Energy-optimal replication for periodic real-time tasks

Xiaolong Cui
mclarencui@cs.pitt.edu

1 Introduction

Energy efficiency is an important design constraint for real-time systems. In large systems. energy costs account for a large portion of the operation expenses. In small systems, such as embeded real-time systems, energy efficiency is extremely important because of the limited battery life. Besides, higher energy consumption has an adverse effect on the environment that attracts more and more attention. A popular technique for energy management is Dynamic Voltage and Frequency Scaling (DVFS). With DVFS, CPU frequency can be reduced by decreasing the voltage supply, which also reduces the power consumption.

At the same time, real-time systems are often deployed for mission-critical tasks, where reliability is a must. Due to electromagnetic interference or cosmic radiation, however, computer systems are susceptible to failures. Therefore, being able to tolerate failures is of extreme importance for real-time critical tasks. With the emergence of multi-core platform, using task replication to improve reliability and tolerate failures becomes a viable solution. Recent study shows that using only two replicas of each task will dramatically improve reliability [1].

Energy efficiency and fault tolerance, however, are two conflicting requirements. Several studies show that DVFS leads to higher failure rate, thereby reducing the reliability of the system [2,3]. On the other hand, running multiple replicas of each task results in higher energy consumption. Therefore, it is challenging to achieve both energy efficiency and fault tolerance simultaneously.

The main objective of this paper is a reliability-oriented energy optimization framework for periodic real-time tasks. Given a real-time task, we study the problem of how to achieve minimal energy consumption while completing the task within its deadline with certain reliability. Three replication techniques are considered in this paper. We use analytical models to study the performance of each replication technique, and an optimization problem is formulated to derive the optimal solution. Furthermore, we build an event-driven simulator to verify the results from our analytical model.

2 Models

2.1 Task and processor model

We consider a single periodic real-time task τ. The task has a Worst Case Execution Time (WCET) of c under the maximum available CPU frequency σ_{max}. The deadline of the task is equal to its period of P. Under the maximum frequency, the utilization U is defined as $\frac{c}{P}$.

With DVFS, each core can operate at a frequency between σ_{min} and σ_{max}. Without loss of generality, we normalize the frequency with respect to the maximum frequency (i.e., $\sigma_{max} = 1$). At frequency σ, a task instance needs up to $\frac{c}{\sigma}$ time to complete.

2.2 Power model

The power consumption of each core consists of both static and dynamic parts. The dynamic power is CPU frequency dependent while the static power is not. The mathematical formulation is $P_{active} = P_s + P_d$. Furthermore, the dynamic power is approximately proportional to the power 3 of frequency. Therefore, we can re-write the power model as $P_{active} = \alpha + (1-\alpha)\sigma^3$, where σ is the CPU frequency and α is a system-dependent constant that measures the weight of static power [4]. With this power model, we can calculate the energy consumption of a task instance as integral of power over the task's execution time.

2.3 Failure model

We consider transient failures in this work, and exponential distribution is assumed. The failure density function is $f(t) = \lambda e^{-\lambda t}$, where λ is the average failure rate. Some research works have shown that using DVFS has a negative effect on reliability. Suppose the average failure rate at the maximum frequency is denoted by λ_0, then the failure rate at frequency λ can be expressed as $\lambda(\sigma) = \lambda_0 10^{\frac{d(1-\sigma)}{1-\sigma_{min}}}$ [2]. d is called the sensitivity factor which measures how quickly the transient failure rate increases when frequency is reduced. As a result, the failure density function is a function of time t and frequency σ, i.e., $f(t,\sigma) = \lambda(\sigma)e^{-\lambda(\sigma)t}$.

The reliability of a single instance of a task is defined as the probability of completing the task successfully. For a task instance with WCET of c and running at frequency σ, its reliability can be calculated as $r(\sigma) = \int_0^{\frac{c}{\sigma}} f(t)dt = e^{-\lambda(\sigma)\frac{c}{\sigma}}$. Conversely, the probability of failure of a task instance is $\phi(\sigma) = 1 - r(\sigma)$.

3 Replication techniques

In order to provide fault tolerance in real-time systems, replication is used that each task has multiple instances that running in parallel on different cores. We say that the task is successfully completed as long as one of the instances is completed without failure and before its deadline. In this work, we only consider two replicas of each task running on two cores. Task reliability R is defined as the probability that both task instances fail before completion.

