Technical Report: CS-TR-2006-001

Reliability-Aware Dynamic Energy Management in Dependable Embedded Real-Time Systems*

Dakai Zhu
Department of Computer Science
University of Texas at San Antonio
San Antonio, TX, 78249
dzhu@cs.utsa.edu

Abstract

Recent studies show that, voltage scaling, which is an efficient energy management technique, has a direct and negative effect on system reliability because of the increased rate of transient faults (e.g., those induced by cosmic particles). In this work, we propose energy management schemes that explicitly take system reliability into consideration. The proposed *reliability-aware* energy management schemes *dynamically* schedule recoveries for tasks to be scaled down to recuperate the reliability loss due to energy management. Based on the amount of available slack, the application size and the fault rate changes, we analyze when it is profitable to reclaim the slack for energy savings without sacrificing system reliability. Checkpoint technique is further explored to efficiently use the slack. Analytical and simulation results show that, the proposed schemes can achieve comparable energy savings as ordinary energy management schemes while preserving system reliability. The ordinary energy management schemes that ignore the effects of voltage scaling on fault rate changes could lead to drastically decreased system reliability.

Index Terms: Energy management; Reliability; Real-time systems

*A preliminary version of the paper will appear in RTAS 2006. Available at http://www.cs.utsa.edu/~dzhu/papers/RTAS06-zhu.pdf

1 Introduction

The performance of modern computing systems has increased at the expense of dramatically increased power consumption. The increased power consumption reduces the operation time for battery-operated embedded systems (e.g., PDAs and cell phones) as well as increases the operation cost for high performance parallel systems (e.g., data centers and server farms), where the excessive amount of heat generated requires high cooling capacity. Many hardware and software techniques have been proposed to manage power consumption in modern computing systems and power aware computing has become an important research area recently. As an efficient energy management technique, *voltage scaling*, which reduces system supply voltage for lower operation frequencies [35, 37], has been used extensively in the recently proposed power management schemes [2, 22, 24, 28].

Another traditionally important avenue in real-time systems research is fault tolerance. For safety-critical real-time systems, where the consequence of a failure can be catastrophic, faults must be detected, and appropriate recovery operations must be completed before the deadline. It has been reported that *transient* faults occur much more frequently than *permanent* faults [5, 15, 16]. Moreover, with continuing scaling of CMOS technologies and adjustment of design margins for higher performance, it is expected that, in addition to the systems that traditionally operate in electronics-hostile environments (such as those in outer space), practically all digital systems will be much more vulnerable to the transient faults [9, 34]. In this work, we will focus on transient faults and explore the *backward error recovery* techniques, which restore the system state to a previous safe state and repeat the computation [25], to tolerate them.

However, both voltage scaling and backward recovery techniques rely on the active use of system slack. When more slack time is dedicated as temporal redundancy for backward recovery to increase system reliability, less slack is available for energy management to save energy. Therefore, there is an interesting trade-off between system reliability and energy consumption [41]. Moreover, it has been shown that voltage scaling has a direct effect on the rate increases of transient faults, especially for those induced by cosmic ray radiations, which further complicates the interplay between system reliability and energy efficiency.

Due to the effects of cosmic ray radiations, soft errors (i.e., transient faults) can be caused by the atmospheric nuclear/high energy particles (alpha-particles, protons and neutrons) when they strike the sensitive region in a semiconductor device. In general, the error rate is exponentially related to the *critical charge* (which is the smallest charge required to cause a soft error in a circuit node) of a circuit [12]. Since the critical charge is proportional to system supply voltage [29], when system supply voltage is reduced, the critical charge decreases and low energy cosmic particles could cause an error. Considering the number of particles with lower energy is much more than that of particles with higher energy in the cosmic rays [44], scaling down voltages and frequencies for energy savings could lead to dramatically increased transient fault rates [41]. Therefore, voltage scaling has a severe effect on system reliability [9, 31, 41] and should be carefully

evaluated before it is applied, especially for safety-critical embedded real-time applications, such as satellite and surveillance systems, where both high level of reliability and low energy consumption are important.

Traditionally, to achieve a certain level of system reliability in the worst case, only static slack in a system has been explored as temporal redundancy. However, as real-time applications exhibit large variations in actual execution time, and in many cases, only consume a small fraction of their worst case execution time [10], large amount of dynamic slack is available during run-time. As mentioned earlier, simply reclaiming this dynamic slack for energy savings through voltage scaling technique could dramatically reduce system reliability due to increased failure rates as well as extended execution time [9, 41]. Therefore, for dependable embedded real-time systems (such as the ones deployed in out-space explorers), where both high system reliability and low energy consumption are equally important, special considerations are needed when exploiting dynamic slack for energy savings.

Though fault tolerance through redundancy and energy management through voltage and frequency scaling have been well studied in the context of real-time systems independently, there are relatively less research addressing the combination of fault tolerance and energy management [7, 8, 21, 27, 33, 39]. In this work, we propose schemes that utilize dynamic slack for energy savings while taking system reliability into consideration. Specifically, the proposed *reliability-aware* energy management schemes *dynamically* schedule recoveries for tasks to be scaled down using dynamic slack to recuperate the reliability loss due to energy management. To the best of our knowledge, this is the first work that addresses the complications of exploring dynamic slack for both energy and reliability. The main contributions of this paper are three-fold:

- First, we propose a reliability-aware dynamic energy management scheme that can achieve significant energy savings without degrading system reliability.
- Second, depending on the amount of available slack and the size of the application, we identify the situation when it is profitable to reclaim dynamic slack for energy savings without sacrificing system reliability.
- Third, checkpointing techniques are further explored for the reliability-aware dynamic energy management scheme to efficiently use dynamic slack.

Analytical and simulation results show that ignoring the effects of voltage scaling on fault rates changes could lead to drastically decreased system reliability and the proposed schemes can achieve comparable energy savings as ordinary energy management schemes while preserving system reliability.

The reminder of this paper is organized as follows. The models and problem description are presented in Section 2. Reliability-aware dynamic energy management is proposed and analyzed in Section 3 and Section 4 explores checkpointing techniques to efficiently use dynamic slack. The simulation results are presented and discussed in Section 5. Section 6 addresses the closely related work and Section 7 concludes the paper.

2 Models and Problem Description

2.1 Power Model

For embedded systems, the power is consumed mainly by the processor, memory, I/O interfaces and underlying circuits. While the power consumption is dominated by dynamic power dissipation, which is quadratically related to supply voltage and linearly related to frequency [4], the static leakage power is ever-increasing and cannot be ignored, especially with the scaled feature size and increased levels of integration [17, 32]. To incorporate all the power consuming components in an embedded system while keeping the power model simple, the power consumption in a system is divided into two major components: *static power* and *active power* [41, 43].

The static power, which may be removed only by powering off the whole system, includes (but not limited to) the power to maintain basic circuits, keep the clock running and the memory in sleep mode [19]. The active power is further divided into two parts: *frequency-independent* active power and *frequency-dependent* active power. Frequency-independent active power consists of part of memory and processor power as well as any power that can be efficiently removed by putting systems into sleep state(s) and is independent of system supply voltages and processing frequencies [6, 19]. Frequency-dependent active power includes processor's dynamic power and any power that depends on system supply voltages and processing frequencies [4, 32]. Considering the almost linear relation between supply voltage and operating frequency [4], *voltage scaling* reduces the supply voltage for lower frequencies [23]. In this paper, we use frequency changes to stand for changing both supply voltage and frequency and adopt the power model developed in [41, 43]:

$$\begin{aligned} P &= P_s + \hbar(P_{ind} + P_d) \\ &= P_s + \hbar(P_{ind} + C_{ef} f^m) \end{aligned} \tag{1}$$

where P_s is the static power, P_{ind} is the frequency-independent active power and P_d is the frequency-dependent active power. Both P_s and P_{ind} are system dependent *constants*. $\hbar = 1$ if the system is *active* (defined as having computation in progress); otherwise (i.e., the system is in sleep mode or turned off) $\hbar = 0$. The effective switching capacitance C_{ef} and the dynamic power exponent m (in general, larger than or equal to 2) are system/application dependent constants [4] and f is the processing frequency. For easy discussion, normalized frequencies are used and the maximum frequency f_{max} is assumed to be 1 (with corresponding normalized supply voltage $V_{max} = 1$). The maximum frequency-dependent active power is denoted by P_d^{max} and we assume $P_s = \alpha P_d^{max}$ and $P_{ind} = \beta P_d^{max}$.

From Equation (1), intuitively, lower frequencies result in less frequency-dependent active energy consumption. But with reduced speed, the application will run longer and thus consume more static energy and frequency-independent active energy. Hence, in general, an *energy-efficient frequency below which voltage*

scaling starts to consume more total energy does exist[1]. In real-time applications, the time and energy overhead of turning on/off a device that is actively used by the application may be prohibitive [3]. For the time interval considered (e.g., within application's deadline), we assume that the working system is always on (but several components may be put to low-power sleep states for energy savings) and P_s is always consumed. Consequently, the total energy consumption of a *running* application at frequency f can be modeled as:

$$E = P_s \cdot D + (P_{ind} + C_{ef} f^m) \cdot \frac{c}{f} \tag{2}$$

where D is the operation interval, c is the worst case execution time of the application at the maximum frequency f_{max} and $\frac{c}{f}$ is the execution time of the application at frequency f. That is, although P_s affects the *total* energy consumption, the amount of *energy savings* from voltage scaling is *independent* of it. For easy discussion, in what follows, we assume $P_s = 0$ and focus on active power[2].

