Priority-aware Coflow Placement and scheduling in Datacenters

Speaker: Lin Wang

Research Advisor: Biswanath Mukherjee

UCDAVIS

Introduction

- **Coflow**
  - Represents a collection of independent flows that share a common performance goal.
  - Coflow's performance depends on its slowest flow.
  - Coflow aware scheduling benefits distributed data processing applications.

- **State-of-arts**
  - Most of current work focus on optimizing the network scheduling algorithm to improve coflows' performance.
  - They assume predetermined coflow placement, i.e. the endpoint locations of a Coflow are preset.
  - But Coflow's placement can be more flexible in practice.

Coflow placement challenge

- **Challenge for inter-flow relationship in a Coflow**
  - E.g., in a one-to-many Coflow, all constituent flows share the same sender location.
  - In many-to-many Coflow, the relationship is even more complex. Because any member flow shares its two ends points with two different groups of flows.
  - Thus, we need to take care of such inter-flow relationship for placement decisions.

2D Placement for Coflow

- **Network Model**
  - Topology designs such as Fat-tree or Clos enable full bisection bandwidth in datacenters.
  - Assume non-blocking N-port switch with link bandwidth B.
  - Switch ports are connected to nodes, which can be host machines or ToR switches.
  - Only edge links are congested and core is congestion free.
- **Scheduling objective**
  - Minimize Coflow completion time (CCT). It is the duration to finish all flows in a Coflow to speed up application level performance.

2D Placement for Coflow

- **Problem Statement**
  - K Coflows arrive at various time . We want to decide the placement for each new-arrived Coflow.
  - The placement of a Coflow can be represented by mapping functions

$$P_k^s : \{s_i\} \rightarrow \{in.1, ..., in.N\} \text{ for senders}$$
$$P_k^r : \{r_j\} \rightarrow \{out.1, ..., out.N\} \text{ for receivers.}$$

  - We assume when a Coflow arrives, its traffic demand D is available.
  - Thus, we need to decide the placement of a new Coflow given the existing previous Coflows, so that the sum of all Coflows' CCTs is minimized.

2D Placement for Coflow

- **Problem Analysis**
  - The sum of CCTs is jointly determined by Coflow's placement and the network scheduling during runtime.
  - First, Coflows' placement decides the optimal sum of CCTs achievable by any network scheduling policy.
  - Second, after Coflows are placed, the sum of CCts will be further determined by the network scheduling policy, which arbitrates bandwidth allocation for each Coflow.
  - 2D Placement focus on finding Coflow placement that minimizes the sum of CCTs under optimal network scheduling.
  - Given specific placement, finding the optimal scheduling policy to minimize the total CCTs is NP-hard.

