The Granularity of Soft-Error Containment in Shared Memory Multiprocessors

TRUSS Brian T. Gold, Jared C. Smolens, Babak Falsafi, and James C. Hoe – Computer Architecture Lab @ Carnegie Mellon (CALCM)

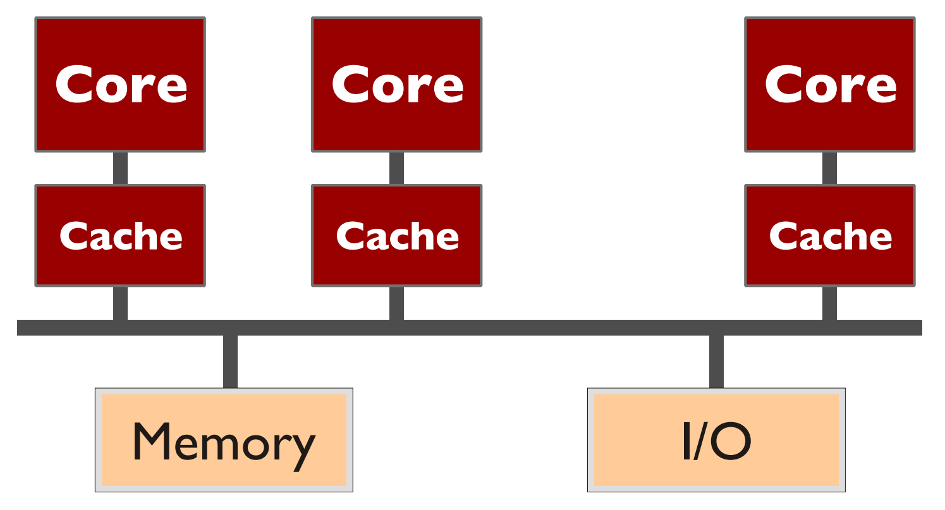
Introduction

Soft-error rates (SER) increasing exponentially

What are the key sources?
- **Radiation:** SER scales with transistor count
- **Variability:** SER increases with level of integration
  - – Manufacturing: device variations within/across dies
  - – Lifetime: transistor performance varies over time

Processor soft errors pose challenging problems
- – Spurious, hidden errors (hard to detect)
- – Complex, timing-critical datapath (ECC infeasible)

Shared-memory multiprocessors particularly vulnerable
- ➔ Error in one core can affect entire system



Generic shared-memory multiprocessor assumed in this work

Key problem: When to detect/recover?
- – Determines how far errors propagate
- – Affects complexity of detection and recovery

Containment boundary

Portion of system where errors originating in processor may propagate before detection. Finer-grained containment keeps errors closer to the processor core, while coarser-grained containment allows errors to affect more of the system.

Range of containment boundaries
- – **Core:** Before instruction retirement
- – **Cache:** At shared memory interaction
- – **Memory:** Before performing device I/O

↑Finer ↓Coarser

Tradeoff complexity in core and complexity of recovery

Background

Dual-Modular Redundancy (DMR)
- – Execute two copies of program
- – Compare results to detect errors

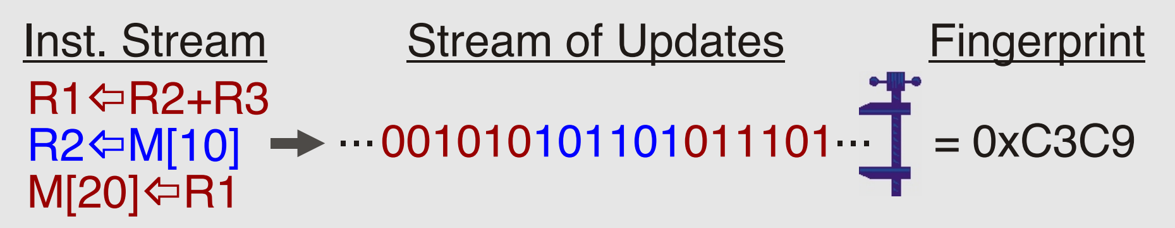
Backwards Error Recovery
- – Create checkpoints of correct system state
- – Roll back state to checkpoint on error detection

Checkpointing – key questions:
- – How much system state must be checkpointed?
- – How often must a new checkpoint be created?

