EINDHOVEN
UNIVERSITY OF
TECHNOLOGY

Adaptive Load Balancing under Assignment Constraints

NETWORKS Conference – Session “Scalable Network Algorithms and Performance”

Céline Comte

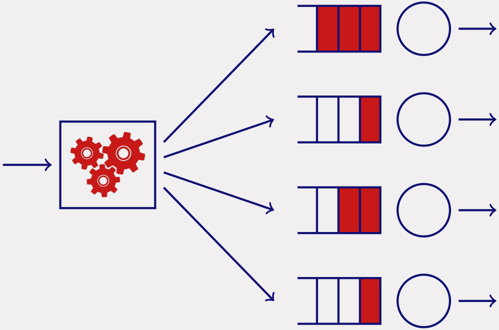
Outline

Outline

Model

Parameters

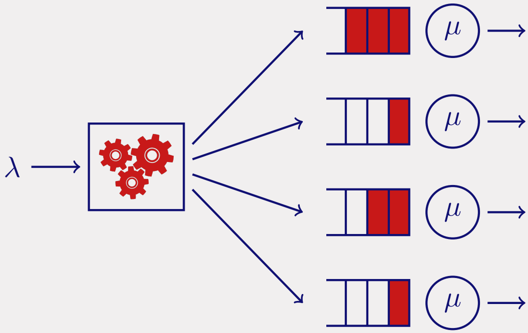
- 1 dispatcher, n servers, jobs



Model

Parameters

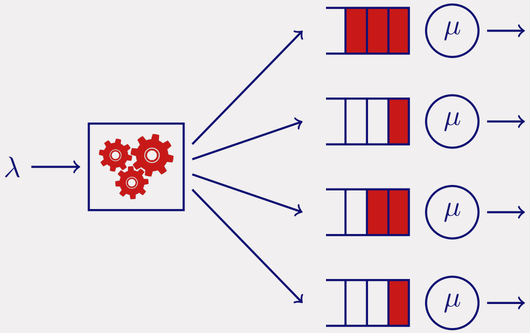
- 1 dispatcher, n servers, jobs
- Job arrival rate λ, service rate μ



Model

Parameters

- 1 dispatcher, n servers, jobs
- Job arrival rate λ, service rate μ
- Stability requires that $\lambda < n\mu$



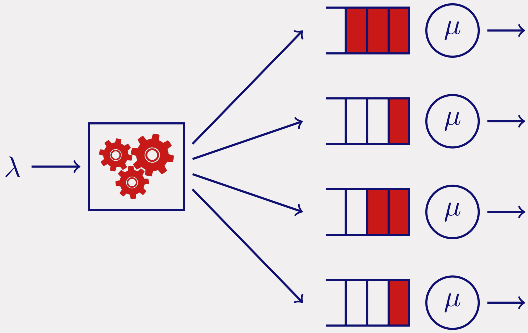
Model

Parameters

- 1 dispatcher, n servers, jobs
- Job arrival rate λ, service rate μ
- Stability requires that $\lambda < n\mu$

Randomness

- Poisson arrival process



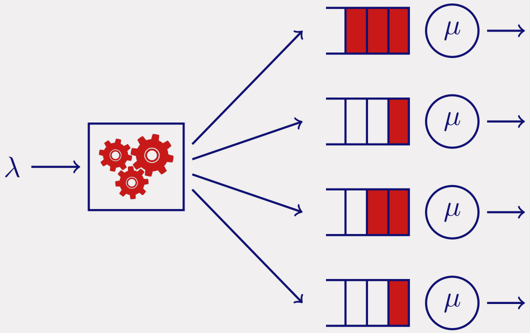
Model

Parameters

- 1 dispatcher, n servers, jobs
- Job arrival rate λ, service rate μ
- Stability requires that $\lambda < n\mu$

Randomness

- Poisson arrival process
- Exponential service times



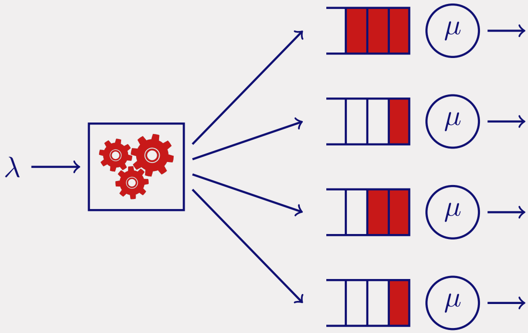
Model

Parameters

- 1 dispatcher, n servers, jobs
- Job arrival rate λ, service rate μ
- Stability requires that $\lambda < n\mu$

Randomness

- Poisson arrival process
- Exponential service times
- Continuous-time Markov chain



Model

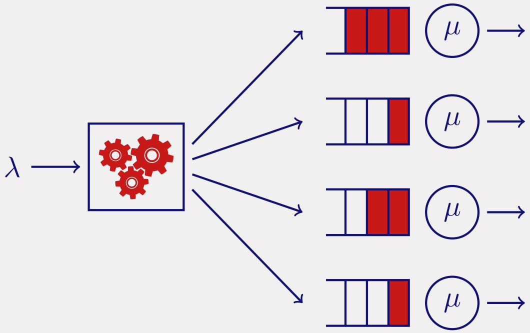
Parameters

- 1 dispatcher, n servers, jobs
- Job arrival rate λ, service rate μ
- Stability requires that $\lambda < n\mu$

Randomness

- Poisson arrival process
- Exponential service times
- Continuous-time Markov chain

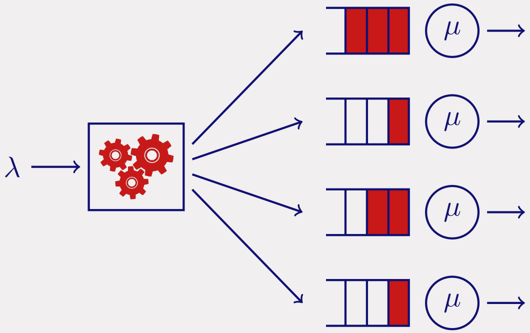
Examples: supermarket, cloud...



