Efficient Coflow Scheduling in Data Center Networks

Yuan Zhong

Columbia University

Joint work with Mosharaf Chowdhury, Zhen Qiu, Cliff Stein and Ion Stoica

Outline

- Motivation and Model
  - The (offline) coflow scheduling problem
- Theory
  - Main result: O(1)-poly time approx. algorithm
  - Proof sketch: (a) LP relaxation; (b) matching schedules
- Systems and Numerical Experiments
- Discussion

Motivation

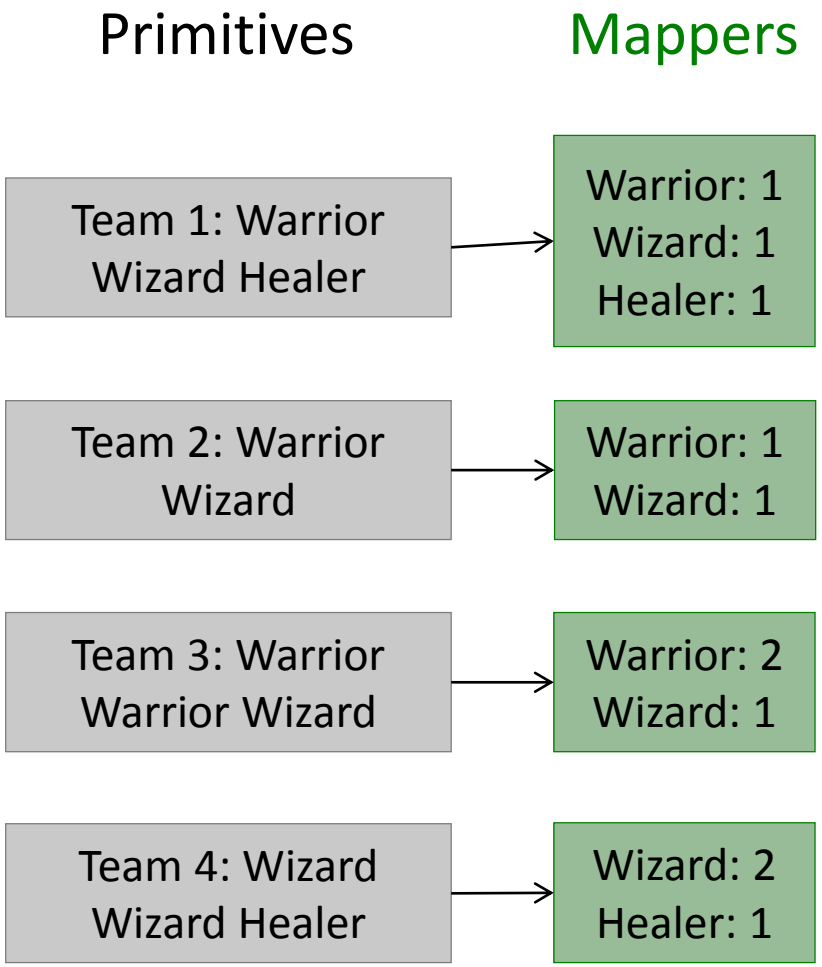
Motivation

- Proliferation of data-parallel computation frameworks: *MapReduce*, *Spark*, *Google Dataflow*, etc
- Process large-scale data-processing jobs in data centers
- Idea: break into smaller pieces that can be processed in parallel
- Simple example of *Mapreduce*: count total #s of characters of different types in an online RPG game

Motivation

Primitives

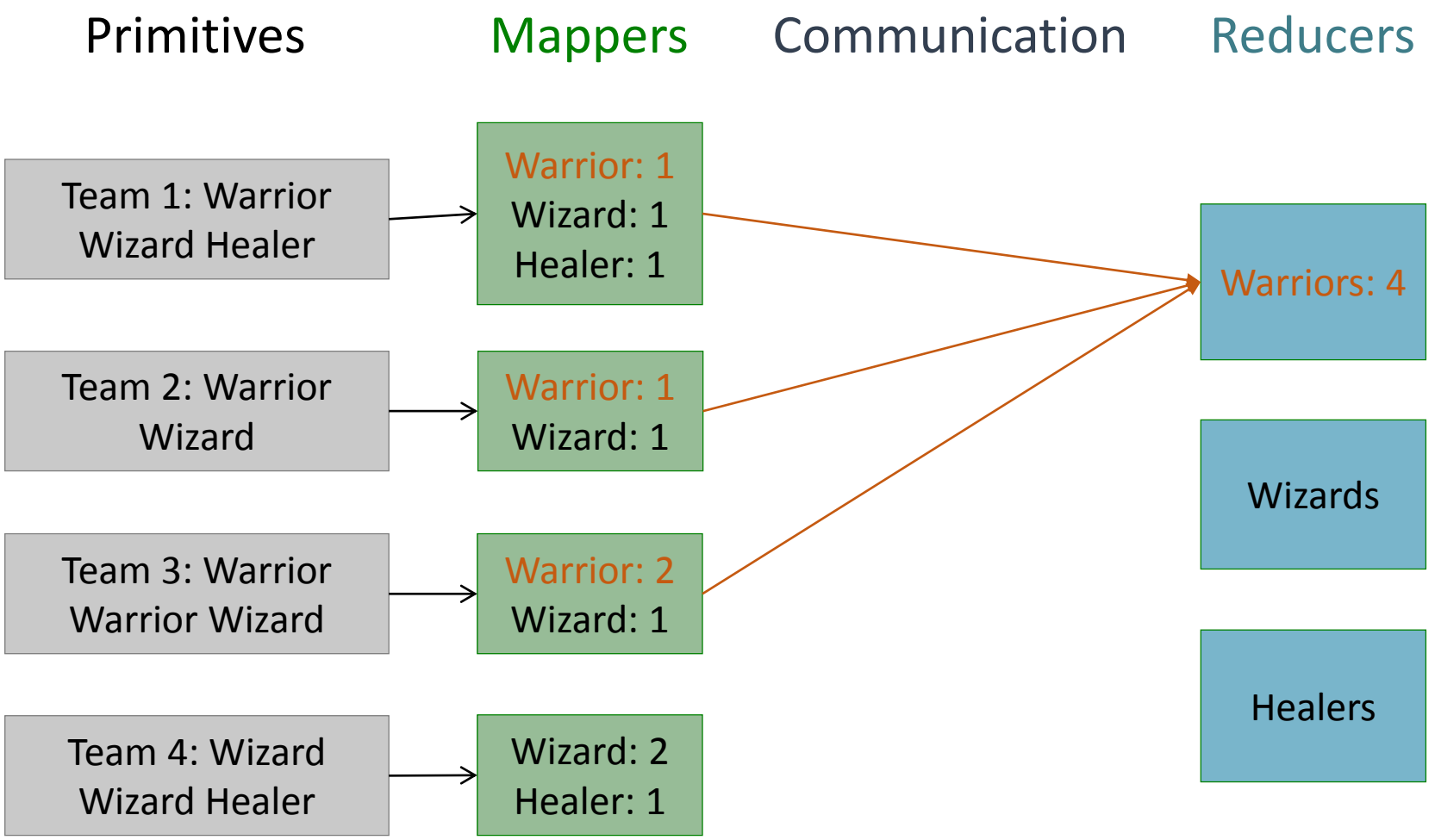
Motivation


Primitives
Mappers
Team 1: Warrior
Wizard Healer
Warrior: 1
Wizard: 1
Healer: 1
Team 2: Warrior
Wizard
Warrior: 1
Wizard: 1
Team 3: Warrior
Warrior Wizard
Warrior: 2
Wizard: 1
Team 4: Wizard
Wizard Healer
Wizard: 2
Healer: 1

