Quantitative evaluation of software rejuvenation of a pool of service replicas

Leonardo Scommegna, Marco Becattini, Giovanni Fontani, Leonardo Paroli, Enrico Vicario

Dept. of Information Engineering, University of Florence

Software Technologies Lab - https://stlab.dinfo.unifi.it

WoSAR'24, Tsukuba, October 2024

This is about:

- Software aging and rejuvenation in cloud-based systems
- Models bridging current technology and practice
- Impact of SAR on service replica pools
- Two inspection-based rejuvenation strategies

{ST LAB} Software Technologies Laboratory

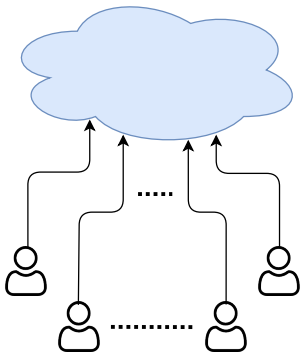
UNIVERSITÀ DEGLI STUDI FIRENZE

DINFO DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

{ST LAB} Software Technologies Laboratory

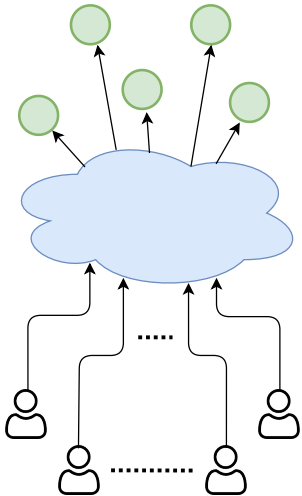
Contextualization

- **Cloud-based systems**: Web applications and online services
- **High workload**: significant number of simultaneous requests to manage
- **Highly availabile**: online service should be “always” reachable
- **Highly reliable**: service interruptions should be as rare as possible
- Single deployment strategies are **not suitable** for these types of systems



Replica-based Deployment Strategy

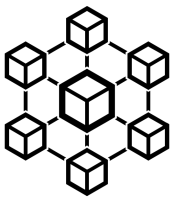
- Deployment of **multiple replicas** of the same service
- **Availability**: if one replica goes offline, others can still handle requests
- **Flexibility**: the number of replicas can be adjusted according to the system's workload

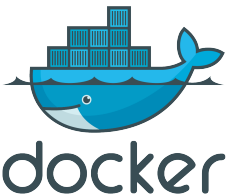


{ST LAB} Software Technologies Laboratory

Replica-based Strategies Pervasivity

- Well-suited for the **current technological context**
- **Microservices Architectures**: microservice-wise replication and a finer-grained adaptation
- **Virtualization/containerization**: easy and flexible deployment of replicas (e.g., Docker)
- **Orchestration Framework**: cloud-native support for replica-based strategies (e.g., Kubernetes)





{ST LAB} Software Technologies Laboratory

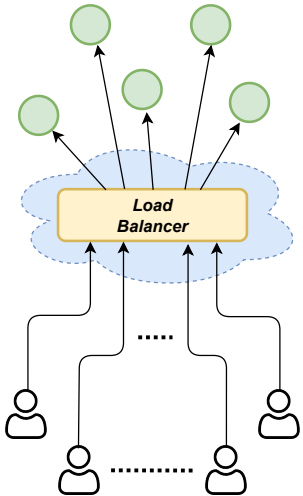
Replication-based Downsides

- **Replicas**: long-running systems are subject to **software aging**
- Software aging in **virtualized environments**:
  - conditioned by various additional factors
  - Aging harder to measure and predict
- Multiple parallel replicas means **multiple points of failure**
- Rejuvenation strategies should take into account the current technological context and state of practice

{ST LAB} Software Technologies Laboratory

System Model Considered

- **Pool of Service Replica**
  - Identical service replica
  - Each replica can be replaced by any other replica of the pool
- **Load Balancer**
  - Evenly distribute the workload among the replicas of the pool
  - Identifies which replicas are still active and which ones have failed
- **Software Aging Assumption**
  - Replicas of the same pool are identical pieces of software
  - Replicas of the same pool experience the same workload
  - Same software aging pattern for each replica



