# Queueing analysis of imbalance between server pools

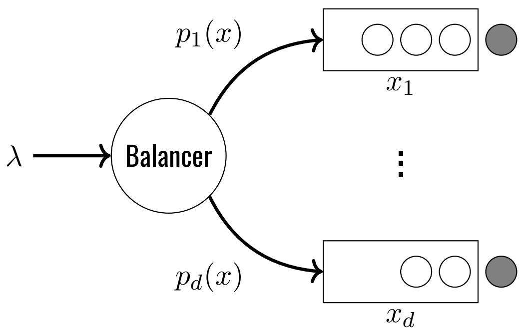
## with applications to 3-phase EV charging

Andres Ferragut and Fernando Paganini

Universidad ORT Uruguay

IFIP Performance 2023 – Northwestern University – November 2023

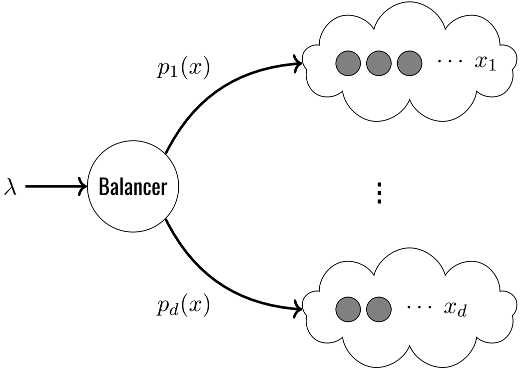
# Outline




- Parallel system with single server queues.
- Balancer attempts to lower queue occupation (delay).
- Main goal: stability, fluid and diffusion level optimality.
- Well studied problem: join the shortest queue, power of $d$, join the idle queue...
- See [Van der Boor et al. 2022].

# Server pool load balancing

Consider a slightly different problem: Balancing between parallel server pools.




- Each pool has many servers.
- Tasks are served in parallel in a dedicated server for each one.
- Main goal: keep occupations balanced between pools.

# Motivation

- Cloud micro-services architectures (e.g. Kubernetes):
  - Tasks are served in containers or pods.
  - Each container is ran in a physical node (the server pool) up to capacity.
  - Keeping a balanced number of active pods lowers server utilization.

# Motivation

- Cloud micro-services architectures (e.g. Kubernetes):
  - Tasks are served in containers or pods.
  - Each container is ran in a physical node (the server pool) up to capacity.
  - Keeping a balanced number of active pods lowers server utilization.
- EV charging in the grid:
  - Grid connection is 3-phase AC.
  - Each charger is monophasic: thus we have 3 server pools.
  - Keeping total current draw balanced minimizes impact on infrastructure.

# Our contributions

- We define a suitable imbalance metric for the system, with practical interest.

# Our contributions

- We define a suitable imbalance metric for the system, with practical interest.
- Then we characterize the baseline imbalance in a large scale system.

# Our contributions

- We define a suitable imbalance metric for the system, with practical interest.
- Then we characterize the baseline imbalance in a large scale system.
- We show that finite-size systems with random routing show less imbalance, and provide suitable bounds valid for any system size and load.

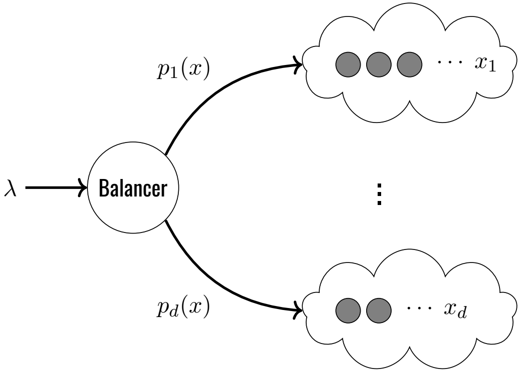
# Our contributions

- We define a suitable imbalance metric for the system, with practical interest.
- Then we characterize the baseline imbalance in a large scale system.
- We show that finite-size systems with random routing show less imbalance, and provide suitable bounds valid for any system size and load.
- We analyze the route to the least loaded pool policy and provide hard bounds on the average imbalance, improving over the random routing scenario.

