INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

ICPP/2020/EDMONTON/CANADA

AUGUST 17-20, 2020

# Selective Coflow Completion for Time-sensitive Distributed Applications with Poco

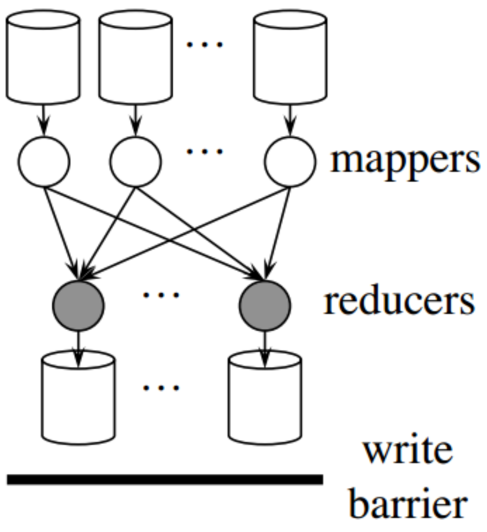
**Shouxi Luo**

Joint work with Pingzhi Fan, Huanlai Xing, and Hongfang Yu

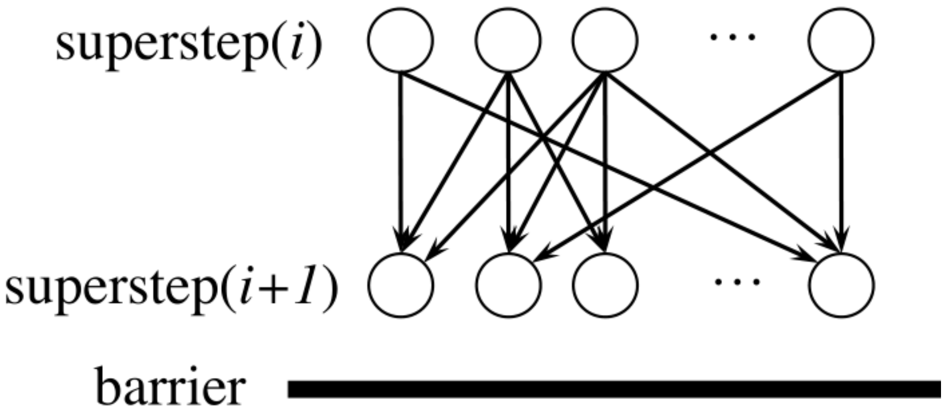
# Outline

- Coflow patterns in DCN
- Existing solutions
- Two trade-offs
- Poco: key designs, service model, and parallelized solver
- Evaluation
- Summary

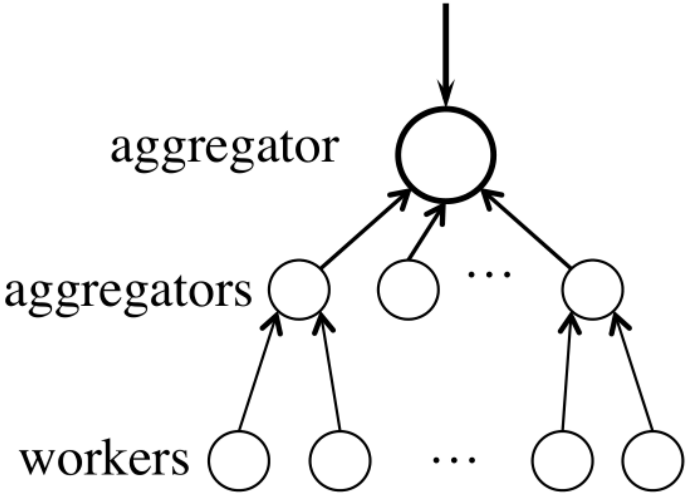
# Coflow patterns in DCN




Map-reduce




Bulk Synchronous Parallel (BSP)




Partition-aggregate

“Each coflow is a collection of flows between two groups of machines with associated semantics.”

Source: HotNets (2012) - Coflow: A networking abstraction for cluster applications

# Coflow patterns in DCN

In many cases, coflows are bounded with deadlines

1. SLA-requirements
2. Time-slotted fair-sharing for concurrent jobs.
3. …

The problem/design goal:
How to let more coflows meet their deadlines?

# Existing solutions

# Existing solutions

- Meeting hard deadlines with admission control
  - Varys[1]

[1] SIGCOMM (2014) - Efficient Coflow Scheduling with Varys

# Existing solutions

- Meeting hard deadlines with admission control
  - Varys[1]
- Deal with soft deadlines with preemptive, prioritized scheduling
  - D2CAS[2]

[1] SIGCOMM (2014) - Efficient Coflow Scheduling with Varys
[2] IEEE ICC (2016) - Decentralized Deadline-Aware Coflow Scheduling for Datacenter Networks

# Existing solutions

- Meeting hard deadlines with admission control
  - Varys[1]
- Deal with soft deadlines with preemptive, prioritized scheduling
  - D2CAS[2]

Limits: overlooking the fact that, many distributed applications can tolerate incomplete data delivery by design

# Existing solutions

- Meeting hard deadlines with admission control
  - Varys[1]
- Deal with soft deadlines with preemptive, prioritized scheduling
  - D2CAS[2]

Limits: overlooking the fact that, many distributed applications can tolerate incomplete data delivery by design

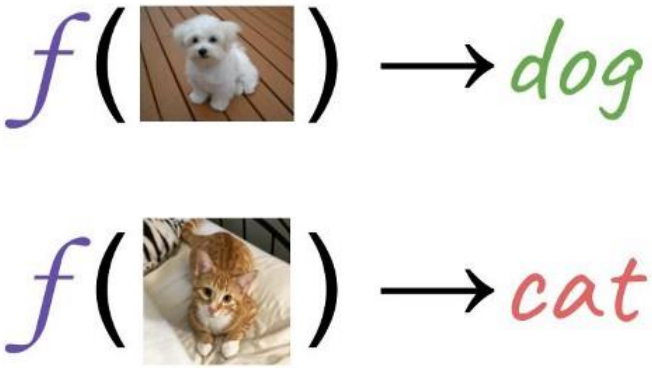

[Source]

