

Analysis of an Energy Efficient Optimistic TMR Scheme*

Dakai Zhu, Rami Melhem, Daniel Mossé
Computer Science Department
University of Pittsburgh
{zdk, melhem, mosse}@cs.pitt.edu

Elmootazbellah (Mootaz) Elnozahy
System Software Department
IBM Austin Research Laboratory
mootaz@us.ibm.com

Abstract

For mission critical real-time applications, such as satellite and surveillance systems, a high level of reliability is desired as well as low energy consumption. In this paper, we propose a general system power model and explore the optimal speed setting to minimize system energy consumption for an Optimistic TMR (OTMR) scheme. The performance of OTMR is compared with that of TMR and Duplex with respect to energy and reliability. The results show that OTMR is always better than TMR by achieving higher levels of reliability and consuming less energy. With checkpoint overhead and recovery, Duplex is not applicable when system load is high. However, Duplex may be more energy efficient than OTMR depending on system static power and checkpointing overhead. Moreover, with one recovery section, Duplex achieves comparable levels of reliability as that of OTMR.

1. Introduction

Energy management through voltage scaling [14], which reduces system supply voltage and processing speed to save energy, has been well studied in the context of real-time systems [13, 18, 23, 24]. Fault tolerance through redundancy has also been extensively explored for high levels of reliability [15, 16]. However, there is relatively less work focusing on the combination of energy and reliability management [5, 12, 22, 25, 26, 27], which is particularly important for mission critical real-time applications, such as satellite and surveillance systems.

Triple Modular Redundancy (TMR) and Duplex with temporal redundancy are common techniques to achieve high levels of reliability [15]. Nevertheless, redundancy also implies more energy consumption. For outer space exploration systems (e.g., Mars Rover), which are generally battery operated, *energy efficient fault tolerance* is desired for longer lifetime. Intuitively, TMR can achieve

higher levels of reliability than Duplex by using one more processing unit, but TMR consumes more energy for operating the additional processing unit.

An Optimistic TMR (OTMR) scheme, which is designed for energy efficiency, reduces system energy consumption for a traditional TMR by slowing down one processing unit, provided that it can catch up and finish the computation before the deadline if the computations on the other two units do encounter an error [5]. However, the maximum energy saving obtained by the OTMR scheme depends on appropriate speed setting for the processing units, which in turn is determined by the system power characteristics. While dynamic power dominates processor power dissipation, static leakage power increases much fast with technology advancements. For example, static leakage power for 1 μ m technology was 0.01% of total processor power, but is approaching 10% for 0.1 μ m technologies [21]. For memory, the leakage power also increases while active power decreases [20]. Thus, considering the whole processing unit, static power will increase substantially with new technologies and should be incorporated in power management schemes.

In this paper, we propose a general system power model and analyze the optimal processing speeds to minimize energy consumption for the OTMR scheme. The system power model has an important effect on energy management schemes. Specifically, an energy efficient speed can be obtained based on system power characteristics, which could be higher than the minimum speed in a system. We show that the performance of OTMR is always better than TMR with respect to both energy and reliability. With checkpoint overhead and recovery, Duplex is not applicable when system load is high. But when system load is low, checkpointing overhead is small and static power is significant, Duplex is more efficient than OTMR in terms of energy consumption. Assuming a Poisson distributed fault model, the reliability achieved by each technique is also analyzed.

Closely Related Work Combined with voltage scaling techniques, the optimal number of checkpoints among a real-time application, uniformly or non-uniformly distributed, to achieve the minimum energy consumption

* This work has been supported by the Defense Advanced Research Projects Agency through the PARTS (Power-Aware Real-Time Systems) project (Contract F33615-00-C-1736).

was explored for Duplex systems in [12]. Assuming a Poisson fault model, Zhang *et al.* proposed an adaptive checkpointing scheme that dynamically adjusts checkpoint intervals to tolerate a fixed number of faults [25]. Elnozahy *et al.* proposed an *Optimistic TMR* scheme to reduce 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’s deadline [5]. For a set of independent periodic tasks, using the primary/backup recovery model, Unsal *et al.* proposed an energy-aware software-based fault tolerance scheme, which postpones as much as possible the execution of backup tasks to minimize the overlap of primary and backup execution and thus to minimize energy consumption [22]. Checkpointing was explored to tolerate a fixed number of faults while minimizing the energy consumption for a task set [27], and the work was further extended in [26] by considering faults within checkpoints.

The paper is organized as follows: the application model, system power model and fault model are discussed in Section 2. We review the energy management for TMR and obtain the optimal speeds for OTMR in Section 3. Section 4 explores the applicability and the energy management for Duplex. Section 5 analyzes the reliabilities. The evaluation is presented and discussed in Section 6. Section 7 concludes the paper.

2. System Models

2.1. Application Model

We consider a frame-based real-time application that is generally characterized by a *worst case execution time* (WCET), L , and a *deadline*, D , which is also the frame size. The execution within each frame is repeated and we will focus on the execution of the application within one frame due to periodicity. Given that variable speed processors are available [7, 8], the WCET of an application depends on the processor speed. Assuming that C is the *worst case number of cycles* for an application, C also depends on the processor speed with memory accesses being considered [19]. However, with a reasonable size cache, C was shown to have very small variations with different speeds [12]. For simplicity, we assume that C is a constant and is the number of cycles needed by an application at the maximum speed f_{max} . Thus, the application’s WCET at f_{max} is $L = \frac{C}{f_{max}}$ and the execution time doubles when the speed is reduced by half¹.

As usual in real-time systems, we assume that $L \leq D$ and define the system load as $\sigma = \frac{L}{D}$. The process-

ing speed is assumed to be able to take any speed² between f_{min} , the minimum processing speed, and f_{max} , the maximum processing speed.

2.2. Power Model

In embedded systems, the power is consumed mainly by the processor, memory, clock and underlying circuits. When there is no computation to execute, we can either put a system into sleep, from which a system can quickly (e.g., in a few cycles) respond to computation requirement, to save energy [5] or turn off the system to get more energy savings. However, turning on/off a system incurs huge time and energy overheads [1]. Thus, we assume that a system has three different states: *off*, *sleep* and *active* (defined as having computation in progress). No power is consumed when a system is off. The system power is correspondingly divided into two categories: *sleep power*, which is a constant and is consumed while a system is in sleep and active states, and *active power*, which is only consumed while a system is active.

The sleep power includes (but is not limited to) the power to maintain basic circuits, keep the clock running and the memory in sleep mode [11]. The active power is further divided into two parts: *speed-independent active power* and *speed-dependent active power*. Speed-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 and is independent of system supply voltages and processing speeds [4, 11]. speed-dependent active power includes processor’s dynamic power and any power that depends on system supply voltages and processing speeds [2, 20]. Thus, the system power can be modeled as:

$$P = P_s + \hbar(P_{ind} + P_d) \quad (1)$$

$$P_d = C_{ef} f^m \quad (2)$$

where P_s is the sleep power, P_{ind} is the speed-independent active power and P_d is the speed-dependent active power. \hbar equals 0 if the system is in sleep and \hbar equals 1 if the system is active. C_{ef} and m (> 2) are system dependent constants and f is the processing speed (in number of cycles per time unit) [2].

