# Scaling resources for sustainable provision of URLLC services

## Static and dynamic optimization

**Pablo Serrano Yáñez-Mingot**

**Brescia, October 10, 2025**

# Acknowledgements


**This presentation is partly supported by**

The Project TUCAN6-CM (TEC-2024/COM-460), funded by CM (ORDEN 5696/2024)

TUCAN6-CM

The Project 6GINSPIRE PID2022-137329OB-C42, funded by MCIN/AEI/10.13039/501100011033/

6G-INSPIRE

The SNS JU EU's HE research and innovation programme under Grant Agreement No. 101192035 (AMAZING-6G)

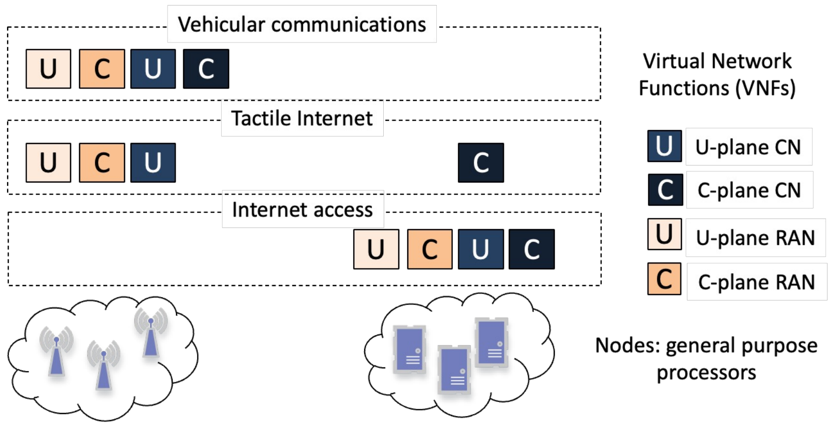
AMAZING-6G

# Context

## *Network Softwarization*

- Network slicing: efficient provision of multiple heterogeneous services on the same infrastructure [1]

- Virtualization (softwarization) of network functions



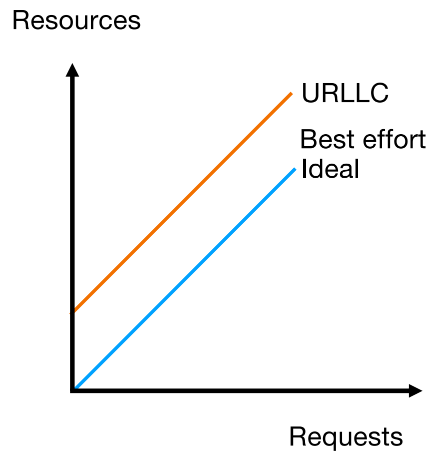

  - Dynamically allocate and share resources (e.g., Nuberu [2])
  - Migrate between servers without disruption (e.g., ACHO [3])

1. P. Serrano, «Tutorial: A Primer on 5G Network Slicing: Concepts, Algorithms, and Practice», IEEE CAMAD 2018, Barcelona,
2. G. Garcia-Aviles et al. «Nuberu: Reliable RAN Virtualization in Shared Platforms», ACM Mobicom '21
3. G. Garcia-Aviles et al. «ACHO: A Framework for Flexible Re-Orchestration of Virtual Network Functions», Computer Networks, 2020

# Efficiency with a URLLC service

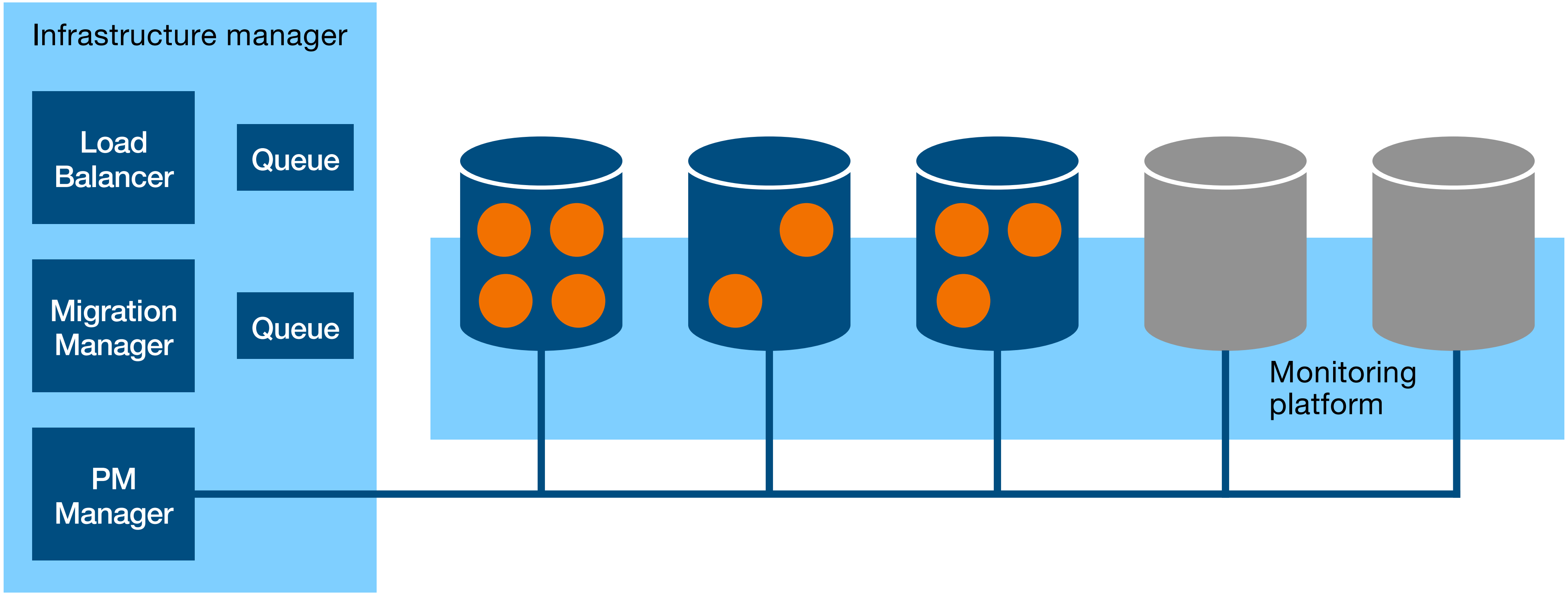
## Very high reliability

- Efficiency: allocate strictly necessary resources
- Obstacle 1: Resource bootstrapping is not instantaneous
- Bursts of requests: they may have to wait
- Obstacle 2: Life spans are finite [1]
- Relocate tasks from servers that are about to fail
- Best effort vs. URLLC [2] (five 9s)

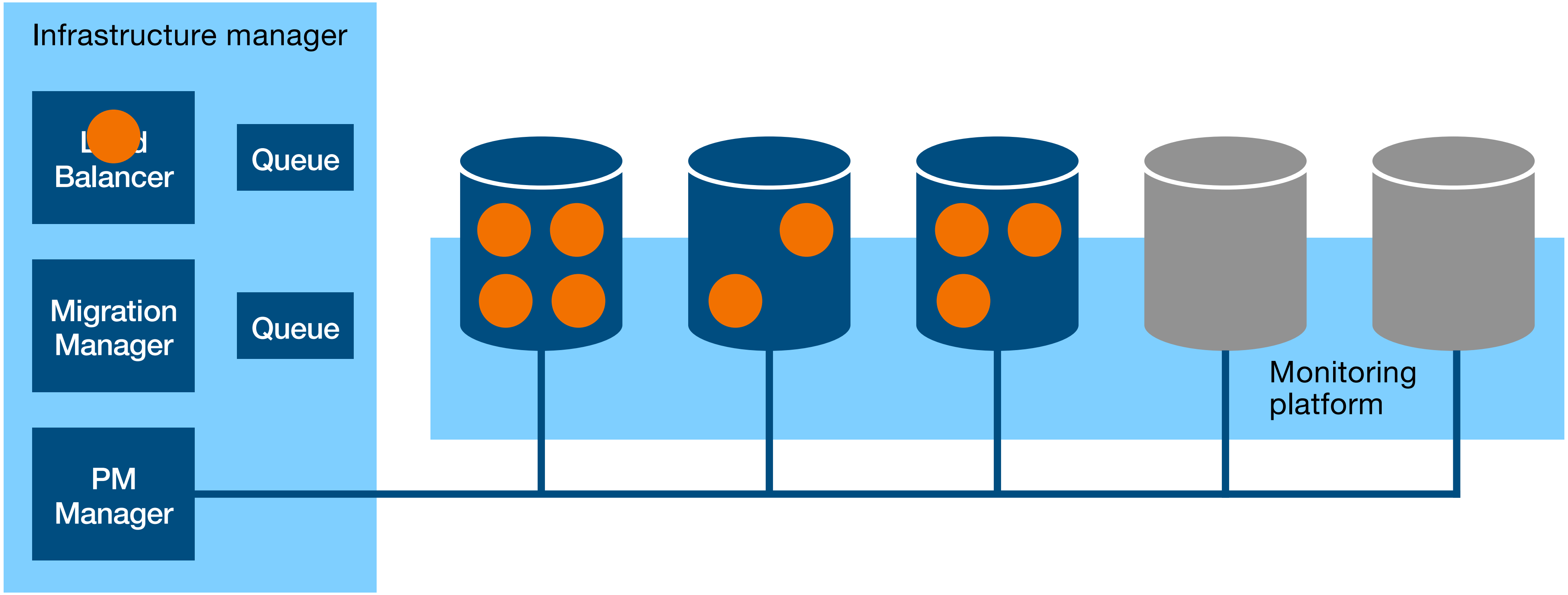



1. Fung Po Tso et al. «The Glasgow Raspberry Pi cloud: A scale model for cloud computing infrastructures». IEEE 33rd International Conference on Distributed Computing Systems Workshops, 2013.
2. W. Nakimuli et al. «Deployment and Evaluation of an Industry 4.0 Use Case over 5G», IEEE Communications Magazine, July 2021

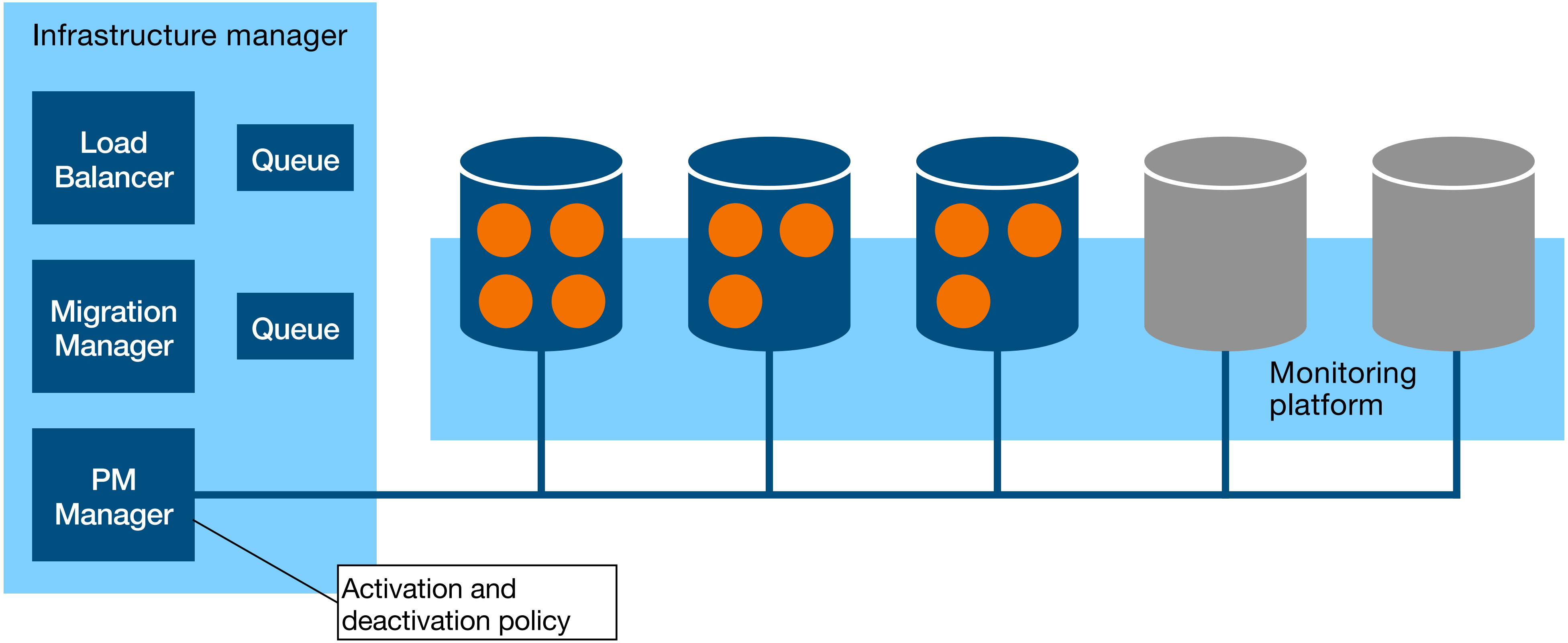
# Example


Infrastructure manager
Load Balancer
Queue
Migration Manager
Queue
PM Manager
Monitoring platform