From Equation 2, it is easy to find out that the *energy efficient frequency* is [41, 43]:

$$f_{ee} = \sqrt[m]{\frac{\beta}{m-1}} \tag{3}$$

For energy consideration, we should never run at a frequency below f_{ee}, since doing so consumes more energy. For simplicity, we assume that $f_{ee} \geq f_{low}$, where f_{low} is the lowest frequency in the system, and define the *minimum energy efficient frequency* as $f_{min} = \max\{f_{low}, f_{ee}\} = f_{ee}$. Moreover, frequency is assumed to be able to change continuously[3] from f_{max} to f_{min}.

2.2 Fault Model

During the execution of an application, a fault may occur due to various reasons, such as hardware failures, software errors and the effects of cosmic ray radiations. Since *transient* faults occur much more frequently than *permanent* faults [5, 15, 16], in this paper, we focus on transient faults, especially the ones caused by cosmic ray radiations, and explore *backward recovery* techniques to tolerate them. It is assumed that faults are detected using sanity or consistency checks [25]. Should an error be detected, the system's state is restored to a previous safe state and the computation is repeated.

Transient faults that are caused by radiations in semiconductor circuits have been known and well studied since the late 1970s [44]. However, considering the various factors that affect the transient fault rate (such as cosmic ray flux, technology feature size, chip capacity, supply voltage and operating frequency), obtaining a

[1]We note that this conclusion has been also reached by several research groups, though through different energy modeling techniques [8, 11, 17, 13, 28].

[2]For systems with multiple processing units, some processing units may be powered off for energy efficiency and the energy savings will be affected by P_s [36, 42].

[3]For discrete frequency levels, we can use two adjacent levels to emulate the execution at any frequency [14].

precise and formal model is an extremely challenging task [30, 31, 45]. In general, transient fault rate, also known as *soft error rate (SER)*, is exponentially-related to the *critical charge* (Q_{crit}), of a circuit is given by the following equation [12]:

$$SER \propto F \times A \times e^{-\frac{Q_{crit}}{Q_s}} \tag{4}$$

where Q_{crit} is the smallest charge needed to cause a soft error; A and Q_s are circuit-related constants; and F is the neutron flux (i.e., radiation intensity). Moreover, the critical charge is proportional to system supply voltage [29]. When the system supply voltage is reduced, the critical charge decreases, which will increase the transient fault rate. For example, increased SERs have been observed with lower supply voltages for both memory [45] and processors [29].

Based on these observations, we have studied the effects of low power techniques on transient fault rates [41]. Assuming that radiation-induced transient faults follow a Poisson distribution with an average fault rate λ [38], for systems running at frequency f ($f \leq f_{max}$, and corresponding supply voltage V), the function giving the average transient fault rate is generally expressed as [41]:

$$\lambda(f) = \lambda_0 g(f) \tag{5}$$

where λ_0 is the average fault rate corresponding to the maximum frequency $f_{max} = 1$ (and supply voltage V_{max}). That is $g(f_{max}) = 1$.

When supply voltage is scaled down, with smaller critical charges, lower energy particles could cause an error with a *higher* probability [44]). Considering the exponential term in Equation (4), and the fact that the number of low-energy particles is two magnitude higher than that of the high-energy particles [44], a more *specific* fault rate model for voltage scaling has been suggested in our previous study [41]:

$$\lambda(f) = \lambda_0 g(f) = \lambda_0 10^{\frac{d(1-f)}{1-f_{min}}} \tag{6}$$

where d (> 0) is a constant. The maximum average fault rate is assumed to be $\lambda_{max} = \lambda_0 10^d$, which corresponds to the lowest frequency f_{min} (and supply voltage V_{min}). That is, reducing the supply voltage and frequency for energy savings results in *exponentially* increased fault rates and larger d indicates that the fault rate is more sensitive to voltage scaling. Although the exponential fault rate model is used in the analysis and simulations, the reliability-aware energy management schemes proposed in this paper are very generic and do not rely on any specific fault model.

2.3 Problem Description

In this work, we consider a real-time application that consists of a set of *aperiodic* tasks. The worst case execution time (WCET) of task T_i at the maximum frequency f_{max} is assumed to be c_i with a deadline D_i ($i = 1, \cdots, n$). When all tasks use their WCETs at the maximum frequency f_{max}, the task set is assumed to be scheduleable. Moreover, considering that the reliability of a real-time system depends on the correct execution of *all* tasks in an application, without loss of generality, the application reliability, $R_0 = \prod_{i=1}^{n} R_i^0$, is assumed to be *satisfactory*. Here, $R_i^0 = e^{-\lambda_0 c_i}$ is the probability of task T_i being executed correctly (from the Poisson fault arrival pattern and the average fault rate λ_0). That is, no recovery tasks are *statically* scheduled to achieve the required reliability[4] R_0, which will be preserved at run-time.

Due to early completion of tasks at run time, dynamic slack will exist during the execution of tasks [10]. **For a given amount of available slack S, we focus on the problem of how to use S for energy savings without sacrificing system reliability, while taking the effects of voltage scaling on fault rates into consideration**.

In order to preserve the reliability, R_0, of an application, for simplicity, we focus on maintaining the reliability of *individual* tasks in this work. That is, we propose schemes to keep the probability of task T_i being correctly executed no less than R_i^0 ($i = 1, \cdots, n$). Recoveries of tasks will be scheduled *dynamically* if needed. The overall performance of the proposed schemes for the whole application will be evaluated through simulations in Section 5.

When errors are detected at a task's completion, the task may be *re-executed* to recover from transient faults. In the next Section, we first consider the case where the amount of available slack S is no less than c_k, the size of the next task T_k, and propose a *reliability-aware* dynamic energy management scheme which *dynamically* schedules a recovery task (i.e., a simple re-execution) for T_k to recuperate the possible reliability loss due to energy management. Section 4 further explores checkpointing techniques to efficiently use the available slack, especially for the case when S is smaller than c_k.

3 Reliability-Aware Dynamic Energy Management

Although sophisticated dynamic power management schemes that explore tasks' statistical information have been proposed [2, 22], we will focus on *greedy* scheme for it's simplicity. Exploring other advanced schemes is beyond the scope of this paper and will be considered in our future work. We first illustrate the problem of ordinary greedy power management on reliability in Section 3.1. Then Section 3.2 presents the new reliability-aware greedy energy management scheme and the analysis.

[4]When recovery tasks are *statically* scheduled to satisfy higher levels of reliability requirements, our proposed schemes will treat such recovery tasks as normal tasks and preserve the higher levels of reliability that should be achieved.

3.1 Ordinary Greedy Power Management

In *ordinary greedy* power management, all the available dynamic slack will be used to scale down the processing of the next task for energy savings provided that such allocation complies with task's timing constraints [2, 22] and/or the minimum energy-efficient frequency limitation [41, 43]. For example, as shown in Figure 1a, due to the early completion of previous tasks, there are 3 units of available dynamic slack at time t, that is, $S = 3$. The WCET of the next ready task T_k is $c_k = 2$. In the figures, the X-axis represents time, the Y-axis represents processing frequency (e.g., cycles per time unit), and the area of the task box defines the workload (e.g., number of cycles) of the task. Recall that D_k is the deadline of task T_k.

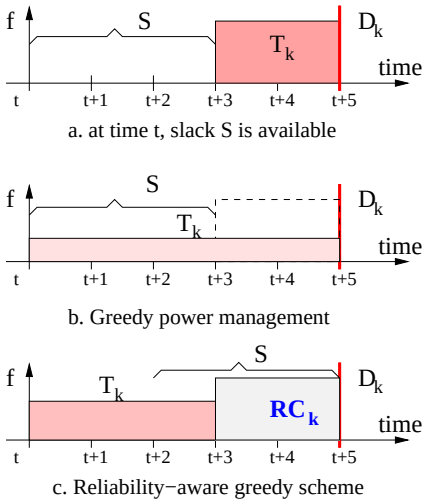


Figure 1: Ordinary and Reliability-Aware Greedy Schemes.

Suppose that $\beta = 0.1$ (i.e., $P_{ind} = 0.1P_d^{max}$) and $m = 3$, we have the minimum energy efficient frequency $f_{ee} = 0.37$ (recall that $f_{max} = 1$, Section 2) [43]. Therefore, all the available dynamic slack S can be allocated to task T_k and the processing speed of T_k can be reduced from $f_{max} = 1$ to $f = \frac{2}{2+3} = 0.4$ as shown in Figure 1b. From Equation 1, it is easy to find that scaling down the processing of T_k could save 63% of the *active energy*.

However, as discussed in Section 2, with reduced processing frequency and supply voltage, the processing of task T_k is more susceptible to transient faults [9, 41]. Suppose that the exponent in the fault rate model is $d = 2$ (see Equation 6 in Section 2), the probability of having fault(s) during the execution of task T_k at the reduced speed will be:

$$\begin{aligned} & \rho_k = 1 - R_k = 1 - e^{-\lambda_0 10^{\frac{d(1-f)}{1-f_{min}}}(S+c_k)} \\ = \;& 1 - e^{-\lambda_0 10^{\frac{d(1-f)}{1-f_{min}}}\frac{c_k}{f}} = 1 - e^{-\lambda_0 c_k 10^{\frac{2(1-0.4)}{1-0.37}}\frac{1}{0.4}} \\ \approx \;& 1 - (R_k^0)^{200} = 1 - (1-\rho_k^0)^{200} \approx 200\rho_k^0 \end{aligned} \tag{7}$$

where ρ_k^0 is the probability[5] of having fault(s) when task T_k uses its WCET at the maximum processing frequency f_{max}. That is, though 63% active energy is saved by scaling down the processing of task T_k, it leads to approximately 200 times higher in the probability of failure! The increase in the probability of failure during the processing of individual tasks will degrade the overall system reliability, which is unbearable, especially for safety-critical systems where the requirement for high levels of reliability is strict.

