

Optimizing Bucket-Based Software Rejuvenation in Microservices Subject To Variable Workload

Leonardo Scommegna¹, Alberto Avritzer², Laura Carnevali¹,
Andrea Janes³, Helena Rodrigues⁴, Enrico Vicario¹

¹University of Florence, Italy ²EsulabSolutions Inc., USA

³Free University of Bozen-Bolzano, Italy ⁴University of Minho, Portugal

WoSAR'26, Limassol, October 2026

This is about:

- Software Aging and Rejuvenation in Microservices
- Bucket-Based Rejuvenation under Variable Workload
- GSPN Model with Bernstein Phase-Type Distributions
- Guidelines for Bucket Configuration



**Software
Technologies
Laboratory**

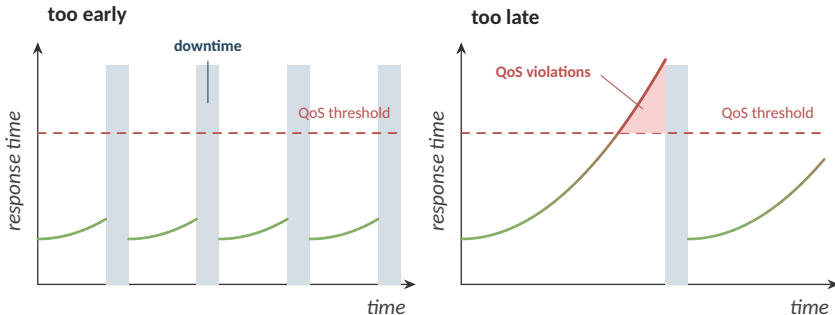


UNIVERSITÀ
DEGLI STUDI
FIRENZE

DINFO

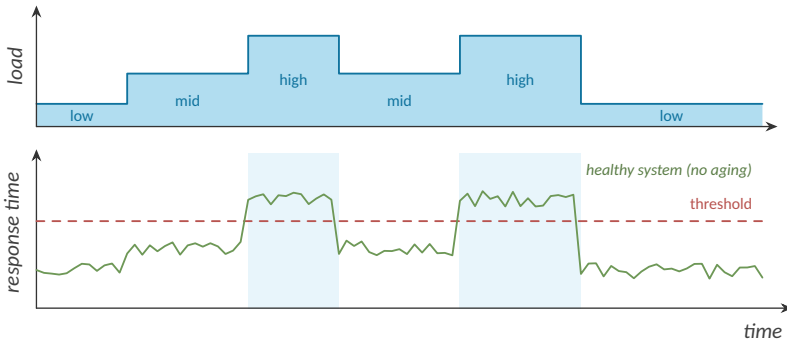
DIPARTIMENTO DI
INGEGNERIA DELL'INFORMAZIONE

SAR Challenge: When to Rejuvenate?



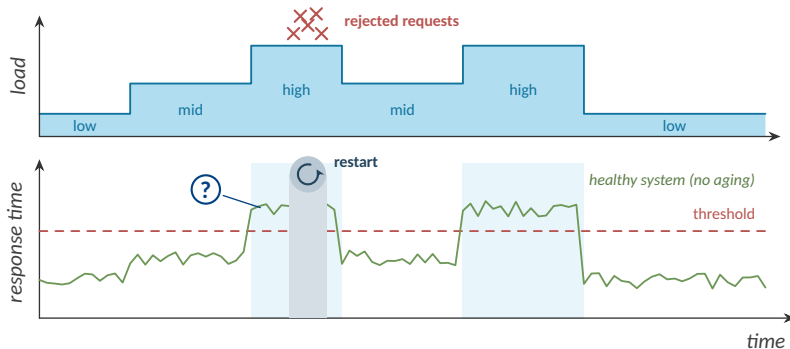
- Too early → **availability** loss
- Too late → **QoS violations** and failures

Time-Varying Workload



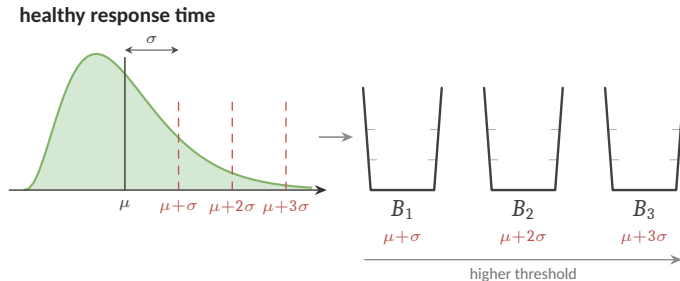
- Load influences **aging**: **time-triggered** rejuvenation cannot adapt
- Response times grow with **load**, even in a **healthy** system

Aging or Load?



