# Varys

*Efficient Coflow Scheduling*

Mosharaf Chowdhury,
Yuan Zhong, Ion Stoica

# Communication is Crucial

# Performance

Facebook analytics jobs spend **33%** of their runtime in communication[1]

*As in-memory systems proliferate,*
*the network is likely to become the* ***primary bottleneck***

*1. Managing Data Transfers in Computer Clusters with Orchestra, SIGCOMM'2011*

# Flow

A sequence of packets between two endpoints

Independent unit of allocation, sharing, load balancing, and/or prioritization

# Optimizing Communication Performance: Networking Approach

*"Let systems figure it out"*

# Optimizing Communication Performance: Systems Approach

*"Let users figure it out"*

| | # Comm. Params* |
|---|---|
| **Spark 1.0.1** | **6** |
| **Hadoop 1.0.4** | **10** |
| **YARN 2.3.0** | **20** |

*Lower bound. Does not include *many* parameters that can indirectly impact communication; e.g., number of reducers etc. Also excludes control-plane communication/RPC parameters.

# Optimizing Communication Performance: Systems Approach

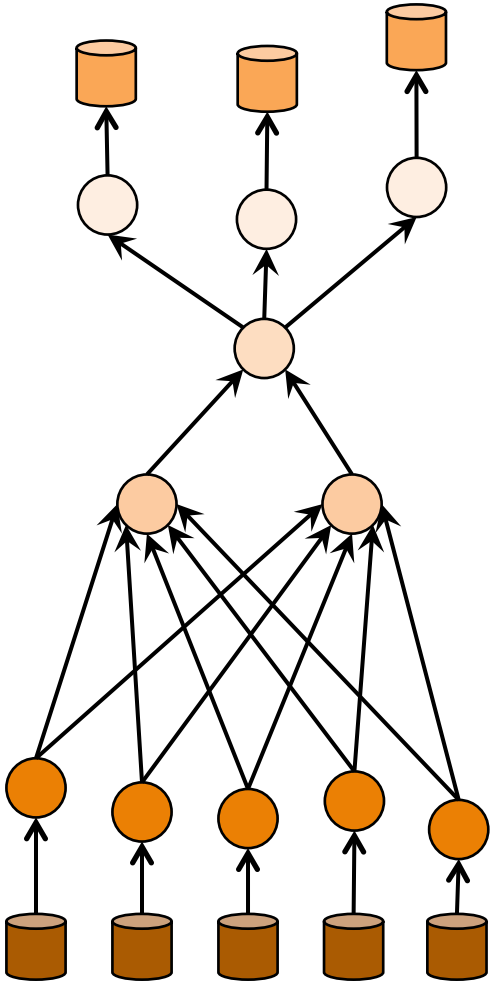
*"Let users figure it out"*

# Optimizing Communication Performance: Networking Approach

*"Let systems figure it out"*

# Optimizing Communication Performance: Systems Approach

*"Let users figure it out"*



# Optimizing Communication Performance: Networking Approach

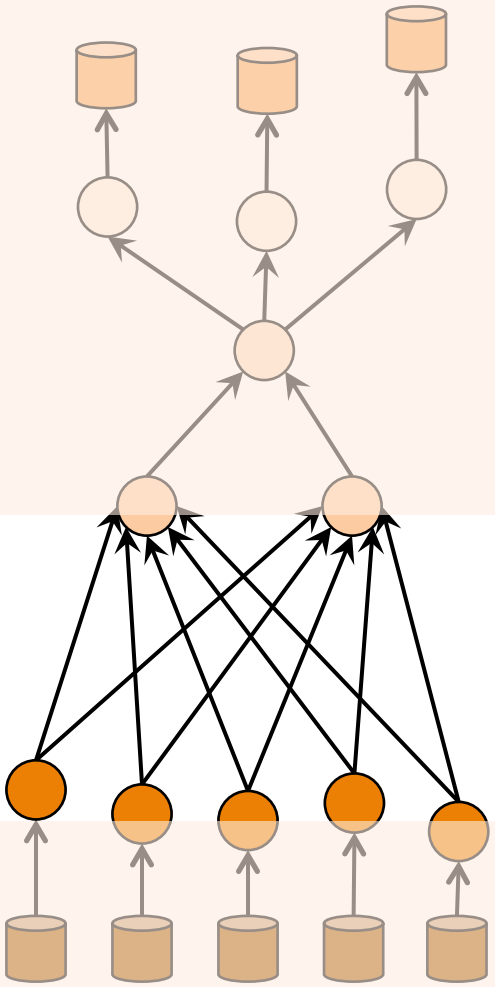
*"Let systems figure it out"*

# Coflow[1]

*A collection of parallel flows*

*Distributed endpoints*

*Each flow is independent*



Completion time depends on *the last flow* to complete

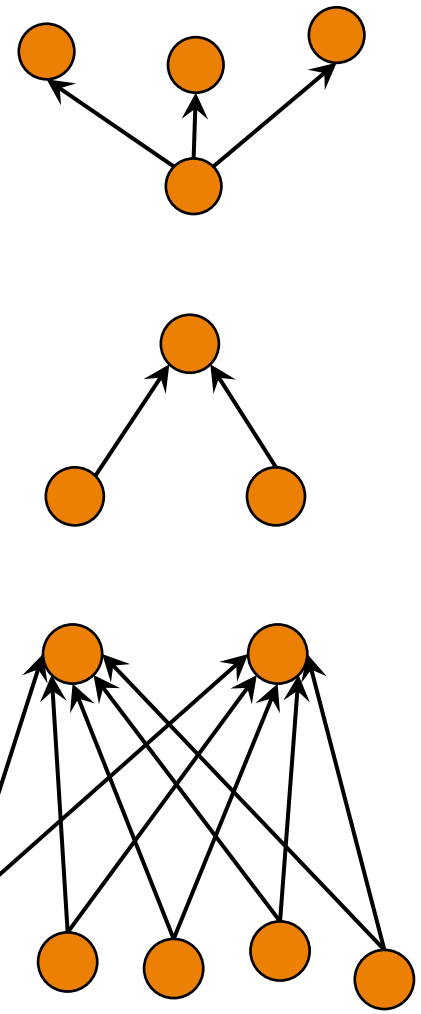
*1. Coflow: A Networking Abstraction for Cluster Applications, HotNets'2012*

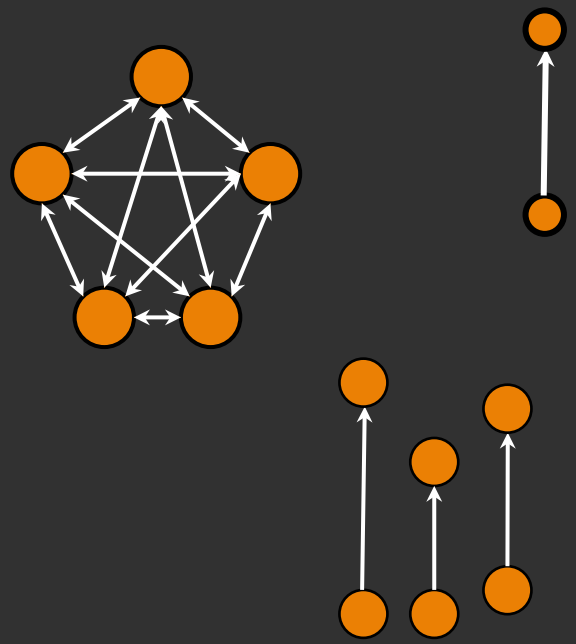
# Coflow[1]

