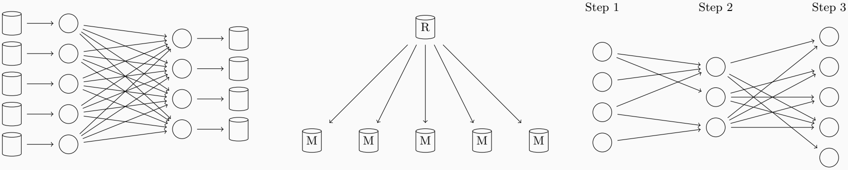
Coflow Deadline Scheduling via Network-Aware Optimization

Shih-Hao Tseng, (pronounced as “She-How Zen”)
joint work with Kevin Tang

October 4, 2018

School of Electrical and Computer Engineering, Cornell University

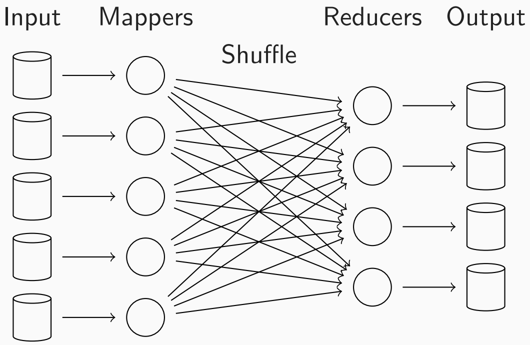
- A coflow is "a collection of flows between two groups of machines with associated semantics and a collective objective" (Chowdhury and Stoica, 2012).



(a) MapReduce **(b)** Hive **(c)** Pregel

MapReduce

- MapReduce is a programming model for large dataset processing on clusters. The well known Apache Hadoop is implemented based on MapReduce.

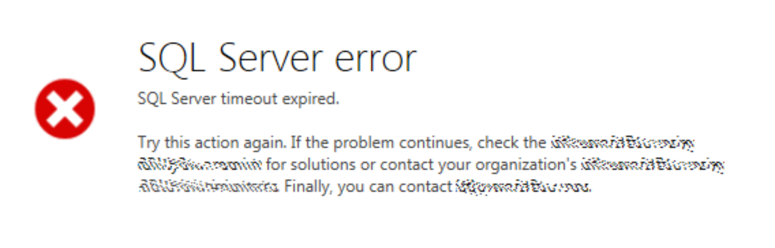


Optimizing over Coflows

- A coflow represents a task, and the task is deemed finished if all the flows in the coflow are finished.
- Instead of optimizing flow-level metrics, we should optimize the coflow-level metrics:
  - coflow completion time (CCT).
  - coflow deadline satisfaction (CDS).

Satisfying More Coflows

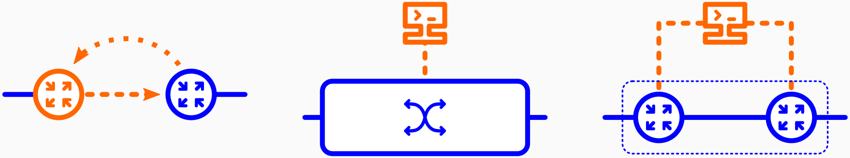
- The state-of-the-art methods aim to minimize the coflow completion time.
- However, meeting the deadline of a coflow can be more critical. $\Rightarrow$ How many deadlines can we satisfy within a horizon $[0, T]$?



C. Wilson et al., "Better Never Than Late: Meeting Deadlines in Datacenter Networks," 2011.

Model: Network Model

- Network-oblivious (decentralized): Baraat, Stream.
- Non-blocking switch: Orchestra, Varys, Aalo.
- Network-aware: RAPIER.



(a) Network-Oblivious **(b)** Non-Blocking Switch **(c)** Network-Aware

Model: Information Availability

- Offline: the information of all the flows is available.
- Online: the information of a flow is known only upon its arrival, including the deadline and the size.
- Myopic: no prior information is available unless it happens.

Model: Information Availability

- Offline: the information of all the flows is available.
- Online: the information of a flow is known only upon its arrival, including the deadline and the size.
- Myopic: no prior information is available unless it happens.