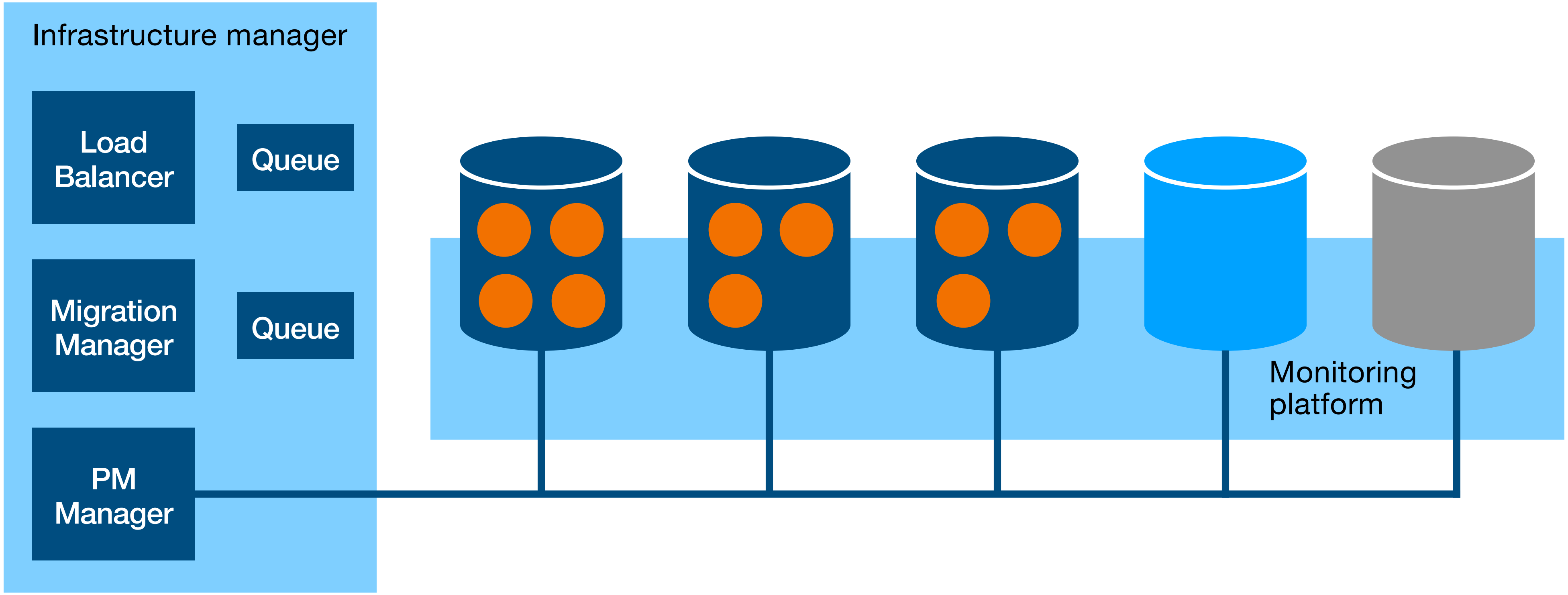
# Example


Infrastructure manager
L d
Balancer
Queue
Migration
Manager
Queue
PM
Manager
Monitoring
platform

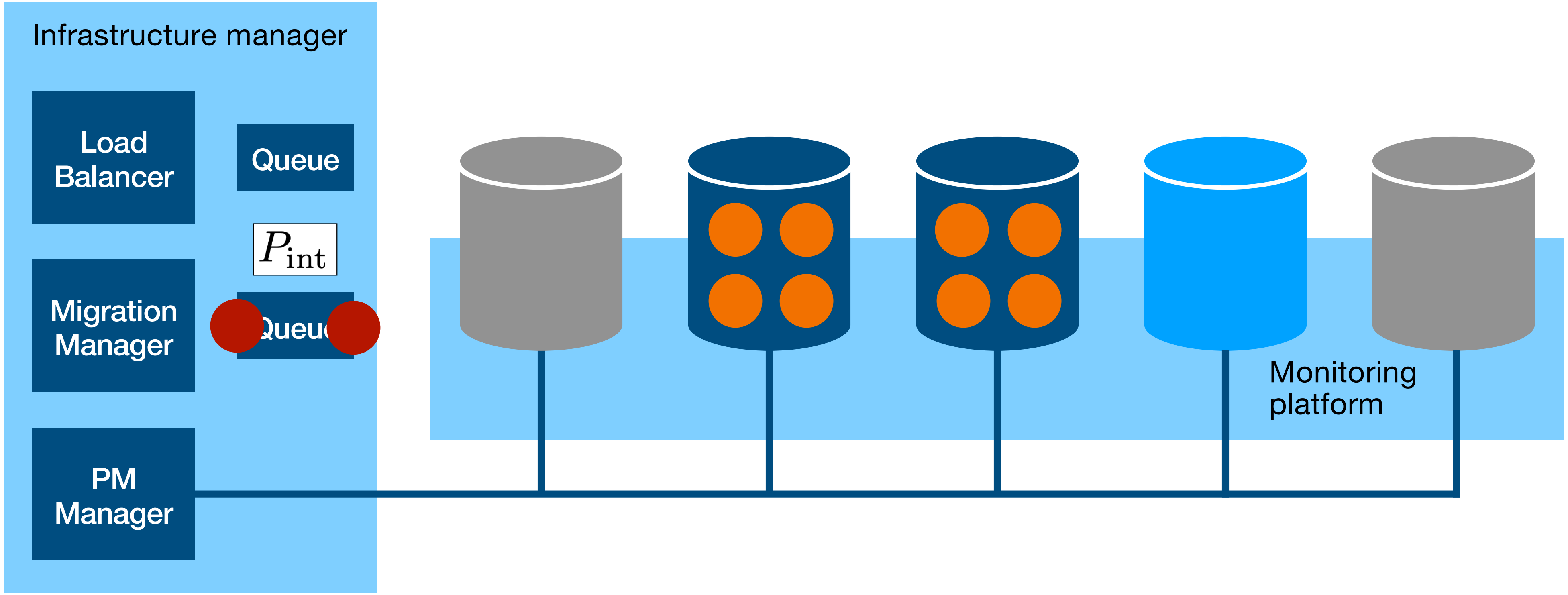
# Example


Infrastructure manager
Load Balancer
Queue
Migration Manager
Queue
PM Manager
Monitoring platform
Activation and deactivation policy

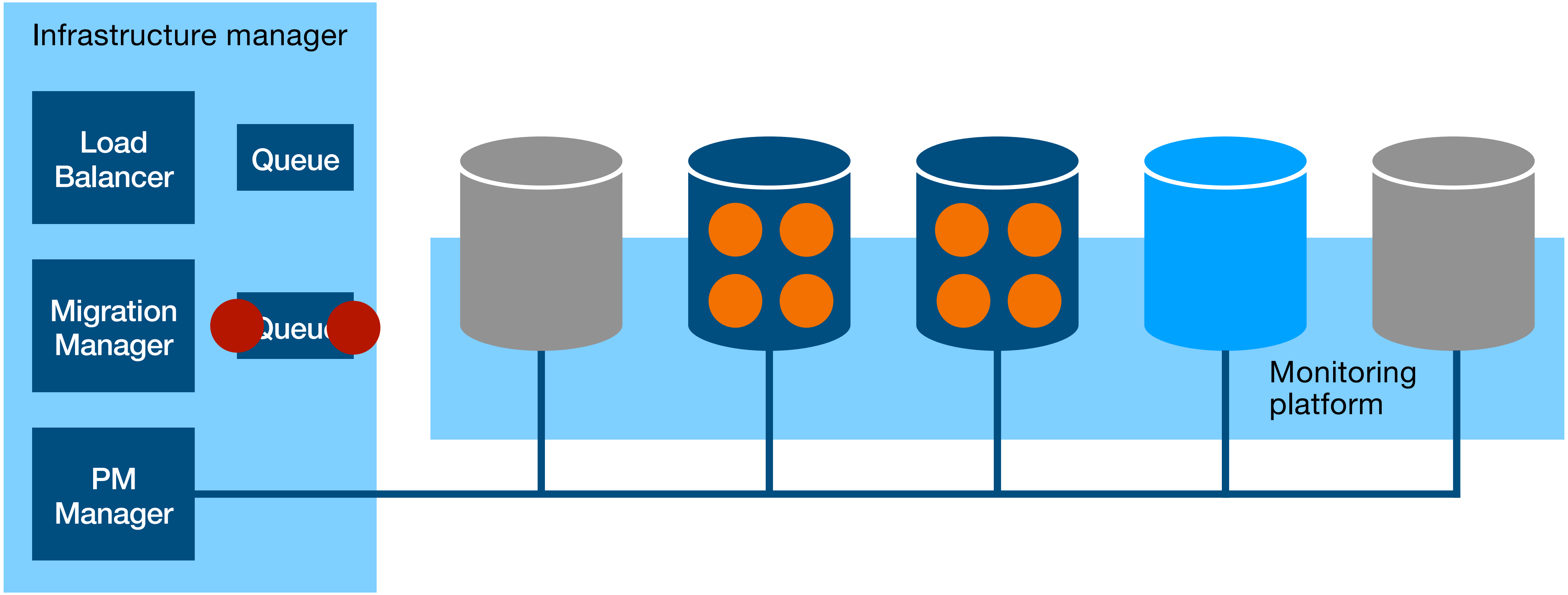
# Example


Infrastructure manager
Load Balancer
Queue
Migration Manager
Queue
PM Manager
Monitoring platform

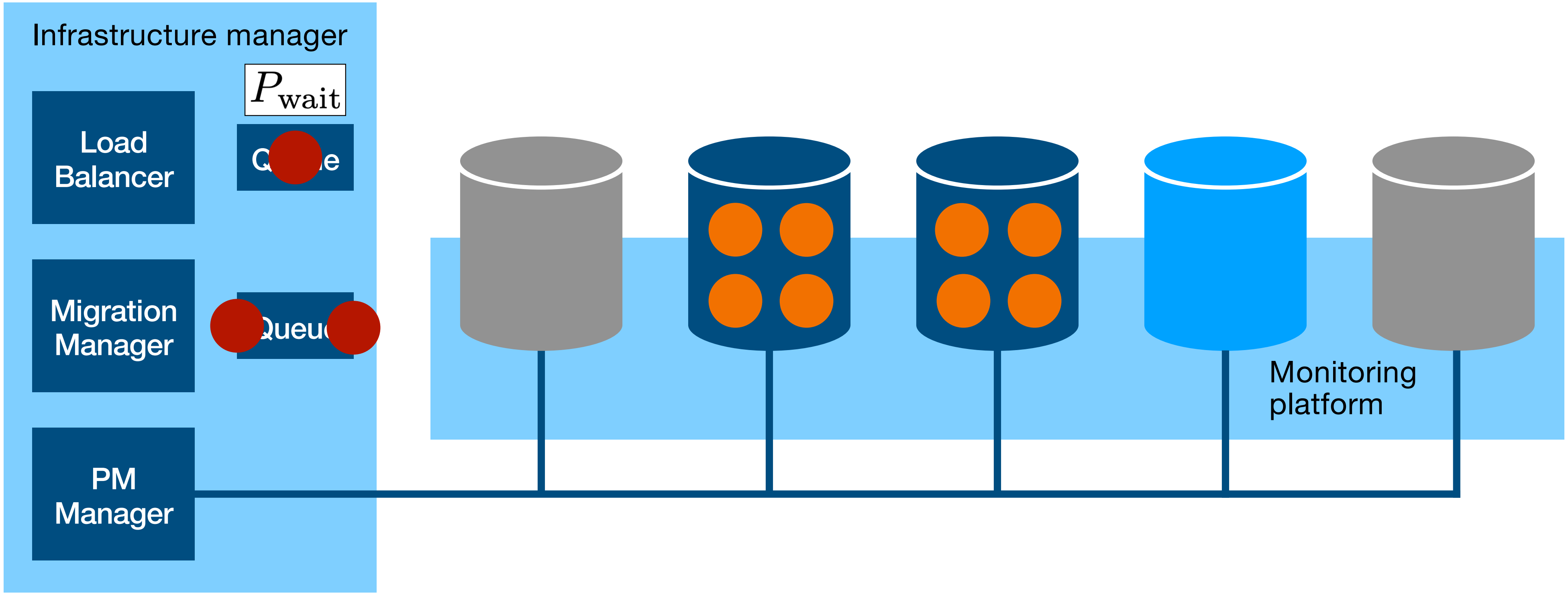
# Example


Infrastructure manager
Load Balancer
Queue
$P_{\mathrm{int}}$
Migration Manager
Queue
PM Manager
Monitoring platform

