2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)

Reco: Efficient Regularization-Based Coflow Scheduling in Optical Circuit Switches

Chi Zhang*, Haisheng Tan*, Chao Xu*, Xiang-Yang Li*, Shaojie Tang‡, and Yupeng Li†
* University of Science and Technology of China (USTC), Hefei, China
† The University of Hong Kong, Hong Kong ‡University of Texas at Dallas, USA

***Abstract*—To improve the application-level data efficiency, the scheduling of coflows, defined as a collection of parallel flows sharing the same objective, is prevailing in recent data centers. Meanwhile, optical circuit switches (OCS) are gradually applied to provide high data rate with low power consumption. However, so far few research outputs have covered the flow scheduling in the context of OCS, let alone the coflow scheduling problems.**

In this paper, we investigate coflow scheduling in the OCS-based data centers. We first derive a novel operation called *regularization* processed respectively on the flow traffic demands and the flow start times. Regularization can be efficiently implemented and reduce the circuit reconfiguration frequency dramatically. We then propose a 2-approximation algorithm, called `Reco-Sin`, for single coflow scheduling to minimize the coflow completion time (CCT). For multiple coflows, we derive another approximation algorithm, called `Reco-Mul`, to minimize the total weighted CCT, which can transform any non-preemptive multi-coflow scheduling in packet switches to that in OCS. Extensive simulations based on `Facebook` data traces show that `Reco-Sin` and `Reco-Mul` outperform state-of-the-art schemes significantly, i.e., one single coflow can be finished up to $2.72\times$ faster with `Reco-Sin`, and multiple coflows can be completed up to $3.44\times$ faster with `Reco-Mul`.

I. Introduction

In current distributed computing frameworks (*e.g.*, MapReduce [1], Dryad [2] and Spark [3]), communication data flows in the network may share a common performance goal as they are likely to correspond to the same job from one application. To capture this kind of application-level requirements, a new abstraction of structured data flows, called *coflow*, is proposed to allow applications to convey their semantics to the network [4]. A coflow is a collection of related parallel individual flows that occur typically between two stages of a multi-stage computing task, such as shuffling in the MapReduce. To improve the application-level performance, instead of analyzing a single flow, we turn to optimize the *coflow completion time* (CCT), defined as the duration from its arrival to the completion of all the individual flows.

Meanwhile, based on the development of micro-electro-mechanical system (MEMS) techniques, optical circuit switches (OCS) become popular in modern cluster networks [5]–[9]. Compared with electrical packet switch, an OCS has much higher data transfer rate but lower power consumption. However, each ingress or egress port in an OCS is restricted to establish at most one circuit for data transmission at a time, called *the port constraint*. A reconfiguration to establish new circuits in OCS will take a fixed period, called *the reconfiguration delay* and denoted as δ, typically in tens of microseconds [10].

In order to minimize the CCT, coflow scheduling, including single coflow scheduling (a.k.a. intra-coflow scheduling on individual flows within a coflow) and multiple coflow scheduling (a.k.a. inter-coflow scheduling), has been extensively studied in traditional electrical packet switches [11]–[14]. In OCS, individual flow scheduling has also been frequently studied (*e.g.* [5]–[10], [15]). However, there are few works on coflow scheduling in OCS. Sunflow [9] mainly focused on the single coflow and provided a simple heuristic policy for multiple coflows. The problem of OCS-based coflow scheduling is more challenging mainly due to: 1) Besides the transmission order of flows, we have to design the establishment and scheduling of the circuits to avoid frequent time-consuming circuit reconfigurations; 2) the port constraint can not be violated when establishing circuits for multiple flow transmission simultaneously; and 3) when multiple coflows coexist, the competition among coflows makes the problem even tougher.

In this paper, we investigate both the single and multiple coflow scheduling problem in OCS-based data centers, aiming at minimizing the CCT and the weighted average CCT, respectively. Our contributions can be summarized as follows:

- We propose a simple and effective operation, called *regularization*, processed respectively on the flow traffic demands (Sec. III-B) and the flow start times (Sec. IV-A) such that the circuit reconfiguration frequency is reduced significantly.
- Based on regularization, we propose a single coflow scheduling algorithm, named `Reco-Sin`, to minimize the CCT with an approximation ratio of 2 (Sec. III). To the best of our knowledge, this is the first constant approximation algorithm for the all-stop OCS [1].
- We further derive a novel algorithm, named `Reco-Mul`, for multiple coflow scheduling to minimize the total weighted CCT (Sec. IV). We theoretically prove that `Reco-Mul` can effectively transform *any* non-preemptive

Haisheng Tan is the Corresponding Author (Email: hstan@ustc.edu.cn).

This work is supported partly by the National Key R&D Program of China 2018YFB0803400, China National Funds for Distinguished Young Scientists No. 61625205, NSFC Grants 61772489, 61751211, Key Research Program of Frontier Sciences (CAS) No. QYZDY-SSW-JSC002, NSF ECCS-1247944, NSF CNS 1526638, and the Fundamental Research Funds for the Central U.

[1]In the all-stop model, a circuit reconfiguration will halt all circuits established in the OCS until the reconfiguration is completed.

2575-8411/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDCS.2019.00020

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on February 01,2023 at 04:39:41 UTC from IEEE Xplore. Restrictions apply.

multi-coflow scheduling scheme in packet switches with an approximation ratio of Δ into a feasible OCS-based coflow scheduling with an approximation ratio of $\Delta \cdot \left(1+\frac{1}{\lfloor\sqrt{c}\rfloor}\right)^2$. Here, we assume that a nonzero traffic demand in OCS is no less than $c\times\delta$, where c is a positive constant and δ is the circuit reconfiguration delay[2].

- Based on real data traces from `Facebook`, we conduct extensive simulations to compare our algorithms with state-of-the-art schemes (Sec. V). Our simulation results demonstrate that the number of reconfigurations in `Reco-Sin` can be reduced to only $\frac{1}{7}$ of that in the baselines, and hence `Reco-Sin` can finish transmitting a single coflow up to $2.72\times$ faster. As for multiple coflow scheduling, `Reco-Mul` outperforms the baselines dramatically, which finishes transmitting multiple coflows up to $3.44\times$ faster on average.

Paper Organization: We present the model and problem definitions in Sec. II. Our algorithms for the single coflow and multiple coflow scheduling are proposed in Sec. III and Sec. IV, respectively. Extensive simulations are in Sec. V. Discussion is presented in Sec. VI. Related work including a comparison among our results and the existing works is in Sec. VII. We conclude the whole paper in Sec. VIII.

II. Problem Formulation

In this section, we present the system model and formally define our coflow scheduling problem in OCS.

A. System Model

Network Model: Similar to existing works ([9], [11], [12], [16]), the fabric of a data center network is abstracted as one non-blocking circuit switch with N ingress ports and N egress ports (Fig. 1). Data flows are buffered at senders to be transferred from the ingress to the egress ports.

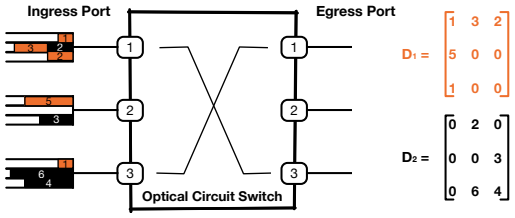


Fig. 1. A 3×3 Network Model: Coflow 1 and 2 are to be transmitted, whose demand matrices are D_1 and D_2, respectively.

Optical Circuit Switch: To transmit data from ingress port i to egress port j, OCS needs to establish a circuit between the two ports. Without loss of generality, the bandwidth of a circuit from one ingress to one egress port is normalized to 1, which is fully occupied by only one flow at a time. Recall that circuit establishment should satisfy the port constraint, that is to say, each ingress(egress) port can only establish a connection to one egress(ingress) port at a time. Denote the circuit reconfiguration delay as δ. Multiple circuits can be reconfigured simultaneously during one reconfiguration if they do not share a common port. Here, we adopt the *all-stop* circuit switch model as [6]–[8], [10], where a circuit reconfiguration would halt all transmissions in the OCS [3]. We define a *circuit establishment*, denoted by $C(u)$, as all the circuits concurrently established in an OCS. Due to the port constraint, $C(u)$ is in fact a matching in an $N\times N$ bipartite graph where the two disjoint vertex groups are the egress and ingress ports, respectively. The duration of the circuit establishment $C(u)$ is denoted as $dur(u)$. We call the pair $(C(u), dur(u))$ as a *circuit assignment*. Furthermore, a *circuit scheduling* $\mathcal{C}$ is composed of a sequence of circuit assignments $\{(C(1), dur(1)), \cdots, (C(m), dur(m))\}$, where m is the number of assignments. A circuit scheduling is *valid* if the port constraint is satisfied.

