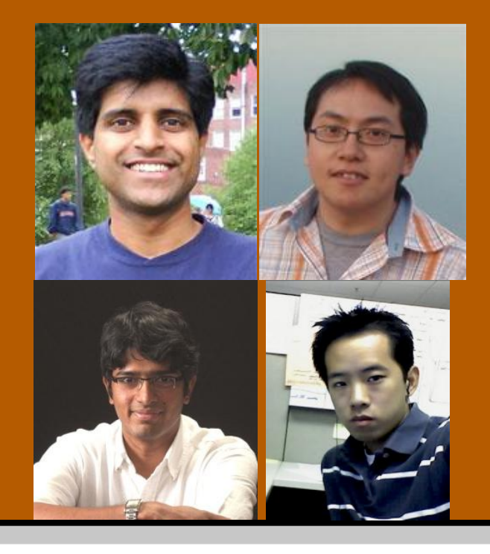
MSWAT: Low-Cost Hardware Fault Detection and Diagnosis for Multicore

Siva Kumar Sastry Hari, Manlap Li, Pradeep Ramachandran, Byn Choi and Sarita Adve
University of Illinois, Urbana-Champaign



Resilient Theme, Task # 1.2.2.5

GSRC Annual
Symposium
Sep 3-4, 2009

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?

Key Challenge: Cross-Core Fault Propagation

Multithreaded apps share data across threads

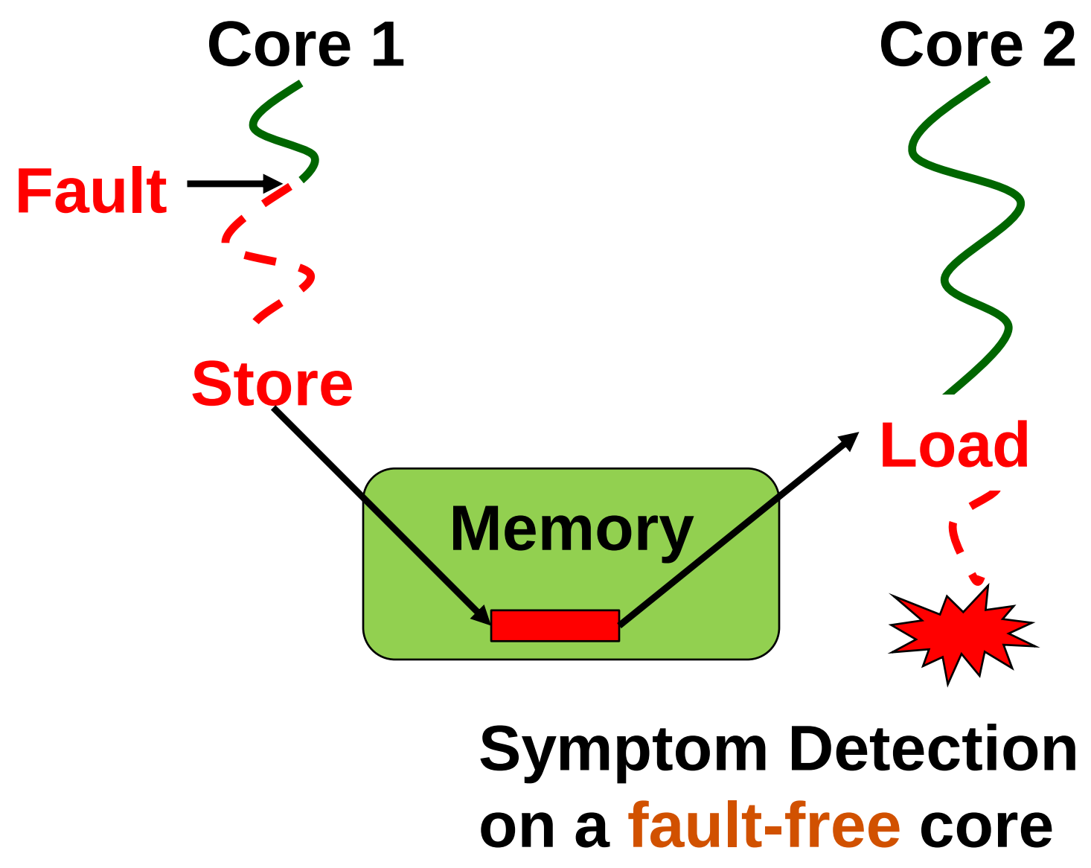
⇒ Fault may propagate across cores

⇒ Is SWAT effective in detecting these faults?

