Fluid NMR - Performing Power/Reliability Tradeoffs for Applications with Error Tolerance

John Sartori, Joseph Sloan, Rakesh Kumar – UIUC

ABSTRACT

N-modular redundancy (NMR) [1] has long been the most prevalent fault-tolerance technique. However, traditional NMR is agnostic of application characteristics (especially, an application's error tolerance) causing it be overly expensive in terms of performance and power. In this paper, we investigate *fluid NMR*, a framework for NMR-based fault tolerance that takes into account an application's error tolerance to perform dynamic power/reliability tradeoffs. Our case study using face detection as an example application with error tolerance demonstrates that a fluid NMR framework can produce significant power and performance benefits over traditional NMR

1. INTRODUCTION

N-modular redundancy (NMR) is a traditional voting-based fault tolerance technique in which N-copies of a program/hardware module are run in parallel, and their results are voted on using some fixed voting strategy. Triple Modular Redundancy (TMR), for example, executes three copies of a program/hardware module in parallel, and compares the results of execution of each of the copies against each other. If two or more results agree (i.e, have the same value), the matching result is accepted as the correct result of execution. If none of the results agree (a highly unlikely case), the three copies are re-executed and voting is done again.

While the simplicity of NMR implementations make them popular [1,2], one notable limitation of NMR-based fault tolerance is high overhead in terms of performance and power. Redundancy represents up to linear opportunity cost in terms of performance and at least linear cost in terms of power. Such costs can be prohibitive if the extent of redundancy needs to be high. Considerable redundancy may be required when a large number of errors must be detected/corrected by the NMR framework or when reliability targets for the framework are high.

One way to reduce the amount of redundancy required by an NMR framework is by making the framework aware of the inherent error tolerance in applications. A large class of important emerging applications [3, 4, 5] has inherent algorithmic error tolerance [6]. Algorithmic error tolerance refers to the property of soft computations [7] to absorb errors in the form of degraded system outputs instead of treating them as exceptions. Traditional NMR is agnostic of any error tolerance that may be inherent in the applications and thus assumes the responsibility for detecting/correcting every error. This places an increased burden on the NMR framework, thereby increasing its power and performance cost.

In this paper, we examine the potential benefits of *fluid NMR*, a framework for NMR-based fault tolerance that takes into account an application's error tolerance to perform dynamic performance/power/reliability tradeoffs. We first present an analytical model that relates system reliability to application characteristics and motivates the need for a fluid NMR framework. Then, through a case study that uses face detection as an example of an application with error tolerance, we demonstrate that there can be significant power and performance benefits from having a fluid NMR framework. The hardware and software implementations of the framework will be part of future work.

2. A SIMPLE ANALYTICAL MODEL RELATING APPLICATION CHARACTERISTICS AND SYSTEM RELIABILITY

System reliability ($Rsys$) can be characterized as a function of the following four factors:

- *Component reliability (p)* – the probability that execution gives the correct result. Note that non-unit component reliability can be due to

a software or a hardware fault.

- *Size of output space (r)* – the number of possible incorrect outputs in the face of faults plus k, the number of acceptable outputs. For applications with error tolerance, a range of outputs (e.g., outputs with higher order bits being correct) may be acceptable (even if some outputs may be less desirable than others). For simplicity, we assume that the probability of a given wrong output is the same for all wrong outputs.
- *Latency guarantee (i_{max})* – the maximum number of re-executions allowed in case NMR fails to proceed successfully [1]. Traditional NMR requires re-executions until an acceptable output is reached. For applications with error tolerance, an inexact/incorrect output may be accepted after i_{max} iterations in order to meet a latency guarantee.
- *Voting strategy (m of n)* – an m of n voting strategy involves executing n copies in parallel and accepting a result that is produced by *at least* m modules. TMR, for example, follows a 2 of 3 voting strategy.