Model

Non-anticipating policies

- First-come-first-served scheduling
- Immediate, irrevocable assignment



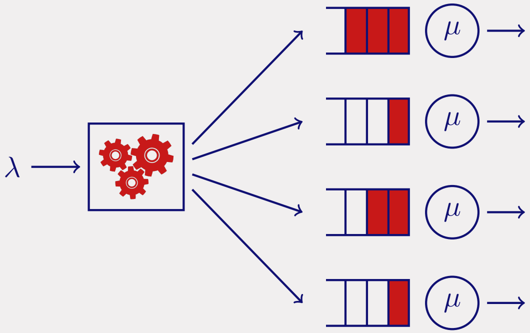
Model

Non-anticipating policies

- First-come-first-served scheduling
- Immediate, irrevocable assignment

Performance metrics

- Waiting probability
- Mean waiting time
- Communication cost (per job)



Model

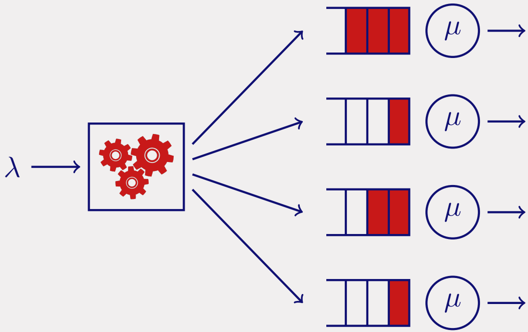
Non-anticipating policies

- First-come-first-served scheduling
- Immediate, irrevocable assignment

Performance metrics

- Waiting probability
- Mean waiting time
- Communication cost (per job)

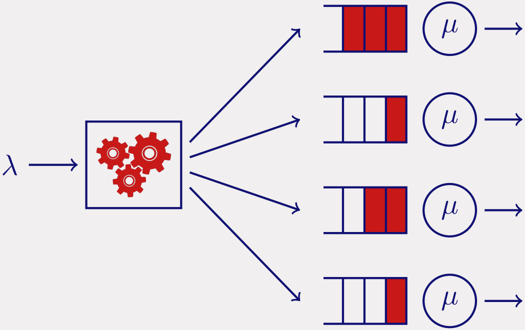
Variants: heterogeneity, scale...



Research questions

Algorithm design

- Efficient use of resources
- Scalability
- Predictable performance



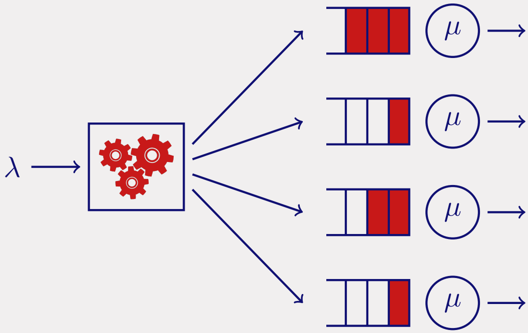
Research questions

Algorithm design

- Efficient use of resources
- Scalability
- Predictable performance

Performance evaluation

- Dimension infrastructure
- Compare algorithms
- Guarantee performance

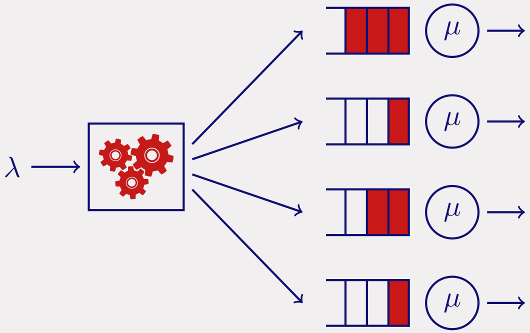


Outline

Join-idle-queue (JIQ)

Definition

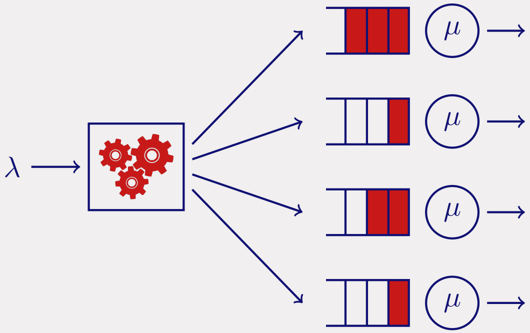
- Each server sends a token to the dispatcher when it becomes idle



Join-idle-queue (JIQ)

Definition

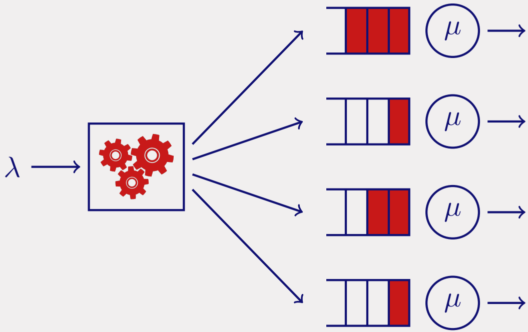
- Each server sends a token to the dispatcher when it becomes idle
- Assign incoming job to the longest idle server, if any



Join-idle-queue (JIQ)

Definition

- Each server sends a token to the dispatcher when it becomes idle
- Assign incoming job to the longest idle server, if any, otherwise to a server chosen uniformly at random



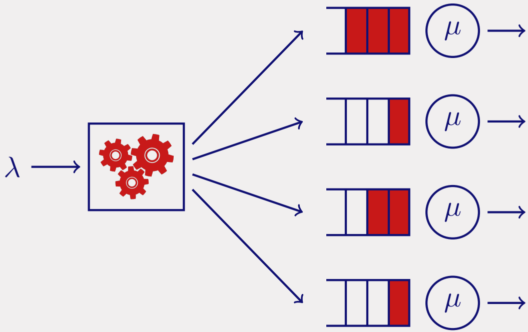
Join-idle-queue (JIQ)

Definition

- Each server sends a token to the dispatcher when it becomes idle
- Assign incoming job to the longest idle server, if any, otherwise to a server chosen uniformly at random