# Existing solutions

- Meeting hard deadlines with admission control
  - Varys[1]
- Deal with soft deadlines with preemptive, prioritized scheduling
  - D2CAS[2]

Limits: overlooking the fact that, many distributed applications can tolerate incomplete data delivery by design


Source



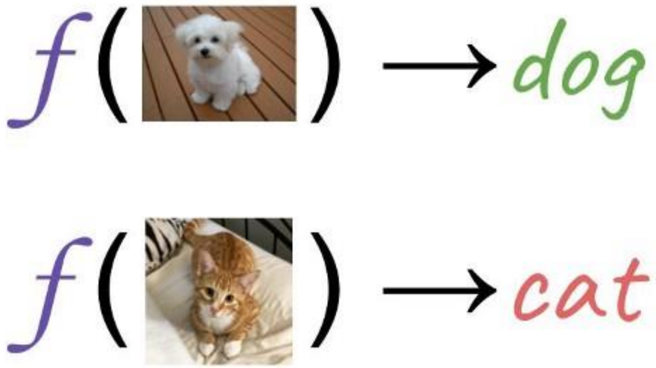

Source

# Existing solutions

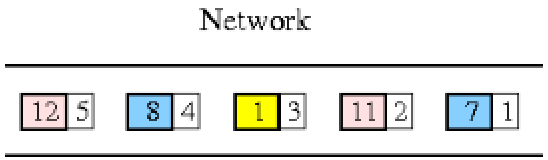
- Meeting hard deadlines with admission control
  - Varys[1]
- Deal with soft deadlines with preemptive, prioritized scheduling
  - D2CAS[2]

**Limits: overlooking the fact that, many distributed applications can tolerate incomplete data delivery by design**


Source




Source




With erasure code

# Existing solutions

- Meeting hard deadlines with admission control
  - Varys[1]
- Deal with soft deadlines with preemptive, prioritized scheduling
  - D2CAS[2]
- Maximize the marginal partial throughput to explore the tolerance of partial transmission
  - Con-myopic[3]

[1] SIGCOMM (2014) - Efficient Coflow Scheduling with Varys
[2] IEEE ICC (2016) - Decentralized Deadline-Aware Coflow Scheduling for Datacenter Networks
[3] IEEE Infocom (2018) - Online Partial Throughput Maximization for Multidimensional Coflow

# Existing solutions

- Meeting hard deadlines with admission control
  - Varys[1]
- Deal with soft deadlines with preemptive, prioritized scheduling
  - D2CAS[2]
- Maximize the marginal partial throughput to explore the tolerance of partial transmission
  - Con-myopic[3]

Limits: inflexible, no performance guarantee

[1] SIGCOMM (2014) - Efficient Coflow Scheduling with Varys
[2] IEEE ICC (2016) - Decentralized Deadline-Aware Coflow Scheduling for Datacenter Networks
[3] IEEE Infocom (2018) - Online Partial Throughput Maximization for Multidimensional Coflow

# Two trade-offs

$$v_f = \int_0^t r_f(t)dt$$

# Two trade-offs

$$v_f = \int_0^t r_f(t)dt$$

$$r_f(t) \leq c_e - \sum_{f' \in \mathcal{F}_e \setminus \{f\}} r_{f'}(t)$$

# Two trade-offs

$$v_f = \int_0^t r_f(t)dt$$

$$r_f(t) \leq c_e - \sum_{f' \in \mathcal{F}_e \setminus \{f\}} r_{f'}(t)$$

#1 Timeliness ⟷ completeness

#2 The completeness of (co)flow A ⟷ that of (co)flow B

# Poco: a POlicy-based COflow scheduler

# Poco: key designs

Two key designs

# Poco: key designs

Two key designs

1. Enable applications to specify coflow requirements explicitly.
   - ✓Timeliness/deadlines
   - ✓Completeness/level of tolerance

**Grammar**

| | | | |
|---|---|---|---|
| $C_i$ | $::=$ | $(\mathcal{F}_i; \mathcal{R}_i)$ | Application-specified coflow request |
| $\mathcal{F}_i$ | $::=$ | $\{\cdots, f_{i,j}, \cdots\}$ | Transfer demands of cofow $C_i$ |
| $\mathcal{R}_i$ | $::=$ | $\{\cdots, (G_{i,k}; \phi_{i,k}), \cdots\}$ | Completeness requirements |
| $f_{i,j}$ | $::=$ | $(\tau_{i,j}; v_{i,j}; p_{i,j})$ | Details of the $j$-th subflow in coflow $i$ |

**More Notation**

| | | |
|---|---|---|
| $\tau_{i,j}$ | : | Expired time of flow $f_{i,j}$ (we have $\forall j : \tau_{i,j} = \tau_i$ in this paper) |
| $v_{i,j}$ | : | Remaining volume of flow $f_{i,j}$ |
| $p_{i,j}$ | : | Path of flow $f_{i,j}$ |
| $G_{i,k}$ | : | Set of flow(s) in the same completeness group |
| $\phi_{i,k}$ | : | Completeness requirement |

# Poco: key designs

Two key designs

1. Enable applications to specify coflow requirements explicitly.
   ✓Timeliness/deadlines
   ✓Completeness/level of tolerance

2. Explore the trade-offs explicitly with a monolithic (time-slotted) Linear Program model.
   ✓Requirements → **linear constraints**

