

# Thanks for Live Snapshots, Where's Live Merge?

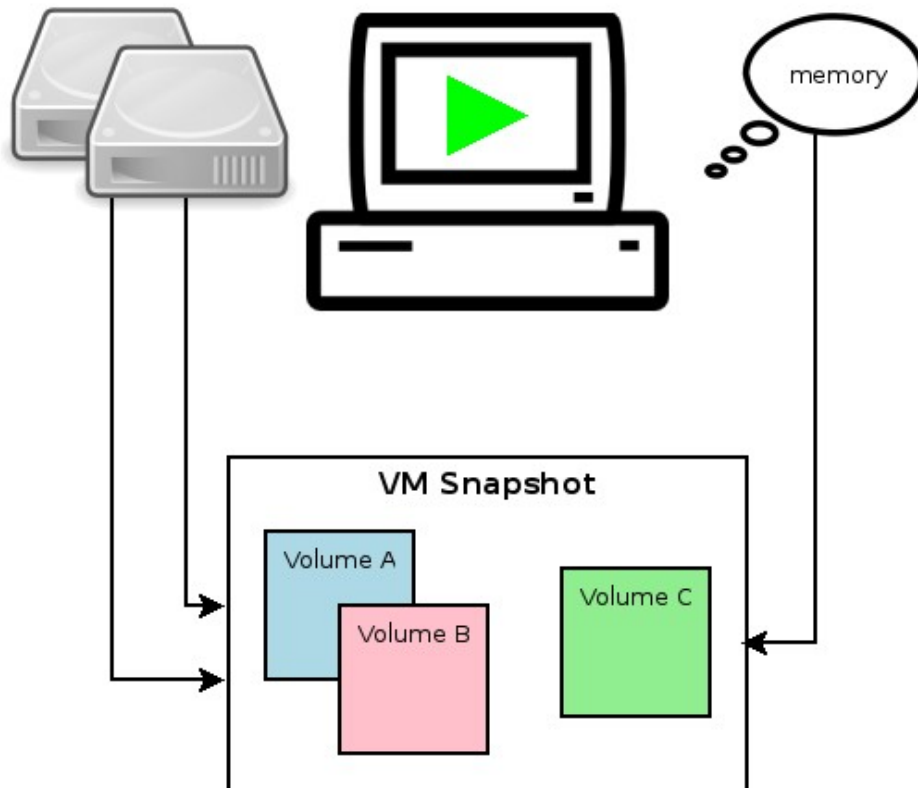
KVM Forum  
16 October 2014

Adam Litke  
Red Hat

- Introduction of Live Snapshots and Live Merge
- Managing Live Merge for Reliability and Simplicity
- Future work

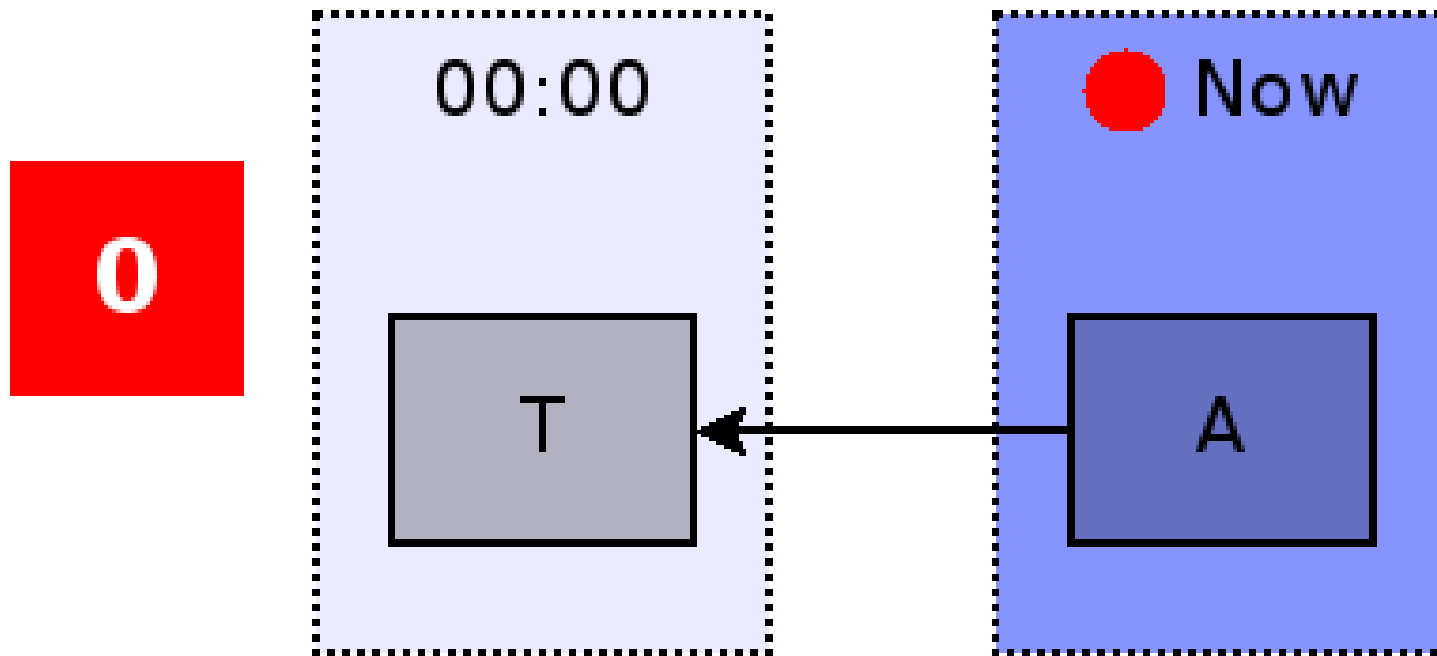
# Introduction of Live Snapshots and Live Merge

