Networks of Multi-Server Queues with Parallel Processing

Céline Comte
Joint work with Thomas Bonald

Télécom ParisTech, France

YEQT - November 9, 2016
Introduction

Datacenters
- Specialized servers
- Massively parallel processing
- Highly variable job requirements

Examples
- Computer clusters
- Content Delivery Networks

Objective: Resource allocation with predictable performance
Background on Order Independent queues

Multi-server queues with parallel processing

Scheduling algorithm
Queue state

- Finite set of job classes \( I = \{\text{□, □}\} \)
  - Poisson external arrivals per class
  - Exponential job sizes with unit mean
- Queue state = sequence of job classes, ordered by arrival
  \[ c = (\text{□, □, □, □, □}) \]

- Total service rate in state \( c = \mu(c) \)
  \[ \mu(c) = \mu(\text{□, □}) - \mu(\text{□}) \]
Service rate

- **Normalized**: $\mu(\emptyset) = 0$ and $\mu(c) > 0$ for any state $c \neq \emptyset$
- **Non-decreasing**: $\mu(c) \leq \mu(c, i)$ for any state $c$ and class $i$
- **Order-independent**: 

\[
\begin{align*}
  c &= (\square, \square, \square) \\
  c &= (\square, \square, \square) \\
  c &= (\square, \square, \square)
\end{align*}
\]
Service rate

- **Normalized:** $\mu(\emptyset) = 0$ and $\mu(c) > 0$ for any state $c \neq \emptyset$
- **Non-decreasing:** $\mu(c) \leq \mu(c, i)$ for any state $c$ and class $i$
- **Order-independent:**

\[
\begin{align*}
\mu(c, 2) \\
\end{align*}
\]

\[
\begin{align*}
c = (\text{ }, \text{ }, \text{ }) & \quad c = (\text{ }, \text{ }, \text{ }) & \quad c = (\text{ }, \text{ }, \text{ })
\end{align*}
\]
Service rate

- **Normalized:** $\mu(\emptyset) = 0$ and $\mu(c) > 0$ for any state $c \neq \emptyset$
- **Non-decreasing:** $\mu(c) \leq \mu(c, i)$ for any state $c$ and class $i$
- **Order-independent:**

\[ c = (\text{̩}, \text{̩}, \text{̩}) \quad c = (\text{̩}, \text{̩}, \text{̩}) \quad c = (\text{̩}, \text{̩}, \text{̩}) \]
Service rate

- **Normalized**: $\mu(\emptyset) = 0$ and $\mu(c) > 0$ for any state $c \neq \emptyset$
- **Non-decreasing**: $\mu(c) \leq \mu(c, i)$ for any state $c$ and class $i$
- **Order-independent**: $c = (\square, \square, \square)$

\[ \begin{align*}
\mu(\square, 2\square) & \quad \mu(\square, \square) \\
\mu(\square) & \quad \mu(\square, \square) \\
(\square, \square, \square) & \quad (\square, \square, \square) & \quad (\square, \square, \square)
\end{align*} \]
Order Independent queues (Berezner and Krzesinski, 1996)

- A stationary measure of the state $c = (c_1, \ldots, c_n)$ is

$$\pi(c) = \pi(\emptyset) \prod_{k=1}^{n} \frac{\lambda_{c_k}}{\mu(c_1, \ldots, c_k)}, \quad \forall c \in I^*$$

- The queue is quasi-reversible
  - The current state of the queue is independent of previous departures and future arrivals
  - Arrivals and departures form independent Poisson processes
Background on Order Independent queues

Multi-server queues with parallel processing

Scheduling algorithm
Server assignment

Set $I$ of job classes

$S$ servers

$S = \{1, 2, 3\}$

Servers that can process red jobs
Server assignment

Set $I$ of job classes

Servers that can process red jobs

$S$ servers

1

2

3
Server assignment

Set $I$ of job classes

$S$ servers

$S_1 = \{1, 2\}$

$S_2 = \{2, 3\}$
Service discipline

1

2

3

Networks of Multi-Server Queues with Parallel Processing - Céline Comte
Service discipline

Networks of Multi-Server Queues with Parallel Processing - Céline Comte
Service discipline

