Coflow Fair Scheduling via Dynamic Progress

***Abstract*—The average coflow completion time (CCT) is the standard performance metric in coflow scheduling. However, standard CCT minimization may introduce unfairness between the data transfer phase of different computing jobs. Static progress guarantees have been introduced in the literature to mitigate this fairness issue, but the trade-off between fairness and efficiency of data transfer seems hard to control. In this paper we introduce a new fairness framework for coflow scheduling based on the concept of *elongation*, a parameter to measure the performance degradation experienced by a coflow compared to isolation. Elongation-based scheduling is proved to offer a dynamic and more flexible means to control the average progress of coflows while minimizing the average CCT. We design an algorithmic solution chosen in the class of the σ-order schedulers to solve the fair coflow scheduling problem in polynomial time. Under feasibility assumptions, the algorithm is proved to be a 4-approximation w.r.t. an optimal scheduler. Extensive numerical results validate the proposed scheme and demonstrate that this approach can trade off average CCT for per-coflow elongation.**

***Index Terms*—data transfer, coflow scheduling, fairness, progress, primal-dual scheduler**

I. Introduction

The coflow abstraction has been introduced in the seminal paper of Chowdhury and Stoica [1]. A coflow denotes a group of flows produced by data-intensive computing frameworks, e.g., Map Reduce, Giraph and Spark [2]–[4], during their data transfer phases. Similarly, the workflows of most data-intensive machine learning applications are based on data-transfer and operated by distributing tasks of the same application over hundreds of individual machines. A well-studied example of data transfer step is the *shuffle phase* of Hadoop MapReduce. The shuffle phase acts as a barrier synchronization since the next phase of the computation cannot be pursued until the end of the shuffle data transfer. Once each mapper node finishes running their job, partial results are fetched in the form of HTTP connections by peer reducer nodes and final results can be generated for the next computation phase or storage. At the application level, the shuffle phase has been shown to represent a significant part of the total computation time [5]. For this reason, efficient coflow scheduling has become key in modern datacenter engineering [6]–[10].

The standard metric to measure coflow scheduling performance is the average Coflow Completion Time (CCT), i.e., the average time by which the last flow of a coflow is completed. Minimizing the average CCT is indeed the appropriate goal in order to increase the number of computing jobs dispatched in a datacenter. The related minimization problem is complicated by the fact that a shared datacenter fabric can be contended by hundreds of coflows at the same time. Thus, multiple congested links may appear under concurrent demands. In the last ten years, the problem of average CCT minimization has been addressed by several authors [1], [5], [8], [11], shading light on its complexity and devising several algorithmic solutions. The problem is found *NP*-hard by reduction to the job scheduling problem on multiple correlated machines, a mainstream operation research problem. While inapproximability below a factor 2 has been proved [7], the best to date deterministic approximation ratio is 4 in the case when coflows are released at same time [8], [11], [12].

In the literature, minimizing the average flow completion time is known to entail potential unfairness among different flows. Thus, the notion of per-flow fairness has been studied to balance the resource allocation, i.e., link utilization, among different flows. Max-min fairness and proportional fairness are reference concepts in this context and a series of fundamental works on utility-based fairness have showed that both the notions can be compounded under the larger concept of α-fairness [13]. Indeed, fairness issues exist in the context of coflow scheduling as well [14] [15] [16] [17].

Coflow fairness. To obtain a resource allocation able to achieve a required trade-off between average CCT and fairness, it is tempting to directly use CCT as the variable in conventional fairness utility functions. But, two major obstacles to this approach exist. First, the definition of CCT entails a minimization: using CCT as argument of standard smooth convex functions such as the α-fair utility leads to minimization problems neither convex nor differentiable. The workaround is to allow for a constant rate allocation [16]. But, standard utility-based per flow fairness decides stationary rates as arguments, while CCT minimization is a finite horizon problem for which the analogy results in suboptimal resource allocation. In this work we parametrize the trade-off while allowing dynamic rate allocation.

Fairness in coflow scheduling relies on the notion of *coflow progress*, which settles the notion of fairness in the max-min sense, i.e., by guaranteeing a minimum static resource allocation to coflows [15]–[17]. In particular, [16] studied the trade-off between performance and fairness by showing that there exists a trade-off between minimum coflow progress and average CCT. In practice, since coflow links have intertwined dependencies, the seminal idea for this approach builds on the notion of DRF (Dominant Resource Fairness) [18] – used also in YARN [19] in order to allocate computing resources. The progress of a coflow tracks the slowest flow – determining the CCT – on the related bottleneck port at each instant; all the remaining flows of the coflow are slowed down in order to improve network utilization without impairing the CCT.

The above notion of fairness for coflow scheduling has limitations and it can be substantially improved. In fact, first, it is defined *per port* based on the bottlenecks seen by coflows,

e.g., the switch links they engage during the shuffle phase. As detailed formally in the next section, such a definition does not account for the structure of coflows. Second, such static allocation corresponds to a non-preemptive definition of progress since it requires every coflow to send a minimum amount of data with no interruption over each engaged port. Any further coflow completion time optimization is pursued on top of this baseline constant rate allocation [16].

Main contributions. In order to overcome the above limitations, in this paper we introduce a new measure of fairness called elongation. The elongation measures the effect of coflow scheduling on individual data transfer time. A fair coflow scheduler should be able to mitigate the elongation experienced by a coflow with respect to isolation. We first prove that this simple notion of fairness is more general than the notion of static per port progress used in the literature and characterize the feasibility of the resulting general scheduler. Hence, we provide an algorithm in the class of σ-order coflow schedulers introduced in [8]. The objective is to attain the target dynamic progress per coflow: in order to do so we use a primal-dual algorithm whose output is a *bottleneck-feasible* σ-order, i.e., it attains the target dynamic progress per coflow, and an approximation factor of 4. To the best of the authors' knowledge, the notion of fairness in data-transfer and the solutions provided in this paper are new contributions to the discussion on enforcing fairness in datacenter networks.

The paper is organized as follows. Sec. II resumes existing works for coflow fairness. In Sec. III we describe the system model and we introduce the notion of dynamic fairness based on coflow elongation. In Sec. IV the framework is formalized as a scheduling problem with elongation constraints and its feasibility is discussed. Algorithmic solutions are hence proposed in Sec. V in the set of bottleneck-feasible σ-order schedulers. Sec. VI reports on numerical results and a concluding section ends the paper.

II. Related works

In the coflow literature only a few papers deal with fairness issues. The seminal work [1] introduced the weighted CCT as objective function and later works pursued same approach, e.g., [20], even though the relation between coflow weights, CCT and coflow fairness remains elusive. In general, fairness metrics based either on weights or on static rate allocation [15], [20] cannot account for the dynamic nature of coflow bandwidth occupation. In the scheduler Varys [21] coflow prioritization has been introduced as a means to mitigate starvation of coflows with low priority. Thus, reserving a fraction of port bandwidth per coflow became the accepted notion of coflow progress adopted later on in the literature [16], [17]. An interesting attempt to develop a formal framework based on max-min fairness has been presented [14], extending the usage of lexicographic ordering used in flow-level rate allocation [22]. However, due to its ease of implementation, the notion of coflow progress has been preferred in the literature. Other authors [23] define the fairness degree of a data-transfer scheme with respect tostandard schemes such as Hug [17] or DFR [18].