- Load-induced latency mistaken for **aging**
- **Unnecessary rejuvenation** when demand is highest

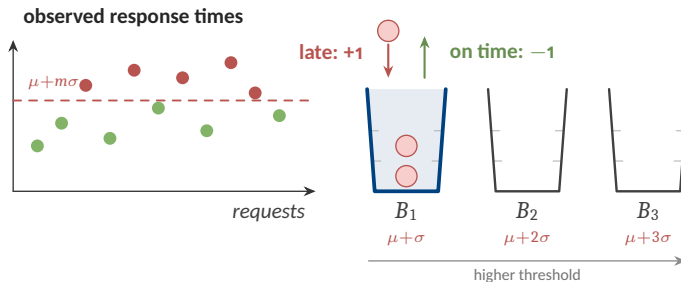
Bucket-Based Monitoring



- **Healthy** response time: mean μ , std. dev. σ
- M buckets¹: B_m with depth D_m and threshold $\mu + m\sigma$
- $M \rightarrow \# \text{buckets}$; $D \rightarrow \text{depth of buckets}$

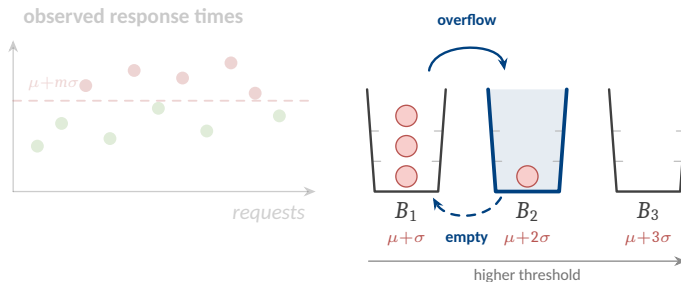
¹ Avritzer, Bondi, Grottko, Trivedi, Weyuker "Performance Assurance via Software Rejuvenation: Monitoring, Statistics and Algorithms" DSN 2006

Threshold Violations



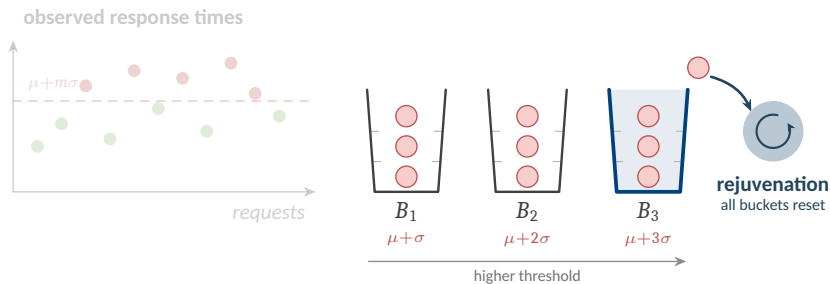
- **Late response** ($> \mu + m\sigma$): **add** a token
- **On-time response**: **remove** a token

Escalation



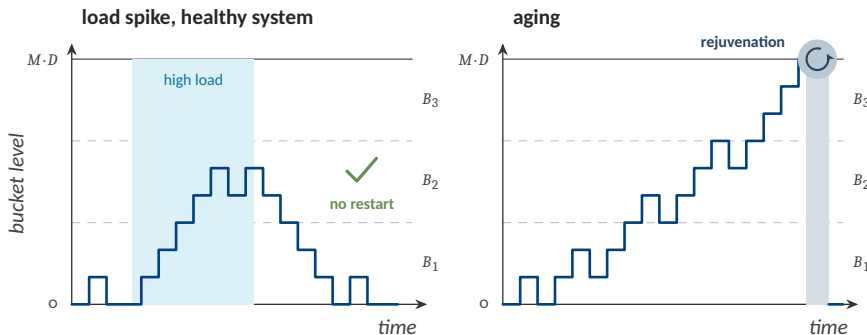
- B_m **overflows**: move up to B_{m+1} (higher threshold)
- B_m **empties**: move back to B_{m-1} (lower threshold)