3.2 Reliability-Aware Greedy Scheme

In addition to being used by energy management schemes for energy savings, slack time can also be used as temporal redundancy to increase system reliability [25]. To recuperate the reliability loss due to energy management, we can reserve some slack as temporal redundancy for scheduling recoveries/backups for tasks to be scaled down. For simplicity, here we assume that the recovery is in the form of *re-execution*[6] and it has the same size of the task to be recovered.

The *reliability-aware greedy (RA-Greedy)* power management scheme will dynamically schedule a *recovery* for the task to be scaled before applying slack reclamation for energy savings. The recovery will be executed (if needed) at the maximum frequency $f_{max} = 1$. Notice that, in this section, the amount of dynamic slack S is assumed to be no less than c_k, the size of next task T_k. After reserving c_k units of dynamic slack for the recovery task, the remaining dynamic slack ($S - c_k$, if any) can be used to scale down the execution of T_k for energy savings. For example, as shown in Figure 1c, a recovery task RC_k is scheduled for task T_k which uses 2 units of dynamic slack. The remaining 1 unit of dynamic slack allows task T_k to run at a lower frequency $f_k = \frac{2}{2+1} = 0.66$ and save energy.

3.2.1 System Reliability under RA-Greedy

With the additional recovery task RC_k, the reliability R_k of task T_k will be the summation of the probability of primary task T_k being executed correctly and the probability of having fault(s) during T_k's execution while RC_k being executed correctly. Notice that, if the execution of the primary task T_k is faulty, the recovery task RC_k will be executed at the maximum frequency f_{max} and the probability of its *fault-free* execution is $e^{-\lambda_0 c_k} = R_k^0$. Therefore, we have:

$$R_k = e^{-\lambda(f_k)S} + \left(1 - e^{-\lambda(f_k)S}\right) R_k^0 > R_k^0 \tag{8}$$

where $\lambda(f_k)$ is the fault rate at the reduced frequency f_k. From the above equation, we can see that, under the RA-Greedy scheme, with the help of the additional recovery task RC_k, the reliability of task T_k is *always* better than R_k^0 regardless different fault rate increases (i.e., different values of d in Equation 6) and the reduced

[5]Note that, ρ_k^0 is a small number (usually $< 10^{-4}$).

[6]Notice that the approach can be generalized to other settings with different recoveries [1].

processing frequency f_k of the primary task T_k. That is, when the amount of dynamic slack is no less than the size of the next task, by dynamically scheduling a recovery task before applying energy management, the RA-Greedy scheme can achieve better reliability for individual tasks, and thus preserve system reliability.

3.2.2 Expected Energy Consumption under RA-Greedy

Suppose that the energy consumption to execute task T_k for time c_k at the maximum frequency f_{max} is[7] $E_k^0 = (P_s + P_{ind} + P_d^{max})c_k = (P_{ind} + P_d^{max})c_k = (\beta + 1)P_d^{max}c_k$. Considering the probability of RC_k being executed, the *expected energy consumption* for processing task T_k will be:

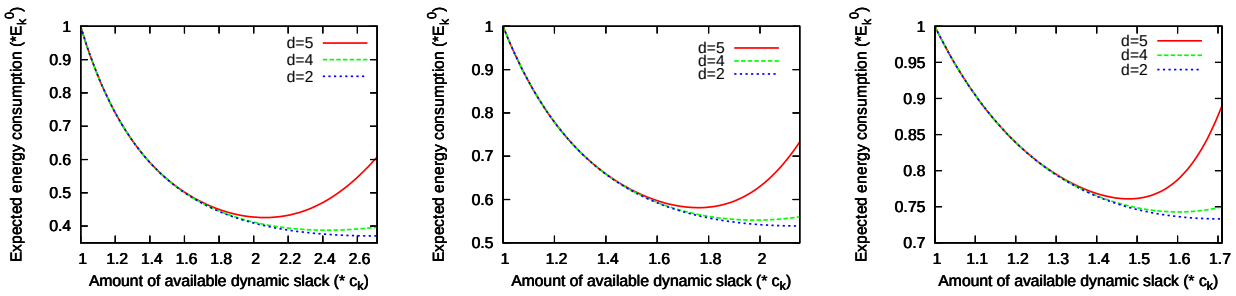
$$\begin{aligned} E_k &= (P_{ind} + C_{ef} f_k^m)S + (1 - e^{-\lambda(f_k)S}) \cdot E_k^0 \\ &= E_k^0 \left[1 - e^{-\lambda(f_k)S} + \frac{(\beta + \frac{c_k^m}{S^m})\frac{S}{c_k}}{1 + \beta} \right] \end{aligned} \tag{9}$$

Intuitively, the more the available dynamic slack is allocated for energy management, the lower the processing frequency can be for executing task T_k, and thus more energy savings can be obtained. However, due to the limitation of the minimum energy efficient frequency f_{ee}, the maximum amount of dynamic slack that should be allocated to task T_k for energy management is limited, which can be easily calculated as $\frac{c_k}{f_{ee}} - c_k$. Consider the amount of slack reserved for recovery is c_k, the maximum amount of total dynamic slack that may be used when processing T_k will be $US_{max} = (\frac{c_k}{f_{ee}} - c_k) + c_k = \frac{c_k}{f_{ee}}$. When more dynamic slack than US_{max} is available, part of the slack will be saved for future tasks due to energy consideration.

Moreover, with reduced processing frequency and supply voltage, the execution of T_k takes more time and the fault rate increases, which results in higher probability of having fault(s) during the execution of T_k. Therefore the probability of recovery task RC_k being executed increases, which may *overshadow* the energy savings and lead to more expected energy consumption. However, considering the exponential component in Equation 9, it is hard to obtain a simple closed formula for the optimal amount of dynamic slack that minimizes the expected energy consumption. In what follows, we present some analytical results to illustrate the relation between the expected energy consumption, the amount of available dynamic slack and the fault rate changes due to energy management.

Without loss of generality, in the analysis, we assume $c_k = 1$ and $\lambda_0 = 10^{-6}$ (which corresponds to 100,000 FITs, *failure in time* in terms of errors per billion hours of use per megabit, that is a reasonable fault rate as reported [30, 45]). Moreover, we assume $\alpha = 0$ (i.e., $P_s = 0$) and $m = 3$. Figure 2 shows the expected energy consumption for executing task T_k, normalized to E_k^0, versus the amount of available dynamic slack under different frequency-independent power (β) and fault rate changes (d). Notice that, one unit (c_k) of dynamic slack is reserved for the recovery task and the minimum amount of dynamic slack considered is c_k.

[7] Recall that we assume $P_s = 0$ and focus on active power in this paper.



a. $\beta = 0.1$ b. $\beta = 0.2$ b. $\beta = 0.4$

Figure 2: The normalized expected energy consumption vs. the amount of available dynamic slack.

From Section 2, for different frequency-independent power $\beta = 0.1, 0.2$ and 0.4, the corresponding energy efficient frequencies are $f_{ee} = 0.37, 0.46$ and 0.58, which in turn limits the maximum amount of dynamic slack used by RA-Greedy scheme $US_{max} = \frac{c_k}{f_{ee}}$ to be $2.70c_k$, $2.17c_k$ and $1.72c_k$, respectively (see the X-axis in the figure).

From the figures we can see that, for the different fault rate increases considered (i.e., $d = 2, 4$ and 5), when the amount of available dynamic slack is more than c_k $(= 1)$, the size of the next task T_k, dynamic slack is available for energy management. The expected energy consumption to execute T_k is less than E_k^0 and up to 60% of energy savings is expected when $\beta = 0.1$ and $d < 4$. As the amount of available dynamic slack increases, more slack is available for energy management and the expected energy consumption for executing task T_k generally decreases. However, when the fault rate increases dramatically with reduced processing frequencies and supply voltages (e.g., $d = 5$), as more dynamic slack is available and the reduced frequency approaches f_{ee}, more expected energy may be consumed due to the increased probability of recovery task being executed. In this case, the optimal amount of dynamic slack to minimize expected energy consumption is less than US_{max}.

Notice that, when the fault rate change is not that severe (e.g., $d \leq 4$), the maximum amount of dynamic slack US_{max} limited by f_{ee} is very close to the optimal amount of slack that minimizes the expected energy consumption. Considering the difficulty of finding the close formula for the optimal amount of slack, for the simulations in Section 5, the amount of dynamic slack that will be allocated for energy management is only limited by f_{ee} (i.e., up to $US_{max} - c_k$ amount of dynamic slack will be used for energy management). Moreover, for higher frequency-dependent active power (i.e., $\beta = 0.4$), f_{ee} increases and US_{max} decreases, which results in less energy savings (note the difference in the scale of Y-axis of Figure 2).

