SWAT: SoftWare Anomaly Treatment

Manlap Li, Pradeep Ramachandran, Swarup Sahoo, Siva Kumar Sastry Hari, Sarita Adve, Vikram Adve

University of Illinois, Urbana Champaign

Resilient Theme, Task #

GSRC Annual Symposium

Sep 29-30, 2008

Motivation

Technology scaling ⇒ increased sources of failure

Wear-out, infant mortality, design defects, etc.

Hardware will fail in-field

Need for on-the-fly detection, diagnosis and recovery/repair

Key Observations

Handle only h/w faults that propagate to s/w

Fault-free cases are common, must be optimized

Strategy

Watch for software anomaly (symptoms)

Zero to low overhead "always-on" monitors

Diagnosis after symptom detected

May incur high overhead, but rarely invoked

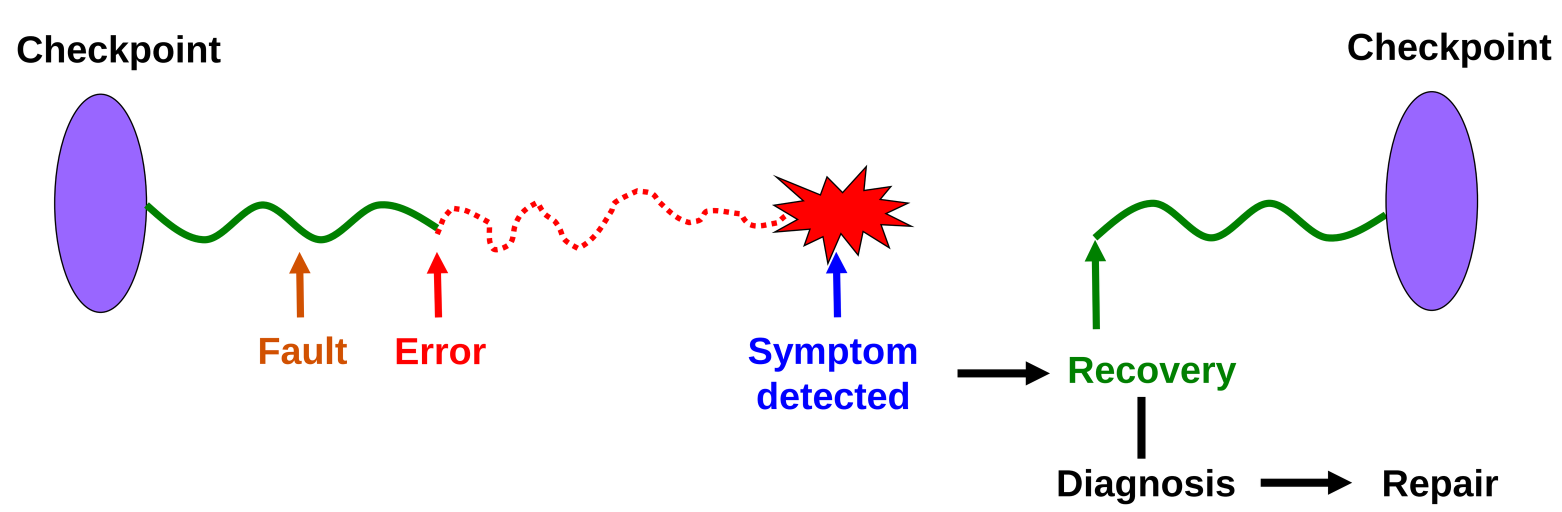