# Live snapshots



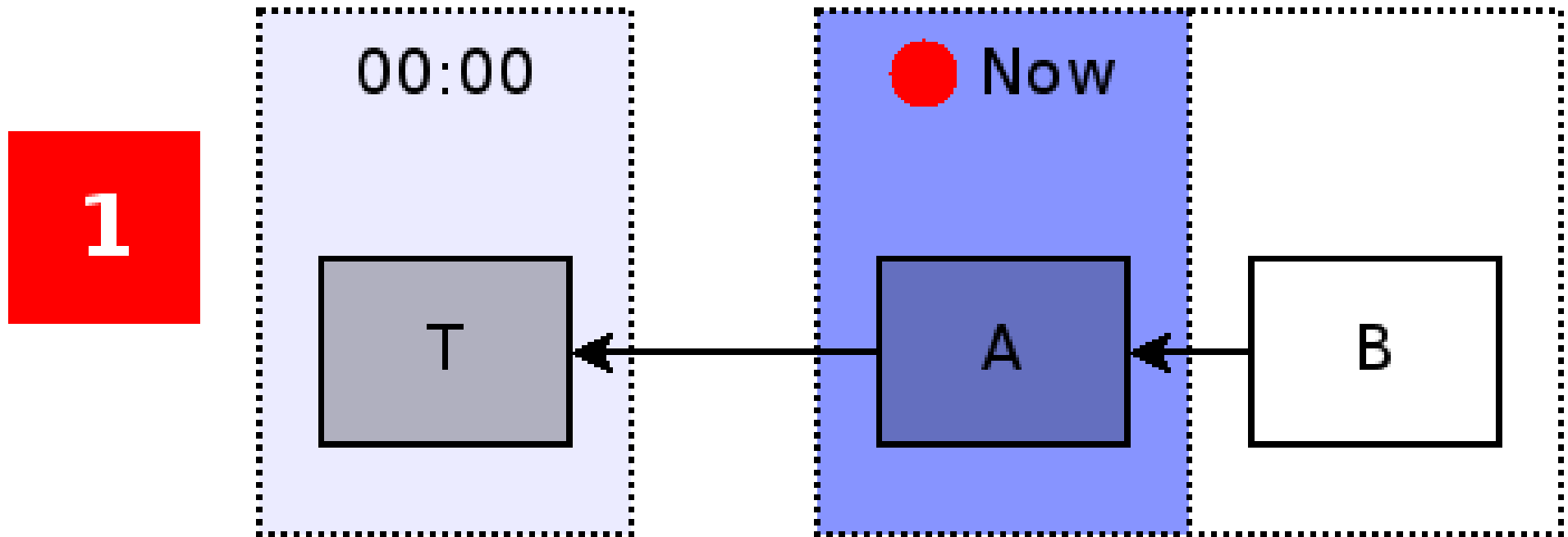
- Capture disks and memory at a point in time
- Implemented using qcow2 volume chains
- Uses
  - Preview or revert
  - VM live backup
  - Live storage migration

