

# Incentivizing Resource Pooling

Chen Chen



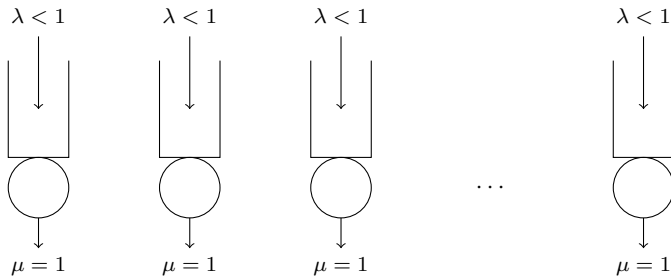
September 2023

*Joint work with:*

Yilun Chen (CUHK-SZ) and Pengyu Qian (Purdue)

# Resource pooling

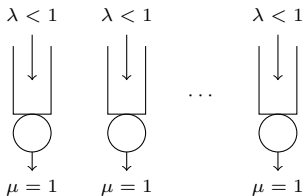
$N$  servers: job arrival rate  $\lambda < 1$ , server processing rate  $\mu = 1$



# Resource pooling

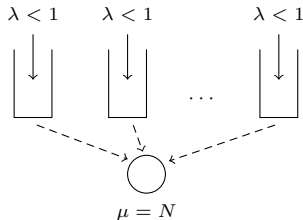
$N$  servers: job arrival rate  $\lambda < 1$ , server processing rate  $\mu = 1$

**Without** resource pooling:



*vs.*

**With** resource pooling:



# jobs in system:  $N \cdot \frac{\lambda}{1-\lambda}$

$\frac{\lambda}{1-\lambda}$

# Motivation

- **Goal:** design mechanism to incentivize resource pooling in a decentralized setting.
- **Applications:** Decentralized computing marketplaces on blockchains



Golem Network  
Market cap: \$170M



Akash Network  
Market cap: \$320M



iExec  
Market cap: \$75M

- Essential aspects of the problem:
  - ▶ Number of servers  $N$  is large.
  - ▶ Servers possess restrictive information about the other servers.

# Motivation

- **Goal:** design mechanism to incentivize resource pooling in a decentralized setting.
- **Applications:** Decentralized computing marketplaces on blockchains



Golem Network  
Market cap: \$170M



Akash Network  
Market cap: \$320M

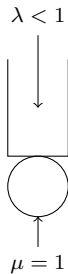


iExec  
Market cap: \$75M

- Essential aspects of the problem:
  - ▶ Number of servers  $N$  is large.
  - ▶ Servers possess restrictive information about the other servers.
- **Main result:** develop a simple **token-based mechanism** that incentivizes **complete resource pooling** in limited information setting when  $N$  is large.  
⇒ System dynamics and performance match those under centralized control in the asymptotics

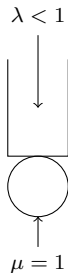
# Model setup

- $N$  strategic servers
- Jobs arrive with  $\text{Poisson}(\lambda)$  where  $\lambda < 1$ ; capacity units arrive with  $\text{Poisson}(1)$



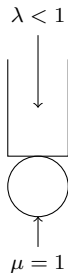
# Model setup

- $N$  strategic servers
- Jobs arrive with  $\text{Poisson}(\lambda)$  where  $\lambda < 1$ ; capacity units arrive with  $\text{Poisson}(1)$
- **Costs:**
  1. Holding cost: each waiting job costs one per unit of time
  2. Processing cost: serving a job costs  $c \geq 0$
- **Servers' objective:** minimizing own time-average total cost



# Model setup

- $N$  strategic servers
- Jobs arrive with  $\text{Poisson}(\lambda)$  where  $\lambda < 1$ ; capacity units arrive with  $\text{Poisson}(1)$
- **Costs:**
  1. Holding cost: each waiting job costs one per unit of time
  2. Processing cost: serving a job costs  $c \geq 0$
- **Servers' objective:** minimizing own time-average total cost
- Limited information:
  - (a) other servers' arrivals and actions are unobservable
  - (b) precise knowledge of number of servers  $N$  not required (except knowing that it is relative large)