# Our contributions

- We define a suitable imbalance metric for the system, with practical interest.
- Then we characterize the baseline imbalance in a large scale system.
- We show that finite-size systems with random routing show less imbalance, and provide suitable bounds valid for any system size and load.
- We analyze the route to the least loaded pool policy and provide hard bounds on the average imbalance, improving over the random routing scenario.
- As application, we show that active routing in an EV parking lot can improve current balance.

# System model

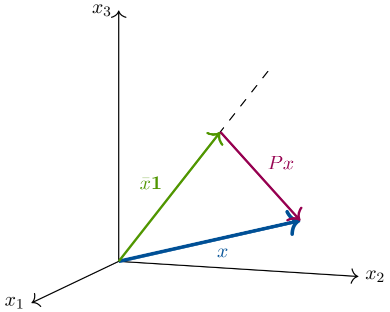



- Tasks arrive as a Poisson process of intensity $\lambda$.
- Each task has $\exp(\mu)$ service time.
- $A := \frac{\lambda}{\mu}$ is the total traffic intensity.
- There are $d$ pools, each pool with $C_i$ servers (possibly infinite).
- Let $x_i$ the number of tasks at pool $i = 1, \ldots, d$.

# Imbalance metric

- The state of the system is $x = (x_1, \ldots, x_d)$.
- A perfectly balanced occupation state is such that $x = \bar{x}\mathbf{1}$ where $\bar{x} = (1/d)\sum_i x_i$.
- Define:

$$Px := x - \bar{x}\mathbf{1} = \left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^T\right)x$$



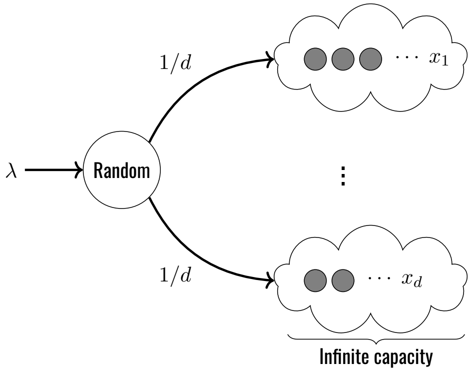

## Imbalance metrics:

$$J_1^{imb} := E[||Px||], \qquad J_2^{imb} := E[||Px||^2]$$

in steady state.

# Infinite server with random routing

## Baseline case




- Just $d$ $M/G/\infty$ server queues in parallel
- Product form steady state:

$$\pi(x) = e^{-A} \prod_{i=1}^{D} \frac{(A/d)^{x_i}}{x_i!}.$$

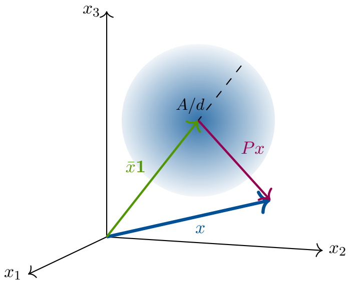
- Simple calculations:

$$J_2^{imb} = E[||Px||^2] = \frac{d-1}{d} A.$$

- Insensitive to sojourn times.

# Infinite server with random routing

## Large scale imbalance




- Let now $A \to \infty$ (large scale system).
- Poisson distribution converges to a Gaussian.
- $||Px||^2$ is the norm of a Gaussian projection.

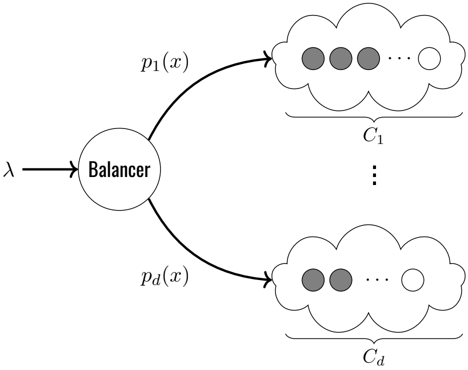
### Proposition (Infinite server, large scale imbalance)

In the infinite server case, as $A \to \infty$:

$$\frac{d}{A}||PX||^2 \overset{w}{\Longrightarrow} \chi^2_{d-1},$$

# Finite size system

Consider now the case with finite capacity in each pool.