Coflow: A set $\mathcal{K}$ of coflows are to be transmitted through the OCS. Denote $K=|\mathcal{K}|$ as the number of coflows. Each coflow $k\in\mathcal{K}$ has a non-negative weight w_k, which indicates its sensitivity to the latency. Denote a_k as the arrival time of coflow k. Similar to [11], [14], [16], we assume that the flows within one coflow arrive at the same time. An $N\times N$ matrix D_k is to denote the data transmission demand of coflow k, which is called the *demand matrix*. Each entry $d^k_{i,j}$ in this matrix represents the amount of data to be transferred from ingress port i to egress port j. For example, in Fig. 1, $d^2_{2,3}=3$ means the data amount in Coflow 2 to transmit from ingress 2 to egress 3 is 3. Since the bandwidth of circuits is normalized to 1, without ambiguity, we also denote $d^k_{i,j}$ as the time needed to transmit all demands from ingress port i to egress port j. As stated in Sec. I, we assume that there is no tiny traffic demand in OCS, *i.e.*, $d^k_{i,j}\geq c\cdot\delta$ for any i, j and k, where c is a constant called the *optical transmission threshold*. We set $f^k_{i,j}$ as the time when the transmission demand from port i to j in coflow k is completed. Then, the time that the whole coflow k is finished is defined as $f_k=\max_{i,j} f^k_{i,j}$. Hence, the *coflow completion time* (CCT) of coflow k, denoted as T_k, is the duration from its arrival to the completion of transmission, calculated as $T_k=f_k-a_k$. Since we consider the data flows have been buffered in the senders, the arrival time of all coflows can be assumed to be the same similar to [11] [14] [16], *i.e.*, $a_k=0,\ \forall k$.

B. Problem Statement

We next formally define coflow scheduling problems for a single coflow and multiple coflows, respectively.

Problem 1 (Single Coflow Scheduling)**.** *Given a single coflow with demand matrix D arriving at a network which is modeled as an $N\times N$ non-blocking OCS, the problem is to find a feasible coflow scheduling so that its CCT is minimized.*

[2]This assumption is reasonable as in practice only elephant flows are transferred through OCS while mice flows can be handled more efficiently by packet switches [7].

[3]We will show that our results for multiple coflow scheduling can be extended to the *not-all-stop model* (See in Sec. VI).

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on February 01,2023 at 04:39:41 UTC from IEEE Xplore. Restrictions apply.

Note that as there is only one coflow in the network (with demand matrix D), a circuit scheduling $\mathcal{C}$ can be directly transformed into a coflow scheduling $\mathcal{S}$, *i.e.*, at each time, scheduling the traffic demand in D to transmit along all the currently established circuits. Therefore, Problem 1 is equivalent to finding a valid circuit scheduling to finish all the traffic demands in D with the minimum time, which is composed of two parts, the aggregate flow transmission time and and the total delay caused by reconfigurations.

Problem 2 (Multiple Coflow Scheduling)**.** *In a network modeled as an $N \times N$ non-blocking OCS, a set $\mathcal{K}$ of coflows arrive with demand matrices $D_k, \forall k \in \mathcal{K}$. The problem is to find a feasible coflow scheduling such that the total weighted coflow completion time, $\sum_{k\in\mathcal{K}} w_k T_k$, is minimized.*

The coflow scheduling problem has already been proved NP-hard [9], [17]. In the following, we will propose our novel efficient algorithms to achieve approximate results.

III. Single Coflow Scheduling

The main challenge for single coflow scheduling in OCS is how to effectively decrease the reconfiguration frequency. We leverage the classical Birkhoff-von Neumann (BvN) decomposition, and design a novel operation, called *regularization*.

A. Birkhoff-von Neumann Decomposition

According to Birkhoff's Theorem [18], any doubly stochastic matrix [4] can be decomposed into a set of permutation matrices with specific coefficients by Birkhoff-von Neumann (BvN) decomposition. An example can be found in Fig. 2. Under BvN decomposition, an $N \times N$ matrix can be decomposed to $m \leq N^2 - 2N + 2$ permutation matrices [19]. It is NP-hard to compute the BvN decomposition with the minimum permutation matrices [20].

Given a single coflow with demand matrix D, we can make it as a doubly stochastic matrix D' through increasing the values of some entries, which is called *stuffing*. A BvN decomposition over D' is exactly a circuit scheduling that meets the coflow traffic demand, *i.e.*, each permutation matrix is a circuit establishment whose duration is the coefficient. In fact, if the reconfiguration delay is zero (*i.e.*, $\delta = 0$), Problem 1 can be solved optimally by stuffing and BvN decomposition [16]. However, with non-negligible reconfiguration delays, it might lead to a terrible CCT to get a circuit scheduling based on the BvN decomposition in previous works (*e.g.*, [7], [10], [16]). This is because when preemption occurs, many tiny residual demands are produced leading to extensive reconfigurations. Specifically, we have the following theorem.

Theorem 1. *Given a coflow with demand matrix $D \in \mathbb{R}^{N\times N}$, the method that generates a valid circuit scheduling to satisfy the demand through stuffing and BvN decomposition has an approximation ratio of $\Omega(N)$ to minimize the CCT in OCS.*

[4]A doubly stochastic matrix is a square matrix of nonnegative real numbers, where the sum of each rows and columns equal to a constant.

Proof. According to [21], there exists a doubly stochastic matrix $D = [d_{i,j}] \in \mathbb{R}^{N\times N}$ with the minimum number of decomposition matrices as $\gamma(D) = \lceil n/2 \rceil \lceil (n+1)/2 \rceil$. We can construct a new matrix $D' = [d'_{i,j}] = [d_{i,j} \cdot \epsilon]$, where ϵ is a small enough positive constant. The minimum number of decomposition matrices of D' is $\gamma(D)$.

D' needs $\gamma(D)$ decomposition matrices based on BvN, that is to say the total reconfiguration delay is $\gamma(D) \cdot \delta$. Thus, the coflow completion time is $\gamma(D)\cdot\delta + t_{trans} = \Theta(N^2)\cdot\delta + t_{trans}$. Here, t_{trans} is the data transmission time depending on the traffic demand that could be close to zero. However, there exists a scheduling could transmit all demand within N circuits and the CCT of this scheduling would be at most $N \cdot \delta + t'_{trans}$, where t'_{trans} could be close to zero. Therefore, the approximation ratio of the algorithm by directly using primary BvN decomposition is $\Omega(N)$. □

B. Regularization on Traffic Demand

The deficiency of primitive BvN-based coflow scheduling is due to the frequent preemption, while non-preemptive scheduling may cause long circuit idle time. To address this issue, we propose a pre-processing operation, named *regularization*, which can greatly reduce the reconfiguration frequency with little cost on the circuit idle time. Given a demand matrix D, the process of regularization is to regularize each entry $d_{i,j} \in D$ to $\lceil \frac{d_{i,j}}{\delta} \rceil \cdot \delta$, and get a new regularized matrix D' [5].

Fig. 2 is an example illustrating the benefit of regularization. A demand matrix D_{ex} is decomposed into 5 permutation matrices each with a coefficient. Each permutation matrix corresponds to a circuit configuration, and each coefficient represents the transmission time on the corresponding configuration. We set $\delta = 100$. Then the time to complete D_{ex} is $(107 + 104 + 101 + 2 + 1) + 5 \times 100 = 815$. If we do *regularization*, the resulting matrix D'_{ex} can be decomposed into 3 permutation matrices each with a coefficient of 200. In fact, the actual transmission time of each circuit may be less than 200, because when one circuit finishes transmitting its demand, the OCS will automatically reconfigure the circuit for further transmission. In view of this, the actual completion time of D'_{ex} is $(106 + 109 + 103) + 3 \times 100 = 618$, which is much less than 815, the completion time before regularization.