When the performance requirement is not the maximum, we can use *voltage scaling*, which reduces system supply voltage for lower speeds³, to further manage the speed-dependent active power [14]. The maximum speed-dependent active power corresponds to the maximum speed f_{max} and is denoted by $P_d^{max} = C_{ef} f_{max}^m$. For convenience, the values of P_s and P_{ind} are assumed to be αP_d^{max} and βP_d^{max} , respectively.

1 Notice that, this is a conservative model. With memory effects, C actually decreases with reduced speeds and the execution time will be less than the modeled time.

2 For the case of discrete processing speeds, we can use two adjacent speeds to emulate any speed not supported by the system [10].

3 In the rest of this paper, we use speed changes to stand for changing both system supply voltage and processing speed.

The Effect of Power Model on Voltage Scaling Intuitively, lower processing speeds result in less speed-dependent active energy consumption. But the application will run longer and thus consume more speed-independent active energy. Therefore, there is an energy efficient speed f_{ee} to minimize the energy consumption [5, 6, 9, 18]. Here, we estimate f_{ee} using our general power model.

Consider an application running at speed f for $L \frac{f_{max}}{f}$ time units, the system energy consumption to execute the application is (the system is put into sleep right after it finishes executing the application):

$$E = P_s D + (P_{ind} + C_{ef} f^m) L \frac{f_{max}}{f} \quad (3)$$

where D is the application's deadline. Notice that the system will be put to sleep for $(D - L \frac{f_{max}}{f})$ time units. Due to the high time and energy overhead of turning on/off a system [1], we assume the system is always on and the sleep power P_s is always consumed.

By differentiating Equation (3) with respect to f , we find that E is minimized when $f = \sqrt[m]{\frac{\beta}{m-1}} f_{max}$, which is defined as the *energy-efficient* speed f_{ee} . Given that f_{min} is the minimum supported processing speed, we define the minimum energy efficient speed as $f_{sys} = \max\{f_{min}, f_{ee}\}$ and $\kappa = \frac{f_{sys}}{f_{max}}$. That is, we may be forced to run at a speed higher than f_{ee} to meet the application's deadline or to comply with the lowest speed limitation, but we should never run at a speed below f_{ee} , since doing so consumes more energy. Thus, **when the system power has a speed-independent component that can be efficiently removed, its effect on voltage scaling is equivalent to imposing an energy efficient speed.**

Notice that an application cannot run faster than the maximum speed f_{max} . If $f_{ee} > f_{max}$, that is, $\beta > m-1$, all applications would run at f_{max} to minimize the energy consumption and no voltage scaling is required. In this paper, we assume that $\beta \leq m-1$.

2.3. Fault Model

During the execution of an application, a fault may occur due to various reasons, such as hardware failures, software errors and the effect of cosmic ray radiations. Since *transient* and *intermittent* faults occur much more frequently than *permanent* faults [3], in this paper, we focus on transient and intermittent faults, which can be recovered through re-execution.

The interarrival time of faults is assumed to follow a Poisson distribution with an average arrival rate of λ . In general, λ varies with different system supply voltages and processing speeds. However, to the best of our knowledge, there is no existing model in the literature that addresses this problem. For simplicity, we assume that λ is a constant. Thus, the probability of having

fault(s) during an application execution time t is:

$$\rho(t) = 1 - e^{-\lambda t} \quad (4)$$

That is, the reliability (i.e., the probability of having no fault) for an application will depend on its execution time. The longer an application runs, the higher the probability of having fault(s) and the lower the reliability is.

3. TMR and Optimistic TMR

A TMR system tolerates one fault by running an application on three identical processing units simultaneously and voting on the three outputs [15]. Expecting that faults are rare, an Optimistic TMR (OTMR) scheme has been proposed to reduce the energy consumption in a TMR system. The idea is to turn off or slow down one processing unit, provided that it can catch up and finish the computation before the deadline if the other two units do encounter an error [5]. Below, we first briefly review the energy management for TMR using our power model, and then derive the optimal speed setting to minimize the system energy consumption for OTMR.

3.1. Energy Management for TMR

With load $\sigma = \frac{L}{D} \leq 1$, we can execute the application with reduced speed (i.e., as low as σf_{max}) and save energy while ensuring that the application's deadline is met. With the minimal energy efficient speed being f_{sys} (see Section 2), an application would run at the speed of $f_{TMR} = \max\{\sigma f_{max}, f_{sys}\}$ to minimize the energy consumption. Thus, the energy consumption for TMR is:

$$E_{TMR} = 3 \left(P_s D + (P_{ind} + C_{ef} f_{TMR}^m) L \frac{f_{max}}{f_{TMR}} \right) \quad (5)$$

Recall that $f_{sys} = \kappa f_{max}$. For low system loads, that is, when $0 < \sigma \leq \kappa$, we have $f_{TMR} = f_{sys}$ and $E_{TMR} = 3 \left(\alpha + (\beta + \kappa^m) \frac{\sigma}{\kappa} \right) P_d^{max} D$. However, when $\kappa < \sigma \leq 1$, we have $f_{TMR} = \sigma f_{max}$ and $E_{TMR} = 3(\alpha + \beta + \sigma^m) P_d^{max} D$.

3.2. Optimal Speeds for OTMR

With the expectation that no fault will occur in the first two units of a TMR system, the third unit could sleep or run at a lower speed as long as we can ensure that it has enough reserved time to finish the execution before D . The speed for the first two units is *crucial* in determining *when* the third unit should begin to run and at *what* speed. In what follows, we explore the optimal speed setting for OTMR to minimize system energy consumption.

Suppose that the optimal speed for the first two units is $f_2 = x f_{max}$. We have $f_2 \geq \max\{\sigma f_{max}, f_{sys}\}$, that is, $x \geq \max\{\sigma, \kappa\}$. The reserved time for the third unit is $D - \frac{L}{x}$. If $D - \frac{L}{x} \geq L$ (i.e., the reserved time is enough for the third unit to finish the computation at f_{max}), no work needs to be done concurrently and the third unit