Rejuvenation Trigger



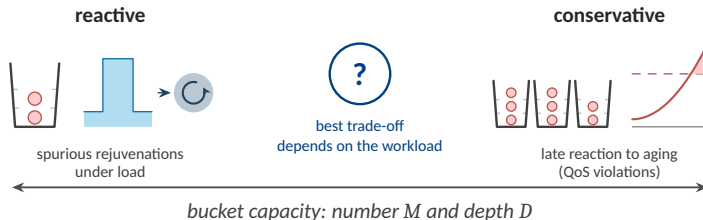
- B_M overflows: **rejuvenation** is triggered

Transient vs. Persistent Violations



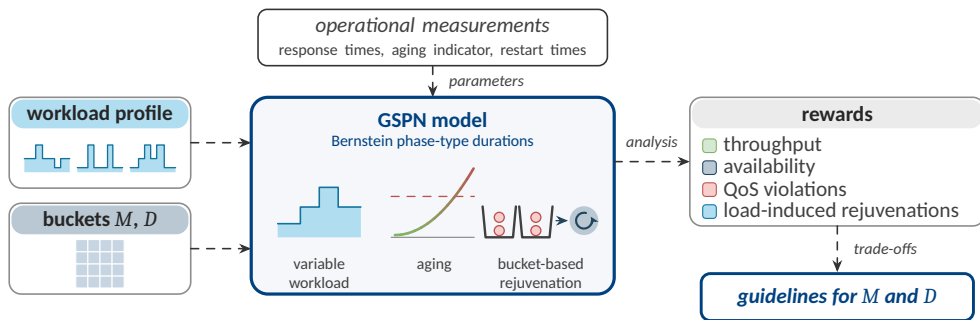
- **Transient** violations (load spikes) are **reabsorbed**
- **Persistent** violations (aging) trigger **rejuvenation**
- **Capacity** $Z = M \cdot D$: late minus on-time completions tolerated before rejuvenation

Problem Statement



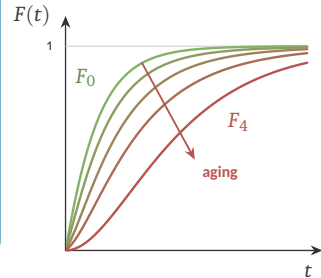
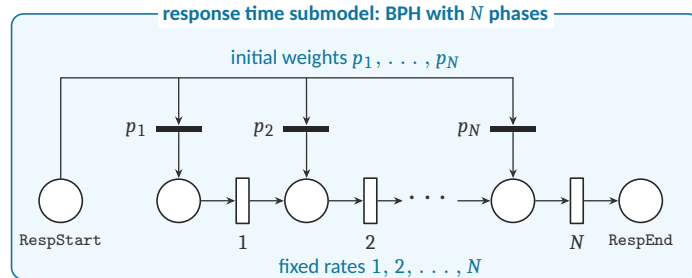
- Effectiveness **depends** on:
 - Number M and depth D of buckets
 - Workload profile
- **Challenge:** configuring M and D to react to **aging**, not to **load**

Aim of the Work



- **GSPN model** with **BPH** durations, parameterized from operational **measurements**
- Impact of the **workload profile** and of M, D on the rewards → **guidelines**

Response Time Submodel

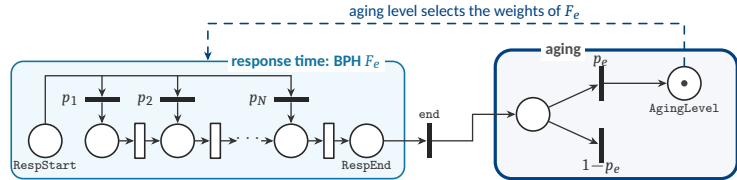


same phases, different weights

- **Bernstein Phase-Type (BPH):** chain of N exponential phases²
- Only the **initial weights** p_n change the distribution