# Creating a live snapshot of a disk



INITIAL STATE

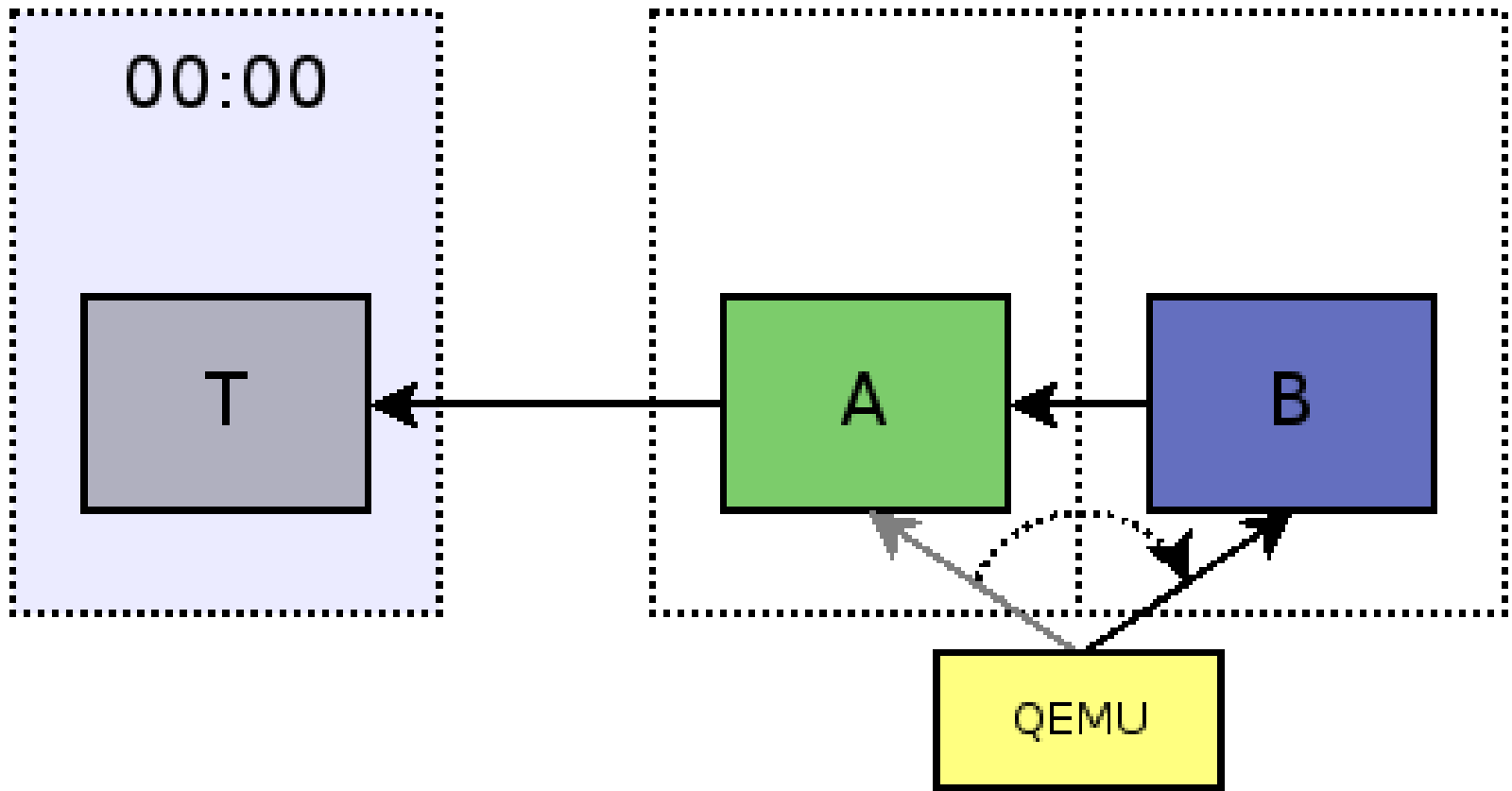
# Creating a live snapshot of a disk



CREATE VOLUME

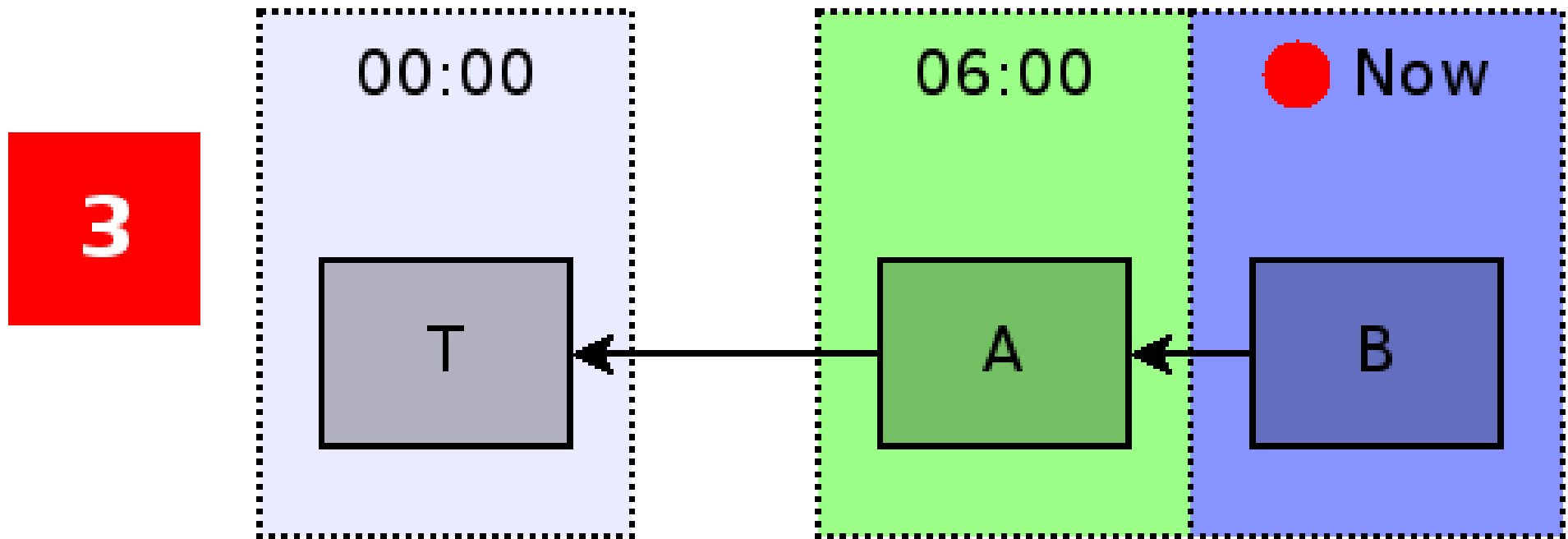
# Creating a live snapshot of a disk

2



PIVOT

# Creating a live snapshot of a disk



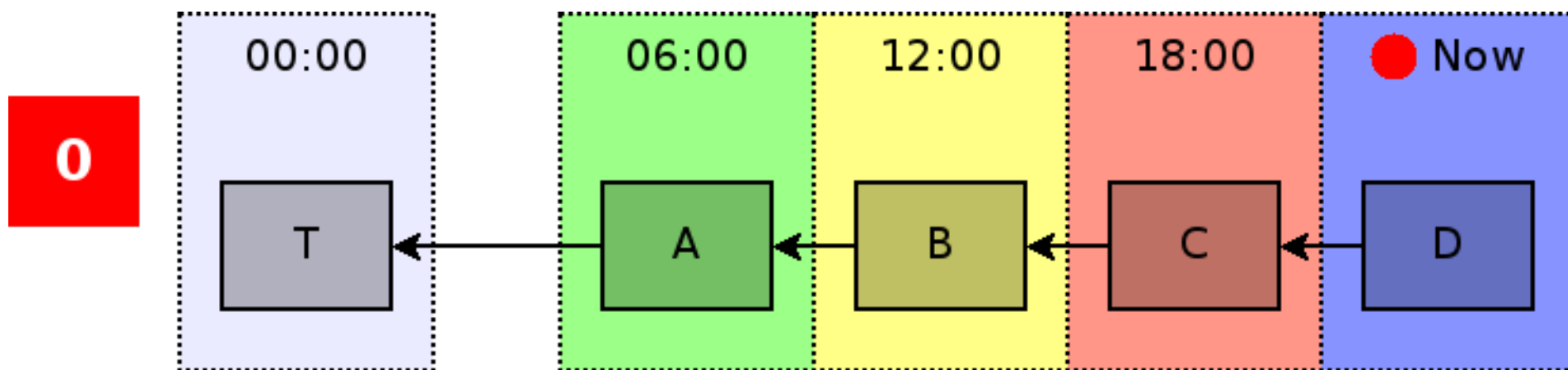
FINAL STATE



# Deleting a snapshot

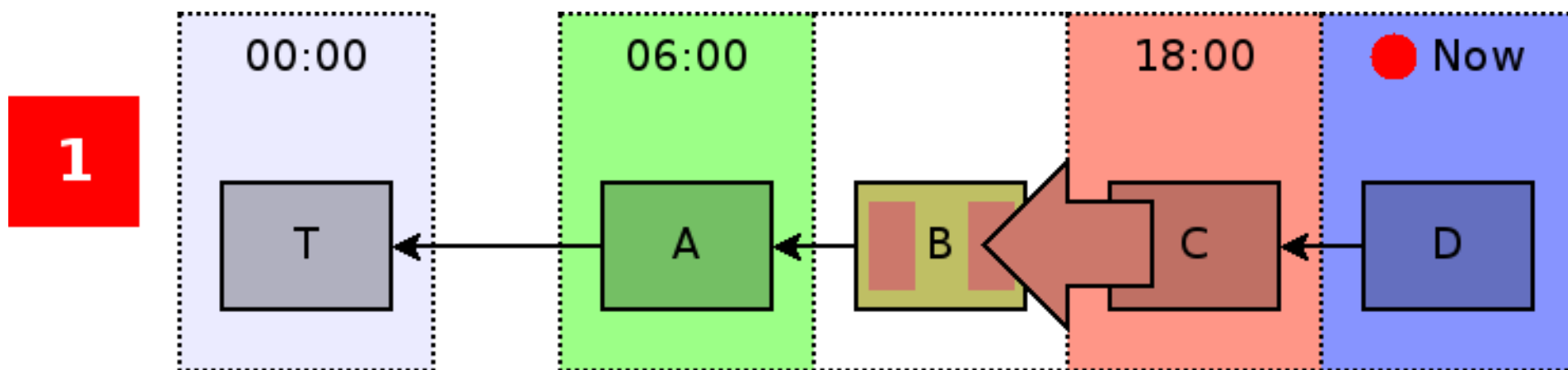
- Why?
  - Increase performance
  - May free up some storage space
  - To support symmetric snapshot operations
- Historically, oVirt has only supported deleting snapshots when a VM is powered off
- Live snapshot deletion is called live merge because two or more adjacent volumes are merged together
- Scenarios
  - Merge direction: forward vs. Backward
  - Merge source: active layer vs. Internal