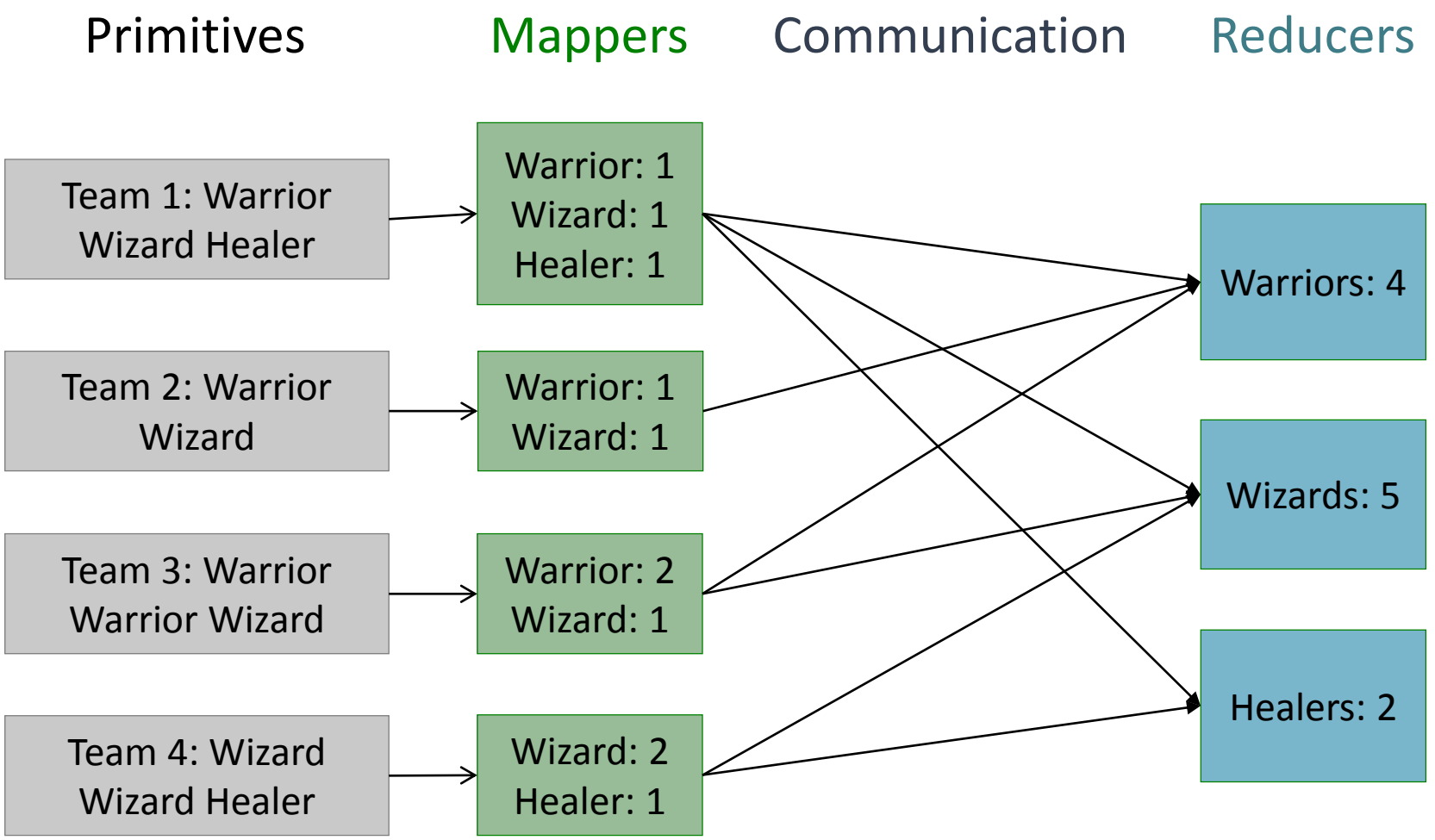
Motivation

Primitives
Mappers
Reducers
Team 1: Warrior Wizard Healer
Warrior: 1 Wizard: 1 Healer: 1
Team 2: Warrior Wizard
Warrior: 1 Wizard: 1
Team 3: Warrior Warrior Wizard
Warrior: 2 Wizard: 1
Team 4: Wizard Wizard Healer
Wizard: 2 Healer: 1
Warriors
Wizards
Healers

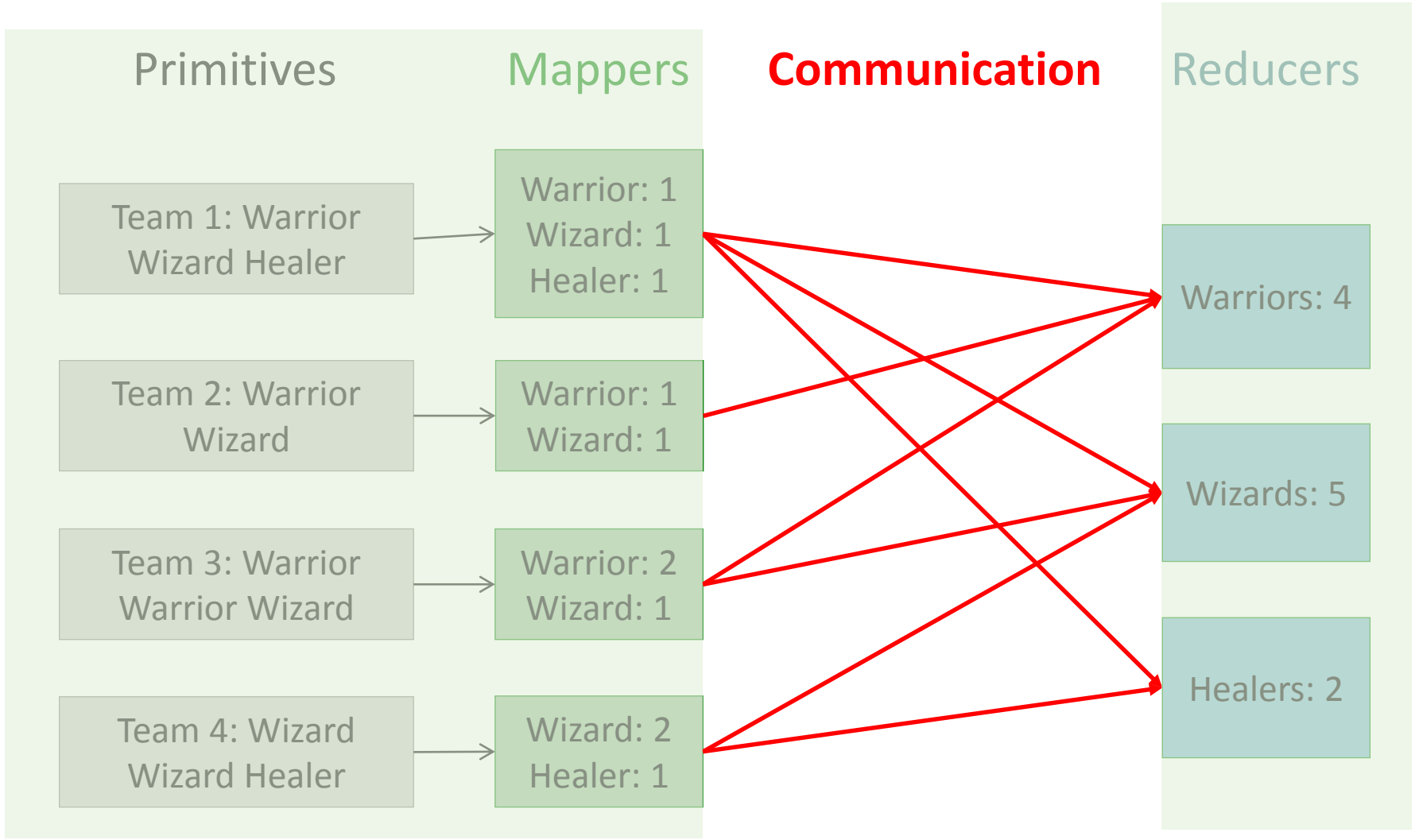
Motivation


Primitives
Mappers
Communication
Reducers
Team 1: Warrior
Wizard Healer
Team 2: Warrior
Wizard
Team 3: Warrior
Warrior Wizard
Team 4: Wizard
Wizard Healer
Warrior: 1
Wizard: 1
Healer: 1
Warrior: 1
Wizard: 1
Warrior: 2
Wizard: 1
Wizard: 2
Healer: 1
Warriors: 4
Wizards
Healers

Motivation


Primitives
Mappers
Communication
Reducers
Team 1: Warrior Wizard Healer
Team 2: Warrior Wizard
Team 3: Warrior Warrior Wizard
Team 4: Wizard Wizard Healer
Warrior: 1 Wizard: 1 Healer: 1
Warrior: 1 Wizard: 1
Warrior: 2 Wizard: 1
Wizard: 2 Healer: 1
Warriors: 4
Wizards: 5
Healers: 2