We have shown that the RA-Greedy scheme can achieve significant energy savings while guaranteeing to preserve system reliability regardless the negative effects of voltage scaling on transient fault rates. However, the amount of slack needed by the RA-Greedy scheme may be considerable, especially for large tasks. To utilize the slack more efficiently, instead of scheduling a whole recovery task, checkpoints may be employed

to re-compute only the faulty section for more energy savings as well as better system reliability [18, 20, 21].

4 Checkpointing for Better Performance

Checkpointing techniques insert checkpoints during the execution of a task. Within a checkpoint, the state of a system is checked and correct states are saved to a stable storage [25]. When faults are detected, the execution is rolled back to the latest correct checkpoint and re-compute the faulty section by exploring the temporal redundancy [18, 20]. Checkpoints can be *uniformly* or *non-uniformly* distributed among an application [21]. In this work, we consider uniformly distributed checkpoints only.

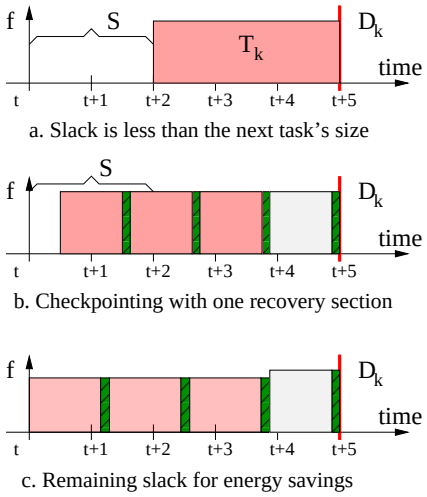


Figure 3: Reliability-Aware Energy Management with Checkpoints.

For example, Figure 3a shows that there are 2 units of dynamic slack available at time t, which is less than $c_k = 3$, the size of the next ready task T_k. Apparently, the RA-Greedy scheme could not use this slack and it may be wasted (if T_k is the last task). If the overhead of employing one checkpoint is $r = 0.125$ and 3 checkpoints are inserted, Figure 3b illustrates the case of one recovery section being scheduled. Here there is 0.5 units of remaining dynamic slack, which can be used to scale down the processing of the primary task sections for energy savings as shown in Figure 3c.

The more the checkpoints are, the smaller a task section is. Thus, less slack is needed for recoveries. However, checkpoints also take time and consume energy. Therefore, there is a tradeoff regarding the number of checkpoints that should be employed to minimize the response time [18] or energy consumption [21]. For simplicity, we focus on the optimal number of checkpoints that minimize the recovery overhead and explore the efficient usage of the slack to save energy while preserving system reliability. That is, **for given amount of available slack S and the next task T_k, we study the relation between the checkpoint overhead, the amount of energy savings and the number of recovery sections needed to preserve system reliability**.

4.1 Checkpoints with Single Recovery Section

To compensate the reliability loss due to energy management and checkpoint overhead, *at least* one recovery section is needed. In this Section, we consider first the simple case that only has a single recovery section and analyze its performance on system reliability. When one recovery is not enough to recuperate the reliability loss, the analysis is further generalized to multiple recovery sections in Section 4.2.

For easy discussion, we assume that the overhead of taking one checkpoint is $r = \gamma \cdot c_k$, where c_k is the WCET of the next task T_k. If n checkpoints are inserted during the execution of T_k, the size of one recovery section will be $\frac{c_k}{n}$ and we have:

$$S \geq n \cdot r + (r + \frac{c_k}{n}) = (n\gamma + \gamma + \frac{1}{n})c_k \tag{10}$$

In order for n to have a real (non-imaginary) solution, we can easily find that the minimum amount of slack needed due to timing constraints is $S_{min}^{time} = (\gamma + 2\sqrt{\gamma})c_k$ with the optimal number of checkpoints being $n_{opt} = \left\lfloor \sqrt{\frac{1}{\gamma}} \right\rfloor$ or $n_{opt} = \left\lceil \sqrt{\frac{1}{\gamma}} \right\rceil$. However, considering the integer property of n_{opt} and the energy overhead incurred by checkpoints, the minimum amount of slack needed for energy savings S_{min}^{energy} should be larger than S_{min}^{time} as illustrated in Section 4.2.2.

With the optimal number of checkpoints n_{opt} and one recovery section, the amount of available slack for energy management will be $S - (n_{opt} + 1)r - \frac{c_k}{n_{opt}}$, which can be used to scale down the execution of the primary sections. Therefore, the reduced frequency to execute the primary sections will be

$$f_{ckpt} = \frac{c_k + n_{opt} \cdot r}{S + c_k - r - \frac{c_k}{n_{opt}}} \tag{11}$$

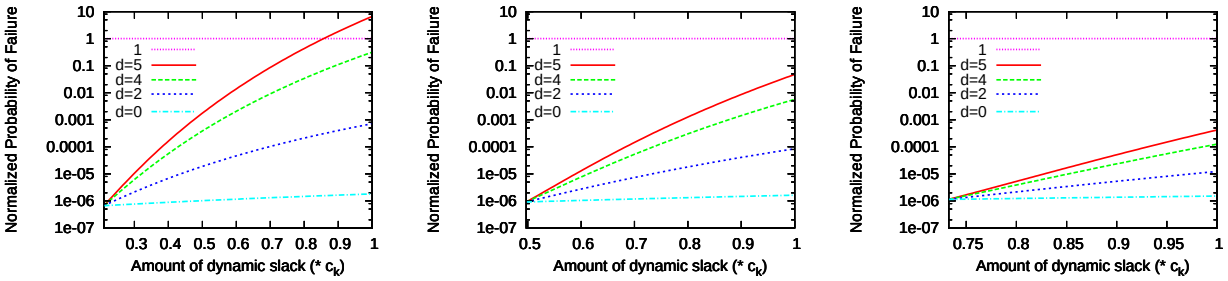
and each primary section will take $t_{primary} = \frac{S + c_k - r - \frac{c_k}{n_{opt}}}{n_{opt}}$ time units. From Section 2, the fault rate at frequency f_{ckpt} will be $\lambda(f_{ckpt}) = \lambda_0 10^{\frac{d(1-f_{ckpt})}{1-f_{min}}}$ and the probability of having fault(s) during the execution of one primary section is $\rho_{primary} = 1 - e^{-\lambda(f_{ckpt})t_{primary}}$. Notice that, the recovery section is executed at f_{max} and the probability of having fault(s) during the execution of the recovery section is $\rho_{recovery} = 1 - e^{-\lambda_0(r + \frac{c_k}{n_{opt}})}$. Therefore, the reliability of executing task T_k is

$$R_k^{ckpt} = (1 - \rho_{primary})^{n_{opt}} + n_{opt} \cdot \rho_{primary}(1 - \rho_{primary})^{n_{opt}-1}(1 - \rho_{recovery}) \tag{12}$$

where the first part is the probability of all primary sections being executed correctly and the second part is the probability of having fault(s) during the execution of one primary section while the recovery section being executed correctly.

From Equation 12, R_k^{ckpt} is determined by the amount of available dynamic slack S, checkpoint overhead r and fault rate changes d. For a given checkpoint overhead, more dynamic slack leads to lower reduced

frequency for the primary sections, which in turn leads to higher probability of failure and lower reliability R_k^{ckpt}. However, due to the complexity of Equation 12, it is hard to find the close formula for S to ensure $R_k^{ckpt} \geq R_k^0$ and we illustrate the relation between S and R_k^{ckpt} in the following analysis.



a. $\gamma = 0.01$ b. $\gamma = 0.05$ c. $\gamma = 0.10$

Figure 4: The normalized expected energy consumption vs. the amount of available dynamic slack.

Figure 4 shows the normalized probability of failure, $\frac{1-R_k^{ckpt}}{1-R_k^0}$, when executing task T_k with different amount of available dynamic slack under different checkpoint overheads. Here, we limit the analysis to the case of $S < c_k$ since the RA-Greedy scheme can guarantee system reliability when $S > c_k$. The same as before, we assume $m = 3$, $\lambda_0 = 10^{-6}$ and $c_k = 1$. Moreover, β is assumed to be 0.1 and we have $f_{ee} = 0.37$. Therefore, for a given checkpoint overhead $r = \gamma c_k$, the amount of dynamic slack considered will be in the range of S_{min}^{time} $(= \gamma + 2\sqrt{\gamma})$ and 1.

From the figure, we can see that, with one recovery section, the normalized probability of failure to execute task T_k is lower than 1 most of the time. The exception comes from the case where the checkpoint overhead is low (i.e., $\gamma = 0.01$; see Figure 4a) which leaves more slack for energy management and the reduced frequency is close to f_{ee}. With the exponent of fault rate model being $d = 5$, the fault rate at f_{ee} is 10^5 time higher than $\lambda_0 = 10^{-6}$ and leads to worse than R_k^0 reliability. However, with moderate fault rate increase (e.g., $d \leq 4$), for the cases we considered, adding checkpoints with one recovery section obtains higher reliability when executing task T_k.

Moreover, the faster the fault rate increases (i.e., larger values of d) with reduced frequencies and supply voltages, the higher the probability of failure and the lower the reliability. Assuming constant fault rate (e.g., $d = 0$) is too optimistic and could lead to lower reliability than expected when exploring slack for energy management, which is the same observation as our previous results [41].

From the above analysis, we can see that checkpointing with single recovery section may not be enough to compensate the reliability loss due to energy management. In the next Section, we consider to exploit multiple recovery sections to enhance reliability, especially for the case of more slack being available (e.g., $S > c_k$).