**Grammar**

| | | | |
|---|---|---|---|
| $C_i$ | ::= | $(\mathcal{F}_i; \mathcal{R}_i)$ | Application-specified coflow request |
| $\mathcal{F}_i$ | ::= | $\{\cdots, f_{i,j}, \cdots\}$ | Transfer demands of cofow $C_i$ |
| $\mathcal{R}_i$ | ::= | $\{\cdots, (G_{i,k}; \phi_{i,k}), \cdots\}$ | Completeness requirements |
| $f_{i,j}$ | ::= | $(\tau_{i,j}; v_{i,j}; p_{i,j})$ | Details of the $j$-th subflow in coflow $i$ |

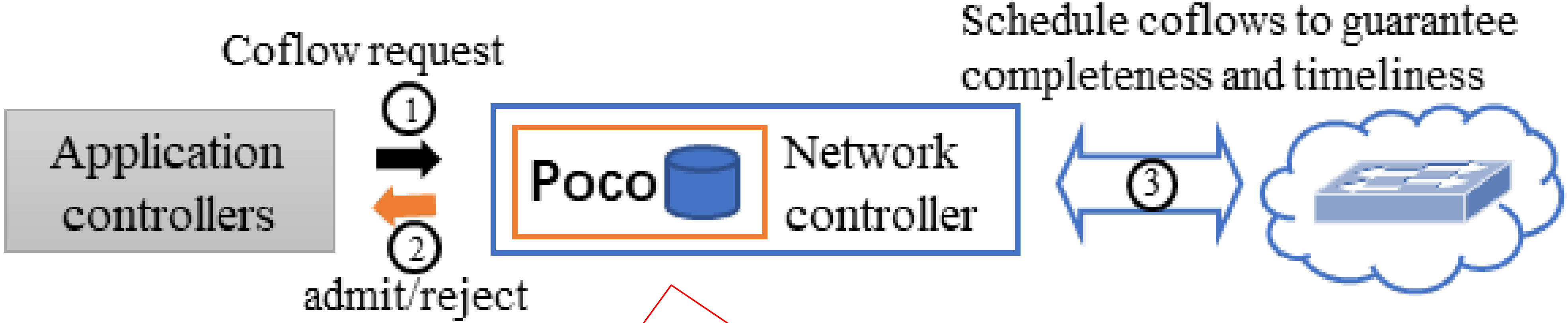
**More Notation**

| | | |
|---|---|---|
| $\tau_{i,j}$ | : | Expired time of flow $f_{i,j}$ (we have $\forall j : \tau_{i,j} = \tau_i$ in this paper) |
| $v_{i,j}$ | : | Remaining volume of flow $f_{i,j}$ |
| $p_{i,j}$ | : | Path of flow $f_{i,j}$ |
| $G_{i,k}$ | : | Set of flow(s) in the same completeness group |
| $\phi_{i,k}$ | : | Completeness requirement |

$$\text{Maximize} \sum_{i=1}^{n} \sum_{j=1}^{|F_i|} \sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \quad s.t.\ (\boxed{4})$$

$$(\boxed{4}) \begin{cases} \sum_{(i,j) \in G_{i,k}} \sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \geq \phi_{i,k}, \quad \forall i, k & \text{(4a)} \\ \sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \leq v_{i,j}, \quad \forall i, j & \text{(4b)} \\ \sum_{(i,j): e \in p_{i,j}} r_{i,j,t} \leq c_{e,t}, \quad \forall e, t & \text{(4c)} \\ r_{i,j,t} \geq 0, \quad \forall i, j, t & \text{(4d)} \end{cases}$$

# Poco: service model




**Solve the involved LP** (4)

$$\text{Maximize} \sum_{i=1}^{n}\sum_{j=1}^{|F_i|}\sum_{t=1}^{\tau_{i,j}} r_{i,j,t}\Delta_T \quad s.t. \ (4)$$

$$(4)\begin{cases} \sum_{(i,j)\in G_{i,k}} \sum_{t=1}^{\tau_{i,j}} r_{i,j,t}\Delta_T \geq \phi_{i,k}, \quad \forall i,k & \text{(4a)} \\ \sum_{t=1}^{\tau_{i,j}} r_{i,j,t}\Delta_T \leq v_{i,j}, \quad \forall i,j & \text{(4b)} \\ \sum_{(i,j):e\in p_{i,j}} r_{i,j,t} \leq c_{e,t}, \quad \forall e,t & \text{(4c)} \\ r_{i,j,t} \geq 0, \quad \forall i,j,t & \text{(4d)} \end{cases}$$

Provide guaranteed performance with admission control

Challenge:
How to solve large-scale LPs efficiently?

Challenge:
How to solve large-scale LPs efficiently?

Parallelize the computation by leveraging the **specific** structure of the LPs

# Poco: parallelized solver

$$\text{Maximize} \sum_{i=1}^{n} \sum_{i=1}^{|F_i|} \sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \qquad s.t.\ (\boxed{4})$$

$$(\boxed{4}) \begin{cases} \displaystyle\sum_{(i,j) \in G_{i,k}} \sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \geq \phi_{i,k}, \quad \forall i, k & \text{(4a)} \\ \displaystyle\sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \leq v_{i,j}, \quad \forall i, j & \text{(4b)} \\ \displaystyle\sum_{(i,j):e \in p_{i,j}} r_{i,j,t} \leq c_{e,t}, \quad \forall e, t & \text{(4c)} \\ r_{i,j,t} \geq 0, \quad \forall i, j, t & \text{(4d)} \end{cases}$$