Symbol	*Meaning*
$\mathscr{L}$	set of switch links $k = 1, \dots, 2M$
$\mathscr{M}$	coflow schedule
$\mathscr{C}$	set of coflows $\mathscr{C} = \{1, \dots, N\}$
$\mathscr{F}_j$	set of flows of coflow j; $n_j = \lvert\mathscr{F}_j\rvert$ (coflow width)
C_j	coflow completion time for coflow j
w_j	weight of coflow j
r_j	release time of coflow j
E	elongation constraint
$x_j^i(t)$	fraction of flow i of coflow j scheduled in slot t
$Y_j^i(t)$	progress of flow i of coflow j at time t
$Z_j(t)$	completed fraction of coflow j at time t
$Z_j^i(t)$	completed fraction of flow i of coflow j at time t
$x_j^i(t)$	fraction of flow i of coflow j at time t
$X_j(t)$	indicating variable for coflow j at slot t (binary)
$X_j^i(t)$	indicating variable for for i of. coflow j at slot t (binary)
B_ℓ	capacity of link ℓ
T	time horizon
Δ	time slot duration

TABLE I
List of Key Notations

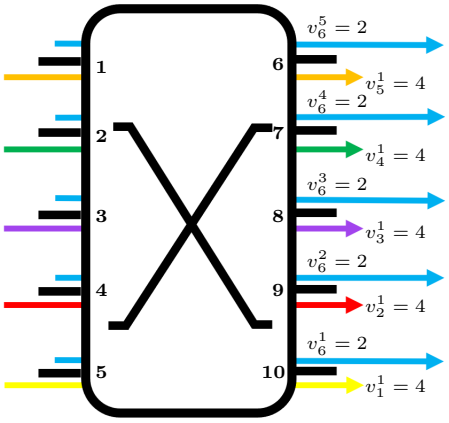
The definition of static coflow progress described in Sec. III has been formalized in [17]. The idea of a progress constraint coupled to the CCT minimization appeared in [16], where the (static) progress of each flow is to be set above certain target threshold. However, the usage of the CCT geometric mean tends to privilege shorter coflows. Furthermore, feasibility issues with progress vectors are not discussed.

III. System Model

The most popular coflow scheduling model is based on a non blocking switch connection of the type reported in Fig. 1, often called the Big Switch model. This model is adequate because the bisection bandwidth of modern datacenters exceeds access capacity, so that congestion events occur at the inbound or outbound ports of top-of-rack switches; from now on the term link and port will be used interchangeably. The set of switch links is $\mathscr{L} = \{1, \dots, 2M\}$, where links $1 \le \ell \le M$ are input links and $M + 1 \le \ell \le 2M$ are output links. Each link has capacity B_ℓ.

Let $\mathscr{C} = \{1, \dots, N\}$ be an input set of coflows, i.e., a *batch*. Each coflow j is a set $\mathscr{F}_j$ of n_j flows and a flow represents a shuffle connection over a pair of input-output ports. The release time $r_j \ge 0$ of coflow j is the time when its shuffle phase starts. A component flow $i \in \mathscr{F}_j$ has volume v_j^i. The *size* of coflow j, that is the total volume of coflow j, is denoted by $V_j = \sum_{i \in \mathscr{F}_j} v_j^i$. Let $\chi_j^i(\ell)$ indicate if flow $i \in \mathscr{F}_j$ is active on port ℓ. Tab. I summarizes all the important notations and their descriptions.

Let C_j be the CCT of coflow j: C_j is the epoch when the last flow in $\mathscr{F}_j$ is fully transferred. We let C_j^0 be the coflow completion time *in isolation*, i.e., if all network resources serve solely the tagged coflow. It holds $C_j^0 =: r_j + \max_{l \in \mathscr{L}} \frac{1}{B_\ell} \sum_{i \in \mathscr{F}_j} v_j^i \chi_j^i(\ell)$. The standard coflow scheduling problem corresponds to determine a coflow schedule $\mathscr{M}$ able to minimize the weighted sum of the CCTs under the capacity



Tab. A

Coflow j	n_j	Flow Volume $\mathbf{v}_j^i$	I/O Ports
$\mathbf{1, 2, 3, 4, 5}$	1	4	$j/M + j$
$\mathbf{6}$	5	2	$i/M + i$

Tab. B

$\mathscr{M}_1$ ($j = 6$ is last)	$t \in [0, 4]$: $r_6^j(t) = 0, r_j^1(t) = 1, j = 1, \ldots, 5$	$t \in [4, 6]$: $r_6^j(t) = 1, r_j^1(t) = 0, j = 1, \ldots, 5$
$\mathscr{M}_2$ ($j = 6$ is first)	$t \in [0, 2]$: $r_6^j(t) = 1, r_j^1(t) = 0, j = 1, \ldots, 5$	$t \in [2, 6]$: $r_6^j(t) = 0, r_j^1(t) = 1, j = 1, \ldots, 5$

Tab. C

	CCT	Elongation	Average CCT	$\mathbf{R}_j$
$\mathscr{M}_1$	$(4, 4, 4, 4, 4, 6)$	$(1, 1, 1, 1, 1, 3)$	$4 + \frac{1}{3}$	$(1, 1, 1, 1, 1, \frac{1}{3})$
$\mathscr{M}_2$	$(6, 6, 6, 6, 6, 2)$	$(\frac{3}{2}, \frac{3}{2}, \frac{3}{2}, \frac{3}{2}, \frac{3}{2}, 1)$	$5 + \frac{1}{3}$	$(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, 1)$

Fig. 1. Example 1: schedules $\mathscr{M}_1$ and $\mathscr{M}_2$ attain a different tradeoff with respect to average CCT and average progress. Tab A: input parameters for the coflow batch. Tab B: rate breakdown for $\mathscr{M}_1$ and $\mathscr{M}_2$. Tab C: performance of $\mathscr{M}_1$ and $\mathscr{M}_2$.

constraints imposed by the network fabric. A coflow schedule $\mathscr{M}$ provides the set of rates which are assigned to each flow of each coflow per port. The large class of σ-order schedulers [7], for instance, prescribes to respect a static preemption priority σ and only require monitoring of coflows at ingress/egress ports of the Big Switch. As proved in [8], the efficiency loss w.r.t. an optimal scheduler respecting the given priority σ is at most a factor 2. In the numerical evaluations in Sec.VI, rates are assigned following a strict priority rule, namely the greedy rate allocation policy [8].

Static coflow progress. In the coflow literature, the standard definition of coflow fairness [15], [17] is based on the notion of coflow progress. A fairness guarantee is defined by ensuring that every coflow has a target minimum progress *per link*, e.g., per input or output port in the Big Switch model. Let a_j^i be the fraction of bandwidth guaranteed to flow i of coflow j. The progress of coflow j is defined as

$$P_j = \min_{\ell \in \mathscr{L}} \min_{\mathscr{F}_j(\ell)} \frac{a_j^i \, B_\ell \, (C_j^0 - r_j)}{v_j^i} \tag{1}$$

where $d_j^i := \frac{v_j^i}{B_\ell \, (C_j^0 - r_j)}$ is also called the rate demand of flow i of coflow j and $\mathscr{F}_j(\ell)$ is the set of flows of coflow j active on link ℓ; from the definition of C_j^0 indeed $P_j \leq 1$. Finally, coflow fairness ensures a per-coflow minimum progress

$$P_j \geq P_0, \quad \forall j \in \mathscr{C} \tag{2}$$

In some works, the constant parameter P_0 is also called as isolation guarantee [17].

Dynamic coflow progress. Now we introduce a dynamic notion of coflow fairness in terms of overall coflow progress guarantee.

Definition 1 (Dynamic progress)**.** *The dynamic progress of coflow j is denoted by* $R_j = \frac{1}{n_j} \frac{V_j}{C_j^0 - r_j}$.

This notion of progress has the physical dimension of a rate, and it represents the average progress per flow during the coflow transmission. Finally, the elongation of a coflow is the ratio between the progress in isolation and the progress under a given schedule and writes as the ratio of the related completion times.

Definition 2 (Coflow elongation)**.** *The elongation of coflow j under coflow schedule $\mathscr{M}$ is the ratio* $\frac{C_j - r_j}{C_j^0 - r_j}$.

The above ratio is a measure of fairness since it captures the relative degradation of the data transfer time when a coflow is scheduled together with a batch of competing ones. By expressing the dynamic progress as elongation constraint, it is possible to avoid static rate allocations.

Example 1. Now we present a motivating example to detail the key differences between static and dynamic coflow progress. In Fig. 1 we have reported a standard example for a fabric with $M = 5$ servers and unitary bandwidth links. One coflow, coflow $j = 6$ engages all input and output ports of the fabric with 5 flows released at time 0 with volume 2 each. There are 5 competing coflows $j = 1, \ldots, 5$ consisting of a single flow and which compete one-to-one with the flows of coflow 6 over I/O ports. We consider two schedules $\mathscr{M}_1$ and $\mathscr{M}_2$. The first schedule $\mathscr{M}_1$ assigns full rate to coflows $j = 1, \ldots, 5$ first and when they are dispatched, serves coflow 6. Using the results of the next section, it possible to prove that the attained average CCT is minimum. However, the elongation experienced for this schedule is 3 for coflow 6 and 1 for the other coflows. Similarly, the dynamic progress is $1/3$ for coflow 6 and 1 for the other coflows.

