Shadow Computing: An Energy-Aware Fault Tolerant Computing Model

Bryan Mills, Taieb Znati, Rami Melhem
Department of Computer Science
University of Pittsburgh
(bmills, znati, melhem)@cs.pitt.edu

Index Terms—**shadow computing, fault tolerance, scheduling, resiliency**

***Abstract*—The current response to fault tolerance relies upon either time or hardware redundancy in order to mask faults. Time redundancy implies a re-execution of the failed computation after the failure has been detected, although this can further be optimized by the use of checkpoints these solutions still impose a significant delay. In many mission critical systems hardware redundancy has traditionally deployed in the form of process replication to provide fault tolerance, avoiding delay and maintaining tight deadlines. Both approaches have drawbacks, re-execution requiring additional time and replication requiring additional resources, especially energy. This forces the systems engineer to choose between time or hardware redundancy, cloud computing environments have largely chosen replication because response time is often critical. In this paper we propose a new computational model called *shadow computing*, which provides goal-based adaptive resilience through the use of dynamic execution. Using this general model we develop *shadow replication* which enables a parameterized tradeoff between time and hardware redundancy to provide fault tolerance. Then we build an analytical model to predict the expected energy savings and provide an analysis using that model.**

I. Introduction

Power consumption is widely recognized as one of the most significant challenges facing data centers in both cloud computing and high performance computing (HPC) [5], [1]. Addressing such a challenge requires building power and energy awareness into the foundations of future computing models. New approaches must be developed to achieve efficient energy management across all hardware and software components of the system. The increase in energy consumption is a direct result of an increase in the number of computing nodes, conversely this has an equally negative effect on the overall system reliability. Even if the individual node failure rate is low, the overall system failure rate quickly becomes unacceptable as the number of components increases. For example, a computing system with 200,000 nodes will experience a mean time between failure (MTBF) of less than one hour, even when the MTBF of an individual node is 5 years [3]. Future systems will clearly need to be fault tolerant and at the same time be energy-aware.

This work is supported in part by NSF grants CNS12-53218 and CNS12-52306

The common response to faults is to restart the execution of the application after a failure, including those components of its software environment that have been affected by the occurring fault. Re-execution techniques can occur at different points of the application, possibly involving the complete re-execution of the task although usually a checkpoint is used to reduce the amount of rework necessary. In either case the re-execution will cause a delay. To avoid this delay most cloud environments have chosen process replication [6], [8], [4] to mask faults removing the delay of re-execution. It is clear that while replication provides a method of achieving fault tolerance it is not energy efficient because, by its very nature, it costs twice as much as a single process.

The main objective of this paper is to explore a new computational model called *shadow computing* by developing an energy-aware fault-tolerance method called *shadow replication*. We show that this method not only provides fault tolerance but can provide energy savings of 15-30% in existing systems.

A. Shadow Computing

We are proposing a new computational model, called *shadow computing*, which provides goal-based adaptive resilience using dynamic execution to meet the requirements of complex applications in highly parallelized faulty environments. Adaptive Resilience is the ability of the system to dynamically harness all available resources to achieve the highest level of QoS for a given application. Dynamic execution is the ability to execute an application while being able to change the QoS of that application. For example, in this paper we exploit the ability to execute processes at variable execution speeds using dynamic voltage and frequency scaling (DVFS), changing the QoS of application response time. The challenge is to maintain or exceed the applications QoS while minimizing the system resources in spite of systems-level changes, such as failures or the availability of additional system resources. In order to achieve adaptive resilience, the shadow computing model associates a set of *shadows* to the main execution, which are dynamically instantiated and adjusted in order to address the current state of the system and maintain the application's QoS requirements. The objective of this paper is to balance the system resource of power with the QoS of application response time in the presence

of system faults. To this end, we propose *shadow replication* which is the ability of the system to adaptively execute either entirely or partially the original task to overcome failures, while minimizing system power.

