Incentivizing Resource Pooling

Chen Chen

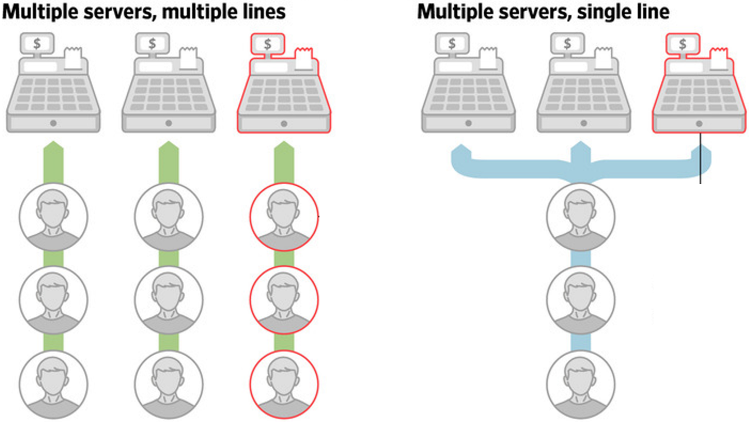
上海纽约大学
NYU SHANGHAI

November 2023

Joint work with:
Yilun Chen (CUHK-SZ) and Pengyu Qian (Purdue)

The Line Dance

Queuing theory, the mathematical study of lines, helps businesses, call centers, computer networks and others figure out how to keep things moving.

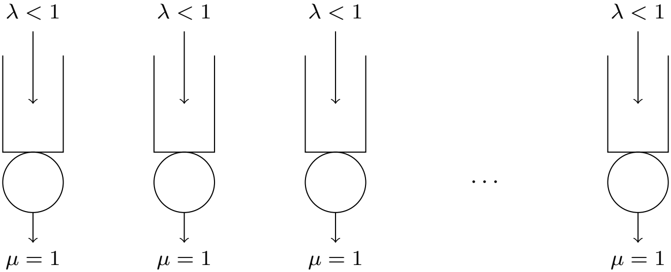


THE WALL STREET JOURNAL.

Resource pooling significantly improves service

Resource pooling: known fact

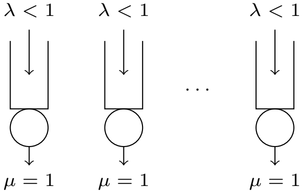
N servers: job arrival rate $\lambda < 1$, server processing rate $\mu = 1$



Resource pooling: known fact

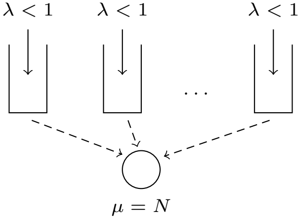
N servers: job arrival rate $\lambda < 1$, server processing rate $\mu = 1$

Without resource pooling:



$vs.$

With resource pooling:



 # jobs in system: $N \cdot \frac{\lambda}{1-\lambda}$ (linear) vs. $\frac{\lambda}{1-\lambda}$ (constant)

Can resource pooling be achieved in **decentralized** systems?

Can resource pooling be achieved in **decentralized** systems?

Decentralization boosts security, privacy, and scalability

Motivation

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

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

Motivation

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

Golem Network	Akash Network	iExec
Market cap: \$170M	Market cap: \$320M	Market cap: \$75M

- Essential aspects of the problem:
  - Large-scale system: number of servers N is large
  - Servers have limited information about one another

Main results

Main results

- Develop a simple token-based mechanism that incentives complete resource pooling in private information setting when N is large

  $\Longrightarrow$ System dynamics and performance match those under centralized control in the asymptotics

Main results

- Develop a simple token-based mechanism that incentives complete resource pooling in private information setting when N is large

  $\Longrightarrow$ System dynamics and performance match those under centralized control in the asymptotics

- Propose an approximation-based analytical framework

Main results

- Develop a simple token-based mechanism that incentives complete resource pooling in private information setting when N is large

  $\Longrightarrow$ System dynamics and performance match those under centralized control in the asymptotics

- Propose an approximation-based analytical framework
  - Simplifies the design and analysis of token-based mechanisms