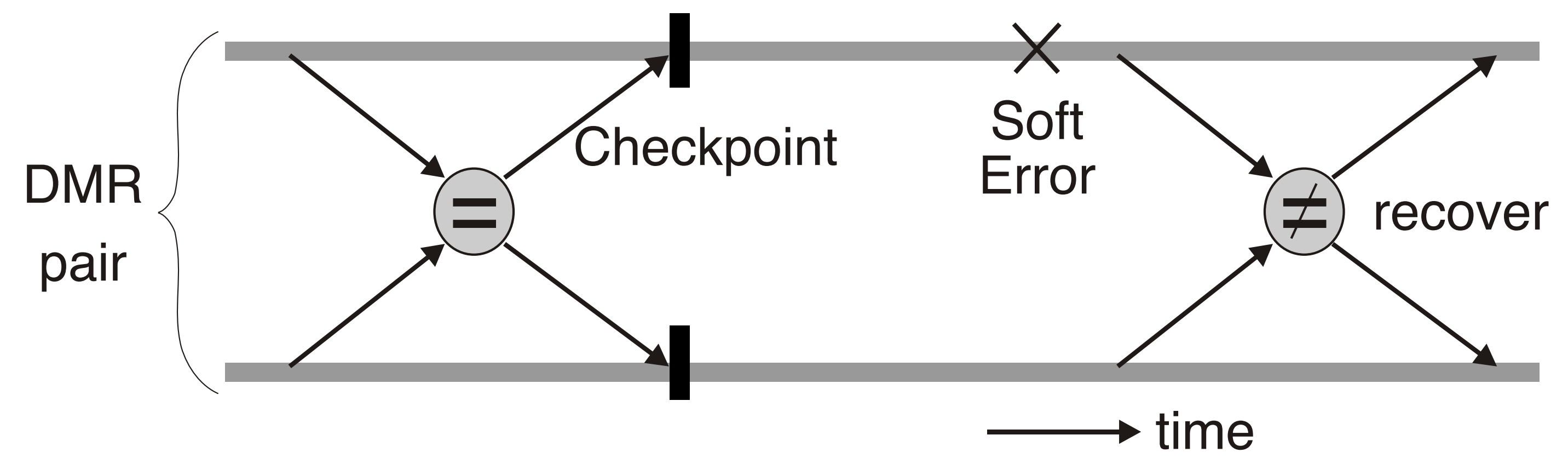
Fingerprinting [Smolens, ASPLOS '04]

Cumulative hash of processor architectural state updates–register and memory values

State comparison across redundant processors
- – Similar coverage to comparing all instructions
- – Low inter-processor bandwidth
- – Triggered on demand