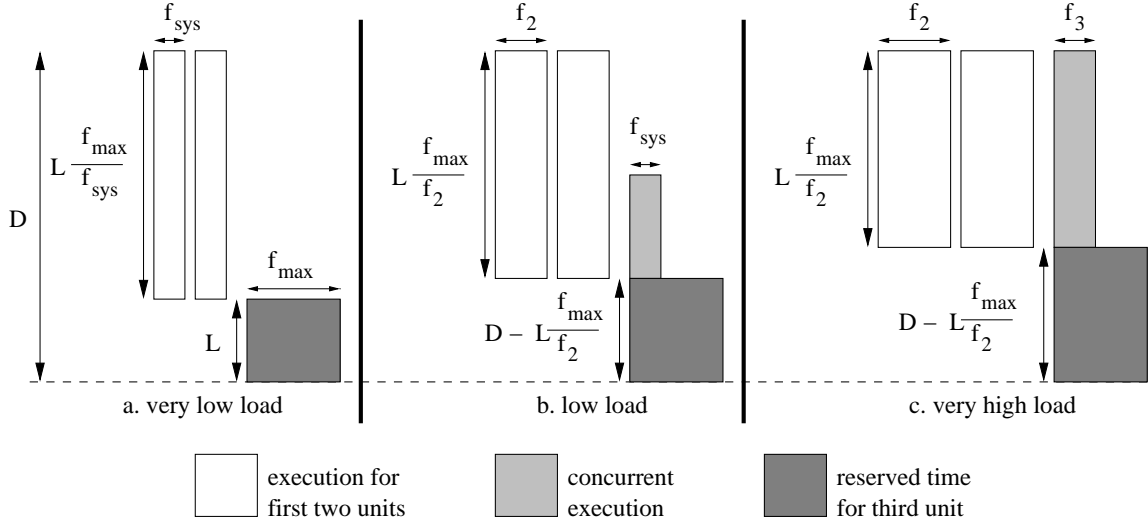


Figure 1. Optimal speeds for OTMR under different loads: a. very low load ($0 \leq \sigma \leq \frac{\kappa}{1+\kappa}$); b. low load ($\frac{\kappa}{1+\kappa} < \sigma \leq \kappa$); c. very high load ($\frac{1}{2-\kappa} < \sigma \leq 1$).

just sleeps initially (see following Case 1). Otherwise, the part of the application that needs to be executed concurrently with the first two units is $L - (D - \frac{L}{x})$ with speed $f_3 = \max\{f_{sys}, \frac{L - (D - \frac{L}{x})}{L} f_{max}\} = \max\{f_{sys}, (1 - \frac{1-\sigma}{\sigma}x)f_{max}\}$. In order to find the optimal speeds, we need to consider the following cases regarding to different system loads.

Case 1: $0 < \sigma \leq \frac{\kappa}{1+\kappa}$ (system load is very low).

In this case, we have $L \leq D - \frac{L}{\kappa}$ (from $\sigma \leq \frac{\kappa}{1+\kappa}$), that is, there is enough reserved time and the third unit just sleeps initially. The optimal speed for the first two units is f_{sys} as shown in Figure 1a. In the figure, the width of a rectangle represents processing speed, and the height represents execution time. The rectangle area represents the number of cycles needed for the application. Since faults are rare, we consider only the energy consumption during *fault free* execution in the following analysis. The minimum fault free energy consumption for OTMR is:

$$\begin{aligned} E_{OTMR} &= 3P_s D + 2(P_{ind} + C_{ef} f_{sys}^m) L \frac{f_{max}}{f_{sys}} \\ &= \left(3\alpha + 2(\beta + \kappa^m) \frac{\sigma}{\kappa}\right) P_d^{max} D \end{aligned}$$

Case 2: $\frac{\kappa}{1+\kappa} < \sigma \leq \kappa$ (system load is low).

In this case, the third unit may have to execute part of the application concurrently with the first two units as shown in Figure 1b. However, the third unit does not need to run at a speed higher than f_{sys} initially to meet the timing constraints, as discussed below.

Notice that, when the reserved time for the third unit is L , the first two units should run at speed $\frac{L}{D-L} f_{max} = \frac{\sigma}{1-\sigma} f_{max}$, which is higher than f_{sys} (notice that $\frac{\kappa}{1+\kappa} < \sigma$ and $f_{sys} = \kappa f_{max}$). Running at a speed higher than

$\frac{\sigma}{1-\sigma} f_{max}$ will needlessly increase the energy consumption for the first two units. Since the optimal speed for the first two units $f_2 = x f_{max}$ is also limited by f_{max} , we have $f_2 \leq \min\{\frac{\sigma}{1-\sigma} f_{max}, f_{max}\}$, that is, $x \leq \min\{\frac{\sigma}{1-\sigma}, 1\}$. Thus, we have $L - (D - \frac{L}{x}) \geq 0$. Recall that the optimal speed for the third unit to execute the overlapping part of the application is $f_3 = \max\{f_{sys}, (1 - \frac{1-\sigma}{\sigma}x)f_{max}\}$. Since $f_2 \geq f_{sys}$, that is, $x \geq \kappa \geq \sigma$, we have $(1 - \frac{1-\sigma}{\sigma}x)f_{max} \leq (1 - \frac{1-\sigma}{\sigma}\sigma)f_{max} = \sigma f_{max} \leq f_{sys}$. Thus, $f_3 = f_{sys}$. However, the start time for the third unit is determined by x and σ . Therefore, the energy consumption for OTMR is:

$$\begin{aligned} E_{OTMR} &= 3P_s D + 2(P_{ind} + C_{ef} f_2^m) \frac{L}{x} \\ &\quad + (P_{ind} + C_{ef} f_{sys}^m) \frac{L - (D - \frac{L}{x})}{\kappa} \\ &= (3\alpha + 2(\beta + x^m) \frac{\sigma}{x}) \\ &\quad + (\beta + \kappa^m) \frac{\sigma - (1 - \frac{\sigma}{x})}{\kappa} P_d^{max} D \end{aligned}$$

Differentiating the above equation with respect to x and setting $\frac{\partial E_{OTMR}}{\partial x} = \frac{2(m-1)x^{m-2}\beta - \frac{\beta}{x^2} - \kappa^{m-1}}{x^2} \sigma = 0$, we conclude that E_{OTMR} is minimized when

$$x = \sqrt[m]{\frac{2\beta + \frac{\beta}{\kappa} + \kappa^{m-1}}{2(m-1)}} \stackrel{def}{=} x_{\beta, \kappa, m}$$

subject to $\kappa \leq x \leq \min\{\frac{\sigma}{1-\sigma}, 1\}$. Thus, if $x_{\beta, \kappa, m} \leq \kappa$, E_{OTMR} is minimized when $x = \kappa$; otherwise, E_{OTMR} is minimized when $x = \min\{1, \frac{\sigma}{1-\sigma}, x_{\beta, \kappa, m}\}$.

Case 3: $\kappa < \sigma \leq \frac{1}{2-\kappa}$ (system load is high).

Noting that $f_2 = x f_{max}$ and $f_3 = \max\{f_{sys}, (1 - \frac{1-\sigma}{\sigma}x)f_{max}\}$. The speed $(1 - \frac{1-\sigma}{\sigma}x)f_{max}$ is smaller than

f_{sys} if $x \geq \frac{1-\kappa}{1-\sigma}$. In this case, $f_3 = f_{sys}$ (same as in Figure 1b) and the optimal x to minimize E_{OTMR} can be solved as in Case 2. However, if $x < \frac{1-\kappa}{1-\sigma}$, we have $(1 - \frac{1-\sigma}{\sigma}x)f_{max} > f_{sys}$ and the third unit will run at speed $f_3 = (1 - \frac{1-\sigma}{\sigma}x)f_{max}$ (as in Figure 1c). The optimal x to minimize E_{OTMR} can be found iteratively as will be shown for Case 4 discussed next. As the result, the optimal x is the one that results in smaller E_{OTMR} among the two sub cases.