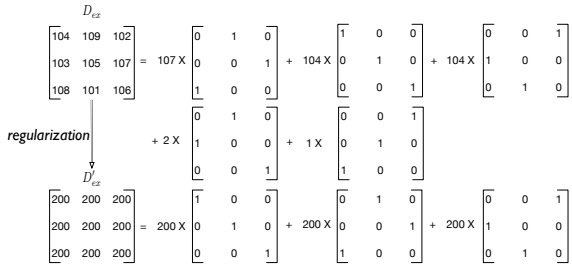


Fig. 2. An example of decomposition and regularization.

[5]As we only increase the entry values during regularization, a valid circuit scheduling satisfying D' will definitely satisfy D.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on February 01,2023 at 04:39:41 UTC from IEEE Xplore. Restrictions apply.

C. Algorithm

Based on the above techniques, given the coflow demand matrix D, our regularization-based single coflow scheduling, called `Reco-Sin`, consists of following steps: 1) Regularize D and get a doubly stochastic matrix D' by stuffing; 2) Run BvN decomposition on D'; and 3) Regard each permutation matrix as a circuit establishment and its coefficient as the duration to derive a sequence of circuit assignments. The details of `Reco-Sin` is described in Algorithm 1.

Algorithm 1: `Reco-Sin`: Single Coflow Scheduling

1 **Input** *coflow demand matrix D, reconfiguration delay δ*
2 **Output** *a circuit scheduling $\mathcal{C}$*
3 $\mathcal{C} \leftarrow \varnothing$;
4 $D' \leftarrow$ Run regularization and stuffing on D with δ;
5 **while** *D' has non-zero entries* **do**
6 Find a matching on D', let P, α be the permutation matrix and the coefficient derived from this matching;
7 Append (P, α) to $\mathcal{C}$;
8 $D' \leftarrow D' - \alpha P$;
9 Return $\mathcal{C}$;

Initially, Algorithm 1 calls the regularization operation (Line 4). It then iteratively calculates a decomposition (Line 6) and computes the permutation matrix with the maximum coefficient (Line 8). We efficiently compute BvN decomposition by max-min matching which is similar to the method in [7]. Each permutation matrix and its coefficient corresponds to a circuit assignment. We repeat the process until the demand matrix has no non-zero entry and return the circuit scheduling $\mathcal{C}$. As discussed in Fig. 2, the actual completion time of the demand matrix after regularization can be much less than the value of coefficients, since the circuit will stop communication as long as the demands on it are finished.

D. Theoretical Analysis

Here we will prove `Reco-Sin` is 2-approximate, which is the first constant approximation algorithm for this problem.

Denote t'_{trans} and t'_{conf} as the transmission and the configuration time of the scheduling obtained from `Reco-Sin`, respectively, while t^*_{trans} and t^*_{conf} are the transmission and the configuration time of the optimal solution respectively.

Lemma 1. *In `Reco-Sin`, the reconfiguration time is no greater than the transmission time,* i.e.*, $t'_{conf} \leq t'_{trans}$.*

Proof. Let m be the number of assignments obtained from `Reco-Sin`. Recall that $t'_{conf} = m \cdot \delta$ and $t'_{trans} = \sum_{i=1}^{m} dur'(i)$. Each $d_{i,j}$ after regularization is an integral multiple of δ's. Thus, by BvN decomposition, each circuit assignment lasts at least a time of δ, which gives $dur'(i) \geq \delta,\ \forall i = \{1, 2, ..., m\}$. This implies $t'_{conf} \leq t'_{trans}$. □

Set T' as the CCT given by `Reco-Sin`, while the optimal is T^*. Based on the above lemma, we have Theorem 2.

Theorem 2. *`Reco-Sin` is 2-approximate,* i.e.*, $T' \leq 2 \cdot T^*$.*

Proof. Suppose D and D' are the original coflow demand matrix and the regularized demand matrix, respectively. Let ρ and ρ' be the maximum value of the sum of each row and column of D and D' respectively. Let τ be the maximum number of non-zero entries of each row or column of D. Obviously, ρ is the lower bound on the transmission time of Problem 1, that is $\rho \leq t^*_{trans}$. As we require at least τ times circuit establishment for D, we have $\tau\delta \leq t^*_{conf}$. Since, by regularization, each entry of D is increased by at most δ, we get $\rho' \leq \rho + \tau\delta$. The regularized algorithm strictly takes the coefficient of BvN decomposition of D' as the duration of each circuit assignment. And we have $\rho' = t'_{trans}$. By Lemma 1, we can derive that

$$\begin{aligned} T' &= t'_{trans} + t'_{conf} \leq 2 \cdot t'_{trans} = 2 \cdot \rho' \\ &\leq 2 \cdot (\rho + \tau\delta) \leq 2 \cdot (t^*_{trans} + t^*_{conf}) = 2 \cdot T^*. \end{aligned}$$

This completes the proof. □

Note that, in the proof of Theorem 2, we do not make use of the assumption that $d^k_{i,j} \geq c \cdot \delta$ set in Sec. II, which means Theorem 2 holds for all coflow demand matrices.

IV. Multiple Coflow Scheduling

To schedule multiple coflows in an OCS-based data center, we propose our algorithm, called `Reco-Mul`, and the theoritical analysis on the approximation ratio.

A. Our algorithm: `Reco-Mul`

Although the multi-coflow scheduling problem for packet switches has been extensively studied in existing literature [11], [12], [14], [16], directly applying these algorithms in optical circuit switches will result in frequent circuit reconfigurations. To merit the benefits of the existing results for packet switches, we design `Reco-Mul` that can transform any non-preemptive multi-coflow scheduling in packet switches to a scheduling in an OCS model. Here, non-preemptive scheduling in packet switches means that there can be at most one flow transmitting at one time on each port, and once a flow starts transmission, it will be completed without preemption. We denote ALG_p as any algorithm that generates a non-preemptive multi-coflow scheduling in packet switches. According to the scheduling $\mathcal{S}_p$ returned by ALG_p, our algorithm `Reco-Mul` uses a regularization-based policy that can reduce the reconfiguration frequency effectively.

As $\mathcal{S}_p$ is a non-preemptive scheduling, it can be transformed into a feasible scheduling in OCS by trivially introducing a reconfiguration delay when the circuit in the OCS changes. Directly using $\mathcal{S}_p$ to schedule coflows in OCS would incur too many reconfigurations. Intuitively, if we can smartly align the start time of the flow transmissions in $\mathcal{S}_p$, we can somehow start multiple conflict-free flows (that do not share the same ports) at the same time such that only one circuit configuration is needed before transmitting these flows. Fig. 3 illustrates an example of the regularization on three conflict-free flows when $\sqrt{c} \cdot \delta = 1$. Before regularization, three reconfigurations are

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on February 01,2023 at 04:39:41 UTC from IEEE Xplore. Restrictions apply.

needed at $t = 0.5,\ 0.7$ and 0.9, while after the start time is regularized to 1, only one reconfiguration is needed. This reduces the cost of reconfiguration greatly.

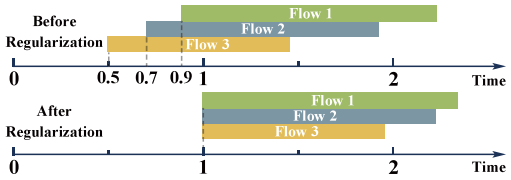


Fig. 3. An example of regularization on the flow start times.

Here, we define a pseudo-time axis, denoted as $\hat{t}$, on which the reconfiguration delay δ shrinks to 0. The start point $\hat{t} = 0$ is corresponding to the real time point $t = 0$ [6]. We describe our algorithm, `Reco-Mul`, in Algorithm 2.