Conversely, $\mathscr{M}_2$ assigns full rate to coflow 6 first and serves coflows $j = 1, \ldots, 5$ when 6 is completed. In this case, the average CCT is increased by one unit, but the dynamic progress is 1 for coflow 6 and $2/3$ for the other coflows. Thus the second schedule attains a dynamic progress of at least $2/3$ per coflow: this means that every flow of every coflow have been receiving at least $2/3$ of the overall I/O bandwidth. But, no scheduler leveraging on static progress can attain a $2/3$ progress. In fact, in order to do so, it should grant $2/3$ of the bandwidth of input and ports to both all the flows of coflow 6 and to each of the other competing coflows, which is clearly impossible. The fact that the best

dynamic progress guarantee per coflow cannot be attained by a static progress scheme is consistent with the results showed in the next section, where the average progress is expressed in terms of elongation constraints. In fact, using the tools and the results introduced in the next section, it is possible to prove numerically that such dynamic progress corresponds to the smallest possible elongation constraint for the batch of coflows in the example. Moreover, for the given coflow batch, $\mathscr{M}_1$ and $\mathscr{M}_2$ attain two different trade-offs between average CCT and dynamic progress. By generalizing the above example for $v^i_{M+1} = v_2$ and $v^1_j = v_1$ for $i, j = 1, \dots, M$, it follows by direct calculations that $\mathscr{M}_2$ performs better in terms of average CCT and worse in terms of dynamic progress than $\mathscr{M}_1$ if and only iff $(v_2/v_1) \leq M \leq (v_2/v_1)^4$.

Example 2. Finally, it is interesting to change the example by replacing coflow $j = 6$ with a coflow with a single flow and having same total volume $V_6 = v^1_6 = 10$ and engaging ports 1 and 6 only. In this case, $\mathscr{M}_1$ produces a CCT vector $(4, 4, 4, 4, 4, 14)$ and an elongation vector $(1, 1, 1, 1, 1, 1.4)$, whereas $\mathscr{M}_2$ produces a CCT vector $(14, 4, 4, 4, 4, 10)$ and an elongation vector $(3.5, 1, 1, 1, 1, 1)$. Because it has only one flow, the elongation experienced by the coflow 6 under $\mathscr{M}_1$ in this example is much less severe as compared to the previous example and such schedule is also the fairer. In fact, in the previous example, all coflows competing with $j = 6$ would only face $1/5$ its aggregated volume; in the example 2, instead, $\mathscr{M}_2$ gives priority to $j = 6$ and penalizes coflow $j = 1$ by occupying its bottleneck M-times longer.

By using the notion of elongation just introduced, it is possible to overcome the fundamental limitation in the standard notion of static coflow progress, i.e., its lack of flexibility. Indeed, static coflow progress prevents pausing the flow transmission on some ports used by a coflow and increase, for example, the rate of transmission of other flows of the same coflow, or even defer/anticipate its transmission. We now present the mathematical framework which permits to combine CCT minimization and dynamic progress guarantees by enforcing a suitable elongation constraint.

IV. Scheduling with elongation constraints

The coflow scheduling problem determines a coflow schedule $\mathscr{M}$ minimizing the weighted sum of the CCTs under the capacity constraints imposed by the network fabric. It is often formulated as a Mixed Integer Linear problem (MILP) for a finite horizon where time is slotted into T time slots of duration Δ. Let $\Delta = 1$ without loss of generality. The decision variables $\{x^i_j(t)\}$ represent the fraction of flow $i \in \mathscr{F}_j$ of coflow j scheduled in slot t. The flow fraction already transmitted by slot t is $Z^i_j(t) = \sum_{v=1}^{t} x^i_j(v)$. The fraction of coflow j transmitted by time t writes $Z_j(t) := \frac{1}{V_j} \sum_{i \in \mathscr{F}_j} v^i_j Z^i_j(t)$; note that $C_j = \min\{t |\, Z_j(t) = 1\}$. Finally, the following MILP represents the Dynamic Progress Scheduling (DPS) problem

$$\text{minimize: } \sum w_j C_j \tag{DPS}$$

$$\text{subj. to: } \sum_{t=1}^{T} x^i_j(t) = 1, \quad \forall j \in \mathscr{C}, i \in \mathscr{F}_j \tag{3}$$

$$X_j(t) \leq Z_j(t), \ \forall j \in \mathscr{C}, \forall t \tag{4}$$

$$Z_j(t) \leq 0, \forall t \leq r_j \tag{5}$$

$$C_j \geq 1 + \sum_{z=1}^{T} (1 - X_j(z)), \ \forall j \in \mathscr{C} \tag{6}$$

$$\frac{C_j - r_j}{C^0_j - r_j} \leq \frac{E}{V_j}, \ \forall j \in \mathscr{C} \tag{7}$$

$$\sum_{j \in \mathscr{C}} \sum_{i \in \mathscr{F}_i} v^i_j x^i_j(t) \chi^i_j(\ell) \leq B_\ell, \ \forall t, \forall \ell \in \mathscr{L} \tag{8}$$

$$x^i_j(t) \in [0, 1], \ X_j(t) \in \{0, 1\}, \ \forall t \tag{9}$$

where E is a non-negative parameter and weight w_j denotes the priority of coflow j. We denote $\{C^*_j\}$ a solution of the DPS and $\{R^*\}$ the corresponding dynamic progress vector.

The MILP makes use of coflow completion variables $Z_j(t)$ and binary variables $X_j(t)$. Note that, due to minimization, $X_j(t)$ behaves as the indicating variable of coflow completion, that is $X_j(t) = 1$ if $Z_j(t) = 1$ and $X_j(t) = 0$ if $Z_j(t) < 1$. Constraint (3) states that each flow should be completed by the end of the time horizon.

Constraints in (4) state that the completion of a coflow is subjected to the transfer of the whole coflow volume. In (5) the release time of coflows imposes the constraint on the beginning of a coflow transmission. Constraint (6) binds coflow completion time and coflow completion variables, which appeared first in [7]. Finally, (8) provides the constraint on the port capacity; $Z_j(t)$ is defined implicitly in (4) and (5) for notation's sake.

Compared to the standard coflow scheduling formulations, we have introduced the elongation constraint (7). It states that for each coflow $j \in \mathscr{C}$, the maximum value of the elongation weighted by its total volume should be a constant $E \geq \max_j\{1, \max V_j\}$. Clearly, adding progress constraints leads to a deterioration of the average CCT with respect to the baseline problem of weighted CCT minimization. It is worth remarking that, for example 1 in the previous section, constraint (7) writes $C_6 \leq E/5$ whereas in the example 2 it writes $C_6 \leq E$, encoding the fact that, for the same coflow size, a coflow concentrated on a few ports is penalized compared to a coflow engaging more ports. Setting $E = 10$ attains the schedule $\mathscr{M}_2$ for example 1 and $E = 14$ attains $\mathscr{M}_1$ for example 2. Those values corresponds to the maximum feasible dynamic progress guarantee.

Next, we compare the notion of dynamic fairness and the static fairness. For the sake of comparison, we introduce the customary optimization problem considered in the literature based on the static progress formulation (1) and (2). This leads to the standard Static Progress Scheduling (SPS) problem

described below:

$$\text{minimize: } \sum w_j C_j \tag{SPS}$$
$$\text{subj. to: (3)(4)(5)(6)(8)} \tag{10}$$
$$x_j^i \geq a_j^i \tag{11}$$
$$x_j^i(t) \in [0,1],\ X_j(t) \in \{0,1\},\ \forall t \tag{12}$$

where the minimum rate allocation $\{a_j^i\}$ obeys to (1) and (2) for a given progress parameter P_0. We denote $\{\overline{C}_j\}$ the output of SPS.

From an analogous result for the weighted CCT minimization, it follows that both DPS and SPS problems are NP-hard and inapproximable below a factor 2 [7].

