Remus: VM Replication

Jeff Chase
Duke University
Recall: virtual machines (VMs)

- Each guest VM runs a complete OS instance over an isolated “sliver” of host physical memory.
- Hypervisors support **migration** and **suspend/resume**.
  - Both operations require an **atomic snapshot (checkpoint)** of VM memory state and register contexts.
  - Capture modified pages and write them to snapshot.
Capturing modified pages

- How to do it?
- Recall the **Address Translation Uses** slides earlier.
- <Discuss.>
Remus checkpoints

• Snapshot the VM, but don’t suspend it.
  – Snapshot periodically as it executes.
  – Snapshot concurrently: keep running while snap is in progress.

• Migrate the VM, but don’t start the remote copy.
  – Just load the snapshot on the remote host.
  – Transmit “live” incremental checkpoints over the network.
  – Update the remote snapshot/copy/instance in place.
  – Remote host is a **warm standby** or **backup replica**.

• All checkpoints are **atomic**: they capture a point in time.
Remus Checkpoints

- Remus divides time into *epochs* (~25ms)
- Performs a *checkpoint* at the end of each epoch
  1. Suspend primary VM
  2. Copy all state changes to a buffer in **Domain 0**
  3. Resume primary VM
  4. Send asynchronous message to backup containing state changes
  5. Backup VM applies state changes

[Diagram showing primary VM and Domain 0 connected to Primary Server and Xen VMM, with an arrow labeled "Periodic Checkpoints (Changes to VM State)", leading to Domain 0 connected to Backup VM and Xen VMM, connected to Backup Server]
RemusDB: efficient and transparent active/standby high availability for DBMS implemented in the virtualization layer
- Propagates all changes in VM state from primary to backup
- High availability with no code changes to the DBMS
- Completely transparent failover from primary to backup
- Failover to a warmed up backup server
Remus

1. Checkpoint
2. Transmit
3. Sync
4. Release

Primary Host
- Completed Execution
- Speculative Execution

Backup Host
- State Buffer

Client’s View
- Committed State
Remus Checkpoints

- After a failure, the backup resumes execution from the latest checkpoint
  - Any work done by the primary during epoch C will be lost (unsafe)
- Remus provides a consistent view of execution to clients
  - Any network packets sent during an epoch are buffered until the next checkpoint
  - Guarantees that a client will see results only if they are based on safe execution
  - Same principle is also applied to disk writes
Outbound packet buffering
Disk (FS) updates

1. Disk writes are issued directly to local disk
2. Simultaneously sent to backup buffer
3. Writes released to disk after checkpoint
Remus implementation
Tardigrade (NSDI-15)

Figure 2: Overview of Tardigrade architecture for replicating lightweight virtual machines (LVMs)
Remus checkpoint latency

- time suspended
- time transmitting

Pages dirtied: 1, 256, 512, 1024

Milliseconds: 0, 5, 20, 50

Error bars indicate variability in the data.
Remus overhead

![Bar chart showing kernel build time (in seconds) vs. checkpoints per second. The chart indicates a trend where a higher number of checkpoints per second results in a longer kernel build time.](image)
Tardigrade

Figure 4: Effect of memory dirtying rate on CDF of latency seen by ping client
Tardigrade

Figure 5: Effect of memory dirtying rate on average checkpoint size of ping server.