# Backward internal merge



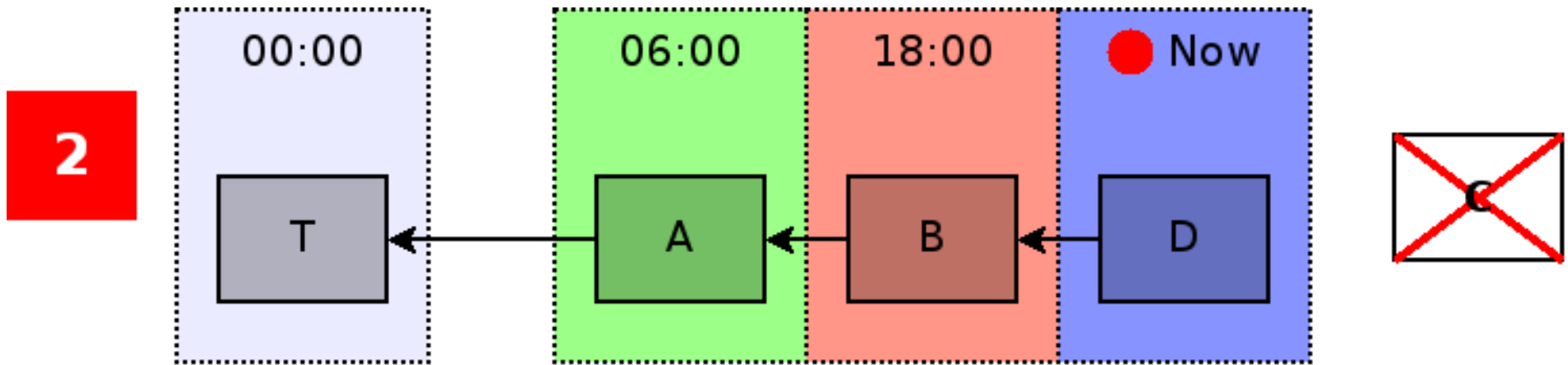
INITIAL STATE

# Backward internal merge



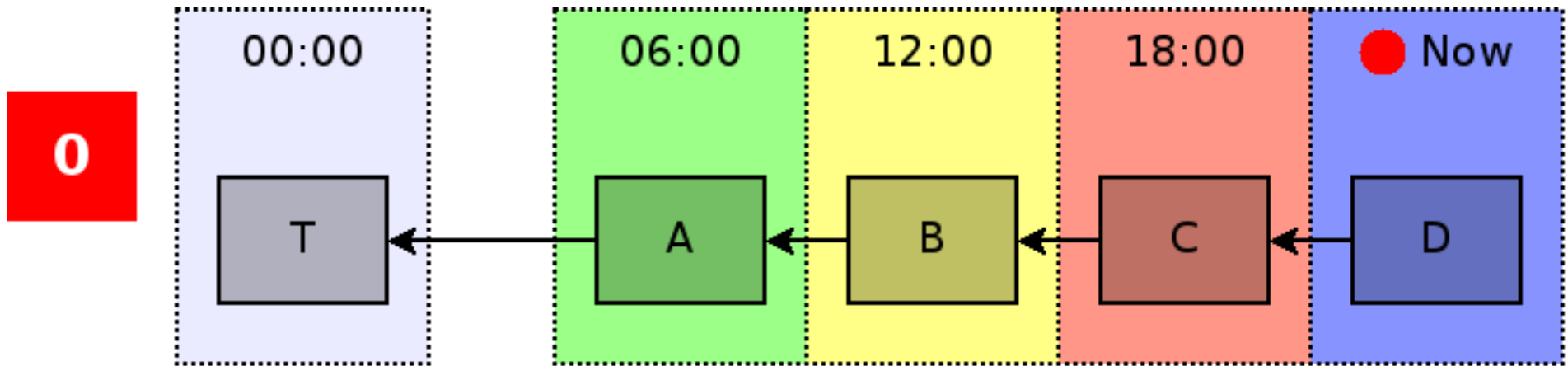
COMMIT ALL BLOCKS

# Backward internal merge



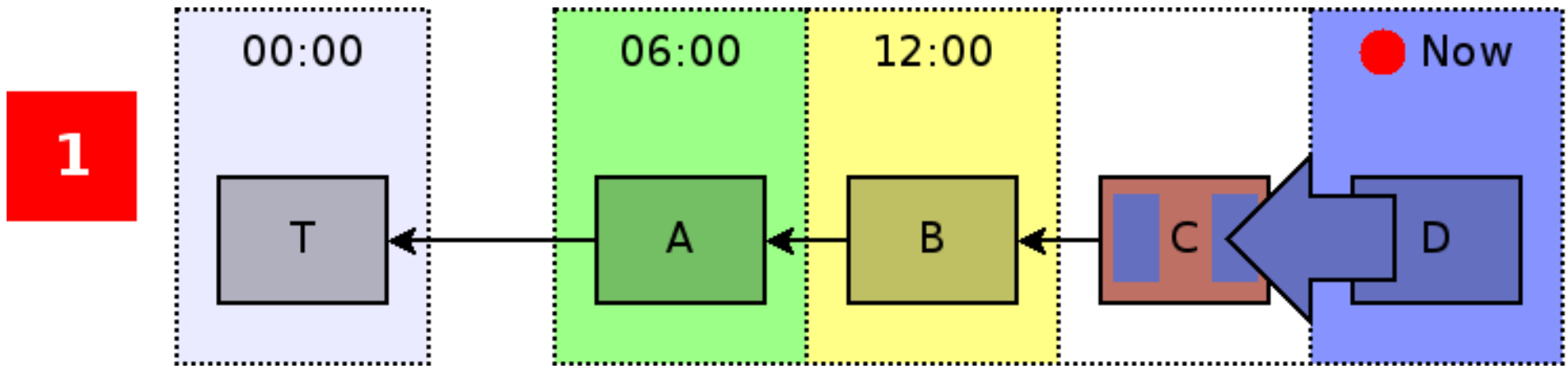
UNLINK AND DELETE MERGED VOLUME

# Backward active merge



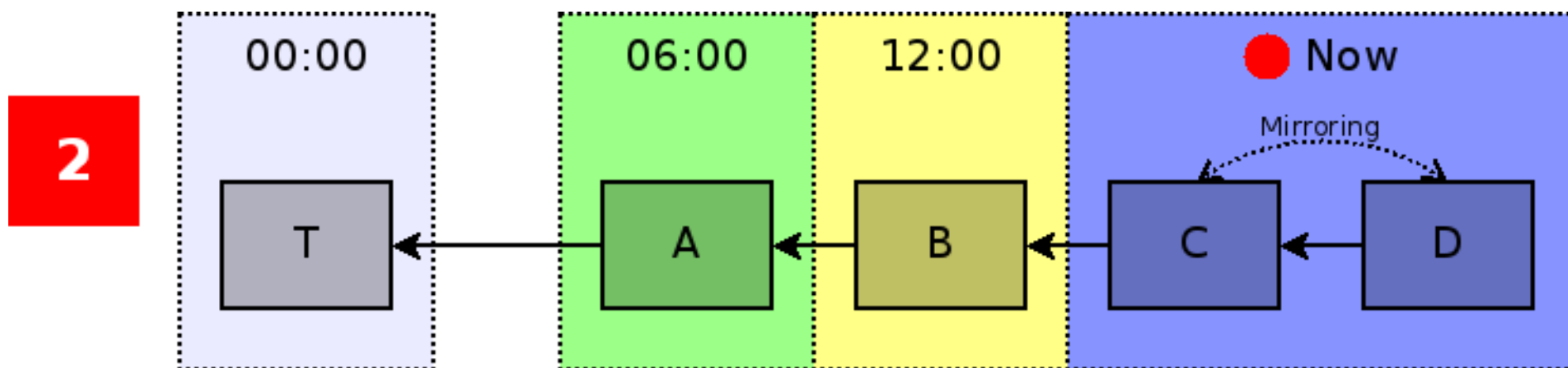
INITIAL STATE

# Backward active merge



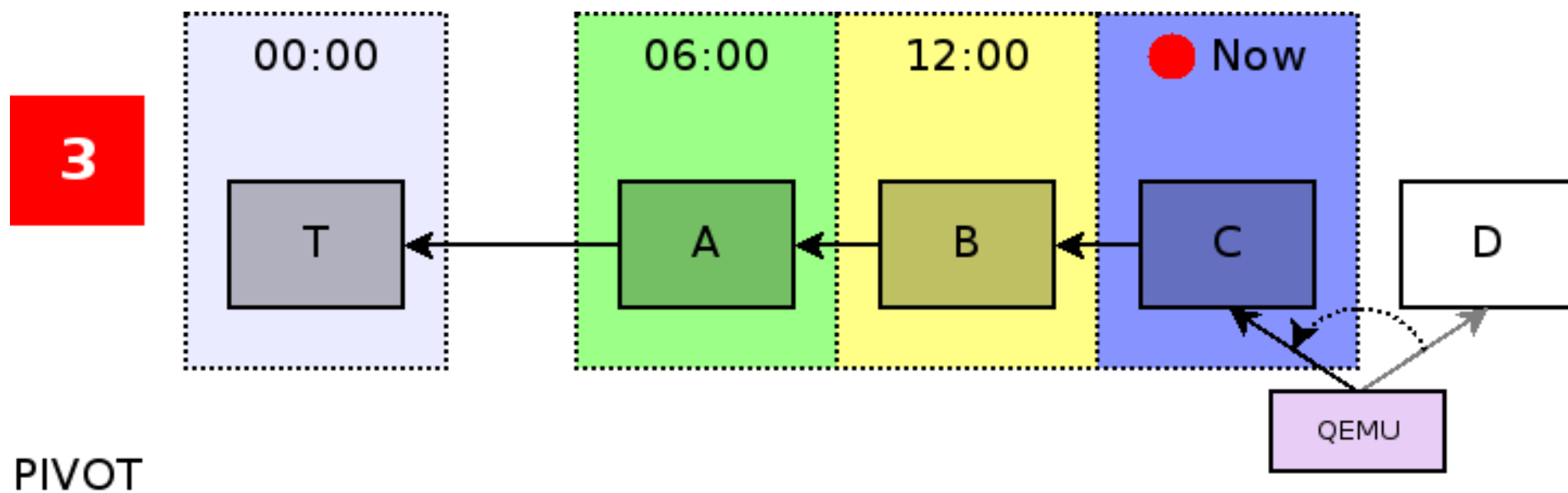
MERGE AND MIRROR WRITES

# Backward active merge



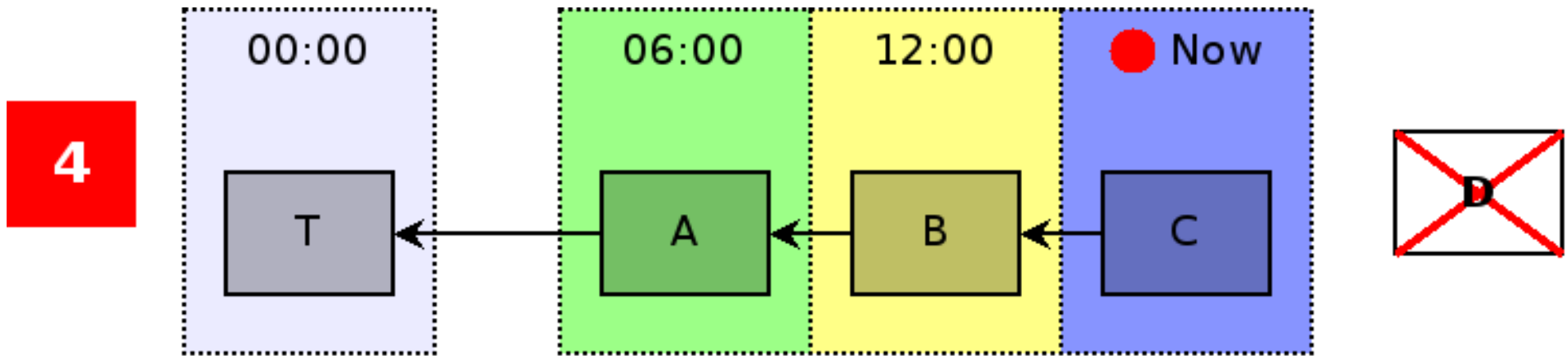
ACTIVE MIRRORING

# Backward active merge



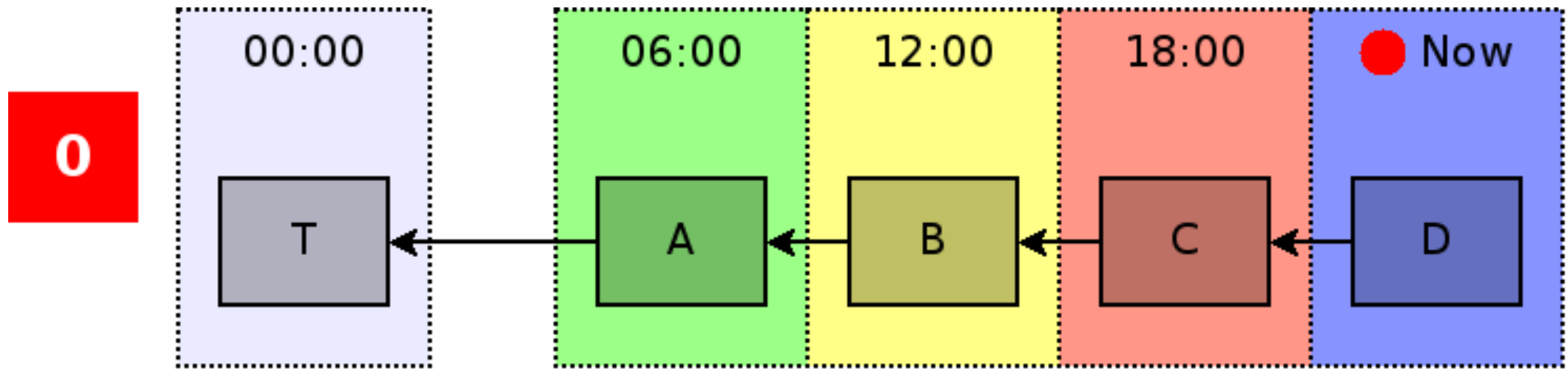


# Backward active merge

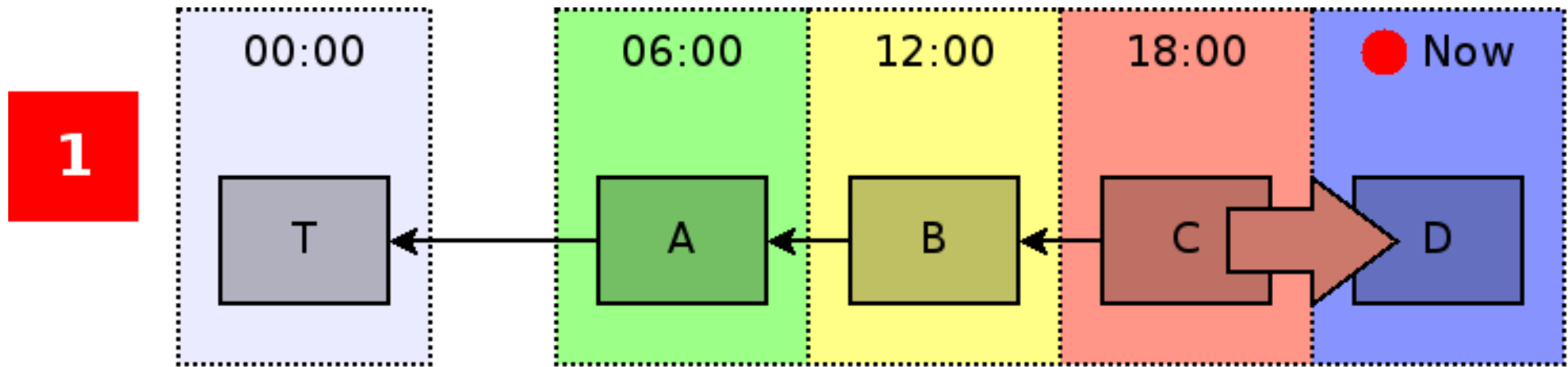


DELETE MERGED VOLUME

# Forward active merge

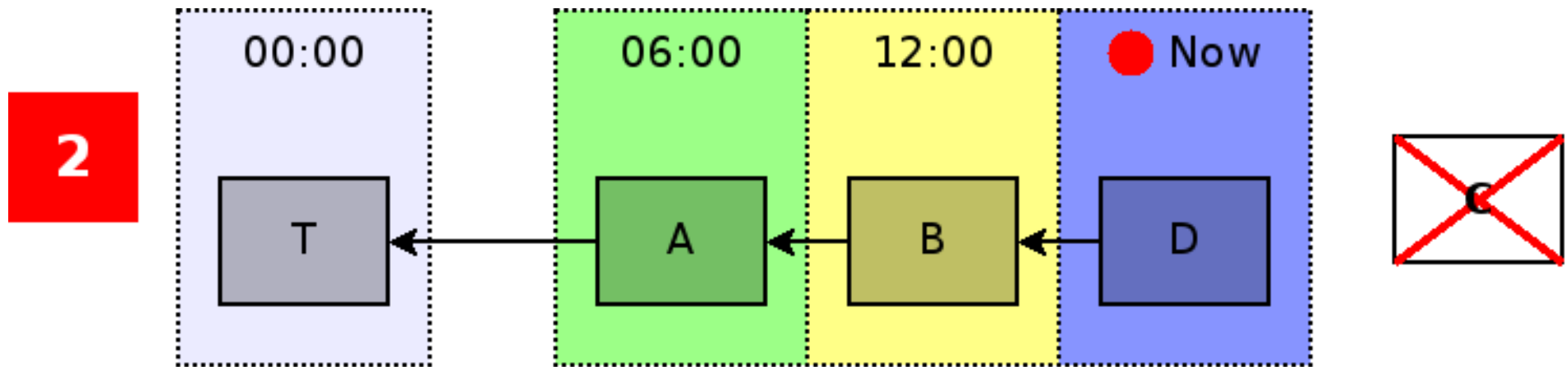