- Parallel processing
Service discipline

- Parallel processing
- First-come first-served per server
Service discipline

- Parallel processing
- First-come first-served per server

State of the queue

$c = (\text{state of queue})$
Service discipline
Service discipline

Networks of Multi-Server Queues with Parallel Processing - Céline Comte
Service discipline
Service discipline
Service discipline
Service discipline
Service discipline
Service rate

- Reinterpretation of (Gardner et al., 2015)

- $\mu$ only depends on the set of active classes

\[
\mu(A) = \sum_{s \in \bigcup_{i \in A} S_i} \text{capacity of server } s
\]

- $\mu$ is submodular: if $A \subset B$ and $i \notin B$,

\[
\mu(A \cup \{i\}) - \mu(A) \geq \mu(B \cup \{i\}) - \mu(B)
\]
Service rate
Service rate

\[ c = (\text{\tiny \#}, \text{\tiny \#}, \text{\tiny \#}) \]
Service rate

\[ c = (\square, \square, \square) \]

\[ \mu(\{\square\}) \]
Service rate

\[ c = (\text{ }, \text{ }, \text{ }) \]

\[ \mu (\{\text{ }, \text{ }\}) \]

\[ \mu (\{\text{ }, \text{ }\}) \]
Service rate

\[ c = (\text{red}, \text{red}, \text{blue}) \]

\[ \mu (\{\text{red}\}) \]

\[ \mu (\{\text{blue}, \text{red}\}) \]
Service rate

\[ c = (\square, \square, \square) \]

\[ \mu (\{\square\}) \]

\[ \mu (\{\square, \square\}) \]
Service rate

\[ c = (\text{red}, \text{blue}, \text{red}) \]

\[ \mu(\{\text{red}\}) \]

\[ \mu(\{\text{blue}, \text{red}\}) \]
Service rate

\[ c = (\text{red}, \text{blue}, \text{red}) \]

\[ \mu (\{\text{red}\}) \]

\[ \mu (\{\text{blue}, \text{red}\}) \]
Service rate

$c = (\text{\red{\circle{}}}, \text{\blue{\circle{}}}, \text{\red{\circle{}}})$

$\mu(\{\text{\red{\circle{}}}\})$

$\mu(\{\text{\blue{\circle{}}}, \text{\red{\circle{}}}, \text{\blue{\circle{}}}\})$
Service rate

$\mu (\{\text{red}\})$  

$\mu (\{\text{blue}\})$

$c = (\text{red}, \text{blue}, \text{red})$
Service rate

\[ c = (\text{ }, \text{ }, \text{ }) \]

\[ \mu (\{\text{ }\}) \]

\[ \mu (\{\text{ }, \text{ }\}) \]
Service rate

\[ c = (\square, \square, \square) \]

\[ \mu(\{\square\}) \]

\[ \mu(\{\square\}) \quad \mu(\{\square, \square\}) \]
Service rate

$\mu(\{\text{red square}\})$  

$c = (\text{blue square}, \text{red square}, \text{red square})$  

Networks of Multi-Server Queues with Parallel Processing - Céline Comte
Service rate

\[ c = (\square, \square, \square) \]
Service rate

\[ c = (\text{ }, \text{ }, \text{ }) \]

\[ \mu (\{\text{ }\}) \]

\[ \mu (\{\text{ }\}) \]

\[ \mu (\{\text{ }, \text{ }\}) \]

\[ \mu (\{\text{ }, \text{ }, \text{ }\}) \]
Service rate

\[ \mu(A) = \sum_{i} A_i \phi_i(A) \]

Networks of Multi-Server Queues with Parallel Processing - Céline Comte
Service rate

Networks of Multi-Server Queues with Parallel Processing - Céline Comte
Service rate

\[ C = \left\{ \phi \in \mathbb{R}_+^I : \forall A \subset I, \sum_{i \in A} \phi_i \leq \mu(A) \right\} \]
Stationary measure (Berezner and Krzesinski, 1996)

- A stationary measure of the queue state is

\[
\pi(c) = \pi(\emptyset) \prod_{k=1}^{n} \frac{\lambda_{c_k}}{\mu(A(c_1, \ldots, c_k))}, \quad \forall c \in I^*
\]

- The queue is quasi-reversible
  - The current state of the queue is independent of previous departures and future arrivals
  - Arrivals and departures form independent Poisson processes
Internal routing