Enables flexibility in *when* to detect



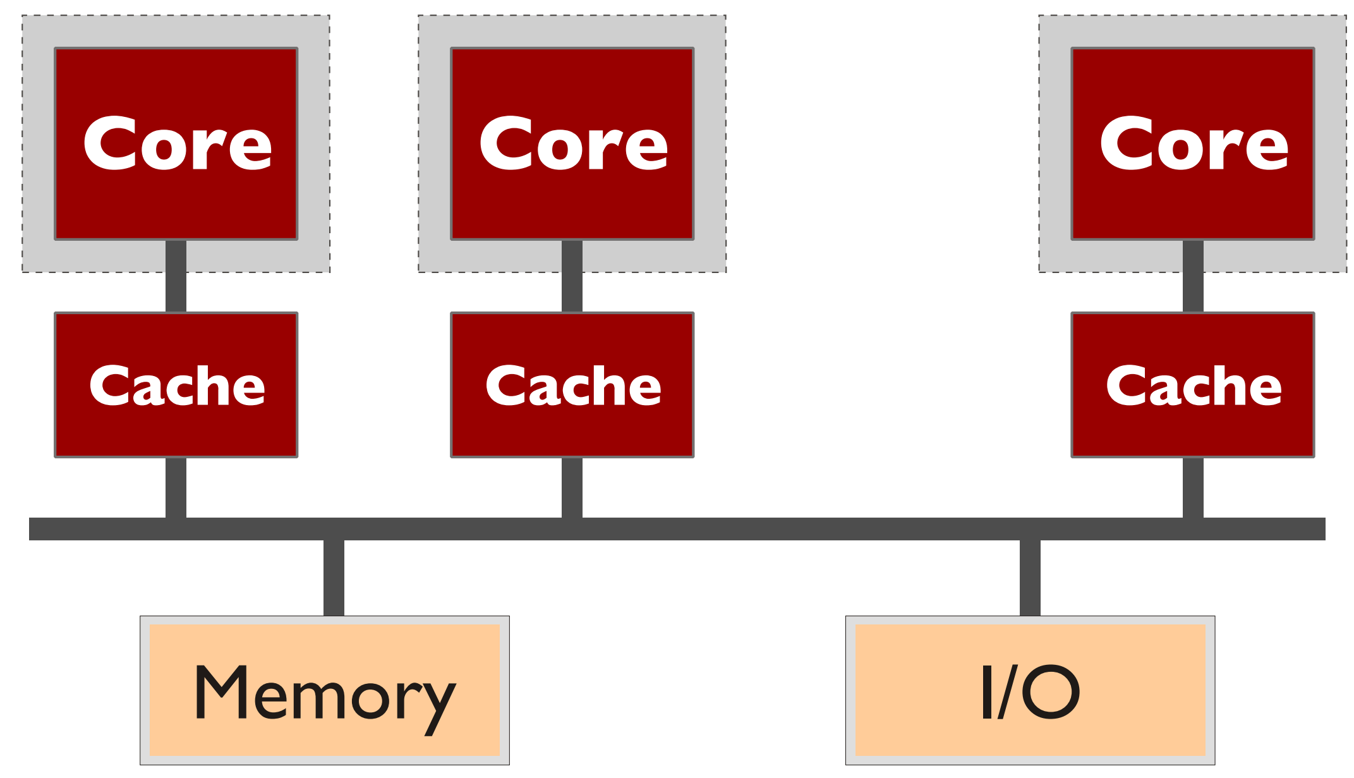
Core Containment

Errors detected/corrected before exiting core

Trigger Events:
- Stores to cacheable memory
- Accesses to peripheral devices

Advantages
- + Localizes error recovery
- + Avoids changes to memory system

Disadvantages
- Complex changes to pipeline front-end
- Error detection added to retirement stage(s)



Case Study: IBM z-series

The IBM z-series mainframe uses a custom processor core that consists of two lockstepped, replicated pipelines. Before retiring an instruction, the results from both executions are compared, and if an error is detected, the instruction is re-executed.

Case Study: Redundant Multithreading

Recent work suggests that time-delayed redundant execution using SMT or CMP architectures can improve resource efficiency over lockstepped DMR. In these proposals, error detection is enforced either before the register file or before the cache. Both satisfy the core containment granularity.