# Related Literature

## Resource pooling:

- Power of resource pooling: [Tsitsiklis and Xu, 2013]
- Decentralized setup with two servers: [Hu and Caldentey, 2023]

## Mean-field equilibrium:

- Analysis of complex operational problems: [Iyer et al., 2014], [Balseiro et al., 2015], [Kanoria and Saban, 2021], [Arnosti et al., 2021]
- Fluid mean-field equilibrium similar in spirit to [Balseiro et al., 2015]

## Scrip system:

- Analysis of scrip system: [Kash et al., 2007], [Kash et al., 2015], [Johnson et al., 2014], [Bo et al., 2018]

## Other related work:

- Cooperative game model: [Anily and Haviv, 2010], [Anily and Haviv, 2014], [Karsten et al., 2015]
- Supermarket game: [Xu and Hajek, 2013], [Yang et al., 2019]

# Outline

- **Motivation, problem, and literature review (done)**
- **Token-based mechanism**
  - ▶ Analytical tool: Fluid mean-field equilibrium (FMFE)
  - ▶ FMFE characterization
  - ▶ Designing key element of mechanism
- **FMFE strategy as a near-optimal best response**
  - ▶ Asymptotic analysis for large markets
  - ▶ Numerical analysis for small markets
- **Extension to heterogeneous servers**
- **Conclusions**

# Token-based mechanism

In the mechanism, a server can:

- Request help from others without recall at any time.
- When a capacity unit arrives, either: (i) process its job, (ii) help others, or (iii) be idle and waste the unit without recall.

# Token-based mechanism

In the mechanism, a server can:

- Request help from others without recall at any time.
  - ▶ Requested jobs relocate to shared pool
- When a capacity unit arrives, either: (i) process its job, (ii) help others, or (iii) be idle and waste the unit without recall. If a server offers help:
  - ▶ The *oldest* job in shared pool is served (if pool is non-empty)
- A **shared pool** to match requests and provisions of help in FCFS order.

# Token-based mechanism

In the mechanism, a server can:

- Request help from others without recall at any time.
  - ▶ Requested jobs relocate to shared pool
- When a capacity unit arrives, either: (i) process its job, (ii) help others, or (iii) be idle and waste the unit without recall. If a server offers help:
  - ▶ The *oldest* job in shared pool is served (if pool is non-empty)
- A **shared pool** to match requests and provisions of help in FCFS order.
  - ▶ Shared pool queue length is unobservable, but servers can infer it.

# Token-based mechanism

In the mechanism, a server can:

- Request help from others without recall at any time.
  - ▶ Requested jobs relocate to shared pool
  - ▶ Each request costs one token
- When a capacity unit arrives, either: (i) process its job, (ii) help others, or (iii) be idle and waste the unit without recall. If a server offers help:
  - ▶ The *oldest* job in shared pool is served (if pool is non-empty)
  - ▶ A token is rewarded with prob  $\phi \in (0, 1)$
- A **shared pool** to match requests and provisions of help in FCFS order.
  - ▶ Shared pool queue length is unobservable, but servers can infer it.
- A **token system** to mitigate free riding.

# Token-based mechanism

In the mechanism, a server can:

- Request help from others without recall at any time.
  - ▶ Requested jobs relocate to shared pool
  - ▶ Each request costs one token
- When a capacity unit arrives, either: (i) process its job, (ii) help others, or (iii) be idle and waste the unit without recall. If a server offers help:
  - ▶ The *oldest* job in shared pool is served (if pool is non-empty)
  - ▶ A token is rewarded with prob  $\phi \in (0, 1)$
- A shared pool to match requests and provisions of help in FCFS order.
  - ▶ Shared pool queue length is unobservable, but servers can infer it.
- A token system to mitigate free riding.
- Servers interact via shared pool

# Token-based mechanism

In the mechanism, a server can:

- Request help from others without recall at any time.
  - ▶ Requested jobs relocate to shared pool
  - ▶ Each request costs one token
- When a capacity unit arrives, either: (i) process its job, (ii) help others, or (iii) be idle and waste the unit without recall. If a server offers help:
  - ▶ The *oldest* job in shared pool is served (if pool is non-empty)
  - ▶ A token is rewarded with prob  $\phi \in (0, 1)$
- A **shared pool** to match requests and provisions of help in FCFS order.
  - ▶ Shared pool queue length is unobservable, but servers can infer it.
- A **token system** to mitigate free riding.
- Servers interact via shared pool
- The value of  $\phi$  is critical to system performance



# Equilibrium concept: Fluid mean-field equilibrium

Approximation methodology similar to (Balseiro et al. 2015)

- **Mean-field approximation:** each server optimizes by assuming state of shared pool is fixed at long-run average  $\implies$ 
  - ▶ Expected waiting time in shared pool is constant  $w \geq 0$ : value determined endogenously by equilibrium
  - ▶ Probability that shared pool is non-empty is constant: equal to  $\phi$  !

# Equilibrium concept: Fluid mean-field equilibrium

Approximation methodology similar to (Balseiro et al. 2015)

- **Mean-field approximation:** each server optimizes by assuming state of shared pool is fixed at long-run average  $\implies$ 
  - ▶ Expected waiting time in shared pool is constant  $w \geq 0$ : value determined endogenously by equilibrium
  - ▶ Probability that shared pool is non-empty is constant: equal to  $\phi$  !

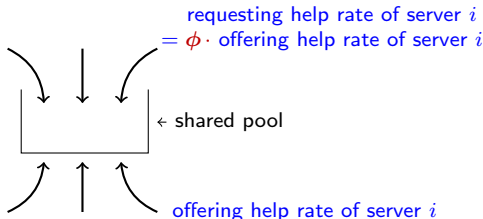
For each server:

$$\begin{aligned} \text{rate of requesting help} &= \text{rate of spending tokens} \\ &= \text{rate of earning tokens} = \phi \cdot \text{rate of offering help} \end{aligned}$$

# Equilibrium concept: Fluid mean-field equilibrium

Approximation methodology similar to (Balseiro et al. 2015)

- **Mean-field approximation:** each server optimizes by assuming state of shared pool is fixed at long-run average  $\implies$ 
  - ▶ Expected waiting time in shared pool is constant  $w \geq 0$ : value determined endogenously by equilibrium
  - ▶ Probability that shared pool is non-empty is constant: equal to  $\phi$  !



For each server:

rate of requesting help = rate of spending tokens

= rate of earning tokens =  $\phi \cdot$  rate of offering help

# Equilibrium concept: Fluid mean-field equilibrium

Approximation methodology similar to (Balseiro et al. 2015)

- **Mean-field approximation:** each server optimizes by assuming state of shared pool is fixed at long-run average  $\implies$ 
  - ▶ Expected waiting time in shared pool is constant  $w \geq 0$ : value determined endogenously by equilibrium
  - ▶ Probability that shared pool is non-empty is constant: equal to  $\phi$  !
- **Fluid relaxation:** allow # of tokens to be negative; only require that tokens satisfy the flow balance constraint in expectation

# Equilibrium concept: Fluid mean-field equilibrium

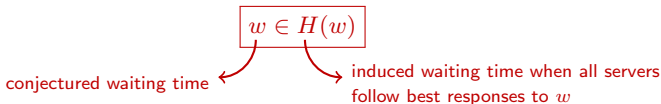
Approximation methodology similar to (Balseiro et al. 2015)

- **Mean-field approximation:** each server optimizes by assuming state of shared pool is fixed at long-run average  $\implies$ 
  - ▶ Expected waiting time in shared pool is constant  $w \geq 0$ : value determined endogenously by equilibrium
  - ▶ Probability that shared pool is non-empty is constant: equal to  $\phi$  !
- **Fluid relaxation:** allow # of tokens to be negative; only require that tokens satisfy the flow balance constraint in expectation
- After simplification: server's best response depends only on its queue length  
 $\implies$  Closed-form characterization (next slide)