Constraint (7) can be seen as the average rate guarantee per coflow: the dynamic progress constraint allows flexible rate allocation where the rate on some link can be increased by reducing or pausing the transmission on some other links used by the coflow. Formally, the following result shows that for every SPS scheduler there exists a coflow elongation parameter E and a DPS scheduler which is better off under the dynamic progress constraint. First, in order to compare the two formulations, we should observe that for SPS problem the dynamic progress guarantee for coflow j corresponding to an allocation $\{a_j^i\}$ is $\overline{R}_j := \frac{1}{n_j} \sum_\ell \sum_{i \in \mathscr{F}_j} a_j^i\ B_\ell$.

Theorem 1. *For every input $\mathscr{C}$, and static progress defined according to* (1) *and* (2)*, there exists a constant E for which a solution of DPS is such that*

$$\sum C_j^* \leq \sum \overline{C}_j, \quad \textit{and} \quad R_j^* \geq \overline{R}_j \quad j \in \mathscr{C}$$

From Thm. 1, scheduling with elongation constraints provides larger feasibility space and at least same guarantees on the average progress of a coflow than static progress. We next discuss the feasibility of such formulation, for general scheduling policies and then formally define σ-order schedulers.

A. Feasibility, σ-order schedulers and bottleneck-feasibility

In analogy with α-fairness, the elongation parameter E can be tuned in order to tradeoff fairness for average CCT. In order to maximize the average progress of coflows, we have to minimize the value of E. Clearly, linear inequalities (7) are such that not every value of E results in a feasible instance for DPS. A simple test of feasibility can be provided in the form of a linear program solving the Elongation Feasibility Problem (EFEA), from which the following result holds

Theorem 2. *Deciding the feasibility of DPS is in* P.

The proof and the actual LP formulation for the EFEA problem are detailed in the Appendix. Let denote E^* the minimum feasible value of E for the DPS problem: it corresponds to the maximum dynamic progress. From Thm. 2, E^* can be determined in pseudo-polynomial time by solving the EFEA LP iteratively with a bisection search over the elongation parameter E. By simple calculations this search is in $O(\frac{(NV)^{3.5}}{\epsilon})$ where $V = \sum V_j$, and $\epsilon > 0$ is a tolerance. The worst case LP bound used here is $O(n^{2.5})$ in the number of variables n.

In the next section we shall introduce a much faster approximated yet accurate algorithm to determine E^* (namely Alg. 1). Also, we will restrict to the class of σ-order preserving schedulers, σ-order schedulers for short, denoted as Σ. These schedulers require to set the priority of coflows apriori and such an order is used by a rate allocation engine respecting the prioritization [8].

Let order $\sigma : \mathscr{C} \to \mathscr{C}$ be a permutation of the coflow set: $j = \sigma(k)$ means that the k-th priority is assigned to coflow j.

Definition 3. *Fix an order $\sigma : \mathscr{C} \to \mathscr{C}$: the scheduler $\mathscr{M}$ is a σ-order preserving scheduler if*

- *it is pre-emptive;*
- *it is work conserving;*
- *it respects the pre-emption order: no flow of coflow $\sigma(j)$ can be pre-empted by a flow of $\sigma(k)$, $k = j+1, \dots N$ on any port on which it has pending flows.*

The greedy rate allocation, for instance, blocks a coflow $\sigma(j)$ on some port iff it is occupied by some coflow $\sigma(k)$, $k < j$. As first proved in [8], the design of σ-order schedulers can be performed using the analogy of the Big Switch ports with correlated machines and using related results for open-shop scheduling [24]. We follow the same idea in order to derive schedulers in Σ to approximate the solution of DPS problem. For analytical tractability, we consider release time $r_j = 0\ \forall j \in \mathscr{C}$, and the general case is considered in future works.

First, let define $p_{\ell,j} = \sum v_j^i \chi_j^i(\ell)$ the total volume to be transferred by coflow j over port ℓ. Suppose $\mathscr{C}_\ell$ is the set of coflow active on port ℓ. Let define the target constraint $D_j = \frac{EC_j^0}{V_j}$ for notation's sake. We shall consider a larger set of schedulers Σ_b, that is the set of σ-order schedulers such that the permutation σ is *bottleneck-feasible*, that is

$$\max_{\ell \in \mathscr{L} | p_{\ell,k} > 0} \frac{1}{B_\ell} \sum_{j=1}^{k} p_{\ell\sigma(j)} \leq D_k, \qquad \forall k \in \mathscr{C} \tag{13}$$

Note that if a schedule $\mathscr{M}$ is feasible for DPS, it is also bottleneck-feasible. In fact, it is sufficient to consider the σ-order induced by the CCT order by the said scheduler. Thus, if there exists a solution for DPS in Σ, then it is to be found in Σ_b which means $\Sigma \subseteq \Sigma_b$. However, since bottleneck feasibility does not account for cross-dependencies among ports, it provides a weaker estimate of the completion time for the flows of a coflow according to a given σ-order. This has two consequences:

1) The minimum value of the parameter E for which the formulation is bottleneck-feasible – should be in general smaller than the minimum possible value E^* which renders DPS feasible. However, they practically coincide. This is because typically the CCT of the last coflow is decided by the most charged port and for that coflow the value D_k is in fact determined by the total volume transferred on this bottleneck. The heuristic algorithm proposed in the next section leverages

Algorithm 1 Algorithm for the computation of E^*.

Input: $\mathscr{C}$, $\{V_j\}$ $\{p_{\ell,j}\}$
Output: E^*
1: $E_\ell = +\infty, \quad \forall \ell \in \mathscr{L}$ % initialise elongation constraint
2: **for** $\ell = \ell_1, \ldots, \ell_{2M}$ **do**
3: Sort $\mathscr{C}_\ell$ in decreasing order of $\frac{V_j}{C_j^0}$ % bottleneck rationale
4: **for** $j \in \mathscr{C}_\ell$ **do**
5: $Z_j = \max\{V_j, \frac{V_j}{C_j^0} \sum_{k=1}^{j} v_k^\ell\}$ % per coflow estimate on ℓ
6: **end for**
7: $E_\ell = \max\{Z_1, ..., Z_{|\mathscr{C}_\ell|}\}$ % all estimates on ℓ
8: **end for**
9: $E^* = \max\{E_{\ell_1}, \ldots, E_{\ell_k}\}$ % lower bound
10: **return** E^*

Algorithm 2 Bottleneck-feasible σ-order computation.

1: **Input:** $\mathscr{C}$, $\{p_{\ell,j}\}$, $\{D_j\}$, $\{\alpha_j\}$
2: **Output:** σ $\{y_{\mu_k,F_k}\}$ $\{C_j\}$
3: $\{y_{\ell,S}\} = 0$ for $S \subseteq \mathscr{C}$ % initialise the dual variables
4: $A_N = \mathscr{C}$ % initial unscheduled coflows
5: $w_j^{(N+1)} = w_j + \alpha_j$ % initial weights
6: **while** $A_k \neq \emptyset$ **do**
7: $\mu_k \leftarrow$ `pivot_bottleneck`(A_k) % pivot bottleneck
8: $F_k \leftarrow F(A_k)$ % set of feasible tail coflows in A_k
9: $\sigma(k) \leftarrow \arg\min_{j \in F_k} \left\{ \frac{w_j^{(k)}}{p_{\mu_k,j}} \right\}$ % pivot coflow (Smith rule)
10: $w_{\sigma(k)}^{(k)} \leftarrow \frac{w_{\sigma(k)}^{(k+1)}}{p_{\mu_k,j}}$ % update pivot coflow weight
11: $y_{\mu_k,F_k} \leftarrow w_k^{(k)}$ % update dual problem solution
12: **for** $j \in A_k \setminus \{\sigma(k)\}$ **do**
13: **if** $j \in \mathscr{F}_k$ **then**
14: $w_j^{(k)} \leftarrow w_j^{(k+1)} - w_{\sigma(k)}^{(k+1)} \frac{p_{\mu_k,j}}{p_{\mu_k,\sigma(k)}}$ % update weights
15: **else**
16: $w_j^{(k)} \leftarrow w_j^{(k+1)}$ % non tail-feasible coflows: unchanged
17: **end if**
18: **end for**
19: $A_{k-1} \leftarrow A_k \setminus \{\sigma(k)\}$ % update unscheduled coflow set
20: **end while**
21: $C_j := \max_{\ell \in \mathscr{L}} \sum_{h=1}^{j} p_{\ell,\sigma(h)}, \quad j \in \mathscr{C}$ % CCT lower bound
22: **return** σ, $\{y_{\mu_k,F_k}\}$, $\{C_k\}$

the structure of the batch of coflows to be scheduled and their volume to compute E^* in polynomial time.
2) A bottleneck-feasible σ-order may not grant the target elongation for all coflows; our objective is to reduce the difference in the elongation of coflows using bottleneck-feasibility as a soft constraint w.r.t. the hard constraint on feasibility appearing in DPS. As described in Sec. VI, in our experiments the fraction of violations in doing so never exceeded a few percents.