Case 4: $\frac{1}{2-\kappa} < \sigma \leq 1$ (the system load is very high).

From $\frac{1}{2-\kappa} < \sigma$, we have $1 < \frac{1-\kappa}{1-\sigma}$. Note that $x \leq \min\{\frac{\sigma}{1-\sigma}, 1\}$ as discussed in Case 2. Therefore, $x \leq 1 < \frac{1-\kappa}{1-\sigma}$, that is, $\kappa < 1 - \frac{1-\sigma}{\sigma}x$. Hence the third unit needs to execute part of the application concurrently with the first two units at speed $f_3 = yf_{max} = (1 - \frac{1-\sigma}{\sigma}x)f_{max} > f_{sys}$ as shown in Figure 1c. Thus, the energy consumption for OTMR is:

$$\begin{aligned} E_{OTMR} &= 3P_s D + 2(P_{ind} + C_{ef}f_2^m) \frac{L}{x} \\ &\quad + (P_{ind} + C_{ef}f_3^m) \frac{L - (D - \frac{L}{x})}{y} \\ &= (3\alpha + 2(\beta + x^m) \frac{\sigma}{x} \\ &\quad + (\beta + y^m) \frac{\sigma - (1 - \frac{\sigma}{x})}{y}) P_d^{max} D \end{aligned}$$

Setting $\frac{\partial E_{OTMR}}{\partial x} = 0$, we conclude that E_{OTMR} is minimized when x satisfies the following equation subject to $\sigma < x \leq \min\{\frac{\sigma}{1-\sigma}, 1\}$:

$$\begin{aligned} &2(m-1)\sigma x^m y^2 - (1-\sigma)(m-1)x^2 y^m + \\ &(m-1) \frac{1-\sigma}{\sigma} (x-\sigma) x y^m + \beta(1-\sigma)x^2 - \beta\sigma y \\ &- 2\beta\sigma y^2 - \frac{(1-\sigma)\beta}{\sigma} (x-\sigma)x - \sigma y^{m+1} = 0 \end{aligned}$$

where $y = 1 - \frac{1-\sigma}{\sigma}x$. It is not clear if there is a close form for the solution x . For a given m, β and σ , however, x can be found iteratively.

In summary, to minimize the energy consumption of OTMR, when $\sigma \leq \frac{\kappa}{1+\kappa}$, the optimal speed for the first two units is f_{sys} and the third units just sleeps. When $\frac{\kappa}{1+\kappa} < \sigma \leq \kappa$, the optimal speed for the first two units is f_{sys} (if $x_{\beta, \kappa, m} \leq \kappa$) or $\min\{1, \frac{\sigma}{1-\sigma}, x_{\beta, \kappa, m}\} f_{max}$, while the third unit runs at f_{sys} , where $x_{\beta, \kappa, m} = \sqrt[m]{\frac{2\beta + \frac{\beta}{\kappa} + \kappa^{m-1}}{2(m-1)}}$. When the system load is high, that is, $\kappa < \sigma \leq 1$, the optimal speeds need to be solved iteratively.

4. Duplex Systems

Duplex can detect fault(s) and recover through re-execution. Checkpointing, as an efficient technique to explore temporal redundancy and achieve high reliability, rolls back the execution to the latest correct state when

there is a fault [15]. In this paper, we first analyze the applicability of Duplex under different checkpoint overheads, then the optimal number of checkpoints to minimize energy consumption is explored using our general system power model. For simplicity, only uniformly distributed checkpoints are considered [12].

Suppose the overhead of taking one checkpoint is r and n is the number of checkpoints taken. The more checkpoints a Duplex has, the smaller a recovery section is (i.e., $\frac{L}{n}$), but the more checkpoint overhead incurred (i.e., nr). If the normalized checkpoint overhead is $\gamma = \frac{r}{L}$, then $r = \gamma L = \gamma \sigma D$.

4.1. Applicability Analysis

Typically, a Duplex system employs only one recovery section because of the usually small failure rates. If there are n checkpoints, the recovery section is $\frac{L}{n}$. Assuming that the *recovery overhead*, which is the time for a Duplex to restore its previous correct state, is also r (the same as a checkpoint overhead), the following condition should be satisfied:

$$L + nr + r + \frac{L}{n} \leq D$$

which, using $L = \sigma D$ and $r = \gamma \sigma D$, can be rewritten as

$$\gamma \sigma n^2 - (1 - \sigma - \gamma \sigma)n + \sigma \leq 0 \quad (6)$$

Solving Equation (6) for n , we get:

$$\begin{aligned} &\frac{(1 - \sigma - \gamma \sigma) - \sqrt{(1 - \sigma - \gamma \sigma)^2 - 4\gamma \sigma^2}}{2\gamma \sigma} \leq \\ n &\leq \frac{(1 - \sigma - \gamma \sigma) + \sqrt{(1 - \sigma - \gamma \sigma)^2 - 4\gamma \sigma^2}}{2\gamma \sigma} \end{aligned}$$

Let σ_γ be the upper bound for the system load that Duplex can handle for a given checkpointing overhead γ . In order for n to have a real (non-imaginary) solution, we should have $\sigma \leq \frac{1}{1+\gamma+2\sqrt{\gamma}} \stackrel{def}{=} \sigma_\gamma$. In other words, Duplex is not applicable if $\sigma > \sigma_\gamma$. In this section, we assume that $\sigma \leq \sigma_\gamma$.

4.2. Energy Management

With n checkpoints, Duplex could run the application and checkpoints at speed $f_D = \frac{L+nr}{D-r-\frac{L}{n}} f_{max} = \frac{\sigma+n\gamma\sigma}{1-\gamma\sigma-\frac{\sigma}{n}} f_{max}$ with enough time being reserved for one recovery section. From Section 2, to minimize the energy consumption, Duplex will execute the application at speed $f_{Dup} = \max\{f_D, f_{sys}\}$ and the fault free energy consumption is:

$$E_{Dup} = 2 \left(P_s D + (P_{ind} + C_{ef}f_{Dup}^m) (L + nr) \frac{f_{max}}{f_{Dup}} \right)$$

When $0 < f_D \leq f_{sys}$, that is, $0 < \frac{\sigma+n\gamma\sigma}{1-\gamma\sigma-\frac{\sigma}{n}} \leq \kappa$, Duplex executes the application and checkpoints at speed

f_{sys} and the energy consumption is:

$$E_{Dup} = 2 \left(\alpha + (\beta + \kappa^m) \frac{\sigma + n\gamma\sigma}{\kappa} \right) P_d^{max} D$$