B. Shadow Replication

The basic idea of *shadow replication* is to associate with each process a suite of "shadow processes", whose size depends on the "criticality" and performance requirements of the underlying application. A shadow process is an exact replica of the main process. In order to overcome failure, the shadow is scheduled to execute concurrently with the main process, but at a different computing node. Furthermore, in order to minimize energy, shadow processes initially execute at decreasingly lower processor speeds. The successful completion of the main process results in the immediate termination of all shadow processes. If the main process fails, however, the primary shadow process immediately takes over the role of the main process and resumes computation, possibly at an increased speed, in order to complete the task. Moreover, one among the remaining shadow processes is then promoted to be the primary shadow process. The main challenge in realizing the potential of the shadow replication stems from the need to compute the speed of execution of the main process and the speed of execution of its associated shadows, both before and after a failure occurs, so that the target response time is met, while minimizing energy consumption.

Since the failure of an individual component is much lower than the aggregate system failure, it is very likely that most of the time the main processes complete their execution successfully. Successful completion of a main process automatically results in the immediate halting of its associated shadow processes, providing a significant savings in energy consumption. Furthermore, the number of shadow processes to be instantiated in order to achieve the desired level of fault-tolerance must be determined based on the likelihood that more than one process fails within the execution time interval of the main task. The completion of a main or its shadow results in the successful execution of the underlying task.

The main contributions of this paper are threefold. First, we present an optimization framework to explore the applicability of the shadow replication to provide fault-tolerance in an energy efficient manner. Second, using the optimization framework, we propose and study two implementation methods of the shadow replication model. The first method, referred to as "stretched" replication, takes the trivial approach to minimizing energy by reducing the speed of execution of both the main and the replica such that the task deadline is maintained. Although simple to implement, stretched replication is oblivious to the dynamics of the failure, potentially resulting in sub-optimal energy performance. The second method, referred to as energy efficient replication, computes energy efficient execution speeds which maintain the expected response time regardless of failure rates. Third, we conduct a performance evaluation to assess the performance of shadow replication, comparing the energy consumption of shadow replication to that of traditional replication.

II. Energy Optimization Model

In this section, we define a framework for evaluating shadow replication and then use this to derive a model for representing the expected energy consumed by the system.

A. Shadow Replication Framework

We consider a distributed computing environment executing an application carried out by a large number of collaborative tasks. The successful execution of the application depends on the successful completion of all of these tasks. Therefore the failure of a single process delays the entire application, increasing the need for fault tolerance. In this paper we focus on the execution of one single task knowing that one task impacts the entire system, both in total time of execution and energy consumption. Each task must complete a specified amount of work, W, expressed in terms of the number of cycles required to complete the task. Each task also has a targeted response time, t_r. Each computing node has a variable speed, σ, given in cycles per second and bounded such that $0 \leq \sigma \leq \sigma_{max}$. Therefore the minimum response time for a given task is $t_{min} = \frac{W}{\sigma_{max}}$.

In order to achieve our desired fault tolerance a shadow process executes in parallel with the main process on a different computing node. Depending on the occurrence of failure during execution, four scenarios are possible. The first scenario, depicted in Figure 1(a), takes place when no failure occurs[1]. The second scenario, depicted in Figure 1(b), takes place when failure of the main process occurs. Upon failure detection, the shadow process increases its processor speed and executes until completion of the task. The third case is if the main process completes successfully and the shadow fails, not depicted due to space constraints. The last case is both processes fail, however this is very unlikely because failure of the nodes are assumed to be independent events.

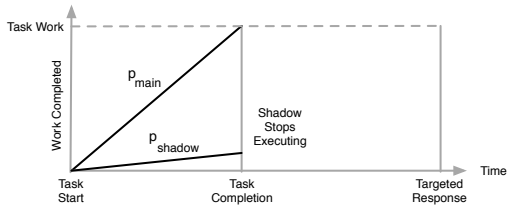
The main process executes at a single execution speed denoted as σ_m. In contrast the shadow process executes at two different speeds, a speed before failure detection, σ_b, and a speed after failure detection, σ_a. This is depicted in Figure 2.