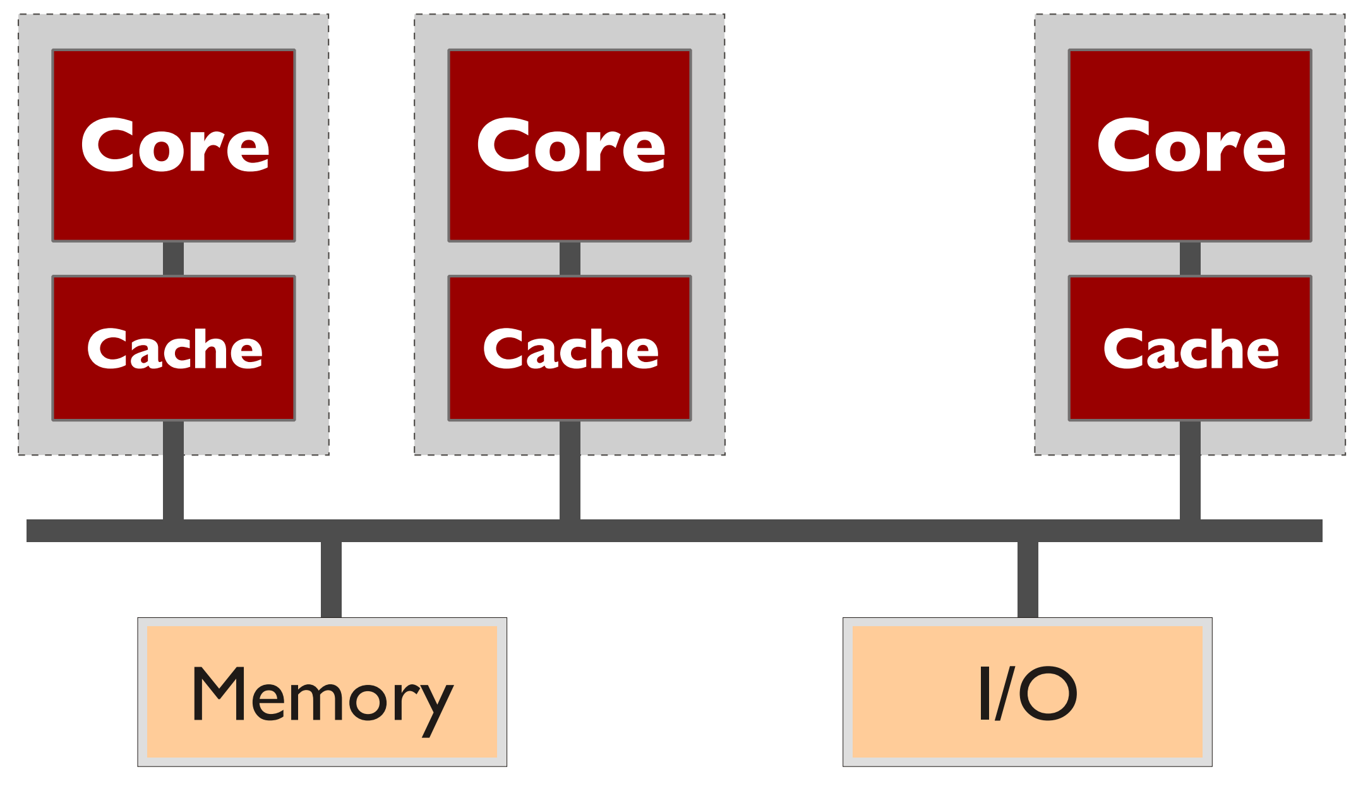
Cache Containment

Errors isolated to core and local caches

Trigger Events:
- Shared-memory interactions (e.g., invalidations)
- Writebacks to main memory
- Accesses to peripheral devices

Advantages
- + Reduces complexity of core changes
- + Avoids coordination of checkpoints among cores

Disadvantage
- Requires checkpoints of cache state



Case Study: TRUSS

The TRUSS server architecture provides a logical separation layer (Membrane) that enables processor cores and caches to locally checkpoint and recover, independent of other processors or memory. Each processor core and cache form a containment boundary such that all shared-memory interactions are guaranteed error-free. With hardware support for checkpointing and error detection, the TRUSS architecture provides software-transparent reliability for soft errors and a broad class of permanent faults.

Other Examples: HP NonStop, Stratus, etc.