- Free spaces routing:

$$p_i(x) = \frac{C_i - x_i}{C - \sum_i x_i}$$

- Routing at random, proportional to free spaces.
- Important application: EV parking, pools are phases.

# Free spaces routing

Markov chain model:

$$\begin{cases} x \mapsto x + e_i : & \lambda_i(x) := \lambda \left[ \dfrac{C_i - x_i}{C - n} \right] \\ x \mapsto x - e_i : & \mu_i(x) := \mu x_i \end{cases}$$

with $n := \sum_{i=1}^{d} x_i$.

# Free spaces routing

Markov chain model:

$$\begin{cases} x \mapsto x + e_i : & \lambda_i(x) := \lambda \left[ \dfrac{C_i - x_i}{C - n} \right] \\ x \mapsto x - e_i : & \mu_i(x) := \mu x_i \end{cases}$$

with $n := \sum_{i=1}^d x_i$.

Key point: the above chain admits a product form $\pi(x) = \pi(0)\Lambda(x)\Phi(x)$ since it is a balanced allocation [Bonald et al. 04], with:

$$\Lambda(x) := \lambda^n \frac{(C-n)!}{C!} \prod_{i=1}^d \frac{C_i!}{(C_i - x_i)!}, \quad \Phi(x) := \frac{1}{\mu^n \prod_{i=1}^d x_i!}.$$

## Theorem

The equilibrium occupancy distribution for the free spaces routing policy is given by:

$$\pi(x) = \pi(0)\frac{A^n}{n!}\frac{\prod_{i=1}^{d}\binom{C_i}{x_i}}{\binom{C}{n}},$$

for $0 \leq x_i \leq C_i$, $i = 1, \ldots, d$ and $\pi(0) = \frac{1}{\sum_{j=0}^{C} A^j/j!}$.

- Total number of tasks as an $M/M/C/C$ (Erlang) queue.
- Given $N = n$, all possible subsets of size $n$ become equiprobable (multivariate hypergeometric distribution).

# Free spaces routing

## Imbalance in steady state

Consider now the case $C_i = C/d$ for all $i$ (homogeneous pools).

### Proposition

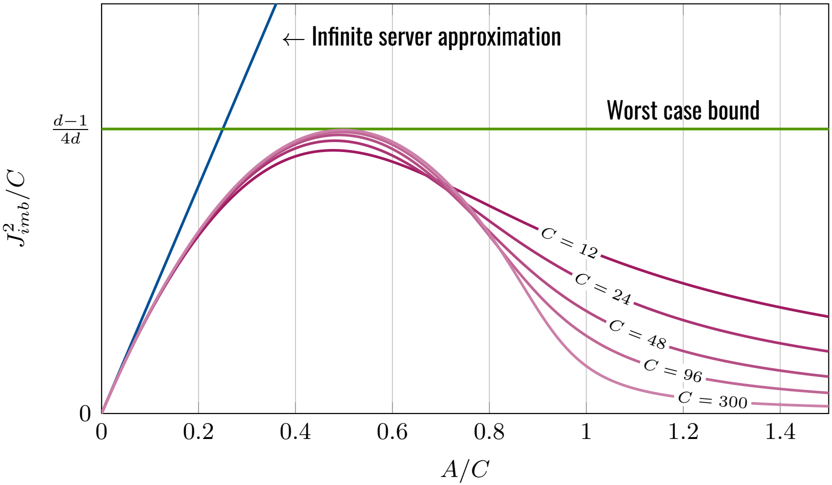
For the free spaces routing policy with homogeneous pools capacity $C_i = C/d$, in steady state:

$$J_{imb}^2 = E\left[||Px||^2\right] \leqslant \frac{d-1}{4d}\frac{C^2}{C-1}.$$

- Proof idea: condition on $N = n$ and use the variance of the hypergeometric distribution.
- Remark: since the system admits a product form, the result holds for general sojourn times.