Consider q, the probability that a module fails with a particular output. Assuming that all incorrect outputs are equally probable [8]:

$$q = \frac{1-p}{r-k}$$

(note that there are $r-k$ wrong outputs)

Pd, the probability that a fault is detected by NMR, can then be calculated as follows:

$$Pd = \sum_{i=0}^{m-1} \binom{n}{i} E(r-k, n-i, m-1) p^i q^{n-i}$$

where

$$E(t,k,s) = \sum_{0<n_{ij}\leq s, \sum_{j=1}^{t} n_{ij}} \frac{k!}{n_{i1}! n_{i2}! \cdots n_{it}!}$$

To calculate R, the probability that NMR accepts the correct result and moves forward, we must consider the cases where $m < ceil(n+1/2)$ and $m \geq ceil(n+1/2)$ separately. When $m < ceil(n+1/2)$, there can be multiple sets of modules of size m or more that agree on different values. To resolve the non-determinism during voting, we use following two policies:

- If there are multiple sets of matching values of size $\geq m$, accept the largest unique set.
- If there is no unique set (i.e., several sets have the largest size), then randomly select the accepted value from one of these sets.

R can then be computed with the following equation (when $m < ceil(n+1/2)$):

$$R = \sum_{i=m}^{n} \binom{n}{i} E(r-k, n-i, i-1) p^i q^{n-i}$$

The equation for R when $m \geq ceil(n+1/2)$ is:

$$R = \sum_{i=m}^{n} \binom{n}{i} p^i (1-p)^{n-i}$$

To calculate expected system reliability, R_{sys}, we assume that no more than i_{max} re-executions are allowed and that the same voting strategy with the same value of r is used for re-executions.

$$Rsys = R \sum_{i=0}^{imax} Pd^i = R \frac{1 - Pd^{imax+1}}{1 - Pd}$$

The above equation relates expected system reliability to application characteristics. The various parameters can be tuned to perform performance/power/reliability tradeoffs for applications with error tolerance.

3. FLUID NMR TRADEOFFS

Figures 1a, 1b, and 1c use the above equation to show the dependence of system reliability on component reliability, latency guarantee, and the size of the output space for NMR when up to 8 nodes can be used for voting. The graphs show system reliabilities only for the voting strategies that are optimal in at least one interval.

The first thing to note in all three graphs is that there are several voting strategies that are optimal in at least one interval. For example, in Figure 1a, while simplex is optimal for component reliability less than 0.32, 5-of-8 is optimal when the component reliability is between 0.44 and 0.72. Similarly, in Figure 1b, while 5-of-8 is the optimal strategy when no more than 10 re-executions are allowed, 5-of-7 is the optimal strategy when up to 16 re-executions are allowed. In Figure 1c, while 5-of-8 is the optimal strategy when the output space is less than 5, 4-of-7 is the best strategy when the output space is between 6 and 8.

The graphs also show that while up to 8 cores are available for voting, there are several optimal strategies that use fewer cores. For example, in Figure 1a, the optimal strategy uses 8 cores when the component reliability is between 0.44 and 0.72, while

[1] A success is defined as the NMR signaling that no error occurred or that an error was masked. NMR can succeed with an incorrect value when several modules agree on an incorrect value; however, this is unlikely.

the optimal strategy uses 5 cores when the component reliability is 0.34. We observe similar cases in Figures 1b and 1c. For certain operating conditions (i.e. output space of voting, voting/checkpoint period, latency guarantees) additional redundancy may lead to lower reliability, since the probability of false positives during voting and/or the probability of not reaching quorum by deadline may be significant.

The results show that a fluid NMR framework that allows dynamic switching between voting strategies, as well as between the number of modules constituting an NMR group (N) based on an application's characteristics (e.g., latency constraints, output space, etc.), has the potential for significant power and performance benefits for the same reliability for applications with error tolerance. This is because any time a module is freed up from NMR (e.g., when then the optimal voting strategy uses fewer cores), it can be either disabled to save power or can be used to do other useful work, improving overall throughput. In Figure 1a, for example, up to 37.5% power can be saved by turning three cores off when component reliability is 0.34. Note that switching to fewer cores in this case means lower power AND higher system reliability.

The results also show that higher reliability targets can be met for the same power or performance (i.e., same value of N) by switching to the currently optimal voting strategy. Optimality, of course, depends on the component reliability as well the current latency constraint and output space of the error-tolerant application.

4. CASE STUDY: FACE DETECTION ON A MULTICORE ARCHITECTURE

To demonstrate the benefits of fluid NMR, we select face detection [9] as the target application. Face detection is naturally robust to errors and does not require strict computational correctness. Errors result in reduced output quality (false positive or negative detections) rather than program failure. Also, our algorithm for face detection [9] is naturally parallelizable – i.e., using more cores can increase the number of images for which detection can be performed with the same accuracy per unit time. Figure 2 shows how image throughput increases with increased parallelism when running face detection on a multicore processor simulator [10] simulating chips with 32 UltraSparc T1-like single-threaded cores. For such an application, a natural tradeoff exists between parallelism (or throughput) and redundancy (or reliability/power) when it is run on a multi-core architecture. The cores of a multicore architecture can either be used to improve the image throughput of face detection or improve the accuracy of face detection.