Noting that $\frac{\partial E_{Dup}}{\partial n} > 0$, we conclude that E_{Dup} is minimized at the smallest n that satisfies $0 < \frac{\sigma + n\gamma\sigma}{1 - \gamma\sigma - \frac{\sigma}{n}} \leq \kappa$. For any $n (\geq 1)$, we have $0 < \frac{\sigma + n\gamma\sigma}{1 - \gamma\sigma - \frac{\sigma}{n}}$. From $\frac{\sigma + n\gamma\sigma}{1 - \gamma\sigma - \frac{\sigma}{n}} \leq \kappa$, we can get a quadratic equation in n :

$$\gamma\sigma n^2 - (\kappa - \sigma - \kappa\gamma\sigma)n + \kappa\sigma \leq 0$$

Solving this equation, we get:

$$n \leq \frac{(\kappa - \sigma - \kappa\gamma\sigma) + \sqrt{(\kappa - \sigma - \kappa\gamma\sigma)^2 - 4\gamma\sigma^2\kappa}}{2\gamma\sigma}$$

From the above equation, to have a real (non-imaginary) n , we must have $\sigma \leq \frac{\kappa}{1 + \kappa\gamma + 2\sqrt{\kappa\gamma}} \stackrel{def}{=} \sigma_{\kappa,\gamma}$. In other words, in order to run the application and checkpoints at f_{sys} , the system load should be smaller than $\sigma_{\kappa,\gamma}$. Thus, the optimal number of checkpoints to minimize E_{Dup} in this case is:

$$n = \max \left\{ 1, \frac{(\kappa - \sigma - \kappa\gamma\sigma) - \sqrt{(\kappa - \sigma - \kappa\gamma\sigma)^2 - 4\gamma\sigma^2\kappa}}{2\gamma\sigma} \right\} \quad (7)$$

When system load is higher ($\sigma_{\kappa,\gamma} < \sigma \leq \sigma_\gamma$), we will have $\kappa < \frac{\sigma + n\gamma\sigma}{1 - \gamma\sigma - \frac{\sigma}{n}}$ and therefore $f_{sys} < f_D$. The application and checkpoints need to run at f_D and the energy consumption is:

$$E_{Dup} = 2 \left[\alpha + \left(\beta + \left(\frac{\sigma + n\gamma\sigma}{1 - \gamma\sigma - \frac{\sigma}{n}} \right)^m \right) (1 - \gamma\sigma - \frac{\sigma}{n}) \right] P_d^{max} D$$

Notice that any n will satisfy $\kappa < \frac{\sigma + n\gamma\sigma}{1 - \gamma\sigma - \frac{\sigma}{n}}$ when $\sigma_{\kappa,\gamma} < \sigma \leq \sigma_\gamma$. Set $\frac{\partial E_{Dup}}{\partial n} = 0$, we find that E_{Dup} is minimized when n satisfies the following equation:

$$mn^2\gamma(1 - \gamma\sigma - \frac{\sigma}{n})(\sigma + n\gamma\sigma)^{m-1} + \beta(1 - \gamma\sigma - \frac{\sigma}{n})^m - (m-1)(\sigma + n\gamma\sigma)^m = 0$$

We could not find a close form for the solution n . However, for given m, β, γ and σ , the value of n that satisfies the above equation can be found iteratively.

In summary, when $0 \leq \sigma \leq \frac{\kappa}{1 + \kappa\gamma + 2\sqrt{\kappa\gamma}} \stackrel{def}{=} \sigma_{\kappa,\gamma}$, the optimal number of checkpoints n to minimize E_{Dup} is given by Equation (7); when $\sigma_{\kappa,\gamma} < \sigma \leq \frac{1}{1 + \gamma + 2\sqrt{\gamma}} \stackrel{def}{=} \sigma_\gamma$, the optimal number of checkpoints n to minimize E_{Dup} can be solved iteratively. If $\sigma > \sigma_\gamma$, Duplex cannot be used.

5. Reliability Analysis

From previous analysis, the optimal speed to minimize energy consumption for a TMR system is $f_{TMR} =$

$\max\{f_{sys}, \sigma f_{max}\}$, that is, the application will run for time $t_{TMR} = \min\{\frac{L}{\kappa}, D\}$. Given a Poisson distributed fault model with the average arrival rate of λ , the probability of having fault(s) on one processing unit would be $\rho_{TMR} = \rho(t_{TMR}) = 1 - e^{-\lambda t_{TMR}}$. Thus, the reliability of a TMR system is:

$$R_{TMR} = (1 - \rho_{TMR})^3 + 3(1 - \rho_{TMR})^2 \rho_{TMR}$$

where the first term is the probability of having no faults during execution and the second term is the probability of having fault(s) only in one processing unit.

For the OTMR scheme, suppose the optimal speed for the first two units to minimize energy consumption is $f_2 = x f_{max}$, the execution will take time $t_2 = \frac{L}{x}$ and the probability of having fault(s) during the execution on one processing unit is $\rho_2 = \rho(t_2) = 1 - e^{-\lambda \frac{L}{x}}$. When the computation on the first two units encounters an error, the third unit will need to finish the execution. Notice that, the reserved time for the third unit is $D - \frac{L}{x}$. If $D - \frac{L}{x} \geq L$, no work needs to be done concurrently and the third unit will run for time $t_3 = \min\{D - \frac{L}{x}, \frac{L}{\kappa}\}$. Otherwise, the amount of work that needs to be done concurrently with the first two units is $L - (D - \frac{L}{x})$. The optimal speed for the concurrent execution is $f_3 = y f_{max} = \max\{f_{sys}, \frac{L - (D - \frac{L}{x})}{\frac{L}{\kappa}} f_{max}\}$. Thus, the total execution time for the third unit would be $t_3 = \frac{L - (D - \frac{L}{x})}{y} + (D - \frac{L}{x})$ and the probability of having fault(s) on the third unit is $\rho_3 = \rho(t_3) = 1 - e^{-\lambda t_3}$. Therefore, the reliability of a OTMR system is:

$$R_{OTMR} = (1 - \rho_2)^2 + 2(1 - \rho_2)\rho_2(1 - \rho_3)$$

where the first term is the probability of having no fault in the first two units and the second term is the probability of having fault(s) on any one of the first two units but no fault on the third unit.

For Duplex, we assume that one processing unit in a duplex system will fail if there is a fault during the execution of application sections or checkpoints⁴ and the recovery section will be executed on both units. Suppose the optimal number of checkpoints to minimize energy consumption is n , the speed to run the application and checkpoints is $f_{Dup} = \max\{f_{sys}, \frac{\sigma + n\gamma\sigma}{1 - \gamma\sigma - \frac{\sigma}{n}} f_{max}\}$. Thus, one section (including one checkpoint and one section of the application) needs time $t_d = (r + \frac{L}{n}) \frac{f_{max}}{f_{Dup}}$ and the probability of having fault(s) on one processing unit during the execution of one section is $\rho_d = \rho(t_d) = 1 - e^{-\lambda t_d}$, while the probability of one section being correctly executed is $R_d = (1 - \rho_d)^2$. Notice that the recovery section executes at speed f_{max} . The

4 A Duplex might not detect a failure if a failure happens during a checkpoint and the section following the faulty checkpoint does not fail. But, for simplicity, we do a pessimistic analysis and assume that a duplex will fail if one checkpoint fails.