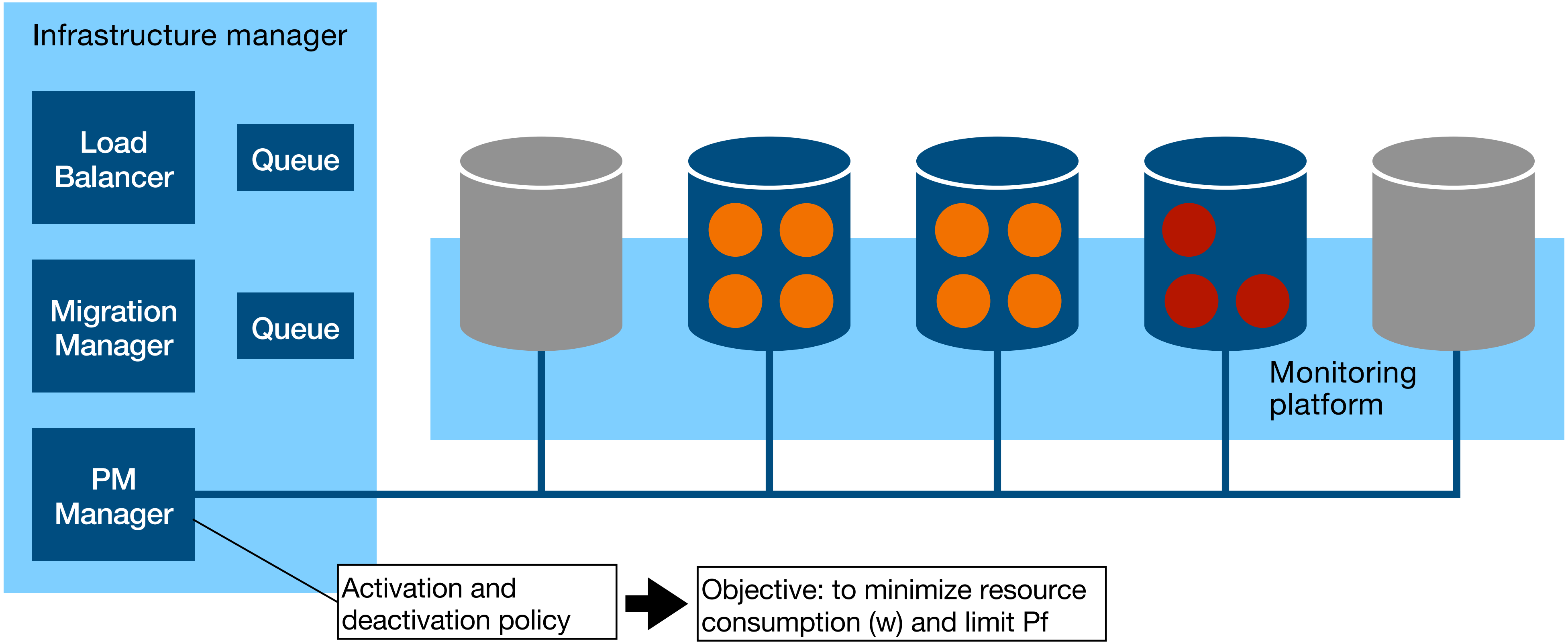
# Example


Infrastructure manager
Load Balancer
Queue
Migration Manager
Queue
PM Manager
Monitoring platform

# Example


Infrastructure manager
$P_{\text{wait}}$
Load Balancer
Migration Manager
PM Manager
Monitoring platform

# Example



Infrastructure manager
Load Balancer
Queue
Migration Manager
Queue
PM Manager
Monitoring platform
Activation and deactivation policy
Objective: to minimize resource consumption (w) and limit Pf

# Static optimization

# System

## Modeling hypotheses

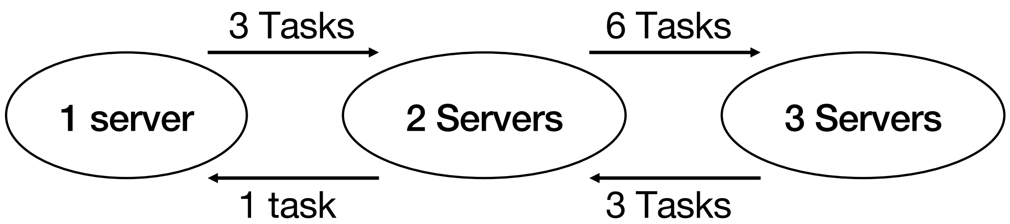
- Tasks arrive following Poisson $\lambda$ and exponential service times $\mu^{-1}$
- A server can serve up to N tasks, there are M servers
- Exponential boot up times: $\alpha^{-1}$ Immediate shut downs.
- Server lifetimes are exponential [1] $\nu^{-1}$
- Energy consumption: 0 off, $P_{idle}$ when on, $P_{load}$ proportional to load [2]
- Tasks can be moved between servers before crashing

1. K. S. Trivedi y A. Bobbio, «Reliability and Availability Engineering: Modeling, Analysis, and Applications», Cambridge University Press, 2017
2. Gong Chen et al. «Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services». NSDI'08.

# System

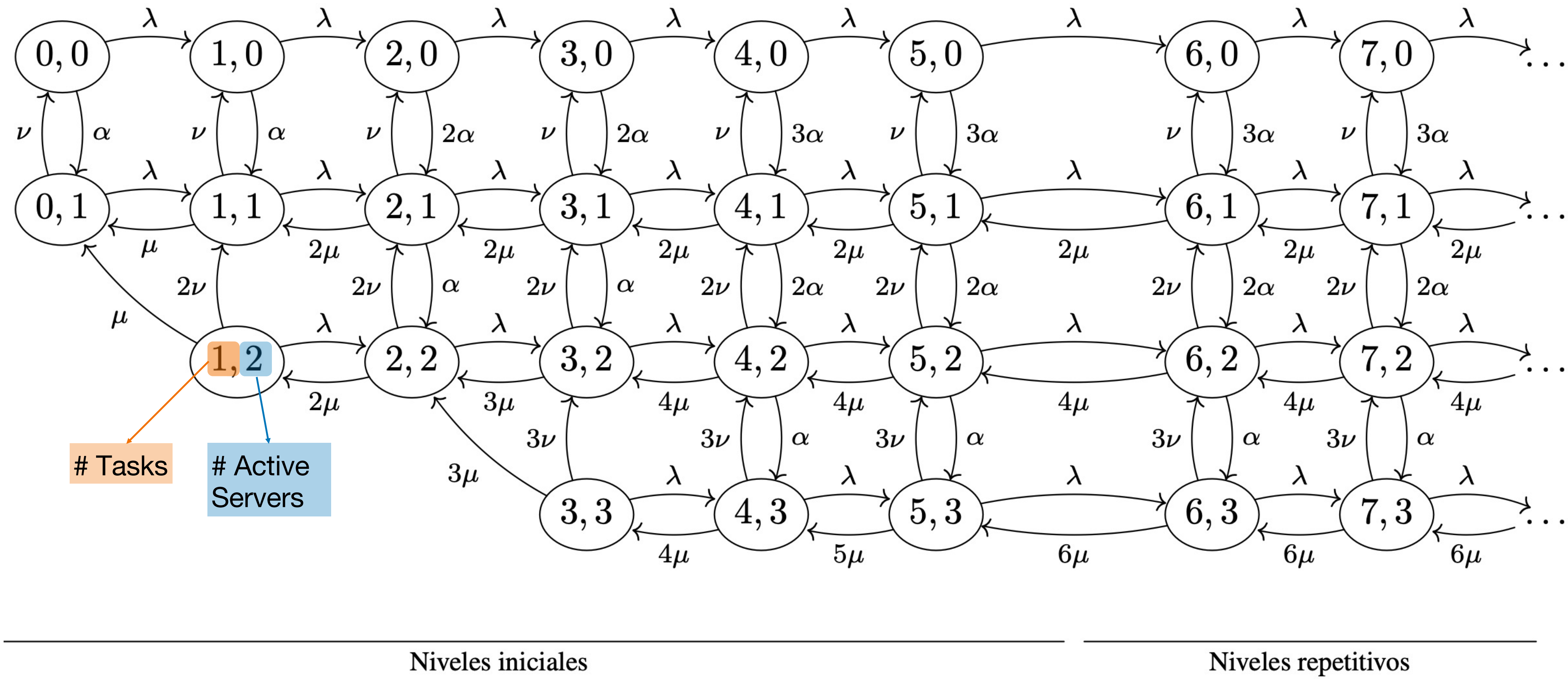
## De/Activation Policy

- Activation threshold $t_{on}(m)$ : Number of tasks on the system that initiates activation of the m-th server
- Deactivation threshold $t_{off}(m)$ : Number of tasks on the system that causes a server to be disabled when there are m active servers
- Example
  - $t_{on}(2) = 3$, $t_{on}(3) = 6$
  - $t_{off}(2) = 1$, $t_{off}(3) = 3$



