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

Chunhui Guo, Hao Wu, Xiayu Hua, Douglas Lautner†, Shangping Ren∗
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616, USA
{cguo13, hwu28, xhua, dlautner}@hawk.iit.edu, ren@iit.edu

Abstract—Software aging is a common phenomenon which is often manifested through system performance degradation. Rejuvenation is one of the most commonly used approaches to handle issues caused by software aging. To combat resource performance degradation and at the same time maintain maximized average resource performance, we present a two-level rejuvenation strategy, i.e., interleaving a set of $n$ warm rejuvenations with one cold rejuvenation. Our target is to find the optimal $n$ that maximizes system average performance. We first define a resource model that takes into consideration of performance degradation and two-level rejuvenations. Based on the resource model, we formally analyze the resource supply and present the MAX-PERFORMANCE algorithm to determine the optimal rejuvenation pattern that maximizes the average resource performance. The simulation results show that with a two-level rejuvenation strategy, we can achieve 25.22% higher average resource performance compared with a single level rejuvenation strategy.

Index Terms—Software Aging, Performance Degradation, Resource Model, Two-Level Rejuvenation, Resource Supply Analysis, Resource Performance Maximization

I. INTRODUCTION

Software aging is a well-known phenomenon in computer systems that slows down the system performance and eventually leads to transient failure [1]. Software aging is usually caused by memory leaks and error accumulation. As modern computer systems are getting more complex and supporting more concurrent applications, software aging becomes more obvious and significantly impacts system performance.

Software aging also has an impact on today’s mobile device performance. To provide evidences for such slowdown phenomena on cellphones, we write an Android APP which computes the multiplication of two $500 \times 500$ matrices and records the computation time. The APP runs on a cellphone with a Qualcomm 1.5GHz dual-core, 1G RAM, and 2G internal storage. The APP is the only application running on the cellphone under Android 2.3.6. Fig. 1 shows the measurements of the computation times of matrix multiplication over about 5 days. Each point represents the average computation time of 300 matrix multiplication computations. From Fig. 1, we have following observations:

1) The computation time within the interval [1, 10], [15, 20], [24, 39], and [42, 53] has an increasing trend, which indicates that the cellphone suffers from aging effects.

2) The computation time within the [10, 15], [20, 24] and [39, 42] intervals have a decreasing trend. The log file indicates that the cellphone was rebooted at point 10; and the matrix multiplication application was restarted at point 20 and 39.

Fig. 1. Aging Effect of Matrix Multiplication Time on Cellphone

The second observation also indicates that both cellphone reboot (cold rejuvenation) and application restart (warm rejuvenation) can restore a cellphone’s performance, but the restore capability of cold rejuvenation is higher than the warm rejuvenation. In addition, the resource performance after the second warm rejuvenation is lower than the first warm rejuvenation.

Currently, smartphone subsystems have reset mechanisms, called “silent resets”, incorporated to restore functionality while minimizing user impact. However, these resets happen in a reactive manner and are not predicted or scheduled. For instance, WiFi has a reset mechanism a.k.a. sub system restart.

†Douglas Lautner is a part-time Ph.D student at CS Dept. IIT. He is also the Director of Software Engineering at Motorola Mobility.

∗The research is supported in part by NSF under grant number CAREER 0746643, CNS 1018731 and CNS 1035894.
It’s a structure for the WiFi Firmware to restart its execution point. The reset is initiated by either a program fault (e.g. out-of-bound memory access, bad instruction jump, or memory corruption) or a health-monitoring trigger (e.g. inability to transmit packets for a period of time (hardware lockup), packet memory overflow, or register value lockup).

For systems that support long lasting applications, software aging is an unavoidable phenomenon which may lead the applications running on the system violate their QoS requirements. Hence, rejuvenation is a necessary process to maintain the system performance at an expected level. However, rejuvenation takes time during which the system is not available to user applications. Different levels of rejuvenation have different overhead, different performance restore capability, and result in different system performance. In this paper, we are to maximize the system’s average performance by using the combination of warm and cold software rejuvenation. In particular, we extend our previous $P^2$-resource model [2] with the consideration of both warm and cold software rejuvenation. Based on the extended resource model, we formally analyze resource supply and give the optimal combination of using warm and cold software rejuvenation that maximizes the system’s average performance.