- We can intentionally schedule to satisfy the deadlines only when we know them before they happen.
  $\Rightarrow$ Offline and Online.

Summary of State-of-the-Art Methods

		Network Model		
		Network-Oblivious	Non-Blocking Switch	Network-Aware
Information Availability	Myopic	Baraat Stream	Orchestra Aalo	RAPIER
	Online	D-CAS	Varys	OMCoflow
	Offline		max-min utility	

Summary of State-of-the-Art Methods

		Network Model		
		Network-Oblivious	Non-Blocking Switch	Network-Aware
Information Availability	Myopic	Baraat Stream	Orchestra Aalo	RAPIER
	Online	D-CAS	Varys	OMCoflow OLPA
	Offline		max-min utility	LPA ILPA

Coflow Deadline Satisfaction Problem (CDS)

$$
\begin{aligned}
\max \quad & \sum_{n \in N} z^n \\
\text{s.t.} \quad & \sum_{\Delta_m \subseteq \tau_j} x_j(\Delta_m)\,|\Delta_m| = s_j z^n && \forall n \in N, j \in J^n \\
& z^n \in \{0,1\} && \forall n \in N \\
& \sum_{j \in J : e \in p_j} x_j(\Delta_m) \le c_e && \forall e \in E, \Delta_m \subseteq [0,T] \\
& x_j(\Delta_m) \ge 0 && \forall j \in J, \Delta_m \subseteq \tau_j \\
& x_j(\Delta_m) = 0 && \forall j \in J, \Delta_m \not\subseteq \tau_j
\end{aligned}
$$

NP-Hardness

Proposition 1
CDS is NP-hard and there exists no constant factor polynomial-time approximation algorithm for CDS unless $P = NP$.

- The proposition justifies the use of heuristics when approaching the problem.

Linear Programming Approximation (LPA)

$$
\begin{aligned}
\max \quad & \sum_{n \in N} z^n \\
\text{s.t.} \quad & \sum_{\Delta_m \subseteq \tau_j} x_j(\Delta_m)\,|\Delta_m| = s_j z^n && \forall n \in N, j \in J^n \\
& z^n \in \{0,1\} && \forall n \in N \\
& \sum_{j \in J : e \in p_j} x_j(\Delta_m) \le c_e && \forall e \in E, \Delta_m \subseteq [0,T] \\
& x_j(\Delta_m) \ge 0 && \forall j \in J, \Delta_m \subseteq \tau_j \\
& x_j(\Delta_m) = 0 && \forall j \in J, \Delta_m \not\subseteq \tau_j
\end{aligned}
$$

Linear Programming Approximation (LPA)

$$
\begin{aligned}
\max \quad & \sum_{n \in N} z^n & \\
\text{s.t.} \quad & \sum_{\Delta_m \subseteq \tau_j} x_j(\Delta_m)\,|\Delta_m| = s_j z^n & \forall n \in N, j \in J^n \\
& \underline{0 \leq z^n \leq 1} & \forall n \in N \\
& \sum_{j \in J : e \in p_j} x_j(\Delta_m) \leq c_e & \forall e \in E, \Delta_m \subseteq [0, T] \\
& x_j(\Delta_m) \geq 0 & \forall j \in J, \Delta_m \subseteq \tau_j \\
& x_j(\Delta_m) = 0 & \forall j \in J, \Delta_m \not\subseteq \tau_j
\end{aligned}
$$

Iterative Linear Programming Approximation (ILPA)

- LPA satisfies the coflows corresponding to $z^n = 1$. For those coflows with $z^n < 1$, LPA also allocates bandwidth to them, which is a waste of bandwidth.
- To prevent the drawback, we can remove a coflow whenever it is no longer possible to be satisfied.
- After removing the coflows that can never be satisfied, can we really find a better schedule through LPA?

Iterative Linear Programming Approximation (ILPA)

Algorithm 1: Iterative Linear Programming Approximation (ILPA)

```
for Δ_m from earliest to the last do
    Remove the coflows that cannot be satisfied anymore.
    Apply LPA to solve for new x_j(Δ_m), x_j(Δ_{m+1}), ....
    Adopt the new LPA schedule if
        1. more coflows can be satisfied, or
        2. the same number of coflows can be satisfied strictly
           earlier.
end for
```

Online Linear Programming Approximation (OLPA)

- We can generalize the idea of ILPA to the online scenario.

