Balanced Splitting: A Framework for Achieving Zero-wait in the Multiserver-job Model

Josu Doncel
University of the Basque Country, UPV/EHU.

Joint work with J. Anselmi (Inria Grenoble)

INFORMS Applied Probability Society Conference. Atlanta, USA.
June 30, 2025

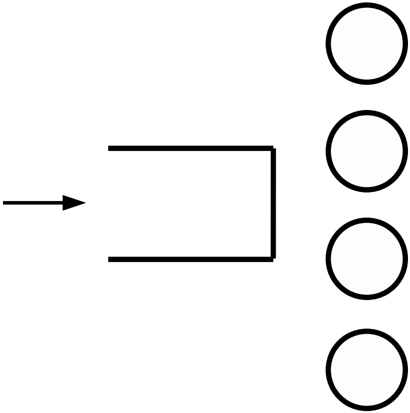
The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

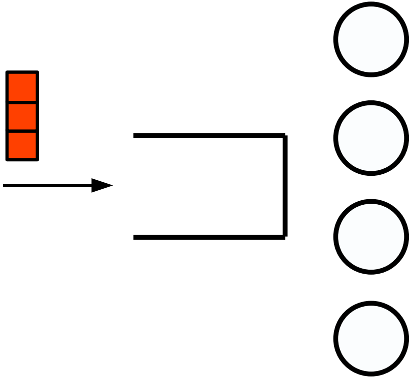
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

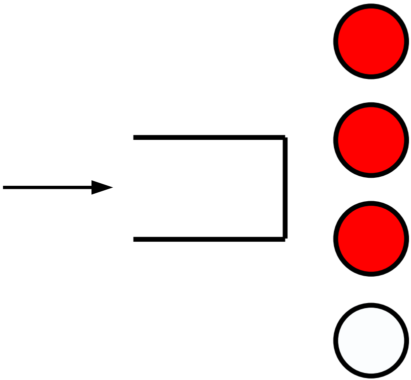
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

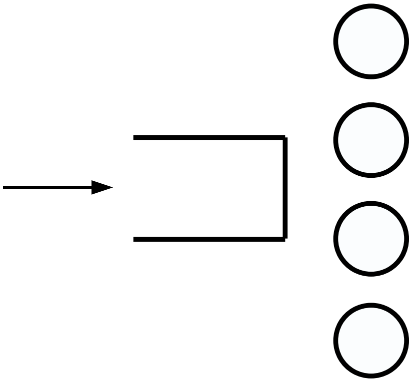
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

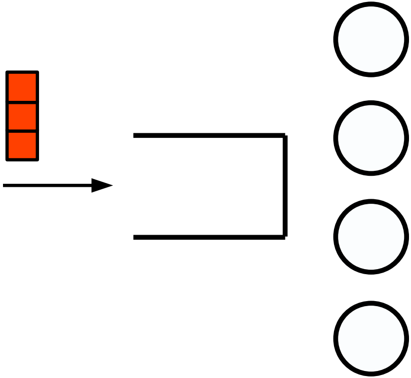
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

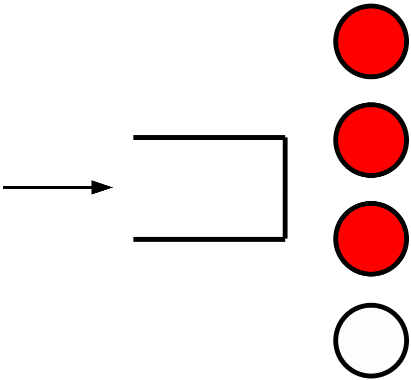
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

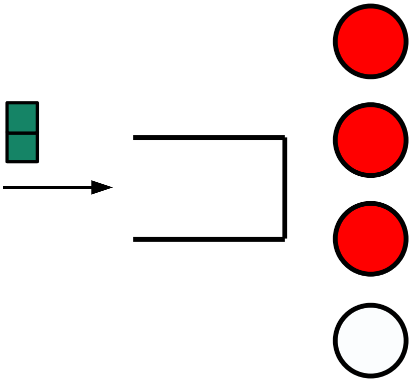
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

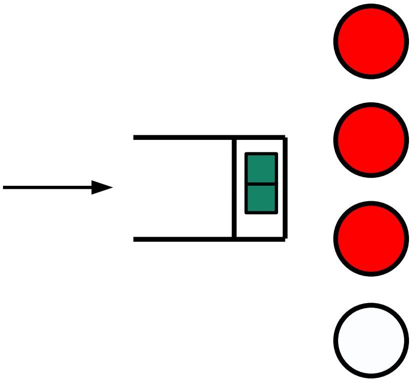
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

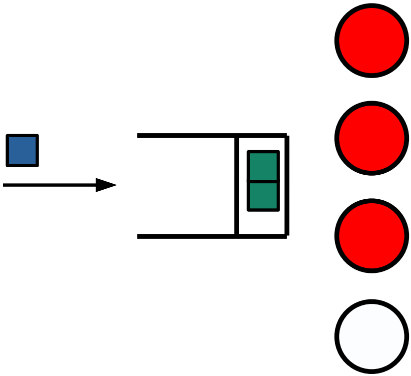
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

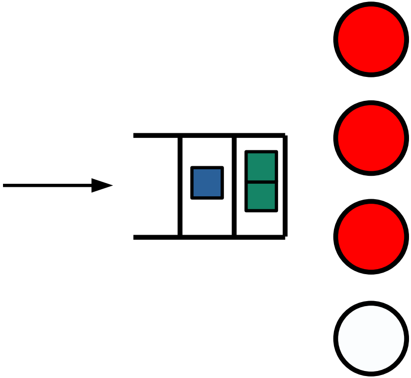
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

