Energy-aware Resilience Approaches for Large-scale Systems

Xiaolong Cui
mclarencui@cs.pitt.edu

1 Introduction

As our reliance on IT continues to increase, the complexity and urgency of the problems our society will face in the future will increase much faster than are our abilities to understand and deal with them. Future IT systems are likely to exhibit a level of interconnected complexity that makes it prone to failure and exceptional behaviors. The high risk of relying on IT systems that are failure-prone calls for new approaches to enhance their performance and resiliency to failure. My research focuses on novel and energy-aware computational models that tolerate failures for large-scale distributed systems, including HPC supercomputers and cloud data centers.

Current fault-tolerance approaches are designed to deal with fail-stop errors, and rely on either time or hardware redundancy for recovery. Checkpoint/restart, which uses time redundancy, requires full or partial re-execution when failure occurs. Such an approach can incur a significant delay, and high energy costs due to extended execution time. On the other hand, Process Replication exploits hardware redundancy and executes multiple instances of the same task in parallel to guarantee completion without delay. This solution, however, requires additional hardware resources and increases the energy consumption proportionally.

My proposed approaches to resiliency go beyond adapting or optimizing well known and proven techniques, and explore radical methodologies to fault tolerance in large-scale computing infrastructures. The proposed solutions differ in the type of faults they manage, their design, fault tolerance approach and the fault tolerance protocol they use. It is not just a scale up of "point" solutions, but an exploration of innovative and scalable fault tolerance frameworks. When integrated, it will lead to efficient solutions for a "tunable" resiliency that takes into consideration the nature of the data and the requirements of the application.

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 $P_{active} = P_s + C_e\sigma^3$, where σ is the CPU frequency and C_e is a system-dependent constant. 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 negtive 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 $\lambda(\sigma) = \lambda_0 10^{\frac{d(1-\sigma)}{1-\sigma_{min}}}$. 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, $f(t,\sigma) = \lambda(\sigma)e^{-\lambda(\sigma)t}$.

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 calcuated 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 as least one task instance can complete without failure.

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 differernt 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 reliablity 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 repliability of $r(\sigma_{max})$, and the task reliability is $R = 1 - \phi(\sigma_{max})^2$. 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 calcuated as $R = 1 - \phi(\sigma_r)^2$, 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 $R = 1 - \phi(\sigma_m) * \phi(\sigma_b)$.

Dependeing 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 calcualted as $E = (P_{active}(\sigma_m) + P_{active}(\sigma_b)) * T$. Since T is failure dependent, energy consumption also depends on failure.

4 Enery-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 differernt response time, task reliability, and energy consumption. In order to derive the frequency for stretched replicaiton 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:

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

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

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

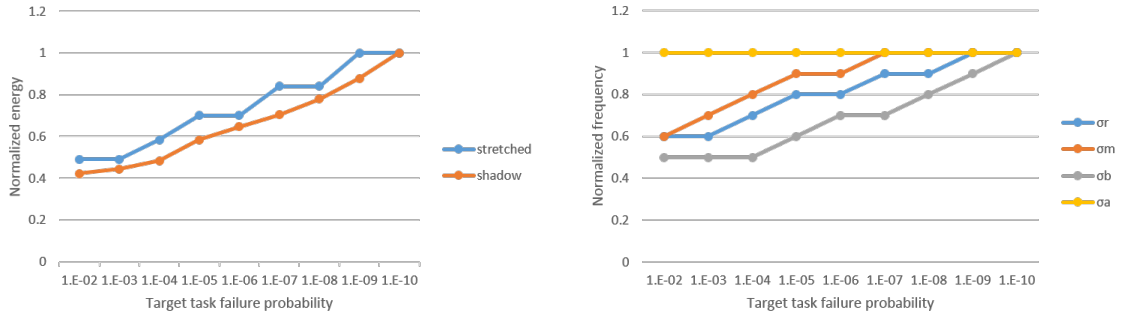
$$R_{target} \leq R \tag{4}$$

Above, Equation (1) says the objective is to minimize energy consumption. Constraint (1) 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'c completion time should be eariler than the deadline. For shadow replication, this is equivalent to that the shadow can complete before deadline. The last constraint requires that the task reliability is no lower than the specified requirement.