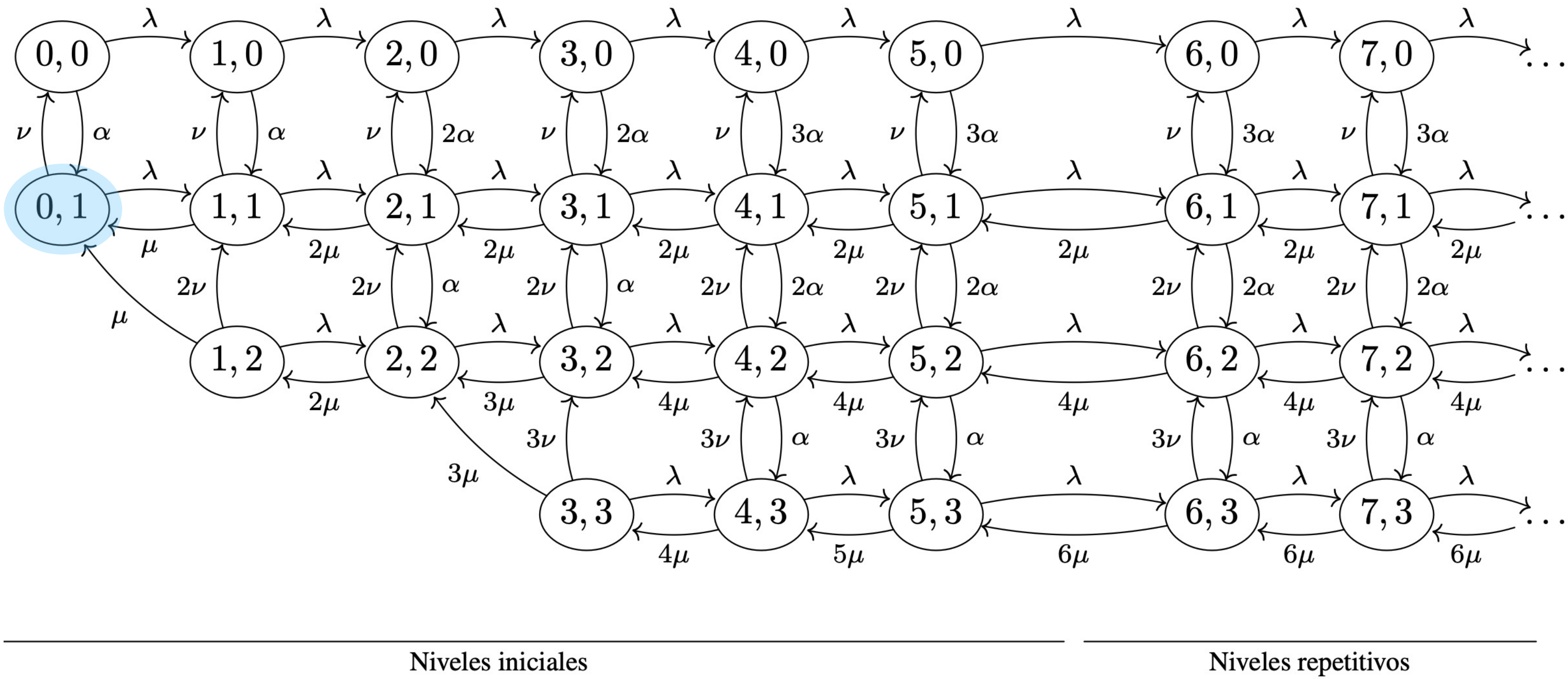
# Analytical model: Markov chain

## Quasi-Birth-Death process



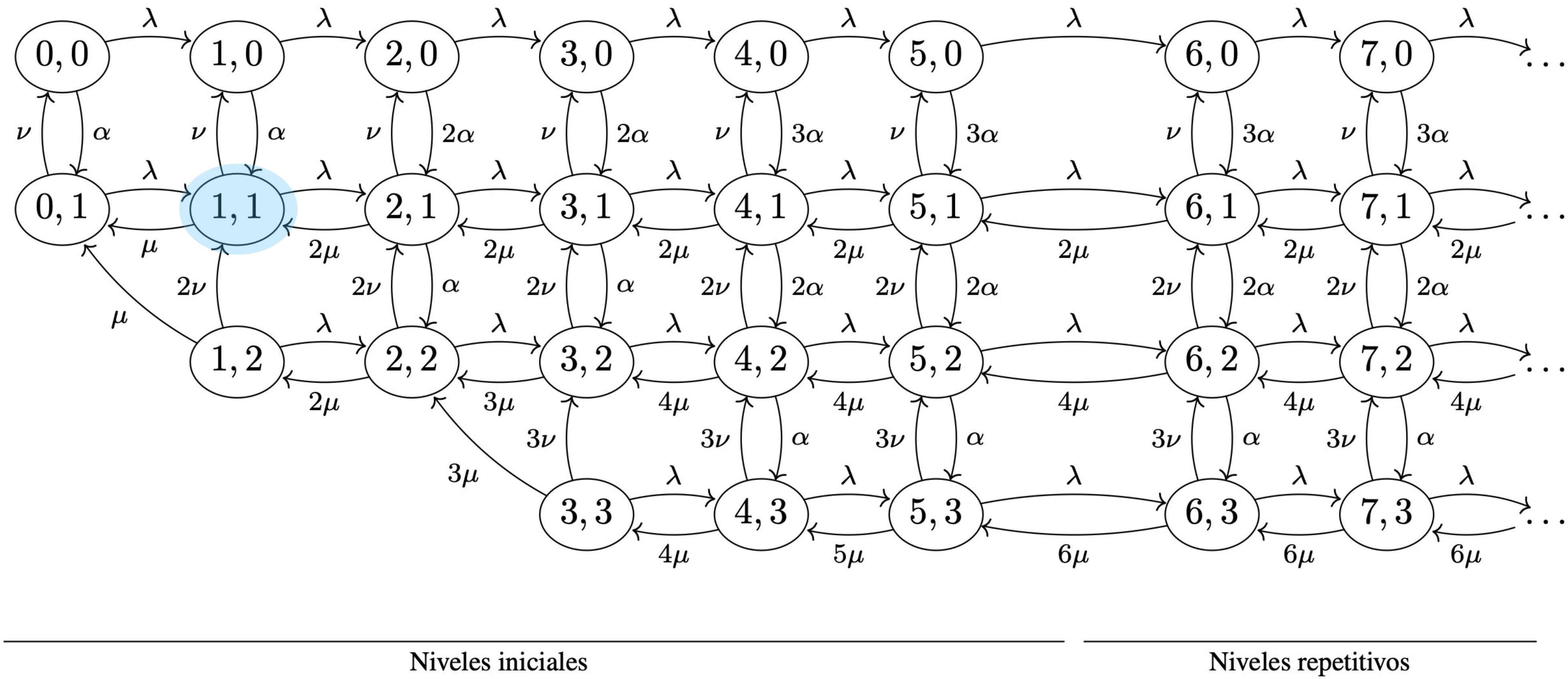
# Analytical model: Markov chain

## Quasi-Birth-Death process



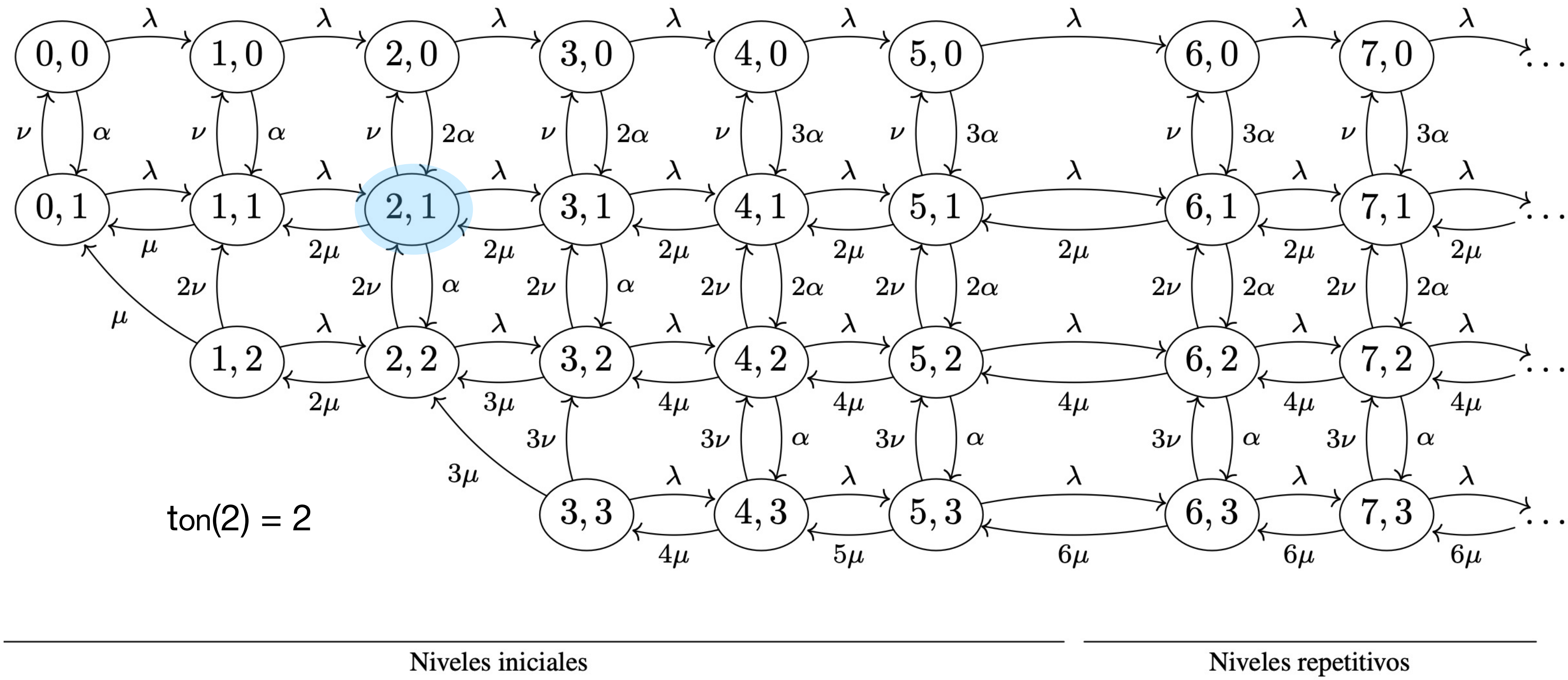
# Analytical model: Markov chain

## Quasi-Birth-Death process



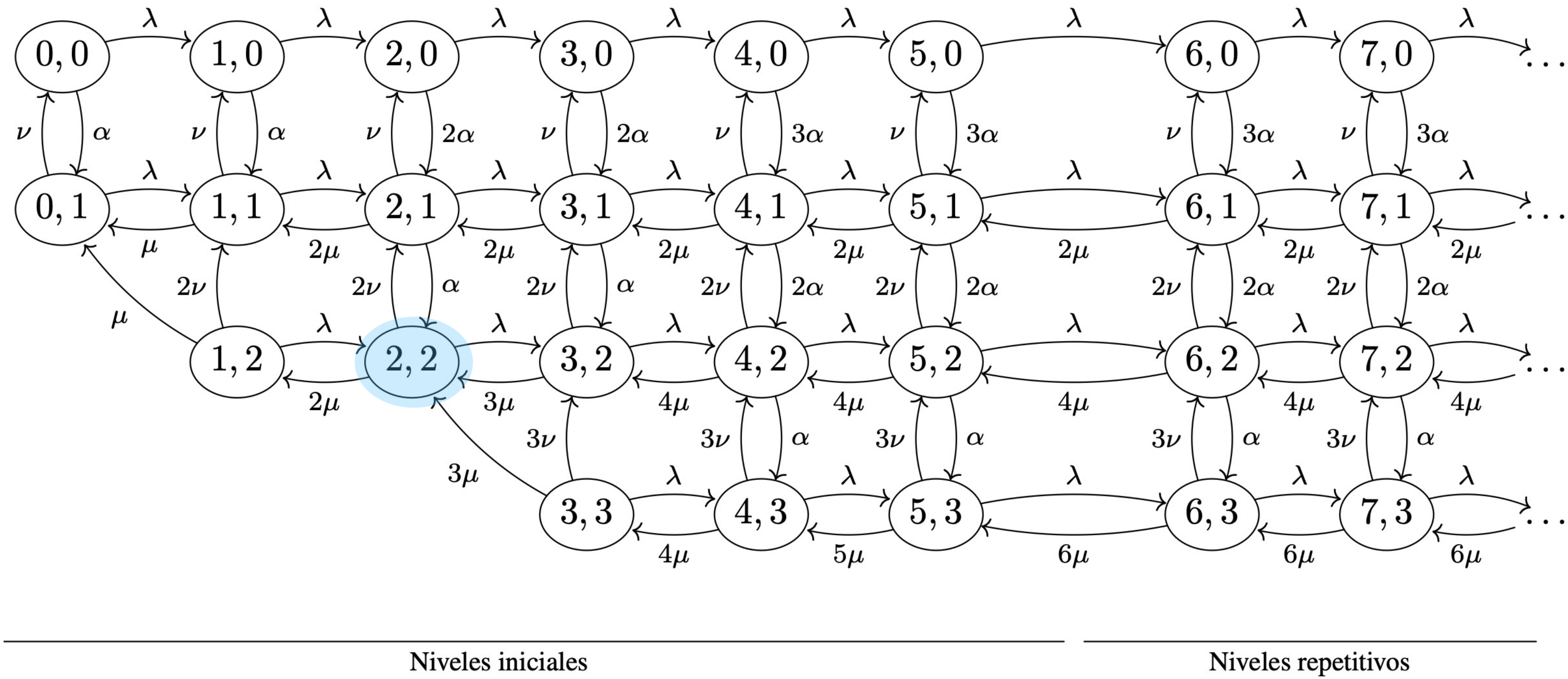
# Analytical model: Markov chain

## Quasi-Birth-Death process



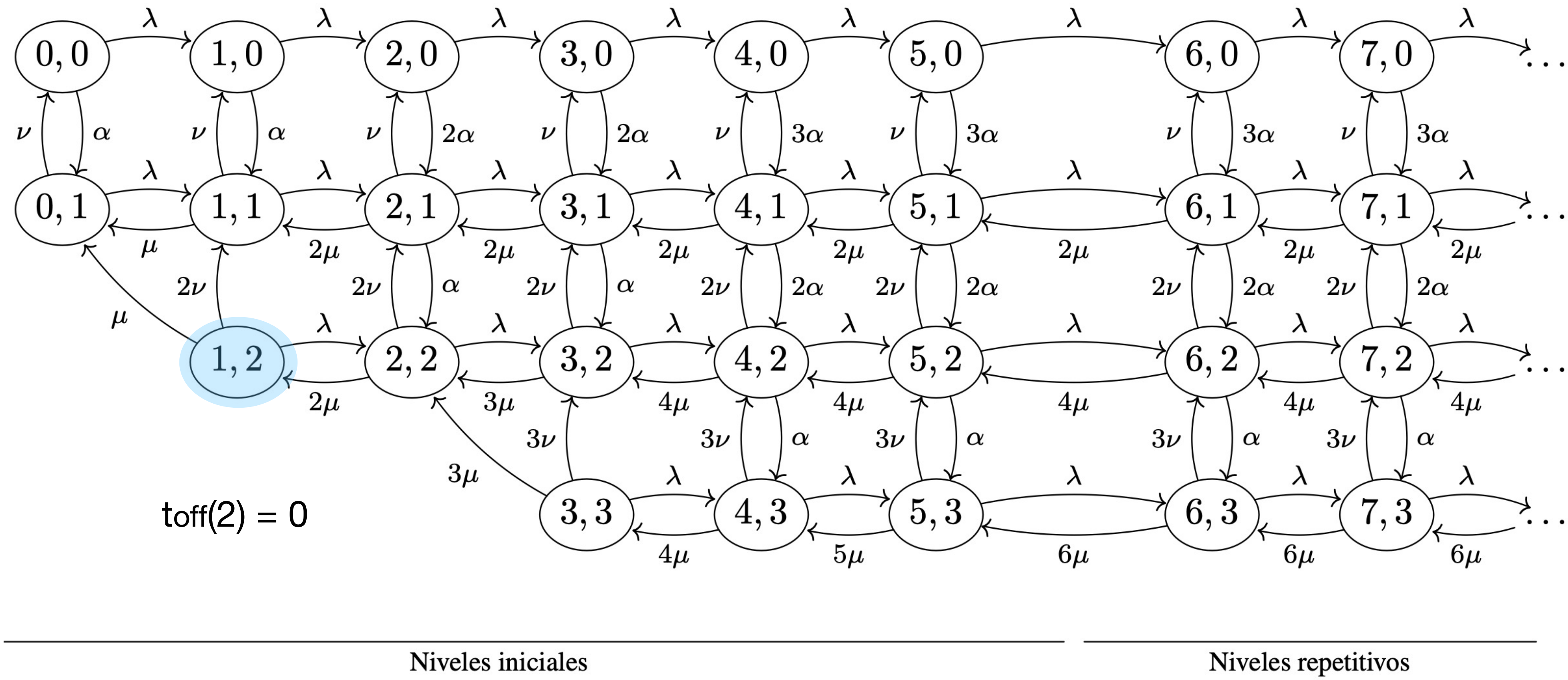
# Analytical model: Markov chain

## Quasi-Birth-Death process



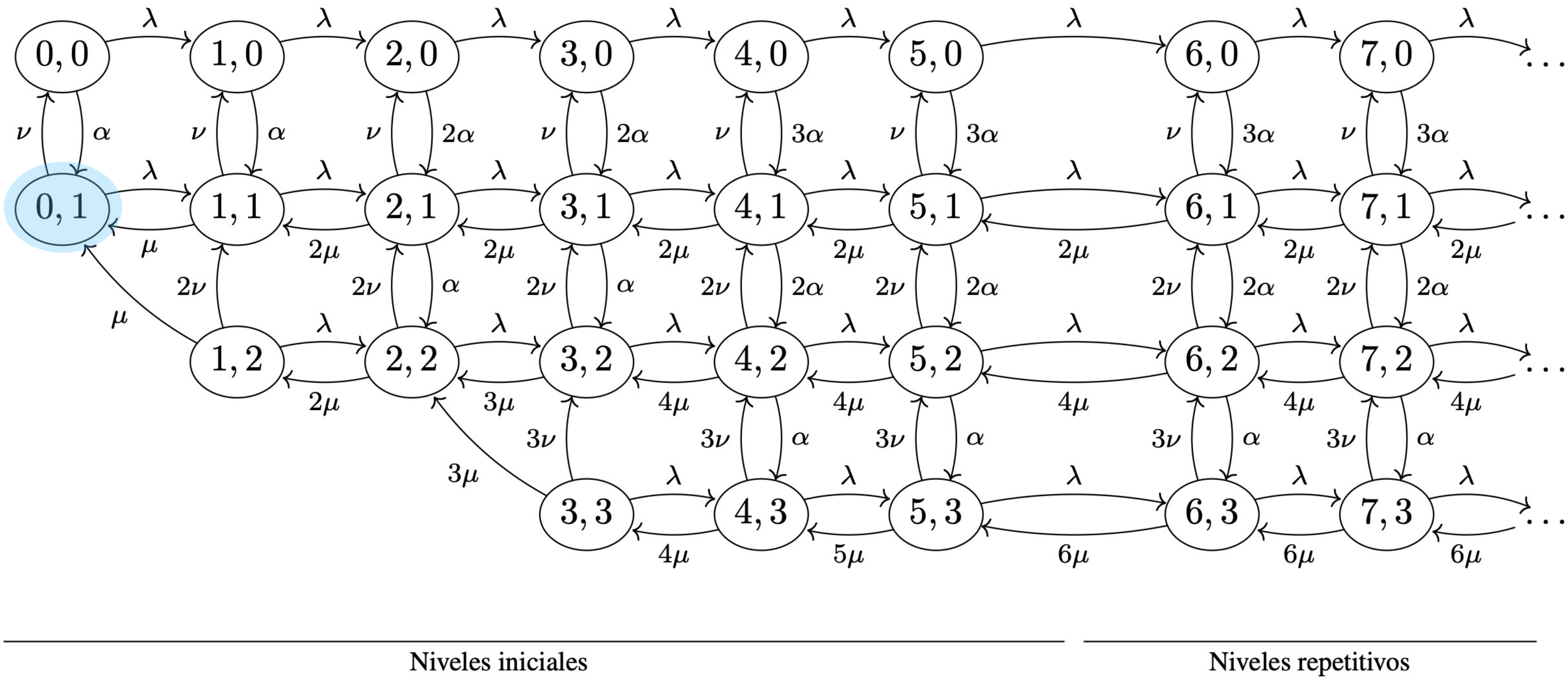
# Analytical model: Markov chain

## Quasi-Birth-Death process



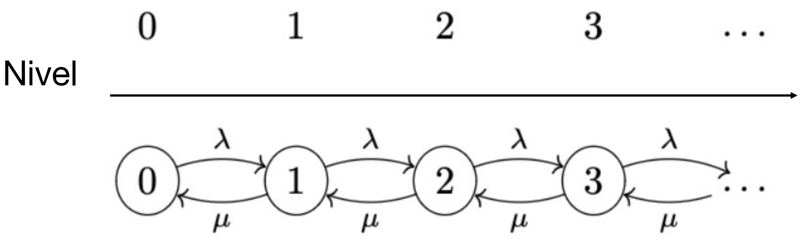
# Analytical model: Markov chain

## Quasi-Birth-Death process



# Birth-Death vs. Quasi-Birth-Death

## Birth death process

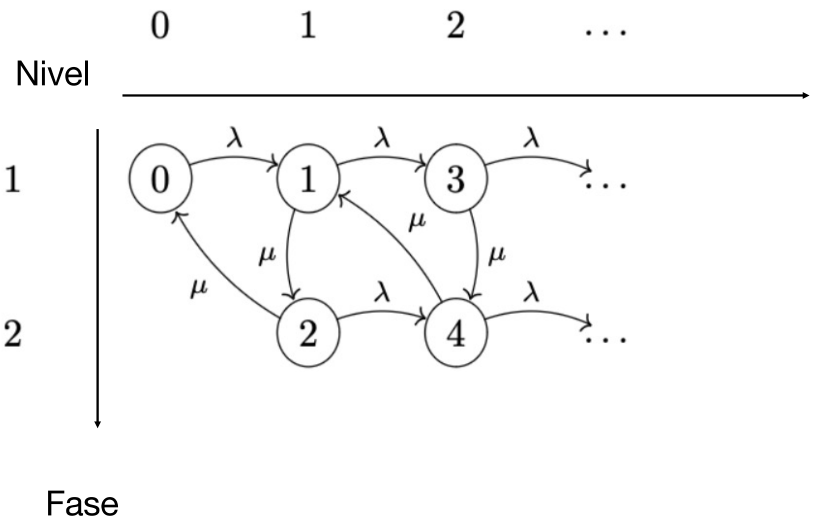



$$\boldsymbol{Q} = \begin{pmatrix} -f_0 & f_0 & & & \\ b_1 & -(b_1+f_1) & f_1 & & \\ & b_2 & -(b_2+f_2) & f_2 & \cdots \\ & & & \vdots & \ddots \end{pmatrix}$$

$$f_i = \lambda$$
$$b_j = \mu$$

# Birth-Death vs. Quasi-Birth-Death

## Quasi-birth-death process




$$\boldsymbol{F}^{(0)} = \begin{pmatrix} \lambda & 0 \end{pmatrix}$$

$$\boldsymbol{B}^{(1)} = \begin{pmatrix} 0 \\ \mu \end{pmatrix} \qquad \boldsymbol{L}^{(0)} = -\lambda$$

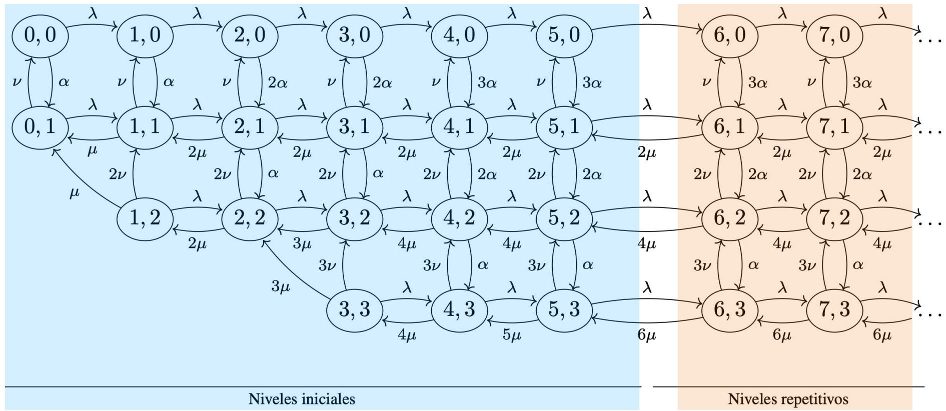
$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{L}^{(0)} & \boldsymbol{F}^{(0)} & & & \\ \boldsymbol{B}^{(1)} & \boldsymbol{L}^{(1)} & \boldsymbol{F}^{(1)} & & \\ & \boldsymbol{B}^{(2)} & \boldsymbol{L}^{(2)} & \boldsymbol{F}^{(2)} & \cdots \\ & & & \vdots & \ddots \end{pmatrix}$$