The redundancy/power tradeoffs are even stronger when there is a direct relationship between power and error rate. For our study, we consider a scenario in which errors are introduced due to voltage overscaling [11, 12], which is used to save power in a processor. Figure 3 shows how error rate increases and power consumption decreases as voltage is scaled down for our simulated processor that consists of gracefully degrading datapath units (e.g., ripple carry adder, Wallace tree multiplier, etc.). Power estimation is done using Wattch [13]. Error rates are estimated through circuit-level simulations of processor modules containing arithmetic logic units. These modules account for approximately 80% of the dynamic power consumption of the processor.

The goal of a fluid NMR framework is to exploit the above tradeoffs for power and throughput benefits. Figure 4 shows how face detection accuracy degrades as voltage is scaled down on the processor, where accuracy is defined as:

$$\frac{good\ detections - false\ positives}{good\ detections + false\ negatives}$$

To simulate the effects of faulty arithmetic units as voltage scales down, we inject errors into the results of arithmetic operations in the face detection program, based on the characteristics of the error distribution in Figure 3.

Figure 4 shows detection accuracy at maximum throughput (70 fps), when all cores are devoted to parallelism. When the target throughput is lower, however, not all cores are needed for throughput, and the fluid NMR framework can make tradeoffs between various objectives to achieve the optimal system configuration in terms of reliability and power. For example, if the objective is power efficiency rather than throughput, fluid NMR can configure the system to use 16 cores instead of 32, trading 10 fps for 46% power savings while achieving the same reliability.

For a target throughput of 45 fps, each face detection module needs 8 parallel threads, allowing for up to 4-way redundancy ($N \leq 4$). Figure 5 shows detection accuracy for available redundancy strategies. In the face detection algorithm, redundant computations vote on whether or not a sub-window in the image contains a face. The first thing to notice is that the strategy that uses the most cores (3 of 4) also has the lowest accuracy. Thus, switching to 2 of 3 for higher voltages provides both power savings (23%) and improved reliability. At 45 fps, a higher accuracy can be obtained than at higher frame rates, evidencing the ability of fluid NMR to trade throughput for reliability. Fluid NMR can also trade reliability for power by switching to simplex and reducing power consumption by 69% while sacrificing at most 11% accuracy (7% on average). At low voltages, this strategy even produces the

highest reliability at an extreme power savings of 84%. Note that these tradeoffs would not be possible in a traditional NMR system.

30 fps represents a common frame rate for many applications. Since this throughput can be sustained with 4 parallel threads, fluid NMR can employ strategies with $N \leq 8$. Figure 6 shows the detection accuracy and power efficiency (in accuracy per watt) achieved by different redundancy strategies.

Figure 6 further motivates fluid NMR by demonstrating several tradeoffs between reliability and power. In scenarios when low power operation is most important, 2 of 3 represents the best configuration, especially for low voltages, where it also provides the highest detection accuracy. However, other strategies (4 of 7 and 3 of 5) provide higher accuracy over a wide range of voltages. Also, since 4 of 7 has higher accuracy than 3 of 5 but lower power efficiency, another tradeoff is possible for a fluid NMR system, depending on the reliability and power requirements of the system.

Finally, not all applications require the same frame rates. Figure 7 shows the optimal strategy in terms of accuracy and power efficiency at each frame rate, demonstrating several throughput/reliability/power tradeoffs that are available in a fluid NMR system but unavailable in traditional NMR.

5. SUMMARY AND CONCLUSIONS

We presented *fluid NMR* – an NMR-based fault tolerance framework that dynamically switches between various voting strategies and the number of modules to be used for voting based on an application's error tolerance. We first developed an analytical model that expresses the relationship between expected system reliability and an error-tolerant application's characteristics, such as latency requirements and output space. The model motivates the need for a fluid NMR framework. Then, we showed significant simulated power and throughput benefits for a face detection algorithm when using fluid NMR. As a large class of emerging applications have algorithmic and cognitive error tolerance, the potential benefits from fluid NMR will only increase. Our future work will address hardware and software implementations of fluid NMR.