*A collection of parallel flows*

*Distributed endpoints*

*Each flow is independent*

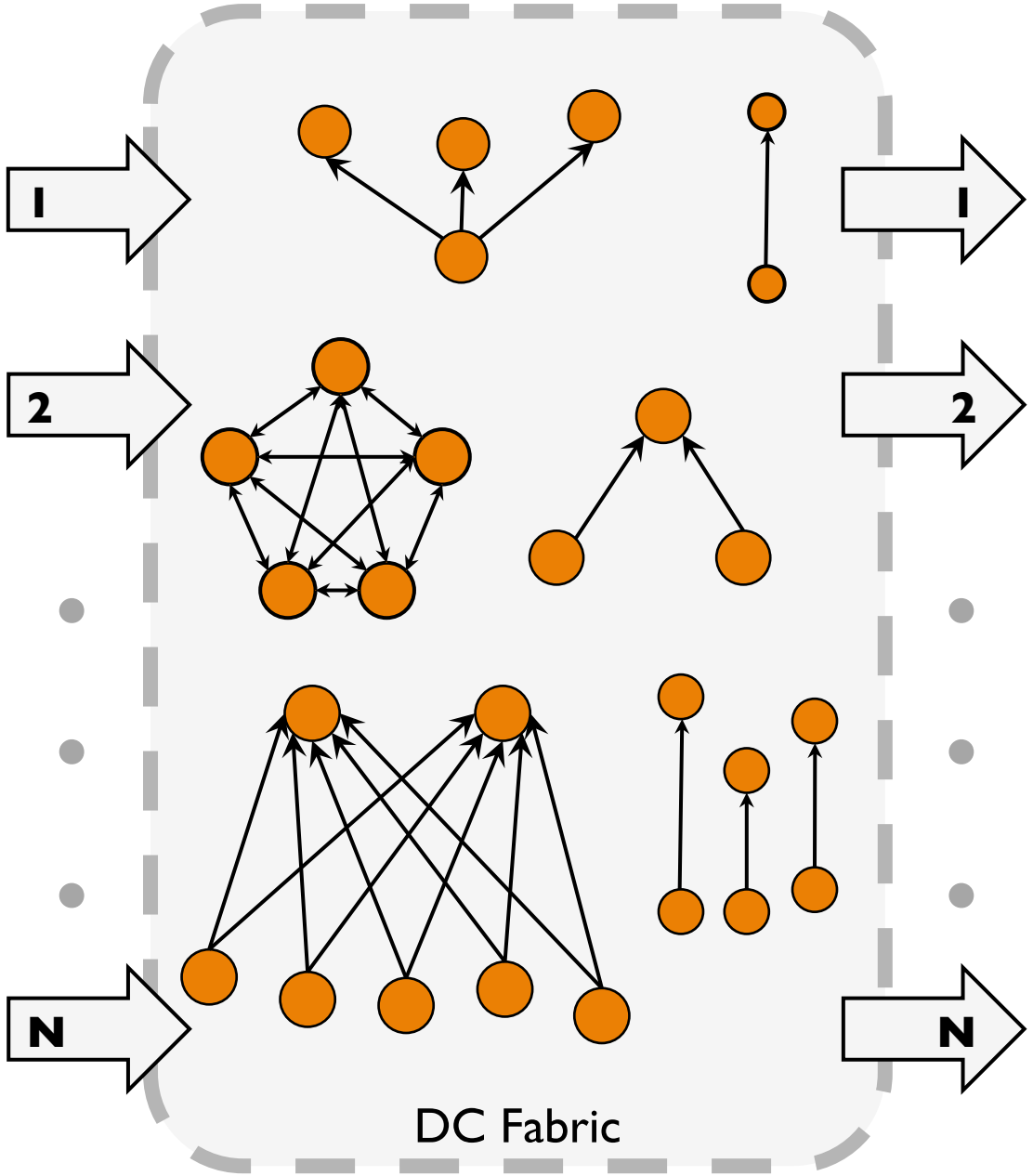




Completion time depends on *the last flow* to complete

*1. Coflow: A Networking Abstraction for Cluster Applications, HotNets'2012*

# How to schedule coflows ...




... for faster
#1 completion of coflows?

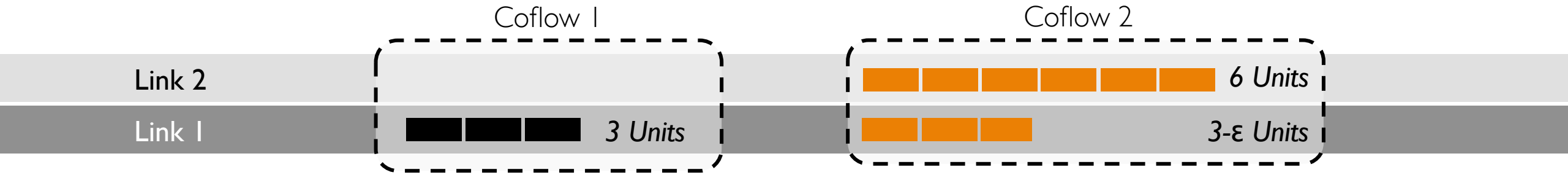
... to meet
#2 more deadlines?