By quasi-reversibility, the stationary measure of the queue state is

$$\pi(c) = \pi(\emptyset) \prod_{k=1}^{n} \frac{\lambda_{c_k}}{\mu(A(c_1, \ldots, c_k))},$$

independently of $p \in [0, 1]$. 
State aggregation

Aggregate state \( x = (x_i : i \in I) \in \mathbb{N}^I \)

\[
x = (1, 2) \quad \rightarrow \quad c \in \left\{ \begin{array}{l}
(\text{ }, \text{ }, \text{ }) , \\
(\text{ }, \text{ }, \text{ }) , \\
(\text{ }, \text{ }, \text{ }) 
\end{array} \right\}
\]

Stationary measure \( \pi(x) = \sum_{c : |c| = x} \pi(c) \)
Stationary distribution (Berezner and Krzesinski, 1996)

The stationary measure of the aggregate state \( x \) satisfies

\[
\pi(x) = \pi(0) \Phi(x) \prod_{i \in I} \lambda_i^{x_i}, \quad \forall x \in \mathbb{N}^I,
\]

where \( \Phi \) is defined by the recursion \( \Phi(0) = 1 \) and, for each \( x \neq 0 \),

\[
\Phi(x) = \frac{1}{\mu(A(x))} \sum_{i \in A(x)} \Phi(x - e_i).
\]
Equivalent Whittle network

$\pi$ is the stationary measure of the state of a Whittle network of $|I|$ queues with arrival rates $\lambda_i, i \in I$, and service rates

$$\phi_i(x) = \frac{\Phi(x - e_i)}{\Phi(x)}, \quad \forall x \in \mathbb{N}^I, \quad \forall i \in A(x).$$

Multi-server queue

1

2

3

Multi-server queue

2

3

Equivalent Whittle network

$x_1$ jobs

$\phi_1(x)$

$\text{averaging}$

$x_2$ jobs

$\phi_2(x)$
Equivalent Whittle network

Per-class service rates:

- $\mu_i(c)$ in state $c$ of the multi-server queue
- $\phi_i(x)$ in state $x$ of the Whittle network

**Theorem 1**

The service rates in the equivalent Whittle network are the average per-class service rates in the multi-server queue:

$$\phi_i(x) = \sum_{c:|c|=x} \frac{\pi(c)}{\pi(x)} \mu_i(c), \quad \forall x \in \mathbb{N}^I, \quad \forall i \in A(x).$$
Average service rates
Average service rates

\[ c = (\text{red}, \text{red}, \text{blue}) \]
and \[ c = (\text{red}, \text{blue}, \text{red}) \]

\[ c = (\text{blue}, \text{red}, \text{red}) \]
Average service rates

\[ x = (\text{ }, 2\text{ }) \]

\[ c = (\text{ }, \text{ }, \text{ }) \]

and \[ c = (\text{ }, \text{ }, \text{ }) \]
Average service rates

\[ x = (\text{ }, 2\text{ }) \]
Average service rates

\[ x = (\phi_1(1, 2), 2\phi_2(1, 2)) \]
Balanced fairness

The average service rates are

- **Balanced:** \( \frac{\phi_i(x - e_j)}{\phi_i(x)} = \frac{\phi_j(x - e_i)}{\phi_j(x)} \), \( \forall x \in \mathbb{N}^I \), \( \forall i, j \in A(x) \).

- **Efficient:** \( \sum_{i \in I} \phi_i(x) = \mu(A(x)) \), \( \forall x \in \mathbb{N}^I \).

**Balanced fairness** in the capacity set

\[
C = \left\{ \phi \in \mathbb{R}^I_+ : \forall A \subset I, \sum_{i \in A} \phi_i \leq \mu(A) \right\}
\]
Balanced fairness

The most efficient **insensitive** resource allocation

- Introduced for dimensioning data networks (Bonald and Proutière, 2003)
- Good approximation of proportional fairness
- Recently applied to Content Delivery Networks (Shah and de Veciana, 2015, 2016)
Stability condition

**Theorem 2**

_The multi-server queue is stable if and only if_

\[ \forall A \subseteq I, \sum_{i \in A} \lambda_i < \mu(A) \]
Aggregation

- Queue state $c$: \( \pi(c) = \frac{\lambda c_n \pi(c_1, \ldots, c_{n-1})}{\mu(c)} \), \( \forall c \neq \emptyset \)