⇒ Symptom causing core is no longer faulty

Implicit assumption in prior SWAT work

Need to detect fault and diagnose faulty core



MSWAT Fault Detection

Symptom Detection

Fatal Traps, Hangs, High OS,

Kernel Panic, No-Forward-Progress

Key Results

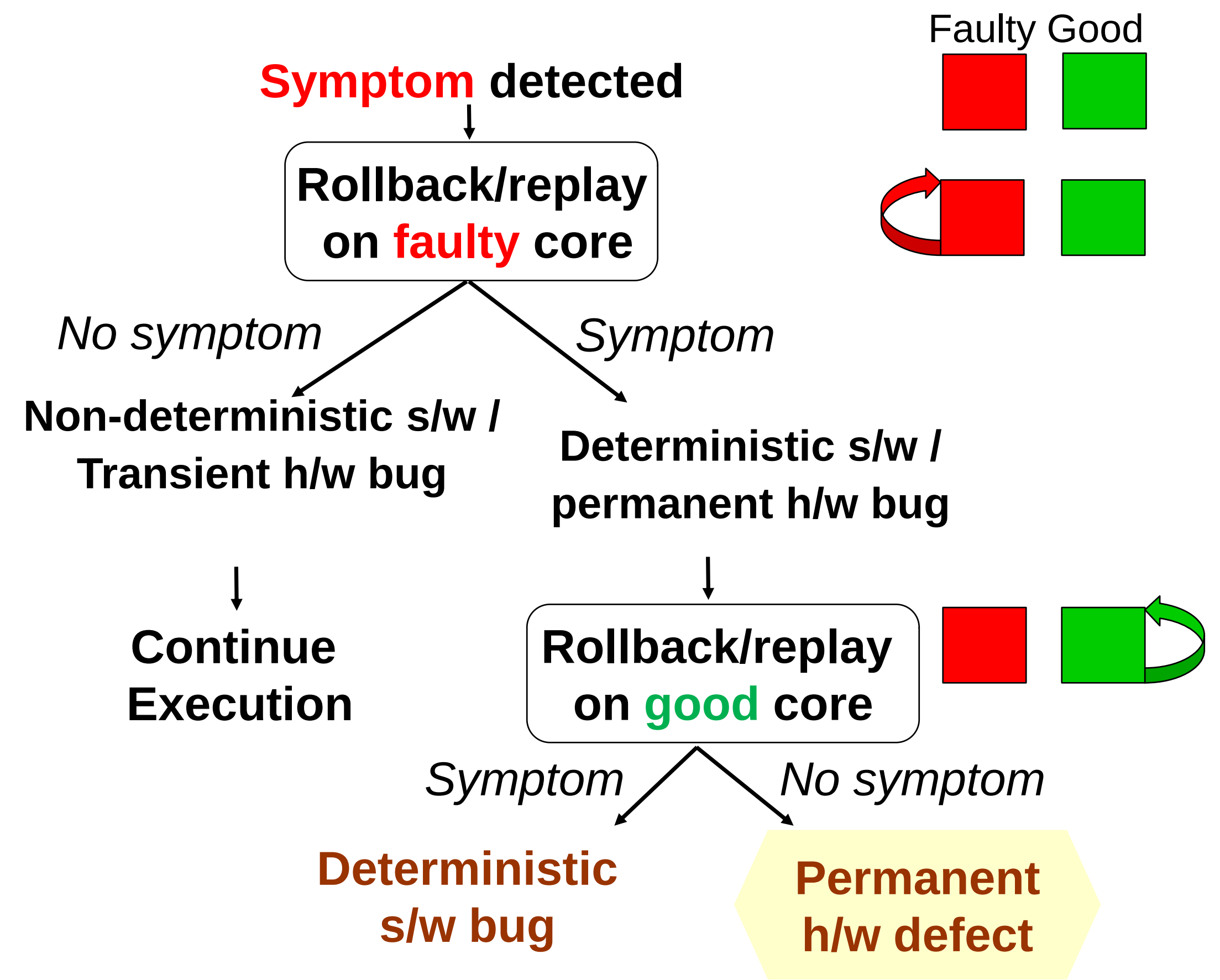
Low SDC rate of 0.2% of injected faults

Several detections from fault-free cores