# Varys

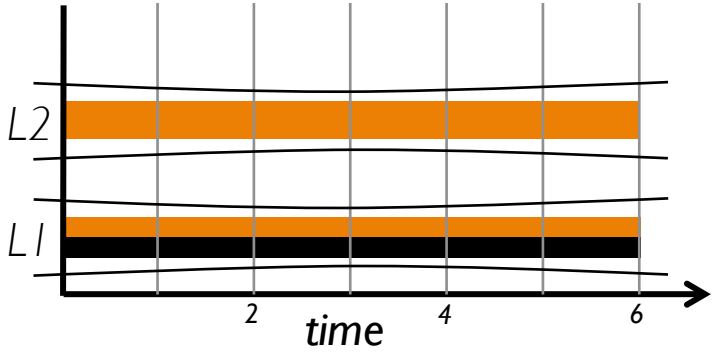
*Enables coflows in data-intensive clusters*

1. **Simpler Frameworks** — *Zero user-side configuration using a simple coflow API*
2. **Better performance** — *Faster and more predictable transfers through coflow scheduling*

# Benefits of Inter-Coflow Scheduling

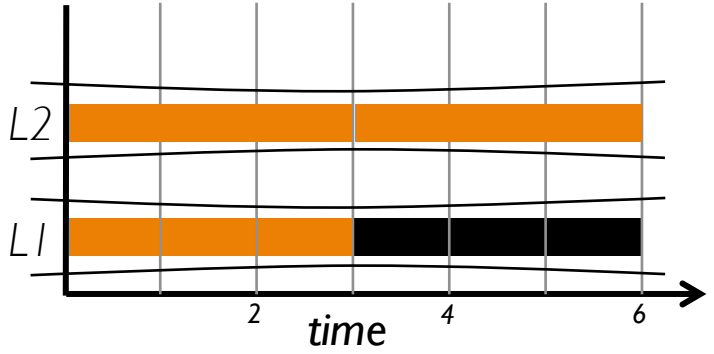



**Fair Sharing**



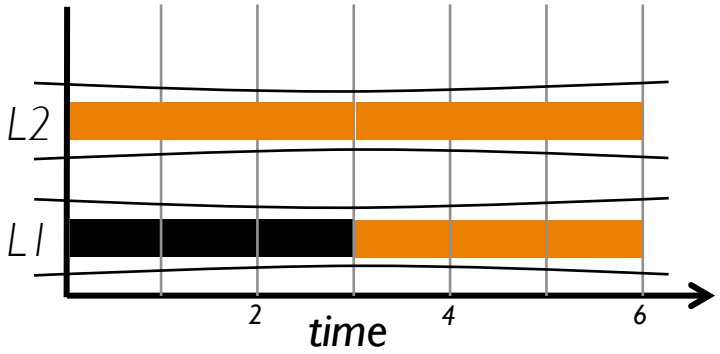

*Coflow1 comp. time = 6*
*Coflow2 comp. time = 6*

**Flow-level Prioritization[1,2]**




*Coflow1 comp. time = **6***
*Coflow2 comp. time = **6***

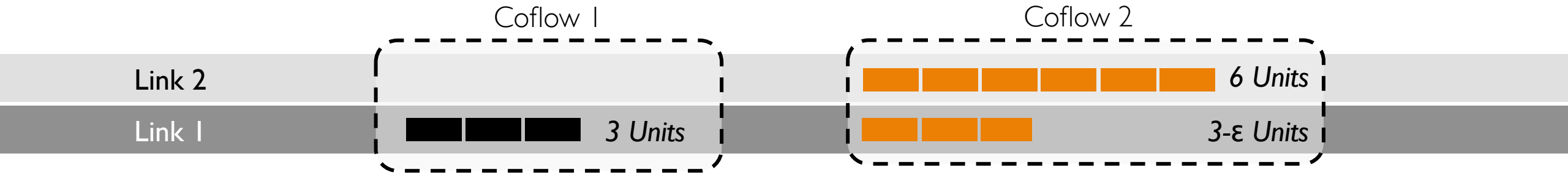
**The Optimal**




*Coflow1 comp. time = **3***
*Coflow2 comp. time = **6***


*1. Finishing Flows Quickly with Preemptive Scheduling, SIGCOMM'2012.*
*2. pFabric: Minimal Near-Optimal Datacenter Transport, SIGCOMM'2013.*