In the next section we shall introduce first the algorithm to calculate E^* and then a primal-dual algorithm to determine a bottleneck-feasible σ-order which generalizes the primal-dual approach [24] to include elongation constraints.

V. Algorithmic solution

One fundamental problem with coflow scheduling techniques based on MILPs is that in production datacenters the number of flows per coflow may range in the order of tenths of thousands [5]. Furthermore, the number of variables involved in such systems increases with the inverse of the time slot: the number of per slot rate decision variables easily range in hundred of thousands for time slots in the order of seconds. Thus, scalability issues do arise even for approximation algorithms obtained by solving LPs from MILP relaxation [7], [11]. Hence, we propose two polynomial time algorithms to determine approximate solutions for DPS and EFEA problems. The pseudocodes of the algorithms described in the following of this section require specific notations.

For the set of coflows $A \subseteq \mathscr{C}$, we let $V_\ell(A) = \sum v_j^i \chi_j^i(\ell)$ the total volume transmitted over port ℓ. The set of ports engaged by coflows in A is $\mathscr{L}(A) = \{\ell \in \mathscr{L} | V_\ell(A) > 0\}$. $\mathscr{C}_\ell(A)$ is the set of coflows in A active over port ℓ. $T_\ell(A) = \frac{1}{B_\ell} V_\ell(A)$ is the minimum time required to complete the last coflow of set A over link ℓ. Coflow j is a feasible *tail coflow* of set A if $T_\ell \leq D_j$ for each link ℓ where $p_{\ell,j} > 0$. Formally, $F(A) = \{j \in \mathscr{C}_\ell(A) | T_\ell(A) \leq D_j, \ \forall \ell \in \mathscr{L}(A)\}$ is the set of feasible *tail coflows* of set A. With no loss of generality, in the rest of this section we shall assume that $B_\ell = 1$.

Computation of E^*. The pseudocode of Alg. 1 describes a heuristic procedure to approximate the minimum value of the parameter E, namely E^* for which DPS is feasible. The objective of this algorithm is to determine the minimum value of E which satisfies the feasibility constraint (7) in DPS in the set of σ-order schedulers. It can be observed that the value of E which satisfies the elongation constraint (7) is minimum when the CCT of the coflow with the largest value of the ratio $\frac{V_j}{C_j^0}$ is minimized. Thus, the algorithm visits each bottleneck ℓ by sorting $\mathscr{C}_\ell$ in decreasing order of the ratio $\frac{V_j}{C_j^0}$. The CCT of a coflow is estimated from the total volume on its bottleneck link by using the left hand term in (13). Alg. 1 starts by sorting the links in decreasing order with respect to the total volume engaged on them at step 1. For each link ℓ, the iteration at step 7 provides a lower bound on E^* per coflow active on ℓ using (7) and the maximum among them is selected at step 9. Finally, the value E^* is the maximum obtained over all bottlenecks at line 11. The complexity of Alg. 1 is in $O(MN \log(M))$.

Primal-dual algorithm. The pseudocode in Alg. 2 describes a primal-dual algorithm to generate a σ-order which is bottleneck-feasible. The primal-dual weighting technique was first introduced in [24] for job scheduling and later applied in the context of coflow scheduling in [8] and [12]. At each step k it identifies a *pivot* bottleneck-coflow pair (line $7-9$). First, the bottleneck μ_k is returned for the unscheduled coflows $A_k = \{\sigma(1), \sigma(2), \ldots, \sigma(k)\}$ using a `pivot_bottleneck` procedure (line 7). Second, it identifies the coflow $\sigma(k)$ to be scheduled *last* over such bottleneck in the set of feasible tail coflows F_k. The feasible tail coflow on the selected bottleneck is chosen according to the Smith rule [25]. Weight updates are only performed for the pivot coflow (line 8) and for feasible tail coflows (line 12). Afterwards, the algorithm eliminates the selected coflow from the set of unscheduled ones (line 17). Weights α_j have the meaning of multipliers and can be used in

order to enforce different σ-orders: as it will be showed in the next section using such weights and `pivot_bottleneck` it is possible to attain all possible bottleneck-feasible σ-orders.

We have implemented two variants of the `pivot_bottleneck` procedure:
1) *Max Load:* selects the most charged bottleneck for the set A_k; this is the baseline rule adopted in [8], [12], [24] for average CCT minimization;
2) *Max Slackness:* mimics the earliest deadline first by selecting the link ℓ for the coflows of set A_k such that the selected coflow k_ℓ has the largest margin with respect to the elongation constraint, i.e., $|T_\ell(A) - D_{k_\ell}|$ is maximized.

Note that not all bottleneck-feasible σ-orders are attainable using the above two bottleneck selection rules. Also, the computational complexity of Alg. 2 is $O(N(M+N))$.

Correctness. Let denote Σ^{LP} the set of solutions of the following linear program (LP) Bottleneck Primal LP. It corresponds to a scheduling formulation over correlated machines [24];

$$\text{minimize:} \sum_{j\in\mathscr{C}} w_j C_j \qquad \text{(Bottleneck Primal LP)}$$

$$\text{subj. to:} \sum_{j\in A} p_{\ell,j} C_j \geq f_\ell(A), \quad \forall A \subseteq \mathscr{C},\ \ell \in \mathscr{L} \qquad (14)$$

$$C_j \leq D_j,\ \forall j \in \mathscr{C} \qquad (15)$$

The set functions $f_\ell(A) = \frac{1}{2}[(\sum_{j\in S} p_{\ell,j})^2 + \sum_{j\in S} p_{\ell,j}^2]$ and (14) are known as parallel inequalities. It is known that $f_\ell(A)$ is strictly supermodular, i.e., $f_\ell(A_1 \cup A_2) \geq f_\ell(A_1) + f_\ell(A_2)$ with equality if and only if $A_1 \cap A_2 = \emptyset$.

The approximation properties of the proposed algorithmic solution rely on the dual formulation

$$\text{maximize:} \sum_{\ell\in\mathscr{L}} \sum_{A\subseteq\mathscr{C}} y_{\ell,A} f_\ell(A) - \sum_j \alpha_j D_j \qquad \text{(Dual LP)}$$

$$\text{subj. to:} \sum_{\ell\in\mathscr{L}} \sum_{A\subseteq\mathscr{C}: j\in S} y_{\ell,A}\, p_{\ell,j} = w_j + \alpha_j, \quad j \in \mathscr{C} \qquad (16)$$

$$y_{\ell,A} \geq 0,\ \alpha_j \geq 0 \qquad (17)$$

The output of the algorithm provides one solution for Bottleneck Primal LP and one solution for Dual LP. In particular, using the following two facts

Lemma 1. *i.* $\Sigma_b \subset \Sigma^{LP}$
ii. If Bottleneck Primal LP is feasible, then sets F_k *in Alg. 2 are such that* $F_k \neq \emptyset$ *for* $k = 1, \ldots, N$.

we can extend the analysis in [24] to the case when multiple candidate bottlenecks exist and we can account for feasibility at once, resulting in the following

Theorem 3. *Let Bottleneck Primal LP be feasible. Alg. 2 produces a bottleneck-feasible solution* $\{C_k\}$ *for Bottleneck Primal LP and a feasible solution* $\{y_{\ell,A}\}$ *for Dual LP.*

The proof of the above results are described in the Appendix; as a byproduct, the proof of Thm. 3 offers a simple interpretation of the Smith rule and of the result in [24] from basic linear algebra, i.e., based on gaussian elimination.

Output completeness. It is possible to show that, for a specific set of weights, the proposed algorithm can produce any bottleneck-feasible σ-order.