Performance

- Vanishing waiting probability



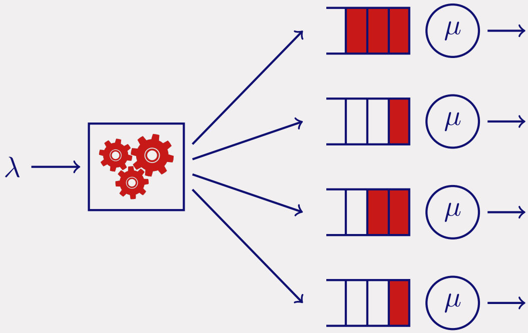
Join-idle-queue (JIQ)

Definition

- Each server sends a token to the dispatcher when it becomes idle
- Assign incoming job to the longest idle server, if any, otherwise to a server chosen uniformly at random

Performance

- Vanishing waiting probability
- Communication $\leq$ 1 per job



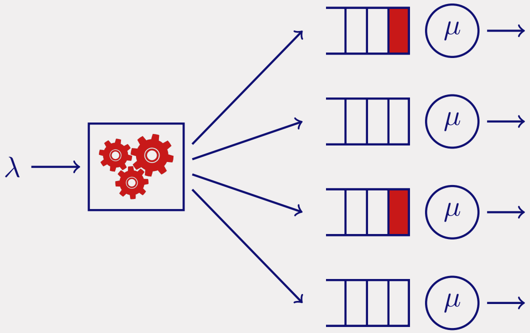
Join-idle-queue (JIQ)

Definition

- Each server sends a token to the dispatcher when it becomes idle
- Assign incoming job to the longest idle server, if any, otherwise to a server chosen uniformly at random

Performance

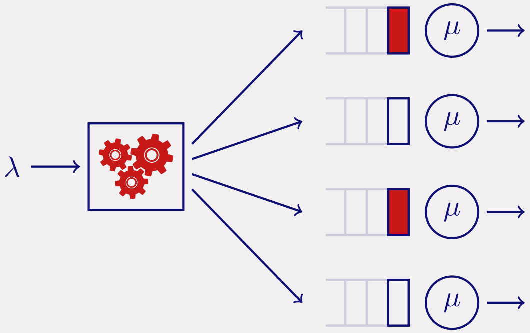
- Vanishing waiting probability
- Communication ≤ 1 per job



Join-idle-queue (JIQ)

Definition

- Each server sends a token to the dispatcher when it becomes idle
- Assign incoming job to the longest idle server, if any, otherwise to a server chosen uniformly at random



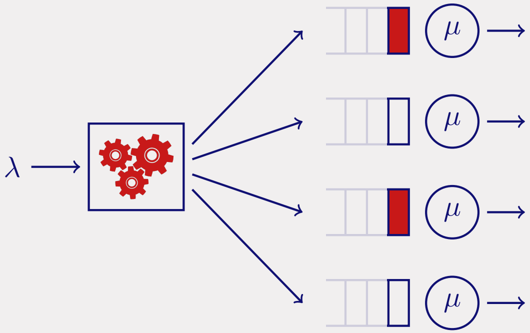
Performance

- Vanishing waiting probability
- Communication ≤ 1 per job

Assign-to-the-longest-idle-server (ALIS)

Definition

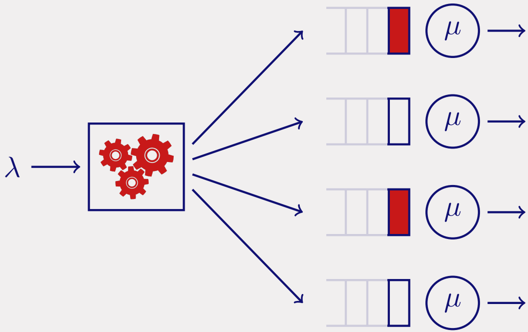
- Each server sends a token to the dispatcher when it becomes idle
- Assign incoming job to the longest idle server, if any~~, otherwise to a server chosen uniformly at random~~



Assign-to-the-longest-idle-server (ALIS)

Definition

- Each server sends a token to the dispatcher when it becomes idle
- Assign incoming job to the longest idle server, if any~~, otherwise to a server chosen uniformly at random~~



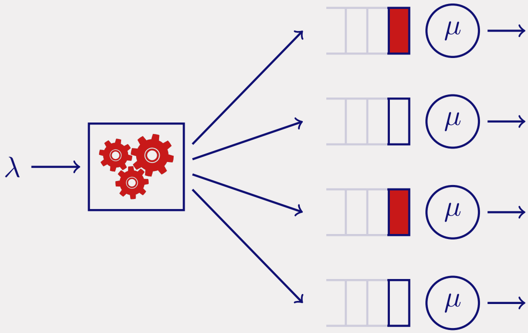
Performance

- Vanishing blocking probability
- Communication $=$ 1 per accepted job

Assign-to-the-longest-idle-server (ALIS)

Mathematical tractability

- Product-form stationary distribution (Erlang loss model)
- Performance is *insensitive* to the job size distribution



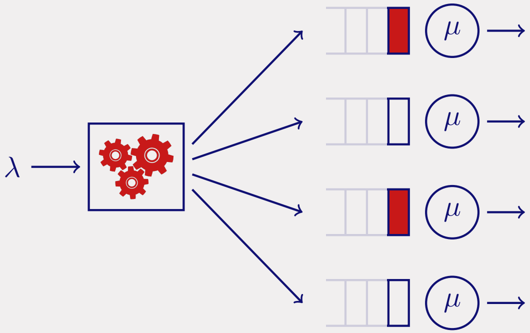
Assign-to-the-longest-idle-server (ALIS)

Mathematical tractability

- Product-form stationary distribution (Erlang loss model)
- Performance is *insensitive* to the job size distribution

Generalizations

- Heterogeneous servers
- Assignment constraints
- Longer queues



Related work

Join-idle-queue (JIQ)

- M. van der Boor et al. “Scalable load balancing in networked systems: A survey of recent advances”. (June 14, 2018). arXiv: 1806.05444
- Y. Lu et al. “Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services”. *Performance Evaluation*. Special Issue: Performance 2011 68.11 (Nov. 1, 2011), pp. 1056–1071
- M. van der Boor et al. “Load balancing in large-scale systems with multiple dispatchers”. IEEE INFOCOM 2017 - IEEE Conference on Computer Communications. May 2017, pp. 1–9