Based upon this framework we define some specific time points signaling system events. The time at which the main process completes a task, t_c, is given as $t_c = \frac{W}{\sigma_m}$. Additionally, we define the time point t_f as the time at which a failure in the main process is detected.

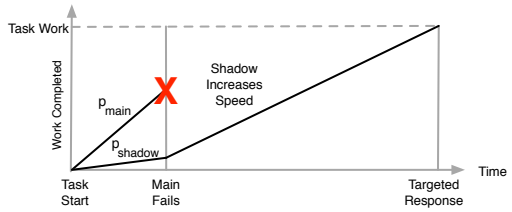
B. Power Model

In the preceding section we have defined shadow replication, now we will derive an analytical model to describe the expected energy consumption of shadow replication. We begin by defining a power model used to describe the power consumed by a single node. It is known that by varying

[1]For the purpose of this discussion, only a single shadow is considered. The discussion can be easily extended to address multiple shadow processes



(a) Case of no failure



(b) Case of failure

Fig. 1. Shadow replication execution model.

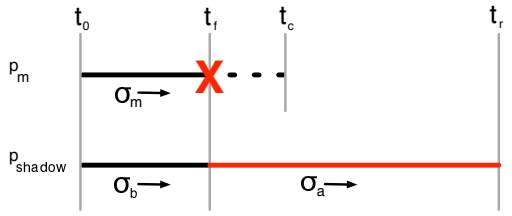


Fig. 2. Overview of Shadow Replication

the execution speed of the computing nodes one can reduce the dynamic CPU power consumption at least quadratically by reducing the execution speed linearly. The dynamic CPU power consumption of a computing node executing at speed σ is given by the function $p(\sigma)$, represented by a polynomial of at least second degree, $p(\sigma) = \sigma^n$ where $n \geq 2$. In the remainder of this paper we assume that the power function is the cubic, $P(\sigma) = \sigma^3$.

We also consider the amount of "static" power which is consumed regardless of the speed of the processor. This consumption of power is made up of both CPU leakage and all other components consuming energy during execution (memory, network, etc.). In this paper we define static power to be a fixed factor of the power consumed when the CPU when operating at full speed, referred to as ρ. The percentage of static power in a system is thus defined as $\frac{\rho}{\rho+1}$. For example by letting ρ be 4.0, when the CPU is executing at maximum speed it will consume 20% of the total power, with static power consuming the remaining 80%.

The energy consumed by a computing node executing at speed σ during an interval $[t_1, t_2]$ is given by $E(\sigma, [t_1, t_2]) = \int_{t=t_1}^{t_2} p(\sigma)dt$. Thus the energy function is defined as the following:

$$E(\sigma, [t_1, t_2]) = \int_{t=t_1}^{t_2} (\sigma^3 + \rho\sigma_{max}^3)dt \\ =(\sigma^3 + \rho\sigma_{max}^3)(t_2 - t_1) \quad (1)$$

C. Failure Model

The failure can occur at any point during the execution of the main or shadow task and the completed work is unrecoverable. Because the processes are executing on different computing nodes we assume failures are independent events. We also assume that only a single failure can occur during the execution of a task. If the main task fails it is therefore implied that the shadow will complete without failure. We can make this assumption because we know the failure of any one node is a rare event thus the failure of any two specific nodes is very unlikely.

We assume that two probability density functions, $f_m(t)$ and $f_s(t)$, exists which expresses the probability of the main or shadow task failing at time t. The model does not assume a specific distribution however in the remainder of this paper we use an exponential probability density function, thus $f_m(t) = f_s(t) = \lambda e^{-\lambda t}$.