# Free spaces routing

## Example



# Free spaces routing

## An explanation

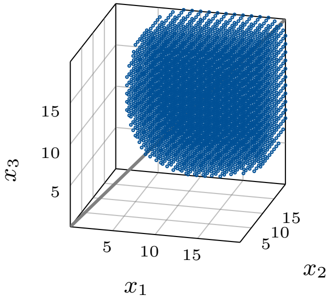
Why imbalance is lower that the infinite server approximation?

# Free spaces routing

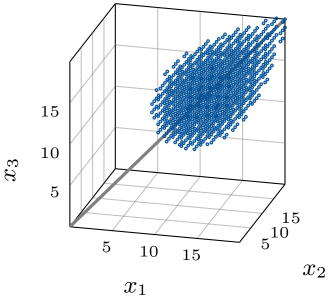
## An explanation

Why imbalance is lower that the infinite server approximation? $\rightarrow$ Because the routing choice naturally tends to balance the system by giving higher probability to the least loaded pool.

Why imbalance is lower that the infinite server approximation? $\rightarrow$ Because the routing choice naturally tends to balance the system by giving higher probability to the least loaded pool.




Infinite server case, $A = 40$.




Free spaces routing, $A = 40, C = 60$.

# Application: EV charging

Consider an EV charging parking lot fed by three-phase AC power.

- Each charger is connected to two of the three phases $\to d = 3$ pools.
- Installation is balanced, $C_i = C/3$ by design.
- If phase occupation is unbalanced $\to$ electrical inefficiencies (overcurrents) appear.
- Arriving cars choose a spot at random oblivious to its phase connection.

# Application: EV charging

Consider an EV charging parking lot fed by three-phase AC power.

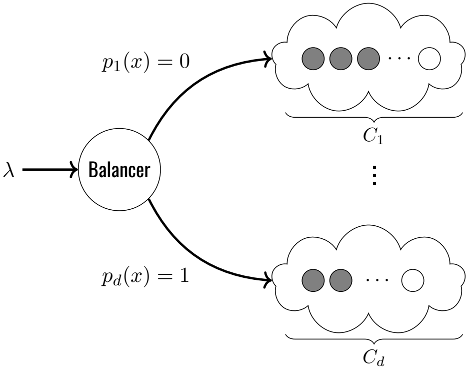
- Each charger is connected to two of the three phases $\to d = 3$ pools.
- Installation is balanced, $C_i = C/3$ by design.
- If phase occupation is unbalanced $\to$ electrical inefficiencies (overcurrents) appear.
- Arriving cars choose a spot at random oblivious to its phase connection.

## Remark

Corresponds to a free spaces routing system! The bounds enables us to compute a relevant measure of imbalance, valid for any sojourn time distribution and any traffic load!

# Actively routing to the least-loaded-pool

To further reduce imbalance, a more proactive policy is needed → join the least loaded pool.




- Let $k(x) := \#\{\arg\min(x_i)\}$, and take

$$p_i(x) = \begin{cases} \frac{1}{k(x)} : & \text{if } i \in \arg\min(x_i) \\ 0 & \text{otherwise.} \end{cases}$$

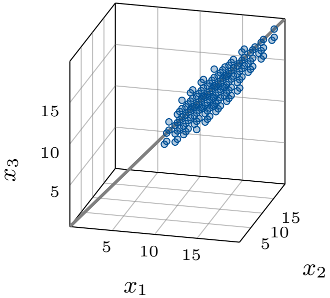
- Idea: Split arrivals equally among the pools with minimum occupation.
- Blocking occurs only at $x = \frac{C}{d}\mathbf{1}$, i.e. when the entire system is full.

Assuming exponential sojourn times, we can go back to the Markov chain:

- Recall $k(x) := \#\{\arg\min(x_i)\}$:

$$\begin{cases} x \mapsto x + e_i : & \lambda \frac{1}{k(x)} \mathbf{1}_{\{i \in \arg\min x_i\}}, \\ x \mapsto x - e_i : & \mu x_i. \end{cases}$$

- **Idea:** trend is to the diagonal, which is what we want.
- **Problem:** no explicit form.