Theorem 4. *For every bottleneck-feasible* σ*-order* σ^**, there exist multipliers* $\{\alpha_j\}$, $\max \alpha_j = 1$ *so that* σ^* *is the output of Alg. 2 under rescaled weights* $\{\kappa \cdot w_j\}$*, for some* $0 \leq \kappa \leq 1$.

Approximation factor. Let C_j be the output of Alg. 2. Let define C_j^{OPT}, $j \in \mathscr{C}$ a solution of DPS and C_j^{LP} the optimal solution of Bottleneck Primal LP. We also define $\widehat{C}_j$ the coflow completion times of a σ-order using a priority output of Alg. 2. C_j^{OPT} is defined accordingly. Let recall a few results [8] [24]:

Lemma 2. *i.* $\sum_{k=1}^N w_k C_k^{LP} \leq \sum_{k=1}^N w_k C_k^{OPT}$
ii. $\left(\sum_{j=1}^k p_{\mu_k\sigma(j)}\right)^2 \leq f_\ell(S)$*, for* $\ell \in \mathscr{L}$ *and* $S \subseteq \mathscr{C}$
iii. $\widehat{C}_j - r_j \leq 2(C_j - r_j)$

Theorem 5. *Alg. 2 produces a bottleneck-feasible schedule such that for the corresponding* σ*-order coflow schedule* σ^*

$$\sum w_j \widehat{C}_j \leq 4 \sum w_j C_j^{OPT} + 4 \sum \alpha_j D_j \qquad (18)$$

Assume $\alpha_j = 0$ *and* σ^* *is feasible for DPS, then it is a* 4 *approximation w.r.t. an optimal scheduler.*

VI. Numerical results

In this section we summarize the results of the numerical experiments we have performed in order to validate the proposed coflow fairness framework and algorithms. The numerical results have been performed on our matlab® coflow scheduling simulator equipped with a coflow generator. As indicated in the analysis of Facebook traces reported in [5], a typical coflow pattern observed in data centers may comprise a significant fraction of small coflows but also coflows with large *width* (i.e., number of flows). In our tests we have considered batches of coflows with a fraction q of single flows and a fraction $(1-q)$ of coflows whose width is drawn uniformly from $M/3, \ldots, M$. Flow volumes are exponentially distributed with average 8 and standard deviation 3 in normalized traffic units. Links have unitary capacity.

Minimum elongation estimation. As in Sec. IV, the minimum elongation E^* is a key parameter corresponding to the largest feasible (dynamic) progress that can be ensured to all coflows. In Tab. II the estimation of E^* obtained by Alg. 1 is compared against the exact value obtained by solving LP EFEA for 6 typical instances; note that the complexity of the related time-indexed LP obliges to limit the samples to moderate size instances. The estimation produced using the bottleneck approximation is quite tight compared to the minimal E^*, with relative errors of few percents in all our experiments.

Elongation-based Fairness. Fig. 2a describes the effect of the elongation constraint E onto the average CCT for a fabric with $M = 30$ ports and $N = 30, 50, 100$ coflows, respectively. Results are averaged on hundred sample batches and normalized against the near optimal solution provided by Sincronia. As expected, the average CCT decreases for

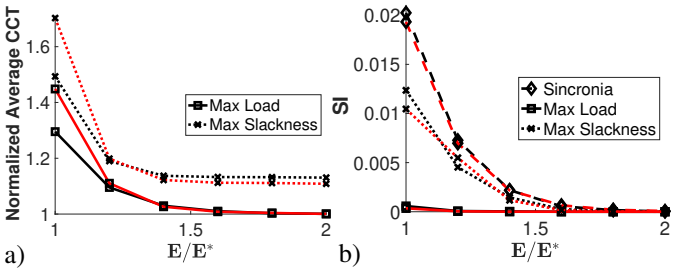


Fig. 2. a) Average CCT normalized against near optimal solution with no elongation guarantees and b) average stretch index; $M = 30$, $N = 50$ (black lines), and $N = 100$ (red lines), $q = 0.8$

increasing E: similarly to the α parameter of classic flow fairness, E can be tuned in order to attain a tradeoff between coflow elongation and average CCT. To measure the fairness enforced by the scheduler, we define the normalized elongation $\widehat{E}_j := \frac{C_j V_j}{C_j^0}$ and $\widehat{E}$ its average; $\widehat{E}_j < E$ for all $j \in \mathscr{C}$ means the feasibility constraint is attained. Let define the Stretch Index $SI := \sum \max(0, (E_j/E) - 1)/N$: it measures the relative magnitude of the violations of the elongation constraint. As reported in Fig. 2b, Alg. 2 with the Max Load heuristic attains a negligible fraction of violations, confirming more effective than Max Slackness. The rather surprising result from Fig. 2a is that Max Slackness – which mimics the classic rule earliest-deadline-first – is outperformed by Max Load which looks just at feasibility on the largest bottleneck at each step. Fig. 3c reports on the cumulative distribution probability of the normalized elongation for $N = 100$ and $q = 0.8$: the elongation constraint manages to prioritize large width coflows, while attaining a very small fraction of violations under Max Load. As reported in Fig. 3d for $N = 30, 50, 100$ and $q = 0.8$ under Max Load, such behavior is independent of the number of coflows per batch. The probability of a violation is $p_v = 0.026, 0.016, 0.007$ for $N = 30, 50, 100$, respectively; in all our tests it appears to decrease with the number of coflows per batch. This confirms that the largest bottleneck provides a tighter approximation for the CCT of the last coflow selected by the algorithm at each step. By comparing Fig. 3d ($q = 0.8$) and Fig. 3e ($q = 0.2$), we note that the elongation distribution is more concentrated for $q = 0.2$ with $p_v < 10^{-3}$. In fact, coflows with large width tend to preempt single flow coflows, which are less frequent for $q = 0.2$. In both cases, however, the algorithm ensures the target elongation per coflow with negligible deviations.

Instance	Alg. 1	EFEA	Rel. Error (%)
$M = N = 6$, $q = 0.2$	67.28	67.8	0.7670
$M = N = 6$, $q = 0.8$	67.95	68	0.0735
$M = N = 10$, $q = 0.2$	234.5	236.85	0.9922
$M = N = 10$, $q = 0.8$	113.87	115.12	1.0858
$M = 10, N = 14$, $q = 0.8$	151.132	152.95	1.1886

TABLE II
MINIMUM ELONGATION E^*: ALG. 1 VS. EFEA

VII. CONCLUSIONS

The data transfer phase of modern computing frameworks requires to serve several coflows in parallel. This work has introduced a new framework for fair coflow scheduling able to enforce the dynamic progress of coflows trading it off for average CCT via a single parameter, namely the elongation. Scheduling under dynamic progress has a larger feasibility set compared to the standard notion of static progress. Also, the maximum possible dynamic progress, i.e., the minimum elongation, can be determined in polynomial time. We have designed an algorithm to minimize the average CCT of a batch of coflows under dynamic progress constraints. The solution is provided in the set of σ-order schedulers. Future works will explore the usage of elongation for non-clairvoyant coflow scheduling, e.g., when the release time or the volume of component flows are only partially known during the data transfer phase.

REFERENCES

[1] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proc. of ACM HotNets*, Redmond, Washington, 2012, p. 31–36.

[2] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," 2004.

[3] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica *et al.*, "Spark: Cluster computing with working sets." *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.

[4] M. Han and K. Daudjee, "Giraph unchained: Barrierless asynchronous parallel execution in pregel-like graph processing systems," *Proceedings of the VLDB Endowment*, vol. 8, no. 9, pp. 950–961, 2015.

[5] N. M. K. Chowdhury, "Coflow: A networking abstraction for distributed data-parallel applications," Ph.D. dissertation, University of California, Berkeley, 2015.

[6] Y. Chen and J. Wu, "Joint coflow routing and scheduling in leaf-spine data centers," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 83–95, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520303683

[7] M. Chowdhury *et al.*, "Near optimal coflow scheduling in networks," in *Proc. of ACM SPAA*, Phoenix, AZ, USA, June 22-24 2019, pp. 123–134.

[8] 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, pp. 16–29.