# Forward active merge



POPULATE

# Forward active merge



UNLINK AND DELETE MERGED VOLUME

# Choosing a merge strategy



- Forward merge
  - libvirt virDomainBlockRebase API
  - Must merge to the active layer
- Backward merge
  - libvirt virDomainBlockCommit API
  - May merge any volume into its parent
- Backward merge is the only feasible choice today but we'd like to have both directions in the future

```
int virDomainBlockCommit(virDomainPtr dom,  
                          const char * disk,  
                          const char * base,  
                          const char * top,  
                          unsigned long bandwidth,  
                          unsigned int flags)
```

- Starts a live merge operation
- Merges top into base
- You can rate-limit the copy operation if desired
- Flags allow you to set special behavior
  - Request an active layer merge
  - Preserve relative backingStore pointers

```
int virDomainGetBlockJobInfo(virDomainPtr dom,  
                             const char * disk,  
                             virDomainBlockJobInfoPtr info,  
                             unsigned int flags)
```

- Get information about currently running block jobs
  - Job type, bandwidth limit, progress cursor
- When jobs finish an event is emitted and they will no longer be reported by this API

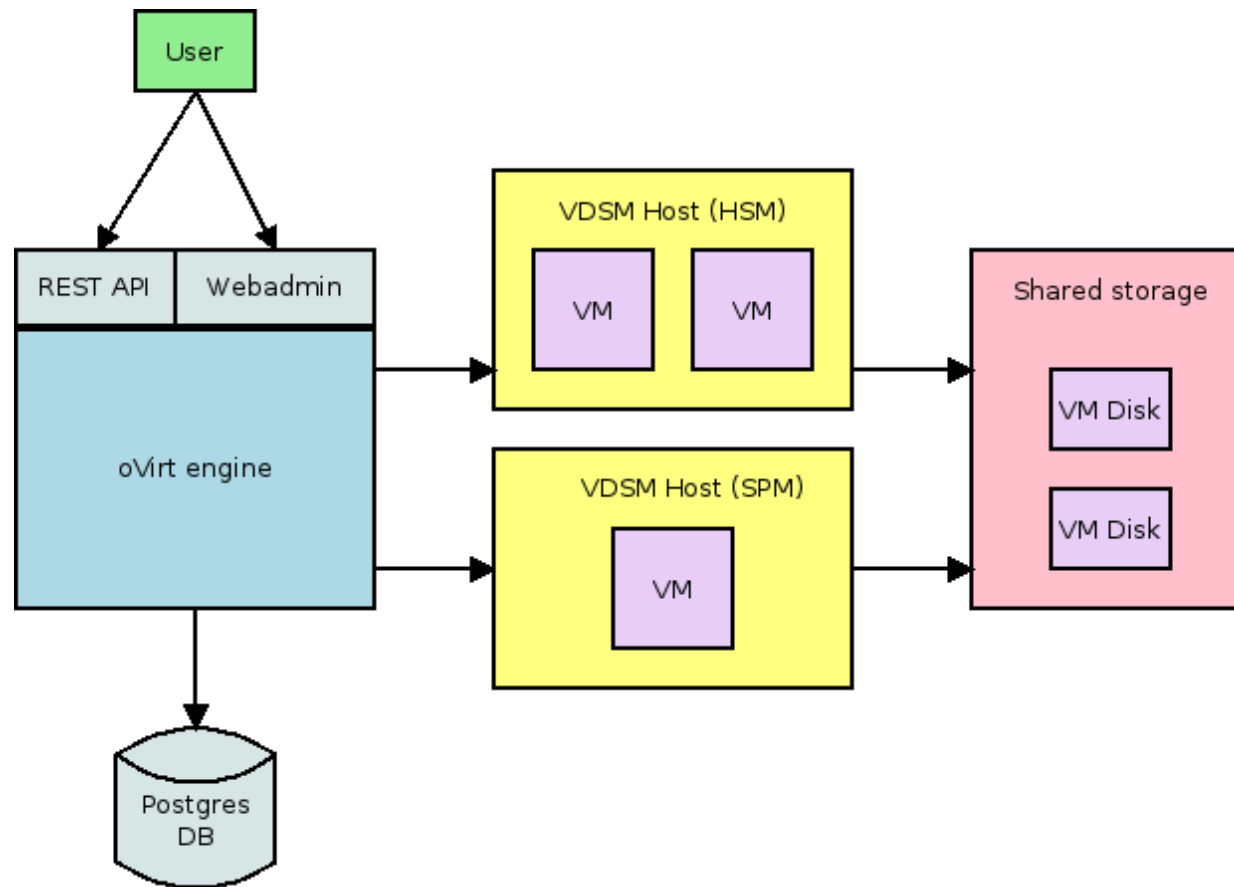
```
int virDomainBlockJobAbort(virDomainPtr dom,  
                           const char * disk,  
                           unsigned int flags)
```