Motivation Example 1

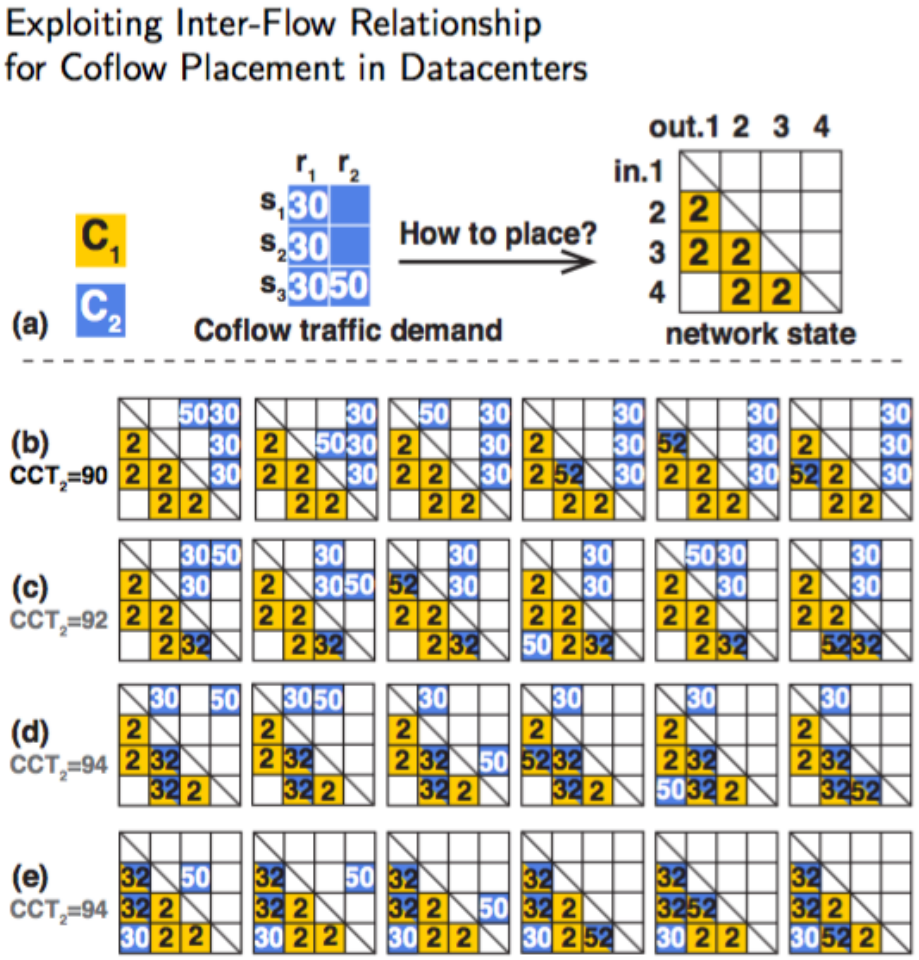


Figure 1: Coflow placement should avoid delaying bottleneck endpoint. (a) Placing 3-by-2 C_2 onto a 4-by-4 network with active flows from 3-by-3 C_1. (b-e) All possible placements of C_2. Cells with two colors indicate links shared by two Coflows. Cell numbers are the aggregated traffic size on the link. The optimal priority order is C_1, C_2. Thus CCT_1 is insensitive to C_2 placement, so only CCT_2 is considered. (b) Optimal placement with C_2's bottleneck r_1 on the least congested *out*.4. (c) Suboptimal placement due to delay of C_2's bottleneck r_1 on *out*.3 (d) Suboptimal placement due to delay of C_2's bottleneck r_1 on *out*.2 (e) Suboptimal placement due to delay of C_2's bottleneck r_1 on *out*.1. For fair comparison, we assume all Coflow traffic should traverse the network.

2D Placement for Coflow

- **Observations**

Hence, we have *Observation 1: When only the CCT of the Coflow to be placed is concerned, the Coflow's placement should avoid delaying the bottleneck.* To achieve this goal, bottleneck endpoint(s) should be placed at ports with sufficient bandwidth resource.

It is interesting to note that, without considering a Coflow's bottleneck, flow-level placement strategy may be suboptimal for Coflow placement. For example, prior work proposes to place a Coflow's constituent flows sequentially in the decreasing order of their flow sizes using a flow-level placement algorithm, because "large flows are more likely to be the critical flows to determine CCT" [14]. However, such strategy would yield suboptimal solutions, as shown in the first column of Figure 1c and Figure 1d. Under this flow-level strategy, the non-critical $f_{3,2}$, despite its largest flow size, takes over *out*.4, leaving suboptimal ports of *out*.2 or *out*.3 for the bottleneck receiver r_2 of C_2.