Related work

Assign-to-the-longest-idle-server (ALIS) and insensitive algorithms

- T. Bonald et al. "Insensitive Load Balancing". SIGMETRICS '04/Performance '04. New York, NY, USA: ACM, 2004, pp. 367–377
- I. Adan and G. Weiss. "A Loss System with Skill-based Servers under Assign to Longest Idle Server Policy". *Probability in the Engineering and Informational Sciences* 26.3 (July 2012), pp. 307–321
- M. Jonckheere and B. J. Prabhu. "Asymptotics of insensitive load balancing and blocking phases". *Queueing Systems* 88.3 (Apr. 1, 2018), pp. 243–278
- C. Comte. "Dynamic load balancing with tokens". *Computer Communications* 144 (Aug. 15, 2019), pp. 76–88

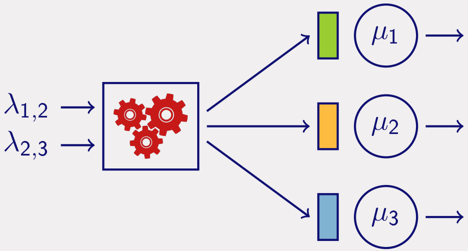
Outline

Assign-to-the-longest-idle-server (ALIS)

I. Adan and G. Weiss. “A Loss System with Skill-based Servers under Assign to Longest Idle Server Policy”. *Probability in the Engineering and Informational Sciences* 26.3 (July 2012), pp. 307–321

Assign-to-the-longest-idle-server (ALIS)

I. Adan and G. Weiss. "A Loss System with Skill-based Servers under Assign to Longest Idle Server Policy". *Probability in the Engineering and Informational Sciences* 26.3 (July 2012), pp. 307–321



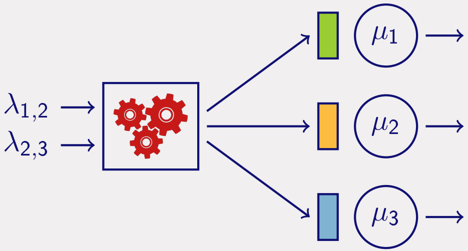
Model

- Arrival rates $\lambda_{1,2}$ and $\lambda_{2,3}$
- Service rates μ_1, μ_2, and μ_3

Assign-to-the-longest-idle-server (ALIS)

Load-balancing algorithm

- Each server sends a token to the dispatcher upon a service completion
- Assign incoming job to the longest idle compatible server, if any



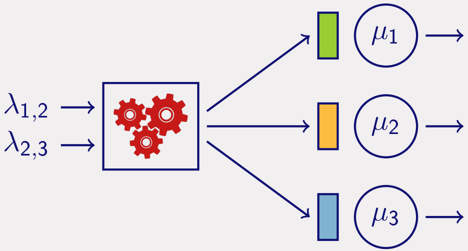
Assign-to-the-longest-idle-server (ALIS)

Load-balancing algorithm

- Each server sends a token to the dispatcher upon a service completion
- Assign incoming job to the longest idle compatible server, if any

Contributions

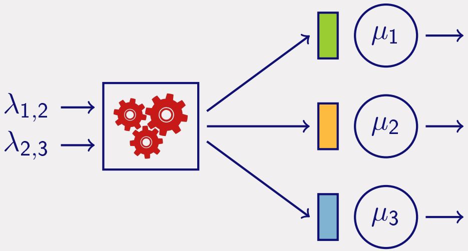
- Product-form stationary distribution



Assign-to-the-longest-idle-server (ALIS)

Load-balancing algorithm

- Each server sends a token to the dispatcher upon a service completion
- Assign incoming job to the longest idle compatible server, if any



Contributions

- Product-form stationary distribution
- Performance is *insensitive* to the job size distribution

Assign-to-the-longest-available-token

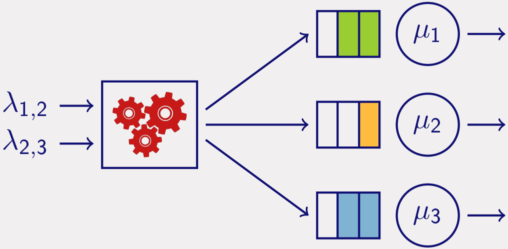
C. Comte. "Dynamic load balancing with tokens". *Computer Communications* 144 (Aug. 15, 2019), pp. 76–88

Assign-to-the-longest-available-token

C. Comte. "Dynamic load balancing with tokens". *Computer Communications* 144 (Aug. 15, 2019), pp. 76–88

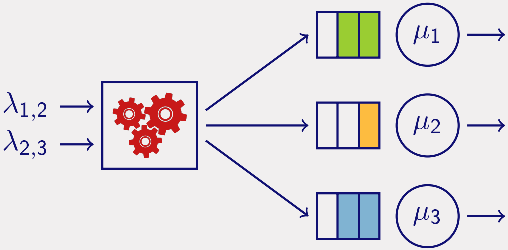
Model

- Arrival rates $\lambda_{1,2}$ and $\lambda_{2,3}$
- Service rates μ_1, μ_2, and μ_3
- Queue length ℓ_1, ℓ_2, and ℓ_3



Assign-to-the-longest-available-token

C. Comte. "Dynamic load balancing with tokens". *Computer Communications* 144 (Aug. 15, 2019), pp. 76–88



Model

- Arrival rates $\lambda_{1,2}$ and $\lambda_{2,3}$
- Service rates μ_1, μ_2, and μ_3
- Queue length ℓ_1, ℓ_2, and ℓ_3

Load-balancing algorithm

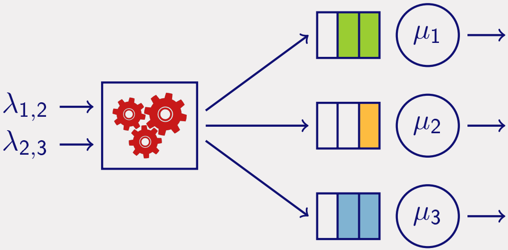
- Each server sends a token to the dispatcher upon a service completion
- Assign incoming job to the server identified by the longest available token