SWAT: SoftWare Anomaly Treatment

Detection: Software symptoms, minimal backup hardware

Recovery: Software/hardware checkpoint and rollback

Diagnosis: Firmware-controlled rollback/replay on multicore

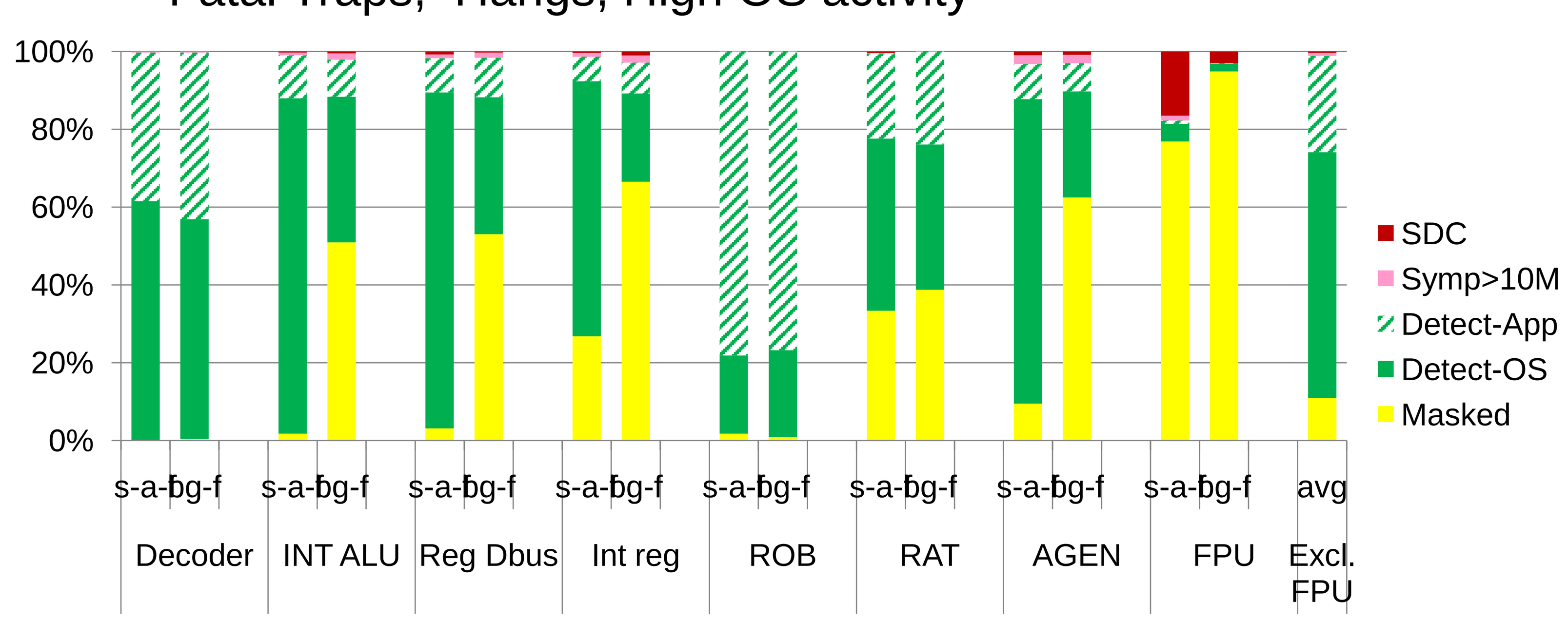
Repair/reconfiguration: Redundant, reconfigurable hardware



Detection [ASPLOS'08]

Low-cost monitoring of software symptoms

Fatal Traps, Hangs, High OS activity



Key Results:

(1) 98% of unmasked are (w/o FPU) detected in 10M instructions

(2) 0.38% of injected faults result in SDC

(3) Many faults corrupt OS state

Recoverable with hardware checkpointing

Invariant Based Detection [DSN'08]

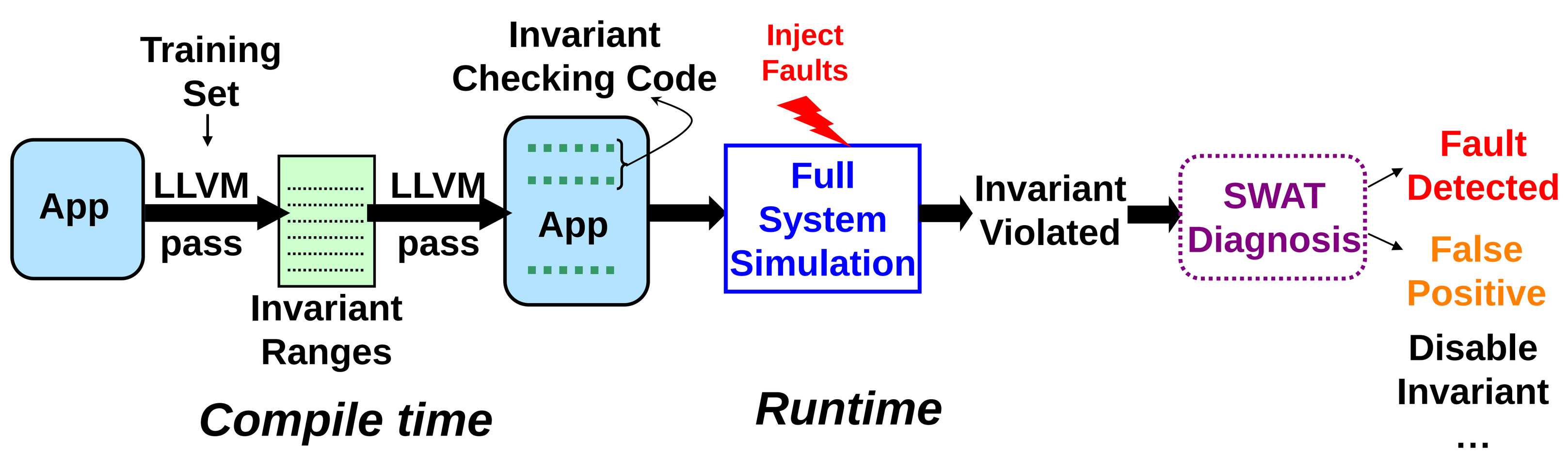
Detection with software reliability techniques