Invariant distribution (numerical) $A = 40, C = 60$.

# Imbalance in least loaded pool

## Lyapunov approach

Take the Lyapunov function:

$$V(x) = (d-1)V_1(x) + V_2(x) = (d-1)\underbrace{\left[\frac{1}{d}\sum_{i=1}^{d} x_i\right]}_{\bar{x}} + \left[||Px||^2\right]$$

Then we can compute the drifts:

$$QV_1(x) = \lambda/d - \mu\bar{x}$$

$$QV_2(x) = 2\lambda\left[\min_i x_i - \bar{x}\right] - 2\mu\left[\sum_i x_i^2 - d\bar{x}^2\right] + \frac{d-1}{d}\lambda + (d-1)\mu\bar{x}.$$

and using Cauchy-Schwarz and the LLP property we get the following bound:

$$QV(x) \leqslant 2\lambda\left(\min_i x_i - \bar{x}\right) + 2\lambda\frac{d-1}{d}.$$

Taking expectations in steady state we get:

$$E\left[\bar{X} - \min_i X_i\right] \leqslant \frac{d-1}{d}.$$

**Useful lemma:** For $x \in \mathbb{R}^d$, $\|Px\| \leqslant \sqrt{d^2 - d}\,(\bar{x} - \min_i x_i)$.

Taking expectations in steady state we get:

$$E\left[\bar{X} - \min_i X_i\right] \leqslant \frac{d-1}{d}.$$

Useful lemma: For $x \in \mathbb{R}^d$, $\|Px\| \leqslant \sqrt{d^2 - d}\,(\bar{x} - \min_i x_i)$.

## Theorem

Under LLP with $d$ server pools, for any value of $\lambda$, $\mu$, in steady state:

$$J^1_{imb} = E[||PX||] \leqslant (d-1)\sqrt{1 - \frac{1}{d}}.$$

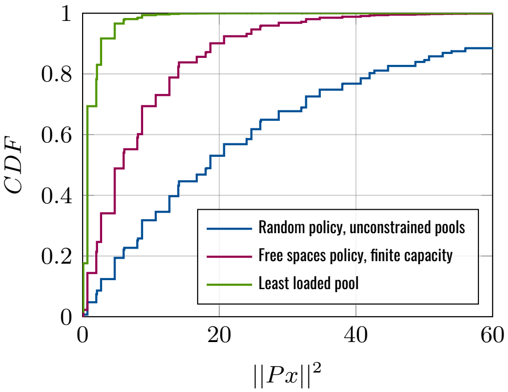
# Least loaded pool

## Remarks

- The bound only depends on $d$!
- $J^1_{imb}$ remains uniformly bounded for any system load and any system size (state space collapse).
- In relative terms, imbalance decays as $O(1/A)$ with system size,
- For random/free spaces routing, we only get $O(1/\sqrt{A})$.

- The bound only depends on $d$!
- $J_{imb}^{1}$ remains uniformly bounded for any system load and any system size (state space collapse).
- In relative terms, imbalance decays as $O(1/A)$ with system size,
- For random/free spaces routing, we only get $O(1/\sqrt{A})$.

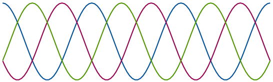
Example: $d = 3$, $A = 40$, $C = 60$.



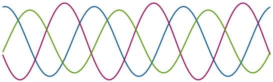
# Application: imbalance in power systems

- Power is delivered by $3-$phase AC. Loads are connected in between phases.
- If current draw differs in each phase, we have electrical imbalance.

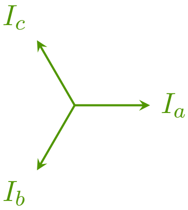
Balanced:

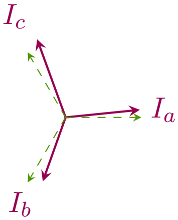


Unbalanced:



Phasor representation:



# Imbalance in EV charging facilities

- In EV charging, imbalance occurs in amplitude, not in phase.
- Due to different current draws because of imbalanced occupation of phases...