[9] R. Mao, V. Aggarwal, and M. Chiang, "Stochastic non-preemptive co-flow scheduling with time-indexed relaxation," in *Proc. of IEEE INFOCOM, DCPerf Workshop*, Honolulu, Hawaii, US, April 16 2018, pp. 385–390.

[10] L. Wang, W. Wang, and B. Li, "Utopia: Near-optimal coflow scheduling with isolation guarantee," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 891–899.

[11] M. Shafiee and J. Ghaderi, "Scheduling coflows in datacenter networks: Improved bound for total weighted completion time," *SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 1, p. 29–30, Jun. 2017.

[12] S. Ahmadi, S. Khuller, M. Purohit, and S. Yang, "On scheduling coflows," *Algorithmica*, vol. 82, no. 12, pp. 3604–3629, Dec. 2020.

[13] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. Netw*, vol. 8, no. 5, pp. 556–567, Oct 2000.

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

[15] L. Chen, Y. Feng, B. Li, and B. Li, "Efficient performance-centric bandwidth allocation with fairness tradeoff," *IEEE TPDS*, vol. 29, no. 8, pp. 1693–1706, 2018.

[16] W. Wang, S. Ma, B. Li, and B. Li, "Coflex: Navigating the fairness-efficiency tradeoff for coflow scheduling," in *Proc. of IEEE INFOCOM*, Atlanta, GA, USA, 1-4 May 2017, pp. 1–9.

[17] M. Chowdhury, Z. Liu, A. Ghodsi, and I. Stoica, "HUG: Multi-resource fairness for correlated and elastic demands," in *Proc. of USENIX NSDI*, Santa Clara, CA, March 2016, pp. 407–424.

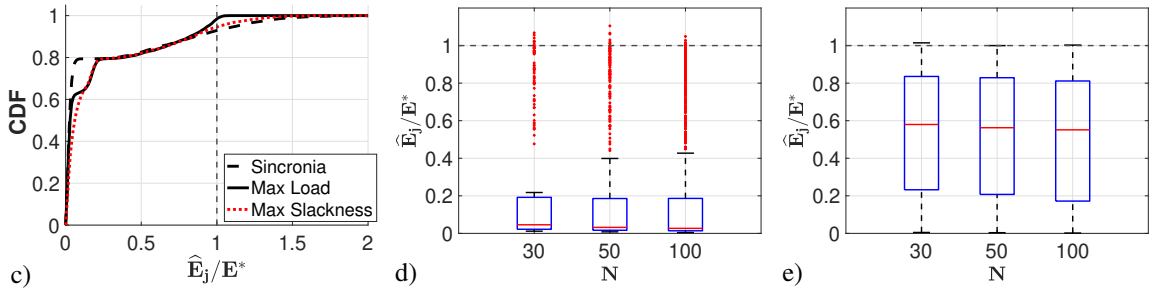


Fig. 3. c) CDF of the normalized elongation for $q = 0.8$ d) Box and Whishers diagrams for $q = 0.2$ and e) $q = 0.8$ for Max Load; $M = 30$ and $N = 100$ for 10 runs; violations occur above the horizontal dashed line.

[18] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proc. of USENIX NSDI*, USA, 2011, p. 323–336.

[19] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*, 2013, pp. 1–16.

[20] T. Zhang, R. Shu, Z. Shan, and F. Ren, "Distributed bottleneck-aware coflow scheduling in data centers," *IEEE TPDS*, vol. 30, no. 7, pp. 1565–1579, July 2019.

[21] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with Varys," in *Proc. of ACM SIGCOMM*, Chicago, Illinois, USA, 2014, p. 443–454.

[22] B. Radunovic and J.-Y. Le Boudec, "A unified framework for max-min and min-max fairness with applications," *IEEE/ACM Trans. on Networking*, vol. 15, pp. 1073 – 1083, 11 2007.

[23] H. Qu, R. Xu, W. Li, W. Qu, and X. Zhou, "Ostb: Optimizing fairness and efficiency for coflow scheduling without prior knowledge," in *Proc. of IEEE HPCC*, 2019, pp. 1587–1594.

[24] M. Mastrolilli, M. Queyranne, A. S. Schulz, O. Svensson, and N. A. Uhan, "Minimizing the sum of weighted completion times in a concurrent open shop," *Operations Research Letters*, vol. 38, no. 5, pp. 390–395, 2010.

[25] M. Queyranne, "Structure of a simple scheduling polyhedron," *Mathematical Programming*, vol. 58, pp. 263–285, 1993.

APPENDIX

PROOF OF THM. 1

Proof: Let denote Σ and $\overline{\Sigma}$ the feasibility sets of SPS and DPS, respectively. In order to prove the statement, it is sufficient to show that we can always determine E such that $\overline{\Sigma} \subseteq \Sigma$. From (1) and (2), the static progress is such that

$$x_j^i \geq a_j^i, \quad \frac{a_j^i \; B_\ell \; (C_j^0 - r_j)}{v_j^i} \geq P_0$$

The quantity $\widetilde{C}_j := r_j + \min_\ell \min_{\mathscr{F}_j(\ell)} \frac{v_j^i}{a_j^i \; B_\ell} \geq r_j + \overline{C}_j$, where $\mathscr{F}_j(\ell)$ as defined in (1). From the definition of static progress, we have

$$\begin{aligned} P_0 \quad &\leq P_j = \min_\ell \min_{\mathscr{F}_j(\ell)} \frac{a_j^i \; B_\ell (C_j^0 - r_j)}{v_j^i} \\ &\leq \frac{C_j^0 - r_j}{\widetilde{C}_j - r_j} \leq \frac{C_j^0 - r_j}{\overline{C}_j - r_j}, \qquad \forall j \in \mathscr{C} \end{aligned}$$

Hence, we obtain that if the static schedule is feasible for a certain P_0, then it holds

$$\frac{\overline{C}_j - r_j}{C_j^0 - r_j} \leq \frac{1}{P_0}, \quad \forall j \in \mathscr{C}$$

and statement follows with the identification $P_0 = \min_j \left\{ \frac{V_j}{E} \right\}$ for $E \geq V_j$ which concludes the proof. ■

PROOF OF THM. 2

Proof: The feasibility of the DPS problem can be formulated with the following linear program, namely the Elongation Feasibility Problem (EFEA)

$$\text{minimize: } 0 \qquad \text{(EFEA)}$$

$$\text{subj. to: } \sum_{t=1}^{T} x_j^i(t) = 1, \quad \forall j \in \mathscr{C}, i \in \mathscr{F}_j \qquad (19)$$

$$Z_j(t) \leq 0, \forall t \leq r_j \qquad (20)$$

$$Z_j(t) \geq 1, \forall t > \frac{E \cdot (C_j^0 - r_j)}{V_j} + r_j \qquad (21)$$

$$\sum_{j \in \mathscr{C}} \sum_{i \in \mathscr{F}_i} v_j^i x_j^i(t) \chi_j^i(\ell) \leq B_\ell, \; \forall t, \forall \ell \in \mathscr{L} \qquad (22)$$

$$x_j^i(t) \in [0, 1], \; \forall t \qquad (23)$$

The derivation of EFEA is obtained by considering any rate allocation for which the constraints of DPS are respected. In particular, constraint (21) has replaced the constraints (6) and (7) in DPS. Since LP feasibility is in *P* this completes the proof. ■

PROOF OF LEMMA 1

Proof: i. Let σ be a bottleneck-feasible σ-order: from (13) $C_{j\ell} \leq D_j$ for all $\ell \in \mathscr{L}$, so that $C_j = \max_{\ell \in \mathscr{L}} C_{j\ell} \leq D_j$, i.e., (15) holds. Furthermore

$$\sum_{j \in A} p_{\ell,\sigma(h)} C_j \geq \sum_{j \in A} p_{\ell,\sigma(h)} C_{j\ell} = f_\ell(A)$$

so that (14) holds as well.

ii. For $k = N$ the statement is true: $A_N = \mathscr{C} \neq \emptyset$ and $F_N \neq \emptyset$, otherwise Bottleneck Primal LP would not be feasible. Let us assume that at step k, $A_k \neq \emptyset$ and $F_k = \emptyset$. Then it means that for all $j \in A_k$ there exists link $\ell_j \in \mathscr{L}(A_k)$ such that $D_j < T_{\ell_j}(A_k)$. If A_k is a singleton, this would contradict the feasibility of Bottleneck Primal LP. If not, this means that over such link ℓ_j, some coflow which belongs to A_k^c should be scheduled before $T_{\ell_j}(A_k) = \sum_{h \in A_k} p_{\ell_j,h}$. However this implies that, for some $h \in A_k$, it would hold $C_h > T_{\ell_j}(A_k) > D_h$. Contradiction. ■

PROOF OF THM. 3

Proof: Let us define a feasible solution for the Bottleneck Primal LP problem. For every pair $(\mu_k, \sigma(k))$ we can define $C_{\sigma(k)} = \sum_{j \le k} p_{\mu_k,\sigma(j)}$; since Bottleneck Primal LP is feasible, the elongation constraint in the primal formulation is satisfied at each step as from Lemma 1. Parallel inequalities hold by applying port-wise results in single machine scheduling [25].