$$\boldsymbol{F}^{(l)} = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \qquad \boldsymbol{L}^{(l)} = \begin{pmatrix} -(\mu + \lambda) & \mu \\ 0 & -(\mu + \lambda) \end{pmatrix}$$

$$\boldsymbol{B}^{(l)} = \begin{pmatrix} 0 & 0 \\ \mu & 0 \end{pmatrix}$$

# Markov chain

## Various initial levels




$$
\mathbf{Q} = \left[\begin{array}{cccccccccc}
B_{00} & B_{01} & 0 & 0 & \dots & 0 & & & & \\
B_{10} & B_{11} & B_{12} & 0 & \dots & 0 & & & & \\
0 & B_{21} & B_{22} & B_{23} & \dots & 0 & & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & \dots & & & & \\
0 & \dots & \dots & B_{I-2,I-3} & B_{I-2,I-2} & B_{I-2,I-1} & 0 & & & \\
0 & \dots & \dots & 0 & B_{I-1,I-2} & B_{I-1,I-1} & A_0 & 0 & & \\
0 & \dots & \dots & 0 & 0 & A_2 & A_1 & A_0 & 0 & \dots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \dots & & & &
\end{array}\right]
$$

$\}$ **B**: matriz para los niveles iniciales

# Theorem

## Convergence Assurance

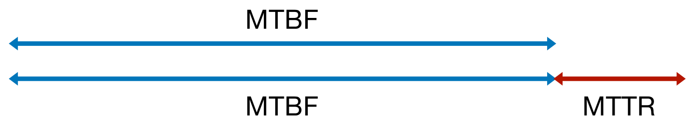
$\boldsymbol{\pi}_A$ **Theorem 1. The QBD process is stable if it holds** $\frac{\lambda}{NM\mu} < \frac{\alpha}{\nu+\alpha}$

- Proof: the drift of the system to the higher levels has to be less than the drift to the lower levels [1]

$$\boldsymbol{\pi}_A \mathbf{A}_0 \mathbf{1} < \boldsymbol{\pi}_A \mathbf{A}_2 \mathbf{1},$$

- Distribution Vector $\mathbf{A} = \mathbf{A}_0 + \mathbf{A}_1 + \mathbf{A}_2$

Interpretation: $\rho < \frac{\mathrm{MTBF}}{\mathrm{MTTR} + \mathrm{MTBF}}$




1. Marcel F. Neuts, «Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach», Dover Publications, 1995. ISBN: 978-0486683423