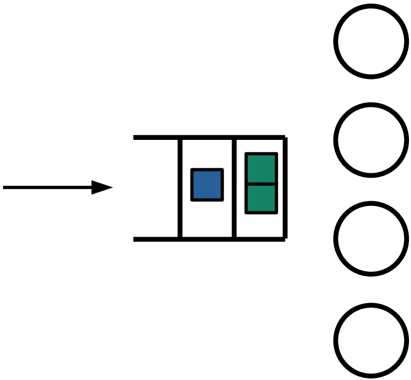
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

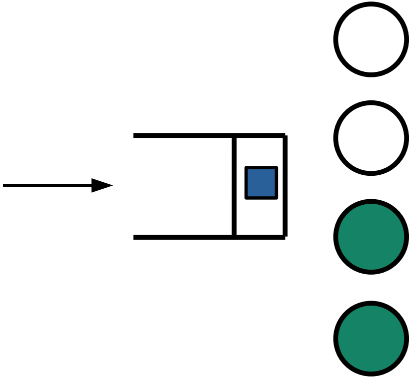
Example with FCFS:



The multiserver-job model

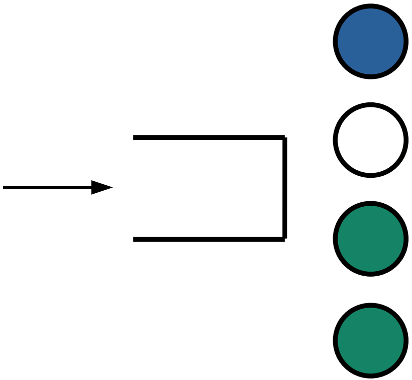
Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

Example with FCFS:



The multiserver-job model

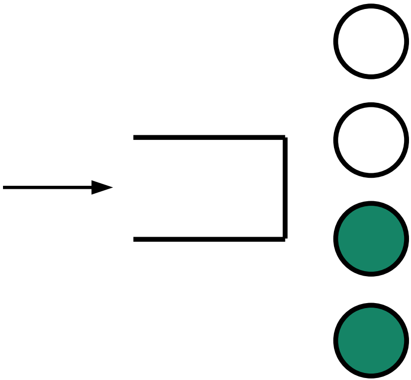
Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

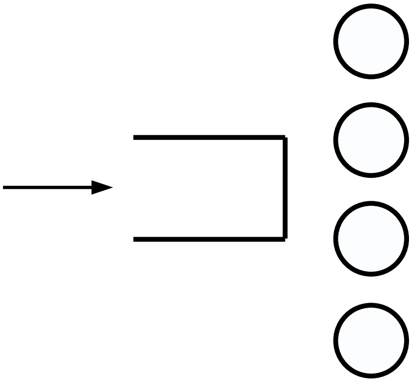
Example with FCFS:



The multiserver-job model

Each job can occupy **simultaneously** multiple servers during the execution of the job (server need)

Example with FCFS:



The importance of the multiserver-job model

Analysis of the traces of Google's Borg Scheduler[1]

Tasks require a specific number of server, which can vary by five orders of magnitude across jobs

[1] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: the next generation," Proceedings of the fifteenth European conference on computer systems, 2020.

The importance of the multiserver-job model

Analysis of the traces of Google's Borg Scheduler[1]

Tasks require a specific number of server, which can vary by five orders of magnitude across jobs

Many works study the MSJ model since then:

M. Harchol-Balter, "The multiserver job queueing model," Queueing Syst. Theory Appl.

I. Grosof, Z. Scully, M. Harchol-Balter, and A. Scheller-Wolf, "Optimal scheduling in the multiserver-job model under heavy traffic," Proc. ACM Meas. Anal. Comput. Syst., vol. 6, no. 3, dec 2022

W. Wang, Q. Xie, and M. Harchol-Balter, "Zero queueing for multiserver jobs," Proc. ACM Meas. Anal. Comput. Syst., vol. 5, no. 1, feb 2021.

Z. Chen, I. Grosof, and B. Berg, "Analyzing Practical Policies for Multiresource Job Scheduling" Accepted to ACM SIGMETRICS, June 2025.

I. Grosof, Y. Hong, and M. Harchol-Balter, "The RESET and MARC Techniques, with Application to Multiserver-Job Analysis" IFIP Performance, November 2023.

...

[1] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: the next generation," Proceedings of the fifteenth European conference on computer systems, 2020.

Fundamental Questions

Investigate scheduling policies

(I1) throughput optimal

(I2) zero-wait property

Fundamental Questions

Investigate scheduling policies

(I1) throughput optimal

(I2) zero-wait property

(I3) characterize mean response time of jobs

(I4) minimize mean response time of jobs

Fundamental Questions

Investigate scheduling policies

(I1) throughput optimal

(I2) zero-wait property

(I3) characterize mean response time of jobs

(I4) minimize mean response time of jobs

Optimality results with some limitations:

(L1) server needs and number of servers: powers of 2

(L2) preemptive scheduling policies

Fundamental Questions

Investigate scheduling policies

(I1) throughput optimal

(I2) zero-wait property

(I3) characterize mean response time of jobs

(I4) minimize mean response time of jobs

Optimality results with some limitations:

(L1) server needs and number of servers: powers of 2

(L2) preemptive scheduling policies

Our Approach: BalancedSplitting-π

We prove that, without (L1) and (L2), it verifies (I1), (I2) and (I3) asymptotically

Fundamental Questions

Investigate scheduling policies

(I1) throughput optimal

(I2) zero-wait property

(I3) characterize mean response time of jobs

(I4) minimize mean response time of jobs

Optimality results with some limitations:

(L1) server needs and number of servers: powers of 2

(L2) preemptive scheduling policies

Our Approach: BalancedSplitting-π

We prove that, without (L1) and (L2), it verifies (I1), (I2) and (I3) asymptotically
Numerical analysis for (I4)

Outline

Outline

Model Description

k servers and arrival rate λ

Model Description

k servers and arrival rate λ

C classes of jobs

- α_i: prob that a job is of class i
- d_i: mean service time of class-i jobs
- n_i: server need of class-i jobs

Model Description

k servers and arrival rate λ

C classes of jobs

- α_i: prob that a job is of class i
- d_i: mean service time of class-i jobs
- n_i: server need of class-i jobs

BalancedSplitting-π

The set of servers is partitioned in $C+1$ sets:
- $\mathcal{A}_i$: set of servers dedicated to class-i jobs
- $\mathcal{H}$: helper set (all types of classes)

Model Description

k servers and arrival rate λ

C classes of jobs

- α_i: prob that a job is of class i
- d_i: mean service time of class-i jobs
- n_i: server need of class-i jobs

BalancedSplitting-π

The set of servers is partitioned in $C+1$ sets:

- $\mathcal{A}_i$: set of servers dedicated to class-i jobs
- $\mathcal{H}$: helper set (all types of classes)

Observation: $|\mathcal{A}_i| = c_i\ n_i$, for $c_i \in \mathbb{N}$, and $|\mathcal{H}| \geq \max_i n_i$

Model Description

k servers and arrival rate λ

C classes of jobs

- α_i: prob that a job is of class i
- d_i: mean service time of class-i jobs
- n_i: server need of class-i jobs

BalancedSplitting-π

The set of servers is partitioned in $C+1$ sets:
- $\mathcal{A}_i$: set of servers dedicated to class-i jobs
- $\mathcal{H}$: helper set (all types of classes)

Observation: $|\mathcal{A}_i| = c_i\ n_i$, for $c_i \in \mathbb{N}$, and $|\mathcal{H}| \geq \max_i n_i$

How to partition the set of servers?

Balanced manner according to $\alpha_i n_i d_i$.

BalancedSplitting-π

How to partition the set of servers?

Balanced manner according to $\alpha_i n_i d_i$.

BalancedSplitting-π

How to partition the set of servers?

Balanced manner according to $\alpha_i n_i d_i$.

$$|\mathcal{A}_i| = n_i \left\lfloor \psi \frac{k}{n_i} \frac{\alpha_i n_i d_i}{\sum_i \alpha_i n_i d_i} \right\rfloor, \qquad |\mathcal{H}| = k - \sum_i |\mathcal{A}_i|$$

where ψ gets the maximum value between 0 and 1 such that $|\mathcal{H}| \geq \max_i n_i$

BalancedSplitting-π

How to partition the set of servers?

Balanced manner according to $\alpha_i n_i d_i$.

$$|\mathcal{A}_i| = n_i \left\lfloor \psi \frac{k}{n_i} \frac{\alpha_i n_i d_i}{\sum_i \alpha_i n_i d_i} \right\rfloor, \qquad |\mathcal{H}| = k - \sum_i |\mathcal{A}_i|$$

where ψ gets the maximum value between 0 and 1 such that $|\mathcal{H}| \geq \max_i n_i$

How does BalancedSplitting work?

BalancedSplitting-π

How to partition the set of servers?

Balanced manner according to $\alpha_i n_i d_i$.

$$|\mathcal{A}_i| = n_i \left\lfloor \psi \frac{k}{n_i} \frac{\alpha_i n_i d_i}{\sum_i \alpha_i n_i d_i} \right\rfloor, \qquad |\mathcal{H}| = k - \sum_i |\mathcal{A}_i|$$

where ψ gets the maximum value between 0 and 1 such that $|\mathcal{H}| \geq \max_i n_i$

How does BalancedSplitting work?

- Incoming class-i jobs are sent to $\mathcal{A}_i$ if there are enough idle servers; otherwise to $\mathcal{H}$

BalancedSplitting-π

How to partition the set of servers?

Balanced manner according to $\alpha_i n_i d_i$.

$$|\mathcal{A}_i| = n_i \left\lfloor \psi \frac{k}{n_i} \frac{\alpha_i n_i d_i}{\sum_i \alpha_i n_i d_i} \right\rfloor, \qquad |\mathcal{H}| = k - \sum_i |\mathcal{A}_i|$$

where ψ gets the maximum value between 0 and 1 such that $|\mathcal{H}| \geq \max_i n_i$

How does BalancedSplitting work?

- Incoming class-i jobs are sent to $\mathcal{A}_i$ if there are enough idle servers; otherwise to $\mathcal{H}$
- $\mathcal{H}$ processes jobs according to a non-preemptive scheduling π

BalancedSplitting-π

How to partition the set of servers?

Balanced manner according to $\alpha_i n_i d_i$.

$$|\mathcal{A}_i| = n_i \left\lfloor \psi \frac{k}{n_i} \frac{\alpha_i n_i d_i}{\sum_i \alpha_i n_i d_i} \right\rfloor, \qquad |\mathcal{H}| = k - \sum_i |\mathcal{A}_i|$$

where ψ gets the maximum value between 0 and 1 such that $|\mathcal{H}| \geq \max_i n_i$

How does BalancedSplitting work?

- Incoming class-i jobs are sent to $\mathcal{A}_i$ if there are enough idle servers; otherwise to $\mathcal{H}$
- $\mathcal{H}$ processes jobs according to a non-preemptive scheduling π
- When a class-i job ends service in $\mathcal{A}_i$, the oldest class-i job waiting in $\mathcal{H}$ starts being served in $\mathcal{A}_i$

BalancedSplitting-π

How to partition the set of servers?