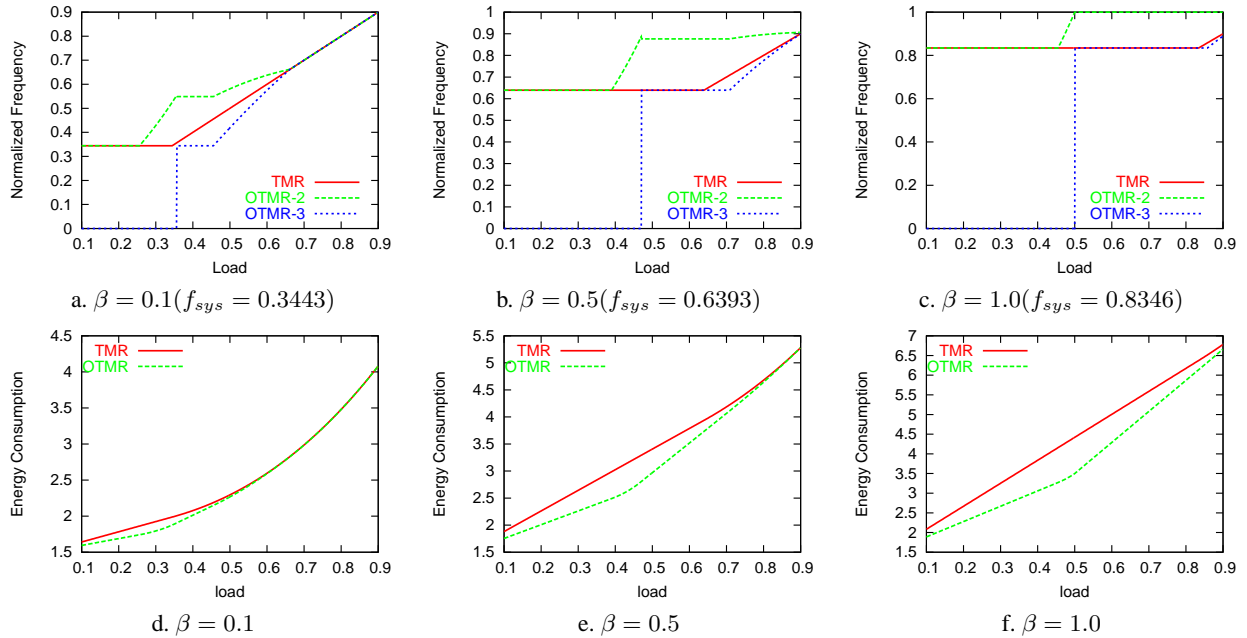


Figure 2. Optimal Speeds and Energy Consumption for OTMR and TMR; $\alpha = 0.5$ and $P_d^{max} D = 1$.

probability of having fault(s) on one processing unit during a recovery section (including recovery overhead) is $\rho_r = \rho(r + \frac{L}{n}) = 1 - e^{-\lambda(r + \frac{L}{n})}$ and the probability of the recovery section correctly executed is $R_r = (1 - \rho_r)^2$. Hence, the reliability of a duplex system is:

$$R_{Dup} = R_d^n + nR_d^{n-1}(1 - R_d)R_r$$

where the first term is the probability of having no fault during all sections and the second term is the probability that one section fails while the recovery section executes correctly.

6. Comparison and Discussion

In this section, we will compare the energy consumption and reliability for TMR, OTMR and Duplex. First, let us determine the system parameters, that is, the values that we should use for α , β and m in the analysis. Recall that α is for sleep power, β is for speed-independent active power and m is the exponent for speed-dependent active power (see Section 2).

For a P-III 600MHz system, the processor consumes the peak power of 27W, memory consumes 5W, and the total system power is around 47W [1]. Considering that processor and memory power can be reduced by up to 98% of their active power when hibernating [4, 11], in our analysis, we will use $\alpha = 0.1, 0.5, 1.0$ and $\beta = 0.1, 0.5, 1.0$. Generally, m is between 2 and 3 for voltage scaling processors [2]; our recent analysis shows that $m \approx 2.6$ for Intel XScale model [7]. Considering other voltage related power, such as some component of the leakage power [20], we expect that m will still

fall between 2 and 3. In this paper, we use $m = 2.6$. For duplex systems, the checkpoint overhead could be very small [17] and we will use $\gamma = 0.01, 0.05, 0.1$, respectively.

For simplicity, we use normalized processing speed with $f_{max} = 1$, and assume that $f_{sys} = \sqrt[m]{\frac{\beta}{m-1}} f_{max}$ (i.e., $f_{min} \leq f_{ee}$, see Section 2).

6.1. Comparison between OTMR and TMR

First, we compare OTMR with TMR on energy consumption. As we discussed earlier, the processing speed for the first two processing units in OTMR systems is critical in energy savings. Figure 2a, 2b and 2c show the optimal processing speeds for the first two units in OTMR (OTMR-2), the third unit of OTMR (OTMR-3) and all units in TMR (TMR), to minimize energy consumption under different system loads.

Notice that, the minimum energy efficient speed $f_{sys} = \sqrt[m]{\frac{\beta}{m-1}} f_{max}$ is determined by β and m . With fixed $m = 2.6$, the larger β leads to larger f_{sys} . From Figure 2a, 2b and 2c, we can see that the optimal speed for TMR is $\max\{f_{sys}, \sigma f_{max}\}$. For OTMR, the optimal speed for the first two units is the same as that for TMR when system load is low ($\sigma \leq \frac{\kappa}{1+\kappa}$), and begins to increase sharply when system load becomes higher. By running the first two units faster, OTMR reserves enough time and the third unit could sleep at the beginning and thus save more energy. For example, when $\beta = 0.5$ (Figure 2b), the optimal speed for the first two units is $f_{sys} = \kappa f_{max} = 0.6393$ and the third