MSWAT: Diagnosis Challenges and Approaches

Previous SWAT diagnosis

Distinguish HW/SW faults

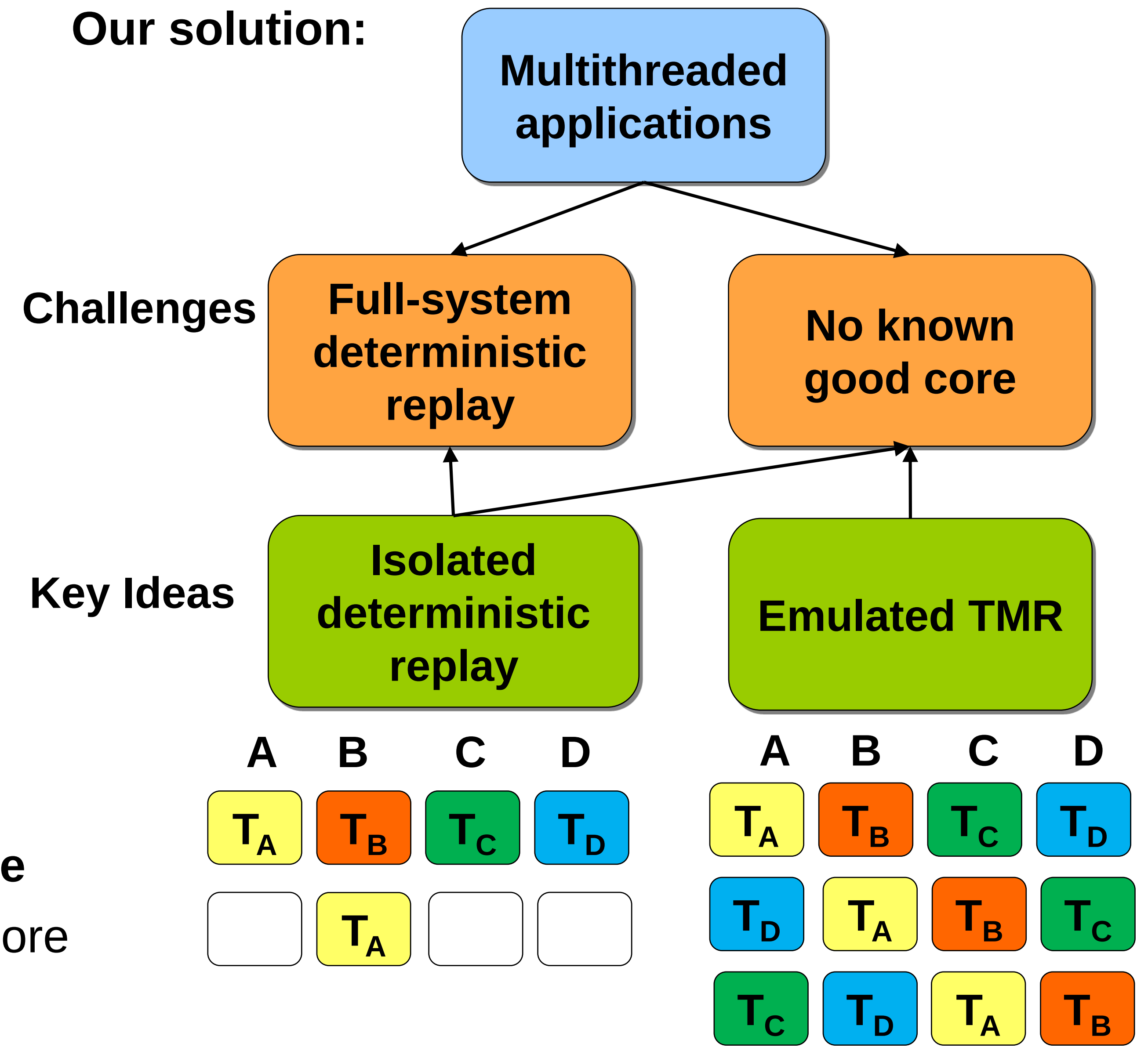

Faulty Good
Symptom detected
Rollback/replay on faulty core
No symptom
Symptom
Non-deterministic s/w / Transient h/w bug
Deterministic s/w / permanent h/w bug
Continue Execution
Rollback/replay on good core
Symptom
No symptom
Deterministic s/w bug
Permanent h/w defect