Algorithm 2: Online Linear Programming Approximation (OLPA)

---

for whenever a flow arrives, expires, or finishes **do**
    Remove the coflows that cannot be satisfied anymore.
    Apply ILPA to schedule the satisfiable coflows.
    Adopt the new ILPA schedule if

1. more coflows can be satisfied, or
2. the same number of coflows can be satisfied strictly earlier.

end for

Comparison with State-of-the-Art Methods

		Network Model		
		Network-Oblivious	Non-Blocking Switch	Network-Aware
Information Availability	Myopic	Baraat Stream	Orchestra Aalo	RAPIER
	Online	D-CAS	Varys	OMCoflow OLPA
	Offline		max-min utility	LPA ILPA

Comparison with State-of-the-Art Methods

		Network Model		
		Network-Oblivious	Non-Blocking Switch	Network-Aware
Information Availability	Myopic	Baraat Stream	Orchestra Aalo	RAPIER
	Online	D-CAS	Varys	OMCoflow OLPA
	Offline		max-min utility	LPA ILPA

Varys, Aalo, and RAPIER

- Varys (M. Chowdhury et al., 2014)
  - Smallest-Effective-Bottleneck-First (SEBF) for coflow completion time minimization: the same as the shortest remaining time first.
  - Earliest deadline first for deadline satisfaction.

- Aalo (M. Chowdhury and I. Stoica, 2015)
  - Discretized Coflow-Aware Least-Attained Service (D-CLAS): multi-level queue scheduling, which prioritizes the coflows based on received sizes.
  - Bandwidth assignment to the flows in a coflow: min-max fair sharing.

Varys, Aalo, and RAPIER

- RAPIER (Y. Zhao et al., 2015)
    - Emphasizing on the combination of routing and scheduling. Here we only test its scheduling.
    - RAPIER schedules as Varys, but instead of considering only the in/out port capacity constraints, it considers the bottleneck of the whole network.

Simulations

- We conduct simulations on ns-3.
- Within the horizon $T = 100$ ms, we generate coflows according to a Poisson process with different means of interarrival time.
- Each coflow is a MapReduce job consisting of 1 to 3 mappers and reducers, which are selected from leaf nodes of the fat-tree network.
- Each reducer requires a data size uniformly distributed over $[1, 100]$ MB from every mapper.

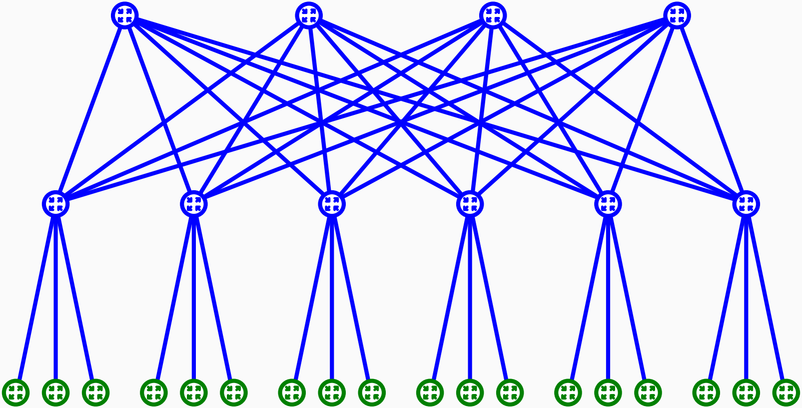


Figure 3: The fat-tree topology. Each link has capacity 10 Gbps.

Simulations

- The lifespan is set according to the tightness parameter q:

$$\tau_j = q \times \text{minimum possible lifespan of the flow.}$$

  Larger $q \Leftrightarrow$ more room for scheduling.

- The satisfaction ratio of a schedule is:

$$\text{satisfaction ratio} = \frac{\text{number of satisfied coflows}}{\text{total number of coflows}}.$$

  Larger satisfaction ratio $\Leftrightarrow$ more flows satisfied.