Main results

- Develop a simple token-based mechanism that incentives complete resource pooling in private information setting when N is large

  $\Longrightarrow$ System dynamics and performance match those under centralized control in the asymptotics

- Propose an approximation-based analytical framework
  - Simplifies the design and analysis of token-based mechanisms
  - Provides tight theoretical guarantees

Main results

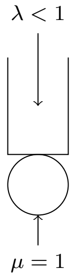
- Develop a simple token-based mechanism that incentives complete resource pooling in private information setting when N is large

  $\Longrightarrow$ System dynamics and performance match those under centralized control in the asymptotics

- Propose an approximation-based analytical framework
  - Simplifies the design and analysis of token-based mechanisms
  - Provides tight theoretical guarantees
  - Can be applied to more general settings

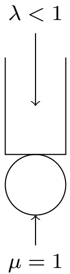
Model setup

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



Model setup

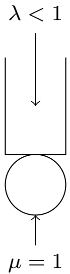
- N strategic servers
- Jobs arrive with Poisson(λ) where $\lambda < 1$; capacity units arrive with Poisson(1)
- **Costs:**
  1. Holding cost: each waiting job costs one per unit of time
  2. Processing cost: serving a job costs $c \geq 0$



- **Servers' objective:** minimizing own long-run average total cost

Model setup

- N strategic servers
- Jobs arrive with Poisson(λ) where $\lambda < 1$; capacity units arrive with 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 long-run average total cost
- Limited information:
  (a) A server's arrivals and actions are private information
  (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, research question, and literature review**
- **Token-based mechanism**
  - Solution concept: Fluid mean-field equilibrium (FMFE)
  - Characterization of FMFE
  - Designing key element of mechanism
- **FMFE strategy as near-optimal best response**
  - Asymptotic analysis for large markets
  - Numerical analysis for small markets
- **Extension to heterogeneous servers**
- **Takeaway**

Token-based mechanism

In the mechanism, a server can:

- Request help from others <u>without recall</u> at any time.

- When a capacity unit arrives, either: (i) serve its job, (ii) help others, or (iii) be idle and waste the unit <u>without recall</u>.

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) serve 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) serve 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) serve 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) serve 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) serve 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 ϕ 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 ϕ !

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 $\Longrightarrow$
  - 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 ϕ !

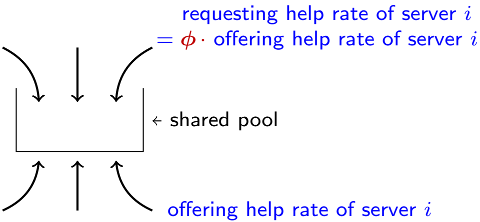
For each server:

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

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 ϕ !



For each server:

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

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 $\Longrightarrow$
  - 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 ϕ !
- **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 ϕ !
- **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 ϕ !
- **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):

$$\boxed{w \in H(w)}$$

conjectured waiting time

induced waiting time when all servers follow best responses to w

Server's best response

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

- Request help only when queue length exceeds a threshold k (which depends on ϕ 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 threshold k (which depends on ϕ and w)
- Offer help only when queue is empty

Proposition. 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.

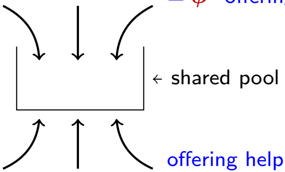
Server's best response

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

- Request help only when queue length exceeds a threshold k (which depends on ϕ and w)
- Offer help only when queue is empty

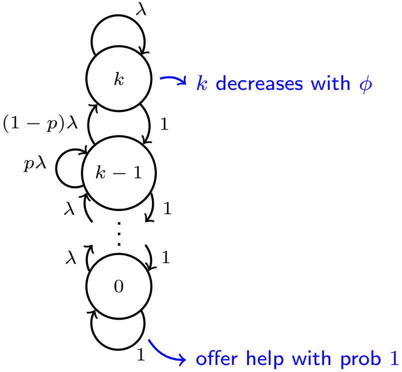
Proposition. Suppose $\exists\, \bar{w} < \infty$ such that all servers believe that $w \le \bar{w}$; then $w = O\left(\frac{1}{N}\right)$.