Balanced manner according to $\alpha_i n_i d_i$.

$$|\mathcal{A}_i| = n_i \left\lfloor \psi \frac{k}{n_i} \frac{\alpha_i n_i d_i}{\sum_i \alpha_i n_i d_i} \right\rfloor , \qquad |\mathcal{H}| = k - \sum_i |\mathcal{A}_i|$$

where ψ gets the maximum value between 0 and 1 such that $|\mathcal{H}| \geq \max_i n_i$

How does BalancedSplitting work?

- Incoming class-i jobs are sent to $\mathcal{A}_i$ if there are enough idle servers; otherwise to $\mathcal{H}$
- $\mathcal{H}$ processes jobs according to a non-preemptive scheduling π
- When a class-i job ends service in $\mathcal{A}_i$, the oldest class-i job waiting in $\mathcal{H}$ starts being served in $\mathcal{A}_i$

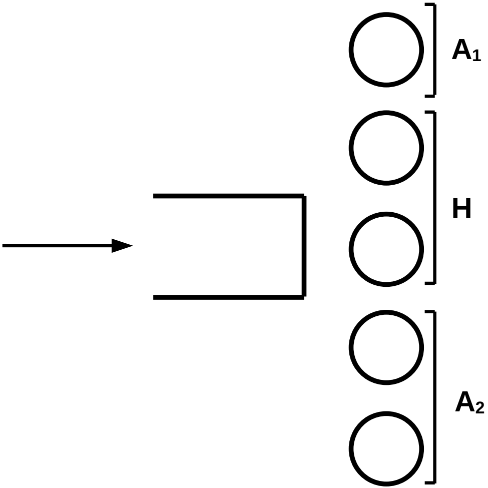
Observation: If the probability of sending jobs to $\mathcal{H}$ is zero, zero-wait property

Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.

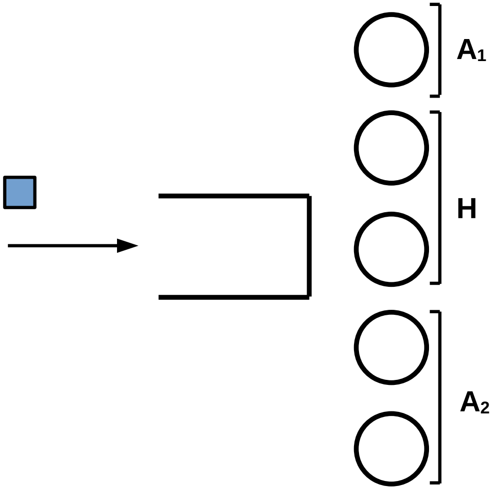
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



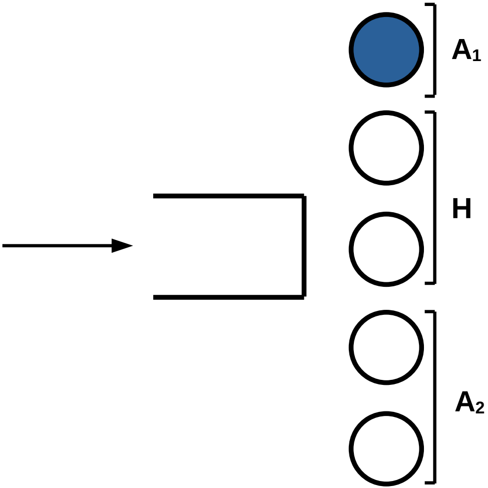
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



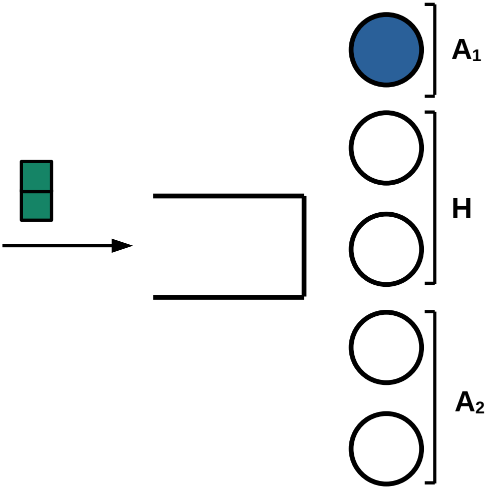
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



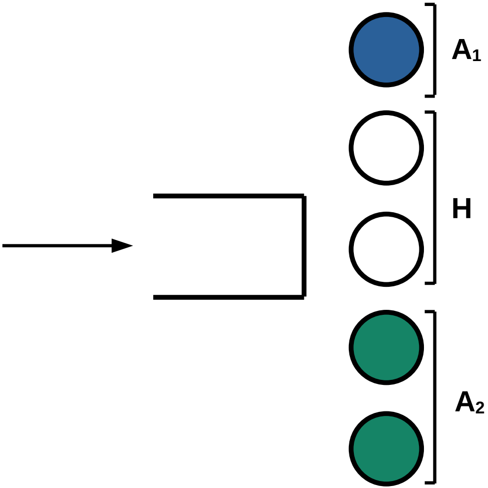
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