# Poco: parallelized solver

$$\text{Maximize} \sum_{i=1}^{n}\sum_{j=1}^{|F_i|}\sum_{t=1}^{\tau_{i,j}} r_{i,j,t}\Delta_T \quad s.t.\ (\boxed{4})$$

$$(\boxed{4}) \begin{cases} \displaystyle\sum_{(i,j)\in G_{i,k}}\sum_{t=1}^{\tau_{i,j}} r_{i,j,t}\Delta_T \geq \phi_{i,k}, \quad \forall i,k & \text{(4a)} \\ \displaystyle\sum_{t=1}^{\tau_{i,j}} r_{i,j,t}\Delta_T \leq v_{i,j}, \quad \forall i,j & \text{(4b)} \\ \displaystyle\sum_{(i,j):e\in p_{i,j}} r_{i,j,t} \leq c_{e,t}, \quad \forall e,t & \text{(4c)} \\ r_{i,j,t} \geq 0, \quad \forall i,j,t & \text{(4d)} \end{cases}$$

$$\Rightarrow$$

$$\text{Minimize} \quad \boldsymbol{w}^{\top}\boldsymbol{x} \quad s.t.\ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

# Poco: parallelized solver

$$\text{Maximize} \sum_{i=1}^{n} \sum_{j=1}^{|F_i|} \sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \quad s.t.\ (\boxed{4})$$

$$(\boxed{4}) \begin{cases} \sum_{(i,j) \in G_{i,k}} \sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \geq \phi_{i,k}, \quad \forall i, k & \text{(4a)} \\ \sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \leq v_{i,j}, \quad \forall i, j & \text{(4b)} \\ \sum_{(i,j):e \in p_{i,j}} r_{i,j,t} \leq c_{e,t}, \quad \forall e, t & \text{(4c)} \\ r_{i,j,t} \geq 0, \quad \forall i, j, t & \text{(4d)} \end{cases}$$

$$\text{Minimize} \quad \boldsymbol{w}^\top \boldsymbol{x} \quad s.t.\ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

The core of interior-point method: solve equations iteratively

$$AD^k A^\top d_y = v$$

# Poco: parallelized solver

$$\text{Maximize} \sum_{i=1}^{n} \sum_{i=1}^{|F_i|} \sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \quad s.t. \ (\boxed{4})$$

$$(\boxed{4}) \begin{cases} \sum_{(i,j) \in G_{i,k}} \sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \geq \phi_{i,k}, \quad \forall i, k & \text{(4a)} \\ \sum_{t=1}^{\tau_{i,j}} r_{i,j,t} \Delta_T \leq v_{i,j}, \quad \forall i, j & \text{(4b)} \\ \sum_{(i,j): e \in p_{i,j}} r_{i,j,t} \leq c_{e,t}, \quad \forall e, t & \text{(4c)} \\ r_{i,j,t} \geq 0, \quad \forall i, j, t & \text{(4d)} \end{cases}$$

$$\text{Minimize} \quad \boldsymbol{w}^{\top} \boldsymbol{x} \quad s.t. \ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

The core of interior-point method: solve equations iteratively

$$\boldsymbol{A}\boldsymbol{D}^k \boldsymbol{A}^{\top} \boldsymbol{d}_y = \boldsymbol{v}$$

Obviously, $\boldsymbol{AD}^k\boldsymbol{A}^T$ is positive-semidefinite, having the Cholesky decomposition of $\boldsymbol{LL}^T$ in most cases.
Accordingly, the original problem can be solved efficiently via $\boldsymbol{Lg} = \boldsymbol{v}$. then $\boldsymbol{L}^T\boldsymbol{d}_y = \boldsymbol{g}$.
In case it is not positive-definite, the equations can be solve with other approximated methods.

# Poco: parallelized solver

**Solution**: parallelize the computation by leveraging the specific structure of the LP

$$\boldsymbol{A x} = \boldsymbol{b}$$

#1 Constraints introduced by the timeliness and completeness requirements of the 1st coflow

$$\boldsymbol{A x} = \begin{bmatrix} \boxed{\boldsymbol{A}_1} & & & & \\ & \boldsymbol{A}_2 & & & \\ & & \ddots & & \\ & & & \boldsymbol{A}_n & \\ \boldsymbol{B}_1 & \boldsymbol{B}_2 & \cdots & \boldsymbol{B}_n & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \\ \boldsymbol{x}^s \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_n \\ \boldsymbol{b}^* \end{bmatrix}$$

# Poco: parallelized solver

**Solution**: parallelize the computation by leveraging the specific structure of the LP

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

$$\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{A}_1 & & & & \\ & \boldsymbol{A}_2 & & & \\ & & \ddots & & \\ & & & \boldsymbol{A}_n & \\ \boxed{\boldsymbol{B}_1} & \boldsymbol{B}_2 & \cdots & \boldsymbol{B}_n & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \\ \boldsymbol{x}^s \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_n \\ \boldsymbol{b}^* \end{bmatrix}$$

#2 Constraints of link capacities involved in the 1st coflow.

# Poco: parallelized solver

Constraints introduced by the 1st subflow's total volume

$$
\boldsymbol{A}_i := \left[\begin{array}{cc|c|c|c|c}
\boldsymbol{a}_{i,1} & 1 & & & & \\
\hline
 & & \boldsymbol{a}_{i,2}\ 1 & & & \\
\hline
 & & & \ddots & & \\
\hline
 & & & & \boldsymbol{a}_{i,|F_i|}\ \ 1 & \\
\hline
\psi_{i,1,1}\boldsymbol{a}_{i,1} & 0 & \cdots & \cdots & \psi_{i,1,|\mathcal{F}_i|}\boldsymbol{a}_{i,1,|F_i|}\ \ 0 & -1 \\
\vdots & & \vdots & \vdots & \vdots\ \ \vdots & \ddots \\
\psi_{i,|R_i|,1}\boldsymbol{a}_{i,1} & 0 & \cdots & \cdots & \psi_{i,|R_i|,|\mathcal{F}_i|}\boldsymbol{a}_{i,|\mathcal{F}_i|}\ \ 0 & -1
\end{array}\right]
$$

Constraints introduced by the 1st completeness requirements

$$
\boldsymbol{a}_{i,j} := [\Delta_T^{(1)}, \cdots, \Delta_T^{(\pi_{i,j})}]
$$

$$
\psi_{i,k,j} := \begin{cases} 1 & (i,j) \in G_{i,k} \\ 0 & otherwise \end{cases}
$$