Assign-to-the-longest-available-token

Contributions

- Product-form stationary distribution:

$$\pi(c_1, \ldots, c_n) \propto \prod_{p=1}^{n} \frac{\mu_{c_p}}{\lambda(\{c_1, \ldots, c_p\})}$$



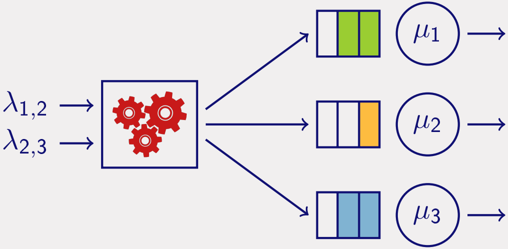
Assign-to-the-longest-available-token

Contributions

- Product-form stationary distribution:

$$\pi(c_1, \ldots, c_n) \propto \prod_{p=1}^{n} \frac{\mu_{c_p}}{\lambda(\{c_1, \ldots, c_p\})}$$

- Performance is insensitive to the job size distribution if the scheduling policy is processor-sharing



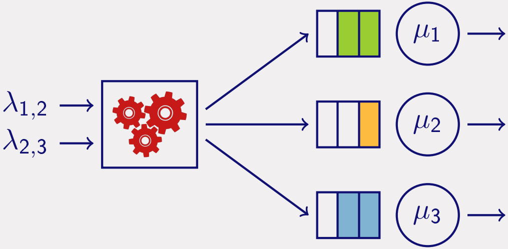
Assign-to-the-longest-available-token

Contributions

- Product-form stationary distribution:

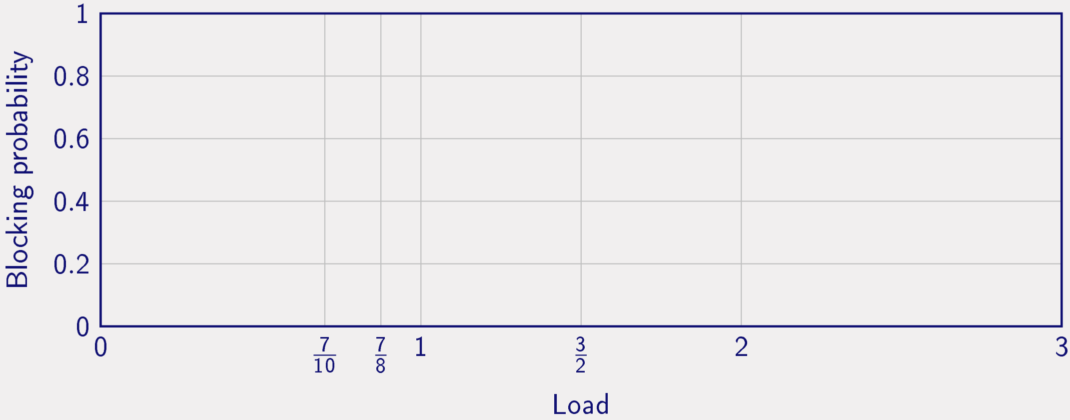
$$\pi(c_1, \dots, c_n) \propto \prod_{p=1}^{n} \frac{\mu_{c_p}}{\lambda(\{c_1, \dots, c_p\})}$$

- Performance is insensitive to the job size distribution if the scheduling policy is processor-sharing
- Connection with existing queueing models and algorithms: order-independent queues, Whittle networks, ...



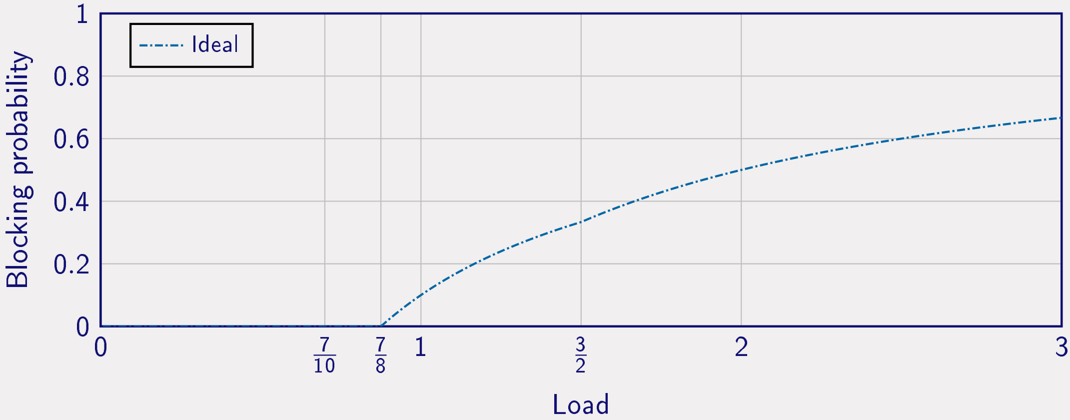
Numerical results

$\mu_1 = \mu_2 = \ldots = \mu_{10} = 1 \qquad \ell_1 = \ell_2 = \ldots = \ell_{10} = 6 \qquad \lambda_{1,2,\ldots,7} = 4\lambda_{4,5,\ldots,10}$



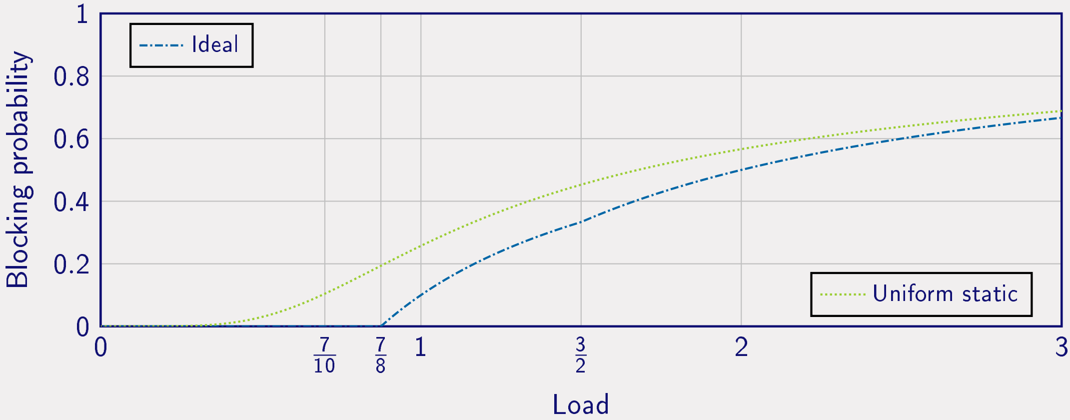
Numerical results

$\mu_1 = \mu_2 = \ldots = \mu_{10} = 1 \qquad \ell_1 = \ell_2 = \ldots = \ell_{10} = 6 \qquad \lambda_{1,2,\ldots,7} = 4\lambda_{4,5,\ldots,10}$