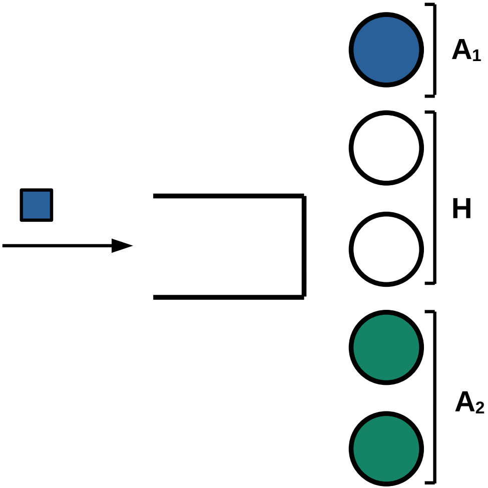
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



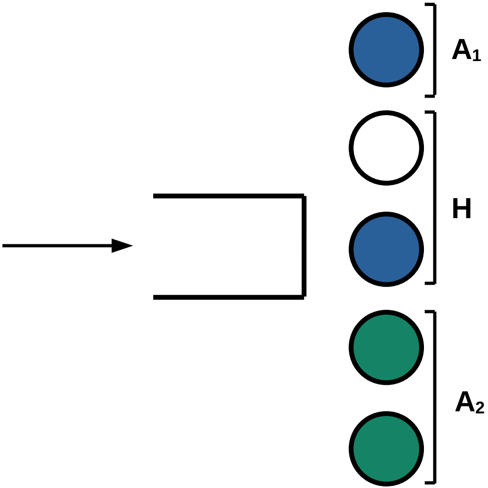
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



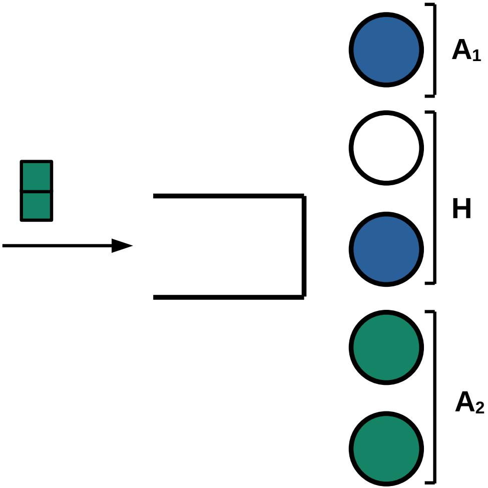
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



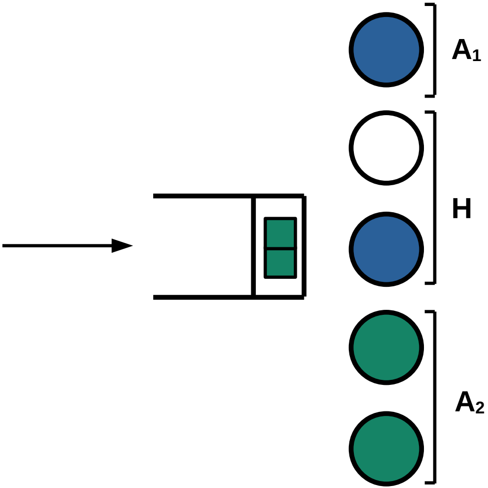
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



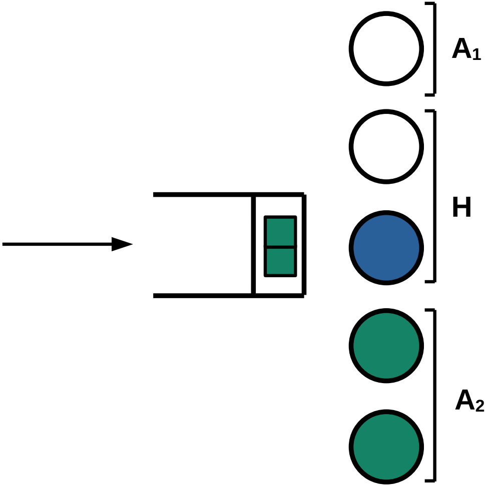
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



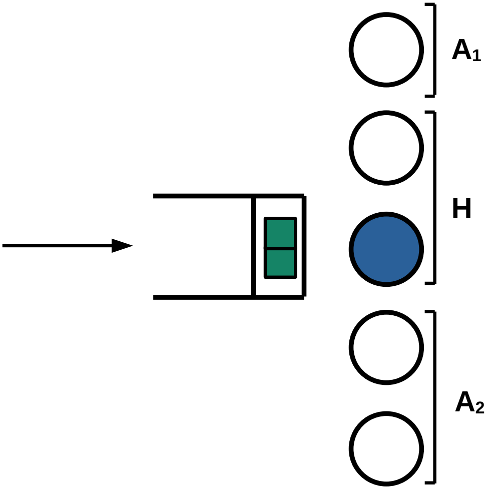
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



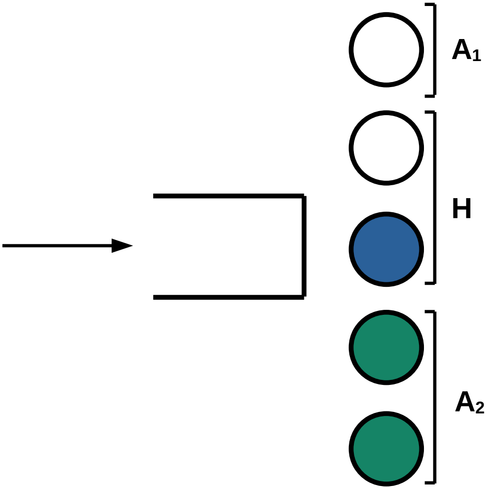
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