Subflow (i,j) is involved in the *k*-th completeness requirement

$$
\boldsymbol{B}_i := \left[\begin{array}{c|cccc|c|c}
\cdots & \boldsymbol{\hbar}_{1,i,j,1} & \cdots & \boldsymbol{\hbar}_{1,i,j,\pi_{i,j}} & 0 & \cdots & \boldsymbol{0} \\
\ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\cdots & \boldsymbol{\hbar}_{|\Gamma|,i,j,1} & \cdots & \boldsymbol{\hbar}_{|\Gamma|,i,j,\pi_{i,j}} & 0 & \cdots & \boldsymbol{0}
\end{array}\right]_{j=1,\cdots,|F_i|}
$$

$$
\hbar_{o,i,j,l} := \begin{cases} 1 & \kappa_o^e \in p_{i,j} \wedge l \le \pi_{i,j} \\ 0 & otherwise \end{cases}
$$

Subflow $(i,j)$ goes through the $o$-th link and is active during the $l$-th time slot/range

# Poco: parallelized solver

Constraints introduced by the 1st subflow's total volume

$$A_i := \left[\begin{array}{cc|c|c|cc|ccc}
a_{i,1} & 1 & & & & & & & \\ \hline
& & a_{i,2}\ 1 & & & & & & \\ \hline
& & & \ddots & & & & & \\ \hline
& & & & a_{i,|F_i|} & 1 & & & \\ \hline
\psi_{i,1,1} a_{i,1} & 0 & \cdots & \cdots & \psi_{i,1,|\mathcal{F}_i|} a_{i,1,|F_i|} & 0 & -1 & & \\
\vdots & & \vdots & \vdots & \vdots & \vdots & & \ddots & \\
\psi_{i,|R_i|,1} a_{i,1} & 0 & \cdots & \cdots & \psi_{i,|R_i|,|\mathcal{F}_i|} a_{i,|\mathcal{F}_i|} & 0 & & & -1
\end{array}\right]$$

Constraints introduced by the 1st completeness requirements

$$a_{i,j} := [\Delta_T^{(1)}, \cdots, \Delta_T^{(\pi_{i,j})}]$$

$$\psi_{i,k,j} := \begin{cases} 1 & (i,j) \in G_{i,k} \\ 0 & otherwise \end{cases}$$

Subflow (i,j) is involved in the *k*-th completeness requirement

$$B_i := \left[\begin{array}{c|cccc|c|c}
\cdots & \hbar_{1,i,j,1} & \cdots & \hbar_{1,i,j,\pi_{i,j}} & 0 & \cdots & \mathbf{0} \\
\ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\cdots & \hbar_{|\Gamma|,i,j,1} & \cdots & \hbar_{|\Gamma|,i,j,\pi_{i,j}} & 0 & \cdots & \mathbf{0}
\end{array}\right]_{j=1,\cdots,|F_i|}$$

$$\hbar_{o,i,j,l} := \begin{cases} 1 & \kappa_o^e \in p_{i,j} \wedge l \le \pi_{i,j} \\ 0 & otherwise \end{cases}$$

Subflow $(i, j)$ goes through the $o$-th link and is active during the $l$-th time slot/range

# Poco: parallelized solver

$$D^k = diag(\frac{x_1^k}{s_1^k}, \frac{x_2^k}{s_2^k}, \cdots)$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & & & & \\ & \boldsymbol{A}_2 & & & \\ & & \ddots & & \\ & & & \boldsymbol{A}_n & \\ \boldsymbol{B}_1 & \boldsymbol{B}_2 & \cdots & \boldsymbol{B}_n & \boldsymbol{I} \end{bmatrix}$$

$$\Downarrow$$

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_1 & & & & \\ & \boldsymbol{L}_2 & & & \\ & & \ddots & & \\ & & & \boldsymbol{L}_n & \\ \boldsymbol{M}_1 & \boldsymbol{M}_2 & \cdots & \boldsymbol{M}_n & \boldsymbol{L}_* \end{bmatrix}$$

# Poco: parallelized solver

$$C = \sum_{i=1}^{n} B_i D_i^k B_i^\top + D_{n+1}^k$$

$$D^k = diag(\frac{x_1^k}{s_1^k}, \frac{x_2^k}{s_2^k}, \cdots)$$

$$A = \begin{bmatrix} A_1 & & & & \\ & A_2 & & & \\ & & \ddots & & \\ & & & A_n & \\ B_1 & B_2 & \cdots & B_n & I \end{bmatrix} \Rightarrow AD^kA^T = \begin{bmatrix} A_1 D_1^k A_1^T & & & & A_1 D_1^k B_1^T \\ & A_2 D_2^k A_2^T & & & A_2 D_2^k B_2^T \\ & & \ddots & & \vdots \\ & & & A_n D_n^k A_n^T & A_n^k D_n^k B_n^T \\ B_1 D_1^k A_1^T & B_2 D_2^k A_2^T & \cdots & B_n D_n^k A_n^T & C \end{bmatrix}$$

$$\Downarrow \qquad\qquad\qquad \Vert$$

$$L = \begin{bmatrix} L_1 & & & & \\ & L_2 & & & \\ & & \ddots & & \\ & & & L_n & \\ M_1 & M_2 & \cdots & M_n & L_* \end{bmatrix} \Rightarrow LL^T = \begin{bmatrix} L_1 L_1^T & & & & L_1 M_1^T \\ & L_2 L_2^T & & & L_2 M_2^T \\ & & \ddots & & \vdots \\ & & & L_n L_n^T & L_n^k M_n^T \\ M_1 L_1^T & M_2 L_2^T & \cdots & M_n L_n^T & G \end{bmatrix}$$