Observation: Undetected faults corrupt data values

Strategy:

(1) Use likely invariants to check data values (*MIN≤value≤MAX*)

(2) Disable invariant when diagnosed as false-positive

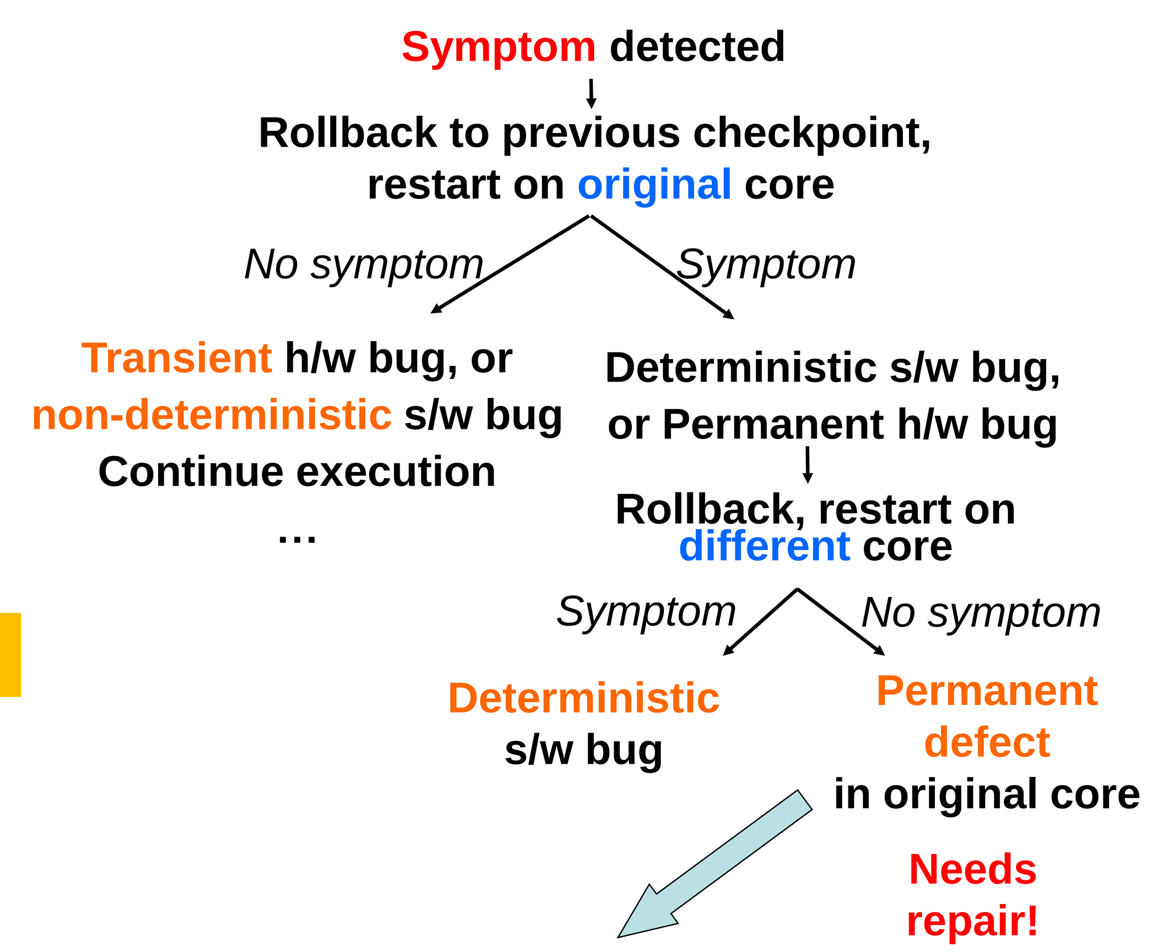


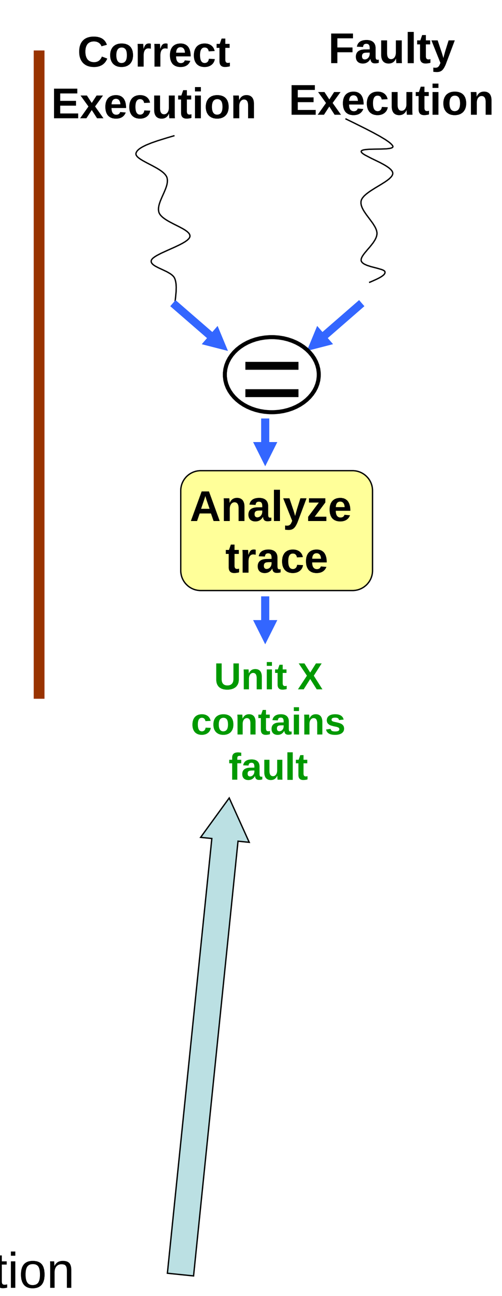
Key Results:

(1) Undetected faults reduced by 28% and SDCs by 74%

(2) Low run-time overhead (5% on x86, 14% on UltraSPARC IIIi)

Diagnosis [DSN'08]





TBFD: Trace-Based Fault Diagnosis

Key Observations

Use reconfigurable units for repair

Fault causes divergence from correct execution

Strategy

(1) Compare faulty and fault-free instruction traces

(2) Mismatches give clues for tracking down faulty hardware

Key Results:

(1) 98% of detected faults can be diagnosed

(2) Logical-physical register mapping faults harder to diagnose

Recovery

Hardware checkpointing effective for recovery

86% of detection that corrupts software recoverable