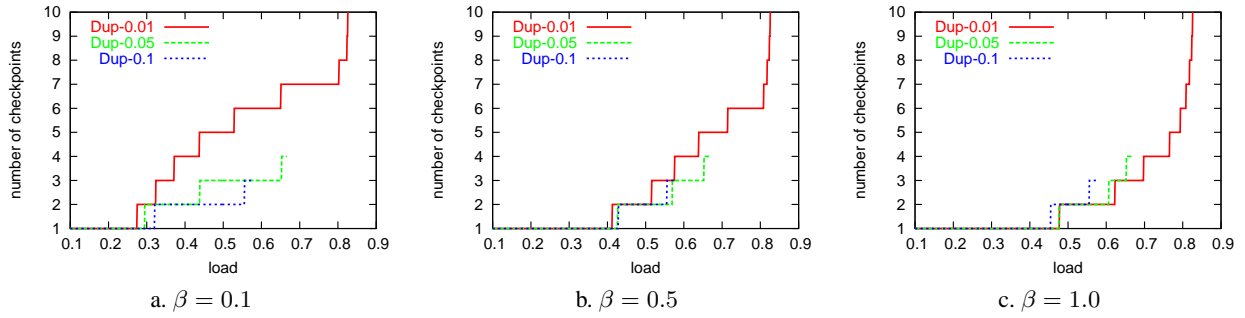


Figure 3. Optimal number of checkpoints for Duplex; $m = 2.6$.

unit sleeps when load $\sigma \leq \frac{\kappa}{1+\kappa} = 0.39$; when load is slightly higher, the optimal speed for the first two units increases to $\frac{\sigma}{1-\sigma}$ (see Section 3) and the third unit continues to sleep until the load reaches $\sigma = 0.4671$, at which point $\frac{\sigma}{1-\sigma} = 0.8764 = x_{\beta,\kappa,m} = \sqrt[m]{\frac{2\beta+\beta/\kappa+\kappa^{m-1}}{2(m-1)}}$; after that the first two units run at the optimal speed $x_{\beta,\kappa,m} = 0.8764$ and the third unit begin to run at $f_{sys} = 0.6393$ (however, the running time for the third unit is not the same as the first two units before load approaches $\frac{1}{2-\kappa} = 0.7349$, after which the third unit's speed is higher than f_{sys}).

Figure 2d, 2e and 2f show the minimum energy consumptions for TMR and OTMR with different system loads when we set $P_d^{max}D = 1$ to normalize the energy units. Notice that the sleep power (indicated by α) is the same for TMR and OTMR and we assume $\alpha = 0.5$. Even though OTMR's first two units run typically faster than TMR, the sleep of the third unit causes OTMR to consume equal or less energy than TMR, at the expense of a more complex speed management scheme.

6.2. Comparison between OTMR and Duplex

The optimal number of checkpoints for Duplex to minimize energy consumption is determined by m , β , γ and σ . Figure 3 shows the optimal number of checkpoints for a duplex system with different speed-independent active power and different checkpoint overhead ($\gamma = 0.01, 0.05, 0.1$ corresponding to Dup-0.01, Dup-0.05 and Dup-0.1, respectively) under different system loads. From the figure, we can see that Duplex is only applicable when the system load is low and/or the overhead of checkpoints is small. With checkpoint overhead increasing, the maximum system load a Duplex can handle decreases. As expected, the optimal number of checkpoints increases when the checkpoint overhead decreases since more checkpoints can be used with smaller checkpoint overhead. The optimal number of checkpoints decreases when speed-independent active power (β) increases. The reason is that the minimum energy efficient speed f_{sys} is higher with high speed-independent ac-

tive power, thus, the application is able to run at f_{sys} with fewer number of checkpoints for the same system load.

Figure 4 shows the optimal energy consumption for Duplex with different checkpoint overheads as well as OTMR when speed-independent active power and static leakage power have different values. When checkpoint overhead is very small (i.e., $\gamma = 0.01$), Duplex outperforms OTMR with less energy consumption for low system load by using one less unit, especially when static leakage power and/or speed-independent active power is significant. When static leakage power is very small (e.g., $\alpha = 0.1$) and checkpoint overhead is big (i.e., $\gamma = 0.1$), OTMR consumes less energy than Duplex with moderate system loads. As expected, OTMR is much worse than Duplex for larger static leakage power (e.g., $\alpha = 1.0$). However, only OTMR is applicable when system load approaches 1.

6.3. Reliability Evaluation

We examine the reliability achieved by each scheme when their processing speeds are optimal for energy consumption. With the assumption that the interarrival time of faults follows Poisson distribution and the average failure rate is λ , the probability of failure on one processing unit during the period of D is $\rho(D) = 1 - e^{-\lambda D}$. Since the deadline D is an application specific parameter, in the following discussion we assume that $\rho(D) = 10^{-3}, 10^{-4}$ and 10^{-5} .

Figure 5 shows the probability of failure for all schemes with different values of $\rho(D)$ (i.e., different failure rates). Lower probability of failure means higher reliability. As expected, when the load σ increases, the reliabilities for all schemes decrease since all schemes use more time to execute the application. The reason is that, with the interarrival time of faults following a Poisson distribution, the longer a processing unit runs, the higher the probability it fails and the lower the reliability is. OTMR achieves slightly better reliability than TMR since OTMR runs faster and uses less time to execute an application. Also note that different checkpoint overheads have no significant effect