# Inter-Coflow Scheduling

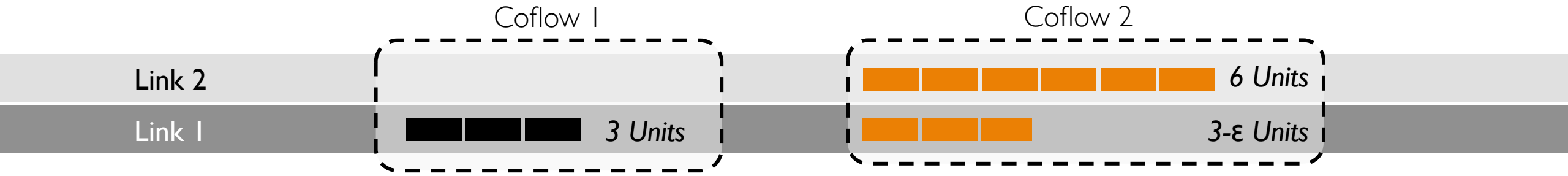



## Concurrent Open Shop Scheduling[1]

- Tasks on independent machines
- Examples include job scheduling and caching blocks
- Use a **ordering** heuristic

1. *A note on the complexity of the concurrent open shop problem, Journal of Scheduling, 9(4):389–396, 2006*