This is a nonlinear optimization problem that can be solved by available tools like MatLab.

4.2 Analysis

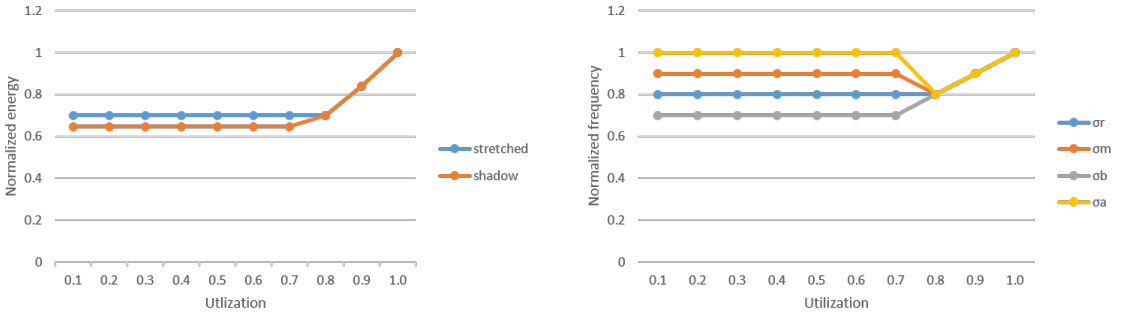
We programmed the energy-optimal replication problem with MatLab and used the nonlinear optimiation algorithms implemented in MatLab to solve the problem. However, this approach is quite time-consuming.



(a) Energy consumption

(b) CPU frequency

Figure 1: Impact of target task failure probability on frequency assignment and energy consumption



(a) Energy consumption

(b) CPU frequency

Figure 2: Impact of utilization on frequency assignment and energy consumption

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.

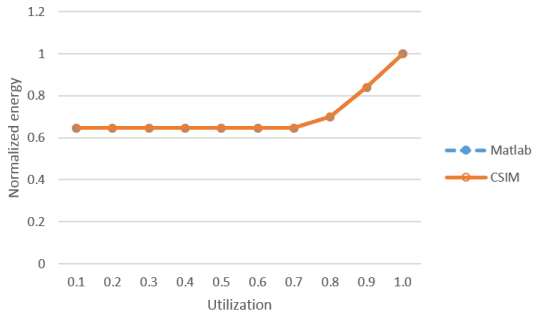
Below we show some results in Fig. 1 and Fig. 2.

5 Simulation

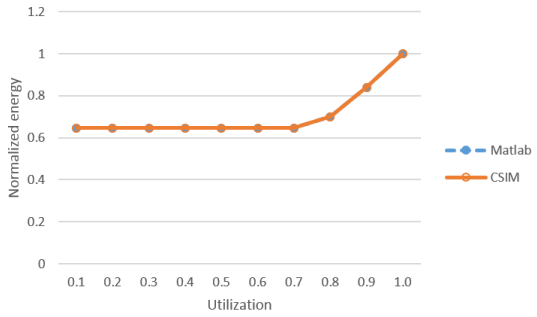
Fig. 3.

6 Conclusion

My current research is enabling new insights into the multi-faceted and challenging resiliency problem in large-scale distributed systems. The goal is to investigate radical approaches to the design of scalable and energy efficient fault tolerant approaches that go beyond state-of-the-art algorithms. Throughout my design, the interplay between resiliency, performance and energy consumption is analyzed carefully to determine the required levels of endurance and redundancy, in order to achieve a desired level of fault tolerance while maintaining a specified level of QoS.



(a) Impact of target task failure probability



(b) Impact of utiliation

Figure 3: Comparison between analytical and simulation results. $c = 10ms$, $f_{min} = 0.5$, $\rho = 0.1$, $d = 4$, $\lambda_0 = 10^{-6}$

Shadow Computing is a novel, scalable, and energy-aware computational model that achieves fault tolerance. Preliminary results show that Shadow Computing is able to achieve significant energy savings while satisfying QoS requirements, and adapt to various large-scale computing environments and requirements. I will continue to explore and optimize Shadow Computing in support of green and sustainable computing.

References