Algorithm 2: `Reco-Mul`: Multiple Coflows Scheduling

1 **Input** *coflow demand matrices $\mathcal{D}$, coflow weights $\mathcal{W}$, reconfiguration delay δ, optical transmission threshold c*
2 **Output** *a multiple coflows scheduling $\mathcal{S}_o$*
3 $\mathcal{S}_o \leftarrow \varnothing, \hat{\mathcal{S}}_o \leftarrow \varnothing$;
4 $\mathcal{S}_p \leftarrow ALG_p(\mathcal{D}, \mathcal{W})$;
5 **for** $(t_1, t_2, i, j, k) \in \mathcal{S}_p$ **do**
6 $\quad \hat{t}_1^{\dagger} \leftarrow t_1 \cdot \frac{\lfloor\sqrt{c}\rfloor+1}{\lfloor\sqrt{c}\rfloor}$;
7 $\quad \hat{t}_1 \leftarrow \lfloor \frac{\hat{t}_1^{\dagger}}{\sqrt{c}} \rfloor \cdot \sqrt{c} \cdot \delta$;
8 $\quad \hat{t}_2 \leftarrow \hat{t}_1 + t_2 - t_1$;
9 $\quad \hat{\mathcal{S}}_o \leftarrow \hat{\mathcal{S}}_o \cup (\hat{t}_1, \hat{t}_2, i, j, k)$;
10 Define $\eta(\hat{\mathcal{S}}_o, \hat{t}\,)$ as the number of reconfiguration required in $\hat{\mathcal{S}}_o$ during the time period $[0, \hat{t}\,)$;
11 **for** $(\hat{t}_1, \hat{t}_2, i, j, k) \in \hat{\mathcal{S}}_o$ **do**
12 $\quad \mathcal{S}_o \leftarrow \mathcal{S}_o \cup (\hat{t}_1 + \delta\eta(\hat{\mathcal{S}}_o, \hat{t}_1), \hat{t}_2 + \delta\eta(\hat{\mathcal{S}}_o, \hat{t}_2), i, j, k)$;
13 Return $\mathcal{S}_o$;

`Reco-Mul` transforms a multi-coflow scheduling of ALG_p into a feasible scheduling (denoted as $\mathcal{S}_o$) in OCS. During the transformation, `Reco-Mul` first maps the scheduling $\mathcal{S}_p$ (obtained in Line 4) to the pseudo-time axis with regularization, which results in a regularized scheduling $\hat{\mathcal{S}}_o$ (Line 5 to Line 9). As $\hat{\mathcal{S}}_o$ is defined on the pseudo-time axis, which ignores the reconfiguration delay, we add the reconfiguration time into $\hat{\mathcal{S}}_o$ to get the final feasible multi-coflow scheduling in OCS, *i.e.*, $\mathcal{S}_o$ (Line 10 to Line 12). During the regularization, `Reco-Mul` *stretches* the start time of the flow transmissions in $\mathcal{S}_p$ to a multiple of $\sqrt{c}\cdot\delta$ (Line 6 and Line 7). Next we show the feasibility of the coflow scheduling returned by `Reco-Mul`.

Lemma 2. *The scheduling returned by* `Reco-Mul`, i.e. $\mathcal{S}_o$, *is a feasible coflow scheduling in OCS.*

[6]For instance, on the real time axis, from t_1 to t_2 the OCS is transmitting flow, halts for δ period of a reconfiguration and then resume transmission at $t_3 = t_2 + \delta$. Then, in our pseudo-time axis, $\hat{t}_1 = t_1$, $\hat{t}_2 = t_2$, while $\hat{t}_3 = \hat{t}_2 = t_2$ but not t_3.

Proof. For any two tuples $e_1 = (t_1, t_2, i_1, j_1, k_1)$ and $e_2 = (t_3, t_4, i_2, j_2, k_2)$ in $\mathcal{S}_p$ with port conflicts, these two elements' transmission time must not overlap with each other as $\mathcal{S}_p$ is a non-preemptive scheduling. We can assume $t_1 < t_2 \le t_3 < t_4$ without loss of generality.

With the regularization operations (Line 5 to Line 9 in Algorithm 2), we have $\hat{t}_1 = \left\lfloor \frac{t_1}{\sqrt{c}\delta} \cdot \frac{\lfloor\sqrt{c}\rfloor+1}{\lfloor\sqrt{c}\rfloor} \right\rfloor \cdot \sqrt{c}\delta$ and $\hat{t}_3 = \left\lfloor \frac{t_3}{\sqrt{c}\delta} \cdot \frac{\lfloor\sqrt{c}\rfloor+1}{\lfloor\sqrt{c}\rfloor} \right\rfloor \cdot \sqrt{c}\delta$. Thus, the time length between $\hat{t}_3$ and $\hat{t}_1$ is given by

$$
\begin{aligned}
\hat{t}_3 - \hat{t}_1 &= \left\lfloor \frac{t_3 \cdot \frac{\lfloor\sqrt{c}\rfloor+1}{\lfloor\sqrt{c}\rfloor}}{\sqrt{c}\delta} \right\rfloor \cdot \sqrt{c}\delta - \left\lfloor \frac{t_1 \cdot \frac{\lfloor\sqrt{c}\rfloor+1}{\lfloor\sqrt{c}\rfloor}}{\sqrt{c}\delta} \right\rfloor \cdot \sqrt{c}\delta \\
&\ge \left\lfloor \frac{t_2 \cdot \frac{\lfloor\sqrt{c}\rfloor+1}{\lfloor\sqrt{c}\rfloor}}{\sqrt{c}\delta} \right\rfloor \cdot \sqrt{c}\delta - \left\lfloor \frac{t_1 \cdot \frac{\lfloor\sqrt{c}\rfloor+1}{\lfloor\sqrt{c}\rfloor}}{\sqrt{c}\delta} \right\rfloor \cdot \sqrt{c}\delta \\
&\ge (t_2 - t_1) \cdot \frac{\lfloor\sqrt{c}\rfloor+1}{\lfloor\sqrt{c}\rfloor} - \sqrt{c}\delta \\
&= (t_2 - t_1) + \frac{1}{\lfloor\sqrt{c}\rfloor}(t_2 - t_1) - \sqrt{c}\delta \\
&\ge (t_2 - t_1) + \frac{\sqrt{c}}{c}(t_2 - t_1) - \sqrt{c}\delta.
\end{aligned}
$$

Recall that we assume $d_{i,j}^k \ge c \cdot \delta$. Thus we have

$$t_2 - t_1 \ge c \cdot \delta \quad \Rightarrow \quad \hat{t}_3 - \hat{t}_1 \ge t_2 - t_1,$$

Therefore, the new tuples $\hat{e}_1 = (\hat{t}_1, \hat{t}_2, i_1, j_1, k_1)$ and $\hat{e}_2 = (\hat{t}_3, \hat{t}_4, i_2, j_2, k_2)$ do not overlap each other in the time range, thus $\hat{\mathcal{S}}_o$ still preserve the port constraint. On the other hand, the transmission time in $\hat{\mathcal{S}}_o$ for each coflow remains the same as that in $\mathcal{S}_p$, thus $\hat{\mathcal{S}}_o$ also satisfies all demand matrices. Because injecting reconfiguration time does not violates the port constraint and the demand requirement, the final scheduling $\mathcal{S}_o$ is still feasible, which completes the proof. □

B. Theoretical Analysis

Denote OPT as the total weighted CCT of the optimal non-preemptive scheduling in OCS, *i.e.*, $OPT = \sum_{k=1}^{K} w_k T_k^*$ where T_k^* is the CCT of coflow k in OCS. We further denote $OPT^p = \sum_{k=1}^{K} w_k T_k^{*,p}$ as the total weighted CCT of the optimal scheduling in packet switches, where $T_k^{*,p}$ is the CCT of the coflow k in packet switches. As the optimal non-preemptive scheduling in packet switches, OPT, has no reconfiguration time, we have $OPT^p \le OPT$.

Denote $T_k^o = t_{trans}^k + t_{conf}^k$ as the CCT of coflow k in the scheduling $\mathcal{S}_o$, where t_{trans}^k and t_{conf}^k are the transmission time (including the time of waiting other coflows' transmissions) and the reconfiguration time of coflow k. Denote T_k^p as the CCT of coflow k in the non-preemptive scheduling in packet switches, *i.e.*, $\mathcal{S}_p$. With any non-preemptive multi-coflow scheduling in packet switches, in the following theorem we show `Reco-Mul` can transform it to a feasible scheduling in OCS by at most losing a constant factor.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on February 01,2023 at 04:39:41 UTC from IEEE Xplore. Restrictions apply.

Theorem 3. `Reco-Mul` *returns a feasible multi-coflow scheduling in OCS with an approximation ratio of* $\Delta \cdot \left(1+\frac{1}{\lfloor\sqrt{c}\rfloor}\right)^2$*, where* Δ *is the approximation ratio of* ALG_p*.*

Proof. The feasibility of the $\mathcal{S}_o$ is proved in Lemma 2. Now, we prove its approximation ratio.