# Equilibrium concept: Fluid mean-field equilibrium

Approximation methodology similar to (Balseiro et al. 2015)

- **Mean-field approximation:** each server optimizes by assuming state of shared pool is fixed at long-run average  $\implies$ 
  - ▶ Expected waiting time in shared pool is constant  $w \geq 0$ : value determined endogenously by equilibrium
  - ▶ Probability that shared pool is non-empty is constant: equal to  $\phi$  !
- **Fluid relaxation:** allow # of tokens to be negative; only require that tokens satisfy the flow balance constraint in expectation
- After simplification: server's best response depends only on its queue length  
 $\implies$  Closed-form characterization (next slide)
- Fluid mean-field equilibrium (FMFE):



# Server's best response

**Closed-form solution:** threshold policy w.r.t. queue length:

- Request help only when queue length exceeds a certain threshold (which depends on  $\phi$  and  $w$ )
- Offer help only when queue is empty

# Server's best response

**Closed-form solution:** threshold policy w.r.t. queue length:

- Request help only when queue length exceeds a certain threshold (which depends on  $\phi$  and  $w$ )
- Offer help only when queue is empty

**Proposition** (informal). Suppose  $\exists \bar{w} < \infty$  such that all servers believe that  $w \leq \bar{w}$ ; then  $w = O\left(\frac{1}{N}\right)$ .

*Proof:* Using a drift analysis.



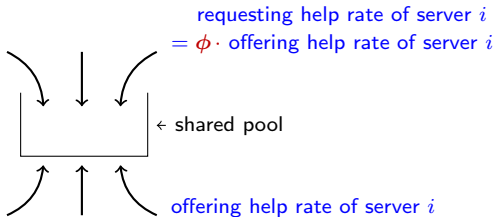
# Server's best response

**Closed-form solution:** threshold policy w.r.t. queue length:

- Request help only when queue length exceeds a certain threshold (which depends on  $\phi$  and  $w$ )
- Offer help only when queue is empty

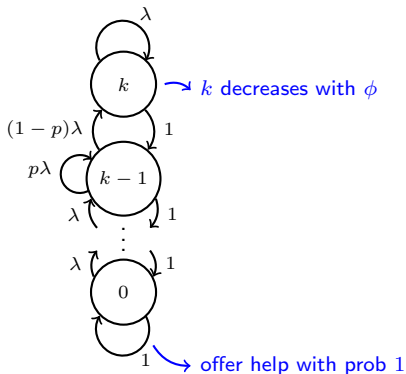
**Proposition** (informal). Suppose  $\exists \bar{w} < \infty$  such that all servers believe that  $w \leq \bar{w}$ ; then  $w = O(\frac{1}{N})$ .

*Proof:* Using a drift analysis.

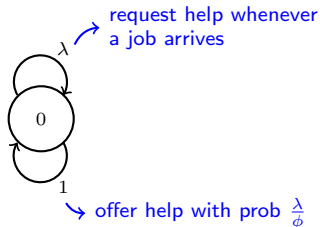


# Best response when $w < 1$

(Unique) best response when  $w < 1$ :



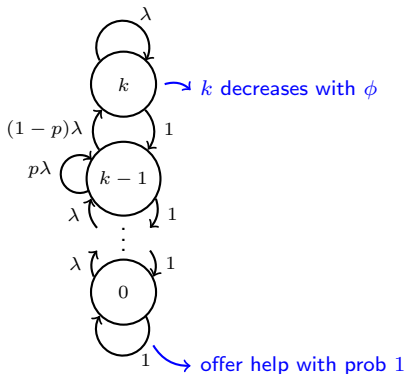
$$\phi \leq \lambda$$



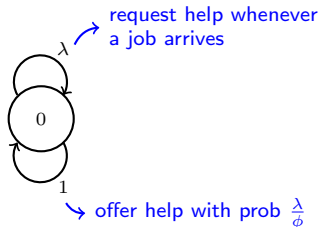
$$\phi \geq \lambda$$

# Best response when $w < 1$

(Unique) best response when  $w < 1$ :