- Aggregate state $x$: \( \pi(x) = \frac{\sum_{i \in A(x)} \lambda_i \pi(x - e_i)}{\mu(x)} \), \( \forall x \neq 0 \)

- Set of active classes $A$
  (Bonald et al., 2003; Shah and de Veciana, 2015, 2016)

\[
\pi(A) = \frac{\sum_{i \in A} \lambda_i \pi(A \setminus \{i\})}{\mu(A) - \sum_{i \in A} \lambda_i}, \quad \forall A \neq \emptyset
\]

→ Closed-form expressions for the performance metrics
  - Proportion of time the queue is idle
  - Mean number of jobs of each class
  - ...

Networks of Multi-Server Queues with Parallel Processing  - Céline Comte
Background on Order Independent queues

Multi-server queues with parallel processing

Scheduling algorithm
Computer cluster

An arriving job is assigned a set of computers independently of the state of the cluster.

Objective: Enforce balanced fairness.
Computer cluster

- Server $s$
  - Fixed capacity $C_s$ in flops
Computer cluster

- Server $s$
  - Fixed capacity $C_s$ in flops
- An arriving job is assigned a set of computers independently of the state of the cluster.
Computer cluster

- Server $s$
  - Fixed capacity $C_s$ in flops

- An arriving job is assigned a set of computers *independently of the state of the cluster*

- Service requirements
  - General distribution with mean $\sigma$ in floating-point operations
Computer cluster

- Server $s$
  - Fixed capacity $C_s$ in flops
- An arriving job is assigned a set of computers *independently of the state of the cluster*
- Service requirements
  - General distribution with mean $\sigma$ in floating-point operations

Objective: Enforce balanced fairness
Single computer

Single-server mono-class cluster, service requirements with mean $\sigma$
Single computer

Single-server mono-class cluster, service requirements with mean $\sigma$

- FCFS at job scale
- Very sensitive
Single computer

Single-server mono-class cluster, service requirements with mean $\sigma$

- FCFS at job scale
- Very sensitive
- Service interruption after $\theta$ floating point operations on average
Single computer

Single-server mono-class cluster, service requirements with mean $\sigma$
Single computer

Single-server mono-class cluster, service requirements with mean $\sigma$

- Initialize a timer $\sim \mathcal{E} \left( \frac{C}{\theta} \right)$
Single computer

Single-server mono-class cluster, service requirements with mean $\sigma$

- Initialize a timer $\sim \mathcal{E}\left(\frac{C}{\theta}\right)$
- Upon service completion, the job leaves the cluster
Single computer

Single-server mono-class cluster, service requirements with mean $\sigma$

- Initialize a timer $\sim \mathcal{E}\left(\frac{C}{\theta}\right)$
- Upon service completion, the job leaves the cluster
- When the timer expires, the service is interrupted
Single computer

Single-server mono-class cluster, service requirements with mean $\sigma$

- Initialize a timer $\sim \mathcal{E}\left(\frac{C}{\theta}\right)$
- Upon service completion, the job leaves the cluster
- When the timer expires, the service is interrupted
Single computer

Single-server mono-class cluster, service requirements with mean $\sigma$

- Initialize a timer $\sim E \left( \frac{C}{\theta} \right)$
- Upon service completion, the job leaves the cluster
- When the timer expires, the service is interrupted

Parameter $m =$ mean number of interruptions per job
When $m \to \infty$, single-server queue under PS service discipline
Computer cluster
Computer cluster

Order Independent queues
Multi-server queues
Scheduling algorithm

Timer $\sim \mathcal{E} \left( \frac{C_1}{\theta} \right)$

Timer $\sim \mathcal{E} \left( \frac{C_2}{\theta} \right)$

Timer $\sim \mathcal{E} \left( \frac{C_3}{\theta} \right)$

When $m \neq 1$, resources allocated according to balanced fairness
Computer cluster

\[ \text{Timer} \sim \mathcal{E} \left( \frac{C_1}{\theta} \right) \]

\[ \text{Timer} \sim \mathcal{E} \left( \frac{C_2}{\theta} \right) \]