$$G = \sum_{i=1}^{n} M_i M_i^\top + L_* L_*^\top$$

# Poco: parallelized solver

$$D^k = diag(\frac{x_1^k}{s_1^k}, \frac{x_2^k}{s_2^k}, \cdots)$$

$$C = \sum_{i=1}^{n} B_i D_i^k B_i^\top + D_{n+1}^k$$

$$A = \begin{bmatrix} A_1 & & & & \\ & A_2 & & & \\ & & \ddots & & \\ & & & A_n & \\ B_1 & B_2 & \cdots & B_n & I \end{bmatrix} \Rightarrow AD^kA^T = \begin{bmatrix} A_1 D_1^k A_1^T & & & & A_1 D_1^k B_1^T \\ & A_2 D_2^k A_2^T & & & A_2 D_2^k B_2^T \\ & & \ddots & & \vdots \\ & & & A_n D_n^k A_n^T & A_n^k D_n^k B_n^T \\ B_1 D_1^k A_1^T & B_2 D_2^k A_2^T & \cdots & B_n D_n^k A_n^T & C \end{bmatrix}$$

$$\Downarrow$$

$$L = \begin{bmatrix} L_1 & & & & \\ & L_2 & & & \\ & & \ddots & & \\ & & & L_n & \\ M_1 & M_2 & \cdots & M_n & L_* \end{bmatrix} \Rightarrow LL^T = \begin{bmatrix} L_1 L_1^T & & & & L_1 M_1^T \\ & L_2 L_2^T & & & L_2 M_2^T \\ & & \ddots & & \vdots \\ & & & L_n L_n^T & L_n^k M_n^T \\ M_1 L_1^T & M_2 L_2^T & \cdots & M_n L_n^T & G \end{bmatrix}$$