{ST LAB} Software Technologies Laboratory

Inspection-based Software Rejuvenation

- **Challenge**: Deciding when to initiate proactive maintenance
- **Inspection-Based Rejuvenation**:
  - Regular system replica inspections
  - Aim: determine the aging levels and the *remaining time to life*
- **Technological Support**:
  - Native monitoring and restart mechanisms
  - Simplifies implementation in real-world systems.
  - Example: Kubernetes Liveness Probe for automatic pod reboot[1]

[1] https://kubernetes.io/docs/concepts/configuration/liveness-readiness-startup-probes/

{ST LAB} Software Technologies Laboratory

Sensitivity and Specificity of Inspections

- **Aging Detection Approaches**:
  - Threshold-based
  - Statistical-based
  - Machine learning-based
- **Aging Indicators**:
  - System metrics used to assess aging levels
  - Help estimate time to unacceptable state and optimal recovery moment
- **Proxy Measures**:
  - Aging parameters are proxy measures of aging
  - Can sometimes be **misleading** and lead to **misclassification**
  - Consider **sensitivity** (true positive rate) and **specificity** (true negative rate)

{ST LAB} Software Technologies Laboratory

Software Rejuvenation Strategies Proposed

- Proposed Rejuvenation Strategies:
  - **Uncoordinated Rejuvenation**
  - **Coordinated Rejuvenation**
- Inspection-Based Paradigm independent of the aging detection approach (e.g., threshold-based)
- How aging and rejuvenation of each **single replica** affect the **entire pool**
- **Quantitative Evaluation**: strategies studied through the analysis of Stochastic Time Petri Net (STPN) with the **Sirio Library** of the **Oris Tool**.[2]

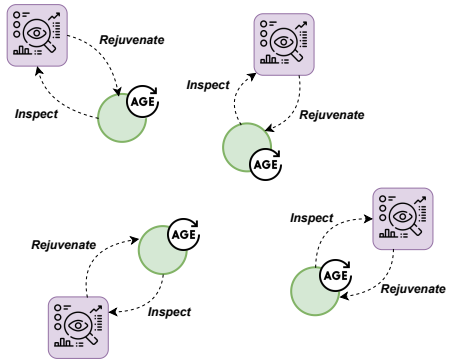
[2] *Paolieri, Biagi, Carnevali, Vicario, "The ORIS Tool: Quantitative Evaluation of Non-Markovian Systems" TSE 2021*

{ST LAB} Software Technologies Laboratory

Uncoordinated Rejuvenation

The Strategy

- **Independent rejuvenation** strategy for each replica
- Each replica is **periodically inspected**
- Each replica is rejuvenated **without considering the overall state** of the pool

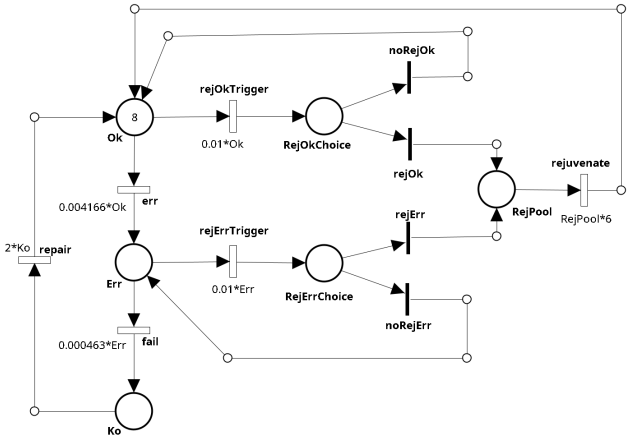


{ST LAB} Software Technologies Laboratory

Uncoordinated Rejuvenation Model

The Aging Process

- **Two-step** failure process as in *Garg et al.*[3]
- Available replicas as `OK+Err+Ko`
- Transition rates proportional to the precondition