D. Energy Model

We define the expected system energy for shadow replication given the power model and the failure distributions. First consider the energy consumed by the main and shadow processes if failure does not occur before reaching t_c.

$$(1 - \int_{t=0}^{t_c} f_m(t)dt) \times (1 - \int_{t=0}^{t_c} f_s(t)dt) \\ \times (E(\sigma_m, [0, t_c]) + E(\sigma_b, [0, t_c])) \quad (2)$$

This first part of this equation is expressing the probabilities that neither process fails during the execution interval of the main task, t_c. The second part is the energy consumed by both the main and shadow processes at their respective processor speeds over the same interval, t_c.

Next, let's consider what happens in the event that the main process completes correctly but the shadow process fails before reaching t_c.

$$(1 - \int_{t=0}^{t_c} f_m(t)dt) \times \\ [E(\sigma_m, [0, t_c]) + \int_{t=0}^{t_c} E(\sigma_b, [0, t]) f_s(t)dt] \quad (3)$$

The first part is the probability that the main process doesn't fail. The second part is the amount of energy the main process will consume if it completes the execution. The last part is the amount of energy consumed by the shadow if it fails before the main task is completed.

Because we are assuming only one failure there is one remaining case to consider, that the main process fails and

the shadow takes over the execution.

$$(1 - \int_{t=0}^{t_r} f_s(t)dt) \times \int_{t=0}^{t_c} [E(\sigma_m, [0,t]) + E(\sigma_b, [0,t]) + E(\sigma_a, [t,t_r])] f_m(t)dt \quad (4)$$

Again the first part of this equation is the probability that the shadow process doesn't fail during the response time. The second half is the energy consumed by the main and shadow process before it fails and then the energy consumed by the shadow taking over the execution. E_{system} is therefore the summation of Equations 2, 3 and 4 which is the total expected system energy consumed by shadow replication assuming only one of the two nodes fails.

E. Optimization Problem

Using this model we formalize our objective as the following minimization problem.

$$\begin{aligned} &\text{minimize } E_{system}(t_r, \lambda, \rho, W, \sigma_m, \sigma_b, \sigma_a) \\ &\text{subject to } t_c \leq t_r \\ &\qquad \sigma_m t_c \geq W \\ &\qquad \sigma_b t_c + \sigma_a(t_r - t_f) \geq W \end{aligned} \quad (5)$$

It is assumed that node failure model properties and task properties are unchangeable system parameters, t_r, λ, ρ and W. Therefore the optimization problem is to find the execution speeds of the processes, σ_m, σ_b and σ_a. This results in one formula and three unknowns ad in the next sections we will make use of non-linear optimization techniques to find the energy efficient execution speeds. Recall that t_c is the time that the main will complete the task given no failure, therefore $t_c = \frac{W}{\sigma_m}$. The time of failure of the main process is defined to be t_f and therefore $t_f \leq t_c$.

III. Determining Execution Speeds

One of the primary goals of cloud computing is to achieve the maximum possible throughput. Thus, when we apply the shadow replication model we must consider the impact failures will have upon the speed and efficiency of the system. Shadow replication gains its energy conserving property by making the assumption that if a failure occurs there is some degree of laxity as to when the work will be completed. Similarly, re-execution techniques assume that if a failure occurs one is willing to endure a time delay to the rollback and re-execute. Because of this we defined targeted response time, t_r, allowing us to bound the laxity. We represent the targeted response time as the laxity factor, α, of the minimum response time, t_{min}. For example if the minimum response time is 100 seconds and the targeted response time is 125 seconds, the laxity factor is 1.25. In contrast, a re-execution technique assumes that if a failure occurs the system will have enough time to re-compute, this results in $\alpha = 2.0$ [2]

[2]This assumes a single failure, if multiple failures occur re-execution has the potential to have $\alpha > 2.0$.

Shadow replication and re-execution techniques both exploit time redundancy to provide fault tolerance, the difference is that shadow replication provides a model to balance the tradeoff between hardware and time redundancy. If one lets $\alpha = 1.0$, this implies there is no laxity, therefore shadow replication will be equivalent to traditional replication. As we increase α shadow replication can begin trading hardware redundancy for time redundancy. We propose two different methods for applying shadow replication.