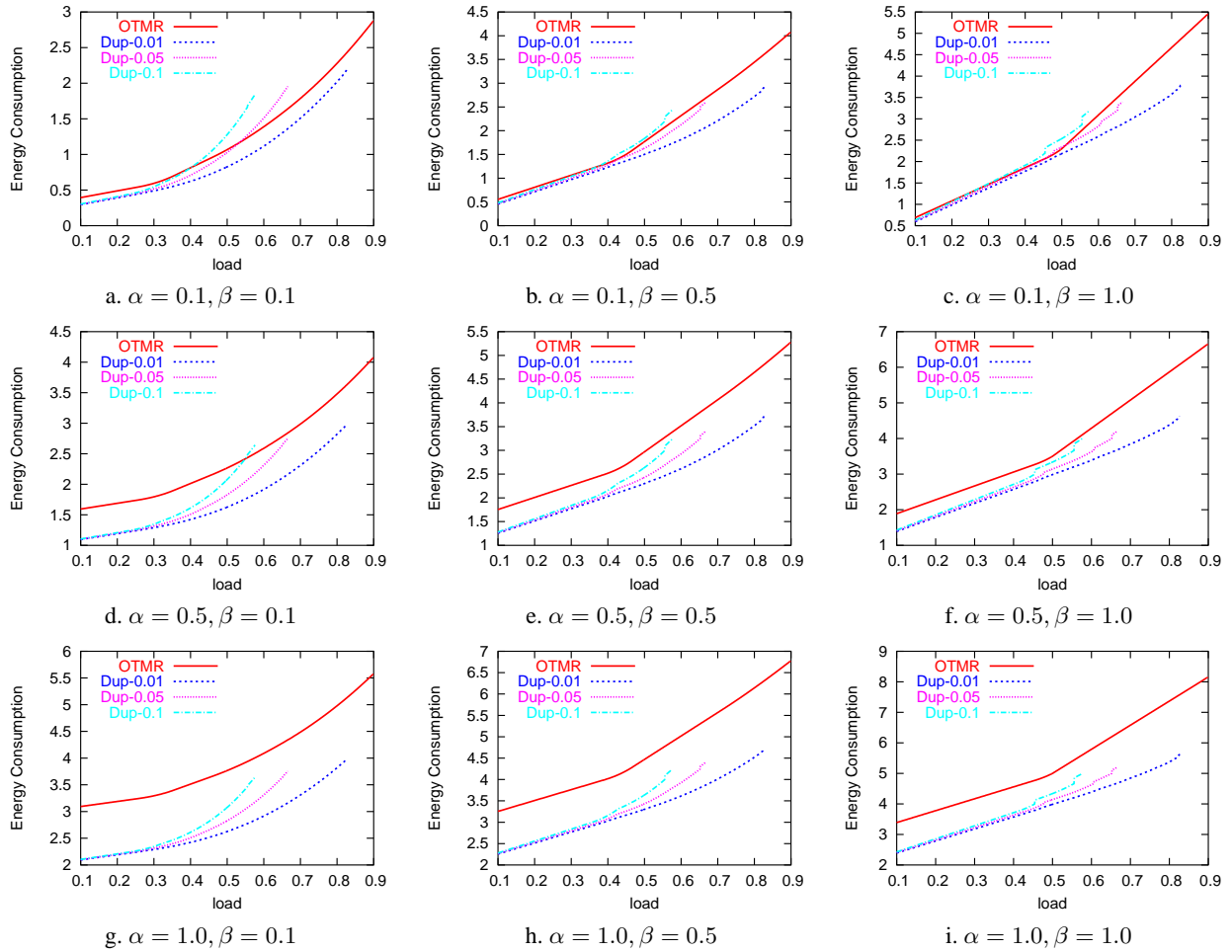


Figure 4. Energy Consumption of OTMR and Duplex; assuming $P_d^{max} D = 1$.

on the reliability achieved by Duplex. With one recovery section, Duplex achieves comparable levels of reliability as that of OTMR, especially with low loads where all executions are performed at the minimum energy efficient speed.

7. Conclusion

An Optimistic TMR (OTMR) scheme has been proposed to reduce the energy consumption for traditional TMR systems by turning off or slowing down one processing unit, provided that it can catch up and finish the computation before the deadline if the computation on the other two units does encounter an error [5]. However, the maximum energy saving obtained by the OTMR scheme depends on appropriate speed setting for the processing units, which in turn is determined by the system power characteristics.

In this paper, we propose a general system power model and analyze the optimal processing speeds to minimize energy consumption for the OTMR scheme. For comparison, the applicability of Duplex with different

checkpoint overheads is studied and the reliability for TMR, OTMR and Duplex is computed by assuming a Poisson distributed fault model.

Our analysis show that if the static power can be efficiently removed when a system is in sleep state (i.e., becomes speed-independent active power), OTMR consumes comparable energy with Duplex and is applicable even when system load approaches 1. However, if the static power, including the power for the underlying circuits, cannot be turned off, Duplex will consume much less energy than OTMR when Duplex is applicable (i.e., when system load is not very high and checkpoint overhead is small). Furthermore, Duplex with optimal checkpoint distribution achieves comparable levels of reliability as that of OTMR.

References

- [1] 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.

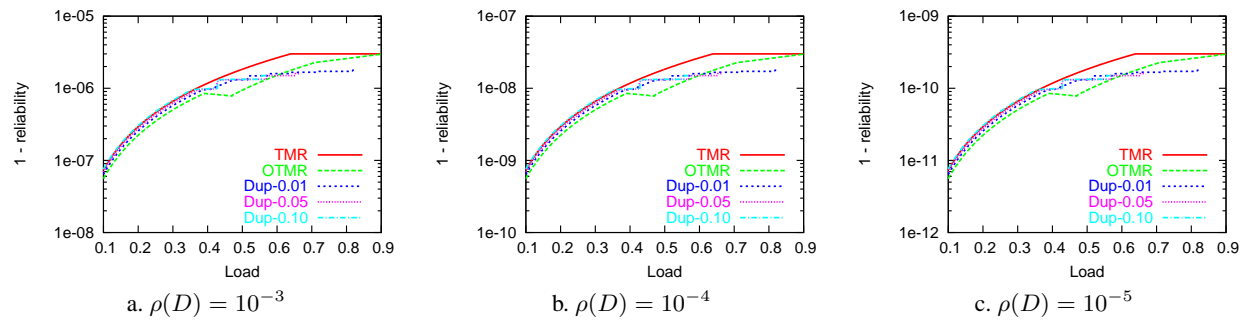


Figure 5. The probability of failure (1-reliability) for all schemes; $\alpha = 0.1$ and $\beta = 0.5$ ($\kappa = 0.6393$).

- [2] T. D. Burd and R. W. Brodersen. Energy efficient cmos microprocessor design. In *Proc. of The HICSS Conference*, pages 288–297, Jan. 1995.
- [3] X. Castillo, S. McConnel, and D. Siewiorek. Derivation and calibration of a transient error reliability model. *IEEE Trans. on computers*, 31(7):658–671, 1982.
- [4] Intel Corp. Mobile pentium iii processor-m datasheet. Order Number: 298340-002, Oct 2001.
- [5] E. (Mootaz) Elnozahy, R. Melhem, and D. Mossé. Energy-efficient duplex and tmr real-time systems. In *Proc. of The 23rd IEEE Real-Time Systems Symposium*, Dec. 2002.
- [6] 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.
- [7] <http://developer.intel.com/design/intelxscale/>.
- [8] <http://www.transmeta.com>.
- [9] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proc. of The 14th Symposium on Discrete Algorithms*, 2003.
- [10] 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*, pages 197–202, Aug. 1998.
- [11] A. R. Lebeck, X. Fan, H. Zeng, and C. S. Ellis. Power aware page allocation. In *Proc. of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 105–116, Nov. 2000.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] D. K. Pradhan. *Fault Tolerance Computing: Theory and Techniques*. Prentice Hall, 1986.
- [16] D. K. Pradhan and N. H. Vaidya. Roll-forward checkpointing scheme: A novel fault-tolerant architecture. *IEEE Trans. on Computers*, 43(10):1163–1174, 1994.
- [17] 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.
- [18] S. Saewong and R. Rajkumar. Practical voltage scaling for fixed-priority rt-systems. In *Proc. of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2003.
- [19] K. Seth, A. Anantaraman, F. Mueller, and E. Rotenberg. Fast: Frequency-aware static timing analysis. In *Proc. of the 24th IEEE Real-Time System Symposium*, 2003.
- [20] A. Sinha and A. P. Chandrakasan. Jouletrack - a web based tool for software energy profiling. In *Proc. of Design Automation Conference*, Jun 2001.
- [21] S. Thompson, P. Packan, and M. Bohr. Mos scaling: Transistor challenges for the 21st century. *Intel Technology Journal*, Q3, 1998.
- [22] 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.
- [23] 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*, pages 13–23, Nov. 1994.
- [24] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Proc. of The 36th Annual Symposium on Foundations of Computer Science*, Oct. 1995.
- [25] 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)*, pages 918–923, 2003.
- [26] 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.
- [27] 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.