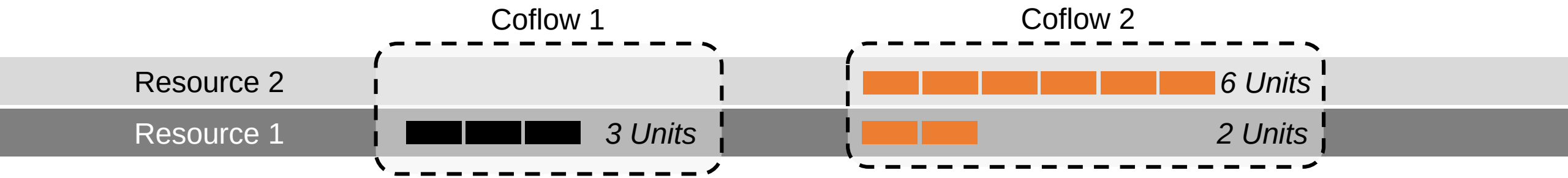
Motivation


Primitives
Mappers
Communication
Reducers
Team 1: Warrior
Wizard Healer
Team 2: Warrior
Wizard
Team 3: Warrior
Warrior Wizard
Team 4: Wizard
Wizard Healer
Warrior: 1
Wizard: 1
Healer: 1
Warrior: 1
Wizard: 1
Warrior: 2
Wizard: 1
Wizard: 2
Healer: 1
Warriors: 4
Wizards: 5
Healers: 2

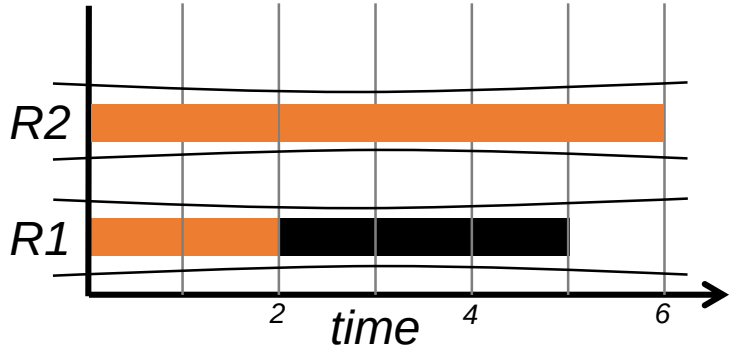
Motivation

- Communication is important for performance!
- Facebook jobs spend ~**25%** of runtime in intermediate communication
- Characteristics of a typical communication pattern
  - Call it a ***coflow***
  - A **Co**llection of many parallel data transfers/**flow**s
  - Completion time is measured by the time the last data flow finishes

Motivation: An Illustrate Example

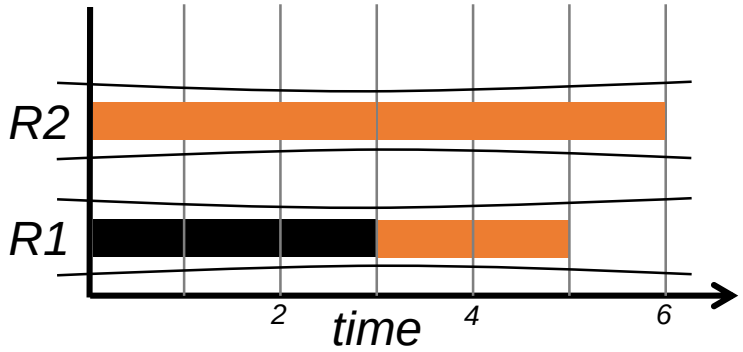


Shortest Processing Time First



Coflow1 comp. time = 5
Coflow2 comp. time = 6

The Optimal



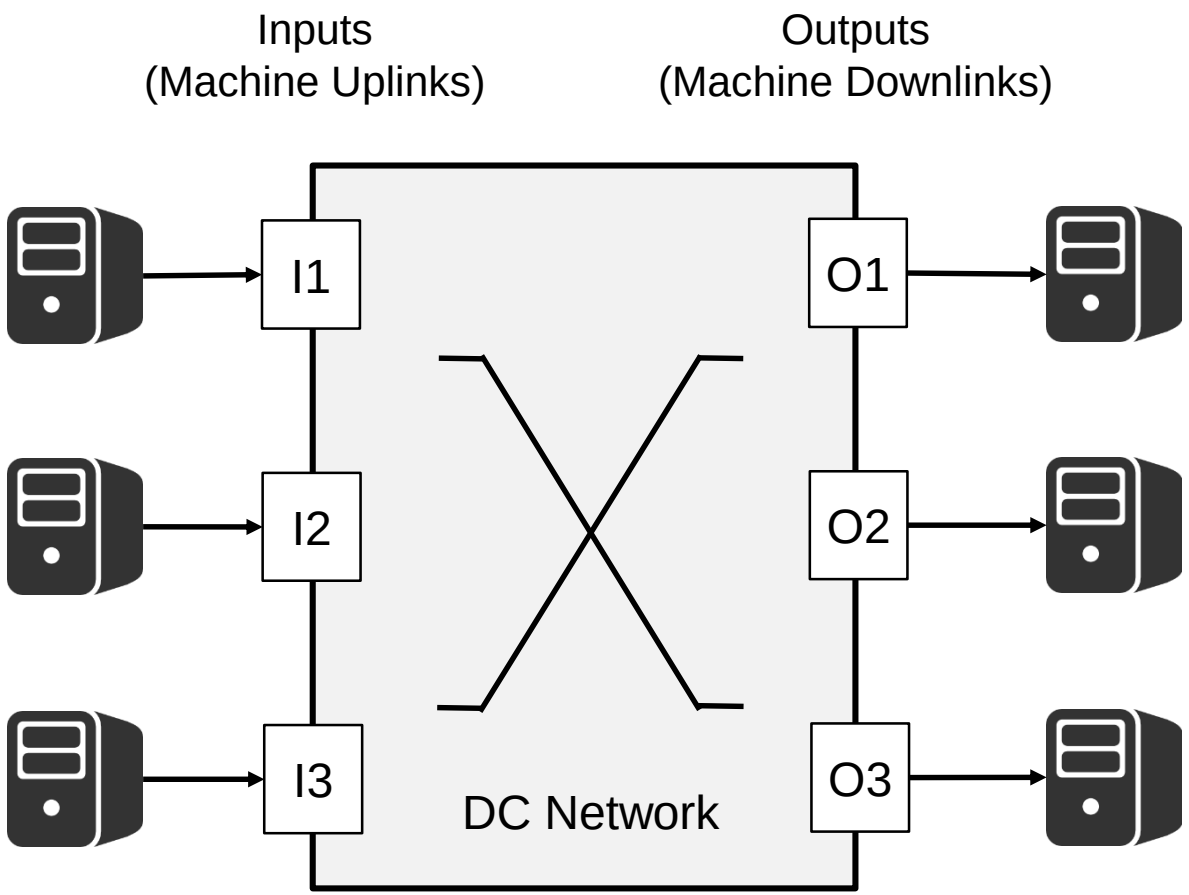
*Coflow1 comp. time = **3***
Coflow2 comp. time = 6