The rest of the paper is organized as follows. First, we discuss related work in Section II. System models and assumptions the paper is based upon are presented in Section III. A formal definition of the problem the paper is to address is also presented in Section III. Section IV analyzes the resource supply within the system longevity with a given rejuvenation pattern. In Section V, we present the MAX-PERFORMANCE algorithm to determine the optimal rejuvenation pattern with the goal of maximizing the average resource performance. We verify the theoretical analysis and evaluate the proposed resource model by simulations in Section VI. Section VII concludes the paper and points out our future work.

II. RELATED WORK

Software rejuvenation is a preventive and proactive maintenance solution for handling system aging effects. Huang et al. [3] first proposed the concept of software rejuvenation and developed a four-state (i.e., Robust State, Failure Probable State, Failure State, and Rejuvenation State) system model to perform rejuvenation. Since then, many rejuvenation models have been developed by the research community [3], [4]. For instance, Koutras et al. extended the initial rejuvenation model by considering two levels of rejuvenation actions [5], i.e., perfect rejuvenation action and minimal rejuvenation action. The perfect rejuvenation (cold rejuvenation) results in the system returning to the Robust State (initial state), while the minimal rejuvenation (warm rejuvenation) results in the system returning to the Failure Probable State (the state before rejuvenation). Alonso et al. experimentally compared the overhead by taking different software rejuvenation technologies [6]. They categorize the software rejuvenation into three different granularities, i.e. application level, operating system (OS) level and hardware level. The application level rejuvenation takes the least time but also has the least impact on the system performance. The hardware level rejuvenation takes the longest time but lead to the best system performance. The OS level rejuvenation is in the middle for both time cost and performance impact.

To analyze software aging and study aging related failures, Trivedi et al. [7] presented two approaches: analytical modeling approach for determining optimal times to rejuvenate and measurement based approach for detection and validation. Tai et al. [8] identified key factors that may impact system reliability and developed an approach to maximizing system reliability by analyzing the optimal interval between maintenance. Guo et al. considered both transient faults caused by software aging effects and network transmission faults and analyzed the optimal software rejuvenation period that maximizes systems reliability [9]. Okamura et al. [10] discussed a maintenance policy that combines aperiodic rejuvenations and periodic checkpoints to maximize the system availability. The estimations of reliability and availability were analyzed in [11], [5].

The two-level rejuvenation model has also been analyzed by the research community. Hong et al. studied two-level closed-loop rejuvenation techniques and proposed an approach to minimize the average rejuvenation cost [12]. Koutras et al. observed the effects of a two-level software rejuvenation model on availability, downtime and rejuvenation cost indicators [13]. The two-level rejuvenation model was also modeled by a Semi-Markov process and analyzed to find the optimal rejuvenation policy to maximize the system availability [14], [15], [16].

As pointed out in [17], a general characteristic of software aging is the gradual performance degradation and/or an increase in the software failure rate. The above works mainly focus on aging related failure effects on QoS and how to perform rejuvenations to optimize the system QoS, such as availability and reliability. However, not much work has been done on aging caused performance degradation analysis and how to perform rejuvenations to improve the resource performance.

Recently, Hua et al. [2] proposed a new resource model ($P^2$-resource) which takes software aging and periodical resource rejuvenations into consideration. It gives formally schedulability analysis under the $P^2$-resource model for both EDF (earliest deadline first) and RM (rate monotonic) scheduling algorithms.

In this paper, we are to extend the $P^2$-resource with the consideration of both warm and cold software rejuvenations along with their impacts on the system performance. Based on the extended resource model, we give the formal analysis of resource supply and present a linear search algorithm to determine the optimal interleaving between warm and cold rejuvenations that maximizes the average resource performance.

III. SYSTEM MODELS AND PROBLEM FORMULATION

A. Models and Assumptions

Resource Performance Function
We use function $f(t)$ to denote the resource performance at time $t$. The resource performance represents the computation cycles per unit time provided by the resource to applications. As the system deteriorates with aging, we assume that the resource performance function $f(t)$ decreases with time $t$ and $f(0) = 1$ [1], [2]. As for any decreasing resource performance function, the strategy to analyze the resource’s performance is the same. Hence, to simplify the discussion of our approach, we further assume that the resource performance function is a linear decreasing function, i.e.,

$$f(t) = 1 - at$$

where $a$ denotes the resource performance decreasing rate which is assumed to be a constant and $0 \leq a < 1$. If $a = 0$, the resource’s performance does not degrade.