Isolating the faulty core

Naïve solution: One spare core

High overhead, single point of failure

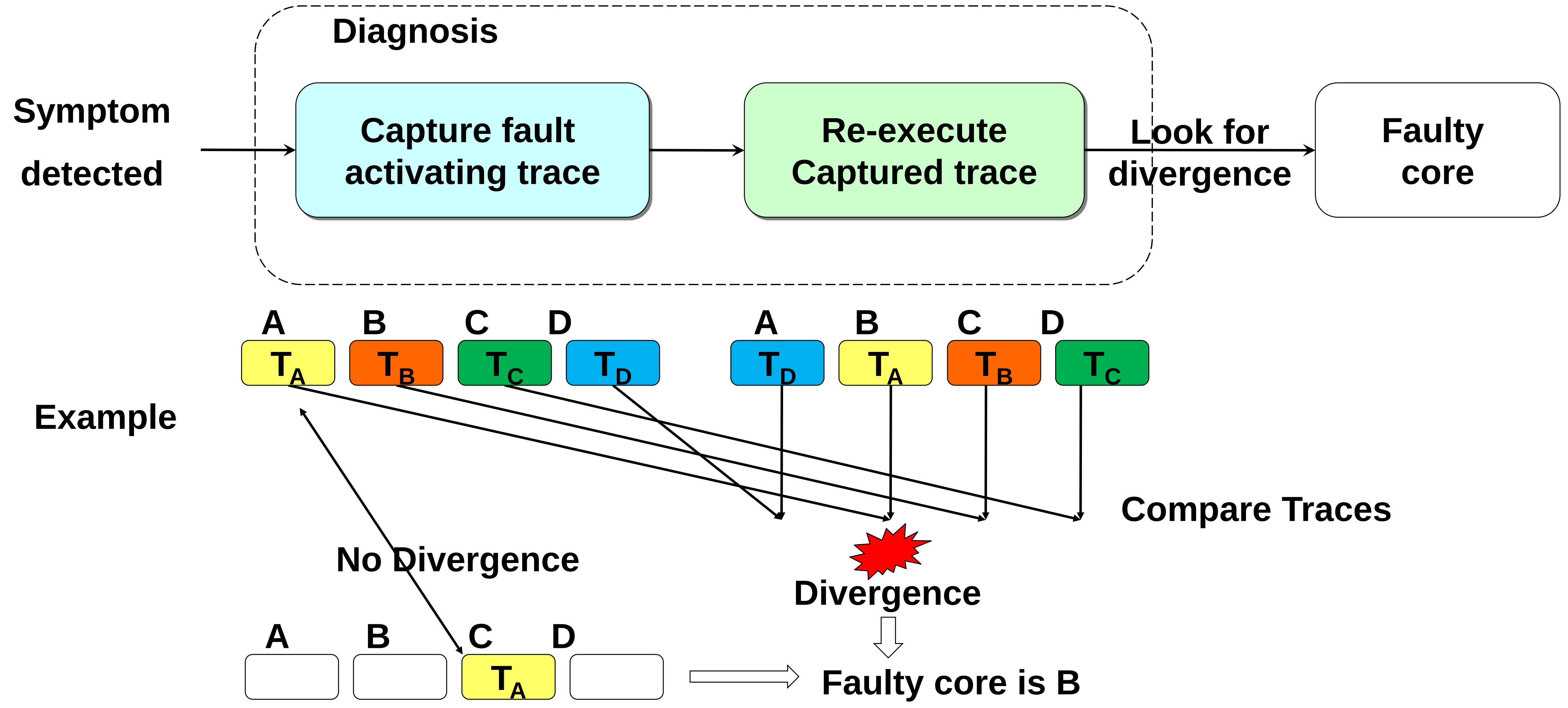
Our solution:



Challenge in multicore: No known good core

MSWAT: Diagnosis without known good core

MSWAT Fault Diagnosis Algorithm



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

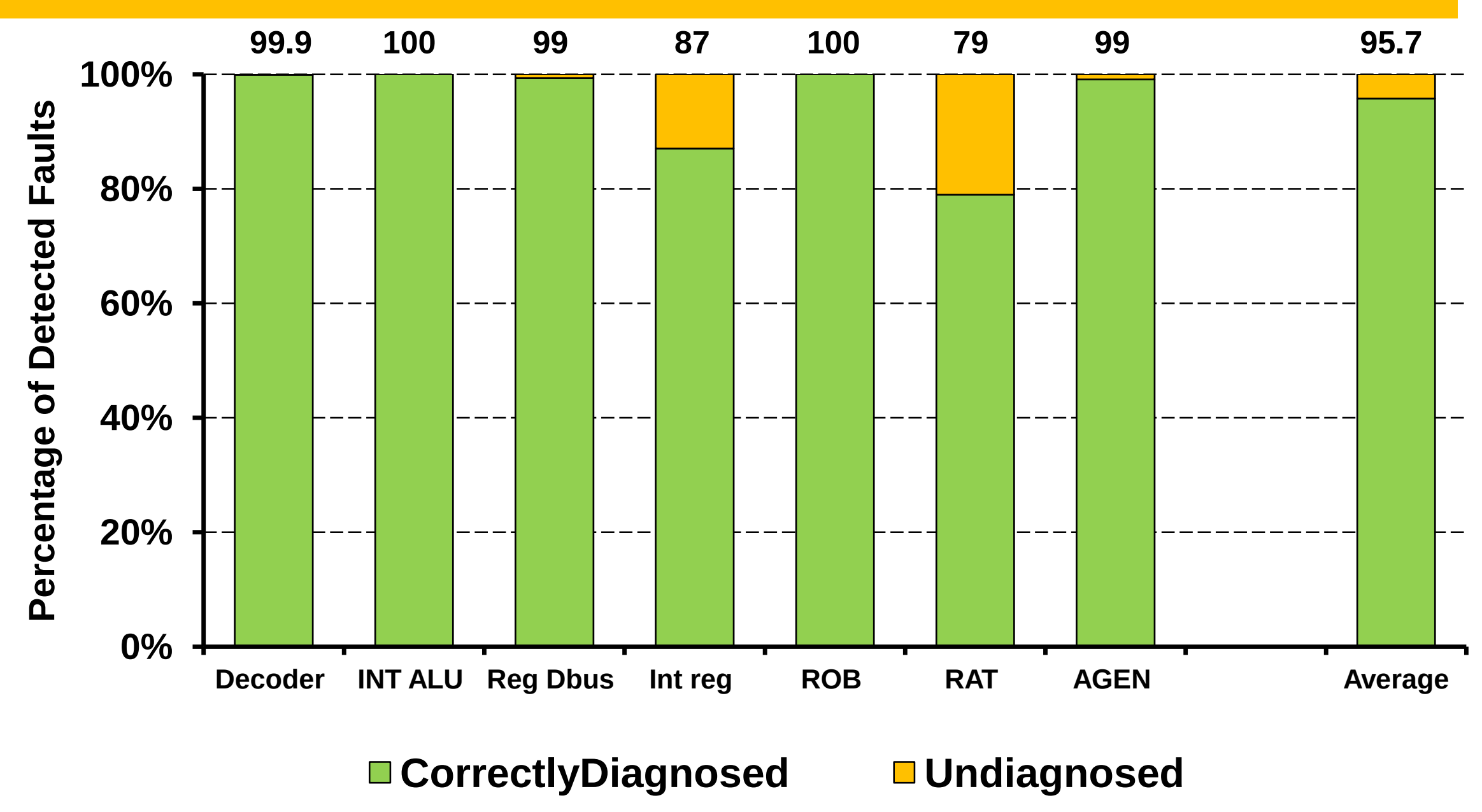
μarch non-determinism ⇒ undiagnosed faults

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