Motivation: An Illustrative Example

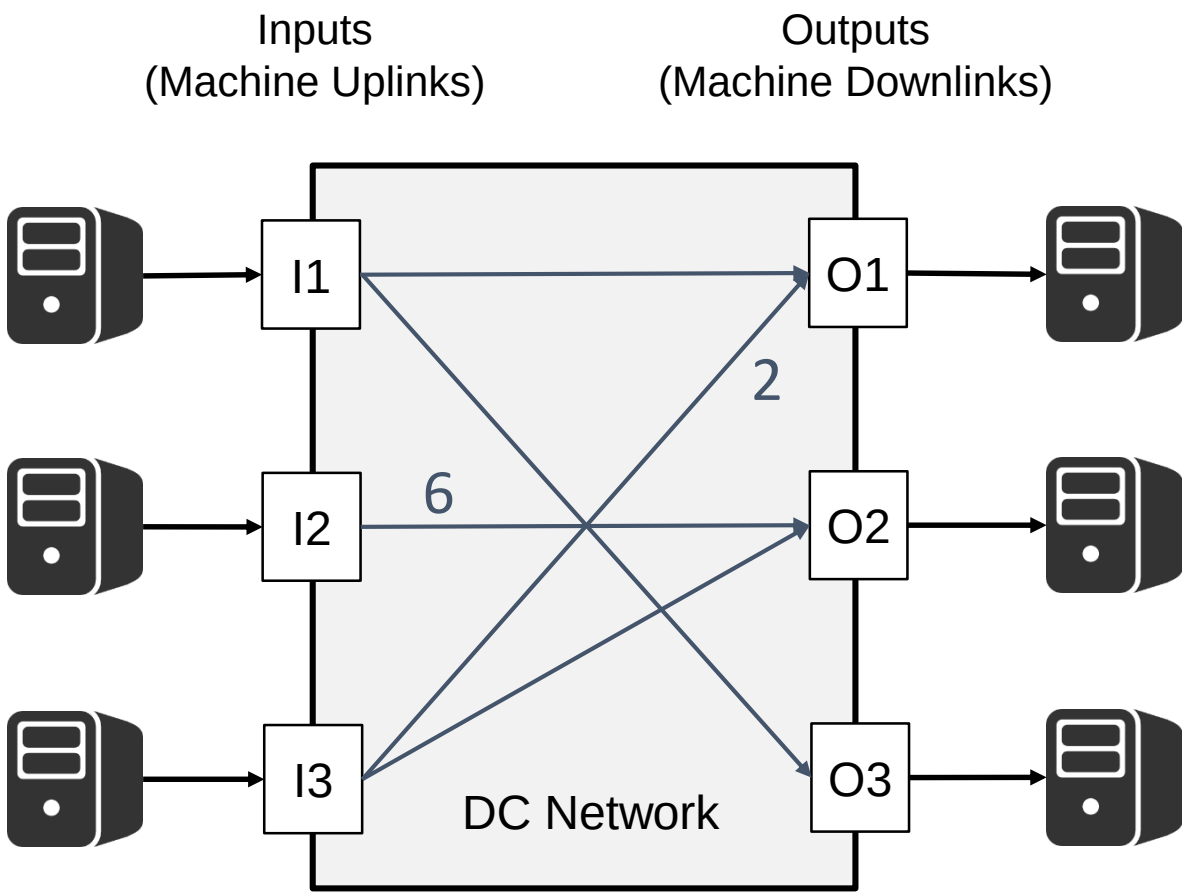
- Coordination needed among scheduling resources
- Example is an instance of the *concurrent open shop scheduling* problem
  - m machines
  - Processing requirement of a job is a vector (p_1, p_2, …, p_m) (p_i on machine i)
  - Completion time is the time the last p_i finishes
- Coflow scheduling problem is more complicated: resources at both the input and output side

Model

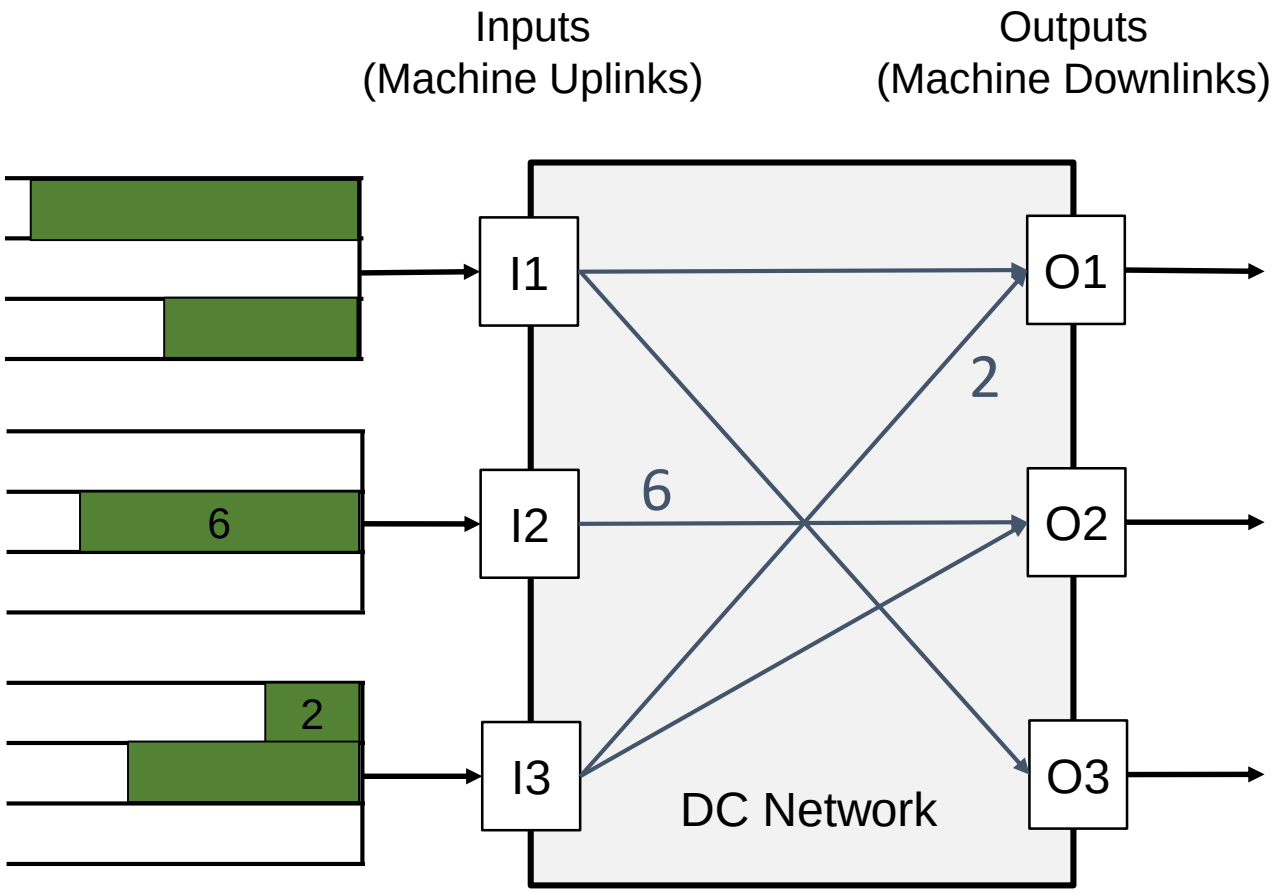
Model


Inputs
(Machine Uplinks)
Outputs
(Machine Downlinks)
I1
I2
I3
O1
O2
O3
DC Network

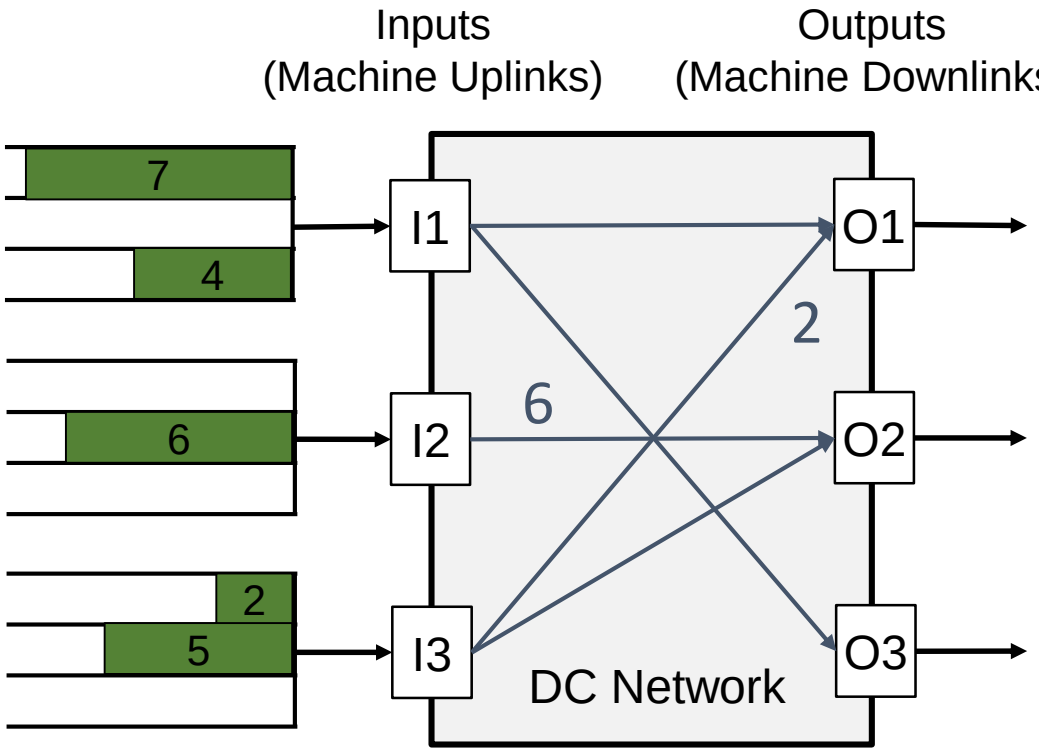
Model


Inputs
(Machine Uplinks)
Outputs
(Machine Downlinks)
I1
I2
I3
O1
O2
O3
2
6
DC Network