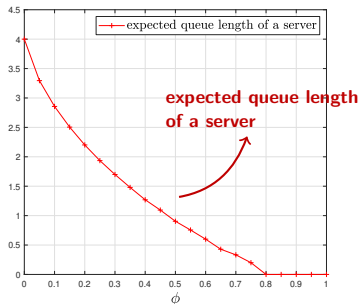
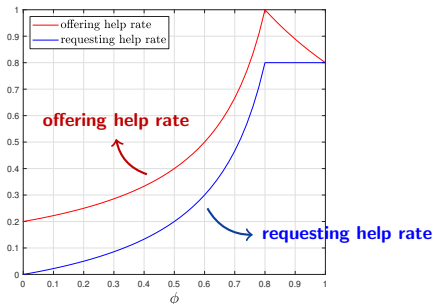
$$\phi \leq \lambda$$



$$\phi \geq \lambda$$

**Proposition.** For any  $\phi \in (0, 1)$ , if all servers play the above strategy, it forms a FMFE when number of servers  $N$  is large.

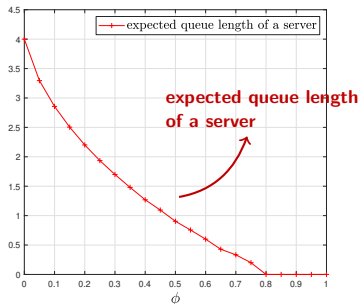
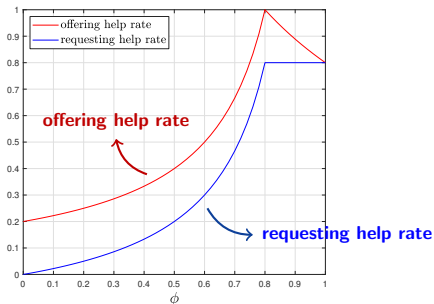
# Optimal value of $\phi$



**Proposition.** The expected total number of jobs in system, denoted by  $Q_{\Sigma}(\phi)$ , satisfies:

1. When  $\phi < \lambda$ :  $\lim_{N \rightarrow \infty} Q_{\Sigma}(\phi)/N = q(\phi) > 0$
2. When  $\phi \geq \lambda$ :  $Q_{\Sigma}(\phi) = \frac{\phi}{1-\phi}$

# Optimal value of $\phi$



**Main result:** The optimal value is  $\phi = \lambda$ . Moreover, this induces complete resource pooling: it is each server's best strategy to (i) request help whenever a job arrives, (i) offer help when queue is empty.

⇒ System's dynamics and performance match those under centralized control

# Asymptotic analysis for large markets

- Servers  $i \geq 2$  follow FMFE strategy; server one minimizes own cost.

# Asymptotic analysis for large markets

- Servers  $i \geq 2$  follow FMFE strategy; server one minimizes own cost.
- Optimal value of fluid mean-field problem with  $w = 0$ :  $c\lambda + \mathbb{E}[Q^F]$   
 $\mathbb{E}[Q^F]$ : a server's queue length when it follows the FMFE strategy

# Asymptotic analysis for large markets

- Servers  $i \geq 2$  follow FMFE strategy; server one minimizes own cost.
- Optimal value of fluid mean-field problem with  $w = 0$ :  $c\lambda + \mathbb{E}[Q^F]$   
 $\mathbb{E}[Q^F]$ : a server's queue length when it follows the FMFE strategy

**Lemma.** If server one also follows FMFE strategy, its time-average total cost is upper-bounded by  $c\lambda + \mathbb{E}[Q^F] + \frac{C_1(\lambda, \phi)}{N}$ .