If $x = (x_1, x_2, x_3)$ is the phase occupation vector, and each charging vehicle draws a current $I_0$, the IEEE imbalance metric is just:

$$E\left[|I^-|\right] = \frac{I_0}{\sqrt{2}} E\left[\|PX\|\right] = \frac{I_0}{\sqrt{2}} J^1_{imb}.$$

Therefore, we can use our previous models to evaluate system imbalance!

# Free spaces routing

If users choose the parking space at random, then the probability of choosing a free spot from phase $i$ is exactly $(C_i - X_i)/(C - N)$, i.e. the free spaces routing policy. Therefore:

## Proposition

For a parking lot with the random free spaces routing policy, in steady state we have:

$$E\left[|I^-|\right] = \frac{I_0}{\sqrt{2}} J^1_{imb} \leqslant \frac{I_0}{\sqrt{2}} \sqrt{J^2_{imb}} \leqslant \frac{I_0}{2\sqrt{3}} \frac{C}{\sqrt{C-1}},$$

for any traffic load $A$.

The relative amount of imbalance satisfies:

$$\frac{1}{I_0 C} E[|I^-|] \leqslant \frac{1}{2\sqrt{3}} \frac{1}{\sqrt{C-1}} \underset{C\to\infty}{\sim} O\left(\frac{1}{\sqrt{C}}\right).$$

# Active routing

## Least-loaded-phase policy

A better result can be obtained if we can direct users to the least-loaded.phase!

### Proposition

For a parking lot actively routing vehicles to the least loaded phase, in steady state we have:

$$E\left[|I^-|\right] = \frac{I_0}{\sqrt{2}} J^1_{imb} \leqslant \frac{2}{\sqrt{3}} I_0$$

for any traffic load $A$ and system size $C$.

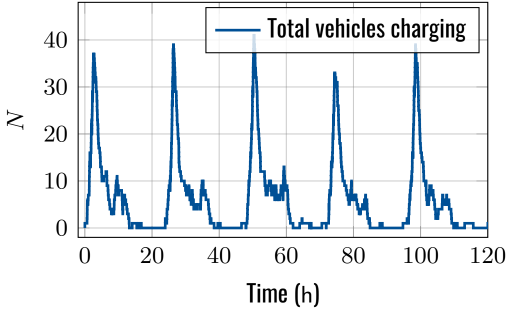
In this case, the relative imbalance decays much faster:

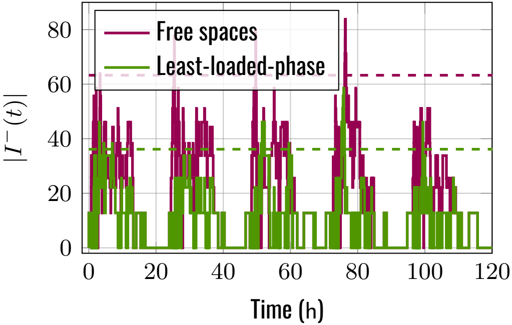
$$\frac{1}{I_0 C} E[|I^-|] \leqslant \frac{2}{\sqrt{3}C},$$

and thus further reducing the strain on the installation.

# Numerical experiments on a real parking lot

- The queueing analysis is steady state, but parking occupation goes through daily cycles.
- Our estimates are independent of the offered traffic $A$, they should be useful for system design!



# Take home messages...

- We analyzed the problem of load balancing between parallel server systems, and defined a suitable metric of imbalance with practical applications.
- Discussed natural random and active policies for load balancing, and obtained estimates of imbalance in several cases.
- The estimates are agnostic to offered load, so they help with system design.

# Take home messages...

- We analyzed the problem of load balancing between parallel server systems, and defined a suitable metric of imbalance with practical applications.
- Discussed natural random and active policies for load balancing, and obtained estimates of imbalance in several cases.
- The estimates are agnostic to offered load, so they help with system design.

## Future work:

- Analyze policies with less communication overhead, such as Power of $d$. Helpful in data-center scenarios.
- Refine the bounds in the LLP case.

# Thank you!

Andres Ferragut
ferragut@ort.edu.uy
http://aferragu.github.io