

VM Snapshots

Patrick Colp

Chris Matthews, Bill Aiello, Andrew Warfield

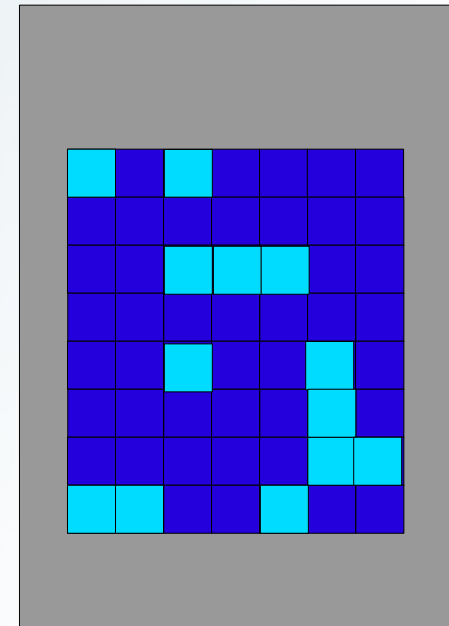
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada

(in collaboration with George S. Coker, II)



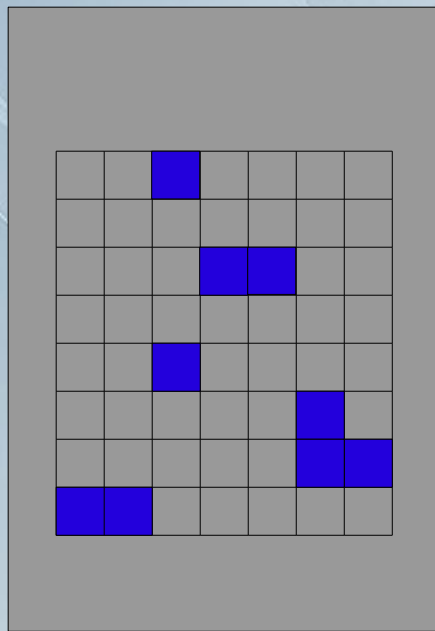
Problem

- Want a fixed image of a VM at a point in time
 - e.g. to dynamically inspect a VM from the outside
- VMs don't sit still
 - memory/state keeps changing

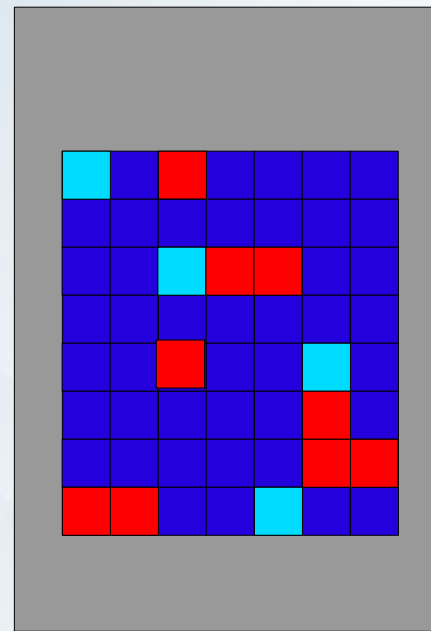


Goal

- Capture a consistent view of a VM
 - Without pausing and still allowing the VM to run
 - Create a lightweight, general purpose mechanism



Snapshot



Overview

- Copy-on-Write frames to snapshot images
 - Allows VM to keep running
 - Lightweight as only dirtied frames are copied
- General purpose mechanism
 - Can use CoW to do other things too!

```
snapshot = vm_snapshot(domid);
```



Degrees of Snapshots

1. Static view of guest memory

- VM introspection (XenAccess, LKIM)
- VM checkpointing (Remus)

2. Execution rollback

- VM undo
- Speculative execution (SpecHint)

3. VM fork

- Quickly spawn template VMs (Potemkin, SnowFlock)