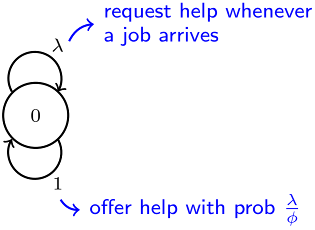
Proof: Using a drift analysis.



Best response when $w < 1$

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

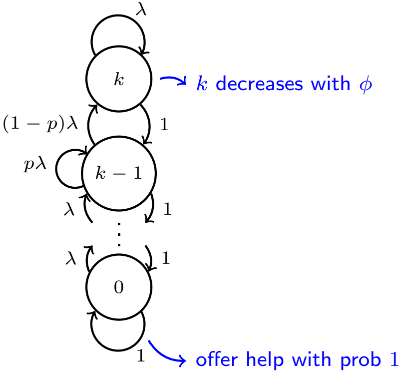




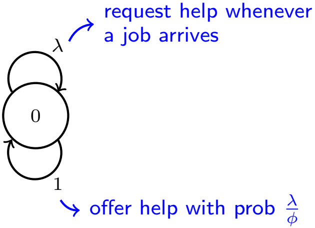
$\boldsymbol{\phi \leq \lambda}$ $\qquad$ $\boldsymbol{\phi \geq \lambda}$

Best response when $w < 1$

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



$\boldsymbol{\phi \leq \lambda}$

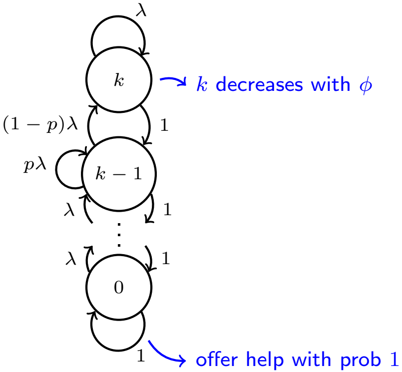


$\boldsymbol{\phi \geq \lambda}$

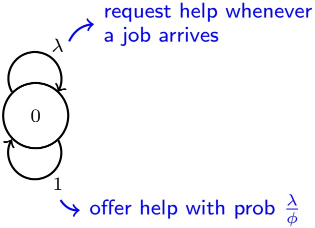
Proposition. For any $\phi \in (0, 1)$, if all servers play the above strategy, it forms a FMFE when number of servers N is large.

Best response when $w < 1$

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



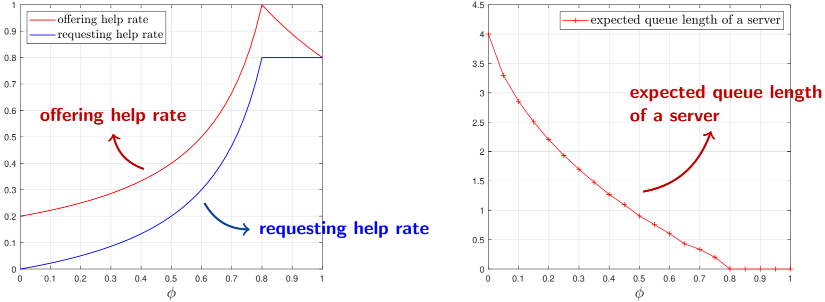
$\boldsymbol{\phi \leq \lambda}$



$\boldsymbol{\phi \geq \lambda}$

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. detail

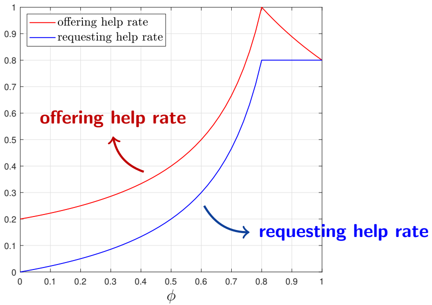
Optimal value of ϕ

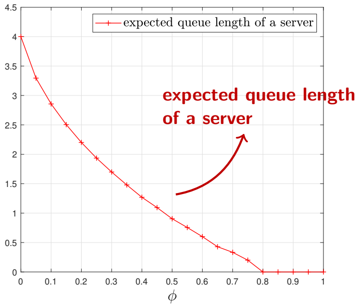


Proposition. The expected total number of jobs in system, denoted by $Q_\Sigma(\phi)$, satisfies:

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

Optimal value of ϕ





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

$\Longrightarrow$ System's dynamics and performance match those under centralized control

FMFE as **good approximation** of servers' strategics

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 $w = 0$: $c\lambda + \mathbb{E}\big[Q^{\mathrm{F}}\big]$

  $\mathbb{E}[Q^{\mathrm{F}}]$: a server's queue length when follows the FMFE strategy in the fluid problem

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 $w = 0$: $c\lambda + \mathbb{E}\big[Q^{\mathrm{F}}\big]$

  $\mathbb{E}[Q^{\mathrm{F}}]$: a server's queue length when follows the FMFE strategy in the fluid problem

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

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 $w = 0$: $c\lambda + \mathbb{E}\big[Q^{\mathrm{F}}\big]$

  $\mathbb{E}[Q^{\mathrm{F}}]$: a server's queue length when follows the FMFE strategy in the fluid problem

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

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

  $\mathbb{E}[Q^{\mathrm{F}}]$: a server's queue length when follows the FMFE strategy in the fluid problem

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

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

Proof sketch:

1. A relaxation to server one's problem: empower the server to empty the shared pool at the end of every interaction with shared pool

   $\Rightarrow$ 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 $w = 0$: $c\lambda + \mathbb{E}\big[Q^{\mathrm{F}}\big]$

  $\mathbb{E}[Q^{\mathrm{F}}]$: a server's queue length when follows the FMFE strategy in the fluid problem

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

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

Proof sketch:

1. A relaxation to server one's problem: empower the server to empty the shared pool at the end of every interaction with shared pool

   $\Rightarrow$ 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 $N \to \infty$
   (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)

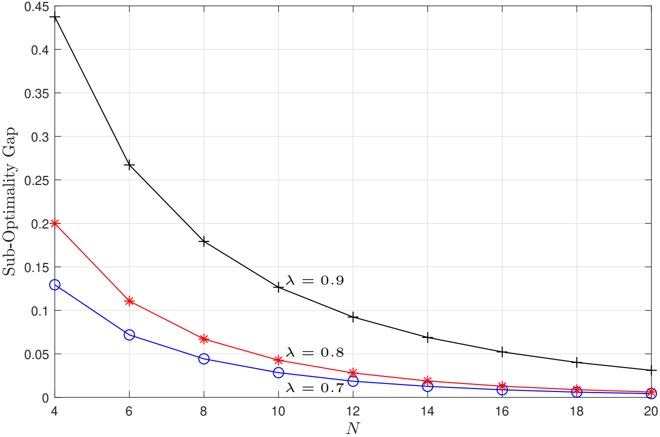
$\Rightarrow$ Optimal strategy depends only on two states: q_1 (own queue length) and q_0

$\Longrightarrow$ 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 λ_i and processing rate μ_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 λ_i and processing rate μ_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

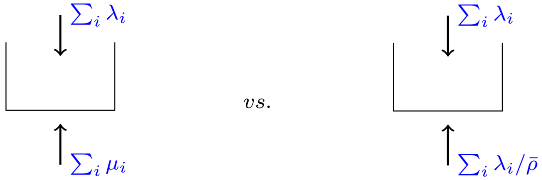
- For each server i: job arrival rate λ_i and processing rate μ_i; let $\rho_i = \frac{\lambda_i}{\mu_i}$
- Assume $0 < \underline{\rho} \leq \rho_i \leq \bar{\rho} < 1$ and $0 < \underline{\lambda} \leq \lambda_i \leq \bar{\lambda}$ for all servers
- Consider token-based mechanism with $\phi = \bar{\rho}$

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 λ_i and processing rate μ_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 $\phi = \bar{\rho}$