Model

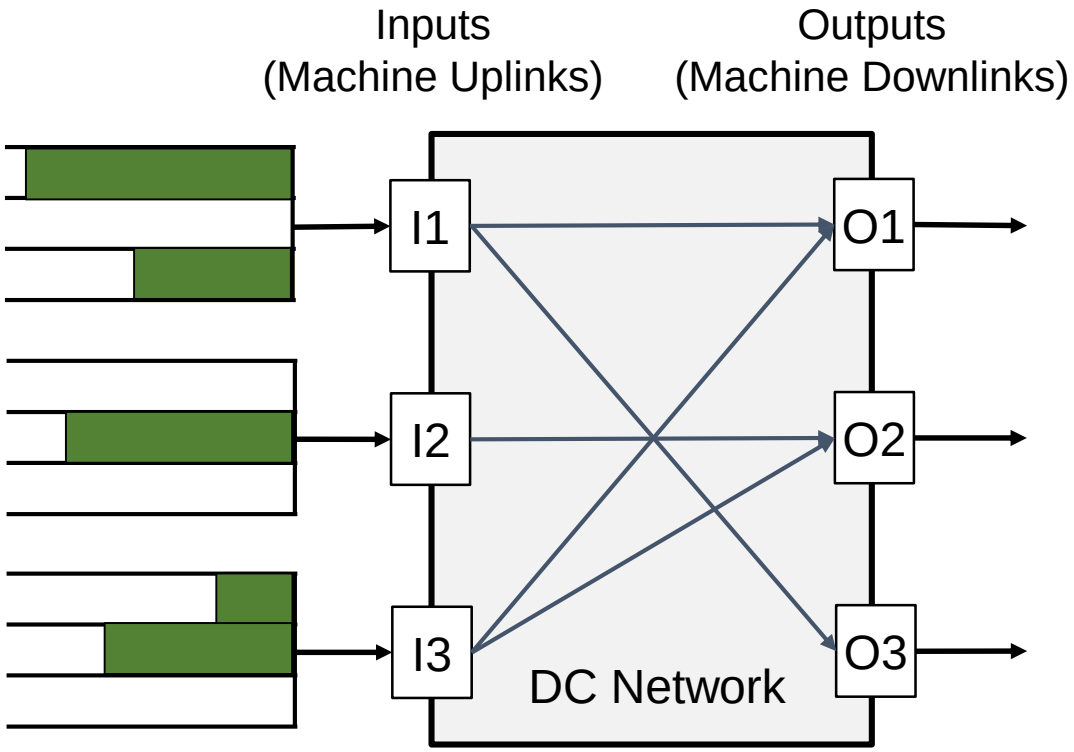

Inputs
(Machine Uplinks)
Outputs
(Machine Downlinks)
I1
I2
I3
O1
O2
O3
6
2
6
2
DC Network

Model



- m inputs, m outputs (m=3 here)
- n coflows released at time 0
- Coflow k represented by matrix $D^{(k)}$
  - $D^{(k)} = \left(d_{ij}^{(k)}\right)_{i,j=1}^{m}$
  - E.g., $d_{22} = 6, d_{31} = 2$
  - $D = \begin{pmatrix} 7 & 0 & 4 \\ 0 & 6 & 0 \\ 2 & 5 & 0 \end{pmatrix}$

Model



- Capacity = 1 uniformly at all inputs and outputs
- Constraints: A feasible schedule specified by a **matching** between inputs and outputs
- Matchings rep. by perm. matrices
- E.g. $\begin{pmatrix} 1 & 2 \\ 2 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + 2\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ can be completed in 3 time slots

Model

Problem statement

- n coflows $D^{(k)} = \left(d_{ij}^{(k)}\right)_{i,j=1}^{m}, k = 1, 2, \dots, n$
- Weight w_k of coflow k
- Objective: Minimize $\sum_{k=1}^{n} w_k C_k$ subject to scheduling constraints
  - C_k: completion time variable for coflow k

Theory

Exact Formulation

$$\text{Minimize} \sum_{k=1}^{n} w_k C_k \qquad \text{subject to}$$

$$\sum_{t=1}^{C_k} y(i,j,k,t) \geq d_{ij}^{(k)}, \text{ for } i,j = 1,\ldots,m, k = 1,\ldots,n; \quad \text{(load)} \qquad (1)$$

$$\sum_{k=1}^{n}\sum_{j'=1}^{m} y(i,j',k,t) \leq 1, \text{ for } i = 1,\ldots,m, \forall t; \qquad \text{(matching)} \qquad (2)$$

$$\sum_{k=1}^{n}\sum_{i'=1}^{m} y(i',j,k,t) \leq 1, \text{ for } j = 1,\ldots,m, \forall t; \qquad \text{(matching)} \qquad (3)$$

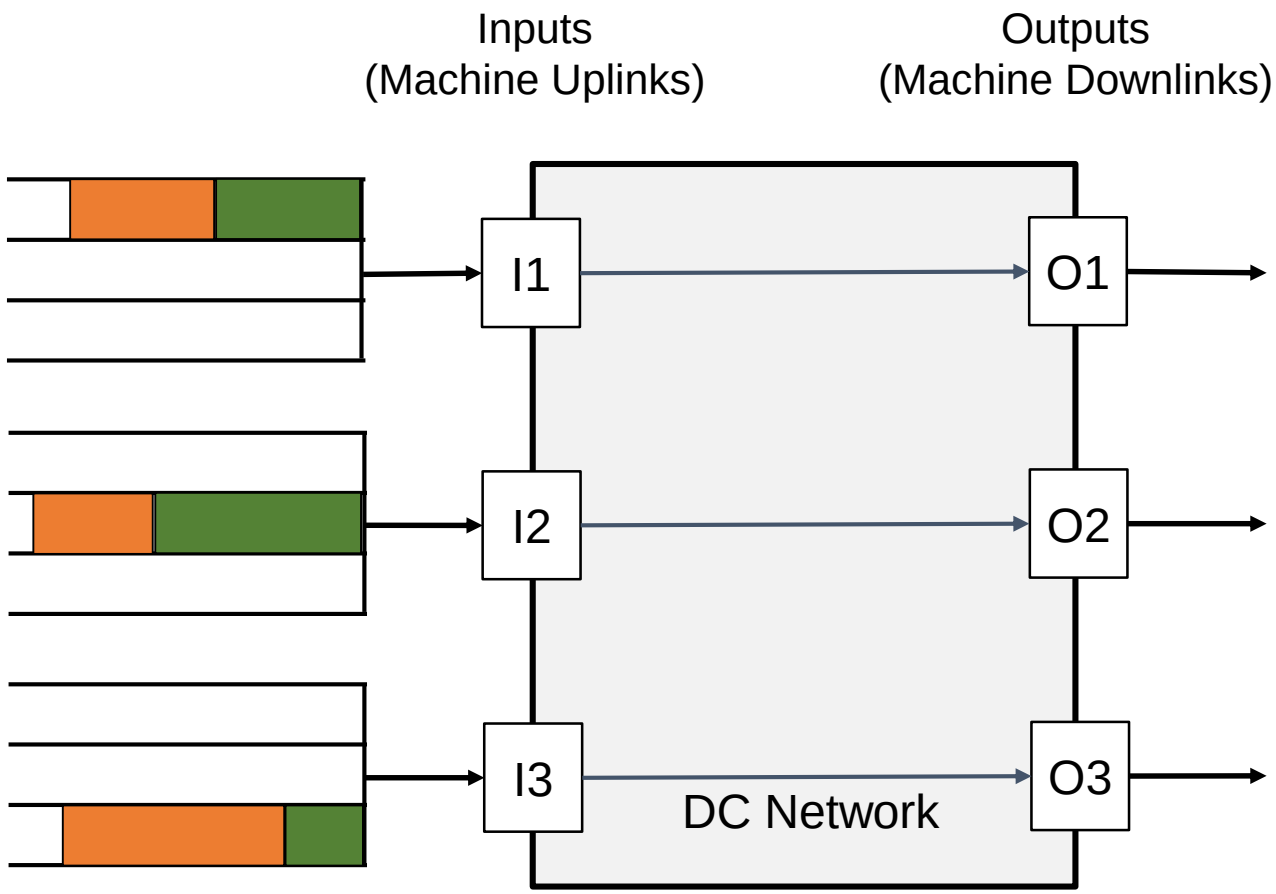
$$y(i,j,k,t) \text{ binary}, \forall i,j,t,k. \qquad (4)$$