6. REFERENCES

[1] D. Jewett, "Integrity s2: a fault-tolerant unix platform," *Fault-Tolerant Computing, 1991. FTCS-21. Digest of Papers., Twenty-First International Symposium*, pp. 512–519, Jun 1991.

[2] A. Shye, T. Moseley, V. J. Reddi, J. Blomstedt, and D. A. Connors, "Using process-level redundancy to exploit multiple cores for transient fault tolerance," in *DSN '07: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 297–306.

[3] *A Platform 2015 Workload Model Recognition, Mining and Synthesis Moves Computers to the Era of Tera*, Intel Corp. [Online]. Available: ftp://download.intel.com/technology/computing/archinnov/plat

[4] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Trans. Comput.*, vol. 33, no. 6, pp. 518–528, 1984.

[5] G. Redinbo, "Generalized algorithm-based fault tolerance: error correction via kalman estimation," *Computers, IEEE Transactions on*, vol. 47, no. 6, pp. 639–655, Jun 1998.

[6] W. Baek, J. Chung, C. C. Minh, C. Kozyrakis, and K. Olukotun, "Towards soft optimization techniques for parallel cognitive applications," in *SPAA '07: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*. New York, NY, USA: ACM, 2007, pp. 59–60.

[7] X. Li and D. Yeung, "Exploiting soft computing for increased fault tolerance," in *In Proceedings of Workshop on Architectural Support for Gigascale Integration*, 2006.

[8] D. McAllister, C.-E. Sun, and M. Vouk, "Reliability of voting in fault-tolerant software systems for small output-spaces," *Reliability, IEEE Transactions on*, vol. 39, no. 5, pp. 524–534, Dec 1990.

[9] P. Viola and M. Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.

[10] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The m5 simulator: Modeling networked systems," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, 2006.

[11] S. Narayanan, G. Lyle, R. Kumar, and D. Jones, "Testing the critical operating point (cop) hypothesis using fpga emulation of timing errors in over-scaled soft-processors," in *In SELSE 5 Workshop - Silicon Errors in Logic - System Effects*, 2009.

[12] L. N. Chakrapani, P. Korkmaz, B. E. S. Akgul, and K. V. Palem, "Probabilistic system-n-a-chip architectures," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 12, no. 3, pp. 1–28, 2007.

[13] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *International Symposium on Computer Architecture*, June 2000.

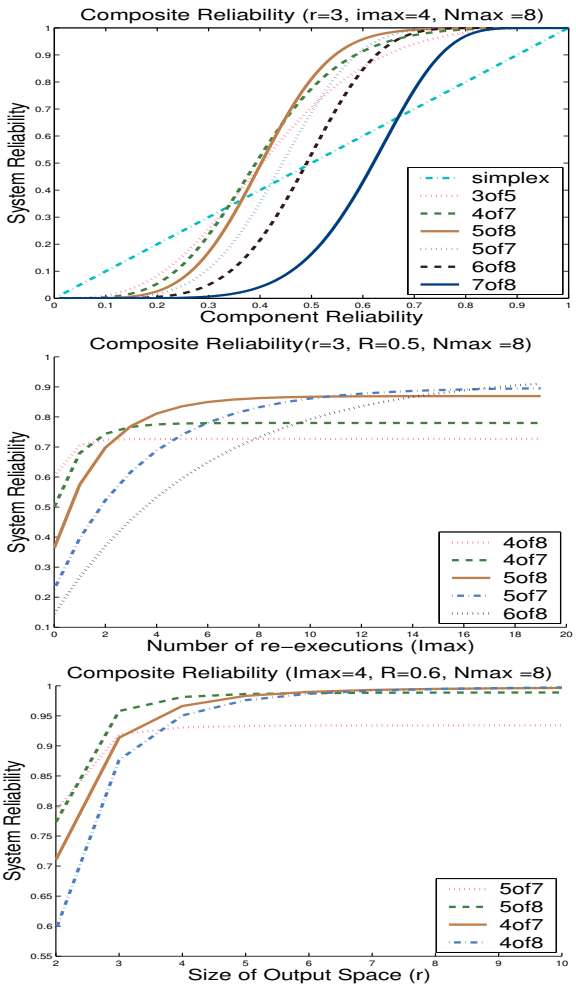

Figure 1: Dependence of System Reliability on (a) Component Reliability, (b) Latency Guarantee, and (c) Size of Output Space for NMR ($N_{max} = 8$)