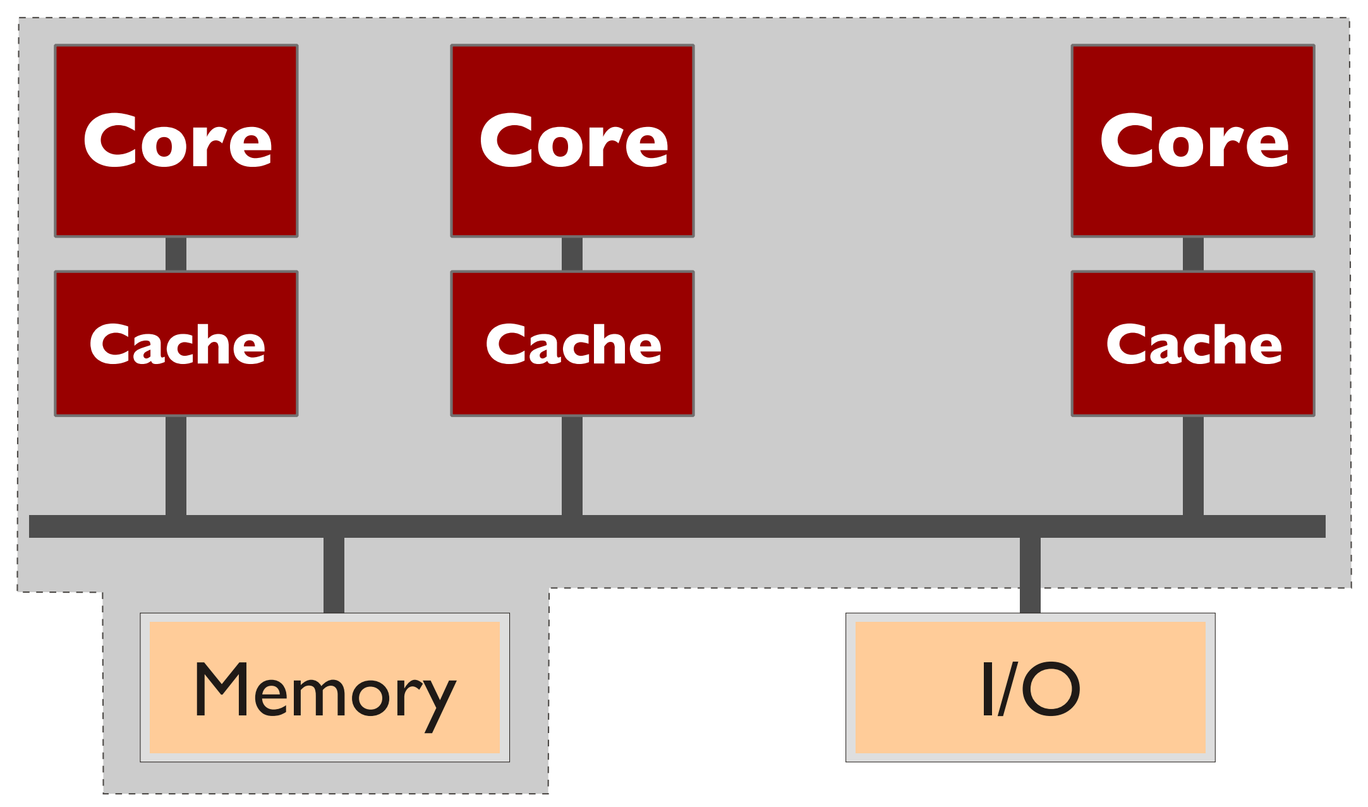
Memory Containment

Errors detected/corrected before commit to I/O

Trigger Events:
- Irrevocable, non-idempotent device I/O

Advantages
- + Minimizes changes to processor core
- + Checkpointing less often (lower overhead)

Disadvantages
- Requires coordinated, system-level checkpoint
- Precludes frequent, asynchronous checkpoints



Case Study: HP NSAA

HP's NonStop Advanced Architecture (NSAA), although not a shared-memory multiprocessor, uses the memory containment granularity. Before performing disk or network I/O, NSAA compares redundant executions. Recovery is accomplished by reverting to a software-created backup process.

Recovery Across I/O

When coordinating checkpoints across all processors, a major challenge becomes the apparent need to recover across I/O. Nakano et al. observe that some common I/O operations are idempotent, which permits coordinated checkpoints for I/O intensive workloads (e.g., TPC-C or HTTP servers).