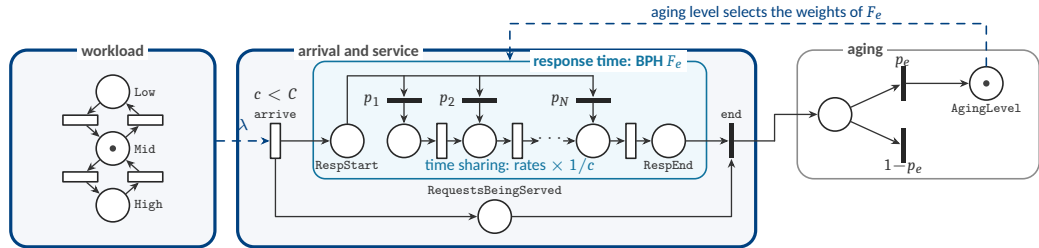
² Horváth, Horváth, Paolieri, Telek, Vicario "Approximation of Cumulative Distribution Functions by Bernstein Phase-Type Distributions" Perform. Eval. 2025

Aging



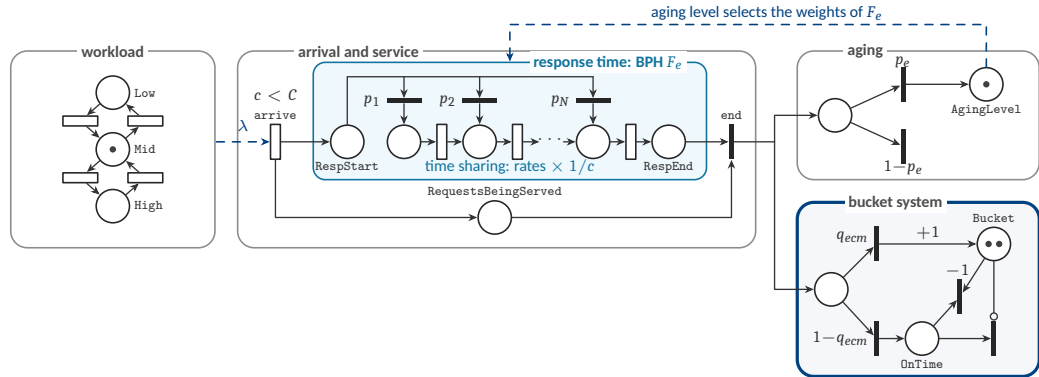
- Aging grows **per served request**: faster under high load
- Aging level L_e selects the response-time BPH F_e

Load and Concurrency



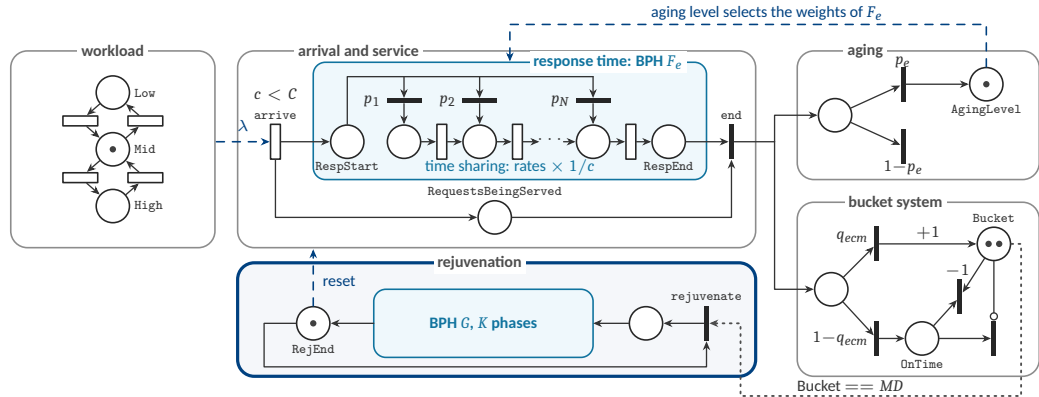
- Arrivals modulated by the **workload** regime (MMPP)
- **Time sharing:** phase rates scaled by $1/c$, at most C requests

