**Motivation**

SoftWare Anomaly Treatment (SWAT) effective for HW faults in single-threaded apps

⇒ High coverage with low SDC rate (dedicated poster)

But multicore systems w/ multithreaded apps here to stay

Does the SWAT approach work for multicore?

**MSWAT: Diagnosis Challenges and Approaches**

Isolating the faulty core

Naïve solution: One spare core

High overhead, single point of failure

Our solution:

Multithreaded applications

**Key Ideas**

Isolated deterministic replay

Emulated TMR

**MSWAT Fault Detection**

Symptom Detection

Fatal Traps, Hangs, High OS,

Kernel Panic, No-Forward-Progress

Key Results

Low SDC rate of 0.2% of injected faults

Several detections from fault-free cores

**MSWAT Fault Diagnosis Algorithm**

Symptom detected

Capture fault activating trace

Re-execute captured trace

Look for divergence

Faulty core

Example

Faulty core is B

Capture fault activating trace

Native execution ⇒ No added support for replay

Record inputs to each thread (loads) for replay

Low hardware overhead for buffering

Re-Execute Captured Trace

Firmware emulated isolated deterministic replay ⇒ Zero hardware overhead

Compare retiring mem/ctrl instructions for divergence ⇒ Fewer comparisons

Iterative Diagnosis to reduce overheads

E.g., capture replay every 100k instructions till divergence

**Diagnosis Results**

>95% of detected faults successfully diagnosed

µarch non-determinism ⇒ undisagnosed faults

97% faults diagnosed in <10M cycles

<10ms on a 1GHz processor ⇒ invisible

93% diagnosed in 1 iteration w/ 100K instructions

<200KB logs ⇒ fit in lower level caches

**Conclusions and Future Work**

SWAT detection effective even for multicore systems with multithreaded apps

Novel diagnosis mechanism with minimal hardware changes

Ongoing and Future Work

Prototyping SWAT on FPGA in collaboration with University of Michigan

Distributed client/server applications

Faults in off-core components