- $y(i,j,k,t)$ decides whether $d_{ij}^{(k)}$ of coflow k gets processed at time t

Related Works

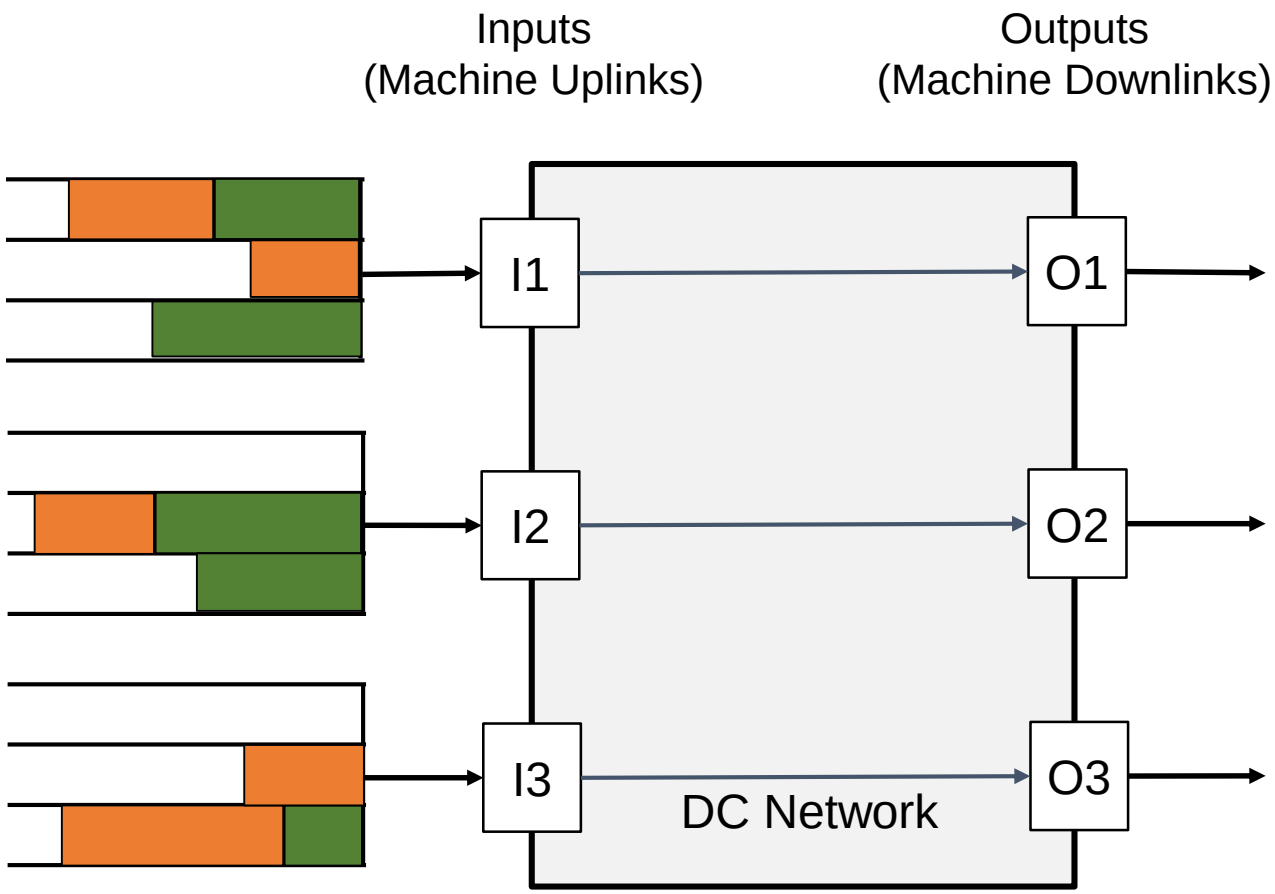
- Systems
  - [Chowdhury & Stoica ’12], [Chowdhury et al ’13], [Chowdhury, Z. & Stoica ’14]
  - Introduced the coflow abstraction and scheduling problem
  - Provided effective scheduling heuristics
  - Showed coflow scheduling is NP-hard – relating to concurrent open shop
- Theory
  - Combinatorial scheduling: e.g., [Hall et al ’97], [Philips et al ’98], [Pinedo ’98], [Skutella ’06], etc
  - Concurrent open shop scheduling: [Ahmedi et al ’05], [Chen & Hall ’07], [Garg et al ’07], [Sung and Yoon ’98], [Leung et al ’07], [Mastrolilli et al 2010], [Wang & Cheng ’07]

Coflow Scheduling is NP-Hard



- This is concurrent open shop scheduling: NP-hard!

Coflow Scheduling: Some Heuristics



- Some natural heuristics
- Order according to $\sum_{i,j} d_{ij}$
- Order according to $\max_{i,j} d_{ij}$
- Order according to load (min finishing time if scheduled alone) – will return to this
- Could only show super-constant approx. ratio

Main Result

Theorem 1 (Qiu, Stein and Z. ’15)

There exists a deterministic polynomial time 64/3 – approximation algorithm A for the coflow scheduling problem, i.e.,

$$\frac{\sum_k w_k C_k(A)}{\sum_k w_k C_k(OPT)} \leq \frac{64}{3}.$$

- Proof overview
  - Step 1: LP relaxation to solve for ordering ($\bar{C}_1 \leq \bar{C}_2 \leq \cdots \leq \bar{C}_n$)
  - Step 2: Schedule algorithm respecting the ordering

Step 1: LP Relaxation

$$\text{Minimize} \sum_{k=1}^{n} w_k C_k \qquad \text{subject to}$$

$$\sum_{t=1}^{C_k} y(i,j,k,t) \geq d_{ij}^{(k)}, \text{ for } i,j = 1,\ldots,m, k = 1,\ldots,n; \quad \text{(load)} \qquad (1)$$

$$\sum_{k=1}^{n}\sum_{j'=1}^{m} y(i,j',k,t) \leq 1, \text{ for } i = 1,\ldots,m, \forall t; \qquad \text{(matching)} \qquad (2)$$

$$\sum_{k=1}^{n}\sum_{i'=1}^{m} y(i',j,k,t) \leq 1, \text{ for } j = 1,\ldots,m, \forall t; \qquad \text{(matching)} \qquad (3)$$

$$y(i,j,k,t) \text{ binary}, \forall i,j,t,k. \qquad (4)$$

- Original formulation too difficult: relax!
- First, upper bound of duration for any reasonable algorithm: $T = \sum_k \sum_{i,j} d_{ij}^{(k)}$
- Second, introduce $z_t^{(k)} = 1$ if coflow k finishes at time t; $= 0$ otherwise

  $\sum_{t=1}^{T} z_t^{(k)} = 1$; $\mathrm{C_k} = \sum_{t=1}^{T} t \cdot z_t^{(k)}$
- Third, relax matching constraints into load constraints

Step 1: LP Relaxation

$$\text{Minimize} \sum_{k=1}^{n} w_k \sum_{t=1}^{T} t z_t^{(k)} \qquad \text{subject to}$$

$$\sum_{u=1}^{t} \sum_{k=1}^{n} \sum_{j'=1}^{m} d_{ij'}^{(k)} z_u^{(k)} \leq t, \text{ for } i = 1, \ldots, m, t = 1, \ldots, T; \quad \text{(load)} \qquad (5)$$

$$\sum_{u=1}^{t} \sum_{k=1}^{n} \sum_{i'=1}^{m} d_{i'j}^{(k)} z_u^{(k)} \leq t, \text{ for } j = 1, \ldots, m, t = 1, \ldots, T; \quad \text{(load)} \qquad (6)$$

$$\sum_{t=1}^{T} z_t^{(k)} = 1, \text{ for } k = 1, \ldots, n;$$

$$z_t^{(k)} \geq 0, \text{ for } k = 1, \ldots, n, \ t = 1, \ldots, T.$$

- Constraint (5): for input i, workload of coflows that complete by time t $\leq$ t
- Formulation is exponentially large; need to solve in poly time