The start time of each flow in $\mathcal{S}_o$ has been regularized to a multiple of $\sqrt{c}\delta$. Since reconfiguration only happens at the start of transmission, we have $t^k_{conf} \leq \frac{1}{\sqrt{c}} t^k_{trans}, \quad \forall k \in \mathcal{K}$. By adding t^k_{trans} at the both sides, we get

$$T^o_k = t^k_{conf} + t^k_{trans} \leq \left(1+\frac{1}{\sqrt{c}}\right) t^k_{trans}. \tag{1}$$

Recall the assumption of $d^k_{i,j} \geq c\delta$. After the regularization operations (Line 5 to Line 9 in Algorithm 2), the following inequality holds:

$$t^k_{trans} \leq (\frac{\lfloor\sqrt{c}\rfloor+1}{\lfloor\sqrt{c}\rfloor}) T^p_k. \tag{2}$$

This is because $\hat{t}_1$ is always not greater than $(\frac{\lfloor\sqrt{c}\rfloor+1}{\lfloor\sqrt{c}\rfloor})\cdot t_1$ (refer to Line 6 and Line 7 in Algorithm 2). Thus, by combining the Eqn. (1) and (2), we have

$$T^o_k \leq \left(1+\frac{1}{\sqrt{c}}\right)(\frac{\lfloor\sqrt{c}\rfloor+1}{\lfloor\sqrt{c}\rfloor}) T^p_k. \tag{3}$$

With weighted summing over all coflows, we have

$$\begin{aligned}\sum_{k\in\mathcal{K}} w_k T^o_k &\leq \left(1+\frac{1}{\lfloor\sqrt{c}\rfloor}\right)^2 \sum_{k\in\mathcal{K}} w_k T^p_k \\ &\leq \Delta\cdot\left(1+\frac{1}{\lfloor\sqrt{c}\rfloor}\right)^2 \sum_{k\in\mathcal{K}} w_k T^{*,p}_k \\ &\leq \Delta\cdot\left(1+\frac{1}{\lfloor\sqrt{c}\rfloor}\right)^2 \sum_{k\in\mathcal{K}} w_k T^*_k.\end{aligned} \tag{4}$$

This completes the proof. □

To the best of our knowledge, the multi-coflow scheduling algorithm proposed by Shafiee and Ghaderi in [17] has the best approximation ratio in packet switches (4-approximate, *i.e.*, $\Delta = 4$). So, we can give the approximation ratio of `Reco-Mul` in the following corollary.

Corollary 1. `Reco-Mul` *is* $4\cdot\left(1+\frac{1}{\lfloor\sqrt{c}\rfloor}\right)^2$*-approximate with the algorithm given by Shafiee and Ghaderi [17].*

V. Performance Evaluation

In this section, we conduct extensive simulations to evaluate `Reco-Sin` and `Reco-Mul` based on real workloads. Our results and analysis indicates that our algorithms *always* outperform the state-of-the-art baselines. We highlight the simulation results as follows:

- `Reco-Sin` outperforms Solstice [7], the well-known single coflow scheduling algorithm, in light of the reconfiguration frequency and the CCT. When we varying the coflow density, `Reco-Sin` reduces reconfiguration frequency and CCT up to 86.4% and 15.9%, respectively. Under different values of δ, `Reco-Sin` can achieve up to 96.94% performance enhancement in the CCT (Sec.V-C).
- `Reco-Mul` outperforms existing schemes in the scenarios of weighted or unweighted CCT, which reduces the CCT by at least 60.31% and 78.76% compared with LP-II-GB and SEBF+Solstice respectively (Sec.V-D).
- Under different values of important parameters (e.g., δ and c), `Reco-Mul` maintains its advantages and can reduce the CCT by up to 73.26% (Sec.V-D).

TABLE I
COFLOW TYPES WITH DIFFERENT DENSITY OF THE DEMAND MATRIX

Density	Sparse	Normal	Dense
Percentage (%)	86.31	5.13	8.56

A. Methodology

Workload: The workload that consists of 526 coflows is based on a Hive/MapReduce trace collected on a 3000-machine 150-rack cluster with 10:1 oversubscription ratio. The trace contains the coflow information which are its arrival time and the size of shuffle data at the reducers. Since there is no information about the size of the data flow from the mappers to the reducers, we uniformly allocate the shuffle data of reducers to the mappers, similar to [14]. Similar to Sunflow [9], we extract each record as a coflow demand matrix, whose rows and columns represent the mappers and reducers, respectively. In addition, we add $\pm 5\%$ perturbation of flow size to simulate the real production environment. Coflows are categorized into three types based on the density of their demand matrices as follows. We call the density of coflow demand matrix as DS. We then define 3 types of coflows as follows: 1) Sparse: DS$\leq$ 0.05, 2) Normal: $0.05 \leq DS \leq 0.5$, and 3) Dense: $DS \geq 0.5$. We show in Table I) the proportion of each coflow type.

Coflow Transmission Mode: We categorize the coflows according to the following transmission modes: 1) a *Single-to-Single (S2S)* coflow contains only one flow from one single ingress port to one egress port; 2) a *Single-to-Multiple (S2M)* coflow contains flows from a single ingress port to multiple egress ports; 3) a *Multiple-to-Single (M2S)* coflow consists of flows from multiple ingress ports to one single egress port, and 4) a *Multiple-to-Multiple (M2M)* coflow consists of flows from multiple ingress ports to multiple egress ports. We hence categorize the coflows into four types summarized in Table II.

TABLE II
CATEGORY OF COFLOWS WITH DIFFERENT TRANSMISSION MODE

Modes	S2S	S2M	M2S	M2M
Numbers%	23.38	9.89	40.11	26.62
Sizes%	0.005	0.024	0.028	99.943

S2S, S2M and M2S transmission modes: Each flow in these types shares one (ingress or egress) port with all other flows.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on February 01,2023 at 04:39:41 UTC from IEEE Xplore. Restrictions apply.

The optimal approach is to schedule them one by one. Hence, both Solstice and `Reco-Sin` can achieve the optimal CCT.

M2M transmission modes: Table II shows that the majority of the coflows in the workload are the M2M transmission type. Their performance is critical to the overall performance. We will see later in this section the difference between Solstice and `Reco-Sin` in their performances to minimize the CCT.

Simulator: We develop a trace-driven flow-level simulator to perform various algorithms based on the embedded LP solver GUROBI [22]. Our experiments are conducted on a server with a Intel Xeon E5-2620v4 CPU with 32 Gigabyte memory and 2TB hard drive.

Metrics: Our metrics is the normalized average (95-percentile) weighted or unweighted CCT of a scheme compared with our algorithms `Reco-Sin` or `Reco-Mul`. For example, the Normalized CCT of Algorithm A is defined as

$$\text{Normalized CCT} = \frac{\text{the CCT of Algorithm A}}{\text{the CCT of } \texttt{Reco-Mul}}.$$

The lower the normalized CCT is, the better the performance of Algorithm A is.

B. Baselines

We compare the performances of our proposed algorithms `Reco-Sin` and `Reco-Mul` with the following baselines for single and multiple coflow scheduling in minimizing the (weighted) CCT.

1. Lower Bound: the theoretical lower bound on the CCT of a single coflow in the OCS, $T_{lb} = \rho + \tau\delta$, where ρ is the maximum value of the sum of the matrix row values and the sum of the column values, and τ is the maximum number of nonzero entries in rows or columns in matrix.

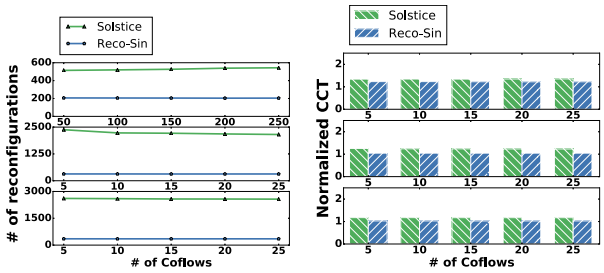
2. Solstice [7]: the state-of-the-art circuit scheduling algorithm which operates in two steps: first, it transforms the demand matrix D_C into a $k-$biostochastic matrix (*stuffing*) and second, it iteratively computes a scheduling of configurations (*slicing* operation) to complete the transmission.