There are multiple replication techniques. The most naive approach is to have the two replicas running at the maximum frequency. If there is slack in the real-time system, we can slow down both replicas using DVFS to save energy. Furthermore, it is possible to run the two replicas at different frequency, which may lead to further energy saving. In the following subsections, we will describe each of the replication techniques in more details, as well as deducing their reliability and energy consumption.

3.1 Naive replication

With the naive replication, both replicas will execute at the maximum frequency which result in the least response time $T = c$. Also, since both replicas run with the same frequency, the two task replicas have the same re pliability of $r(\sigma_{max})$, and the task reliability is $R = \phi(\sigma_{max})^2$. Since reliability decreases with frequency, it is clear that naive replication will have the highest task reliability. In other words, naive replication has the highest guarantee that at least one task instance will complete without failure. However, this technique is likely to consume the most energy since all cores are always running at the maximum frequency. The energy consumption can be calculated as $E = 2P_{active}(\sigma_{max}) * T$.

3.2 Stretched replication

From above power model we can see that by reducing the execution frequency we can reduce the dynamic power and reduce the total power. If there is enough slack in the system that allows the task to slow down,

we can "stretch" the task execution and still meet the task's deadline, while providing some energy saving. With stretched replication at frequency σ_r, the response time of the task is equal to the response time of each task instance, and the value is $T = \frac{c}{\sigma_r}$. Similar to naive replication, the task reliability can be calculated as $R = \phi(\sigma_r)^2$, and the energy consumption is $E = 2P_{active}(\sigma_r) * T$.

3.3 Shadow replication

Instead of running both replicas at the same frequency, we can run them with different frequencies. Inspired by this, we applied shadow computing to this, i.e., we designate one replica as the main one that runs at a high frequency and one replica as the shadow that runs at a lower frequency. As soon as the main completes, we can terminate the shadow for energy saving. If the main fails, however, we speed up the shadow to complete the task by its deadline. In this technique, there are three frequencies, one is the frequency of the main, σ_m, one is the frequency of the shadow before failure of the main, σ_b, and the last one is the frequency of the shadow after failure of the main, σ_a.

According to our previous definition of task reliability, the reliability of the shadow is intractable as the shadow may change frequency if the main fails. Therefore, we approximate the reliability of the shadow as if it only uses σ_b. This is a pessimistic approximation because $sigma_a$ should be larger than σ_b and should result in higher reliability. With this approximation, the task reliability is modeled as $R = \phi(\sigma_m) * \phi(\sigma_b)$.

Depending on whether the main would fail or not, the task may have two response time. The response is $T = \frac{c}{\sigma_m}$ if the main does not fail; otherwise, the response time is $T = t_f + \frac{c - \sigma_b * t_f}{\sigma_a}$, where t_f if the time when the main fails. The energy consumption can be calculated as $E = (P_{active}(\sigma_m) + P_{active}(\sigma_b)) * T$. Since T is failure dependent, energy consumption also depends on failure.

4 Energy-optimal replication problem

From the above section, it is easy to see that, with different CPU frequency, both stretched replication and shadow replication can have different response time, task reliability, and energy consumption. In order to derive the frequency for stretched replication and shadow replication respectively, we come up with the following problem: Given a task and its task reliability target, determine the frequency assignment such that the energy consumption is minimized while guarantee that the task can be completed before its deadline with required reliability.

4.1 Problem formulation

Assuming CPU frequency is continuous, the problem can be mathematically formulated as:

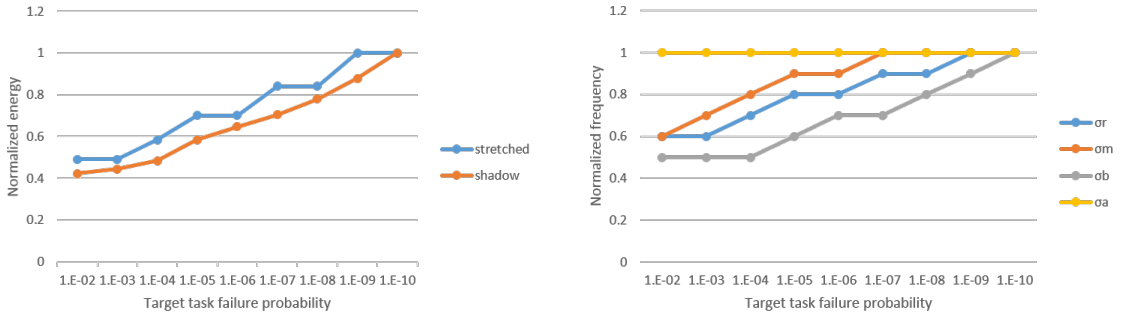
$$\min_{\sigma} \quad E \tag{1}$$

$$s.t. \quad \sigma_{min} \leq \sigma \leq \sigma_{max} \tag{2}$$

$$T \leq P \tag{3}$$

$$R \leq R_{target} \tag{4}$$

Above, Equation (1) says the objective is to minimize energy consumption. Constraint (2) ensures a valid frequency is used for each task instance. For stretched replication, σ_r need to satisfy this requirement, and for shadow replication, σ_m, σ_b, and σ_a all need to satisfy this condition. Constraint (3) says the task's completion time should be earlier than the deadline. For shadow replication, this is equivalent to that the shadow can complete before deadline. The last constraint specifies that the task failure probability should be low enough to satisfy the reliability requirement.



(a) Energy consumption

(b) CPU frequency

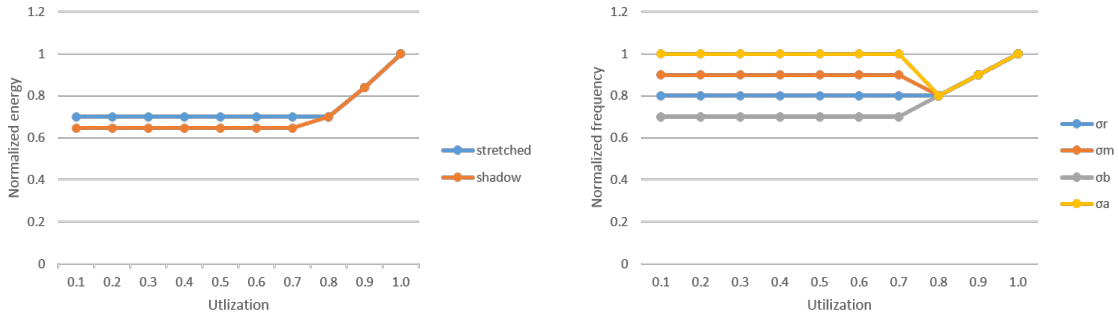
Figure 1: Impact of target task failure probability on frequency assignment and energy consumption. $c = 10ms$, $P = 100ms$, $f_{min} = 0.5$, $\alpha = 0.1$, $d = 4$, $\lambda_0 = 10^{-6}$.

4.2 Analysis

We programmed the energy-optimal replication problem with MatLab and used the nonlinear optimization algorithms implemented in MatLab to solve the problem. However, this approach is quite time-consuming. It takes more than half an hour to solve one instance of the problem. Therefore, we decide to take another path that uses brute force search to find out the optimal frequency assignment. This approach ended up to be much faster than the first approach, as a processor should only have a few valid frequency choices.

Below we show some results in Fig. 1 and Fig. 2. Fig. 1(a) shows the energy consumption comparison among the three replication techniques for different reliability requirement in terms of task failure probability.The parameters are adopted from [3] and shown in the figure title. With two replicas under the specified parameters, the lowest task failure probability that can be achieved is 10^{-10}, which is the rightmost point in the figure. All the results are normalized to the energy consumption of naive replication. It is clear that both stretched replication and shadow replication consume less energy than naive replication. When target task failure probability is high (left side of the figure), both stretched replication and shadow replication are able to slow down to a large extent and achieve the most energy saving compared to naive replication. As target task failure probability increases, the space for slowing down decreases. As a result, the energy saving decreases as well. At the highest reliability requirement, both stretched replication and shadow replication are forced to use the maximum frequency, i.e., they converge to naive replication. Compared to stretched replication, shadow replication is able to save 4% to 10% of energy. The frequency assignment to achieve the optimal energy for stretched replication and shadow replication are shown in Fig. 1(b). As we described above, the assigned frequency increases as the reliability requirement increases, except for σ_a that always uses the maximum frequency.

Fig. 2 shows the energy consumption and frequency assignment for different task utilizations. As expected, the right side of the figures show that the lower the utilization is, the larger space there is for slowing down, and the more energy saving there is for stretched replication and shadow replication. When utilization reaches 0.8, shadow replication converges to stretched replication. The energy and frequency are flat at the left side because the target task failure probability also restricts how much the tasks can slow down, as shown in Fig. 1.



