Use Two-Level Rejuvenation to Combat Software Aging and Maximize Average Resource Performance

Chunhui Guo, Hao Wu, Xiayu Hua, Douglas Lautner, Shangping Ren
Email: {cguo13, hwu28, xhua, dlautner}@hawk.iit.edu, ren@iit.edu
Illinois Institute of Technology

August 24, 2015
Introduction

Software aging is a well-known phenomenon and has two effects:

- failure rate increase
- performance degradation

Software rejuvenation is a preventive and proactive maintenance solution for handling system aging effects.

- Rejuvenation Models: four-state model [Huang et al., 1995], two-level model [Koutras and Platis, 2011]
- Handle Failure Rate Increase: maximize reliability [Guo et al., 2015] and availability [Koutras and Platis, 2011]
- Handle Performance Degradation: $P^2$-resource model with one-level rejuvenation [Hua et al., 2015]
Figure: Aging Effect of Matrix Multiplication Time on Cellphone
Outline

1 Introduction

2 Resource Model and Problem Formulation

3 Resource Supply Analysis

4 Average Resource Performance Maximization

5 Rejuvenation Strategy Impact Factors

6 Conclusion and Future Work
Due to aging, the resource performance decreases with time.

We assume $f(t) = 1 - a \times t$, where $a$ is a constant and $0 \leq a < 1$.

If $a = 0$, the resource’s performance does not degrade.
The resource can perform two-level rejuvenations

- **Cold Rejuvenation:** \( f(t_0 + \Phi_C) = 1 \)

- **Warm Rejuvenation:** \( f(t_1 + \Phi_W) = f_s \times p \), where \( f_s \) denotes the resource performance after previous rejuvenation and \( 0 < p < 1 \)

- **Time Cost:** \( \Phi_C > \Phi_W \)
Rejuvenation Threshold: $f(t) \leq r$, where $0 \leq r < 1$

Rejuvenation Pattern: $n (n \in \mathbb{N})$ warm rejuvenations followed by one cold rejuvenation

Periodic Rejuvenations: repeatedly rejuvenated by the pattern with period $\Pi$ (rejuvenation hyperperiod)

Max $n$: $N_{\text{max}} = \lfloor \log_p r \rfloor$

Figure: Resource Rejuvenation Pattern
Resource Model

\[ R = (f(t), r, p, \Phi_W, \Phi_C, n) \]

- \( f(t) \): resource performance function
- \( r \): resource performance threshold to rejuvenate
- \( p \): resource performance restore factor of a warm rejuvenation
- \( \Phi_W \): warm rejuvenation time cost
- \( \Phi_C \): cold rejuvenation time cost
- \( n \): number of warm rejuvenations before a cold rejuvenation
Average Resource Performance

\[ f_{\text{ave}} = \frac{S_L}{L} \]

where \( L \) is system longevity and \( S_L \) is total resource supply within \( L \).
## Problem Formulation

### Problem Definition
Given a resource $R(f(t), r, p, \Phi_W, \Phi_C, n)$, decide $n$ that maximizes the average resource performance, i.e., $f_{\text{ave}}$, within its operational interval $[0, L]$.

### Strategy
- First, we analyze the total resource supply $S_L$ with a given $n$.
- Second, we present the MAX-AVE-PERFORMANCE algorithm to determine the optimal $n$ with respect to maximizing average resource performance.
Outline

1. Introduction
2. Resource Model and Problem Formulation
3. Resource Supply Analysis
4. Average Resource Performance Maximization
5. Rejuvenation Strategy Impact Factors
6. Conclusion and Future Work
First, we analyze the resource supply $S_\Pi$ within a rejuvenation hyperperiod $\Pi$.

Second, we formalize the total resource supply $S_L$ within the system longevity $L$ on the basis of $S_\Pi$. 
Resource Supply within Rejuvenation Hyperperiod $\Pi$

\[ S_i = \int_{f^{-1}(p^{i-1})}^{f^{-1}(r)} f(t) dt = \int_{\frac{1-p^{i-1}}{a}}^{\frac{1-r}{a}} f(t) dt \]

\[ S_\Pi = \sum_{i=1}^{n+1} S_i \]

Figure: Resource Supply Analysis
Resource Supply within System Longevity $L$

- Case 1 ($L \ mod \ \Pi = 0$): $S_L = S_\Pi \cdot \frac{L}{\Pi}$
- Case 2 ($L \ mod \ \Pi \neq 0$): $S_L = S_\Pi \cdot \left\lfloor \frac{L}{\Pi} \right\rfloor + S_R$

  - Case 2.1 ($I_R$ ends during a rejuvenation): $S_R = \sum_{i=1}^{j} S_i$
  - Case 2.2 ($I_R$ ends when the resource is available): $S_R = \sum_{i=0}^{j-1} S_i + \int_{f^{-1}(p^{i-1})}^{f^{-1}(p^{j-1})+l_R-\sum_{i=0}^{j-1} l_i-(j-1)\Phi_W} f(t)dt$
As \( n \in \mathbb{N} \) and \( 0 \leq n \leq N_{\text{max}} \), the possible choices of \( n \) are limited. We present a linear search method to determine \( N^* \) maximizing \( f_{\text{ave}} \).

**Algorithm 1** MAX-AVE-PERFORMANCE

1: \( N^* = 0 \)
2: \( f_{\text{max}} = 0 \)
3: \( N_{\text{max}} = \lfloor \log_p r \rfloor \)
4: for \( n = 0 \) to \( N_{\text{max}} \) do
5: Calculate \( S_L \)
6: \( f_{\text{ave}} = S_L/L \)
7: if \( f_{\text{ave}} > f_{\text{max}} \) then
8: \( N^* = n \)
9: \( f_{\text{max}} = f_{\text{ave}} \)
10: end if
11: end for
12: return \( N^* \) and \( f_{\text{max}} \)
1. Introduction

2. Resource Model and Problem Formulation

3. Resource Supply Analysis

4. Average Resource Performance Maximization

5. Rejuvenation Strategy Impact Factors

6. Conclusion and Future Work
We evaluate the rejuvenation strategy impact factors from two aspects:

- the relationship between warm rejuvenation number $n$ and average resource performance $f_{\text{ave}}$;
- the impacts of warm/cold rejuvenation time cost on the optimal warm rejuvenation number $N^*$ that maximizes the average resource performance $f_{\text{ave}}$. 
The resource model with two-level rejuvenations achieves 25.22% higher average resource performance than the resource model with one-level rejuvenations \((n = 0)\).
(a) Optimal Number of Warm Rejuvenation

(b) Maximal Average Resource Performance

Figure: Warm/Cold Rejuvenation Time Cost Impact
Conclusion

- Propose the resource model using a two-level rejuvenation strategy to combat resource performance degradation due to software aging.

- Formally analyze the resource supply function of the proposed resource model.

- Present the MAX-AVE-PERFORMANCE algorithm to maximize the average resource performance.

- Validate the resource supply analysis through simulations.

- Compared with the resource model with one-level rejuvenations, the proposed resource model with two-level rejuvenations achieves 25.22% higher average resource performance.
Future Work

- Analyze task schedulability of the resource model with two-level rejuvenations, and study the optimal rejuvenation pattern maximizing the task schedulability.

- Obtain resource performance degradation function from experiments.


Questions?

Thank You