Parallel Assertion Processing using Memory Snapshots

Junaid Haroon Siddiqui
Muhammad Faisal Iqbal
Derek Chiou

UCAS5
26 April 2009

Motivation

- Importance of Assertions
- Parallel Assertion Processing
- Memory Snapshots

Importance of Assertions

- Assertions are important in catching bugs early
- Some assertions are costly like validating class invariants
  - Consider validating a binary tree
- Limited use due to Performance penalty and serial nature

Parallel Assertion Processing

- We propose costly assertions be executed in parallel
- Potentially increase efficiency
- Two issues
  - Retaining state of memory for assertion
  - Programmer's view of assertions

Memory Snapshots

- We propose memory snapshots
  - Low cost snapshot at assertion point
  - Maintain snapshot with copy-on-write
  - Performance benefits depend on workload characteristics

Outline

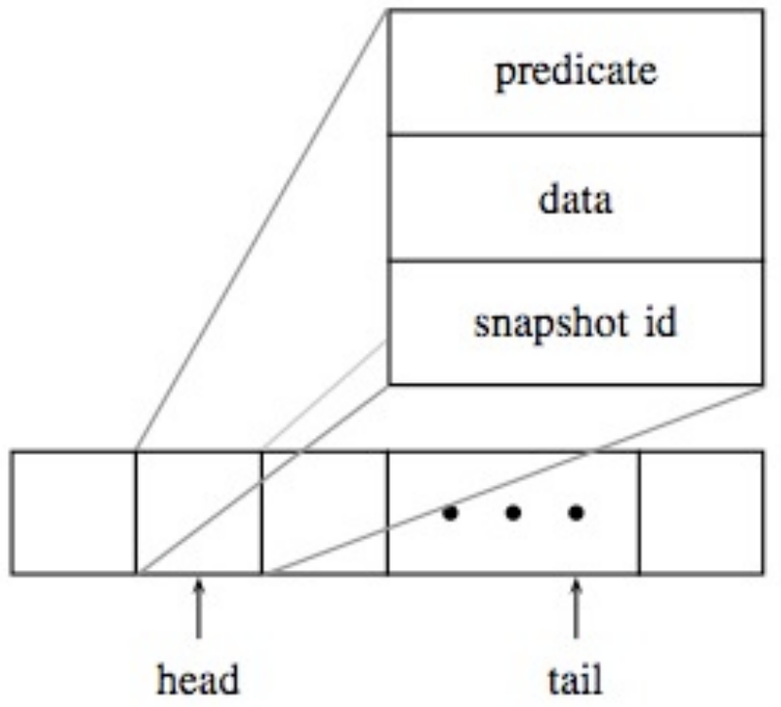
- Parallel Assertion Processing
- User Interface to Memory Snapshots
- Implementation of Memory Snapshots
- Evaluation & Results
- Future Work & Conclusions

Parallel Assertion Processing

- Assertion Task Queue
- Steps at Assertion Point
- Steps at Assertion Task Processing
- Processor Allocation for Assertion Task

Assertion Task Queue

- Fixed Size Circular Buffer
- Predicate is the assertion routine
- Data is an address passed to that routine
- Snapshot id identifies the snapshot to be used



Steps at Assertion Point

- Take Snapshot
  - Blocking?
  - Optional & Compulsory Assertions
- Enqueue Assertion Task

Steps at Assertion Task Processing

- Dequeue Assertion Task
- Evaluate Assertion
- Possible Halt
  - Rollback?
- Release Snapshot

Processor Allocation for Assertion Task

- Many models exist once assertion and its processing are decoupled
  - Dedicated threads with Processor *affinity*
  - Low priority task in global work queue
  - Event handler serving own assertions after signaling completion

User Interface to Memory Snapshots

- Accessing Snapshots by Memory Mapping
- Accessing Snapshots by Virtual Address
- Accessing Snapshots from a Software Library

Accessing Snapshots by Memory Mapping

- System call like *mmap*
  `void* mmap_snapshot(void* p, size_t len)`
- Flexibility of smaller snapshots
- Practically need snapshot of whole heap
- Overhead of kernel setting up page tables for copy-on-write