**Resource Rejuvenation Pattern**

Similar to [15], [11], [6], the system can perform two levels of rejuvenations, i.e., cold rejuvenation and warm rejuvenation. Once the resource’s performance $f(t)$ degrades to a threshold $r$ ($0 \leq r < 1$), we take a warm or cold rejuvenation to restore its performance. After a cold rejuvenation, the system returns to the Robust State, and the resource performance is restored to $f(t) = 1$. When a warm rejuvenation is completed, the system goes back to the Failure Probable State, and the resource performance is only restored to $f(t) = f_s \cdot p$ where $f_s$ denotes the resource start performance ($f(t)$ after last rejuvenation) and $p$ is the resource performance restore factor ($0 < p < 1$). The resource is unavailable when it goes through the rejuvenation process. The downtime caused by each cold rejuvenation or warm rejuvenation is assumed to be a constant $\Phi_C$ and $\Phi_W$, respectively. We further assume $\Phi_C > \Phi_W$.

As the resource performance after each warm rejuvenation is smaller than the previous warm rejuvenation, if we only take warm rejuvenations, the resource performance will eventually be below the threshold $r$ and hence a cold rejuvenation becomes necessary. We define the rejuvenation pattern as $n$ ($n \in \mathbb{N}$) warm rejuvenations followed by one cold rejuvenation, as shown in Fig. 2. The time interval of an entire rejuvenation pattern is denoted as rejuvenation hyperperiod $\Pi$. We assume that the resource is repeatedly rejuvenated by the above pattern with period $\Pi$.

![Fig. 2. Resource Rejuvenation Pattern](image)

As the initial resource performance is $f(0) = 1$, the resource performance after $n$ warm rejuvenations is $f(t) = p^n$. The resource performance after the $n$th warm rejuvenation must not be smaller than the threshold, i.e., $p^n \geq r$, otherwise the $n$th rejuvenation should be a cold rejuvenation. Hence, we have $n \leq \log_p r$ and the maximal warm rejuvenation number before a cold rejuvenation in the rejuvenation pattern is $N_{\text{max}} = \lfloor \log_p r \rfloor$.

**Resource Model**

The resource model is characterized by a 6-tuple $(f(t), r, p, \Phi_W, \Phi_C, n)$, where $f(t)$ is the resource performance function, $r$ is the resource performance threshold to start a cold rejuvenation, $p$ is the resource performance restore factor of a warm rejuvenation, $\Phi_W$ is the warm rejuvenation time cost, $\Phi_C$ is the cold rejuvenation time cost, and $n$ is the number of warm rejuvenations before a cold rejuvenation in the rejuvenation pattern. We assume the resource starts at time zero.

If the resource only takes cold rejuvenations, i.e., $n = 0$, the resource model degenerates to the $P^2$-resource model in [2].

**Average Resource Performance**

We define the average resource performance within a system’s longevity $L$ as the ratio between the total resource supply $S_L$ within $L$, i.e.,

$$f_L = \frac{S_L}{L} \tag{1}$$

**B. Problem Formulation**

The problem we are to address is defined below:

**Problem:** Given a resource $R(f(t), r, p, \Phi_W, \Phi_C, n)$, decide $n$ that maximizes the average resource performance, i.e., $f_L$, within its operational interval $[0, L]$.

According to Eq. (1), maximizing the average resource performance $f_L$ is to maximize the total resource supply $S_L$ with given system longevity $L$. We take two steps to address the problem. First, we analyze the total resource supply $S_L$ with a given rejuvenation pattern (Section IV). Second, we present the MAX-PERFORMANCE algorithm (Section V) to determine the optimal rejuvenation pattern, i.e., $n$ with respect to maximizing average resource performance.

**IV. RESOURCE SUPPLY ANALYSIS**

In this section, we first analyze the resource supply $S_\Pi$ within a rejuvenation hyperperiod, and then formalize the total resource supply $S_L$ within the system longevity on the basis of $S_\Pi$.

**A. Resource Supply within Rejuvenation Hyperperiod $\Pi$**

Suppose the rejuvenation pattern is given as follows: $n$ warm rejuvenations followed by one cold rejuvenation, as shown in Fig. 2.