Virtually all corrupted OS state recoverable

Ongoing work

OS/App-aware checkpointing techniques

Tolerate longer detection latency

Hardware checkpointing to recovery OS state

Firmware/OS-level checkpointing for App

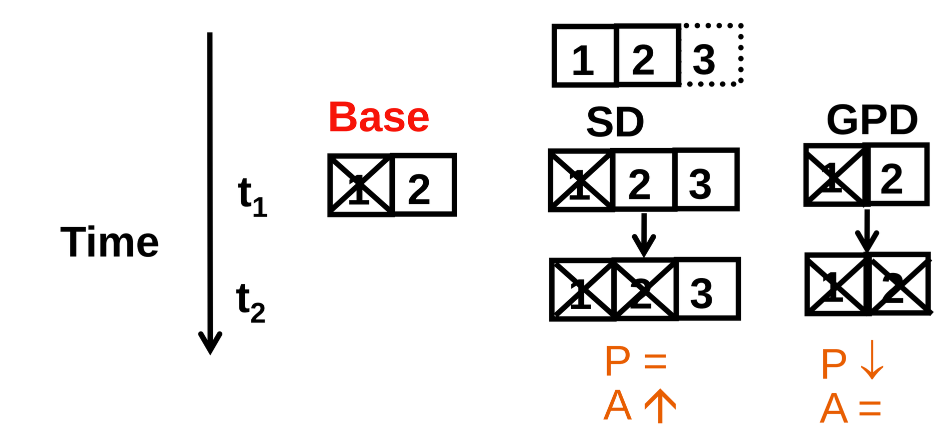
Repair and Configuration

Fine-grain reconfiguration improves processor lifetime [ISCA'05]

Structural Duplication (SD)

Cold spares that operate when component fails

Graceful Performance Degradation (GPD)



SWAT on Multicore Systems (Ongoing work)

Additional Multicore detection Symptoms

No Forward Progress – No core retiring app instructions

CPU Panic more prominent in multi-threaded apps

Key Results:

Coverage: 98.75% (w/o FPU), SDC: 0.6%

Insights:

20% of injected faults affect Faulty-Free Core execution

Most of the faults are propagated through OS

Longer OS detection latencies

Need to better understand the fault manifestation in OS