- Stretched Replication - Based upon the classic approach of slowing down to the minimum execution speeds such that the targeted response time is still achieved. This results in $\sigma_m = \sigma_b = \sigma_a = W/t_r$.
- Energy Efficient Replication - Execute at the energy efficient speeds for both the main process and the shadow process. This requires us to find σ_m, σ_b and σ_a that minimize the expected energy.

To solve the optimization problem we begin with the observation that the speed of the shadow after failure, σ_a, is dependent upon the speed before failure, σ_b, and the time of failure, t_f. In this optimization we then let σ_a be the slowest possible speed to finish by the targeted response time, t_r. Therefore σ_a is no longer constant with respect to the time of failure. From this observation the following value of σ_a can be derived.

$$\sigma_a = (W - \sigma_b * t_f)/(t_r - t_f) \quad (6)$$

This reduces the output of our optimization to two variables, σ_m and σ_b.

To ensure that the shadow process has the ability to complete all the necessary work and maintain the targeted response time we must further constrain σ_b. We can derive this constraint from Equation 6 by letting $t_f = t_c$ and $\sigma_a = \sigma_{max}$. This assumes failure occurs at the last possible time, t_c, which would force the speed of the shadow after failure to be the maximum possible execution speed. We can then derive the following "work constraint".

$$t_c * \sigma_b + (t_r - t_c) * \sigma_{max} \geq W \quad (7)$$

This constraint implies that the faster the main process executes the more time available to recover from failure but the more energy we consume in the non-failure case. If we execute the main process slower, the non-failure case can conserve more energy, however we have less time to recover from a failure.

Using this model and the defined constraints we use Mathmatica's non-linear optimization routines to find energy optimal execution speeds, σ_m and σ_b. Without loss of generality, we assume σ_{max} is normalized such that $\sigma_{max} = 1$.

IV. Analysis

Many cloud providers are using traditional replication, therefore we begin our comparison by looking at the potential energy savings shadow replication can provide over this form of replication. In traditional replication each process is executed simultaneously with at least one replica process and each

process executes at the maximum speed. One would expect that shadow replication will save energy but the question is how much energy and under what circumstances. All execution speeds in this section will be presented as factors of σ_{max}.

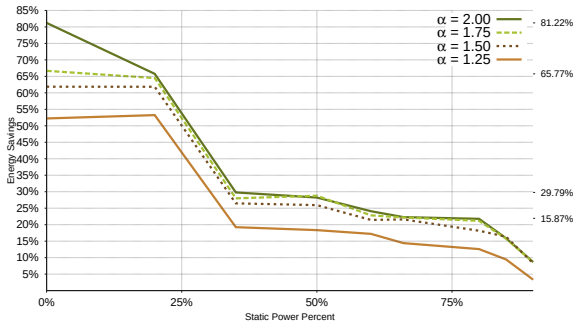


Fig. 3. Energy savings of energy efficient replication when compared to pure replication as we vary static power. $W = 1$ hour and $MTBF = 5$ years.

In Figure 3 we show the energy savings for multiple values of α, as expected the higher the α value the more energy that can be saved, demonstrating the tradeoff between time and hardware redundancy. We choose to plot α values from 1-2 because the amount of savings plateaus after 2, especially for high static power percentages. We show these results of work size of 1 hour because our sensitivity analysis of the energy model shows work size has very little effect on the energy consumption when the job sizes are much smaller than the node MTBF, $W << MTBF$. Given that our node MTBF is 5 years all most reasonable values for work are in this category.

The amount of energy savings is also dependent upon the static power factor, ρ. We therefore vary the percentage of static power present in the system in Figure 3. As we increase the static power the potential energy savings decreases and eventually matches pure replication. The maximum savings occurs when the static power is lowest and α is highest, at this point we can save up to 81% of expected energy for a given task. As static power approaches 100% the amount of savings decreases to 0%. There is experimental data to support a variety of static power values, some studies show that the CPU accounts 40-50% [2] of the consumed power and more recent studies show that number to be 15-30% [7]. Therefore we believe we should be focusing on the acheivable savings when static power is between 50-85%. In this range energy efficient replication can save between 15-30% of the energy consumed by traditional replication.