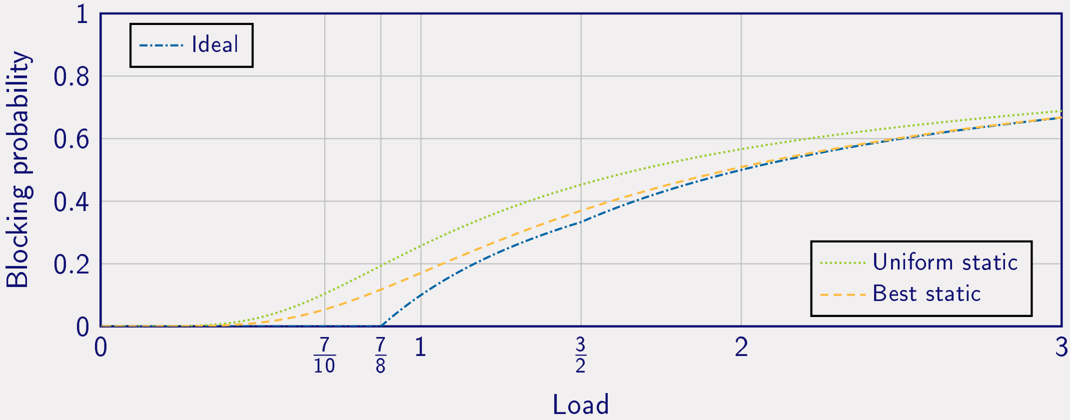
Numerical results

$\mu_1 = \mu_2 = \ldots = \mu_{10} = 1 \qquad \ell_1 = \ell_2 = \ldots = \ell_{10} = 6 \qquad \lambda_{1,2,\ldots,7} = 4\lambda_{4,5,\ldots,10}$



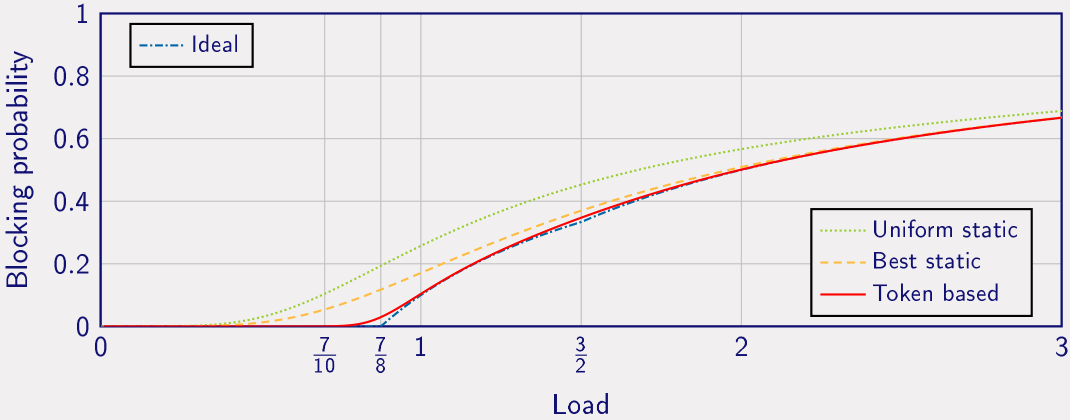
Numerical results

$\mu_1 = \mu_2 = \ldots = \mu_{10} = 1 \qquad \ell_1 = \ell_2 = \ldots = \ell_{10} = 6 \qquad \lambda_{1,2,\ldots,7} = 4\lambda_{4,5,\ldots,10}$



Numerical results

$\mu_1 = \mu_2 = \ldots = \mu_{10} = 1 \qquad \ell_1 = \ell_2 = \ldots = \ell_{10} = 6 \qquad \lambda_{1,2,\ldots,7} = 4\lambda_{4,5,\ldots,10}$



Outline

Model and contributions

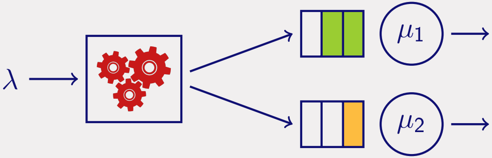
Based on a joint work with Mark van der Boor that will be presented at IWQoS 2021.

Model and contributions

Based on a joint work with Mark van der Boor that will be presented at IWQoS 2021.

Model

- Arrival rate λ
- Service rates μ_1 and μ_2
- Queue lengths ℓ_1 and ℓ_2 with $\ell_1 + \ell_2 = L$

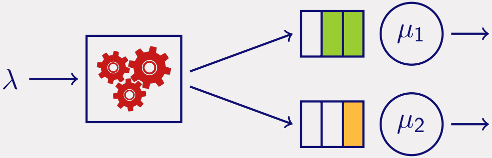


Model and contributions

Based on a joint work with Mark van der Boor that will be presented at IWQoS 2021.

Model

- Arrival rate λ
- Service rates μ_1 and μ_2
- Queue lengths ℓ_1 and ℓ_2 with $\ell_1 + \ell_2 = L$



Load-balancing algorithm

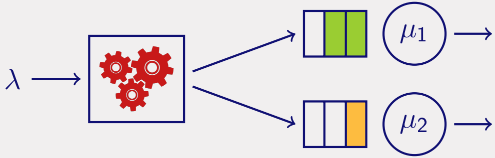
- Each server sends a token to the dispatcher upon a service completion
- Assign incoming job to the server identified by the longest available token

Model and contributions

Based on a joint work with Mark van der Boor that will be presented at IWQoS 2021.