# Asymptotic analysis for large markets

- Servers  $i \geq 2$  follow FMFE strategy; server one minimizes own cost.
- Optimal value of fluid mean-field problem with  $w = 0$ :  $c\lambda + \mathbb{E}[Q^F]$   
 $\mathbb{E}[Q^F]$ : a server's queue length when it follows the FMFE strategy

**Lemma.** If server one also follows FMFE strategy, its time-average total cost is upper-bounded by  $c\lambda + \mathbb{E}[Q^F] + \frac{C_1(\lambda, \phi)}{N}$ .

**Lemma.** Regardless of the strategy server one uses, its time-average total cost is lower-bounded by  $c\lambda + \mathbb{E}[Q^F] - \frac{C_2(\lambda, \phi, \delta)}{N^{1-\delta}}$  for any  $\delta \in (0, 1)$ .

# Asymptotic analysis for large markets

- Servers  $i \geq 2$  follow FMFE strategy; server one minimizes own cost.
- Optimal value of fluid mean-field problem with  $w = 0$ :  $c\lambda + \mathbb{E}[Q^F]$   
 $\mathbb{E}[Q^F]$ : a server's queue length when it follows the FMFE strategy

**Lemma.** If server one also follows FMFE strategy, its time-average total cost is upper-bounded by  $c\lambda + \mathbb{E}[Q^F] + \frac{C_1(\lambda, \phi)}{N}$ .

**Lemma.** Regardless of the strategy server one uses, its time-average total cost is lower-bounded by  $c\lambda + \mathbb{E}[Q^F] - \frac{C_2(\lambda, \phi, \delta)}{N^{1-\delta}}$  for any  $\delta \in (0, 1)$ .

*Proof sketch:*

1. A relaxation to server one's problem: grant an additional power to empty the shared pool at the end of every interaction  
 $\Rightarrow$  Request help only when a job arrives

# Asymptotic analysis for large markets

- Servers  $i \geq 2$  follow FMFE strategy; server one minimizes own cost.
- Optimal value of fluid mean-field problem with  $w = 0$ :  $c\lambda + \mathbb{E}[Q^F]$   
 $\mathbb{E}[Q^F]$ : a server's queue length when it follows the FMFE strategy

**Lemma.** If server one also follows FMFE strategy, its time-average total cost is upper-bounded by  $c\lambda + \mathbb{E}[Q^F] + \frac{C_1(\lambda, \phi)}{N}$ .

**Lemma.** Regardless of the strategy server one uses, its time-average total cost is lower-bounded by  $c\lambda + \mathbb{E}[Q^F] - \frac{C_2(\lambda, \phi, \delta)}{N^{1-\delta}}$  for any  $\delta \in (0, 1)$ .

*Proof sketch:*

1. A relaxation to server one's problem: grant an additional power to empty the shared pool at the end of every interaction  
 $\Rightarrow$  Request help only when a job arrives
2. A coupling argument and a drift analysis to show:
  - (a) shared pool's queue length transitions to stationary distribution quickly as  $N \rightarrow \infty$
  - (b) in stationary distribution, shared pool is non-empty with probability  $\phi - \frac{c(\lambda, \phi, \delta)}{N^{1-\delta}}$

# Analysis for small market

- Mechanism uses  $\phi = \lambda$ .
- Consider the fluid setup: tokens can go negative but expected rates of earning and spending tokens are equal.
- Servers  $i \geq 2$  adopt complete resource pooling; server one is strategic and minimizes own cost.

# Analysis for small market

- Mechanism uses  $\phi = \lambda$ .
- Consider the fluid setup: tokens can go negative but expected rates of earning and spending tokens are equal.
- Servers  $i \geq 2$  adopt complete resource pooling; server one is strategic and minimizes own cost.

Grant server one additional information edge:

- Complete information about the shared pool's queue length (denoted by  $q_0$ )

# Analysis for small market

- Mechanism uses  $\phi = \lambda$ .
- Consider the fluid setup: tokens can go negative but expected rates of earning and spending tokens are equal.
- Servers  $i \geq 2$  adopt complete resource pooling; server one is strategic and minimizes own cost.