Accessing Snapshot by Virtual Address bits

- Reserve bits for snapshot identifier
  - No unused bits even in 64-bit processors
  - Using valid bits means OS has to change address space model
  - Kernel can setup these at fault time

Accessing Snapshots from a Software Library

- Added cost on software developer
- High availability being in software
- Snapshot Manager
  - Modify Notifications
- Assertion Queue

```
class snapshot_manager {
  snapshot_id take(bool block);
  void release(snapshot_id id);
  void modify_notification(void* p, size_t len);
  void* read_ptr(void* p, snapshot_id id);
};
class assertion_queue {
  assertion_queue(snapshot_manager& sm);
  bool queue(predicate_type pred, void* data,
bool block);
  bool process(bool block);
};
```

Implementation of Memory Snapshots

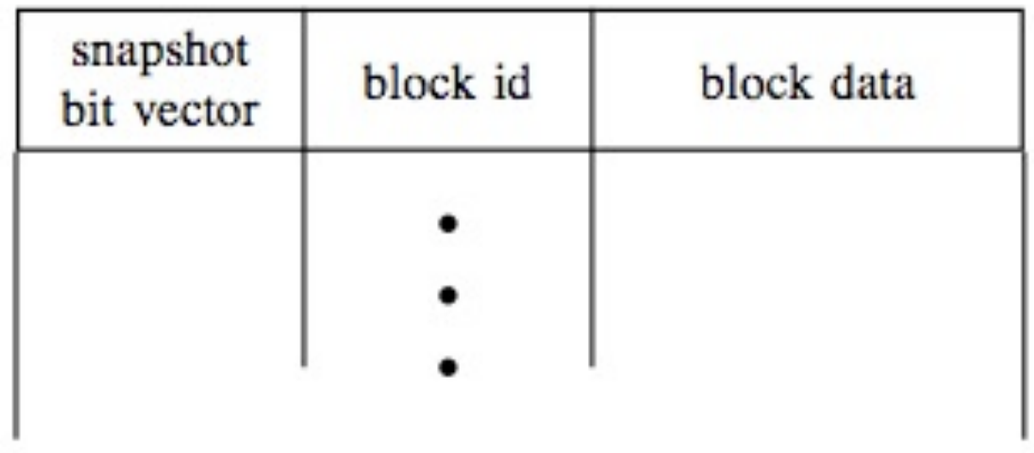
- Snapshots of Physical Address Space
- Snapshots of Virtual Address Space
- Snapshots User Library

Snapshots of Physical Address Space

- Use physical address space bits
- Practical for 64-bit machines
- Snapshot register to identify active snapshots
- Divide memory in blocks for copy-on-write
- Snapshot Cache for historical copies

Snapshot Cache

- When snapshot taken
- When memory is written
- When snapshot released



Snapshot Aware Cache

- Performance advantage of cache support
- Cache aware of snapshot identifier bits in physical address
- Every cache has its own snapshot register
  - Broadcast changes to this register
- Every cache entry has a snapshot bit vector

Behavior of Snapshot Aware Cache

- When snapshot taken
- When data is read or written
- When data written, and other bits set
- When snapshot released

snapshot bit vector	tag	state	data
011101000	X	Shared	· · ·
000000001	X	Exclusive	· · ·

Physical Address Space Snapshot Issues

- Limited Snapshot cache
- Performance dependency between applications
- Issues caused by paging
- Applications truly interested in snapshots of virtual address space

Snapshots of Virtual Address Space

- Kernel runs the show
- All three user interfaces possible
- All based on setting up page table for copy-on-write operation
- Copy-on-write overhead
- Kernel support with software library?

Snapshots User Library

- Snapshot Manager
  - Snapshot cache in software map
    - Read request
    - Modify notifications
    - Snapshots taken and released
- Assertion Queue

Issues with Snapshots User Library

- Advantage is no change required in kernel or hardware
- Big performance overhead due to lookups
- Missing modify notification
- Hybrid software-kernel scheme?