# Metrics

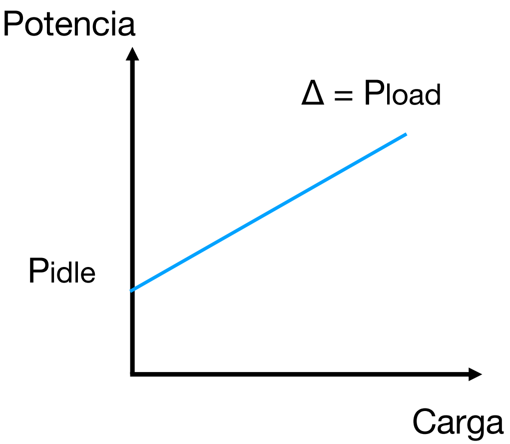
## Power Consumption (w)

- Markov chain solved

$$\pi_{i,j} \rightarrow \text{Prob. de } i \text{ tareas y } j \text{ servidores}$$

- Average power consumed

$$\omega = \sum_{(i,j)\in S} \pi_{i,j}\ j \left( P_{\text{idle}} + P_{\text{load}} \frac{\min(i, jN)}{jN} \right)$$



# Metrics

## Probability of failure (Pf)

- Assuming Probabilities ≈ 0

$$P_f = \boxed{P_{\text{wait}}} + \boxed{P_{\text{int}}}.$$

Indicator function

- Failure bc wait on arrival (PASTA)

$$\boxed{P_{\text{wait}}} = \sum_{(i,j)\in S} \mathbf{1}(i \geq jN)\pi_{i,j},$$

Task drop rate

Server crash rate

- Failure bc server crash

$$\boxed{P_{\text{int}}} = \frac{\gamma}{\lambda} \qquad \gamma = \sum_{(i,j)\in S} \pi_{i,j}\ j\nu\ F^r_{i,j},$$

$$F^r_{i,j} = \min(\max(\boxed{i} - \boxed{(j-1)N}, 0), N),$$

Number of Tasks Affected

Capacity after crash

# Validation

## Simulation parameters

Rack

8 Servers Dell Power Edge 32 GB
$P_{max}$ 270 W y $P_{idle}$ 150 W [1]
Boot up: 3 min [measured]
MTBF: 768 hours [2]

Nano

64 Servers Raspberry Pi 4b 4 GB
$P_{max}$ 7.6 W y $P_{idle}$ 4.6 W [3]
Boot up: 20 s [measured]
MTBF: 1/4 of the above

| Párametro | Servidores rack | Servidores nano |
|---|---|---|
| M | 8 servidores | 64 servidores |
| N | 32 tareas/servidor | 4 tareas/servidor |
| $1/\mu$ | 1 hora | 1 hora |
| $1/\alpha$ | 3 min. | 20 s. |
| $1/\nu$ | 32 días | 8 días |
| $P_{\text{idle}}$ | 150 W | 4.6 W |
| $P_{\text{load}}$ | 120 W | 3.0 W |

$$1/\mu = 1 \text{ h}$$

1. TPCDB: http://www.tpcdb.com/product.php?id=2325
2. G. L. Santos et al., «Analyzing the IT subsystem failure impact on availability of cloud services». ISCC 2017
3. TPCDB: http://www.tpcdb.com/product.php?id=4417

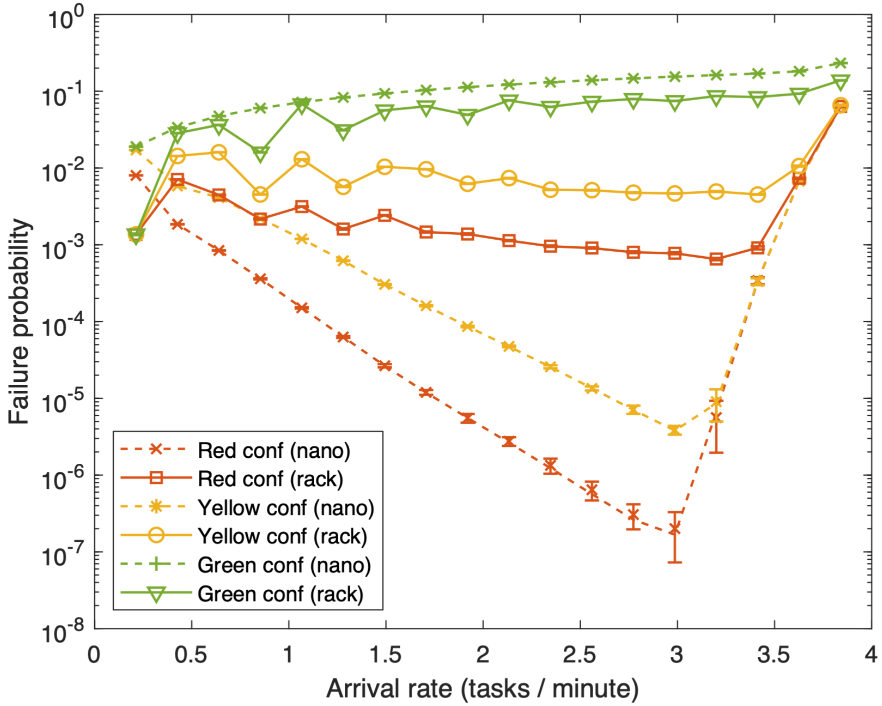
# Validation

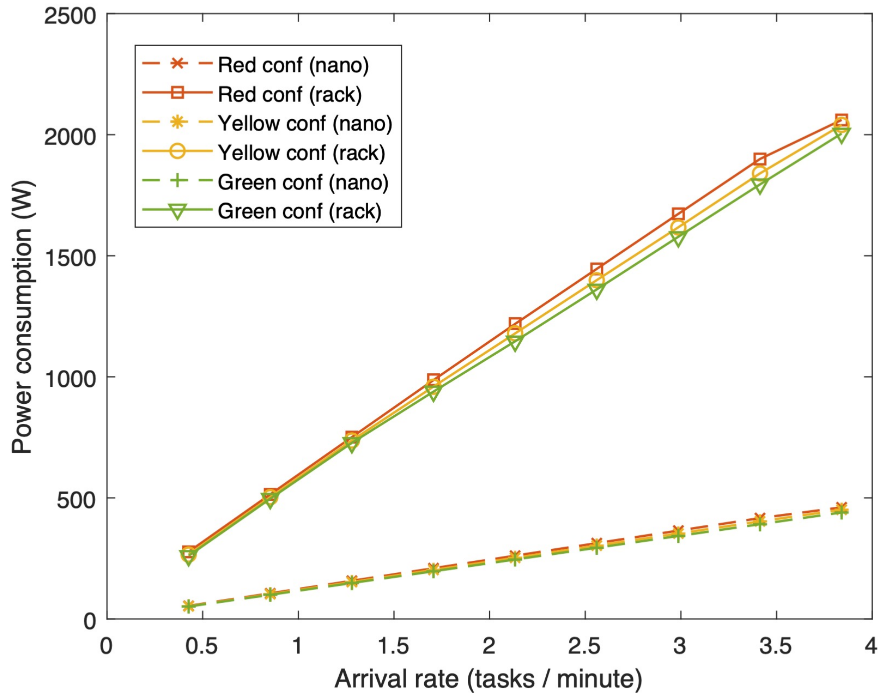
## Power On and Off Policies

| | | Example<br>Serv. rack (M=32) |
|---|---|---|
| **Green** | Turn on if all active servers are busy<br>Turn off if, doing so, leaves room for a task | ton(5) = 128<br>toff(5) = 127 |
| **Red** | Turn on when occupancy exceeds 95%<br>Turn off if occupancy would fall below 85% | ton(5) = 122<br>toff(5) = 108 |
| **Yellow** | Turn on when occupancy exceeds 95%<br>Turn off if occupancy would fall below 95% | ton(5) = 122<br>toff(5) = 121 |

# Results

## Failure probability & Power consumption



# Optimal configuration

## Formulation

- Model allows predicting performance for a given configuration
- Challenge: meeting a reliability criterion (prob. failure) and minimizing consumption

$$\min_{\{t_m^{\mathrm{on}}\},\{t_m^{\mathrm{off}}\}} \quad \omega$$

$$\text{sujeto a} \quad P_f \leq T_f$$

Bound for Pf
*Target failure probability*

- Problem: Large space size of possible combinations

# Optimal configuration (approach)

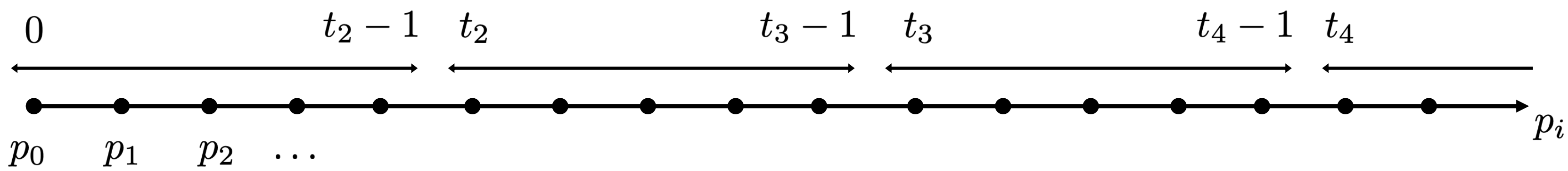
## Assumptions

- Threshold policy: $t_m^{\text{off}} = t_m^{\text{on}} - 1 \longrightarrow t_m^{\text{on}} \equiv t_m$
- Server crash and activation does not affect task distribution $\{p_i\}$
  - Behaves as a classical M/M/C, C = MxN
  - Power consumed as a function of thresholds

$$\omega_{m,t_m}^{\text{idle}} = \sum_{i=t_m}^{\infty} p_i P_{\text{idle}}$$

$$\omega = \sum_{m=1}^{M} \omega_{m,t_m}^{\text{idle}} + \sum_{i=1}^{\infty} p_i \min(i, C) P_{req}$$

- Decoupling the effect of each threshold on Pf:



# Optimal configuration (approach)

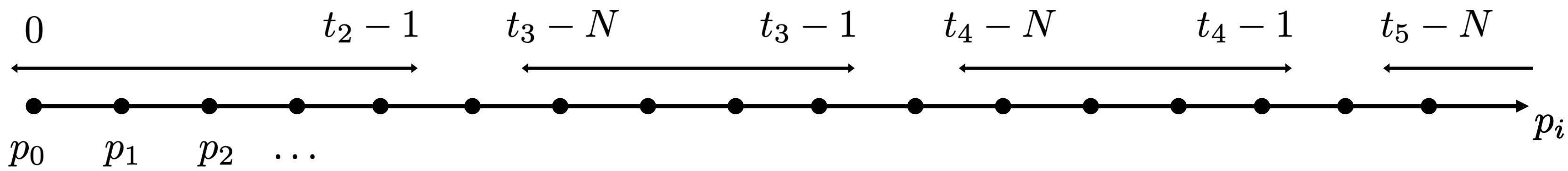
## Assumptions

- Threshold policy: $t_m^{\text{off}} = t_m^{\text{on}} - 1 \longrightarrow t_m^{\text{on}} \equiv t_m$
- Server crash and activation does not affect task distribution $\{p_i\}$
  - Behaves as a classical M/M/C, C = MxN
  - Power consumed as a function of thresholds

$$\omega_{m,t_m}^{\text{idle}} = \sum_{i=t_m}^{\infty} p_i P_{\text{idle}}$$

$$\omega = \sum_{m=1}^{M} \omega_{m,t_m}^{\text{idle}} + \sum_{i=1}^{\infty} p_i \min(i, C) P_{req}$$

- Decoupling the effect of each threshold on Pf:



# Computing the optimal configuration

## *Multiple-choice knapsack problem*

$$\min_{\{t_m^{\text{on}}\},\{t_m^{\text{off}}\}} \quad \omega$$

$$\text{sujeto a} \quad P_f \leq T_f$$

1 if the threshold 'm' is equal to 'k'

$$\min_{\{x_{m,k}\}} \quad \sum_{m=2}^{M} \sum_{k \in \mathcal{M}_m} x_{m,k}\,\omega_{m,k}^{\text{idle}}$$

$$\text{sujeto a} \quad \sum_{m=2}^{M} \sum_{k \in \mathcal{M}_m} x_{m,k} P_{m,k}^{f} \leq T_f$$

$$\sum_{k \in \mathcal{M}_m} x_{m,k} = 1, \quad \forall\, m = 2, \ldots, M$$

$$x_{m,k} \in \{0,1\}, \quad \forall\, m = 2, \ldots, M, k \in \mathcal{M}_m$$

**Post - optimization**

Threshold adjustment $t_m$

If Pf > Tf, lower it

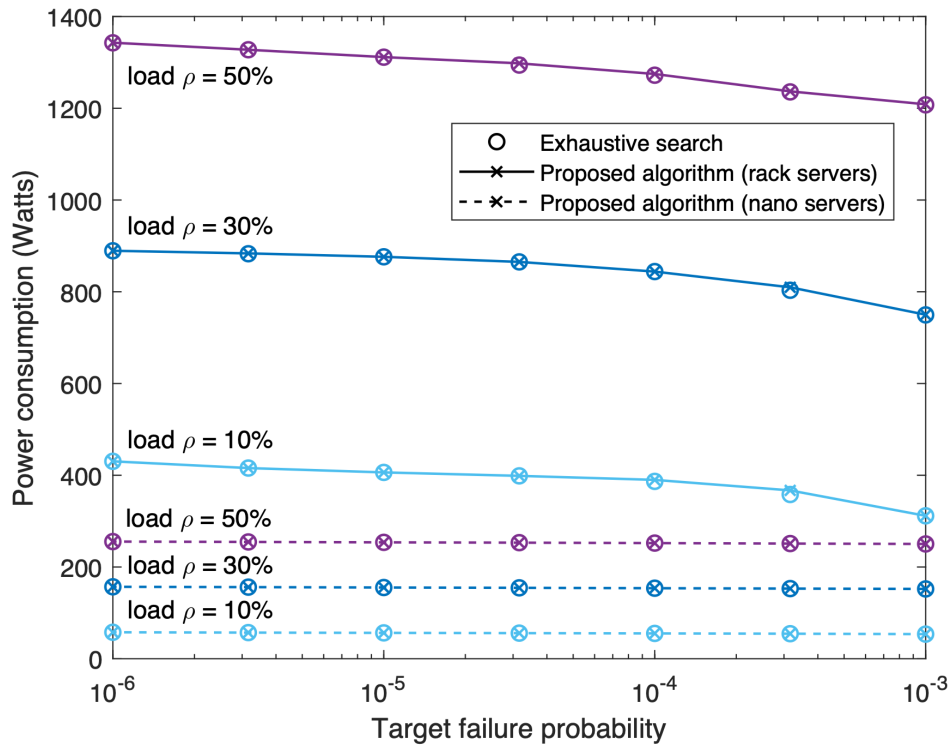
If Pf < Tf, increase it

# Evaluation

## Validation of the proposed configuration

- Different load and Tf values
- Complexity
  - Algorithm: 20 s -- 37 s
  - Search: 3 h (rack), 10 d (nano)

    Pseudo exhaustive
- Nano servers are more efficient



# Evaluation

## Comparison with other strategies


1. A. Beloglazov et al. «Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing», Future generation computer systems 28.5 (2012), págs. 755-768.
2. S. Telenyk et al. «Modeling of the Data Center Resource Management Using Reinforcement Learning», PIC S&T 2018

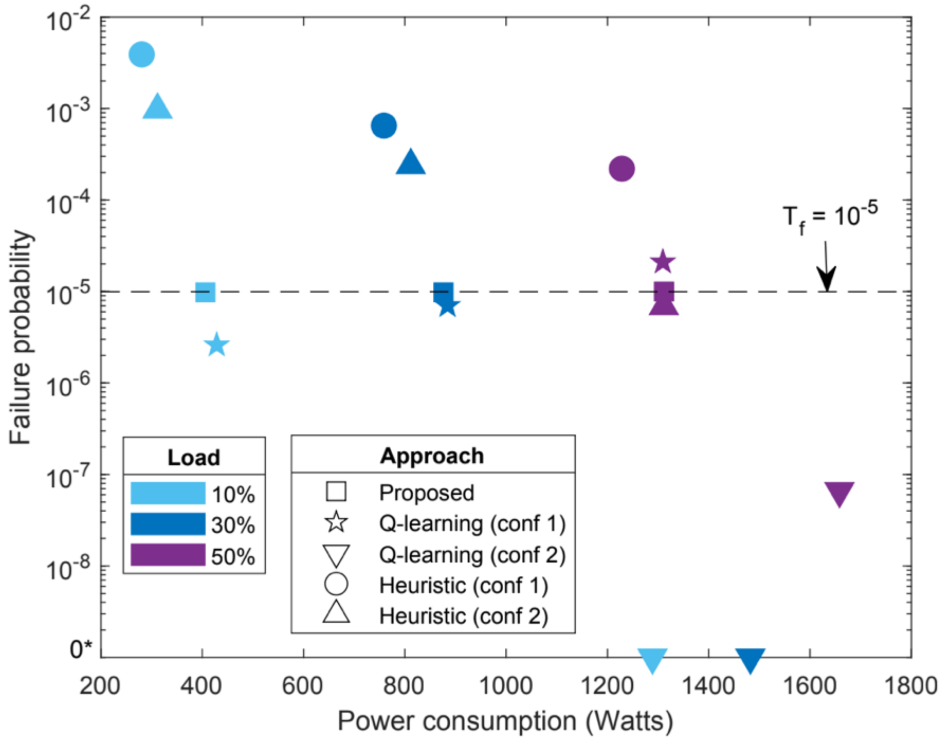

Heuristic [1]

Two threshold sets to de/activate
*Minimization of migrations* (MM)
Threshold: {50%, 90%} y {40%, 80%}

Q-learning [2]

State: # tasks, # servers
Actions: Power on/off/keep
Penalty: failures y consumption

$$p_t = \beta n_t + (1-\beta)\omega_t,$$
$$\beta = \{0.8, 1\}$$



# Dynamic optimization

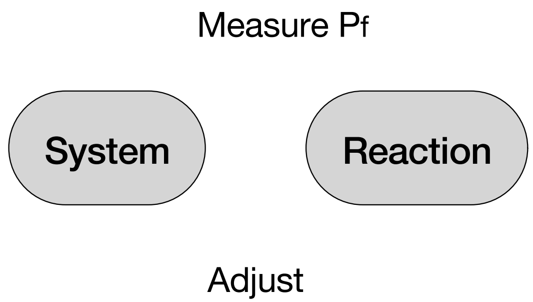
# Motivation

## Static vs. Dynamic

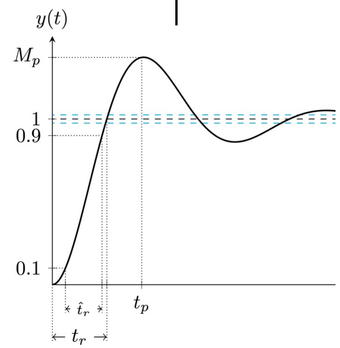
- Static optimization: calculation of thresholds a priori
  - Need to estimate parameters
  - Based on certain modeling hypotheses

Poisson Arrivals
Exponential service
Exponential boots

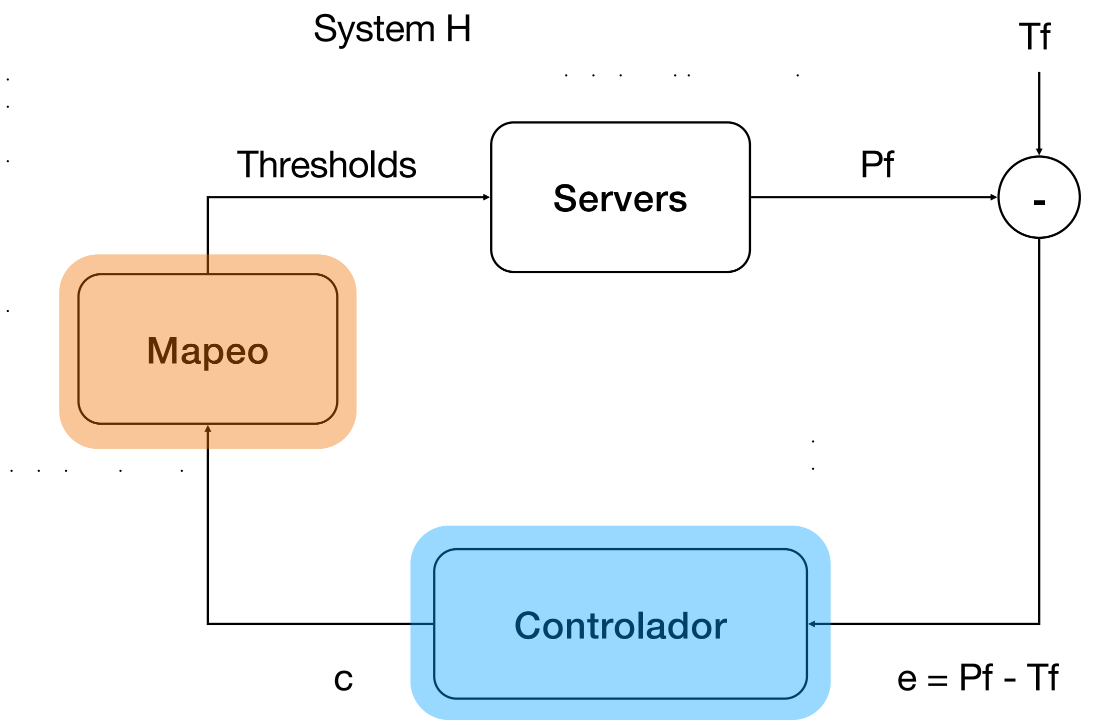
- Dynamic optimization: changing thresholds as a reaction
- Lower thresholds $\longrightarrow \downarrow P_f \uparrow \omega$
- Raise thresholds $\longrightarrow \uparrow P_f \downarrow \omega$




Control theory

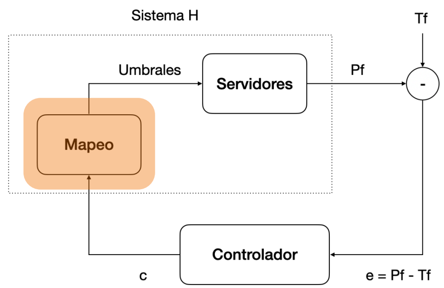


# System Design: A3S



# System Design: A3S

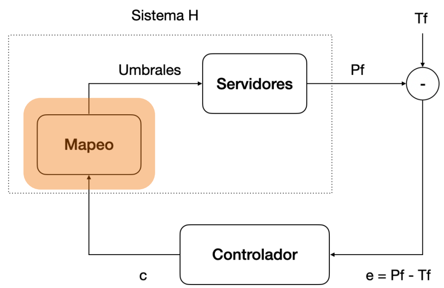
## Mapping: from ‘c’ to thresholds




- Threshold Policy $t_m^{\text{off}} = t_m^{\text{on}} - 1 \longrightarrow t_m^{\text{on}} \equiv t_m$
- Error $e = P_f - Tf$
    - Positive Error -> Lower Thresholds
    - Negative Error -> Raising Thresholds
- Minimum thresholds: all on
- Maximum thresholds: only turns on when full **(green)**

# System Design: A3S

## Mapping: from ‘c’ to thresholds




- Threshold Policy $t_m^{\text{off}} = t_m^{\text{on}} - 1 \longrightarrow t_m^{\text{on}} \equiv t_m$
- Error $e = P_f - Tf$
  - Positive Error -> Lower Thresholds
  - Negative Error -> Raising Thresholds
- Minimum thresholds: all on
- Maximum thresholds: only turns on when full **(green)**

**Example**
**M=4, N=3**

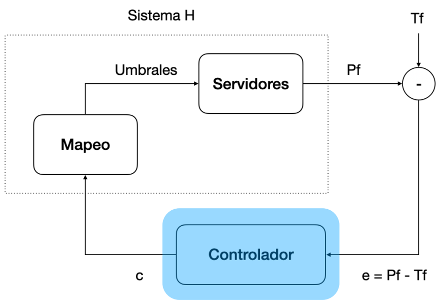
| c | t1 | t2 | t3 | t4 |
|---|---|---|---|---|
| **0** | 0 | 3 | 6 | 9 |
| **1** | 0 | **2** | 6 | 9 |
| **2** | 0 | 2 | **5** | 9 |
| **3** | 0 | 2 | 5 | **8** |
| **4** | 0 | **1** | 5 | 8 |
| **...** | ... | ... | ... | ... |
| **18** | **0** | **0** | **0** | **0** |

$$q = \left\lfloor \frac{c}{M-1} \right\rfloor, \qquad r = c - (M-1)\left\lfloor \frac{c}{M-1} \right\rfloor$$

$$t_m = \begin{cases} (m-1)M - q - 1, & \text{si } m \le r+1, \\ (m-1)M - q, & \text{en otro caso.} \end{cases}$$