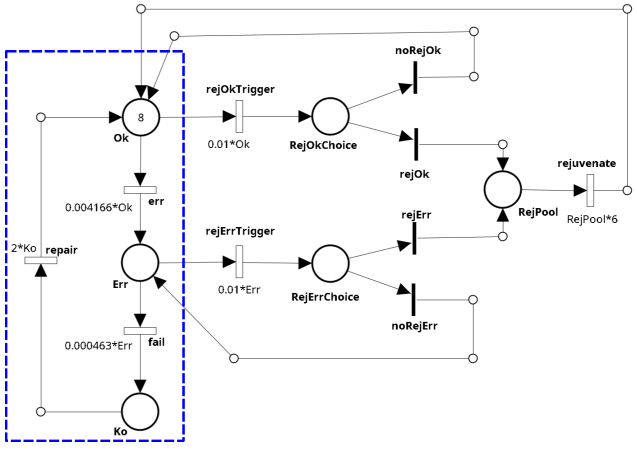
[3] *Garg, Puliafito, Telek, Trivedi "Analysis of software rejuvenation using Markov regenerative stochastic Petri net", ISSRE 1995*

{ST LAB} Software Technologies Laboratory

Uncoordinated Rejuvenation Model

The Aging Process

- **Two-step** failure process as in *Garg et al.*[4]
- Available replicas as `OK+Err+Ko`
- Transition rates proportional to the precondition



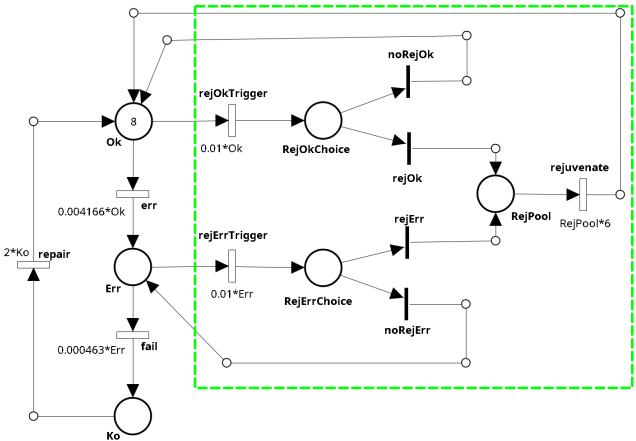
[4] *Garg, Puliafito, Telek, Trivedi "Analysis of software rejuvenation using Markov regenerative stochastic Petri net", ISSRE 1995*

{ST LAB} Software Technologies Laboratory

Uncoordinated Rejuvenation Model

The Rejuvenation Process

- **Periodic Inspections**: regardless the replica state (`Rej"*"Trigger` Transitions)
- **Fallibility of inspections**:
  - `RejOk`: false positives
  - `NoRejErr`: false negatives
  - 0.1 misclassification probability

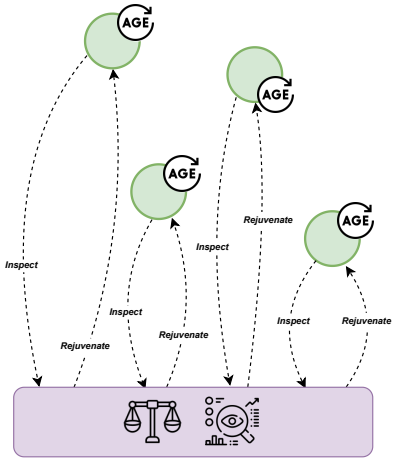


{ST} LAB Software Technologies Laboratory

Coordinated Rejuvenation

The Strategy

- **Centralized** Rejuvenation Strategy
- **Pool** periodically inspected
- Rejuvenation choice supported by replica state **comparison**
- **Prevent** simultaneous rejuvenations
- **Reduce misclassification**