Evaluation & Results

- Two schemes evaluated
  - Pure Software library with Software library interface
  - Snapshots of physical address space with no User interface

Software Library Approach

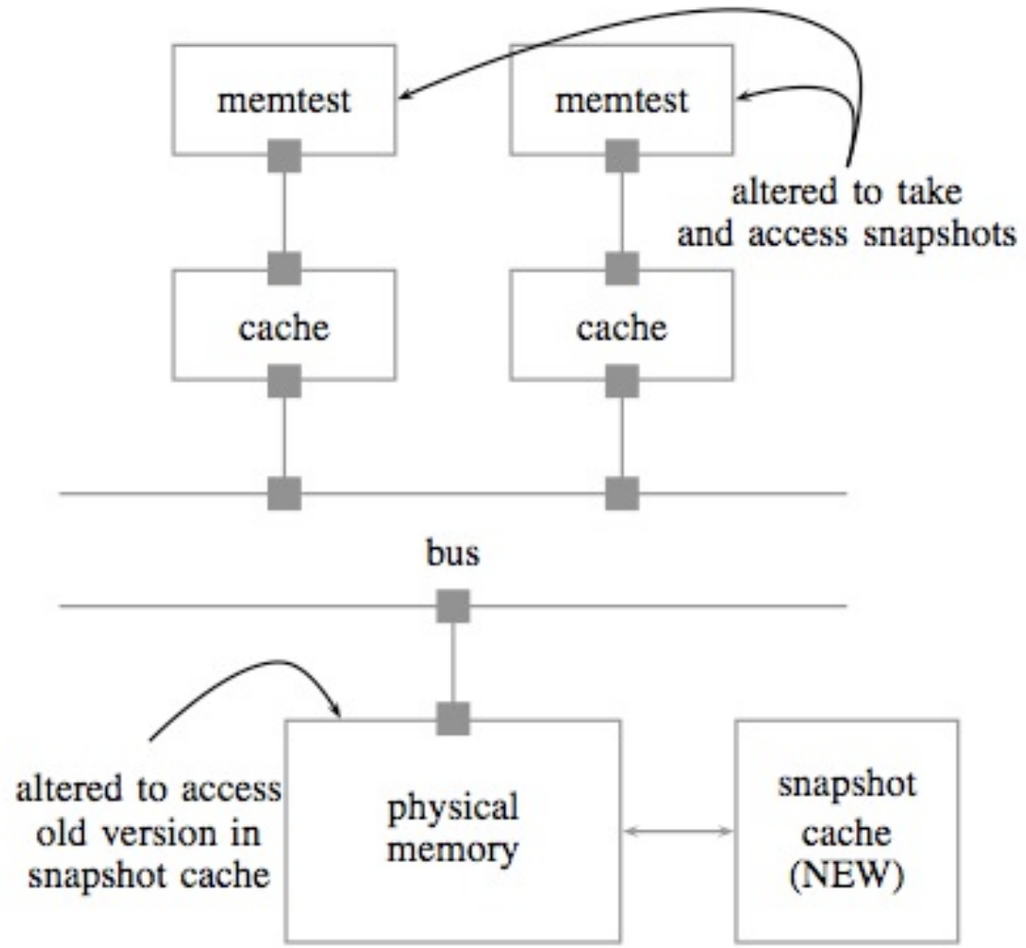
- Successful implementation
- Correctness verification done
- Performance results are disappointing
  - Nature of Application?

Physical Address Space Snapshots Approach

- Evaluated Gems but modifying functional behavior of memory *probably* not possible
- Evaluated PIN but changing every data accessed by an instruction *probably* not possible
- Evaluated M5 and it gave all the necessary flexibility

Physical Address Space Snapshots in M5

- Added a snapshot cache to physical memory
- Altered physical memory controller
- Altered *memtest* to take and access snapshots
- Correctness testing done but no Performance testing



Conclusions

- Extracting costly assertions from critical path can improve efficiency
- Exact performance benefit depends on many factors
- Our work most useful for high cost assertions and infrequent modifications
- Our works shows feasibility of this scheme

Future Work

- Workload characterization where different schemes are most useful
- Support for Rollback to a snapshot
  - Reverse Debugging?
  - Software Controlled Rollback?
- Multiple active versions of memory

Questions?