Motivation Example 2

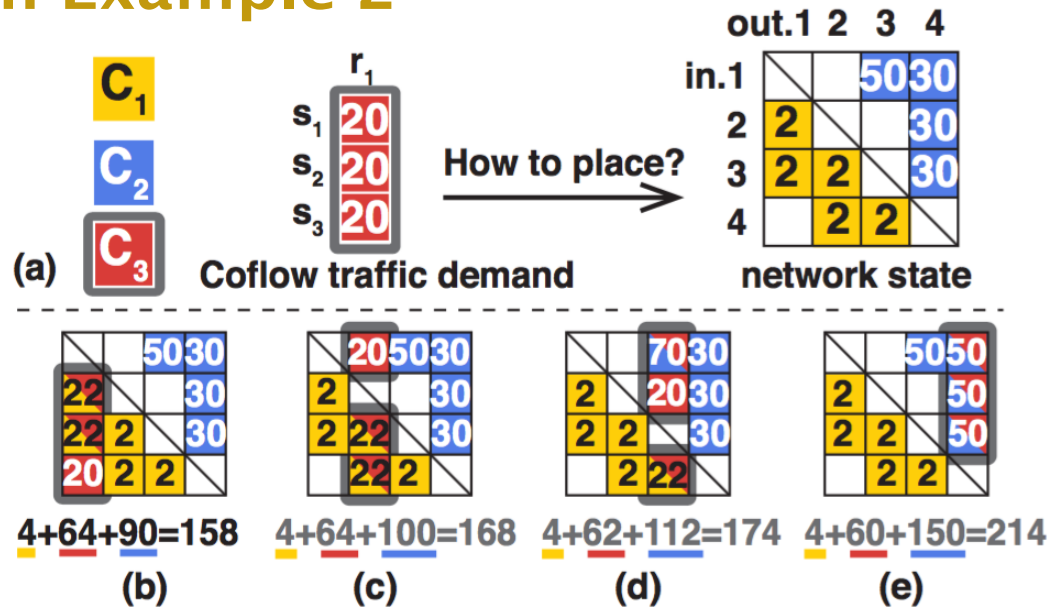


Figure 2: Coflow placement should avoid contentions among critical endpoints with heavy traffic load. (a) Placing incast C_3 onto a 4-by-4 network with active flows from C_1 and C_2. Optimal priority order is C_1, C_3, C_2. (b-e) All possible placements of C_3. CCTs are presented in $CCT_1+CCT_3+CCT_2$. (b) Optimal placement, placing C_3 on *out*.1. (c) Suboptimal placement, delaying C_2 on *in*.1. (d) Suboptimal placement, delaying C_2 on *out*.3. (e) Suboptimal placement, delaying C_2 on *out*.4. Legends and assumptions follow Figure 1.

2D Placement for Coflow

- **First step**
  - Calculate the traffic demand requested on each endpoints for Coflow to place.
- **Second step**
  - 2D-Placement considers each sender (or receiver) in the descending order of their requested demand, and place the sender (or receiver) onto the input (or output) port with the minimum traffic load.
- **Complexity**
  - O(n^2).

Algorithm 1 2D-Placement

Input: Coflow to place C_{new}, remaining load $E^s[.]$ and $E^r[.]$
Output: Placement of all senders $P^s(.)$ and receivers $P^r(.)$

1: **for all** $(s_i, r_j, d_{i,j})$ in C_{new} **do** ▷ Requested load
2: Load on sender $L^s[s_i] \mathrel{+}= d_{i,j}$
3: Load on receiver $L^r[r_j] \mathrel{+}= d_{i,j}$
4: **end for**
5: **for all** s_i in the descending order of $L^s[.]$ **do**
6: $P^s(s_i) = \text{argmin}\, E^s[.]$ ▷ Place sender
7: $E^s[P^s(s_i)] \mathrel{+}= L^s[s_i]$ ▷ Update load on port
8: **end for**
9: **for all** r_j in the descending order of $L^r[.]$ **do**
10: $P^r(r_j) = \text{argmin}\, E^r[.]$ ▷ Place receiver
11: $E^r[P^r(r_j)] \mathrel{+}= L^r[r_j]$
12: **end for**

Simulation

- **Apply two network schedulers**
  - Varys assume accurate Coflow traffic request.
  - Aalo tries to approximate Varys with unknown sizes so as to tolerate error in the requested demand.

Scale factor	×0.5	×0.75	×1	×1.25	×1.5
Aalo	0.87	0.82	0.77	0.77	0.87
Varys	1.00	0.96	0.79	0.74	0.78

Table 1: 2D-Placement's average CCT normalized on Neat's average CCT. Normalized average CCT less than 1 means 2D-Placement is better.

Exploit benefits of Priority-aware Coflow Placement and scheduling in Datacenters

- **Weakness of previous work**
  - In previous work, Coflow placement and scheduling are considered independently.
  - 2D-placement only considers current workload when placing Coflow.

Exploit benefits of Priority-aware Coflow Placement and scheduling in Datacenters

- **Priority-aware Coflow placement and scheduling**
  - When placing a new Coflow, we also take scheduling method into consideration. In this way, Coflow placement and scheduling can benefit each other as they are more consistent.
  - Following the observation in 2D-placement method, we still place the new coflow onto light weighted nodes. But we will not only consider the size of current flow, but also other measurements of current flow. For example, we can either place the new flow onto the minimum distant node or the minimum load node.
  - In our PSON architecture, longest queue fist (LQF), largest number of packets first (LNPF), oldest packet first (OPF), and less space switch tuning time first (LSSTTF) can all be taken into account.

Exploit benefits of Priority-aware Coflow Placement and scheduling in Datacenters

B. *Priority-Aware (PA) Scheduling Algorithm*

We propose a priority-aware (PA) scheduling algorithm, which assigns priorities to VoQs based on four possible queueing strategies: longest queue fist (LQF), largest number of packets first (LNPF), oldest packet first (OPF), and less space switch first (LSSF). (See Fig. 5 for its flowchart.)