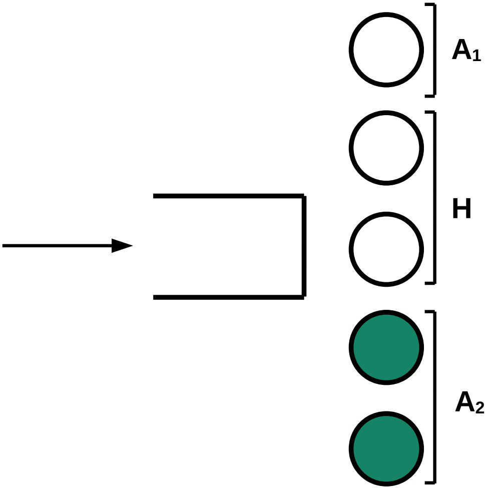
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



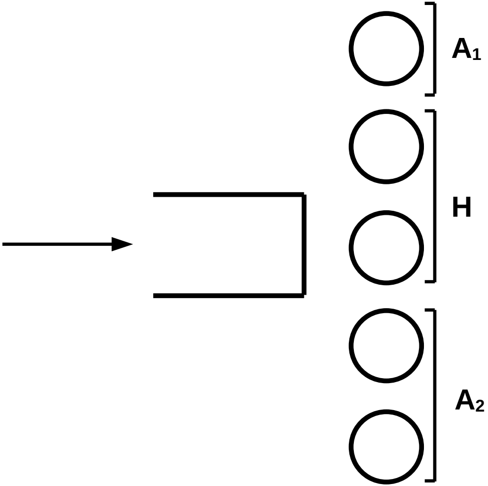
Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



Example

Two classes, 5 servers and $n_1 = 1, n_2 = 2$.



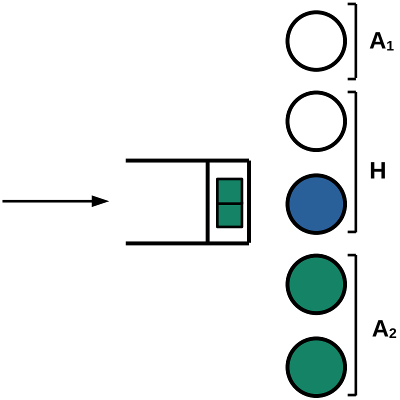
Outline

Throughput Optimality

Throughput Optimality

BalancedSplitting-π

Waste of capacity occurs $\Rightarrow$ Not throughput optimal in general!



Throughput Optimality

Throughput Optimality

Interesting property

Under BalancedSplitting-π, the set of servers dedicated to class-i jobs are $M/G/|\mathcal{A}_i|/|\mathcal{A}_i|$ queues
$\Rightarrow$ Erlang's loss formula!

Throughput Optimality

Interesting property

Under BalancedSplitting-π, the set of servers dedicated to class-i jobs are $M/G/|\mathcal{A}_i|/|\mathcal{A}_i|$ queues
$\Rightarrow$ Erlang's loss formula!

Stability

We only need to study $\mathcal{H}$.

Stability

We only need to study $\mathcal{H}$.
Let $E_{|\mathcal{A}_i|}(\lambda \alpha_i d_i)$ be the probability that a class-i job is sent to $\mathcal{H}$

Sufficient condition for stability

Stability

We only need to study $\mathcal{H}$.
Let $E_{|\mathcal{A}_i|}(\lambda \alpha_i d_i)$ be the probability that a class-i job is sent to $\mathcal{H}$

Sufficient condition for stability

$$\frac{\lambda}{|\mathcal{H}|} \sum_{i=1}^{C} \alpha_i n_i d_i E_{|\mathcal{A}_i|}(\lambda \alpha_i d_i) < 1 \qquad \text{(STAB-COND)}$$

Sketch of the proof

(1) We study a modified version of BalancedSplitting-π (MBS-π) scheduling. The load of $\mathcal{H}$ under MBS-π is larger than the load of $\mathcal{H}$ under BalancedSplitting-π

Stability

We only need to study $\mathcal{H}$.
Let $E_{|\mathcal{A}_i|}(\lambda\alpha_i d_i)$ be the probability that a class-i job is sent to $\mathcal{H}$

Sufficient condition for stability

$$\frac{\lambda}{|\mathcal{H}|}\sum_{i=1}^{C}\alpha_i n_i d_i E_{|\mathcal{A}_i|}(\lambda\alpha_i d_i) < 1 \tag{STAB-COND}$$

Sketch of the proof

(1) We study a modified version of BalancedSplitting-π (MBS-π) scheduling.
The load of $\mathcal{H}$ under MBS-π is larger than the load of $\mathcal{H}$ under BalancedSplitting-π
(2) We show that, under MBS-π, $\mathcal{H}$ is stable iff (STAB-COND) holds.

Asymptotic Regimes

Regime 1: Arrival rate and n. of servers tend to ∞, but the load is constant.
Regime 2: Arrival rate and the servers grow to infinity and the load tends to one (Halfin-Whitt)

Asymptotic Regimes

Regime 1: Arrival rate and n. of servers tend to ∞, but the load is constant.
Regime 2: Arrival rate and the servers grow to infinity and the load tends to one (Halfin-Whitt)

Theorem

Under Regime 1 or Regime 2, the probability that a job is sent to $\mathcal{H}$ tends to 0

Consequences

Asymptotic Regimes

Regime 1: Arrival rate and n. of servers tend to ∞, but the load is constant.
Regime 2: Arrival rate and the servers grow to infinity and the load tends to one (Halfin-Whitt)

Theorem

Under Regime 1 or Regime 2, the probability that a job is sent to $\mathcal{H}$ tends to 0

Consequences

- Mean response time tends to $\sum_i \alpha_i d_i$
- BalancedSplitting is throughput optimal and has the zero-wait property (in both regimes)

Asymptotic Regimes

Regime 1: Arrival rate and n. of servers tend to ∞, but the load is constant.
Regime 2: Arrival rate and the servers grow to infinity and the load tends to one (Halfin-Whitt)

Theorem

Under Regime 1 or Regime 2, the probability that a job is sent to $\mathcal{H}$ tends to 0

Consequences

- Mean response time tends to $\sum_i \alpha_i d_i$
- BalancedSplitting is throughput optimal and has the zero-wait property (in both regimes)

Sketch of the proof

(1) We study a modified version of BalancedSplitting-π (MBS-π) scheduling. The probability that a job is sent to $\mathcal{H}$ is larger under MBS-π than under BalancedSplitting-π.