For the Dual LP, we consider solutions of the type

$$y_{\ell,A} = \begin{cases} \theta_k > 0 & \ell = \mu_k \text{ and } A = F_k \text{ for some } k \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

The corresponding form of the constraints of the dual problem

$$\sum_{k=j}^{N} q_{\mu_k,\sigma(j)}\, y_{\mu_k,F_k} = w_{\sigma(j)} + \alpha_{\sigma(j)}, \quad j \in \mathscr{C} \tag{25}$$

where $q_{\mu_k,j} = p_{\mu_k,j} \mathbf{1}\{j \in F_k\}$. If at each step $k = N, \dots, 2, 1$ the selection is

$$\sigma(k) = \underset{j \in F_k}{\operatorname{argmin}} \left\{ \frac{w_j^k}{p_{\mu_N,j}} \right\} \tag{26}$$

the algorithm produces a non-negative solution of the linear system (25). In fact, let first consider step $k = N$. The full rank system of the constraints of the dual system writes

$$\left(\begin{array}{cccc|c} q_{\mu_N\sigma(N)} & 0 & \dots & 0 & w^{(N+1)}_{\sigma(N)} \\ q_{\mu_N\sigma(N-1)} & q_{\mu_{N-1}\sigma(N-1)} & \dots & 0 & w^{(N+1)}_{\sigma(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ q_{\mu_N\sigma(1)} & q_{\mu_{N-1}\sigma(1)} & \dots & q_{\mu_1\sigma(1)} & w^{(N+1)}_{\sigma(1)} \end{array}\right)$$

The gaussian elimination using $\sigma(N)$ as pivot renders the new equivalent system

$$\left(\begin{array}{cccc|c} 1 & 0 & \dots & 0 & \frac{w^N_{\sigma(N)}}{p_{\mu_N,\sigma(N)}} s \\ 0 & p_{\mu_{N-1}\sigma(N-1)} & \dots & 0 & w^N_{\sigma(N-1)} - w^N_{\sigma(N)} \frac{p_{\mu_N\sigma(N-1)}}{p_{\mu_N\sigma(N)}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & p_{\mu_{N-1}\sigma(1)} & \dots & p_{\mu_1,\sigma(1)} & w^N_{\sigma(1)} - w^N_{\sigma(N)} \frac{p_{\mu_N\sigma(1)}}{p_{\mu_N\sigma(1)}} \end{array}\right)$$

Observe that the first column is nullified irrespective of the chosen permutation $\sigma(j)$ for $j < N$. The system has been transformed in the equivalent one where Gaussian elimination can now be performed on the rightmost square sub-matrix. Let replace in the column on the right with the following values

$$\begin{aligned} w_j^{N-1} &= \frac{w^N_{\sigma(N)}}{p_{\mu_N\sigma(N)}}, j = \sigma(N) \\ w_j^{N-1} &= w_j^N - w^N_{\sigma(N)} \frac{p_{\mu_N,j}}{p_{\mu_N,\sigma(N)}} \ge 0, j \in F_k \setminus \{\sigma(N)\} \\ w_j^{N-1} &= w_j^N, j \notin F_k \end{aligned}$$

where $w_j^{N-1} \ge 0$ for all $j \in \mathscr{C}$. Indeed, $y_{\mu_N F_N} = \frac{w^N_{\sigma(N)}}{p_{\mu_N\sigma(N)}}$. The algorithm iterates the procedure on the remaining subsystem using the corner element of $\sigma(k)$ as pivot so that at each step $y_{\mu_k,F_k} = \theta_k = \frac{w^N_{\sigma(k)}}{p_{\mu_k\sigma(k)}} \ge 0$. ∎

Note that unscheduled coflows which are non bottleneck-feasible at a given step do not take part to gaussian elimination.

PROOF OF THM. 4

Proof: The sketch of the proof is as follows. First, by linearity rescaling weights $\{\kappa \cdot w_j\}$ is irrelevant. The Smith ratio at the k-th step for a candidate coflow

$$\frac{w^{(N-k)}_{\sigma(N-k)}}{P_{\mu_k\sigma(N-k)}} = \frac{w_{\sigma(N-k)} + \alpha_{\sigma(N-k)} - \sum_{h=1}^{k} w^{(N-k+h)}_{\sigma(N-k+h)} \frac{P_{\mu_h\sigma(N-k)}}{P_{\mu_h\sigma(N-k+h)}}}{P_{\mu_k\sigma(N-k)}}$$

Hence, to enforce the desired σ-order σ^* as output, let start by an input set $\alpha_j = 1$, for all j, and choose $\alpha_{\sigma^*(N-k)}$ such in a way that the Smith ratio of for $\sigma^*(N-k)$ be minimal among all candidate coflows, simply by rendering $(w_{\sigma(N-k)} + \alpha_{\sigma(N-k)})$ small enough as compared to the other coflows, possibly rescaling all weights w_j by κ. Using $\{\alpha_j\}$ as input to the algorithm produces the desired output σ^*. ∎

PROOF OF THM. 5

Proof: For C_j, output of Alg. 2, it holds

$$\begin{aligned} \sum_{j=1}^{N} w_j C_j &= \sum_{j=1}^{N} C_j \Big(\sum_{\ell \in \mathscr{L}} \sum_{A \subseteq \mathscr{C} : j \in S} y_{\ell,A}\, p_{\ell,j} - \alpha_j \Big) \\ &= \sum_{j=1}^{N} C_{\sigma(j)} \Big(\sum_{k=j}^{N} p_{\mu_k,\sigma(j)} y_{\mu F_k} - \alpha_{\sigma(j)} \Big) \\ &\overset{(i)}{\le} \sum_{k=1}^{N} y_{\mu_k F_k} \sum_{j=1}^{k} p_{\mu_k\sigma(j)} C_{\sigma(j)} - \sum_{j=1}^{N} \alpha_j C_j \\ &\overset{(ii)}{\le} \sum_{k=1}^{N} y_{\mu_k F_k} \Big(\sum_{j=1}^{k} p_{\mu_k\sigma(j)} \Big)^2 - \sum_{j=1}^{N} \alpha_j C_j \\ &\le 2 \Big(\sum_{k=1}^{N} y_{\mu_k F_k} f_{\mu_k}(F_k) - \alpha_k C_k \Big) + \sum_{j=1}^{N} \alpha_j (2D_j - C_j) \\ &\overset{(iii)}{\le} 2 \sum_{k=1}^{N} w_k C_k^{\text{LP}} + \sum_{j=1}^{N} \alpha_j (2D_j - C_j) \\ &\le 2 \sum_{k=1}^{N} w_k C_k^{\text{OPT}} + 2 \sum_{j=1}^{N} \alpha_j D_j \end{aligned}$$

Where we have used the following arguments: (i) follows from the fact that in general $j \in F_k$ does not imply $j \in F_{k+1}$ and (ii) follows from the fact that $C_{\sigma(k)} = \sum_{j=1}^{k} p_{\mu_k\sigma(j)} \ge C_{\sigma(j)}$ for all $k = 1, \dots, j$. According to Lemma. 2, it holds

$$\sum_{j=1}^{N} w_j \widehat{C}_j \le 2 \sum_{j=1}^{N} w_j C_j \le 4 \sum_{k=1}^{N} w_k C_k^{\text{OPT}} + 4 \sum_{j=1}^{N} \alpha_j D_j$$

which concludes the proof. ∎