4.2 Checkpoints with Multiple Recovery Sections

Suppose that b (≥ 1) recovery sections are needed/scheduled to preserve system reliability. Equation 10 can be generalized as:

$$S \geq n \cdot r + b \cdot (r + \frac{c_k}{n}) = \left((n+b)\gamma + \frac{b}{n}\right) c_k \tag{13}$$

With the optimal number of checkpoints $n_{b,opt}$ (i.e., $\left\lfloor \sqrt{\frac{b}{\gamma}} \right\rfloor$ or $\left\lceil \sqrt{\frac{b}{\gamma}} \right\rceil$), the minimum amount of slack needed due to timing constraints can be found as $S^{time}_{b,min} = (b \cdot \gamma + 2\sqrt{b \cdot \gamma})c_k$. When more slack is available, the remaining slack $S - n_{b,opt} \cdot r - b(r + \frac{c_k}{n_{b,opt}})$ (if any) can be used by energy management schemes to scale down the execution of the primary sections for energy savings. The reduced frequency to execute the primary sections will be

$$f_{b,ckpt} = \frac{c_k + n_{b,opt} \cdot r}{c_k + S - b(r + \frac{c_k}{n_{b,opt}})} \tag{14}$$

4.2.1 Reliability for Multiple Recovery Sections

From Equation 14, each primary section will take $t_{b,primary} = \frac{c_k + S - b(r + \frac{c_k}{n_{opt}})}{n_{opt}}$ time units. Considering that the fault rate at frequency $f_{b,ckpt}$ is $\lambda(f_{b,ckpt}) = \lambda_0 10^{\frac{d(1-f_{b,ckpt})}{1-f_{min}}}$ (see Section 2), the probability of having fault(s) during the execution of one primary section is $\rho_{b,primary} = 1 - e^{-\lambda(f_{b,ckpt})t_{b,primary}}$. Notice that, the recovery sections are executed at f_{max} and the probability of having fault(s) during the execution of one recovery section is $\rho_{b,recovery} = 1 - e^{-\lambda_0(r + \frac{c_k}{n_{b,opt}})}$.

With b recovery sections, the reliability $R_k(b)$ of task T_k will be the summation of the probability of *all* primary sections being executed correctly and the probability of having *any* x ($x = 1, \ldots, b$) faulty primary sections while there are x recovery sections being executed correctly. Notice that, the recovery sections are activated *one at a time* when they are needed. For example, to recover x faulty primary sections, if there is no faults during the execution of the first x recovery sections, it is not necessary to invoke the remaining recovery sections. Otherwise, the next recovery section is activated and so on, until there are x recovery sections are correctly executed. Therefore, we have

$$R_k(b) \;=\; \sum_{x=0}^{b} \left[\binom{n_{b,opt}}{x} \cdot (1 - \rho_{b,primary})^{n_{b,opt}-x} \cdot \rho^x_{b,primary} \cdot Pr(b, x, \rho_{b,recovery}) \right] \tag{15}$$

where $\binom{n_{b,opt}}{x}$ is the number of combinations of having x faulty sections in the $n_{b,opt}$ primary sections. $Pr(b, x, \rho_{b,recovery})$ is the probability of x faulty primary sections being successfully recovered by the b recovery sections. Notice that, when $x = 0$, no faulty primary section needs to be recovered and there is

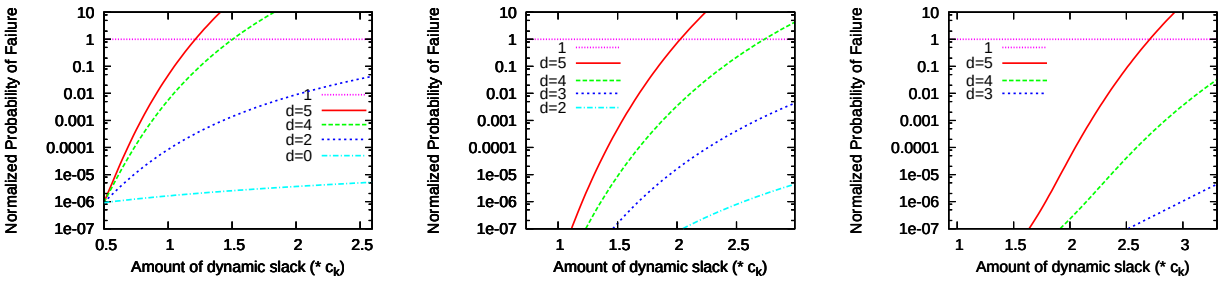
$Pr(b, 0, \rho_{b,recovery}) = 1$. Therefore, we have

$$Pr(b, x, \rho_{b,recovery}) = \begin{cases} 1 & x = 0 \\ (1 - \rho_{b,recovery})^{x-1}(1 - \rho_{b,recovery}^{b-x+1}) & x \geq 1 \end{cases} \tag{16}$$

From Equations 14, 15 and 16, clearly it is not practical to seek the close formula for the minimum number of recovery sections needed to ensure $R_k(b) \geq R_k^0$. In what follows, we show some analysis results regarding the number of recovery sections employed and the reliability achieved versus the amount of available slack.

Notice that, the reduced frequency f_{ckpt} is limited by the energy efficient frequency f_{ee}. From Equation 14, we can find that the maximum amount of slack can be used to processing T_k to be

$$US_{b,ckpt}^{max} = \left(\frac{1 + n_{b,opt} \cdot \gamma}{f_{ee}} + b \left(\gamma + \frac{1}{n_{b,opt}} \right) - 1 \right) c_k \tag{17}$$



a. $b = 1$ b. $b = 2$ c. $b = 3$

Figure 5: The normalized expected energy consumption vs. the amount of available dynamic slack.

Figure 5 shows the normalized probability of failure, $\frac{1-R_k^{ckpt}}{1-R_k^0}$, for different numbers of recovery sections. Here, the same parameters as in last Section are used and the checkpoint overhead is assumed to be $\gamma = 0.05$. From the figures, we can see that, as the number of recovery sections increases, the minimum amount of slack needed increases, which is close to c_k when three recovery sections (i.e., $b = 3$) are scheduled (Figure 5c). Moreover, as the amount of slack increases, more slack is available for energy management which leads to lower processing frequencies, higher fault rates and thus lower levels of reliability achieved, which is the same observation as in Section 4.1.

For the case of low fault rate increase (e.g., $d \leq 2$), when there are three recovery sections (Figure 5c), the normalized probability of failure is less than 10^{-7} and is not shown in the figures, which is the same for $d = 0$ and $b = 2$ in Figure 5b. When the fault rate increase is high (e.g., $d = 5$), even three recovery sections cannot guarantee R_k^0 (i.e., the normalized probability of failure is larger than 1). However, as shown in next Section, for the cases where R_k^0 is preserved, compared with the RA-Greedy scheme, checkpointing will result in smaller recovery sections, and more energy savings may be expected.

4.2.2 Expected Energy Consumption with Checkpoints

With reduced frequency $f_{b,ckpt}$ (see Equation 14), the energy consumption for executing each primary section is $E_{primary} = \left(\beta + \left(\frac{f_{b,ckpt}}{f_{max}}\right)^m\right) P_d^{max} \cdot t_{b,primary}$. Recall that recovery sections are executed (if needed) at the maximum frequency f_{max}. The energy consumption for executing one recovery section will be $E_{recovery} = (\gamma + \frac{1}{n_{b,opt}})E_k^0$, where E_k^0 is the energy consumption to execute task T_k for time c_k at f_{max}.

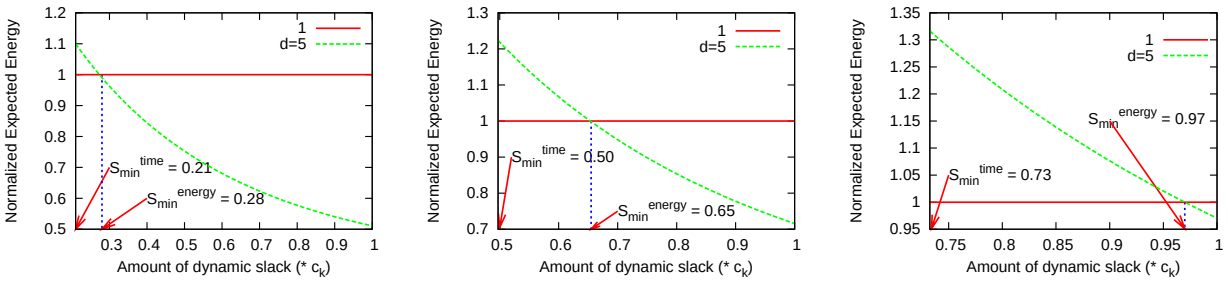
Notice that, for the q^{th} recovery section ($q = 1, \ldots, b$), it will *not* be invoked if the number of faulty primary sections x is less than q (i.e., $0 \leq x < q$) and the first $(q-1)$ recovery sections successfully recover all the x faulty primary sections. From previous discussion, when there are x ($0 \leq x < q$) faulty primary sections, the probability of these faulty primary sections being successfully recovered by the first $(q-1)$ recovery sections can be given by $Pr(q-1, x, \rho_{b,recovery})$. Therefore, considering the probability of each recovery section being executed, the expected energy consumption for executing task T_k will be

$$E_k^{b,ckpt} = n_{opt} \cdot E_{primary} + \sum_{q=1}^{b} (Pr_q \cdot E_{recovery}) \tag{18}$$

where the first part is always consumed and is the energy for executing the primary sections (including the checkpoints), and the second part is the expected energy consumption for executing the recovery sections. Pr_q is the probability of the q^{th} recovery section being invoked, which is given as

$$Pr_q = 1 - \sum_{x=0}^{q-1} \left[\binom{n_{b,opt}}{x} \cdot (1 - \rho_{b,primary})^{n_{b,opt}-x} \cdot \rho_{b,primary}^x \cdot Pr(q-1, x, \rho_{b,recovery}) \right] \tag{19}$$

Due to the overhead of checkpoints, in order to obtain energy savings (i.e., $E_k^{ckpt} < E_k^0$), there is a minimum amount of dynamic slack S_{min}^{energy} needed for energy management. Again, due to the complexity of Equation 18, it is hard to get the close formula for S_{min}^{energy} and we illustrate the relation between S_{min}^{energy} and checkpoint overhead γ in the following analysis.