Grant server one additional information edge:

- Complete information about the shared pool's queue length (denoted by  $q_0$ )

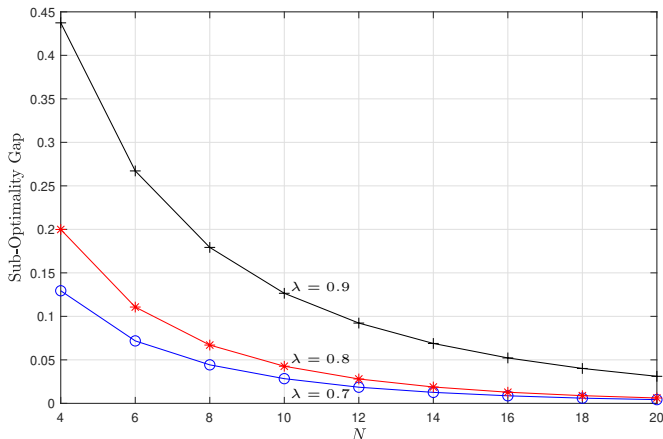
$\Rightarrow$  Optimal strategy depends only on two states:  $q_1$  (own queue length) and  $q_0$

$\implies$  Tractable optimization problem!

# Numerical results

(a) job processing cost  $c = 1$ ; (b) job arrival rate  $\lambda \in \{0.7, 0.8, 0.9\}$

Sub-optimality gap =  $\frac{\text{Cost of complete resource pooling} - \text{Cost of optimal strategy}}{\text{Cost of optimal strategy}}$



- The value of playing strategically is small even with few servers (and when server one can perfectly monitor the shared pool)

## Extension: heterogeneous servers

- For each server  $i$ : job arrival rate  $\lambda_i$  and processing rate  $\mu_i$ ; let  $\rho_i = \frac{\lambda_i}{\mu_i}$
- Assume  $0 < \underline{\rho} \leq \rho_i \leq \bar{\rho} < 1$  and  $0 < \underline{\lambda} \leq \lambda_i \leq \bar{\lambda}$  for all servers



## Extension: heterogeneous servers

- For each server  $i$ : job arrival rate  $\lambda_i$  and processing rate  $\mu_i$ ; let  $\rho_i = \frac{\lambda_i}{\mu_i}$
- Assume  $0 < \underline{\rho} \leq \rho_i \leq \bar{\rho} < 1$  and  $0 < \underline{\lambda} \leq \lambda_i \leq \bar{\lambda}$  for all servers
- Consider token-based mechanism with  $\phi = \bar{\rho}$

## Extension: heterogeneous servers

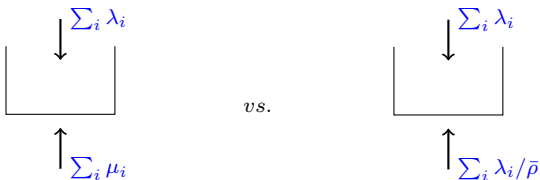
- For each server  $i$ : job arrival rate  $\lambda_i$  and processing rate  $\mu_i$ ; let  $\rho_i = \frac{\lambda_i}{\mu_i}$
- Assume  $0 < \underline{\rho} \leq \rho_i \leq \bar{\rho} < 1$  and  $0 < \underline{\lambda} \leq \lambda_i \leq \bar{\lambda}$  for all servers
- Consider token-based mechanism with  $\phi = \bar{\rho}$

**Proposition** *It is FMFE and approximate equilibrium for each server to (i) request help for all incoming jobs, and (ii) offer help with probability  $\rho_i/\bar{\rho}$  when a capacity unit arrives, when number of servers is large.*