Model

- Arrival rate λ
- Service rates μ_1 and μ_2
- Queue lengths ℓ_1 and ℓ_2 with $\ell_1 + \ell_2 = L$



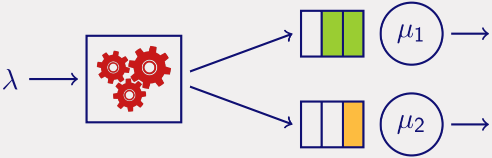
Load-balancing algorithm

- Each server sends a token to the dispatcher upon a service completion
- Assign incoming job to the server identified by the longest available token

Question: Which queue length minimizes the blocking probability?

Optimal queue lengths

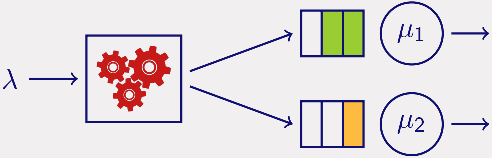
Theorem 1: There is $\lambda_* > 0$ such that, for $\lambda \leq \lambda_*$, the blocking probability is minimized when $\frac{\ell_1}{L} \simeq \frac{\mu_1}{\mu_1+\mu_2}$ and $\frac{\ell_2}{L} \simeq \frac{\mu_2}{\mu_1+\mu_2}$.



Optimal queue lengths

Theorem 1: There is $\lambda_* > 0$ such that, for $\lambda \leq \lambda_*$, the blocking probability is minimized when $\frac{\ell_1}{L} \simeq \frac{\mu_1}{\mu_1+\mu_2}$ and $\frac{\ell_2}{L} \simeq \frac{\mu_2}{\mu_1+\mu_2}$.

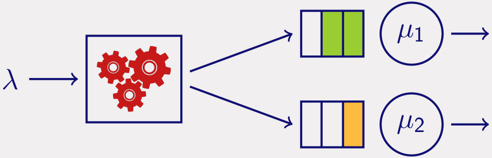
Theorem 2: There is $\lambda^* > 0$ such that, for $\lambda \geq \lambda^*$, the blocking probability is minimized when $\frac{\ell_1}{L} \simeq \frac{1}{2}$ and $\frac{\ell_2}{L} \simeq \frac{1}{2}$.



Optimal queue lengths

Theorem 1: There is $\lambda_* > 0$ such that, for $\lambda \leq \lambda_*$, the blocking probability is minimized when $\frac{\ell_1}{L} \simeq \frac{\mu_1}{\mu_1+\mu_2}$ and $\frac{\ell_2}{L} \simeq \frac{\mu_2}{\mu_1+\mu_2}$.

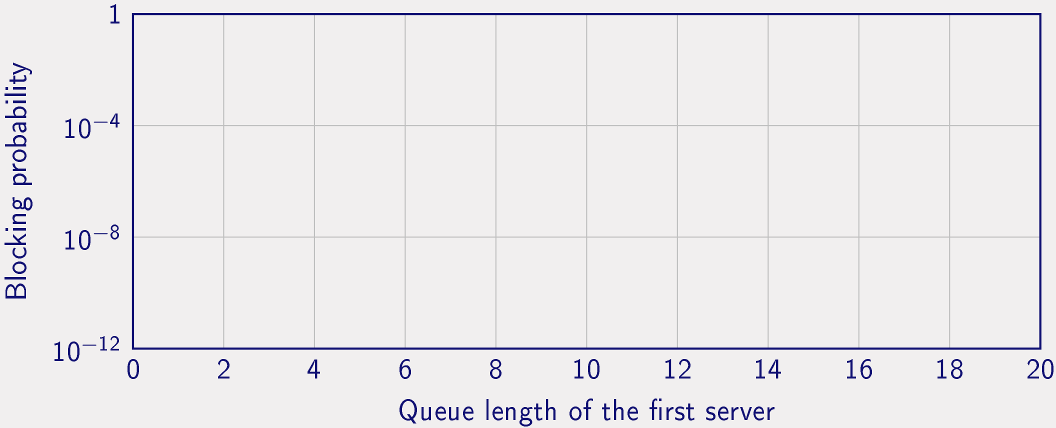
Theorem 2: There is $\lambda^* > 0$ such that, for $\lambda \geq \lambda^*$, the blocking probability is minimized when $\frac{\ell_1}{L} \simeq \frac{1}{2}$ and $\frac{\ell_2}{L} \simeq \frac{1}{2}$.



Theorem 3: The optimal queue length of the fastest server, in terms of the blocking probability, is decreasing with the arrival rate λ.