Before the $i$th ($1 \leq i \leq n + 1$) rejuvenation, the start performance and the end performance of the resource are $p^{i-1}$ and $r$, respectively. The resource available time length of the $i$th rejuvenation is

$$l_i = f^{-1}(r) - f^{-1}(p^{i-1}) = \frac{p^{i-1} - r}{a} \tag{2}$$
The resource supply of the $i$th rejuvenation is
\[
S_i = \int_{f^{-1}(p_i-1)}^{f^{-1}(p_i)} f(t)dt = \int_{\frac{l_i+n\cdot \Phi_W+\Phi_C}{\Pi}}^{\frac{l_i+n\cdot \Phi_W+\Phi_C+1}{\Pi}} f(t)dt.
\] (3)

To generalize Eq. (2) and Eq. (3), we assume $l_0 = 0$ and $S_0 = 0$.

A rejuvenation pattern contains $n+1$ resource available intervals, $n$ warm rejuvenations, and one cold rejuvenation. The rejuvenation hyperperiod is
\[
\Pi = \sum_{i=1}^{n+1} l_i + n \cdot \Phi_W + \Phi_C.
\] (4)

The resource supply within the rejuvenation hyperperiod $\Pi$ is the summation of $n+1$ resource available intervals, i.e.,
\[
S_\Pi = \sum_{i=1}^{n+1} S_i.
\] (5)

B. Resource Supply within System Longevity $L$

In practical cases, the system longevity is much larger than the rejuvenation hyperperiod, i.e., $L \gg \Pi$ [3]. We divide the analysis of the resource supply $S_L$ within the longevity into two parts based on if the longevity $L$ is divisible by the rejuvenation hyperperiod $\Pi$ or not.

Case 1: $L \mod \Pi = 0$

In this case, the system longevity contains $L/\Pi$ entire rejuvenation hyperperiods. The total resource supply within the longevity is the sum of $L/\Pi$ resource supplies $S_{\Pi}$ in a rejuvenation hyperperiod, i.e.,
\[
S_L = S_\Pi \cdot \frac{L}{\Pi}.
\] (6)

Case 2: $L \mod \Pi \neq 0$

In this case, we divide the total resource supply into two parts: the resource supply in the interval containing $\lfloor L/\Pi \rfloor$ entire rejuvenation hyperperiods and the resource supply of the remaining time interval $I_R$. Hence, the total resource supply within the longevity is
\[
S_L = S_\Pi \cdot \left\lfloor \frac{L}{\Pi} \right\rfloor + S_R.
\] (7)

where $S_R$ is the resource supply of the remaining time interval $I_R$ with length $l_R = L \mod \Pi$.

We further divide the analysis of the remaining resource supply $S_R$ into two cases based on if the remaining time interval $I_R$ ends in the time period when a rejuvenation is in process.

Case 2.1: $I_R$ ends during a rejuvenation

As shown in Fig. 3, the remaining interval $I_R$ ends during the $j$th rejuvenation implies
\[
\begin{align*}
&\sum_{i=1}^{j} l_i + (j-1)\Phi_W \leq l_R \leq \sum_{i=1}^{j} l_i + j\Phi_W \quad \text{if} \ 1 \leq j \leq n \\
&\Pi - \Phi_C \leq l_R \leq \Pi \quad \text{if} \ j = n + 1
\end{align*}
\] (8)

where $j \in \mathbb{N}$.

The resource supply $S_R$ within $I_R$ is hence
\[
S_R = \sum_{i=1}^{j} S_i
\] (9)

where the value of $j$ can be calculated from $l_R$ and Eq. (8).

Case 2.2: $I_R$ ends when the resource is available

Similar to Case 2.1, $I_R$ may end at the $j$th resource available interval, which implies
\[
\sum_{i=0}^{j-1} l_i + (j-1)\Phi_W \leq l_R \leq \sum_{i=0}^{j} l_i + (j-1)\Phi_W
\] (10)

where $1 \leq j \leq n + 1$ and $j \in \mathbb{N}$.

Hence, as shown in Fig. 3, the resource supply within $I_R$ is
\[
S_R = \sum_{i=0}^{j-1} S_i + \int_{f^{-1}(p_{j-1})}^{f^{-1}(p_j)} f(t)dt
\] (11)

where the value of $j$ can be calculated from $l_R$ and Eq. (10).