$$G = \sum_{i=1}^{n} M_i M_i^\top + L_* L_*^\top$$

Parallel cholesky decomposition

**C1:** Calc $L_i$ via $L_i L_i^T = A_i D_i^k A_i^T$

i=1,2,…,n

**C2:** Calc $L_i^{-1}$

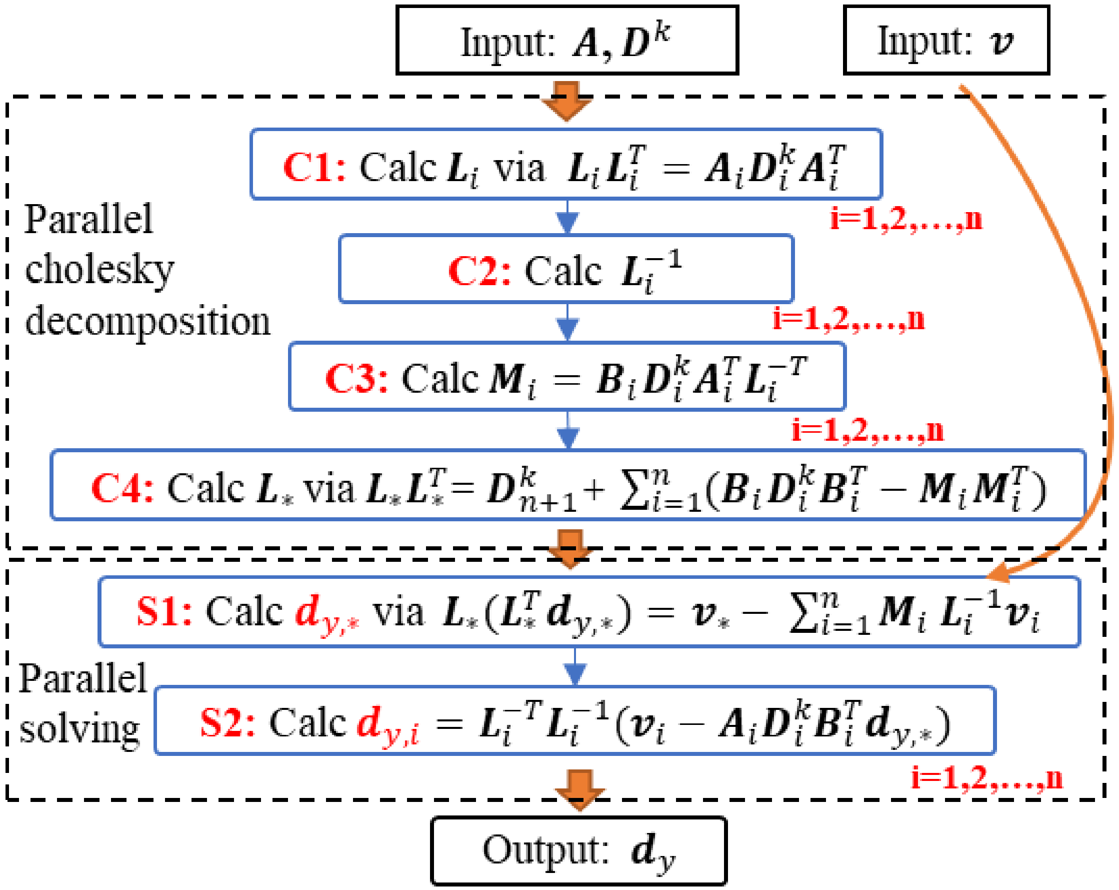
i=1,2,…,n

**C3:** Calc $M_i = B_i D_i^k A_i^T L_i^{-T}$

i=1,2,…,n

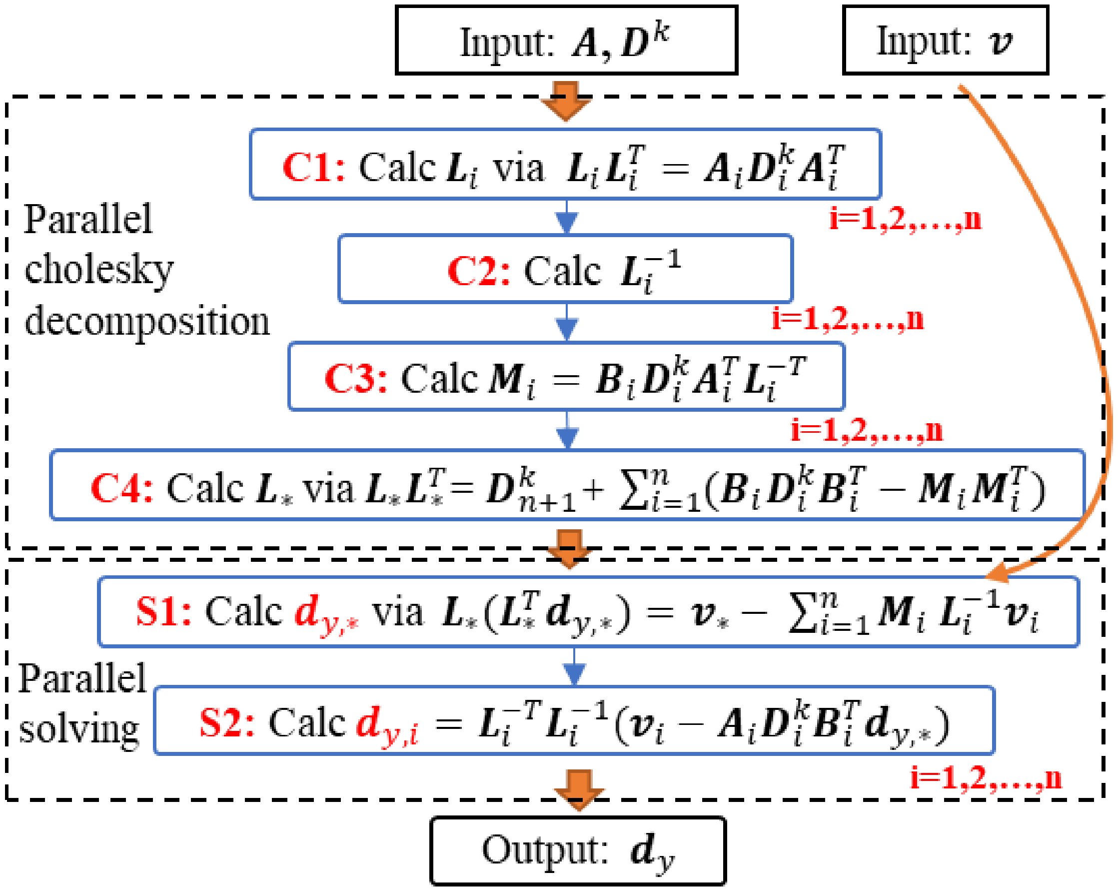
**C4:** Calc $L_*$ via $L_* L_*^T = D_{n+1}^k + \sum_{i=1}^{n}(B_i D_i^k B_i^T - M_i M_i^T)$

# Poco: parallelized solver




Note: in rare cases the involved matrix is not positive-definite, we can solve the associated $d_y$ with approximated methods

# Poco: parallelized solver



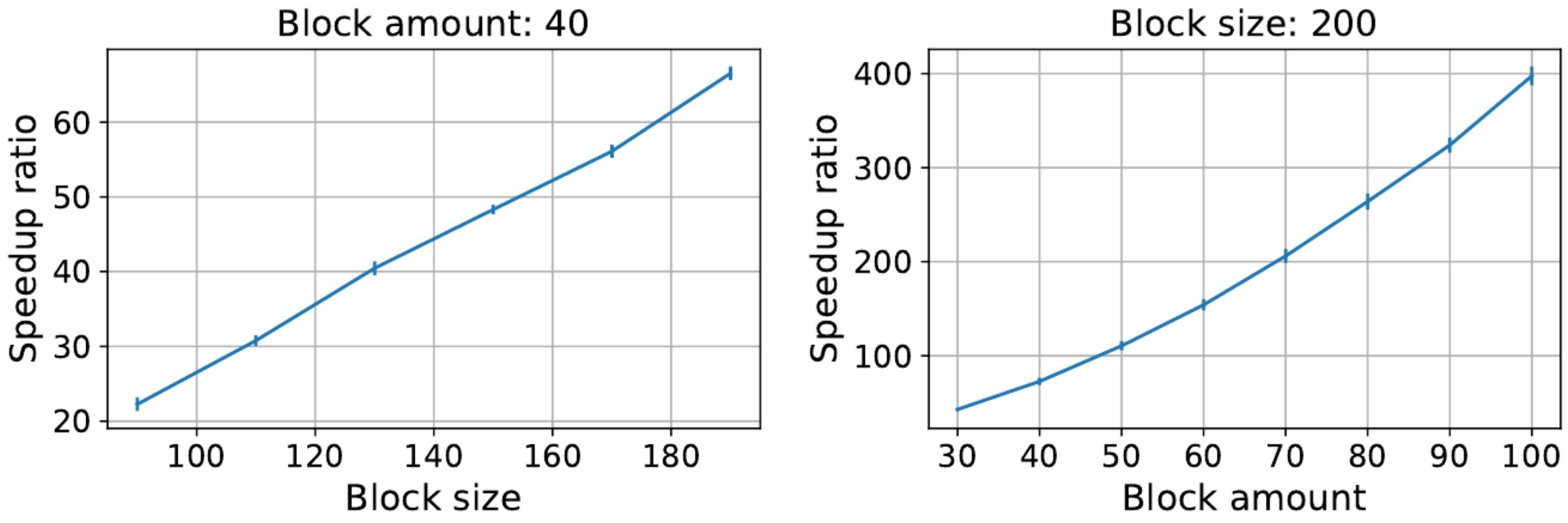

Benefits:

✓Explore the sparsity of ***A*** explicitly

✓Make both Cholesky decompaction and solving parallelized

Note: in rare cases the involved matrix is not positive-definite, we can solve the associated $\boldsymbol{d}_y$ with approximated methods

