive/Mapreduce Trace*. Accessed: Aug. 10, 2020. [Online]. Available: <https://github.com/coflow/coflow-benchmark>
- [45] X. Xu, W. Li, K. Li, H. Qi, and Y. Jin, "Optimizing the cost-performance tradeoff for coflows across geo-distributed datacenters," *IEEE Access*, vol. 6, pp. 24488–24497, 2018.
- [46] *Linux Traffic Control*. Accessed: Aug. 10, 2020. [Online]. Available: <http://lartc.org/manpages/tc.txt>



ZHAOXI WU received the B.E. degree from Shanghai Jiao Tong University, China, in 2014. He is currently pursuing the Ph.D. degree with the School of Information Science and Technology, ShanghaiTech University. His research interests include data-center networks and network optimization.

...