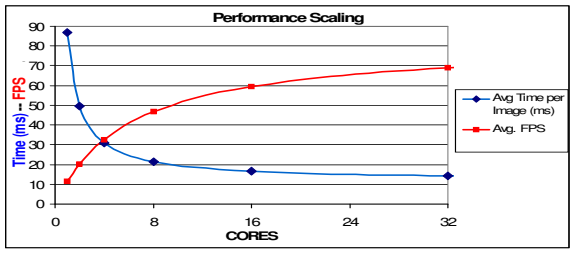

Figure 2: Devoting more cores to parallel face detection increases throughput in fps.

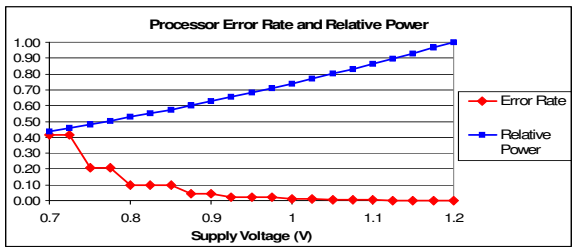

Figure 3: As voltage decreases, power consumption decreases at the expense of increased error rates due to timing violations.

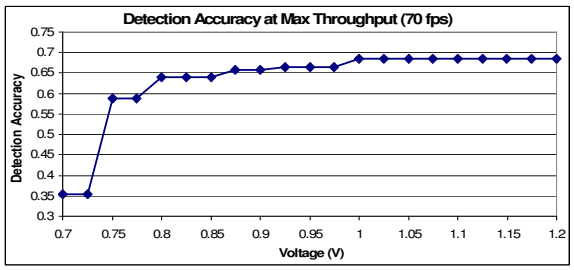

Figure 4: Detection accuracy at maximum throughput, when no cores are devoted to increasing reliability through redundant computation.

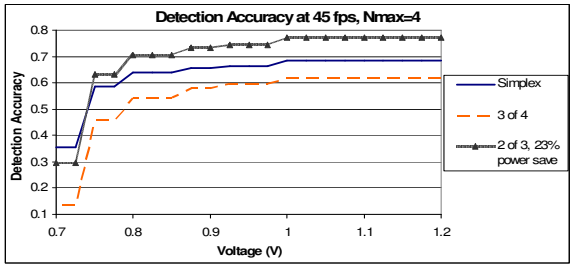

Figure 5: At 45 fps, tradeoffs involving reliability open up, since some cores are free for redundant computation.

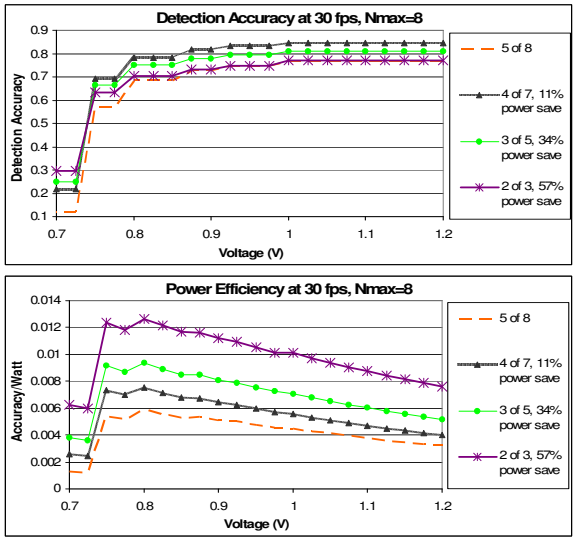

Figure 6: Comparing detection accuracy and power efficiency of several techniques for a frame rate of 30 fps.

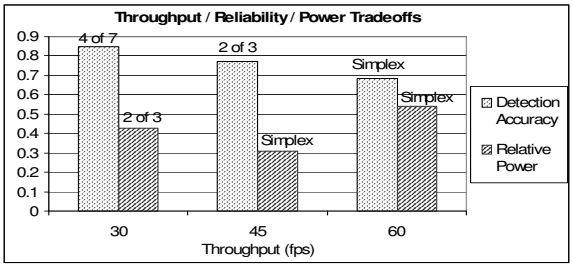

Figure 7: The optimal redundancy strategy varies depending on application and system requirements.