Asymptotic Regimes

Regime 1: Arrival rate and n. of servers tend to ∞, but the load is constant.
Regime 2: Arrival rate and the servers grow to infinity and the load tends to one (Halfin-Whitt)

Theorem

Under Regime 1 or Regime 2, the probability that a job is sent to $\mathcal{H}$ tends to 0

Consequences

- Mean response time tends to $\sum_i \alpha_i d_i$
- BalancedSplitting is throughput optimal and has the zero-wait property (in both regimes)

Sketch of the proof

(1) We study a modified version of BalancedSplitting-π (MBS-π) scheduling. The probability that a job is sent to $\mathcal{H}$ is larger under MBS-π than under BalancedSplitting-π.
(2) We prove that, in both asymptotic regimes, the blocking probability of $\mathcal{A}_i$ tends to zero under MBS-π.

What is MBS-π?

What is MBS-π?

BalancedSplitting-π

- Incoming class-i jobs are sent to $\mathcal{A}_i$ if there are enough idle servers; otherwise to $\mathcal{H}$
- $\mathcal{H}$ processes jobs according to a non-preemptive scheduling π
- When a class-i job ends service, the oldest class-i job waiting in $\mathcal{H}$ starts being served in $\mathcal{A}_i$

What is MBS-π?

BalancedSplitting-π

- Incoming class-i jobs are sent to $\mathcal{A}_i$ if there are enough idle servers; otherwise to $\mathcal{H}$
- $\mathcal{H}$ processes jobs according to a non-preemptive scheduling π
- When a class-i job ends service, the oldest class-i job waiting in $\mathcal{H}$ starts being served in $\mathcal{A}_i$

MBS-π

- Incoming class-i jobs are sent to $\mathcal{A}_i$ if there are enough idle servers; otherwise to $\mathcal{H}$
- $\mathcal{H}$ processes jobs according to a non-preemptive scheduling π
- A job that is sent to $\mathcal{H}$ is executed in $\mathcal{H}$

Outline

Numerical Analysis

We consider traces from HPC systems [2]

[2] http://www.cs.huji.ac.il/labs/parallel/workload/logs.html

We consider traces from HPC systems [2]

We extracted the parameters of SDSC and of Karlsruhe Institute of Technology (KIT) System and simulated mean response times of different scheduling policies

[2] http://www.cs.huji.ac.il/labs/parallel/workload/logs.html

Numerical Analysis

We consider traces from HPC systems [2]

We extracted the parameters of SDSC and of Karlsruhe Institute of Technology (KIT) System and simulated mean response times of different scheduling policies

Restricted to Powers of two

The scheduling policy that minimizes mean response times is ServerFilling-SRPT
$\Rightarrow$ Is BalancedSplitting-π close to be optimal?

[2] http://www.cs.huji.ac.il/labs/parallel/workload/logs.html

Numerical Analysis

We consider traces from HPC systems [2]

We extracted the parameters of SDSC and of Karlsruhe Institute of Technology (KIT) System and simulated mean response times of different scheduling policies

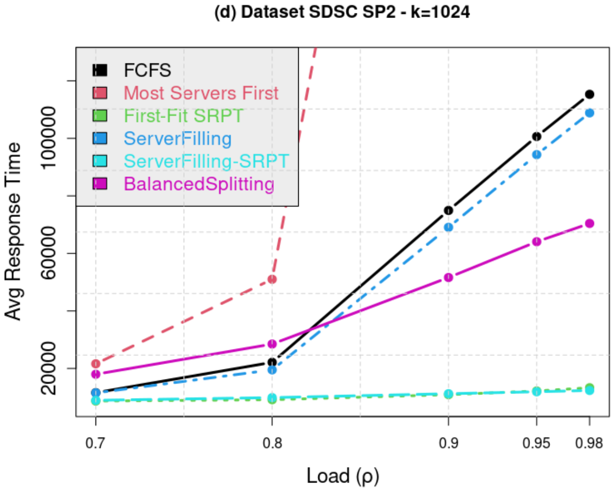
Restricted to Powers of two

The scheduling policy that minimizes mean response times is ServerFilling-SRPT
$\Rightarrow$ Is BalancedSplitting-π close to be optimal?