\[ \text{Timer} \sim \mathcal{E} \left( \frac{C_3}{\theta} \right) \]
Computer cluster

\[
\begin{align*}
C_1 &: \text{Timer } \sim \mathcal{E} \left( \frac{C_1}{\theta} \right) \\
\quad &\quad \left\{ \begin{array}{l}
\text{Interrupted after} \\
\quad \sim \mathcal{E} \left( \frac{C_1+C_2}{\theta} \right)
\end{array} \right.
\\
C_2 &: \text{Timer } \sim \mathcal{E} \left( \frac{C_2}{\theta} \right) \\
\quad &\quad \left\{ \begin{array}{l}
\text{Interrupted after} \\
\quad \sim \mathcal{E} \left( \frac{C_3}{\theta} \right)
\end{array} \right.
\\
C_3 &: \text{Timer } \sim \mathcal{E} \left( \frac{C_3}{\theta} \right)
\end{align*}
\]
Computer cluster

Timer $\sim \mathcal{E}\left(\frac{C_1}{\theta}\right)$

Interrupted after $\sim \mathcal{E}\left(\frac{C_1+C_2}{\theta}\right)$

Timer $\sim \mathcal{E}\left(\frac{C_2}{\theta}\right)$

Timer $\sim \mathcal{E}\left(\frac{C_3}{\theta}\right)$

Interrupted after $\sim \mathcal{E}\left(\frac{C_3}{\theta}\right)$

When $m \neq 1$, resources allocated according to balanced fairness.
Computer cluster

Order Independent queues
Multi-server queues
Scheduling algorithm

When $m \neq 1$, resources allocated according to balanced fairness.

Networks of Multi-Server Queues with Parallel Processing - Céline Comte
Computer cluster

Timer $\sim \mathcal{E} \left( \frac{C_1}{\theta} \right)$

Timer $\sim \mathcal{E} \left( \frac{C_2}{\theta} \right)$

Timer $\sim \mathcal{E} \left( \frac{C_3}{\theta} \right)$
Computer cluster

When $m \neq 1$, resources allocated according to balanced fairness

$$\text{Timer } \sim \mathcal{E} \left( \frac{C_1}{\theta} \right) \quad \left\{ \begin{array}{l} \text{Interrupted after} \\ \sim \mathcal{E} \left( \frac{C_1}{\theta} \right) \end{array} \right.$$ 

$$\text{Timer } \sim \mathcal{E} \left( \frac{C_2}{\theta} \right)$$

$$\text{Timer } \sim \mathcal{E} \left( \frac{C_3}{\theta} \right)$$

$$\text{Interrupted after}$$

$$\sim \mathcal{E} \left( \frac{C_2 + C_3}{\theta} \right)$$
Computer cluster

When $m \to \infty$, resources allocated according to balanced fairness
Numerical results

- Performance metric:
  Mean service rate seen by class-\(i\) jobs

- Hyperexponential job size distribution
  \( \sim \mathcal{E}(1/5) \) with probability 1/6
  \( \sim \mathcal{E}(5) \) with probability 5/6
Numerical results: Shared pool

- Balanced fairness
  - $m = 1$
  - $m = 2$
  - $m = 5$

Mean service rate vs Load

Order Independent queues
Multi-server queues
Scheduling algorithm
Numerical results: Random assignment

- $d$ servers chosen uniformly and independently at random

- By (Gardner et al., 2016),

\[
\frac{1}{\gamma} = \sum_{j=d}^{S} \frac{1}{S\mu \frac{(S-1)}{(j-1)} - SL}
\]
Numerical results: \( S = 100, \; d = 2 \)
Numerical results: $S = 100, \ d = 3$
Conclusion

- Multi-server queues with parallel processing
  - Sequential version of a class of Whittle networks
  - Stable whenever each set of classes can handle its own load
  - Closed-form expressions for the performance metrics

- Scheduling algorithm in computer clusters
  - Service interruptions implemented by exponential timers
  - Insensitive resource allocation

- Future works
  - Generalize these results to Order Independent queues
  - Assert robustness of the algorithm
Bibliography


Bibliography

T. Bonald and A. Proutière (2003). “Insensitive Bandwidth Sharing in Data Networks”. In: Queueing Syst. 44.1, pp. 69–100.