Bucket System Submodel



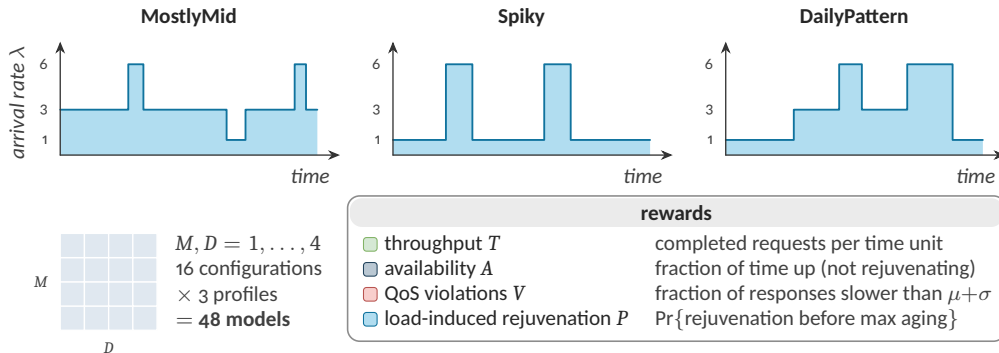
- Predefined threshold violation: $q_{ecm} = 1 - F_e((\mu + m\sigma)/c)$
- **Depending on:** Aging level e , concurrency c , current bucket m

Rejuvenation



- Bucket system full $\rightarrow$ **rejuvenation**, duration BPH G
- Microservice **reset**

Experimental Setup

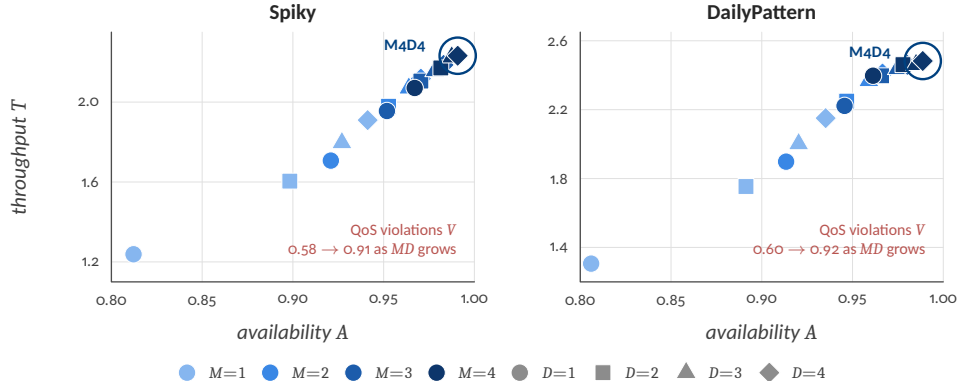


- $E = 4$ aging levels, BPH with $N = K = 5$ phases, $C = 4$
- Rewards from the **transient analysis** between consecutive rejuvenations
- model solved with ORIS / Sirio³

³Paolieri, Biagi, Carnevali, Vicario "The ORIS Tool: Quantitative Evaluation of Non-Markovian Systems" IEEE TSE 2021

M and D impact on Throughput Availability and QoS

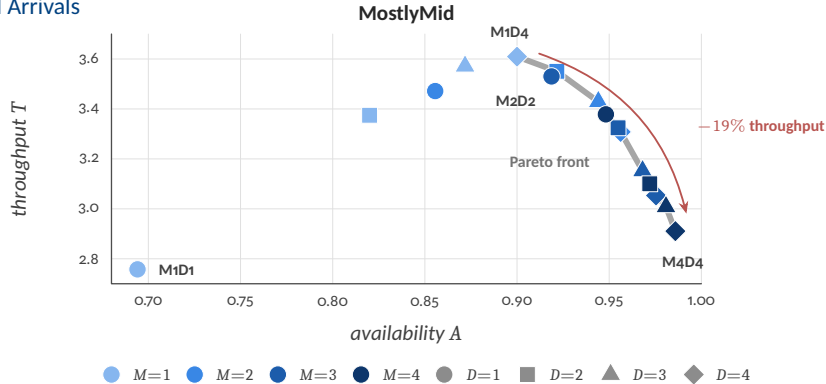
Intermittent Arrivals



- **Throughput** bounded by arrivals not by service
- Aging is not avoided: **QoS violations** grow with capacity $M \cdot D$