V. AVERAGE RESOURCE PERFORMANCE MAXIMIZATION

As $n \in \mathbb{N}$ and $0 \leq n \leq N_{\text{max}}$, the possible choices of the number of warm rejuvenations $n$ that maximizes the average resource performance $f_L$ are limited. We present a linear search method, i.e., the MAX-PERFORMANCE algorithm, to determine the optimal number of warm rejuvenations $N^*$ before a cold rejuvenation. The MAX-PERFORMANCE algorithm is given as Algorithm 1.

In particular, the MAX-PERFORMANCE algorithm initializes both the optimal number of warm rejuvenations $N^*$ and the maximal average resource performance $f_{\text{max}}$ as 0 (line 1-2), and calculates the possible maximal number of warm rejuvenations $N_{\text{max}}$ (line 3). In the for loop (line 4-11), for each possible number of warm rejuvenations $n$, we calculate the total resource supply $S_L$ (line 5) according the analysis in Section IV and the average resource performance $f_L$ (line 6). Then we determine if the current $n$ maximizes the average resource performance (line 7-10). The algorithm finally returns the maximal average resource performance $f_{\text{max}}$ and the corresponding number of warm rejuvenations $N^*$ (line 12).

Based on the resource supply analysis in Section IV, the resource supply calculation in Algorithm 1 (line 5) costs $O(n)$ time. Hence, the time complexity of Algorithm 1 is $O(n^2)$. 

![Fig. 3. Resource Supply Analysis](image)
we calculate the average resource performance and the maximal average resource performance \( f_{\text{max}} \) during the system longevity.

A. Relationship between \( n \) and \( f_L \)

To evaluate the relationship between the number of warm rejuvenations \( n \) and average resource performance \( f_L \), we conduct a simulation with the following parameters:

- Resource performance degradation rate: \( a = 0.005 \)
- Resource performance threshold: \( r = 0.3 \)
- Resource performance restore factor of a warm rejuvenation: \( \rho = 0.95 \)
- Cold rejuvenation time cost: \( \Phi_C = 150 \)
- Warm rejuvenation time cost: \( \Phi_W = 45 \)
- System longevity: \( L \in \{1 \times 10^4, 3 \times 10^4, 5 \times 10^4, 1 \times 10^5\} \)

The possible maximal number of warm rejuvenations is \( N_{\text{max}} = \lfloor \log_p r \rfloor = 23 \). With a given system longevity \( L \), for each possible warm rejuvenation number, i.e., \( n \in [0, N_{\text{max}}] \), we calculate the average resource performance \( f_L \) according to the analysis in Section IV and Eq. (1). Fig. 4 shows the average resource performance under different numbers of warm rejuvenations for each system longevity. From Fig. 4, we have the following observations:

1) When the number of warm rejuvenations \( n \) increases, the average resource performance \( f_L \) first increases and then decreases. For instance, \( f_L \) increases when \( n \) increases from 0 to 3 and starts to decrease when \( n \) increases from 3 to 23.
2) When the number of warm rejuvenations \( n \) is too small or too large, the average resource performance \( f_L \) is relatively low. For instance, when \( n = N_{\text{max}} = 23 \), \( f_L \) reaches its minimal value.
3) The system longevity \( L \) does not have a significant impact on the rejuvenation behavior when \( L >> \Pi \).

In our models given in Section III-A, we assume a rejuvenation pattern starts with the initial state, i.e., \( f(t) = 0 \), which indicates the rejuvenation behaviors in each rejuvenation hyperperiod are the same. In addition, we have also made the assumption that the system periodically repeats the rejuvenation pattern with period \( \Pi \). If \( L >> \Pi \), the system longevity does not have a significant impact on rejuvenation effects. This observation is evidenced from the following aspects:

1) For different system longevity, the optimal number of warm rejuvenations that maximize the average resource performance are the same. In particular, for the tested four longevity cases, \( N^* = 3 \).
2) With the same number of warm rejuvenations \( n \), the average resource performance \( f_L \) of the four longevity cases is similar. For instance, the maximal difference of \( f_L \) for four longevity cases is 3.76%.
3) The average resource performance trend changing over the number of warm rejuvenations are similar.

The observations are consistent with our analysis, i.e., there is an optimal number of warm rejuvenations between 0 and \( N_{\text{max}} \) that maximizes the average resource performance. When \( n = 0 \), i.e., the system only takes cold rejuvenations, the resource becomes a \( P^2 \)-resource [2]. The simulation results also show that the extended resource model achieves 25.22% higher average resource performance than the \( P^2 \)-resource model.