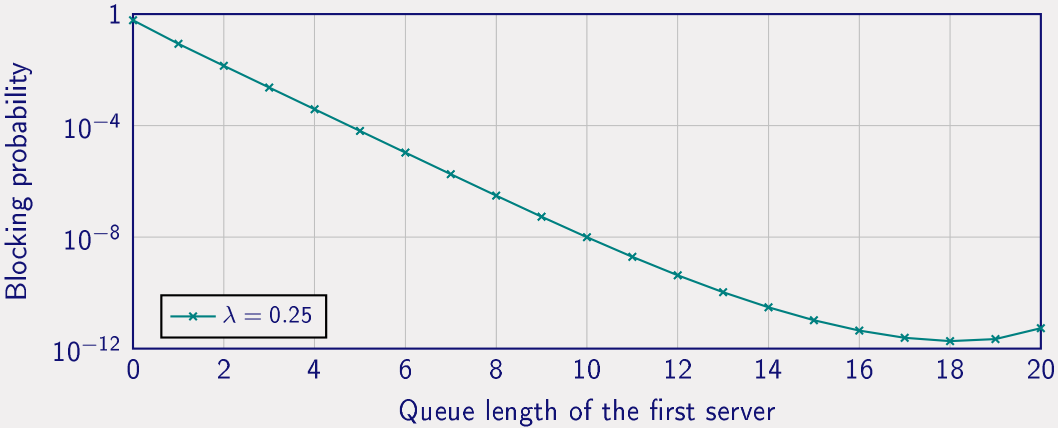
Numerical results

$L = 20$ $\mu_1 = 0.9$ $\mu_2 = 0.1$



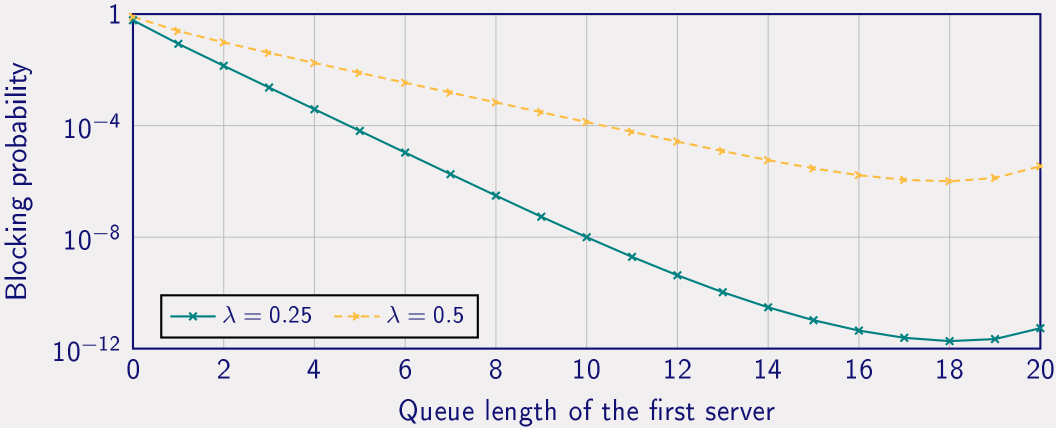
Numerical results

$L = 20 \qquad \mu_1 = 0.9 \qquad \mu_2 = 0.1$



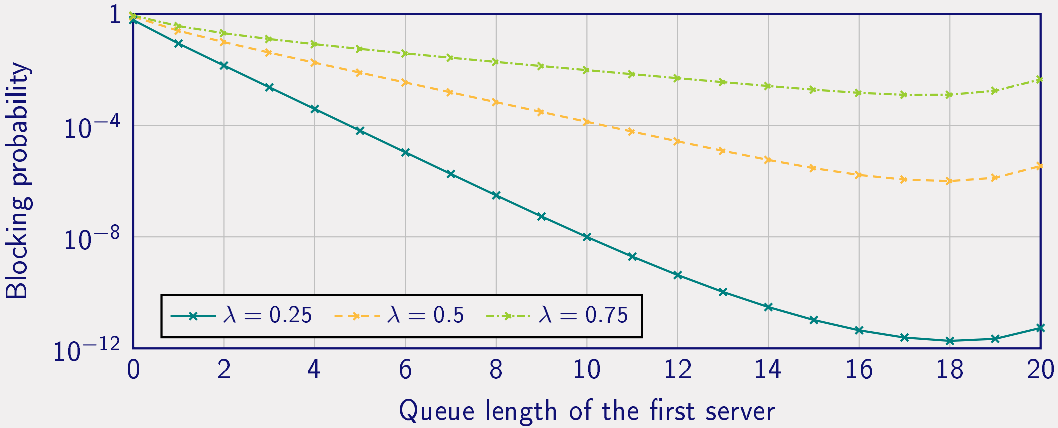
Numerical results

$L = 20 \qquad \mu_1 = 0.9 \qquad \mu_2 = 0.1$



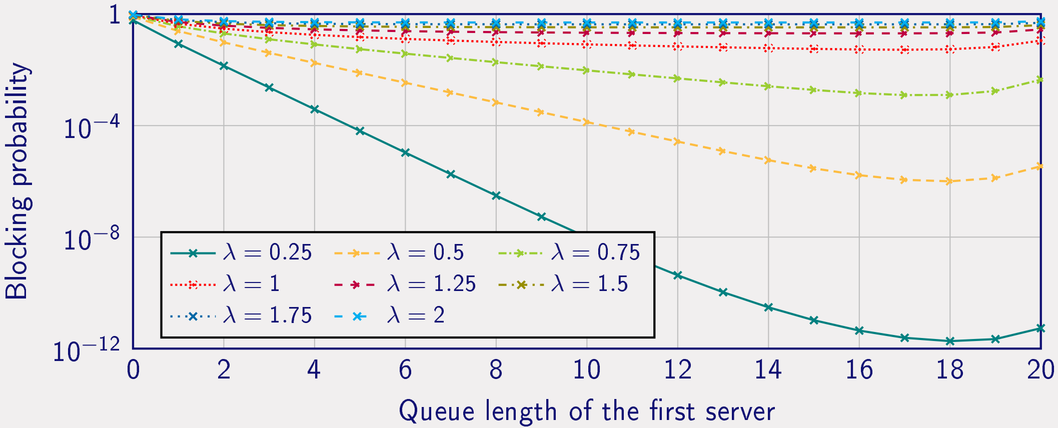
Numerical results

$L = 20 \qquad \mu_1 = 0.9 \qquad \mu_2 = 0.1$



Numerical results

$L = 20 \qquad \mu_1 = 0.9 \qquad \mu_2 = 0.1$



Conclusion

Take-away messages

- Token-based algorithms offer a scalable method to deal with assignment constraints.

Conclusion

Take-away messages

- Token-based algorithms offer a scalable method to deal with assignment constraints.
- ALIS and its generalizations are mathematically tractable in finite systems, and they give insights into possible generalizations of JIQ.

Conclusion

Take-away messages

- Token-based algorithms offer a scalable method to deal with assignment constraints.
- ALIS and its generalizations are mathematically tractable in finite systems, and they give insights into possible generalizations of JIQ.
- ALIS and its generalization are insensitive to the job size distribution.

Conclusion

Take-away messages

- Token-based algorithms offer a scalable method to deal with assignment constraints.
- ALIS and its generalizations are mathematically tractable in finite systems, and they give insights into possible generalizations of JIQ.
- ALIS and its generalization are insensitive to the job size distribution.

Research questions

- Generalize blocking (insensitive) variants.

Conclusion

Take-away messages

- Token-based algorithms offer a scalable method to deal with assignment constraints.
- ALIS and its generalizations are mathematically tractable in finite systems, and they give insights into possible generalizations of JIQ.
- ALIS and its generalization are insensitive to the job size distribution.

Research questions

- Generalize blocking (insensitive) variants.
- Formalize the relation between JIQ and ALIS.