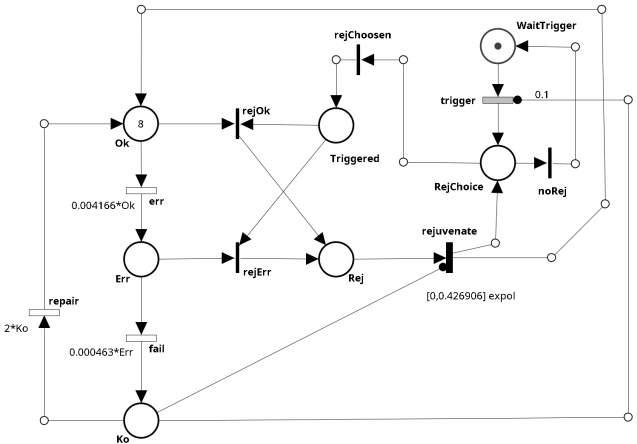


{ST LAB} Software Technologies Laboratory

Coordinated Rejuvenation Model

The Rejuvenation Process

- Pool-wise **periodic inspection**: `trigger` transition
- `RejChoice` switch depends on the **pool state**
- Multiple rejuvenations can occur **only sequentially**

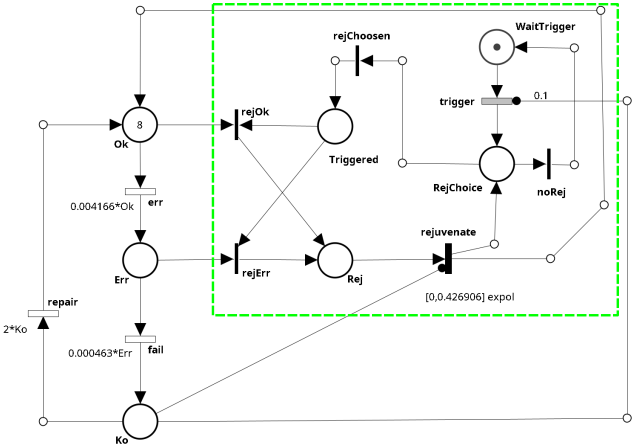


{ST LAB} Software Technologies Laboratory

Coordinated Rejuvenation Model

The Rejuvenation Process

- Pool-wise **periodic inspection**: `trigger` transition
- `RejChoice` switch depends on the **pool state**
- Multiple rejuvenations can occur **only sequentially**

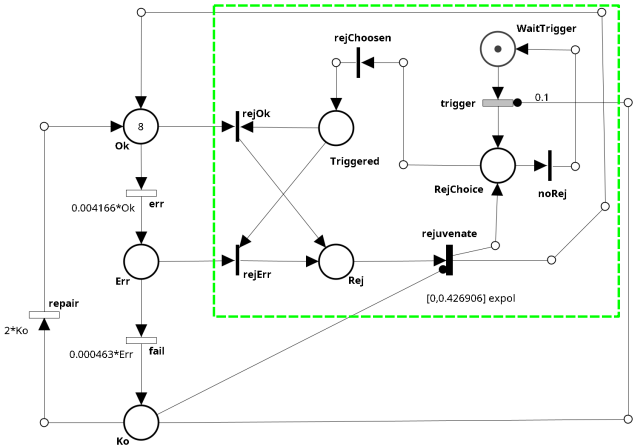


{ST LAB} Software Technologies Laboratory

Coordinated Rejuvenation Model

Comparison-based Rejuvenation Scheduling

- Selecting the next replica to rejuvenate by comparison with active replicas
- `Triggered` probabilistic switch
- p: misclassify an healthy over aged replica in a comparison
- k: aged replicas
- m: healty replicas
- $k * p^m$: false positive classification probability