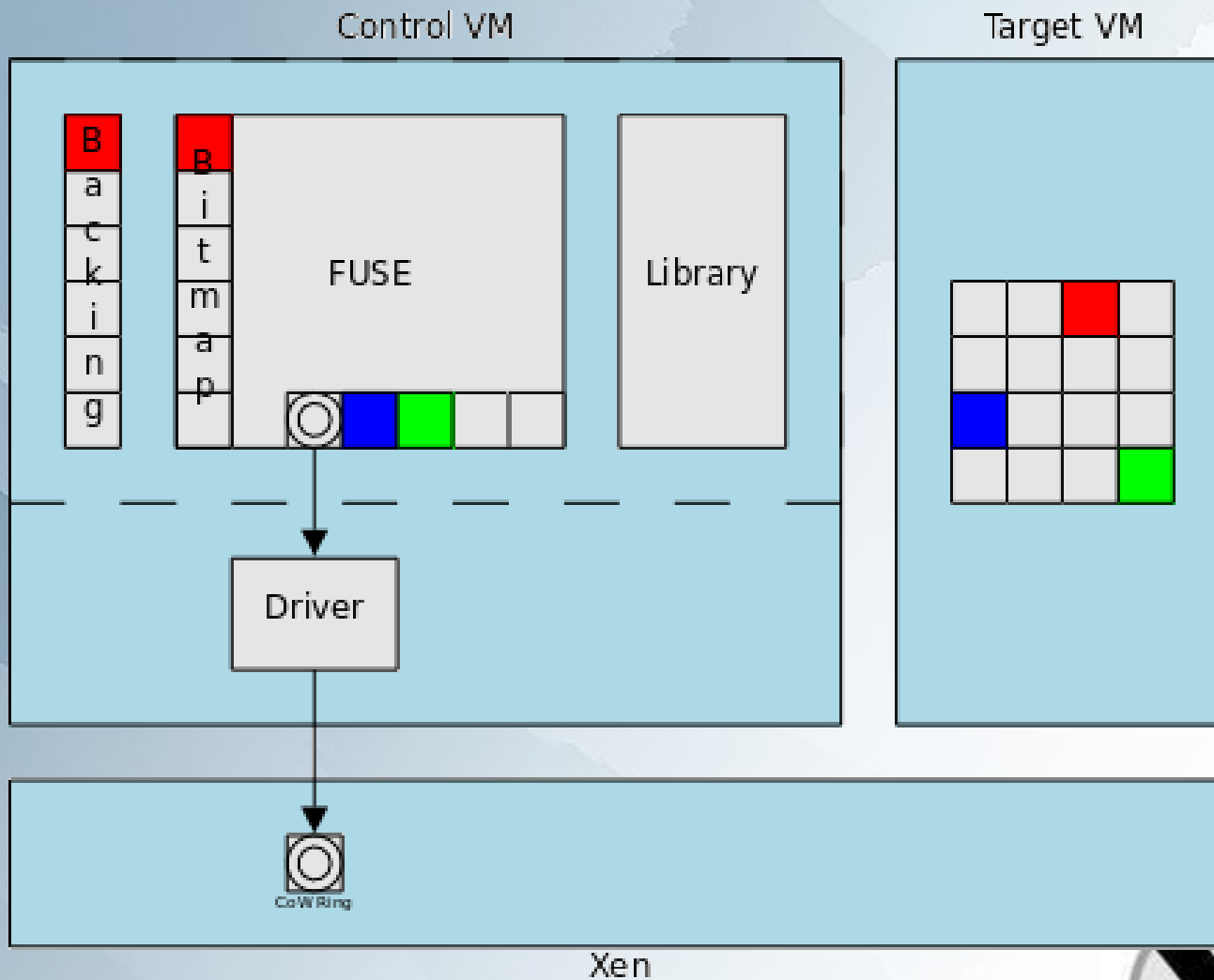


Implementation

- Save initial state
 - Registers, shared frames, *etc.*
- Copy-on-Write VM's memory
 - Mark memory read-only
 - Catch write faults (log dirty)
- What about disks?
 - Easy: use Parallax for disk snapshotting
 - Alternatively can use VHD/QCoW



Architecture



Architecture

- Library
 - Provide a nice API for snapshotting
 - Integrate with things like XenAccess and Remus
- FUSE driver
 - Allocate per snapshot buffer space
 - Copy buffered frames into backing file
 - Fetch frames from target VM as requested
 - Exposes snapshot through `read()` and `mmap()`