# Inter-Coflow Scheduling is NP-Hard



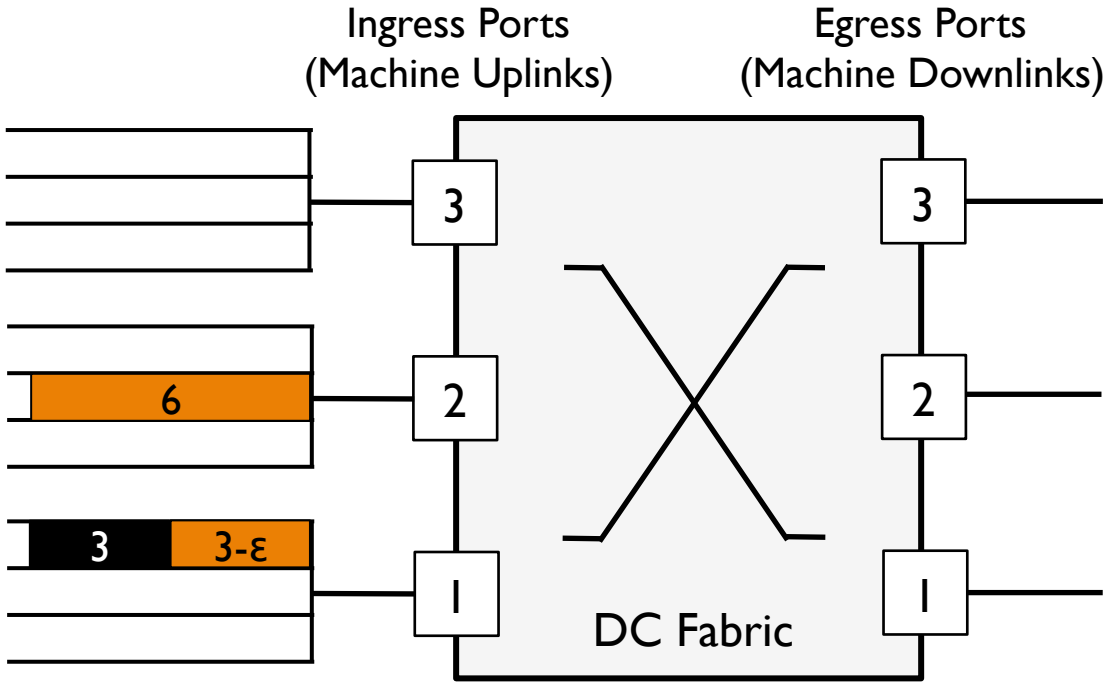

*with coupled resources*

Concurrent Open Shop Scheduling^

- Flows on dependent links
- Consider **ordering** and **matching** constraints

Characterized COSS-CR

Proved that list scheduling might not result in optimal solution

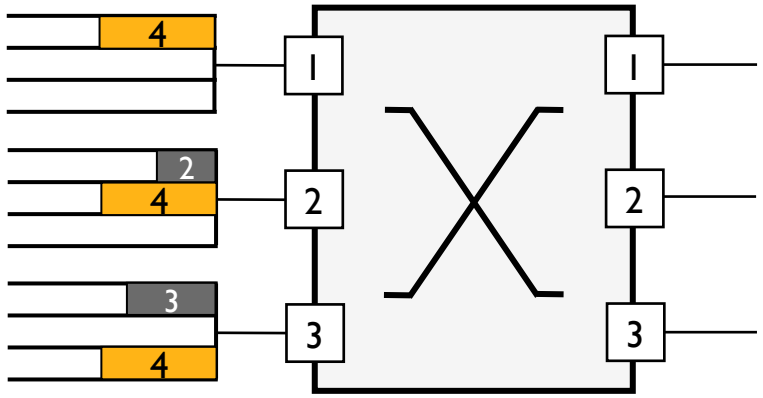


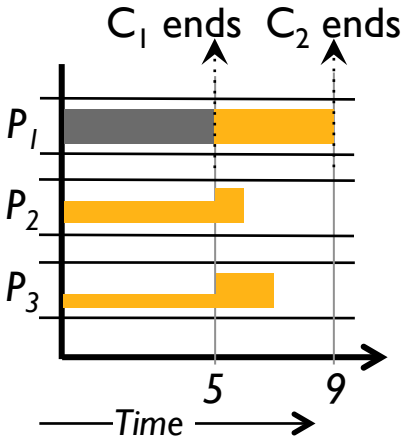
# Varys

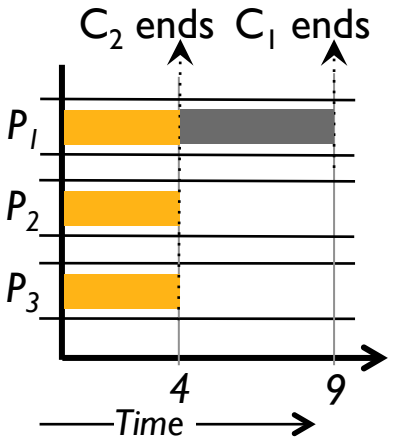
*Employs a two-step algorithm to minimize coflow completion times*

1. **Ordering heuristic** — *Keeps an ordered list of coflows to be scheduled, preempting if needed*
2. **Allocation algorithm** — *Allocates minimum required resources to each coflow to finish in minimum time*

# Ordering Heuristic: SEBF




| | $C_1$ | $C_2$ |
|---|---|---|
| Length | 3 | 4 |
| Width | 2 | 3 |
| Size | 5 | 12 |
| Bottleneck | **5** | 4 |

*Shortest-First*

*Narrowest-First*

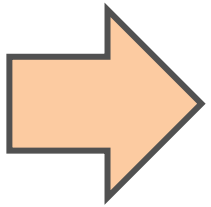
*Smallest-First*

*Smallest-Effective-Bottleneck-First*

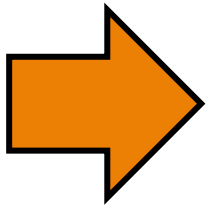
# Allocation Algorithm

**MADD**

*A coflow cannot finish before its very last flow*



*Finishing flows faster than the bottleneck cannot decrease a coflow's completion time*



*Ensure minimum allocation to each flow for it to finish at the desired duration;*

*for example,*
*at bottleneck's completion, or*
*at the deadline.*

# Varys

*Enables frameworks to take advantage of coflow scheduling*