{ST LAB} Software Technologies Laboratory

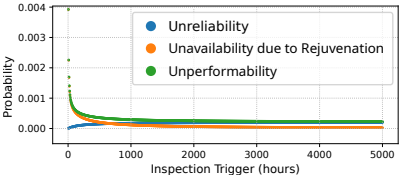
Analysis of the Models

- **Redundancy** keeps the service accessible even in case of unavailability of replicas
- **Performability problem**:
  - Unavailability of replicas causes workload redistribution
  - Performance drops and possible non-functional requirement violations
- **Metrics of Interest**:
  - **Unreliability**: unplanned outages
  - **Unperformability**: failure to meet QoS requirements when n or more replicas are offline
  - **Unavailability due to Rejuvenation**: planned outages
- **Parameters variations**:
  - **Pool size**: 1, 2, 4, and 8 replicas with n equal to 0, 1, 2, and 3 respectively
  - **Inspection period**: from 5 to 5000 hours incrementing by 5 hours

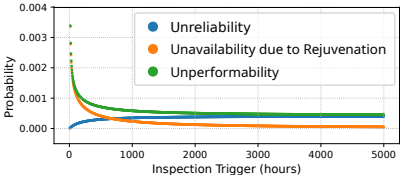
{ST LAB} Software Technologies Laboratory

Uncoordinated Rejuvenation Results

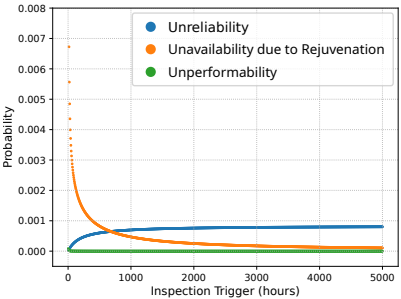
(a) Pool size=1 and n=0



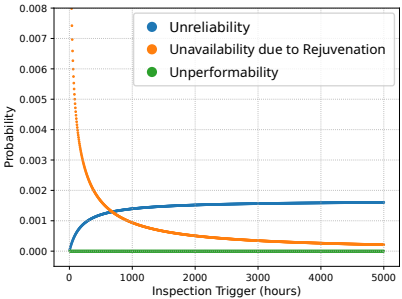
(b) Pool size=2 and n=1



(c) Pool size=4 and n=2



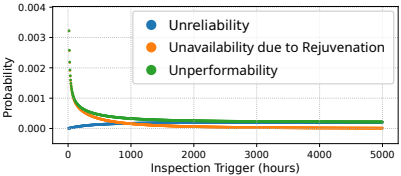
(d) Pool size=8 and n=3



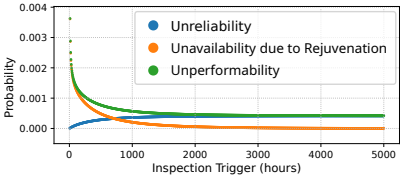
{ST LAB} Software Technologies Laboratory

Coordinated Rejuvenation Results

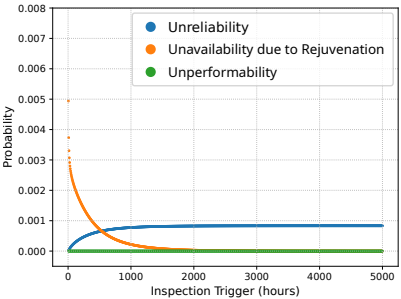
(a) Pool size=1 and n=0



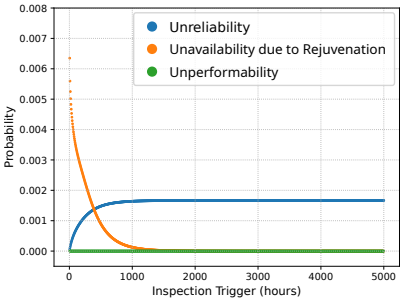
(b) Pool size=2 and n=1



(c) Pool size=4 and n=2



(d) Pool size=8 and n=3



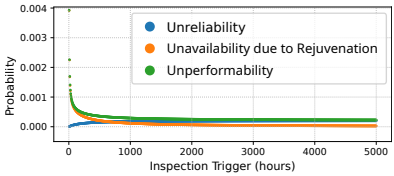
{ST LAB} Software Technologies Laboratory