![Fig. 4. Average Resource Performance vs Warm Rejuvenation Number](image)

B. Warm/Cold Rejuvenation Time Cost Impact

We conduct a simulation to evaluate the impact of warm/cold rejuvenation time cost on the optimal number of warm rejuvenations \( N^* \) that maximizes \( f_L \) and average resource performance \( f_{\text{max}} \). The simulation parameters are set the same as in Section VI-A except the following two parameters:

- System longevity: \( L \in \{1 \times 10^4, 3 \times 10^4, 5 \times 10^4, 1 \times 10^5\} \)
- Warm/cold rejuvenation time cost: \( \Phi_C = 150 \)

Algorithm 1 MAX-PERFORMANCE

**Input:** A resource \( R(f(t), r, \Phi_W, \Phi_C, n) \) and the system longevity \( L \).

**Output:** The optimal warm rejuvenation number \( N^* \) before a cold rejuvenation that maximizes the average resource performance, and the maximal average resource performance \( f_{\text{max}} \) during the system longevity.

```
1: \( N^* = 0 \)
2: \( f_{\text{max}} = 0 \)
3: \( N_{\text{max}} = \lceil \log_p r \rceil \)
4: for \( n = 0 \) to \( N_{\text{max}} \) do
5: \( f_L = S_L^* \) according to Eq. (6) or Eq. (7)
6: if \( f_L > f_{\text{max}} \) then
7: \( N^* = n \)
8: \( f_{\text{max}} = f_L \)
9: end if
10: end for
11: return \( N^* \) and \( f_{\text{max}} \)
```
Cold rejuvenation time cost: $\Phi_C \in \{100, 150, 200, 300\}$
Warm rejuvenation time cost: $\Phi_W \in [0, 100]$ with step 5
System longevity: $L = 10 \times 10^4$

With a given cold rejuvenation time cost $\Phi_C$, for each warm rejuvenation time cost $\Phi_W$ choice, we use the MAX-PERFORMANCE algorithm (Algorithm 1) to determine the optimal number of warm rejuvenations $N^*$ that maximizes $f_L$ and average resource performance $f_{\text{max}}$. Fig. 5(a) and Fig. 5(b) depict the warm/cold rejuvenation time cost impact on $N^*$ and $f_{\text{max}}$, respectively. From Fig. 5, we have the following observations:

1) In general, the optimal number of warm rejuvenations $N^*$ decreases when the warm rejuvenation time cost $\Phi_W$ increases; it increases when the cold rejuvenation time cost $\Phi_C$ increases.
2) The maximal average resource performance $f_{\text{max}}$ decreases when both warm and cold rejuvenation time costs increase.
3) Both the optimal number of warm rejuvenation $N^*$ and the maximal average resource performance $f_{\text{max}}$ decrease with warm/cold rejuvenation time cost ratio $\Phi_W/\Phi_C$ increasing.

The observations are consistent with the intuition behind the proposed resource model. If the warm/cold rejuvenation costs less/more time, i.e., the ratio $\Phi_W/\Phi_C$ is smaller, we should perform more warm rejuvenations to take its low time cost advantage. As the resource is unavailable during rejuvenations, the average resource performance decreases if the rejuvenation’s time cost increases. When the ratio $\Phi_W/\Phi_C$ is larger, the proposed resource model can benefit more from the low time cost advantage of warm rejuvenations, i.e., results in higher average resource performance $f_{\text{max}}$.

To combat resource performance degradation due to software aging, we have extended our previous $P^2$-resource model by using a two-level rejuvenation strategy to maintain resource performance. Based on the extended resource model, we have formally analyzed the resource supply function and presented the MAX-PERFORMANCE algorithm to determine the optimal rejuvenation pattern that maximizes the average resource performance. We have also verified the theoretical analysis and compared the extended resource model through simulations. Compared with the $P^2$-resource model, the extended resource model achieves 25.22% higher average resource performance.

The current rejuvenation pattern is $n$ warm rejuvenations followed by one cold rejuvenation. The paper does not consider tasks deployed on the resource. Our future work is to analyze task schedulability and study the optimal rejuvenation pattern for a given task set with the goal of maximizing the task set schedulability.

REFERENCES