(a) Energy consumption

(b) CPU frequency

Figure 2: Impact of utilization on frequency assignment and energy consumption. $c = 10ms$, $R_{target} = 10^{-6}$, $f_{min} = 0.5$, $\alpha = 0.1$, $d = 4$, $\lambda_0 = 10^{-6}$.

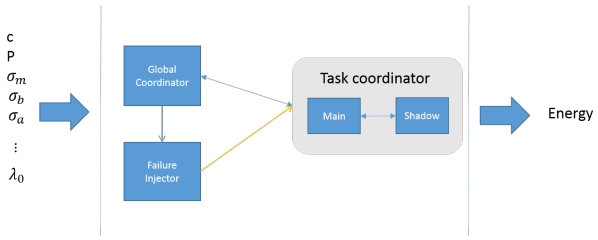


Figure 3: Simulator architecture

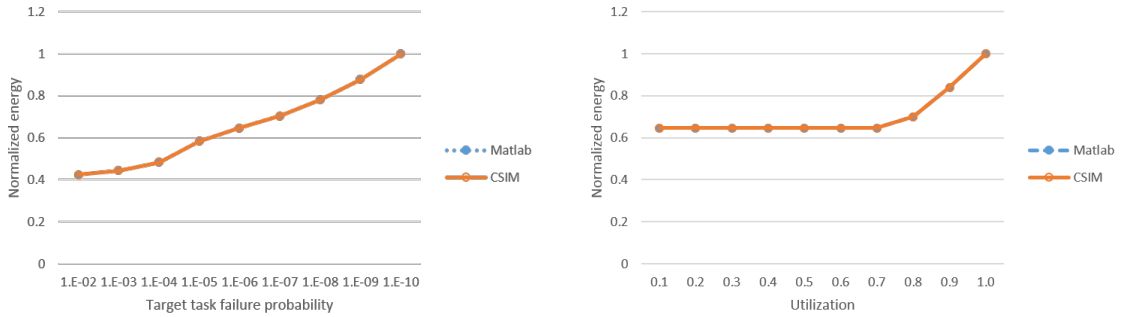
5 Simulation

In order to verify the analytical models, we develop an event-driven simulator for shadow replication with CSIM. The architecture of the simulator is shown in Fig. 3. The simulator has 5 CSIM processes and they interact with each other through CSIM events. The first CSIM process is Global Coordinator that launches all other CSIM processes and output statistics after the simulation is done. Failure Injector is another CSIM process that injects failures to the running task instances according to a specified failure distribution. Main and Shadow are two processes that "execute" the real-time task. They will run at the frequencies derived from analytical models. Finally, we have a Task coordinator that terminates the Shadow process if the Main process finishes, or speed up the Shadow process if the Main process fails. The input to the simulator are the parameters for the task, failure, and frequency. After each simulation, the energy consumption is reported.

Using the simulator, we repeat the study for shadow replication as shown in Fig. 1 and Fig. 2. The results obtained from our simulation are shown in Fig. 4 as the orange lines, and the analytical results are shown as blue dotted lines. Each data point is the average of 20 runs. Due to the very low failure rate ($\lambda_0 = 10^{-6}$), we only see one failure during all the simulation runs. It is clear from Fig. 4 that the results from simulation closely match our previous results from analytical models.

6 Conclusion

Through both analytical models and simulations, we show that using non-uniform frequency assignment among replicas can achieve energy saving while providing the same level of reliability for real-time tasks.



(a) Impact of target task failure probability

(b) Impact of utilization

Figure 4: Comparison between analytical and simulation results. $c = 10ms$, $f_{min} = 0.5$, $\alpha = 0.1$, $d = 4$, $\lambda_0 = 10^{-6}$.

References

[1] H. Casanova, Y. Robert, F. Vivien, and D. Zaidouni, "Combining process replication and checkpointing for resilience on exascale systems," *[Research Report] RR-7951, INRIA*, 2012.

[2] D. Zhu, R. Melhem, and D. Mosse, "The effects of energy management on reliability in real-time embedded systems," in *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on*, Nov 2004, pp. 35–40.

[3] M. Haque, H. Aydin, and D. Zhu, "Energy-aware task replication to manage reliability for periodic real-time applications on multicore platforms," in *Green Computing Conference (IGCC), 2013 International*, June 2013, pp. 1–11.

[4] X. Cui, B. Mills, T. Znati, and R. Melhem, "Shadow replication: An energy-aware, fault-tolerant computational model for green cloud computing," *Energies*, vol. 7, no. 8, pp. 5151–5176, 2014.