First, each ingress module maintains status information and gets priority values for the n VoQs based on their status information. Priority value is calculated as follows:

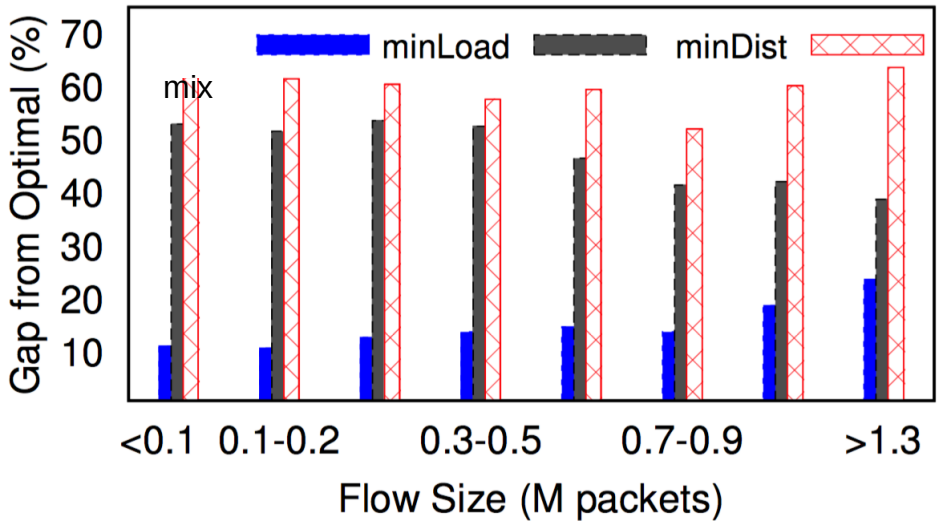
$$W_{ij} = l_{ij} * w_l + p_{ij} * w_p + d_{ij} * w_d + s_{ij} * w_s \qquad (1)$$

Exploit benefits of Priority-aware Coflow Placement and scheduling in Datacenters

- **Simulation Settings**
  - We use benchmark Hadoop (map-reduce) and web-search workloads. These workloads contain a diverse mix of short and long flows with a heavy-tailed flow size distribution. In the web-search workload, more than 75% of all bytes are from 50% of the flows with sizes in the range 1 to 20 MB. The Hadoop workload is less skewed: ~50% of the flows are less than 100MB in size and 4% flows are larger than 80GB.
  - We compare our mixed weight priority method with minDist and minLoad methods.

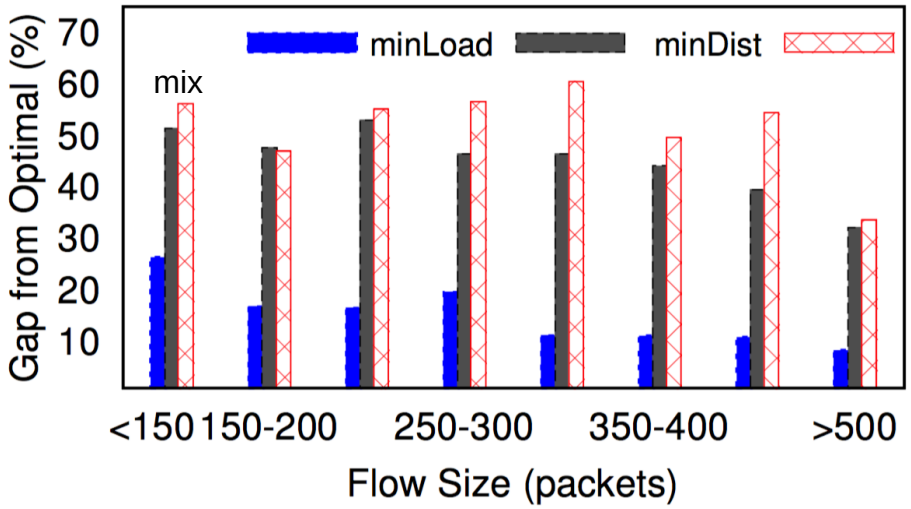
Exploit benefits of Priority-aware Coflow Placement and scheduling in Datacenters

- **Simulation Results**
  - Hadoop data flows



Exploit benefits of Priority-aware Coflow Placement and scheduling in Datacenters

- **Simulation Results**
  - Web search data flows



UCDAVIS
UNIVERSITY OF CALIFORNIA

amlwang@ucdavis.edu