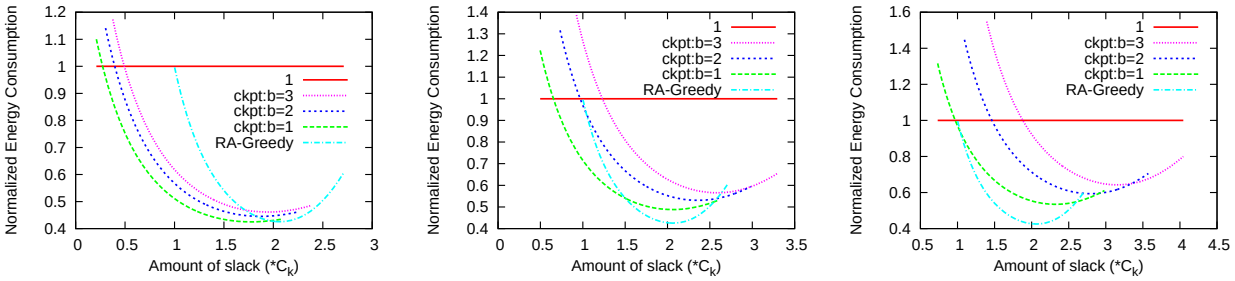


a. $\gamma = 0.01$ b. $\gamma = 0.05$ c. $\gamma = 0.10$

Figure 6: The normalized expected energy consumption vs. the amount of available dynamic slack.

Corresponding to the reliability analysis with one recovery section in Section 4.1, Figure 6 shows the normalized expected energy consumption, $\frac{E_k^{1,ckpt}}{E_k^0}$, for different checkpoint overheads with the amount of slack being limited by c_k. For different fault rate changes (i.e., different values of d), due to the low probability of recovery section being executed (lower than 10^{-5} even when $d = 5$), the expected energy consumption is almost the same for a given checkpoint overhead and amount of available dynamic slack. Therefore, we only show the normalized expected energy consumption for the worst case of $d = 5$.

From the figures, we can see that, although it is feasible to employ checkpoints when the amount of dynamic slack is larger than S_{min}^{time}, due to the energy overhead of checkpoints, no energy savings could be obtained until the amount of slack is more than S_{min}^{energy}. The smaller the checkpoint overhead, the lower the value of S_{min}^{energy} and the more energy savings could be obtained for a given amount of dynamic slack. When the checkpoint overhead is large (e.g., $\gamma = 0.1$, Figure 6c), almost no energy savings could be obtained for the case considered.



a. $\gamma = 0.01$ b. $\gamma = 0.05$ c. $\gamma = 0.10$

Figure 7: The comparison of the normalized expected energy consumption for checkpointing with multiple recovery sections and the RA-Greedy scheme.

Figure 7 further shows the normalized energy consumption for multiple recovery sections considering the extended range of available slack. For comparison, the normalized energy consumption for the RA-Greedy scheme is also shown in the figure. Notice that, the ranges of the amount of slack could be exploited are different for different numbers of recovery sections being employed. From the figure, we can see that, when more than one recovery sections are employed, checkpointing with recoveries consume more energy than the RA-Greedy scheme, except for the case where overhead is small (e.g., $\gamma = 0.01$ in Figure 7a). When checkpoint overhead is relatively large (e.g., $\gamma \geq 0.05$), checkpointing scheme should not be exploited when the RA-Greedy scheme is applicable (i.e., when $S > c_k$).

Considering both reliability (Section 4.2.1) and energy savings (Section 4.2.2), checkpointing should *not* be employed when the checkpoint overhead is relatively large (e.g., $\gamma \geq 0.05$). No more than one recovery section may be employed even when checkpoint overhead is relatively small (e.g., $\gamma = 0.01$). Moreover, although more energy savings could be obtained by checkpointing compared with the RA-Greedy scheme,

limitation may exist on the amount of employed slack due to reliability consideration, especially for the case where the fault rate increases dramatically with reduced frequencies and supply voltages (e.g., $d = 5$).

We have analyzed the performance of the reliability-aware energy management schemes for a single task. In what follows, to illustrate the merits of our proposed schemes and see how they performs for overall system reliability and energy savings, we present simulation results for dependable real-time applications that consist of a set of aperiodic tasks. We compare the energy savings as well as system reliability of the new proposed schemes with ordinary energy management schemes.

5 Simulation Results and Discussion

In the simulations, we consider four different schemes: a) *no power management (NPM)*, which is used as the baseline for comparison; b) *ordinary greedy power management (Greedy)*, which allocates all available dynamic slack for next ready task to save energy without considering system reliability; c) *reliability-aware greedy power management (RA-Greedy)*, which dynamically allocates a recovery for the next ready task before applying greedy power management. When the amount of available dynamic slack is less than the size of next ready task, the slack is not used and saved for future tasks; d) *reliability-aware power management with checkpoints (Ckpt)*, which is the same as RA-Greedy except that checkpoints are employed when the amount of available dynamic slack is less than the size of next ready task. As discussed in last Section, only one recovery section is considered in the simulations.

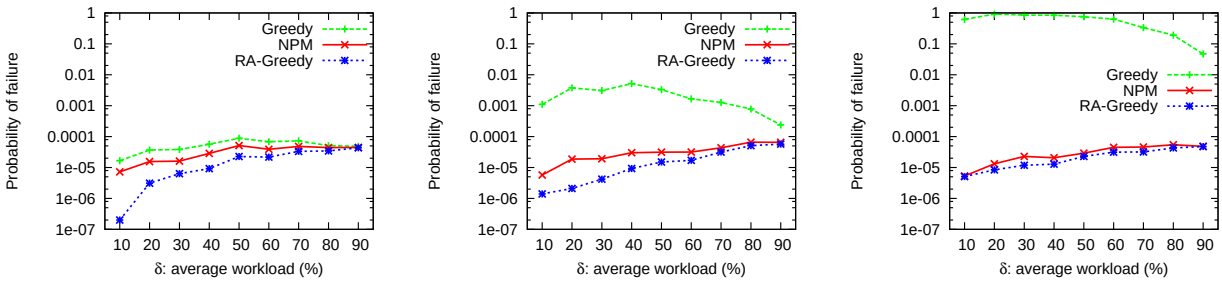
For the system parameters, as discussed in Section 2, we use normalized frequency with $f_{max} = 1$ and assume frequency can be changed continuously. Moreover, corresponding to the analysis in Section 3 and 4, we assume $\alpha = 0$, $\beta = 0.1$ and $m = 3$. That is, we assume the working system is always on and focus on system active power. For the effects of different values of α and β on energy management, see [42, 43] for more discussions. The same as in Section 3, we assume that faults follow a Poisson distribution with an average fault rate as $\lambda_0 = 10^{-6}$ at f_{max} (and corresponding V_{max}). We vary the values of d (as $0, 2$ and 5 respectively) for different changes in fault rates due to the effects of frequency and voltage scaling [9]. An application fails if *any* task in the application fails and there is no recovery *or* both the task and its recovery fail.

The number of tasks in an application is randomly generated between 5 and 20, where the WCETs of tasks are uniformly distributed in the range of 1 and 10. When every task in an application uses its WCET, we assume that the application finishes just in time and the system reliability is satisfactory. To emulate the run-time behaviors of tasks, a parameter σ is used as an application-wide average over worst execution time, which also indicates the amount of dynamic slack available on average during execution. Smaller values of σ imply more dynamic slack. The value of σ_i for task T_i in the application is generated from a uniform distribution with an average value of σ. The actual execution time of T_i follows a similar uniform distribution

with an average value of $\sigma_i \cdot c_i$, where c_i is the WCET of task T_i. For each result point in the graphs, 100 task sets are generated and each task set is executed $100,000$ times, and the result is the average of all the runs.

5.1 Performance of RA-Greedy

First, we compare the performance of *Greedy* and *RA-Greedy* on reliability and energy consumption. For different fault rate changes, Figure 8 shows the probability of failure when executing the applications with different average system loads (i.e., different amounts of dynamic slack).