# System Design: A3S

## Controller: Proportional Integral (PI)




- Control signal

$$c(t) = K_p e(t-1) + K_i \sum_{t'=0}^{t-2} e(t')$$

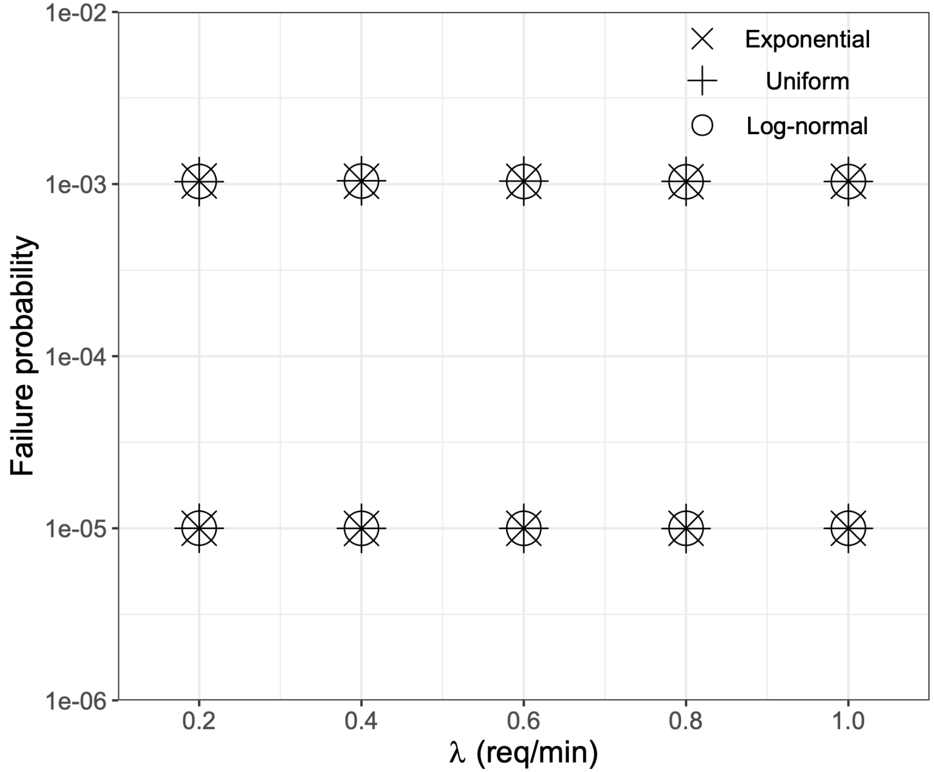
- Parameter values (Ziegler–Nichols) [1, 2]

$$K_p = \frac{0.4}{\hat{H}} \qquad K_i = \frac{0.4}{\hat{H} \cdot 0.85 \cdot 2} \qquad \text{donde } \hat{H} > |H| \qquad \hat{H} = \frac{T_f}{M-1}$$

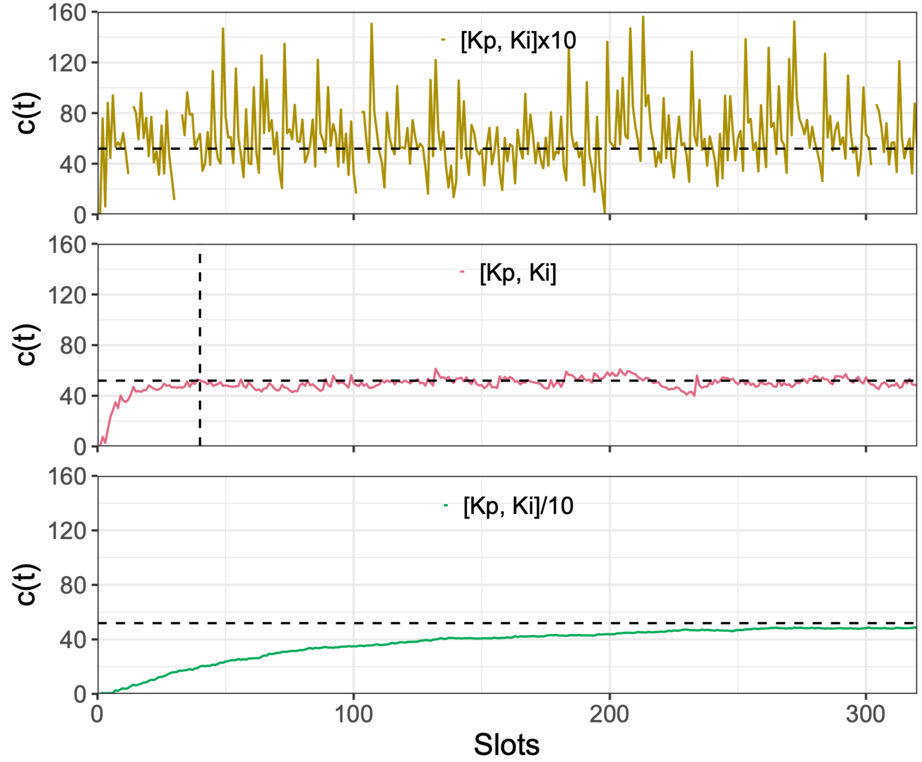
1. A. Garcia-Saavedra et al., «Adaptive Mechanism for Distributed Opportunistic Scheduling», IEEE Transactions on Wireless Communications, 2015
2. P. Serrano et al., «Control Theoretic Optimization of 802.11 WLANs: Implementation and Experimental Evaluation», Elsevier Computer Networks, 2013

# Validation

## Efficiency and Configuration




$$T_f = 10^{-3}$$
$$\lambda = 0.2 \text{ tareas/min}$$

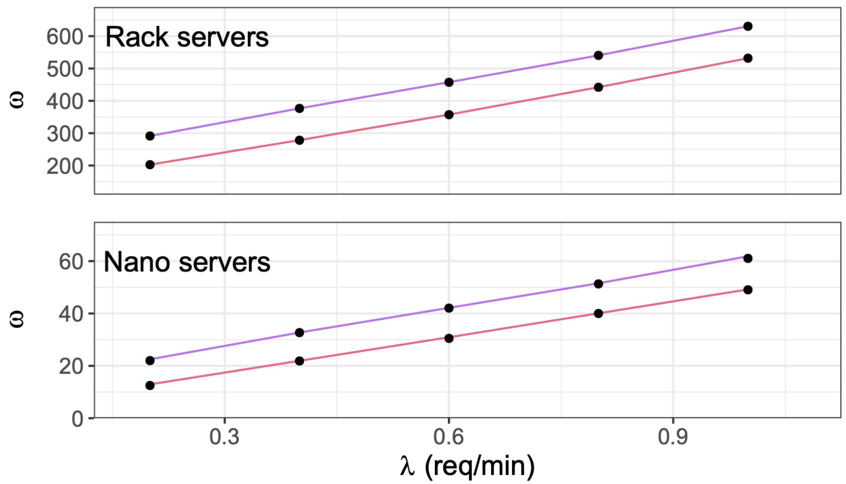


# Evaluation

## Vs. Search & Static Optimal

- Comprehensive threshold search
- Vs. Static configuration
    - Three 24-hour periods of Google Workflow trace [1]
    - M x N = 128
    - I = {30, 44, 64} Erlangs
    - $T_f = \{10^{-3}, 10^{-5}\}$

1. Google, «Workflow Trace Archive Google trace», Zenodo, Jun. 24, 2019. doi: 10.5281/zenodo.3254540.




| | A3S | | Static | |
|---|---|---|---|---|
| Load | $P_f$ | $\omega$ (W) | $P_f$ | $\omega$ (W) |
| Small | $10^{-3}$ | 516.1 | $0.3{\cdot}10^{-3}$ | 665.2 |
| | $10^{-5}$ | 687.4 | $0.8{\cdot}10^{-5}$ | 710.8 |
| Medium | $10^{-3}$ | 727.5 | $0.4{\cdot}10^{-3}$ | 773.5 |
| | $10^{-5}$ | 880.8 | $0.4{\cdot}10^{-5}$ | 942.3 |
| High | $10^{-3}$ | 1 020.3 | $0.5{\cdot}10^{-3}$ | 1 062.1 |
| | $10^{-5}$ | 1 212.8 | $0.8{\cdot}10^{-5}$ | 1 232.5 |

# Comparison

## Vs. Reinforcement learning (RL)

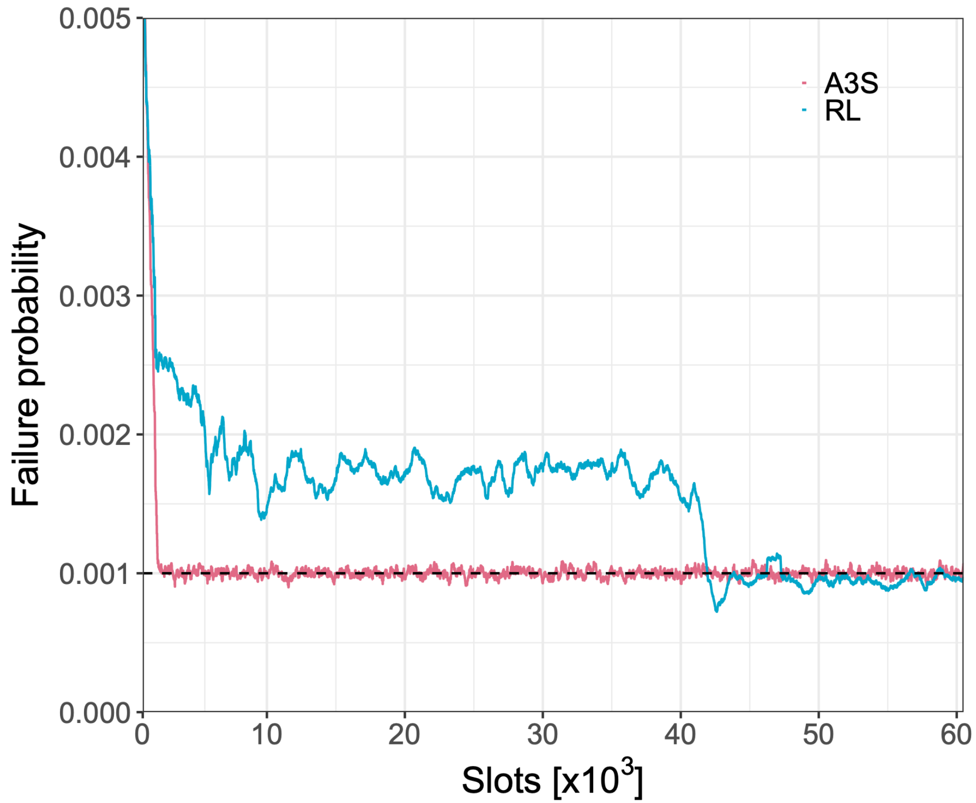
1. S. Telenyk et al. «Modeling of the Data Center Resource Management Using Reinforcement Learning», PIC S&T 2018

- Same technique as before [1]
- Status: # tasks, # servers
- Action: Turn on/off/hold
- Numerical search to achieve $T_f$

$$p_t = \beta n_t + (1-\beta)\omega_t$$

- Convergence: 90% $Q_{values}$

$$Q(s_t, a_t) = (1-\alpha)Q(s_t, a_t) + \alpha \left[ p_t + \gamma \min_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \right]$$



# Conclusiones y Trabajo futuro

## Conclusiones

- The softwarization of networks poses opportunities and challenges
- Sustainable operation but guaranteeing high performance
- Two solutions
- Static: Estimation
- Dynamic: adaptation
- Advantages vs. RL

## Trabajo futuro

- Other service models: heterogeneous requests
- Optimizing the Farm Design
- Support for different services with different requirements
- Impact of the transmission medium

# Additional information

## Publications

| | |
|---|---|
| Static optimization | • J. Ortín et al., «Analysis of scaling policies for NFV providing 5G/6G reliability levels with fallible servers», **IEEE Transactions on Network and Service Management (JCR Q2)**, Junio 2022 |
| Dynamic optimization | • J. Perez-Valero et al., «Energy-Aware Adaptive Scaling of Server Farms for NFV with Reliability Requirements», **IEEE Transactions on Mobile Computing (JCR Q1)**, Junio 2023 |
| Other scenarios | • J. Perez-Valero et al., «Performance Trade-offs of Auto Scaling Schemes for NFV with Reliability Requirements», **Computer Communications (JCR Q1)**, Diciembre 2023 |
| Design optimization | • J. Pérez-Valero et al. «Minimum-Cost Design of Auto-Scaling Server Farms Providing Reliability Guarantees», **IEEE Open Journal of the Communications Society (JCR Q1)**, Julio 2025 |