Step 1: LP Relaxation

- Divide time horizon into geometrically increasing time intervals
  - $[0,1],(1,2],(2,4],(4,8],\ldots,(2^{L-2},2^{L-1}]$ with $2^{L-1} \geq T$
  - Time points: $\tau_0 = 0, \tau_1 = 1, \ldots, \tau_L = 2^{L-1}$
- Replace $z_t^{(k)}$ by $x_l^{(k)}, l = 1, 2, \ldots, L-1$

$$\text{Minimize} \sum_{k=1}^{n} w_k \sum_{l=1}^{L} \tau_{l-1} x_l^{(k)} \qquad \text{subject to}$$

$$\sum_{u=1}^{l} \sum_{k=1}^{n} \sum_{j'=1}^{m} d_{ij'}^{(k)} x_u^{(k)} \leq \tau_l, \text{ for } i = 1, \ldots, m, l = 1, \ldots, L; \quad \text{(load)} \qquad (7)$$

$$\sum_{u=1}^{l} \sum_{k=1}^{n} \sum_{i'=1}^{m} d_{i'j}^{(k)} x_u^{(k)} \leq \tau_l, \text{ for } j = 1, \ldots, m, l = 1, \ldots, L; \quad \text{(load)} \qquad (8)$$

$$\sum_{l=1}^{L} x_l^{(k)} = 1, \text{ for } k = 1, \ldots, n;$$

$$x_l^{(k)} \geq 0, \text{ for } k = 1, \ldots, n, \ l = 1, \ldots, L.$$

Step 1: LP Relaxation

- Solve for LP relaxation to get $\bar{x}_l^{(k)}$
- Approximated completion time: $\bar{C}_k = \sum_l \tau_{l-1} \bar{x}_l^{(k)}$

Coflow ordering

$$\bar{C}_1 \leq \bar{C}_2 \leq \cdots \leq \bar{C}_n$$

- Now design scheduling algorithm (call it A) that respect this order
- Relate $C_k(A)$ to $C_k(OPT)$
- Define V_k to relate them: $C_k(A) \leq 4V_k \leq \frac{64}{3} C_k(OPT)$

Step 1: LP Relaxation

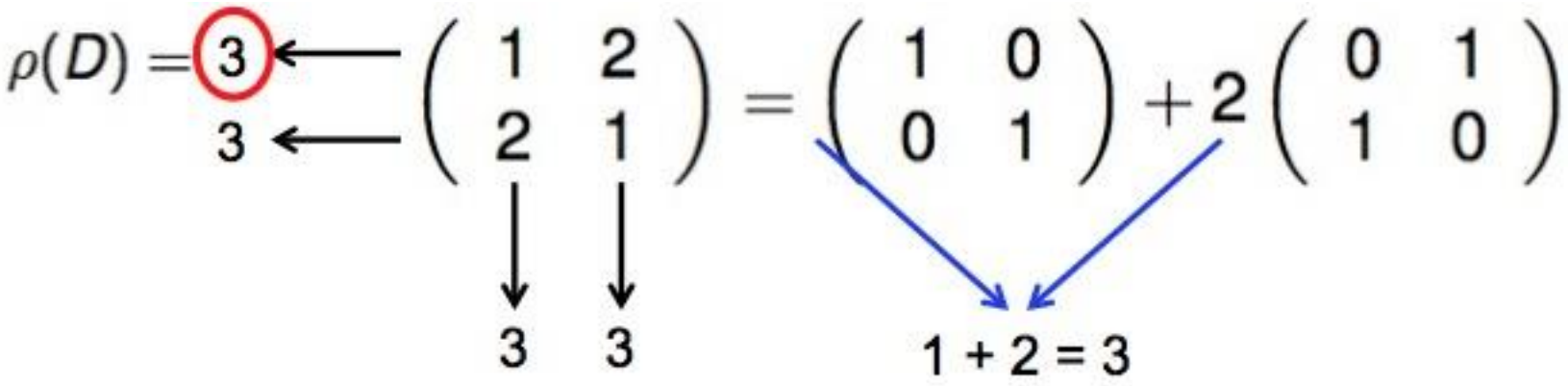
- V_k is the workload of the first k coflows
- Load of coflow g on input i is $\sum_{j'} d_{ij'}^{(g)}$
- Total load of coflows $1, 2, \ldots, k$ on input i: $I_i^{(k)} = \sum_{g=1}^{k} \sum_{j'} d_{ij'}^{(g)}$
- Can define $J_j^{(k)}$ for each output j
- $V_k = \max\left\{I_1^{(k)}, I_2^{(k)}, \ldots, I_m^{(k)}; J_1^{(k)}, J_2^{(k)}, \ldots, J_m^{(k)}\right\}$: lower bound on the time to finish processing first k coflows
- Standard argument to show $V_k \leq \frac{16}{3} C_k(OPT)$ [Wang & Cheng '07] (Left with $C_k(A) \leq 4V_k$)

Step 2: Scheduling Algorithm

- **Fact**: an integer version of the Birkhoff-von Neumann theorem

Birkhoff-von Neumann Decomposition (BvN)
Let $D = (d_{ij})$ be a non-negative, integer-valued matrix. Let $\rho(D)$ be the max row/column sum of D. Then, D can be decomposed (in poly time) as a sum of $\rho(D)$ permutation matrices.