3. SEBF [11] + Solstice [7]: Smallest-Effective-Bottleneck-First (SEBF) [11] is to prioritize the coflow with the smallest effective bottleneck flow. However, SEBF can not be applied to the OCS directly as the bandwidth allocation operation can not be used in the context of OCS. Thus, we modify it a bit and combine it with Solstice for multiple coflow scheduling. For single coflow scheduling in OCS, we leverage the aforementioned Solstice.

4. LP-II-GB [16]: LP-II-GB is designed for multiple coflow scheduling problem. It is mainly based on a time interval indexed linear program which can be solved in polynomial time. The algorithm solves a relaxed linear program capturing the primitive problem, and the optimal solution is used to derive an estimation value of the CCT for each coflow. Based on these estimation value computed, it determines the scheduling order of the coflows. For single coflow scheduling, they adopt the BvN method.

C. Evaluation of `Reco-Sin`

We set the parameters of the optical switches according to the practical scenarios [5], [6]. The link bandwidth is set as 100Gbps. The value of the reconfiguration delay, δ, ranges from 100ms to 100μs, and its default value is set as 100μs.

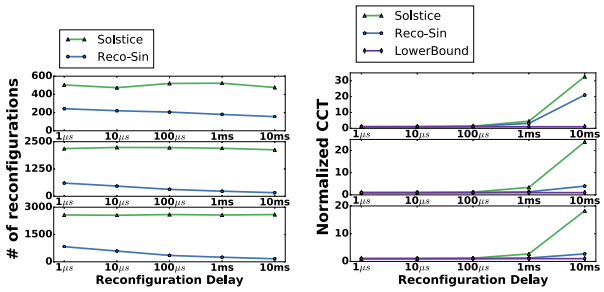


(a) Reconfiguration Difference (b) CCT Difference

Fig. 4. Performance in single coflow scheduling.

1) Reconfiguration frequency: The density of demand matrix can greatly impact the reconfiguration frequency of coflow. Thus, we fix δ as the default value, and compare `Reco-Sin`'s reconfiguration frequency with Solstice's. (In Fig. 4(a), Fig. 4(b), Fig. 5(a) and Fig. 5(b), from top to bottom, coflow's density is sparse, normal and dense.) In general, we can observe from Fig. 4(a) that `Reco-Sin` has a lower reconfiguration frequency than Solstice does. When the demand matrix of coflow is sparse, normal and dense, Solstice spends 2.58×, 7.07× and 7.36× more reconfigurations than `Reco-Sin`, respectively. Note that the performance gap becomes larger with the density increasing. The reason is that the number of permutation matrix after BvN decomposition grows along with the increasing of coflow's density. We can conclude that `Reco-Sin` outperforms Solstice in optimizing reconfiguration times for all types of coflows.

2) CCT: In this section, we evaluate the performance of `Reco-Sin` and Solstice in minimizing the CCT. We observe from Fig. 4(b) that Solstice needs more time to finish the same coflow than `Reco-Sin` does. Specifically, Solstice needs 1.19×, 1.15× and 1.14× more time than `Reco-Sin` to decompose demand matrix when scheduling *sparse*, *normal* and *dense* coflows respectively. This is because of the advantage of less reconfiguration frequency `Reco-Sin` has and that the proportion of reconfiguration time in CCT declines with the increase of transmission time (i.e., the increase of demand matrix density). In short, `Reco-Sin` outperforms Solstice in minimizing CCT when scheduling one single coflow.



(a) Reconfiguration Difference (b) CCT Difference

Fig. 5. Performance in single coflow scheduling.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on February 01,2023 at 04:39:41 UTC from IEEE Xplore. Restrictions apply.

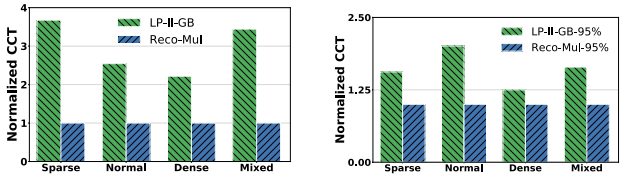
3) Impact of δ: The variation of δ is an important characteristic decided by the hardware of OCS, and it impact the CCT directly. Similar to previous simulations, from the perspective of reconfiguration times and CCT, we analyze the strength and weakness of `Reco-Sin` and Solstice.

The curves in Figure.5(a) present that Solstice spends more time on reconfiguration to complete the same coflow than `Reco-Sin`. In fact, the variation of δ would not have effect on reconfiguration frequency of Solstice. The Figure.5(a) depicts when coflow density changes, the number of reconfigurations of Solstice basically remain invariable when δ changes. Meanwhile, the amount of reconfigurations of `Reco-Sin` declines with the increase of δ. The main reason is that our regularization operation is directly correlated with δ. When δ increases, the elements in the coflow demand matrix will be more aligned after regularization, which in turn reduces the reconfiguration time. When decomposing sparse coflow, Solstice spends $2.10 \sim 3.10\times$ reconfigurations as many as `Reco-Sin` does. When the matrix is not sparse, the reconfiguration demands of Solstice can be $7.55 \sim 8.12\times$ as many as that of `Reco-Sin`. Figure. 5(b) illustrates their performance on CCT. In our experiments, we adopt the theoretical lower bound of CCT as the normalized benchmark and indicate performance of `Reco-Sin` and Solstice through their difference. When the coflow demand matrix grows from sparse to dense, the advantage of `Reco-Sin` is declining. The reason is that the proportion of reconfiguration time in CCT decreases when the coflow matrix is more sparse. Meanwhile, regardless of the demand matrix density, the gap between `Reco-Sin` and Solstice expands when δ increases. It is due to that the reconfiguration time becomes dominant in CCT with the increasing of δ, and the advantage of `Reco-Sin` becomes more prominent as it has less frequent reconfigurations. As for the different coflows shown in Fig. 5(b) , Solstice can consume $32.66\times$, $23.89\times$ and $18.26\times$ time as much as the lower bound, while the CCT of `Reco-Sin` consumes only $21.00\times$, $3.96\times$ and $2.72\times$, respectively.

D. Evaluation of `Reco-Mul`

In this section, we present our evaluation on `Reco-Mul`

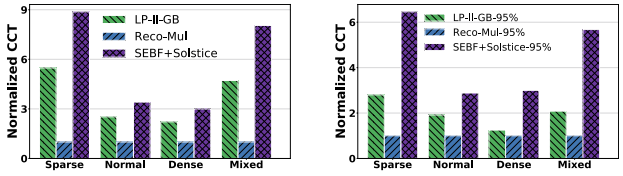
1) Weighted CCT: SEBF+Solstice can be used to optimize the unweighted CCT, while `Reco-Mul` and LP-II-GB are for weighted (and, of course, unweighted) CCT. Thus, in this part, we compare the performance of `Reco-Mul` with LP-II-GB's in minimizing weighted CCT. The coflow weights are set uniformly from $[0, 1]$. Fig. 6 shows that, `Reco-Mul` outperforms LP-II-GB regardless of the coflow density level. Specifically, for the cases of sparse, normal and dense coflows, `Reco-Mul` has $72.75\%(35.85\%)$, $60.62\%(50.17\%)$ and $54.75\%(19.91\%)$ performance improvement than LP-II-GB in minimizing the average (95-percentile) weighted CCT. When multiple coflows contain all the coflows (i.e., coflows with all kinds of density level), `Reco-Mul` is $3.44\times(1.64\times)$ better than LP-II-GB. The reason of `Reco-Mul`'s advantage is basically the alignment of the flow start time. While coflow's density increases, the effect of the start time alignment on the performance reduces, which accounts for the performance differences among the cases with different coflow sparsities.



(a) Average (b) 95th percentile

Fig. 6. Performance in minimizing weighted CCT for multiple coflows.

2) Unweighted CCT: In this part, we set the coflow weights equally and compare `Reco-Mul`'s, LP-II-GB's and SEBF+Solstice's performances in minimizing unweighted CCT. The results are shown in Fig. 9. SEBF+Solstice has the worst performance; LP-II-GB comes the second and `Reco-Mul` performs the best. When coflows are sparse, SEBF+Solstice is $8.87 \times (6.56\times)$ worse than `Reco-Mul`, and LP-II-GB is $5.47 \times (2.80\times)$ worse than `Reco-Mul`. For the cases of normal and dense coflows, the improvements of `Reco-Mul` over LP-II-GB and SEBF+Solstice are $2.52 \times (1.91\times)$ and $3.41 \times (2.88\times)$ respectively. When scheduling all (i.e., mixed) kinds of coflows, LP-II-GB and SEBF+Solstice need $4.71 \times (2.08\times)$ and $8.04 \times (5.67\times)$ more time than `Reco-Mul`, respectively. The reason of the above performance is similar to the one when minimizing the weighted CCT. Due to the limited space, we omit the details. In short, `Reco-Mul` outperforms LP-II-GB and SEBF+Solstice in minimizing weighted and unweighted CCT.