1. Exposes the coflow API
2. Enforces through a centralized scheduler

# Evaluation

*A 3000-node trace-driven simulation matched against a 100-node EC2 deployment*

1. Does it improve performance?
2. Can it beat non-preemptive solutions?

**YES**

# Faster Jobs

| | Comm. Improv. | Job Improv. |
|---|---|---|
| Avg. | **1.85X** | **1.25X** |
| 95th | **1.74X** | **1.15X** |

# Faster Jobs

| | Comm. Improv. | Comm. **Heavy**[1] Job Improv. |
|---|---|---|
| Avg. | **3.16X** | **2.50X** |
| 95th | **3.84X** | **2.94X** |

*1. 26% jobs spend at least 50% of their duration in communication stages.*

# Better than Non-Preemptive Solutions

| | w.r.t. FIFO[1] |
|---|---|
| Avg. | **5.65X** |
| 95th | **7.70X** |

**NO**

Perpetual Starvation?

*1. Managing Data Transfers in Computer Clusters with Orchestra, SIGCOMM'2011*

# #1

## Coflow Dependencies

# #2

## Unknown Flow Information

# #3

## Decentralized Varys

in the Context of *Multipoint-to-Multipoint* Coflows

# #4

## Theory Behind
## “Concurrent Open Shop Scheduling *with* Coupled Resources”

# Varys

*Greedily schedules coflows without worrying about flow-level metrics*

- Consolidates network optimization of data-intensive frameworks
- Improves job performance by addressing the COSS-CR problem
- Increases predictability through informed admission control

http://varys.net/

Mosharaf Chowdhury - @mosharaf