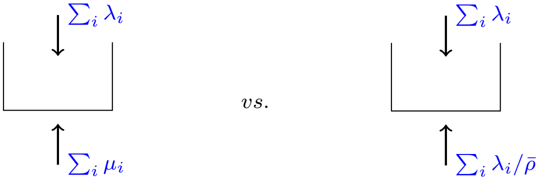
Proposition *It is FMFE and approximate equilibrium for each server to (i) request help for all incoming jobs, and (ii) offer help with probability $\rho_i/\bar{\rho}$ when a capacity unit arrives, when number of servers is large.*



Extension: heterogeneous servers

- For each server i: job arrival rate λ_i and processing rate μ_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.*

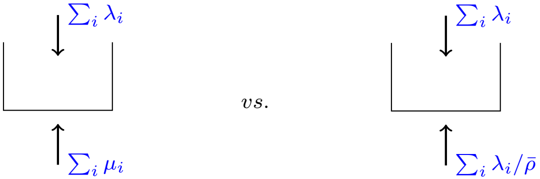


- Number of jobs in centralized setting: between $\frac{\underline{\rho}}{1-\underline{\rho}}$ and $\frac{\bar{\rho}}{1-\bar{\rho}}$
  Number of jobs within our mechanism: $\frac{\bar{\rho}}{1-\bar{\rho}}$

Extension: heterogeneous servers

- For each server i: job arrival rate λ_i and processing rate μ_i; let $\rho_i = \frac{\lambda_i}{\mu_i}$
- Assume $0 < \underline{\rho} \le \rho_i \le \bar{\rho} < 1$ and $0 < \underline{\lambda} \le \lambda_i \le \bar{\lambda}$ 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: $\frac{\bar{\rho}}{1-\bar{\rho}}$
- 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 setting, where servers have 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 even with only a few servers

Summary

- We study incentivizing resource pooling in a decentralized setting, where servers have 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 even with only a few servers
- **Ongoing work.** Applying the mechanism and analytical framework to other decentralized systems, e.g., multi-hospital kidney exchange.

Summary

- We study incentivizing resource pooling in a decentralized setting, where servers have 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 even with only a few servers

- **Ongoing work.** Applying the mechanism and analytical framework to other decentralized systems, e.g., multi-hospital kidney exchange.

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

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

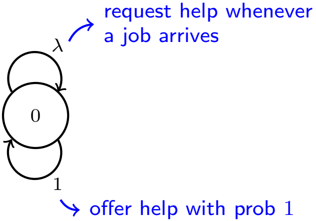
Appendix

Minimum number of servers to sustain FMFE

- FMFE necessitates $w \leq 1$.
- Minimum $\#$ servers can be specified analytically or numerically.

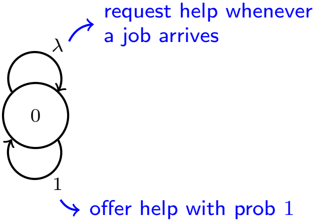
Minimum number of servers to sustain FMFE

- FMFE necessitates $w \leq 1$.
- Minimum # servers can be specified analytically or numerically.
- Example: suppose $\phi = \lambda$



Minimum number of servers to sustain FMFE

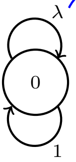
- FMFE necessitates $w \leq 1$.
- Minimum # servers can be specified analytically or numerically.
- Example: suppose $\phi = \lambda$
  - Shared pool is an $M/M/1$ queue
    $\Rightarrow w = \frac{\lambda}{1-\lambda} \cdot \frac{1}{N\lambda} = \frac{1}{(1-\lambda)N}$
  - $w \leq 1 \Rightarrow N \geq \left\lceil \frac{1}{1-\lambda} \right\rceil$



Minimum number of servers to sustain FMFE

- FMFE necessitates $w \leq 1$.
- Minimum # servers can be specified analytically or numerically.
- Example: suppose $\phi = \lambda$
  - Shared pool is an $M/M/1$ queue
    $\Rightarrow w = \frac{\lambda}{1-\lambda} \cdot \frac{1}{N\lambda} = \frac{1}{(1-\lambda)N}$
  - $w \leq 1 \Rightarrow N \geq \left\lceil \frac{1}{1-\lambda} \right\rceil$

request help whenever
a job arrives



offer help with prob 1

back