(a) Average (b) 95th percentile

Fig. 7. Performance in minimizing unweighted CCT for multiple coflows.

3) Reconfiguration frequency: In Sec.V-C, we present that our proposed `Reco-Sin` needs less circuit reconfigurations than Solstice does. In this part, we compare the performance of `Reco-Mul` and LP-II-GB, in minimizing the reconfiguration frequency. Fig. 8 presents our experiment results. Specifically, if we consider the case where there are coflows of all different density levels, the reconfiguration time of LP-II-GB is $2.59\times$ larger than `Reco-Mul`'s. When scheduling the sparse, normal and dense coflows, the performance gaps between `Reco-Mul` and LP-II-GB are $4.37\times$, $2.56\times$ and $1.48\times$ respectively. In brief, when the coflow density increases, the performance gap between `Reco-Mul` and LP-II-GB decreases. Because higher coflow density reduces the proportion of fragmentary flow demand, thereby diminishes the advantage of `Reco-Mul`.

4) Impact of δ: Similar to the simulation of `Reco-Sin`, δ is also a crucial variable which can affect the performance

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on February 01,2023 at 04:39:41 UTC from IEEE Xplore. Restrictions apply.

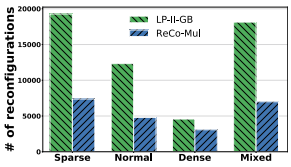
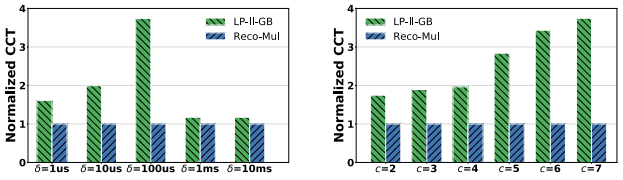


Fig. 8. Comparison of reconfiguration frequency for multiple coflows.

of Reco-Mul. In Reco-Mul, a critical step is stretching the scheduling order of flows to eliminate their conflict on transmission time. Moreover, δ is directly relevant of this stretch process. According to Fig. 9(a), when δ increases from $1us$ to $100us$, Reco-Mul gradually expands its advantages. For instance, when $\delta = 1us$, LP-II-GB needs $1.61\times$ time to finish multiple coflow scheduling than Reco-Mul. When $\delta = 10us$ and $\delta = 100us$, Reco-Mul need only 50.25% and 26.74% time of LP-II-GB. Because the reconfiguration frequency of Reco-Mul will decline while δ enlarges. In the meantime, due to reconfiguration time is a product of reconfiguration frequency and δ, generally, the reconfiguration time of Reco-Mul declines while δ increases to $100us$. On the other hand, reconfiguration frequency of LP-II-GB will not be affected by the variation of δ, so CCT of LP-II-GB becomes larger with δ's enlargement. However, we observe that the gap between Reco-Mul and LP-II-GB drops when δ increases to $1ms$ and $10ms$. The reason is the excessive growth of δ. When δ increases to ms, the reconfiguration time becomes dominant in CCT. The rapid growth of δ weakens the benefit from reconfiguration frequency, so Reco-Mul is not able to take advantage of less reconfiguration frequency. If δ approaches infinity, the performance gap between Reco-Mul and LP-II-GB will approach to the ratio of their reconfiguration frequency. When δ equals to $1ms$ and $10ms$, Reco-Mul is $1.17\times$ and $1.18\times$ better than LP-II-GB.

In conclusion, Reco-Mul is able to maintain the advantages when varying the δ's. When the reconfiguration delay is $100us$, Reco-Mul performs the best. As a matter of fact, our conclusion reflects that Reco-Mul is adaptable to different kinds of industrial OCS with different configuration delays.



(a) various δ (b) various c

Fig. 9. The impact of δ and c in multiple coflow scheduling.

5) Impact of the constant c: Recall that we assume that there is no tiny flow in OCS, and any traffic demand is at least $c \cdot \delta$ (Sec. II). In practice, the value of c corresponds to the real industrial environment. As shown in Fig. 9(b), obviously the advantage of Reco-Mul gradually is strengthened when c increases. For c=2 to 4, the time required by LP-II-GB for scheduling increases from $1.74\times$ to $1.96\times$. When c ranges from 5 to 7, Reco-Mul strengthens its advantages in minimizing CCT from $2.83\times$ to $3.744\times$. Thus, the simulation result validates our theoretical analysis. The reconfiguration frequency and time of Reco-Mul will decrease along with the increase of c by the stretching operation. Consequently, the CCT of Reco-Mul will continuously decrease.

VI. Discussion

Mice Flows: OCS is designed for the high-throughput transmission due to its large bandwidth, and the reconfiguration process is a natural handicap to transmit mice flows. Let τ be the maximum number of non-zero entries of each row or column of a demand matrix, then $\tau\delta$ is a lower bound of the total reconfiguration delay, which even the optimal scheduling in OCS can not avoid. It implies that OCS is not ideal to transmit mice flows. Thus, in practical systems, only elephant flows are transferred through the OCS and mice flows can be handled more efficiently by packet switches [7]. Therefore, in this work, we assume that there are no tiny flows in OCS.

Not-all-stop OCS: As stated in Sec. II-A, we consider *all-stop* OCS commonly used in previous works [6]–[8], [10]. Another circuit establishment model is *not-all-stop* as adopted by Sunflow [9], where during reconfiguration, communication stops only on the affected ports including the ports to be set up and those to be torn down, while the circuits unchanged can keep transmission. Based on the current manufacturing technique, a pure non-all-stop OCS is still hard to implement. Intuitively, a feasible scheduling in the *all-stop* model is still feasible in the *not-all-stop* model. Moreover, our proposed algorithm Reco-Mul can have the same approximation ratio in the *not-all-stop* model. Note that Eqn. 4 in the approximation ratio proof of Reco-Mul(Theorem 3) is still correct by replacing the optional weighted CCT in the *all-stop* model with that in the *not-all-stop*. Therefore, as shown in Table III, combining the results of Sunflow, the coflow scheduling in both OCS models can achieve constant approximations.

TABLE III
Approximation Ratios for Coflow Scheduling in OCS

Model / Algorithm	Single coflow		Multiple coflow	
	N	A	N	A
Sunflow [9]	2	-	-	-
Reco-Sin, Reco-Mul	-	2	$4 \cdot f(c)$	$4 \cdot f(c)$

N: Not-all-stop model, A: All-stop model, $f(c) = \left(1 + \frac{1}{\lfloor\sqrt{c}\rfloor}\right)^2$

VII. Related Work

In this section, we present related works on circuit scheduling and coflow scheduling. We list some of the most representative related works and make a comparison in Table IV.

A. Circuit Scheduling:

Helios [5] and *c-Through* [8] proposed hybrid circuit/packet switch architectures. Both focused on the circuit management to minimize the flow completion time. They adopted the well-known Edmonds maximum weighted matching algorithm to

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on February 01,2023 at 04:39:41 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
COMPARISON AMONG THE RELATED WORK AND OUR RESULTS

Solutions	Coflow-aware	OCS-enable	Performance Guarantee
Qiu et al. [16], Chen et al. [23], Shafiee et al. [17], Sincronia [24] and etc.	✓	✗	✓
Varys [11], Aalo [12],CODA [13]	✓	✗	✗
Heilos [5], c-through [8], Porter et al. [10], Liu et al. [7], and etc.	✗	✓	✗
Sunflow [9]	✓	✓	Single Coflow ✓ Mulitple Coflows ✗
`Reco`[this work]	✓	✓	Single Coflow ✓ Mulitple Coflows ✓

schedule the circuits [18]. Later, Porter *et al.* [10] proposed an improved circuit scheduling algorithm, called Traffic Matrix Scheduling (TMS). Borrowing the idea of BvN [25], [26], their algorithm decomposed the traffic demand matrix into permutation matrix associated with a series of circuit assignments for a predetermined period of time. However, BvN decomposition did not work well in minimizing the number of configurations, which could incur unavoidable and significant reconfiguration delays as we claimed in Sec. III-A. Paper [27] achieved an approximation algorithm for circuit scheduling under specific settings, where the demands and reconfiguration time were integers, and the minimum demand transmission time was assumed as an integral multiple of the reconfiguration time. Liu *et al.* [7] developed a flow scheduling algorithm called *Solstice* to minimize the maximum flow completion time particularly for the hybrid circuit/packet networks. Solstice effectively improved the circuit utilization to reduce the number of configurations, and thus outperformed BvN. Our algorithm `Reco-Sin`, compared with Solstice, reduces the CCT with even less frequent reconfigurations. and guarantees a performance with a constant approximation ratio.