Discussion

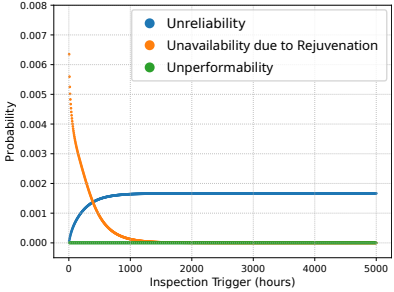
Common Results

- Lower inspection trigger:
  - **Lower unreliability**
  - **Higher unavailability due to rejuvenation**
- Larger pool size:
  - **Higher unreliability**
  - **Higher unavailability due to rejuvenation**
  - **Very Low Unperformability** from sizes above 2
- Lesson learned:
  - Inspection **frequency should increase** as the pool size increases
  - higher unavailability due to rejuvenation but preserved QoS

(a) Uncoordinated Pool size=1 and n=0



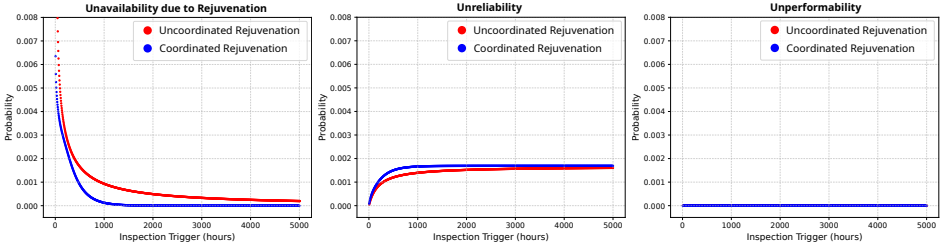
(b) Coordinated - Pool size=8 and n=3



{ST LAB} Software Technologies Laboratory

Discussion

Strategies Comparison

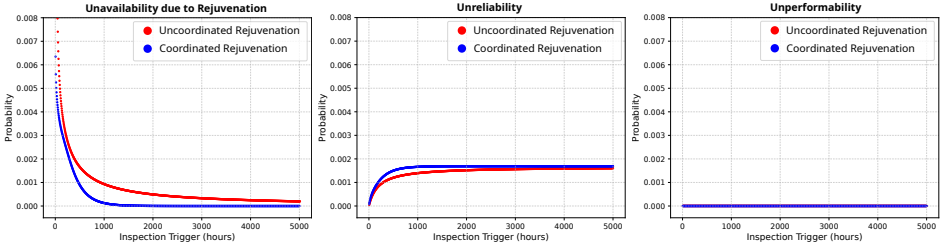


- Exemplary comparison for Pool Size $= 8$ and $n = 3$
- **Unavailability due to rejuvenation**: better with coordinated rejuvenation
- **Unreliability**: better for uncoordinated rejuvenation
- **Unperformability**: very similar behavior

{ST LAB} Software Technologies Laboratory

Discussion

Unreliability



- Coordinated rejuvenation: aged replicas are rejuvenated sequentially
- Some replicas may fail while waiting for their turn
  - **Model limitation**: just aged/non-aged classification
  - **Strategy variation**: allowing controlled simultaneous rejuvenations

{ST LAB} Software Technologies Laboratory

Conclusion and Future Directions

- Approach for modeling **SAR in replica-based systems**
- Modeling aligned with the **modern technological context**
- Two rejuvenation strategies: **coordinated** and **uncoordinated** rejuvenation
- Opens paths for **future extensions**:
  - Validation through **real system implementation**
  - Possibility of defining **additional rejuvenation strategies**
  - Granular aging representation to prioritize rejuvenation of **older replicas**

Replication package

Keep in contact