# Extension: heterogeneous servers

- For each server  $i$ : job arrival rate  $\lambda_i$  and processing rate  $\mu_i$ ; let  $\rho_i = \frac{\lambda_i}{\mu_i}$
- Assume  $0 < \underline{\rho} \leq \rho_i \leq \bar{\rho} < 1$  and  $0 < \underline{\lambda} \leq \lambda_i \leq \bar{\lambda}$  for all servers
- Consider token-based mechanism with  $\phi = \bar{\rho}$

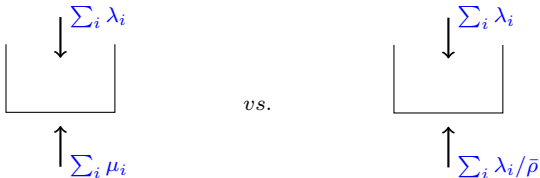
**Proposition** *It is FMFE and approximate equilibrium for each server to (i) request help for all incoming jobs, and (ii) offer help with probability  $\rho_i/\bar{\rho}$  when a capacity unit arrives, when number of servers is large.*



# Extension: heterogeneous servers

- For each server  $i$ : job arrival rate  $\lambda_i$  and processing rate  $\mu_i$ ; let  $\rho_i = \frac{\lambda_i}{\mu_i}$
- Assume  $0 < \underline{\rho} \leq \rho_i \leq \bar{\rho} < 1$  and  $0 < \underline{\lambda} \leq \lambda_i \leq \bar{\lambda}$  for all servers
- Consider token-based mechanism with  $\phi = \bar{\rho}$

**Proposition** *It is FMFE and approximate equilibrium for each server to (i) request help for all incoming jobs, and (ii) offer help with probability  $\rho_i/\bar{\rho}$  when a capacity unit arrives, when number of servers is large.*

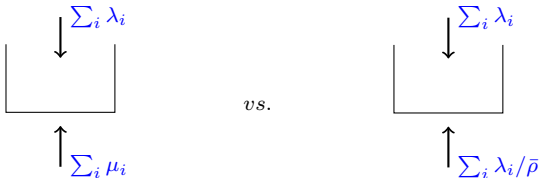


- Number of jobs in centralized setting: between  $\frac{\underline{\rho}}{1-\underline{\rho}}$  and  $\frac{\bar{\rho}}{1-\bar{\rho}}$   
 Number of jobs within our mechanism:  $\frac{\bar{\rho}}{1-\bar{\rho}}$

# Extension: heterogeneous servers

- For each server  $i$ : job arrival rate  $\lambda_i$  and processing rate  $\mu_i$ ; let  $\rho_i = \frac{\lambda_i}{\mu_i}$
- Assume  $0 < \underline{\rho} \leq \rho_i \leq \bar{\rho} < 1$  and  $0 < \underline{\lambda} \leq \lambda_i \leq \bar{\lambda}$  for all servers
- Consider token-based mechanism with  $\phi = \bar{\rho}$

**Proposition** *It is FMFE and approximate equilibrium for each server to (i) request help for all incoming jobs, and (ii) offer help with probability  $\rho_i/\bar{\rho}$  when a capacity unit arrives, when number of servers is large.*



- Number of jobs in centralized setting: between  $\frac{\rho}{1-\underline{\rho}}$  and  $\frac{\bar{\rho}}{1-\bar{\rho}}$   
 Number of jobs within our mechanism:  $\frac{\bar{\rho}}{1-\bar{\rho}}$
- Job processing costs are allocated  $\propto \mu_i$  versus  $\propto \lambda_i$   
 $\Rightarrow$  Costs allocated fairly in our mechanism!

# Summary

- We study incentivizing resource pooling in a decentralized multi-server system, where servers possess limited information about others
- Operational takeaway: A simple **token-based mechanism** incentivizes *complete* resource pooling when number of servers is large
  - ▶ Analysis based on fluid mean-field equilibrium.
  - ▶ Numerical results show that benefit from unilateral deviation is small with only a few servers.

**Reference:** C. Chen, Y. Chen, and P. Qian. 2023. Incentivizing Resource Pooling.

Working paper available at <https://papers.ssrn.com/abstract=4586771>

# Appendix