B. Coflow Scheduling

Most previous works on network-level optimization were agnostic on the application-level performance metrics (*e.g.*, [28]). which introduces negative impact on the application performance (examples can be found in [4], [11], [12]). The *coflow* abstraction was first proposed in [4] to bridge the above gap. Some existing works on coflows focused on minimizing the average CCT (see [4], [11]–[14], [16], [23], [24], [29]–[32]). Varys [11] proposed effective heuristics to schedule coflows aiming at minimizing the average CCT. Without prior knowledge of coflows, Aalo [12] adopted Discretized Coflow-Aware Least-Attained Service (D-CLAS) to separate coflows into priority queues based on the amounts they have already sent. CODA [13] was the first work to recognize coflows among individual flows using machine learning techniques. An error-tolerant coflow scheduler was then proposed and implemented. The above papers studied heuristics-based solutions and mainly focused on system implementation, which are lack of theoretical performance guarantees. Qiu *et al.* [16] proposed the first deterministic algorithm with a constant approximation ratio for multiple coflow scheduling, which defined the flow completion time as the duration from a common original time point to the completion of the flow regardless of its real arrival time. Paper [17] further improved the approximation ratio in the setting that the coflows arrived to the network at the same time. Essentially, the solutions in [16] and [17] determined the scheduling order of the coflows based on the solutions to a carefully formulated linear program. Li et al. [14] studied the coflow scheduling and routing problem, and proposed the first online algorithm with performance guarantees for the problem. Liang and Modiano [33] assumed stochastic coflow arrivals, and studied the optimal scaling of coflow-level delay in an $N \times N$ input-queued switch as $N \to \infty$. Chen *et al.* [23] studied the utility optimization for coflow scheduling, who proposed a utility-based scheduler to provide differential treatment to coflows with different degrees of sensitivity. Tian *et al.* [29] studied dependent coflows of multi-stage jobs to minimize total weighted job completion time. Utopia [30] focused on the near-optimal coflow scheduling with provable isolation guarantee. Sincronia [24] presented a near-optimal network design for coflows that can be implemented on top of any transport layer to support priority scheduling. Huang *et al.* [9] might be the first to consider coflow scheduling in optical circuit switches, which presented a constant approximation for single coflow scheduling and a heuristic to schedule multiple coflows in the not-all-stop model. Paper [34] investigated coflow scheduling in OCS under online settings, where a heuristic algorithm was proposed without any theoretical performance guarantee. To the best of our knowledge, our algorithm, `Reco-Mul` is the first approximation algorithm for multiple coflow scheduling in OCS.

VIII. CONCLUSION

In this paper, we study the coflow scheduling problem in the context of optical circuit switches enabled data centers. We first propose a novel operation called *regularization* which is simple but effective to avoid frequent circuit reconfigurations. Then, for single coflow scheduling, we present `Reco-Sin`, an efficient 2-approximation algorithm. As for multiple coflow scheduling, we present `Reco-Mul`, based on the regularization operation, which achieves a constant approximation ratio to the optimum as well. Extensive simulations based on real data traces show that our proposed algorithms significantly outperform the state-of-the-art schemes in optimizing the weighted CCT. In practical systems, the information of a coflow might be known only when it arrives to the network. To this end, one interesting future direction is to derive online coflow scheduling schemes for OCS-based networks.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on February 01,2023 at 04:39:41 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[2] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," in *ACM SIGOPS/Eurosys*, 2007.

[3] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster Computing with Working Sets," in *HotCloud*, 2010.

[4] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *ACM HotNets*, 2012.

[5] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," in *ACM SIGCOMM*, 2010.

[6] C.-H. Wang, T. Javidi, and G. Porter, "End-to-end scheduling for all-optical data centers," in *IEEE INFOCOM*, 2015.

[7] H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen, M. Kaminsky *et al.*, "Scheduling techniques for hybrid circuit/packet networks," in *ACM CoNEXT*, 2015.

[8] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," in *ACM SIGCOMM*, 2010.

[9] X. S. Huang, X. S. Sun, and T. E. Ng, "Sunflow: Efficient optical circuit scheduling for coflows," in *ACM CoNEXT*, 2016.

[10] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," in *ACM SIGCOMM*, 2013.

[11] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in *ACM SIGCOMM*, 2014.

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

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

[14] Y. Li, S. H.-C. Jiang, H. Tan, C. Zhang, G. Chen, J. Zhou, and F. Lau, "Efficient online coflow routing and scheduling," in *MobiHoc*, 2016.

[15] H. Liu, F. Lu, A. Forencich, R. Kapoor, M. Tewari, G. M. Voelker, G. Papen, A. C. Snoeren, and G. Porter, "Circuit switching under the radar with reactor." in *USENIX NSDI*, 2014.

[16] Z. Qiu, C. Stein, and Y. Zhong, "Minimizing the total weighted completion time of coflows in datacenter networks," in *SPAA*, 2015.

[17] M. Shafiee and J. Ghaderi, "An improved bound for minimizing the total weighted completion time of coflows in datacenters," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 4, pp. 1674–1687, 2018.

[18] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of mathematics*, vol. 17, no. 3, pp. 449–467, 1965.

[19] M. Marcus and R. Ree, "Diagonals of doubly stochastic matrices," *The Quarterly Journal of Mathematics*, vol. 10, pp. 296–302, 1959.

[20] F. Dufossé and B. Uçar, "Notes on birkhoff–von neumann decomposition of doubly stochastic matrices," *Linear Algebra and its Applications*, vol. 497, pp. 108–115, 2016.

[21] R. A. Brualdi, "Notes on the birkhoff algorithm for doubly stochastic matrices," *Canadian Mathematical Bulletin*, vol. 25, no. 2, pp. 191–199, 1982.

[22] GUROBI, "Gurobi," http://www.gurobi.com.

[23] L. Chen, W. Cui, B. Li, and B. Li, "Optimizing Coflow Completion Times with Utility Max-Min Fairness," in *IEEE INFOCOM*, 2016.

[24] S. Agarwal, S. Rajakrishnan, A. Narayan, R. Agarwal, D. Shmoys, and A. Vahdat, "Sincronia: near-optimal network design for coflows," in *Proc. of ACM SIGCOMM*, 2018.

[25] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser. A*, vol. 5, pp. 147–151, 1946.

[26] J. Von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem," *Contributions to the Theory of Games*, vol. 2, pp. 5–12, 1953.

[27] B. Towles and W. J. Dally, "Guaranteed scheduling for switches with configuration overhead," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 835–847, 2003.

[28] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal Near-Optimal Datacenter Transport," in *ACM SIGCOMM*, 2013.

[29] B. Tian, C. Tian, H. Dai, and B. Wang, "Scheduling coflows of multi-stage jobs to minimize the total weighted job completion time," in *IEEE INFOCOM*, 2018.

[30] L. Wang, W. Wang, and B. Li, "Utopia: Near-optimal coflow scheduling with isolation guarantee," in *IEEE INFOCOM*, 2018.

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

[32] S. Im, M. Shadloo, and Z. Zheng, "Online partial throughput maximization for multidimensional coflow," in *IEEE INFOCOM*, 2018.

[33] Q. Liang and E. Modiano, "Coflow Scheduling in Input-Queued Switches: Optimal Delay Scaling and Algorithms," in *INFOCOM*, 2017.

[34] C. Xu, H. Tan, J. Hou, C. Zhang, and X.-Y. Li, "Omco: Online multiple coflow scheduling in optical circuit switch," in *Proc. of IEEE ICC*, 2018.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on February 01,2023 at 04:39:41 UTC from IEEE Xplore. Restrictions apply.