Simulations

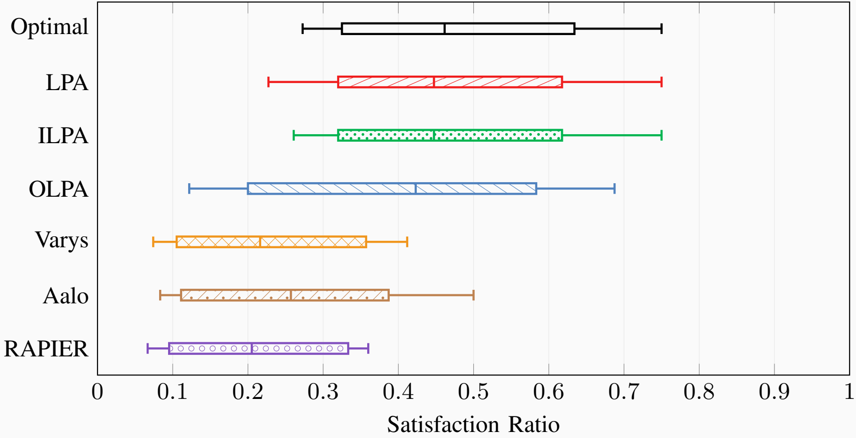


Figure 4: The $1^{\text{st}} - 5^{\text{th}} - 50^{\text{th}} - 95^{\text{th}} - 99^{\text{th}}$ percentiles under $q = 2$ and mean of interarrival time $= 3$ ms.

Simulations

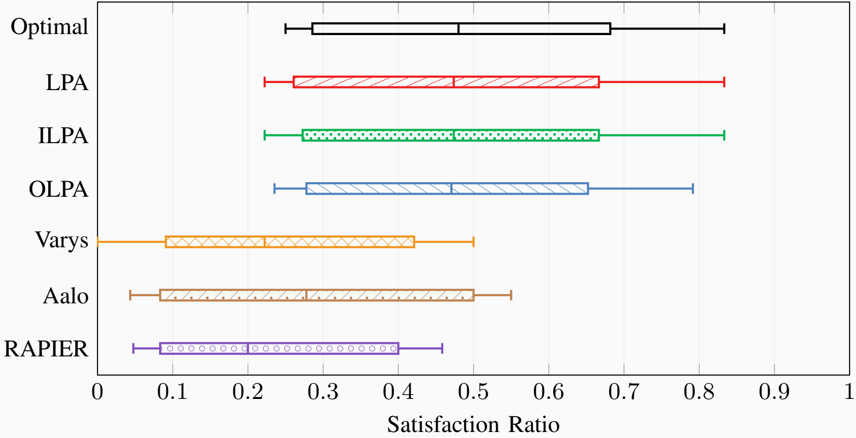


Figure 5: The $1^{\text{st}} - 5^{\text{th}} - 50^{\text{th}} - 95^{\text{th}} - 99^{\text{th}}$ percentiles under $q = 2$ and mean of interarrival time $= 5$ ms.

Simulations

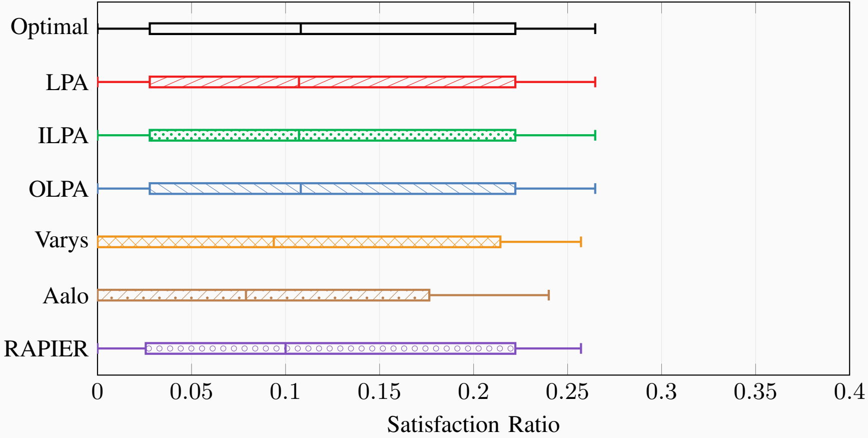


Figure 6: The $1^{\mathrm{st}} - 5^{\mathrm{th}} - 50^{\mathrm{th}} - 95^{\mathrm{th}} - 99^{\mathrm{th}}$ percentiles under $q = 1$ and mean of interarrival time $= 3$ ms.