- E.g.,

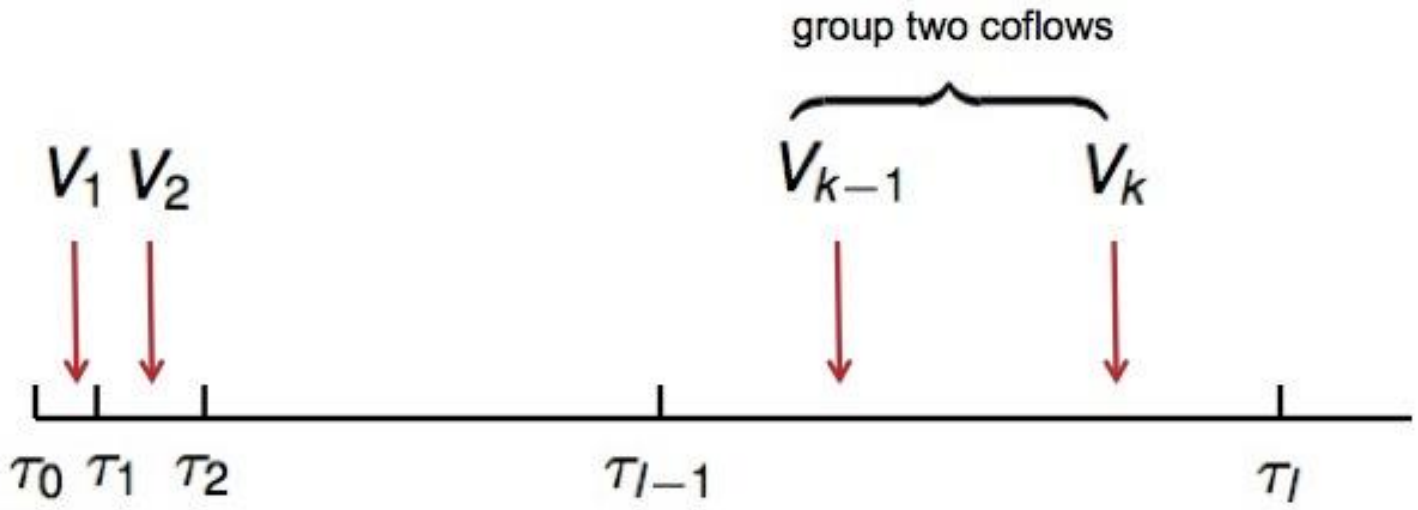


Step 2: Scheduling Algorithm

- **Recall**: coflow order $\bar{C}_1 \leq \bar{C}_2 \leq \cdots \leq \bar{C}_n$

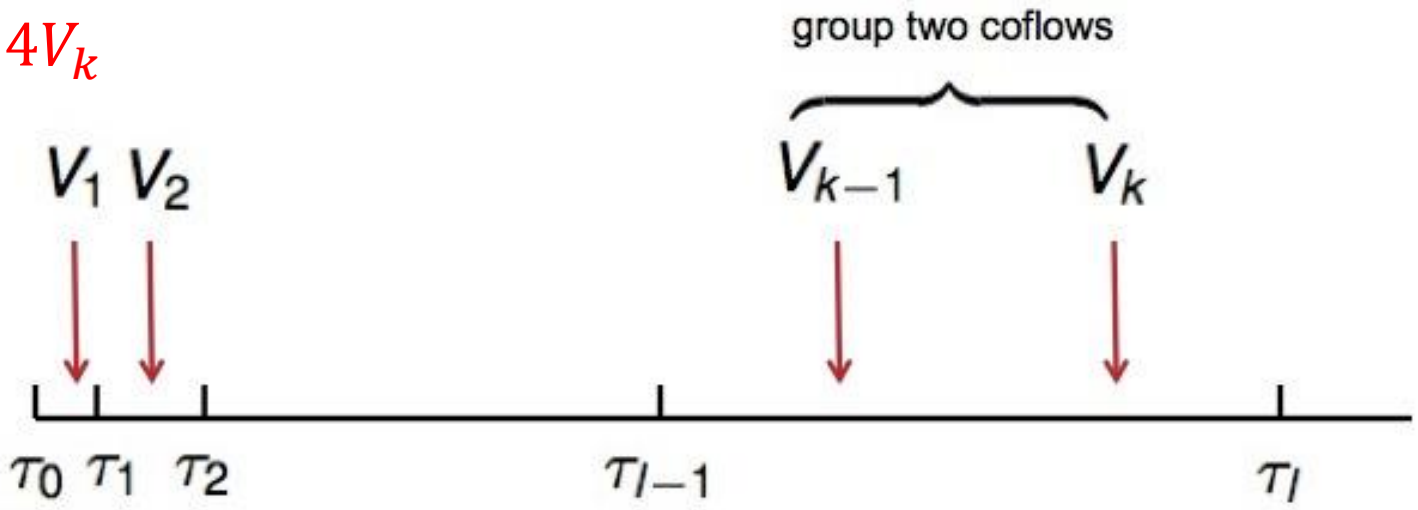
- Group/add coflows according to whether corresponding $V_k \in (\tau_{l-1}, \tau_l]$
  - E.g., add up $D^{(k-1)}$ and $D^{(k)}$ in example; set $M_l = D^{(k-1)} + D^{(k)}$

- Process in the order $M_1, M_2, \ldots, M_L$ using BvN for each M_l



Step 2: Scheduling Algorithm

- Coflow k, suppose $V_k \in (\tau_{l-1}, \tau_l]$
- Time to finish M_l alone $\leq V_{k(last)} \leq \tau_l$
- Time to finish $M_l \leq \tau_1 + \tau_2 + \cdots + \tau_l \leq 2\tau_l = 4\tau_{l-1} \leq 4V_k$
- I.e., $C_k(A) \leq 4V_k$



System and Experiments

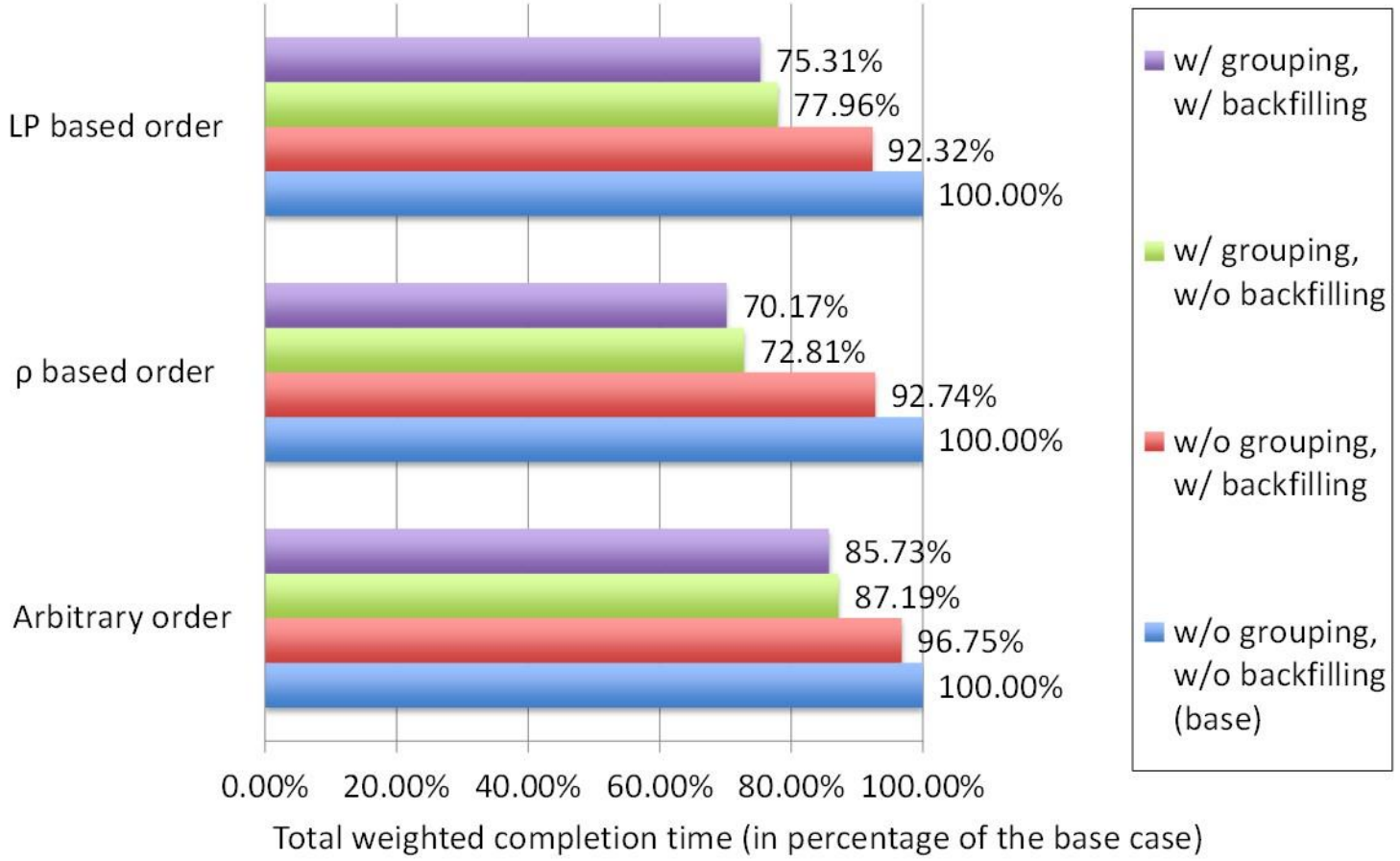
System

- LP-based approx. algo runs in poly time but still complex to implement
- A useful heuristic: order coflows according to $\rho(D)$ (=max row/column sum)
- Online extension: recompute $\rho(D)$ and reorder if new coflows arrive
- Open source code at http://varys.net/; still early stage in development
- Central scheduler
  - Communication messages may be an overhead
  - Apply only to coflows of size $\geq$ 25MB

Experiments

- **Goal**: test performance of various heuristic/algo on real data
- **Data**: a Mapreduce trace at Facebook collected on a 3000-machine cluster with 150 racks (network is 150×150)
- Takeaways of LP-based approx. algo: ordering + grouping + decomposition
- Backfilling also often used in practice
- Will test effect of ordering, grouping and backfilling

Experiments



- Grouping helps
- Backfilling helps

Experiments

- LP-based ordering ≈ ρ based ordering
- LP-based ordering + grouping + backfilling has near optimal performance

Discussion

Discussion

- First O(1)-approx. algo for coflow scheduling

- Extensions:
  - 67/3 approx. ratio with release dates
  - Improved approx. ratio when using random intervals
  - General capacity constraints

- Many open directions
  - Improve approx. ratio
  - Prove tight hardness result
  - *Flow time* minimization
  - Online algorithm
  - Uncertain/unknown coflow matrix
  - Distributed scheduling?