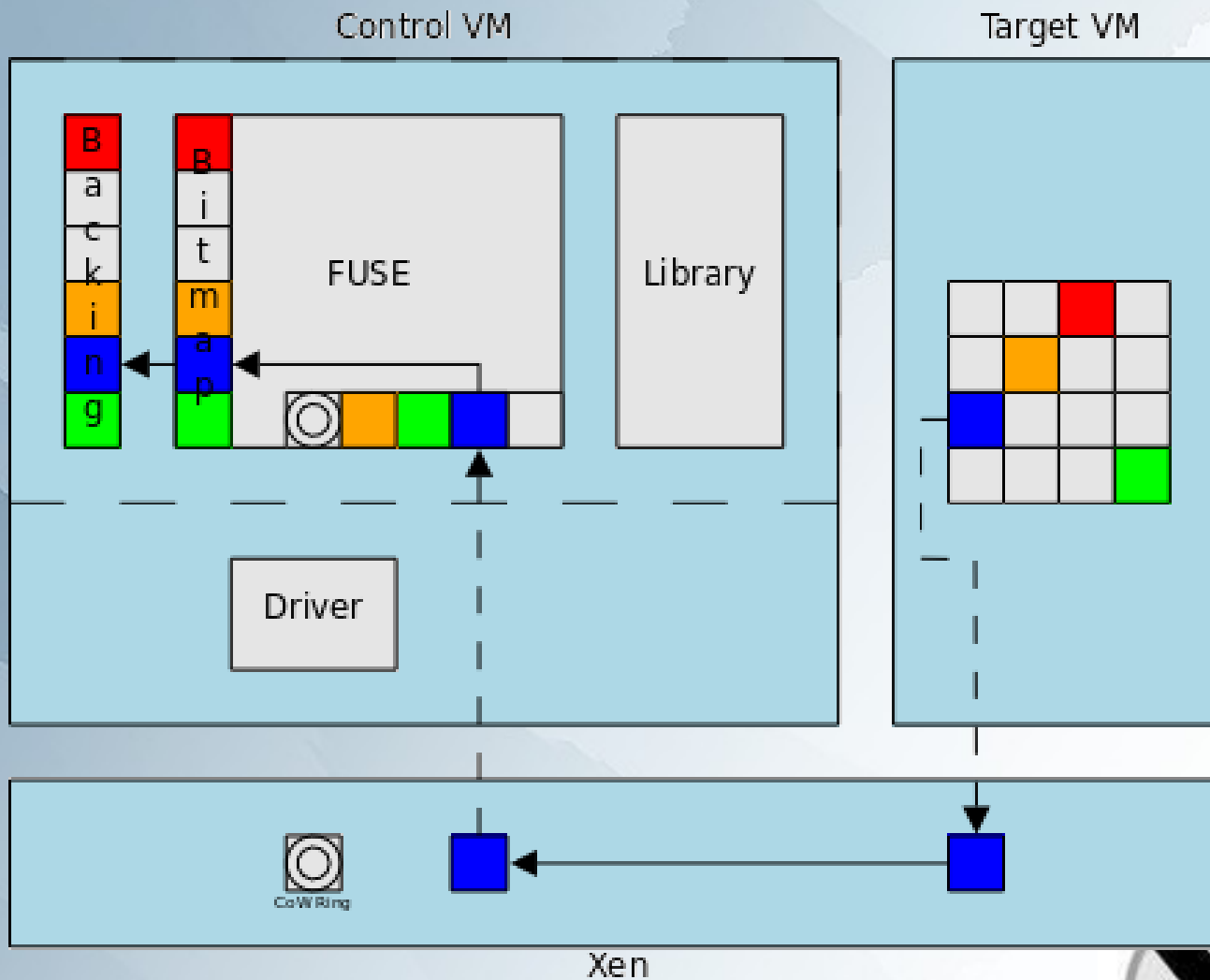


Architecture

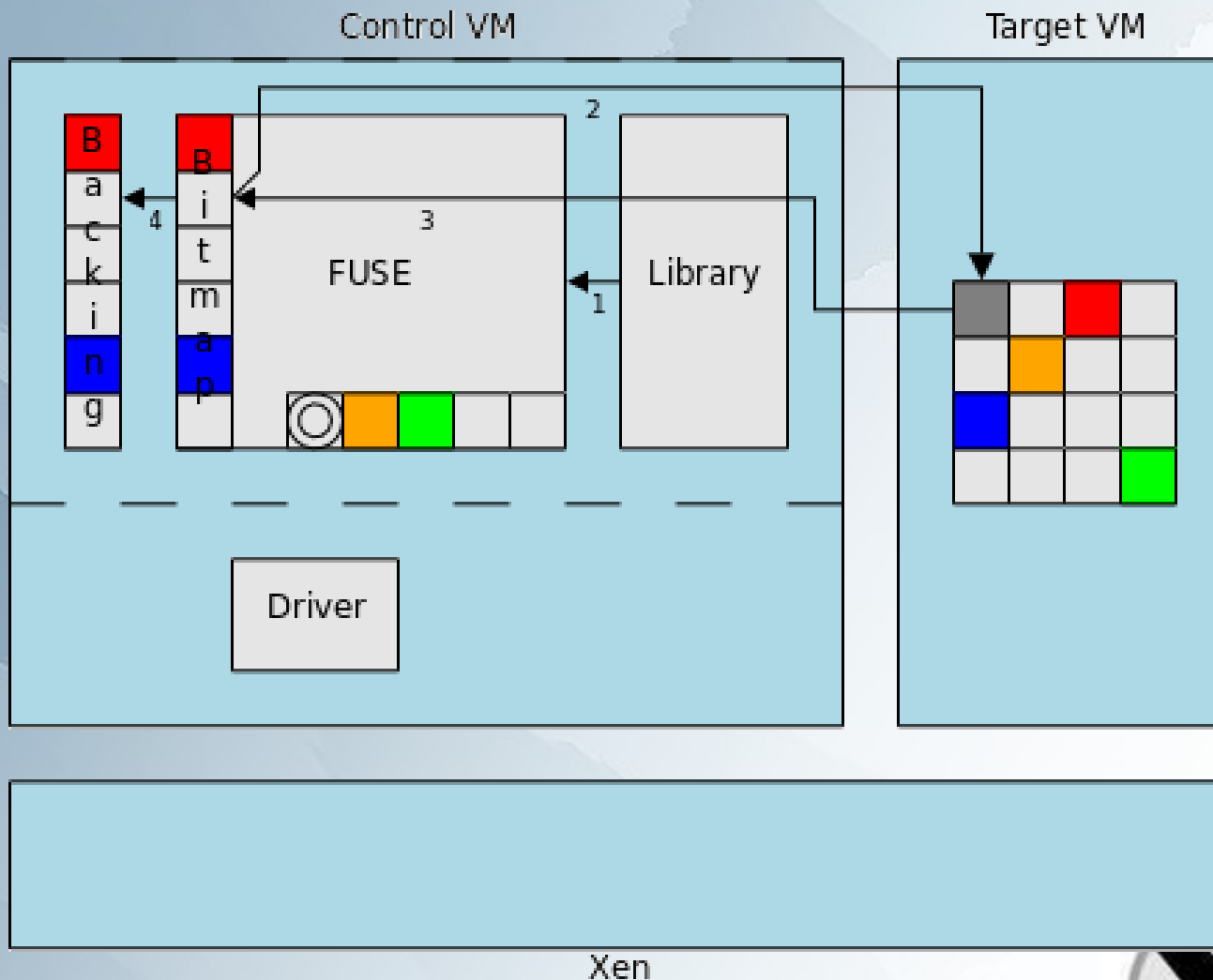
- Kernel driver
 - Validate and pin user buffer frames
 - Populate ring with MFNs and pass to Xen
- Xen
 - domctl to take snapshots
 - Copy frames into snapshot buffers on write faults



Fault Path Flow



Accessing CoW Memory



Trickier Bits

- Write after read
- Catching all writes
- Running out of buffer space



Write After Read

- Problem:
 - Check snapshot buffer for frame (not found)
 - Read VM's memory directly
 - VM writes to frame after buffer was checked but before it was read
- Solution:
 - After reading from VM's memory directly, re-check snapshot buffer for frame



Catching All Writes

- Problem:
 - Normal writes, mark dirty called before writes
 - Emulated mode/QEMU, mark dirty after writes
 - Grant tables, mark dirty called on unmap
- Solution:
 - Normal writes, save frame when mark dirty is called
 - Emulated mode/QEMU, add new hypercall before write
 - Grant tables, save frame on map



Running Out of Buffer Space

- Problem:
 - Buffer is limited size and filled in fault context
 - Once faulted, it's too late to flush the buffer
- Solution:
 - Guarantee enough buffer space before faulting
 - Threshold with minimum number of frames (~16)
 - Pause target domain and flush buffer on return from fault



Conclusion

- VM snapshots
- General purpose, lightweight mechanism

Coming soon to Xen-devel!

I'd be happy to talk about OCaml XenStore too