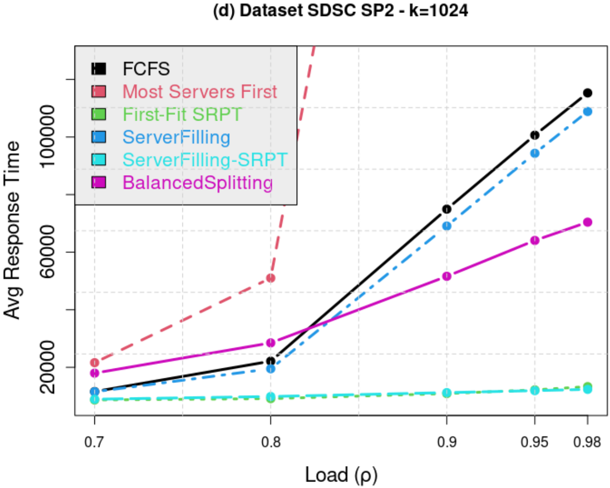
We consider as π the FCFS scheduling

[2] http://www.cs.huji.ac.il/labs/parallel/workload/logs.html

Real Data from SDSC

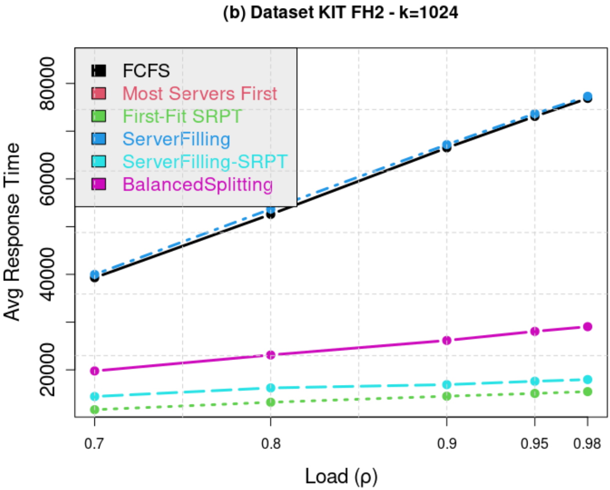


Real Data from SDSC

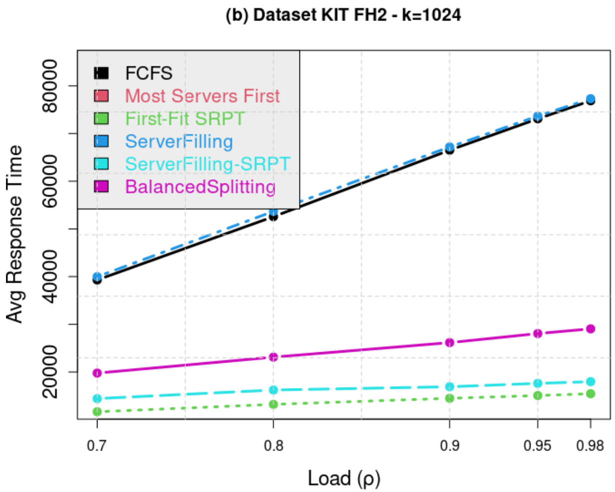


BalacedSplitting outperforms ServerFilling (preemptive)!

Real Data from KIT



Real Data from KIT



BalacedSplitting is close to optimal!

Outline

Conclusions and Future Work

Investigate scheduling policies

(I1) throughput optimal

(I2) zero-wait property

(I3) characterize mean response time of jobs

(I4) minimize mean response time of jobs

Optimality results with some limitations:

(L1) server needs and number of servers: powers of 2

(L2) preemptive scheduling policies

Our Approach: BalancedSplitting-π

We prove that, without (L1) and (L2), it verifies (I1), (I2) and (I3) asymptotically
Numerical analysis for (I4)

Conclusions and Future Work

Investigate scheduling policies

(I1) throughput optimal

(I2) zero-wait property

(I3) characterize mean response time of jobs

(I4) minimize mean response time of jobs

Optimality results with some limitations:

(L1) server needs and number of servers: powers of 2

(L2) preemptive scheduling policies

Our Approach: BalancedSplitting-π

We prove that, without (L1) and (L2), it verifies (I1), (I2) and (I3) asymptotically
Numerical analysis for (I4)

Future Work

- Balanced Size-Aware Dispatching: job size knowledge is required
- Parallel Servers Systems

Thank you very much

Thanks for your attention. Questions?

J. Anselmi, J. Doncel. "Balanced Splitting: A Framework for Achieving Zero-wait in the Multiserver-job Model" IEEE Transactions on Parallel and Distributed Systems, Vol 36, Issue 1, 2025

Details of Both Asymptotic Regimes

Let $f_k = o(k), f_k \in \mathbb{N}$

Regime 1: $k \to \infty$

$\lambda^{(k)} = \lambda \frac{k}{f_k}$
$n_i^{(k)} = n_i f_k$
$\alpha_i^{(k)} = \alpha_i$
$d_i^{(k)} = d_i$

Details of Both Asymptotic Regimes

Let $f_k = o(k), f_k \in \mathbb{N}$

Regime 1: $k \to \infty$

$\lambda^{(k)} = \lambda \frac{k}{f_k}$
$n_i^{(k)} = n_i f_k$
$\alpha_i^{(k)} = \alpha_i$
$d_i^{(k)} = d_i$

Regime 2:

$\lambda^{(k)} \to \infty$
$(1 - \rho^{(k)})\sqrt{\frac{k}{f_k}} \to \theta$, with $\theta > 0$
$n_i^{(k)} = n_i f_k$
$\alpha_i^{(k)} = \alpha_i$
$d_i^{(k)} = d_i$

Details of Both Asymptotic Regimes

Let $f_k = o(k), f_k \in \mathbb{N}$

Regime 1: $k \to \infty$

$\lambda^{(k)} = \lambda \frac{k}{f_k}$
$n_i^{(k)} = n_i f_k$
$\alpha_i^{(k)} = \alpha_i$
$d_i^{(k)} = d_i$

Regime 2:

$\lambda^{(k)} \to \infty$
$(1 - \rho^{(k)})\sqrt{\frac{k}{f_k}} \to \theta$, with $\theta > 0$
$n_i^{(k)} = n_i f_k$
$\alpha_i^{(k)} = \alpha_i$
$d_i^{(k)} = d_i$