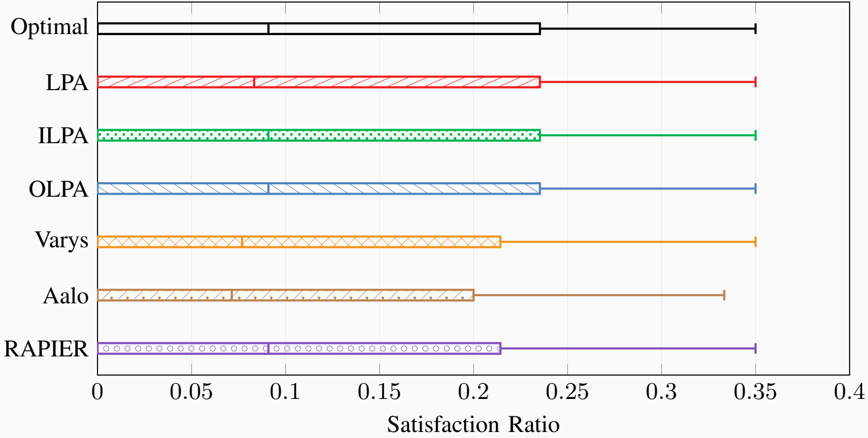


Figure 7: The $1^{\mathrm{st}} - 5^{\mathrm{th}} - 50^{\mathrm{th}} - 95^{\mathrm{th}} - 99^{\mathrm{th}}$ percentiles under $q = 1$ and mean of interarrival time $= 5$ ms.

Conclusion

- The coflow deadline scheduling problem is NP-hard. Moreover, it cannot be approximated within a constant factor in polynomial time (unless $P = NP$).
- We develop optimization-based offline and online algorithms.
- Simulation results show that the proposed algorithms are effective.

Questions & Answers

References

- J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
- A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: A warehousing solution over a map-reduce framework," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1626–1629, 2009.
- G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proc. ACM SIGMOD*. ACM, 2010, pp. 135–146.

References

M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *HotNets*. ACM, 2012, pp. 31–36.

C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," *ACM SIGCOMM CCR*, vol. 41, no. 4, pp. 50–61, 2011.

F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, "Decentralized task-aware scheduling for data center networks," *ACM SIGCOMM CCR*, vol. 44, no. 4, pp. 431–442, 2014.

References

- H. Susanto, H. Jin, and K. Chen, "Stream: Decentralized opportunistic inter-coflow scheduling for datacenter networks," in *IEEE ICNP*. IEEE, 2016, pp. 1–10.
- M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," *ACM SIGCOMM CCR*, vol. 41, no. 4, pp. 98–109, 2011.
- M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," *ACM SIGCOMM CCR*, vol. 45, no. 4, pp. 393–406, 2015.

References

- Y. Zhao, K. Chen, W. Bai, M. Yu, C. Tian, Y. Geng, Y. Zhang, D. Li, and S. Wang, "RAPIER: Integrating routing and scheduling for coflow-aware data center networks," in *Proc. IEEE INFOCOM*. IEEE, 2015, pp. 424–432.
- S. Luo, H. Yu, Y. Zhao, B. Wu, S. Wang *et al.*, "Minimizing average coflow completion time with decentralized scheduling," in *Proc. IEEE ICC*. IEEE, 2015, pp. 307–312.
- M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with Varys," *ACM SIGCOMM CCR*, vol. 44, no. 4, pp. 443–454, 2014.

References

- Y. Li, S. H.-C. Jiang, H. Tan, C. Zhang, G. Chen, J. Zhou, and F. Lau, "Efficient online coflow routing and scheduling," in *Proc. ACM MOBICHOC*. ACM, 2016, pp. 161–170.
- L. Chen, W. Cui, B. Li, and B. Li, "Optimizing coflow completion times with utility max-min fairness," in *Proc. IEEE INFOCOM*. IEEE, 2016, pp. 1–9.