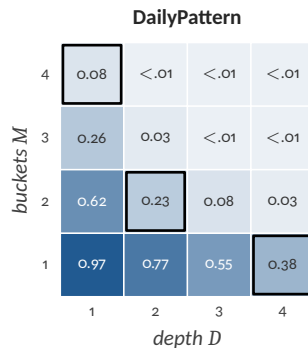
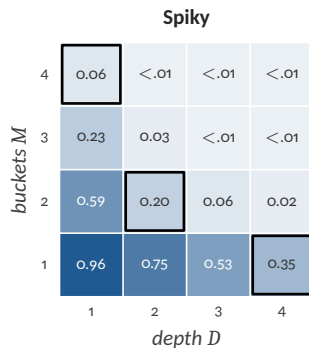
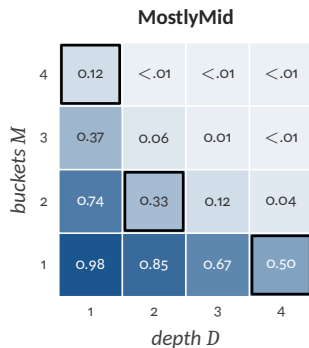
M and D impact on Throughput Availability and QoS

Sustained Arrivals



- Real **trade-off**: 11 Pareto-optimal configurations
- Throughput peaks at **intermediate** capacity $MD == 4$ (M_1D_4 and M_2D_2)
- **Aging** limits **throughput**: late rejuvenations keep the system slow

Load-Induced Rejuvenations



- Small M and D : frequent **premature** rejuvenations
- Same capacity $M \cdot D$ (boxed): **many shallow** buckets filter load best

Shallow vs. Deep Buckets

few deep

M1D4



many shallow

M4D1



same capacity $Z = MD = 4$

Spiky, $MD = 4$

load-induced rejuvenation probability

M1D4 0.349

M4D1 0.0626

QoS violation rate V

M1D4 0.75

M4D1 0.83

- Later buckets need **larger** violations: rejuvenation is deferred
- Fewer load-induced restarts, but more **QoS violations**

Guidelines

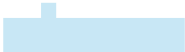
bursty arrivals
Spiky, DailyPattern



maximize $M \cdot D$

↑ throughput ↑ availability
↑ QoS violations

sustained arrivals
MostlyMid



intermediate $M \cdot D$

aging limits throughput:
avoid long aged periods

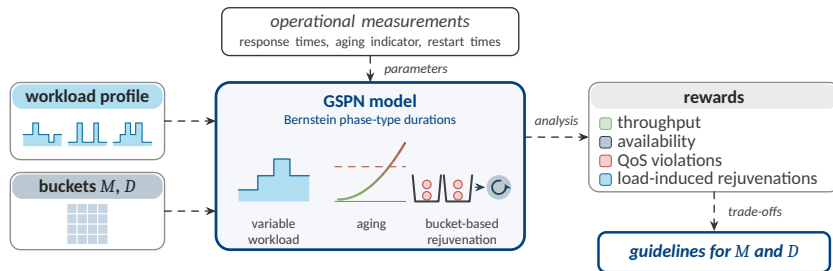
fixed capacity $M \cdot D$
how to split it



many shallow → **uptime**
few deep → **response times**

- The bucket configuration must follow the **workload profile**

Conclusion



- Best configuration is **workload-dependent**
- Larger buckets pay off only under **intermittent** arrivals
- Future work: experimentation on a real deployment

Replication package: github.com/oris-tool/WoSAR-2026-bucket-system-rep-pkg