In Figure 4 we look at the energy savings achievable with stretched replication when compared to that of traditional replication. Similar to energy efficient replication, we observe that as the static power increases the amount of energy savings decreases. Additionally as static power increases the higher the α value the less energy one can save with stretched replication. Because while decreasing the execution speeds saves power at a given point in time it also increases the execute time thus causing stretched replication to consume more overall energy.

Comparing Figures 3 and 4 one can see that energy efficient replication saves more energy for most percentages of static

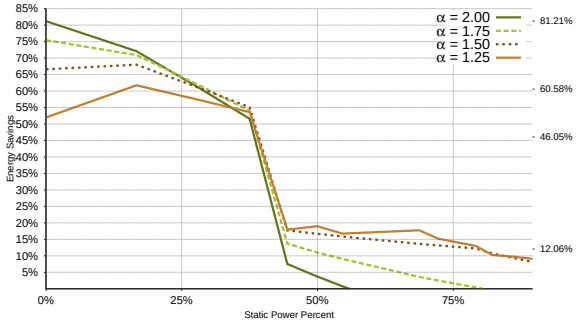


Fig. 4. Energy savings of stretched replication when compared to pure replication as we vary static power. $W = 1$ hour and $MTBF = 5$ years.

power. However, as static power reaches 100% the energy savings of both stretched and energy efficient replication converge to zero. This is understandable because both of these energy-aware replication techniques make the assumption that slowing down the execution speed will be able to reduce energy consumption.

V. Conclusion

The major contribution of this paper is a new technique, called *shadow replication* that provides energy-aware fault tolerance in large-scale distributed computing environments. We also develop a general framework for evaluating the energy consumption of replication-based fault tolerant techniques. Using this energy model we then compare shadow replication to current replication techniques.

We show that shadow replication has the potential of providing energy savings of 15-30% of the energy consumed by traditional replication assuming a static power is 50-85%, which is typical of most systems today. We further show that a simple implementation of stretched replication has the ability to also save energy but that energy efficient replication provides additional energy savings of 5-10%.

Most importantly these results show that shadow replication has the potential to provide fault tolerance while also providing energy savings over existing fault tolerant techniques. Motivated by these results we have began work on implementing shadow replication. This implementation will then be used to measure the actual energy savings achievable by such a system.

References

[1] L. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[2] X. Feng, R. Ge, and K. Cameron. Power and energy profiling of scientific applications on distributed systems. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 34–34, 2005.

[3] K. B. Ferreira. *Keeping Checkpoint/Restart Viable for Exascale Systems*. PhD thesis, University of New Mexico, Albuquerque, New Mexico USA, June 2011.

[4] B. Li, S. Song, I. Bezakova, and K. Cameron. Energy-aware replica selection for data-intensive services in cloud. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on*, pages 504–506, 2012.

[5] U. D. of Energy Office of Science. The opportunities and challenges of exascale computing, 2010.

[6] W.-T. Tsai, P. Zhong, J. Elston, X. Bai, and Y. Chen. Service replication strategies with mapreduce in clouds. In *Proceedings of the 2011 Tenth International Symposium on Autonomous Decentralized Systems*, ISADS '11, pages 381–388, Washington, DC, USA, 2011. IEEE Computer Society.

[7] K. Yoshii, K. Iskra, R. Gupta, P. Beckman, V. Vishwanath, C. Yu, and S. Coghlan. Evaluating power-monitoring capabilities on ibm blue gene/p and blue gene/q. In *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, pages 36–44, 2012.

[8] Q. Zheng. Improving mapreduce fault tolerance in the cloud. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–6, 2010.