Distributed Snapshots with Virtual Machines

Matei Ripeanu
with Andrew Warfield, Brendan Cully, Mike Feeley
Motivation

- Single system: snapshots allow
  - ‘Time travel’
    - Reduce failure costs
    - Debugging
    - System audit
    - [Scalability studies]
  - Migration

- Distributed/parallel system:
  - All the above would be (even more) useful ...
  - ... but we do not have an transparent & efficient snapshot mechanism
Challenge problem (1):
Parallel application checkpoint/restart

- Large parallel applications can expect multiple failures during a normal run
  - This situation is getting worse as we scale up
- Possible solutions
  - Buy reliable components (expensive!)
  - Reduce the cost of failures through checkpoint/restart mechanisms

\[ MTBF = \frac{1}{\sum_{i=1}^{N} \lambda_i} \approx \frac{1}{N \lambda} \]

- Challenge: checkpoint/restart a real application on a 1K nodes cluster with reasonable overhead
Challenge problem (2):
Migration of an e-commerce application

- Most e-commerce applications deployed on large clusters
- Scenario: the whole cluster is going down. What do we do?

  Challenge: <1s observable downtime while migrating an entire e-commerce application
A bit of distributed snapshots theory ...

- Non-deterministic problem
  - Guarantee that: Snapshot state is reachable from initial state, final state is reachable from snapshot

- Main approaches
  - Coordinated snapshots
    - Assumes: synchronized clocks
    - But: Unrealistic
  - Uncoordinated snapshots [Chandi’85]
    - Assumes: communication channels with in-order delivery, bounded message propagation time
    - But: Memory expensive – not-bounded!
State of the art

Snapshots for parallel applications
- Application level
- Library level (modified MPI, OS support)
- System level

Operations: most HPC centers do not provide any support
- Checkpointing left entirely to the application developer
- Some systems (Cray) do, but limited to single node
- Job management systems (e.g., LSF) have the appropriate plug-ins
How can a platform based on VMs help?

- **Coordination backbone**
  - Application runs in a VM. Second VM on the same node for signaling/management.

- **High-level abstraction**
  - Bundles: memory, registers, file-descriptors, sockets

- **Time travel**
  - Can mask suspended state duration:
    - Applications that rely on timing can still work correctly
    - E.g., timing the core computation to determine progress, communication timers.

- **Delayed message delivery**
  - Enables network "flush"
Brief detour: Devices in Xen

**Problem 2: Stability.** Drivers are a huge amount of (often buggy) code. We don’t want to count on them for reliability. Sometimes they crash.

*Option 1:* VMM runs physical device driver. VM drivers for “virtual” device. Either real (emulated) HW, or idealized.
Brief detour: Devices in Xen.

**Option 2:** VMM exports physical hardware to a device VM. Use OS driver, OS mechanisms (e.g. packet forwarding)
Brief detour: Devices in Xen

Option 2: VMM exports physical hardware to a device VM. Use OS driver, OS mechanisms (e.g. packet forwarding).

Exploit this architecture to queue messages while draining the network!
Putting everything together: Coordination protocol

Assumptions
- Bounded (known) message propagation time – to ‘flush’
- Bounded state saving time – to detect node failures at barrier
Status

Just starting:

- Putting together an experimental platform
- Experimenting: How far unmodified Xen and the coordinated checkpoint algorithm takes us with real applications?
Discussion: potential stoppers

- Runtime overheads
  - New hardware allows proper virtualization (no more full- & para-virtualization)
  - Communication overheads: already accepted
- Checkpoint overheads: large snapshots to persist on disk
  - Incremental checkpoint techniques
  - To exploit: similar state at all machines
- Application correctness depends on wall-clock time
  - Parameterize the type of time the VMM provides?
Discussion: Interesting problems

- Data preservation problems
  - Reduce aggregate checkpoint state based on similarities
  - Optimized file system: multiple write / (maybe) one write (slow)
  - Improve scalability & snapshot availability by peering nodes

- What does it mean to ‘virtualize’ a distributed platform?
  - Snapshots
  - Clocks
  - Internal routing

- What does it take to scale to 1K/10k/100k nodes?
- Fast migration?
Summary

- VM-based platform natural match for distributed snapshots
  - Converging view: virtualization trend extends to distributed platforms
    - State collection (Snapshots) - basic functionality for virtualization platform

- Benefits
  - ‘time-travel’
  - Enables competitive compute resource market:
    - Migration enables detaching resource and application/service providers
  - Improve resource utilization of HPC clusters
  - Integrate with Virtual Clusters (ANL)
Thank you

- Questions
What’s the state of the art?

- Live (single) machine migration

- Projects that reconfigure platform based on observed traffic (NWU, UFlorida, Purdue, others)

- Virtual playgrounds (ANL)