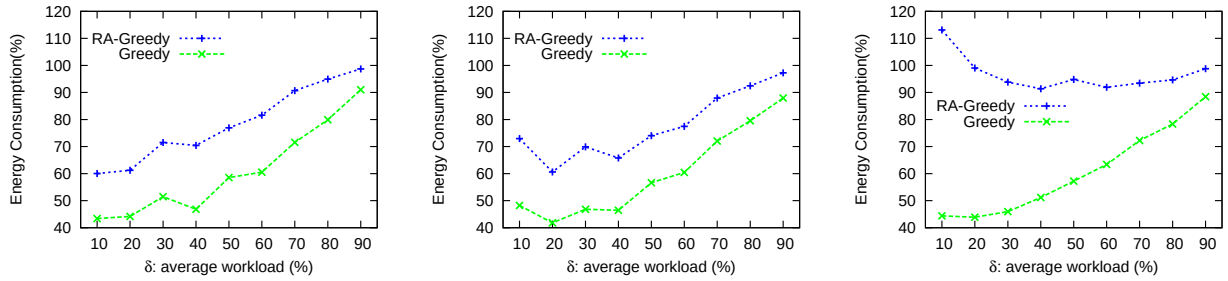


a. $d = 0$ b. $d = 2$ c. $d = 5$

Figure 8: The probability of failure vs. different average system loads.

Notice that, *NPM* executes every task at the maximum frequency f_{max} and the fault rate is always $\lambda_0 = 10^{-6}$. From Figure 8, we can see that for a given average system load, the probability of failure under *NPM* is roughly the same, which is not affected by the different fault rate changes (i.e., different values of d). When the average system load increases, the applications run longer and the probability of failure under *NPM* increases linearly. Note the log scale of Y-axis in Figure 8.

From the figure, it can be seen that the *Greedy* scheme results in higher probability of failure than *NPM* even when $d = 0$ (i.e., constant fault rate), which comes from the extended execution of tasks due to energy management. When fault rate increases with reduced frequencies and supply voltages (i.e., $d > 0$), the probability of failure under *Greedy* scheme increases exponentially as d increases. For example, when $d = 5$, *Greedy* scheme almost always leads to system failure (with probability of failure close to 1), especially for the case of low average system loads where more dynamic slack exists. When the average system load increases, the probability of failure under *Greedy* scheme increases first and then decreases, the reason is because of the limitation of $f_{ee} = 0.37$. When the average system load is extremely low (e.g., $\sigma < 20\%$), tasks in an application always run at f_{ee} and the probability of failure mainly depends on the execution time, which increases as average system load increases. However, as average system load continues to increase, less slack is available and tasks need to run at higher frequencies than f_{ee}, which has lower fault rates and thus leads to lower probability of failure. Moreover, from Figure 8, we can also see that *RA-Greedy* scheme always has a lower probability of failure (i.e., higher system reliability) than *NPM* regardless the fault rate changes, which coincides with the analysis in Section 3.2.1.



a. $d = 0$ b. $d = 2$ c. $d = 5$

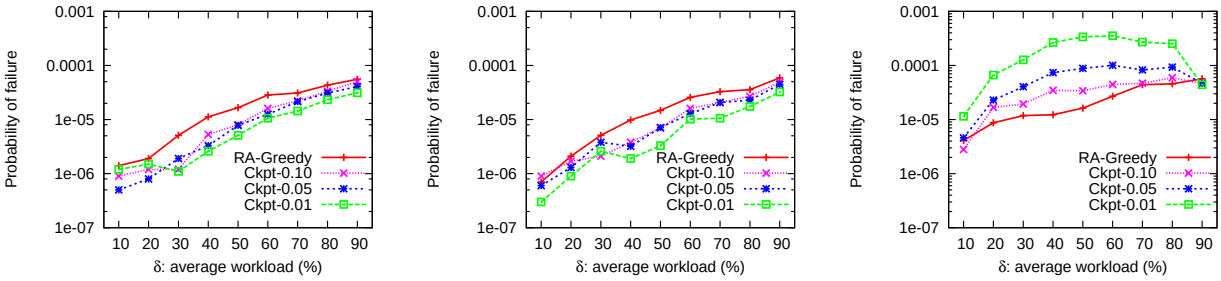
Figure 9: The normalized expected energy consumption vs. average system loads.

Figure 9 shows the corresponding normalized energy consumption for *Greedy* and *RA-Greedy* schemes with the one consumed by *NPM* as a baseline. As *Greedy* scheme does not consider system reliability when reclaiming dynamic slack for energy savings, the normalized energy consumption for *Greedy* scheme only depends on the average system load and is roughly the same for different fault rate changes. For *RA-Greedy* scheme, by providing an additional recovery for maintaining system reliability, it consumes from 10% to 20% more energy than *Greedy* scheme when the fault rate only increases moderately with reduced frequencies and supply voltages (i.e., $d \leq 2$). However, when the fault rate increases dramatically (e.g., $d = 5$), the probability of failure for the original scaled-down execution is close to 1 when the average system load is low (see Figure 8c) and the recovery task is almost always executed, which leads to higher energy consumption than *NPM* (Figure 9c). Therefore, when the fault rate increases dramatically with reduced frequencies and supply voltages, it will be more energy efficient to use less dynamic slack for energy management to keep the fault rate at a reasonable level.

5.2 Effects of Checkpoints

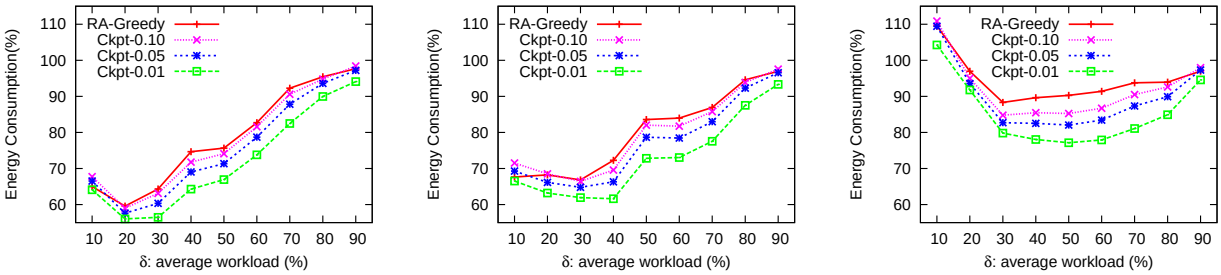
Considering the checkpoint overhead could be very small [26], we use $r = 0.01, 0.05, 0.1$, which corresponds to *Ckpt-0.01*, *Ckpt-0.05* and *Ckpt-0.10* in the following figures, respectively. Recall that the size of tasks is in the range of $[1, 10]$ inclusively, which leads to the average $\gamma = 0.002, 0.01$ and 0.02, smaller than the ones we used in the analysis in Section 4.

Figure 10 shows the probability of failure for the schemes of *RA-Greedy* and *Ckpt* with different checkpoint overheads. From the figure, when the fault rate increase is moderate (i.e., $d \leq 2$), *Ckpt* achieves slightly better system reliability (lower probability of failure) by providing an additional recovery when the amount of available dynamic slack is less than the size of the next ready task. When the checkpoint overhead is smaller, *Ckpt* has more chances to use the dynamic slack and generally gets better system reliability. However, when the fault rate increase is high (e.g., $d = 5$), the additional recovery is almost always executed and overall probability of failure increases due to the execution overhead of checkpoints. Smaller checkpoint overhead leads to higher probability of using checkpoints and thus results in higher probability of failure.



a. $d = 0$ b. $d = 2$ c. $d = 5$

Figure 10: The probability of failure with checkpoints.



a. $d = 0$ b. $d = 2$ c. $d = 5$

Figure 11: The normalized expected energy consumption with checkpoints.

Figure 11 further shows the corresponding normalized energy consumption for *RA-Greedy* and *Ckpt* with different checkpoint overheads. With additional chances for energy management, *Ckpt* with smaller checkpoint overhead consumes less energy, and all of them is less than the one consumed by *RA-Greedy*. The same reason as before, due to the limitation of f_{ee} and higher failure rates, the normalized energy consumption decreases first and then increases as the average system load increases. All schemes consumes more energy than *NPM* when $d = 5$ and $\sigma \leq 10\%$.

6 Closely Related Work

Using the primary/backup recovery model, Unsal *et al.* proposed to postpone the execution of backup tasks to minimize the overlap of primary and backup execution and thus the energy consumption [33]. The optimal number of checkpoints, evenly or unevenly distributed, to achieve minimal energy consumption while tolerating one transient fault was explored by Melhem *et al.* in [21]. Elnozahy *et al.* proposed an *Optimistic TMR* scheme that reduces the energy consumption for traditional TMR systems by allowing one processing unit to slow down provided that it can catch up and finish the computation before the application deadline [8]. The optimal frequency settings for OTMR was further explored in [43]. Assuming a Poisson fault model, Zhang

et al. proposed an adaptive checkpointing scheme that dynamically adjusts checkpoint intervals for energy savings while tolerating a fixed number of faults for a single task [38]. The work is further extended to a set of periodic tasks [40], and moreover, faults within checkpoints are also considered [39].

Most of the previous research either focused on tolerating fixed number of faults [8, 21] or assumed constant fault rate [38, 39, 43] when applying frequency and voltage scaling for energy savings. The work reported in this paper is different from all previous work in that we address the system reliability problem when exploring dynamic slack for energy savings, while explicitly taking the effects of energy management on fault rates into consideration.

7 Conclusions

As fault rates generally increase with reduced supply voltages, energy management exploring slack time through voltage scaling will reduce system reliability, which is undesirable, especially for mission critical applications (e.g., satellite and surveillance systems), where system reliability is as important as (or even more important than) energy consumption. Considering the effects of voltage scaling on fault rates, we propose *reliability-aware* dynamic energy management schemes that preserve system reliability while exploring dynamic slack for energy savings.

