Adapting to Intermittent Faults in Future Multicore Systems

Philip M. Wells, Koushik Chakraborty, Gurindar S. Sohi
Computer Sciences Department
University of Wisconsin, Madison
$\{$*pwells, kchak, sohi*$\}$*@cs.wisc.edu*

1 Intermittent Faults

As technology continues to scale, future multicore processors become more susceptible to a variety of hardware failures. In particular, *intermittent faults*, are expected to become especially problematic [1, 2]. A circuit is susceptible to intermittent faults when manufacturing process variation or in-progress wear-out causes the parameters (e.g., resistance, threshold voltage, etc.) of devices within the circuit to vary beyond design expectations [2]. This susceptibility, combined with certain operating conditions, such as thermal hot-spots and voltage fluctuations, can result in timing errors — even if these temperatures and voltages, for example, are well within the specified "acceptable" margins.

Unlike transient faults, which disappear quickly, or permanent faults, which persist indefinitely, the occurrence of intermittent faults is bursty in nature. Depending on the cause, these bursts of frequent faults can last from several cycles to several seconds or more, effectively rendering a core useless during this time.

2 Adapting to Intermittent Faults

For this work, we assume suitable techniques are available for detection and recovery from intermittent faults, and focus on the necessity of *adapting* to the dynamically changing resource availability caused by these faults. We investigate three existing adaptation techniques, across a range of intermittent fault durations, and demonstrate their different implications for software. The techniques are 1) using spare cores, 2) pausing execution on a temporarily faulty core without notifying the OS, and 3) asking the OS to reconfigure itself to not use the faulty core. To remedy drawbacks of first three, we propose a fourth technique: using a thin hardware/firmware virtualization layer to manage an *overcommitted* system — one where the OS is configured to use more virtual processors than the number of currently available physical cores. To facilitate virtualizing unmodified Solaris, we use hardware spin detection [3] and fast thread migration.

3 Results Summary

Across a range of fault durations, reserving one or more cores as *spares* has little impact on performance during faults, but high fault-free cost. The viability of the two other existing techniques depends upon the fault duration. For short ($<$1ms) faults, we show that simply pausing execution on the core is possible. On the other hand, OS reconfiguration is inappropriate, since Solaris requires several milliseconds to stop using a processor — and requires that processor to continue to function properly in the meantime.

The situation reverses for longer duration faults. For faults of 10ms, OS reconfiguration does incur a 12–55% overhead on an 8-core system, since all cores are involved in bringing one offline, but this overhead is amortized as fault durations increase. For these longer faults, however, we demonstrate that simply pausing a core is no longer attractive, as the the effects are not isolated to the paused core. In fact, for two workloads, more than half the remaining processors are prevented from making forward progress within 1ms, and within 12ms, no processors are making progress.

We show that a thin virtualization layer, including hardware spin detection and fast thread migration, can gracefully degrade performance during periods of overcommitment due to intermittent faults. Of the four, this technique is applicable across the widest range of fault durations, and is the only technique which holds overheads to less than 1% in all experiments. This technique also achieves stable and predictable performance, maintains fairness among virtual processors, has only moderate complexity, and supports legacy OS and application software.

References

[1] S. Borkar, T. Karnik, J. Tschanz, A. Keshavarzi, and V. De. Parameter variations and impact on circuits and microarchitecture. In *DAC*, 2003.

[2] C. Constantinescu. Intermittent faults in VLSI circuits. In *SELSE*, 2007.

[3] P. M. Wells, K. Chakraborty, and G. S. Sohi. Hardware support for spin management in overcommitted virtual machines. In *PACT*, 2006.