SafetyNet:

improving the availability of shared memory multiprocessors with global checkpoint/recovery

Daniel J. Sorin, Milo M. K. Martin, Mark D. Hill, David A. Wood

In Proceedings of the 29th Annual International Symposium on Computer Architecture (ISCA) 2002.

Henry Cook CS258 4/7/2008

Goals

- Create a system-wide, lightweight checkpoint and recovery mechanism
- Provide globally consistent logical checkpoints
- Have low runtime overhead
- Prevent crashes in the face of hard or soft errors
- Decouple recovery from detection

System Overview

QuickTime™ and a decompressor are needed to see this picture.

QuickTime™ and a decompressor are needed to see this picture.

Challenge 1

- Saving every update, write, or response is expensive
  - Checkpoint at coarse granularity (100K)
  - Only log the first such action per checkpoint

Challenge 2

- All procs, caches, and mems must recover to a consistent point
  - Global logical time
  - Logically atomic coherence transactions
    - Point of atomicity
  - Avoid checkpointing transient state or in flight messages by waiting for transactions to complete

Challenge 2 - Global logical time

- Broadcast/snooping: count number of coherence requests received
- Distribute perfectly synchronous physical clock
- Distribute loosely synchronized checkpoint clock
  - Valid base if skew < communication time between nodes

Challenge 2 - Transactions

1. Processor requests block B
2. Memory processes request
3. Cp#2-5 not validated until transaction completes

QuickTime™ and a decompressor are needed to see this picture.

Challenge 3 - Validation

- Validate only once all previous points are validated
- Each component must declare it has received fault-free responses to all reqs
- Validation latency dependent on fault detection latency

Challenge 3

- SafetyNet must advance recovery point
  - Pipeline checkpoint validation off of the critical path
  - Hide latency of fault detection mechanisms
    - Continue execution even if detection is a long latency mechanism

Recovery

- If recovery point cannot be advanced for a given amount of time, error must have occurred preventing message delivery
- State is rolled back or restored
- In-flight transactions are discarded
- Restart message is broadcast when recovery (and reconfiguration) completes

Implementation

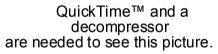
- Checkpoint Log Buffer logs stored state
  - Add CN to blocks, log update if CCN $\geq$ CN
- Shadow registers hold reg checkpoints
- Service processors coordinate recovery

QuickTime™ and a
decompressor
are needed to see this picture.

Evaluation

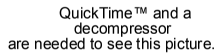
- Hard or soft faults
  - Dropped message, failed switch
- Multiple benchmarks
  - OLTP, SPECjbb, Apache, dynamic web service, SPASH scientific
- Simulate 16 proc system with Simics
  - 100 cycle register checkpoint, 8 cycle store logging, 100K checkpoint interval

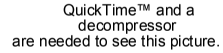
Performance



- Insignificant difference for fault-free
- No crash on faults
- Energy efficiency?

Sensitivity





- Stores requiring log entry decrease as checkpoint interval decreases
- CLB size is dependent on interval and program behavior, not cache size

Generalizing

- SafetyNet can recover from any fault where:
  - A mechanism in the system can detect the fault (or its absence)
  - Faults are detected while a recovery point is still being maintained