By scheduling an additional recovery task before reclaiming dynamic slack for energy management, the proposed reliability-aware energy management scheme ensures that the system reliability achieved is higher than the case when there is no power management. Checkpointing techniques are further explored to more efficiently use the dynamic slack especially for the case where the slack is not enough to schedule a recovery for a whole task. The performance of the proposed schemes is analyzed and evaluated through simulations for both system reliability and energy savings. The results show that, the proposed schemes can achieve comparable energy savings as ordinary energy management schemes while preserving system reliability. For checkpointing techniques, no more than one recovery section may be exploited for energy efficiency, especially when the checkpoint overhead is large. Ignoring the effects of energy management on fault rates is too optimistic and the ordinary energy management schemes could lead to drastically decreased system reliability.

Acknowledgements

The author would like to thank Prof. Rami Melhem and Prof. Daniel Mossé for their inspiring discussions and suggestions in the early stages of this work. Also, the author would like to thank the anonymous reviewers whose comments helped to improve the paper. Moreover, the author would like to thank Dr. Aydin Hakan for his insightful discussions when revising the paper.

References

[1] H. Aydin. On fault-sensitive feasibility analysis of real-time task sets. In *Proc. of The* 25^{th} *IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2004.

[2] H. Aydin, R. Melhem, D. Mossé, and P. Mejia-Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proc. of The* 22^{th} *IEEE Real-Time Systems Symposium*, Dec. 2001.

[3] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. *The case for power management in web servers*, chapter 1. Power Aware Computing. Plenum/Kluwer Publishers, 2002.

[4] T. D. Burd and R. W. Brodersen. Energy efficient cmos microprocessor design. In *Proc. of The HICSS Conference*, Jan. 1995.

[5] X. Castillo, S. McConnel, and D. Siewiorek. Derivation and caliberation of a transient error reliability model. *IEEE Trans. on computers*, 31(7):658–671, 1982.

[6] Intel Corp. Mobile pentium iii processor-m datasheet. Order Number: 298340-002, Oct 2001.

[7] A. Ejlali, M. T. Schmitz, B. M. Al-Hashimi, S. G. Miremadi, and P. Rosinger. Energy efficient seu-tolerance in dvs-enabled real-time systems through information redundancy. In *Proc. of the Int'l Symposium on Low Power and Electronics and Design (ISLPED)*, 2005.

[8] E. (Mootaz) Elnozahy, R. Melhem, and D. Mossé. Energy-efficient duplex and tmr real-time systems. In *Proc. of The* 23^{rd} *IEEE Real-Time Systems Symposium*, Dec. 2002.

[9] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner. Razor: circuit-level correction of timing errors for low-power operation. *IEEE Micro*, 24(6):10–20, 2004.

[10] R. Ernst and W. Ye. Embedded program timing analysis based on path clustering and architecture classification. In *Proc. of The International Conference on Computer-Aided Design*, pages 598–604, Nov. 1997.

[11] X. Fan, C. Ellis, and A. Lebeck. The synergy between power-aware memory systems and processor voltage. In *Proc. of the Workshop on Power-Aware Computing Systems*, 2003.

[12] P. Hazucha and C. Svensson. Impact of cmos technology scaling on the atmospheric neutron soft error rate. *IEEE Trans. on Nuclear Science*, 47(6):2586–2594, 2000.

[13] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proc. of The* 14^{th} *Symposium on Discrete Algorithms*, 2003.

[14] T. Ishihara and H. Yauura. Voltage scheduling problem for dynamically variable voltage processors. In *Proc. of The 1998 International Symposium on Low Power Electronics and Design*, Aug. 1998.

[15] R.K. Iyer and D. J. Rossetti. A measurement-based model for workload dependence of cpu errors. *IEEE Trans. on Computers*, 33:518–528, 1984.

[16] R.K. Iyer, D. J. Rossetti, and M.C. Hsueh. Measurement and modeling of computer reliability as affected by system activity. *ACM Trans. on Computer Systems*, 4(3):214–237, Aug. 1986.

[17] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proc. of the* 41^{st} *annual Design automation conference (DAC)*, 2004.

[18] C. M. Krishma and A. D. Singh. Reliability of checkpointed real-time systems using time redundancy. *IEEE Trans. on Reliability*, 42(3):427–435, 1993.

[19] A. R. Lebeck, X. Fan, H. Zeng, and C. S. Ellis. Power aware page allocation. In *Proc. of the* 9^{th} *International Conference on Architectural Support for Programming Languages and Operating Systems*, Nov. 2000.

[20] H. Lee, H. Shin, and S. Min. Worst case timing requirement of real-time tasks with time redundancy. In *Proc. of Real-Time Computing Systems and Applications*, 1999.

[21] R. Melhem, D. Mossé, and E. (Mootaz) Elnozahy. The interplay of power management and fault recovery in real-time systems. *IEEE Trans. on Computers*, 53(2):217–231, 2004.

[22] D. Mossé, H. Aydin, B. R. Childers, and R. Melhem. Compiler-assisted dynamic power-aware scheduling for real-time applications. In *Proc. of Workshop on Compiler and OS for Low Power*, Oct. 2000.

[23] T. Pering, T. Burd, and R. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In *Proc. of Int'l Symposium on Low Power Electronics and Design*, Aug. 1998.

[24] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proc. of* 18^{th} *ACM Symposium on Operating Systems Principles (SOSP'01)*, Oct. 2001.

[25] D. K. Pradhan. *Fault Tolerance Computing: Theory and Techniques*. Prentice Hall, 1986.

[26] F. Quaglia and A. Santoro. Nonblocking checkpointing for optimistic parallel simulation: Description and an implementation. *IEEE Trans. on Parallel and Distributed Systems*, 14(6):593–610, 2003.

[27] M. W. Rashid, E. J. Tan, M. C. Huang, and D. H Albonesi. Exploiting coarse-grain verification parallelsim for power-efficient fault tolerance. In *Proc. of the* 14^{th} *Int'l Conference on Parallel Architecture and Compilation Techniques (PACT)*, 2005.

[28] S. Saewong and R. Rajkumar. Practical voltage scaling for fixed-priority rt-systems. In *Proc. of the* 9^{th} *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2003.

[29] N. Seifert, D. Moyer, N. Leland, and R. Hokinson. Historical trend in alpha-particle induced soft error rates of the alpha^{TM} microprocessor. In *Proc. of the* 39^{th} *Annual International Reliability Physics Symposium*, 2001.

[30] Tezzaron Semiconductor. Soft errors in electronic memory: A white paper. available at http://www.tachyonsemi.com/about/papers/, 2004.

[31] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Proc. of the International Conference on Dependable Systems and Networks*, 2002.

[32] A. Sinha and A. P. Chandrakasan. Jouletrack - a web based tool for software energy profiling. In *Proc. of Design Automation Conference*, Jun 2001.

[33] O. S. Unsal, I. Koren, and C. M. Krishna. Towards energy-aware software-based fault tolerance in real-time systems. In *Proc. of The International Symposium on Low Power Electronics Design (ISLPED)*, Aug. 2002.

[34] N.J. Wang, J. Quek, T.M. Rafacz, and S.J. Patel. Characterizing the effects of transient faults on a high-performance processor pi peline. In *Proc. of the 2004 International Conference on Dependable Systems and Networks (DS N)*, Jun. 2004.

[35] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced cpu energy. In *Proc. of The First USENIX Symposium on Operating Systems Design and Implementation*, Nov. 1994.

[36] R. Xu, D. Zhu, C. Rusu, R. Melhem, and D. Mossé. Energy efficient policies for embedded clusters. In *Proc. of the Conference on Language, Compilers, and Tools for Embedded Systems (LCTES)*, Jun. 2005.

[37] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Proc. of The* 36^{th} *Annual Symposium on Foundations of Computer Science*, Oct. 1995.

[38] Y. Zhang and K. Chakrabarty. Energy-aware adaptive checkpointing in embedded real-time systems. In *Proc. of IEEE/ACM Design, Automation and Test in Europe Conference(DATE)*, 2003.

[39] Y. Zhang and K. Chakrabarty. Task feasibility analysis and dynamic voltage scaling in fault-tolerant real-time embedded systems. In *Proc. of IEEE/ACM Design, Automation and Test in Europe Conference(DATE)*, 2004.

[40] Y. Zhang, K. Chakrabarty, and V. Swaminathan. Energy-aware fault tolerance in fixed-priority real-time embedded systems. In *Proc. of International Conference on Computer Aided Design*, Nov. 2003.

[41] D. Zhu, R. Melhem, and D. Mossé. The effects of energy management on reliability in real-time embedded systems. In *Proc. of the International Conference on Computer Aidded Design (ICCAD)*, Nov. 2004.

[42] D. Zhu, R. Melhem, and D. Mossé. Energy efficient configuration for qos in reliable parallel servers. In *Proc. of the Fifth European Dependable Computing Conference (EDCC)*, Apr. 2005.

[43] D. Zhu, R. Melhem, D. Mossé, and E.(Mootaz) Elnozahy. Analysis of an energy efficient optimistic tmr scheme. In *Proc. of the* 10^{th} *International Conference on Parallel and Distributed Systems (ICPADS)*, Jul. 2004.

[44] J. F. Ziegler. Terrestrial cosmic ray intensities. *IBM Journal of Research and Development*, 42(1):117–139, 1998.

[45] J. F. Ziegler. Trends in electronic reliability: Effects of terrestrial cosmic rays. available at http://www.srim.org/SER/SERTrends.htm, 2004.