# Poco: parallelized solver




(a) Under various block size (b) Under various block amount

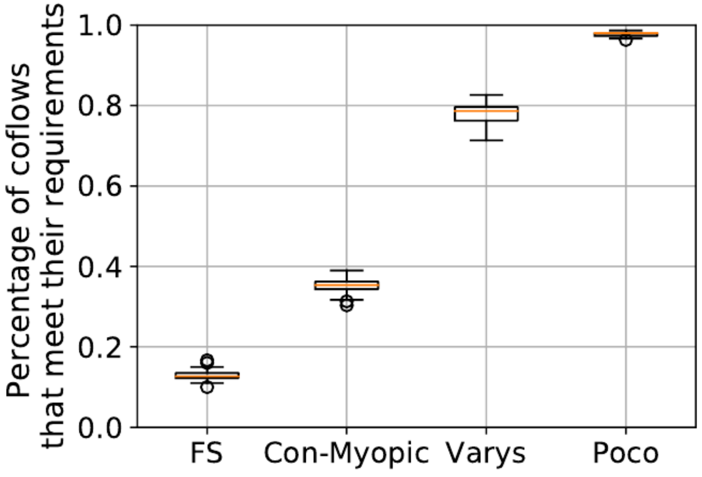
Parallelization speeds up the solving greatly.

- ❖ Naive implementations upon scipy/numpy,
- ❖ Ubuntu 18.04, Intel Xeon(R) Silver 4210 CPU, 16G RAM, Python3

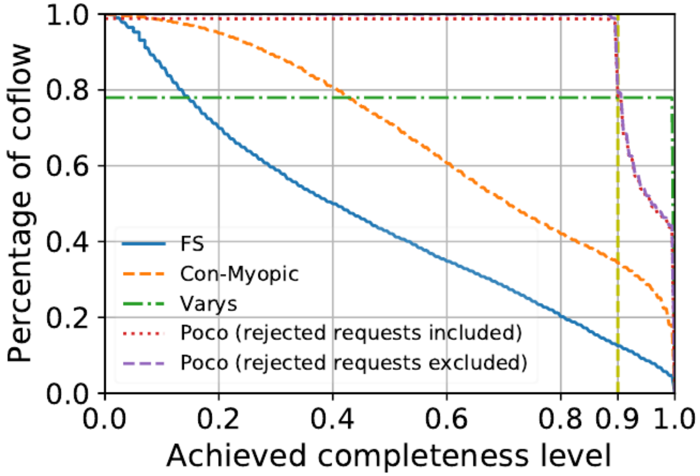
# Evaluation

- Flow-level simulator in Python3
- Inputs
  - Synthesized with Facebook traces
  - Completeness-requirement: 0.9, deadline: 1 + U[1; 2]
- Baselines
  - Con-Myopic
  - FS (per-flow fair-sharing)
  - Varys
- Metrics
  - Percentage of coflows that meet their requirements
  - Achieved completions/delivered data volumes

# Evaluation

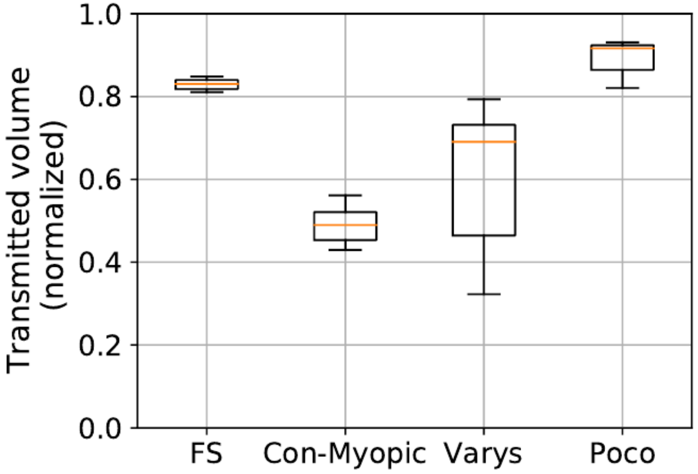



(a) Requirement-satisified coflow

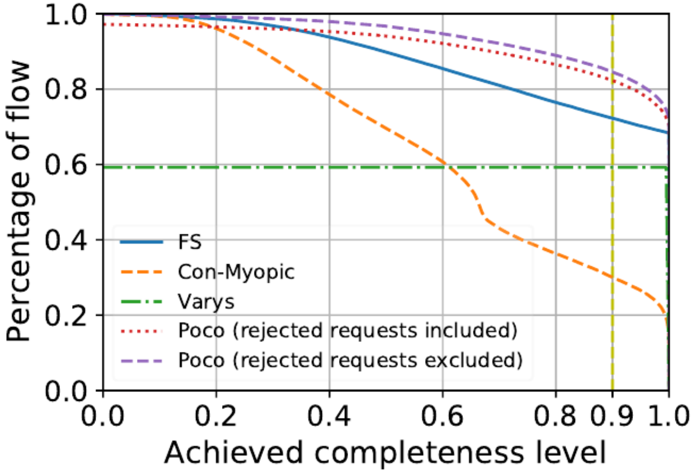



(b) Achieved completeness

Poco outperforms existing solutions greatly.




(c) Transmitted volume




(d) Achieved completeness of flow

Poco is very flexible.

# Summary

## Poco

1. Enables distributed applications to specify their requirements explicitly along with their coflow requests;
2. Explores the trade-offs explicitly with a monolithic (time-slotted) Linear Program (LP) model;
3. Parallelizes the solving of LP using the specific structure of the model.

Refer to the paper for more details

Join our slack discussion: **Parallel Algorithms II (Thursday, August 20$^{th}$, 12:30pm-1:00pm)**

Drop me emails at sxluo[at]swjtu.edu.cn