- Cancel the currently running block job on a disk
- Also used to request a pivot after active layer merge



# Managing Live Merge for Reliability and Simplicity

# oVirt Architecture



# Entry points



The screenshot shows the oVirt web interface. At the top, there's a navigation bar with 'oVirt OPEN VIRTUALIZATION MANAGER', a user dropdown 'admin', and links for 'Configure', 'Guide', 'About', and 'Feedback'. Below this is a search bar labeled 'Vms:'. A main navigation menu includes 'Data Centers', 'Clusters', 'Hosts', 'Networks', 'Storage', 'Disks', 'Virtual Machines', 'Pools', 'Templates', and 'Vol'. A secondary menu contains actions like 'New VM', 'Edit', 'Remove', 'Clone VM', 'Run Once', 'Migrate', 'Cancel Migration', 'Make Template', 'Export', and 'Create Snapshot'. The main content area displays a table of VMs:

| Name         | Host  | IP Address | FQDN | Cluster | Data Center | Memory |
|--------------|-------|------------|------|---------|-------------|--------|
| HostedEngine | ale   |            |      | Default | Default     | 0%     |
| test         | lager |            |      | Default | Default     | 0%     |

Below the VM table, there's a detailed view for a selected VM with tabs for 'General', 'Network Interfaces', 'Disks', 'Snapshots', 'Applications', 'Affinity Groups', 'Permissions', 'Sessions', and 'Events'. The 'Snapshots' tab is active, showing a table of snapshots:

| Date               | Status | Memory                              | Description |
|--------------------|--------|-------------------------------------|-------------|
| Current            | Ok     | <input type="checkbox"/>            | Active VM   |
| 2014-Oct-01, 10:51 | Ok     | <input checked="" type="checkbox"/> | snap1       |

Buttons for 'Create', 'Preview', 'Commit', 'Undo', 'Delete', and 'Clone' are visible above the snapshot table. The 'Delete' button is circled in red. A right-hand pane shows VM details: 'General', 'Disks', 'Network Interfaces', and 'Installed Applications'. The 'General' tab is selected, showing: 'Defined Memory: 1024MB', 'Physical Memory Guaranteed: 1024MB', and 'Number of CPU Cores: 1 (1 Socket(s), 1 Core(s) per Socket)'. At the bottom, a task bar shows 'Last Task: 2014-Oct-01, 10:52 Creating VM Snapshot snap1 for VM test' and 'Alerts (2)'. The oVirt logo is also present in the bottom left corner of the interface.

# Entry points



Advanced Rest Client

Last used: Friday, 2014 April 25 15:50:38 UTC-4

Save

Open

▶ <http://192.168.2.102/ovirt-engine/api/vms/c17b1fce-0076-4693-b3e8-4ef79e45ef64/snapshots/0b231bf5-e39f-4eec-8c1a-e2ea3edef3f4>

GET  POST  PUT  PATCH  DELETE  HEAD  OPTIONS  Other

Raw

Form

Headers

Raw

Form

Files (0)

Payload

[Encode payload](#) [Decode payload](#)

application/xml

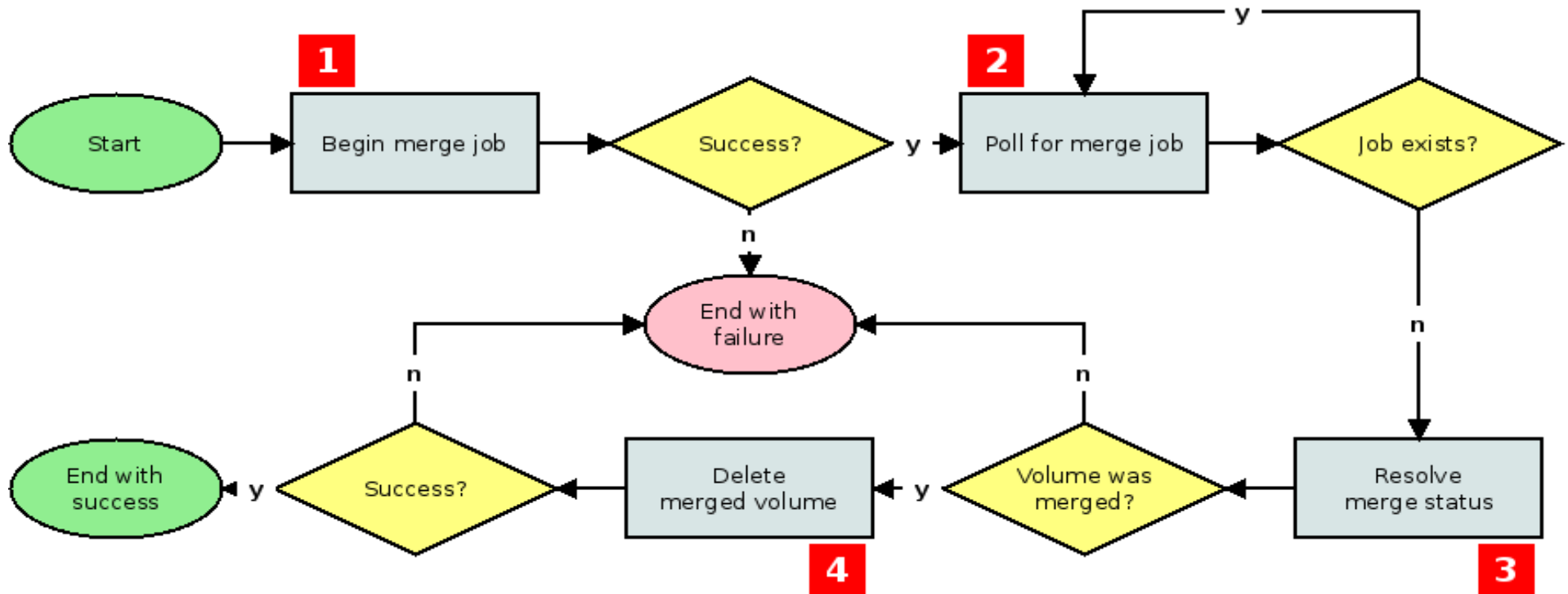
Set "Content-Type" header to overwrite this value.

Clear

Send

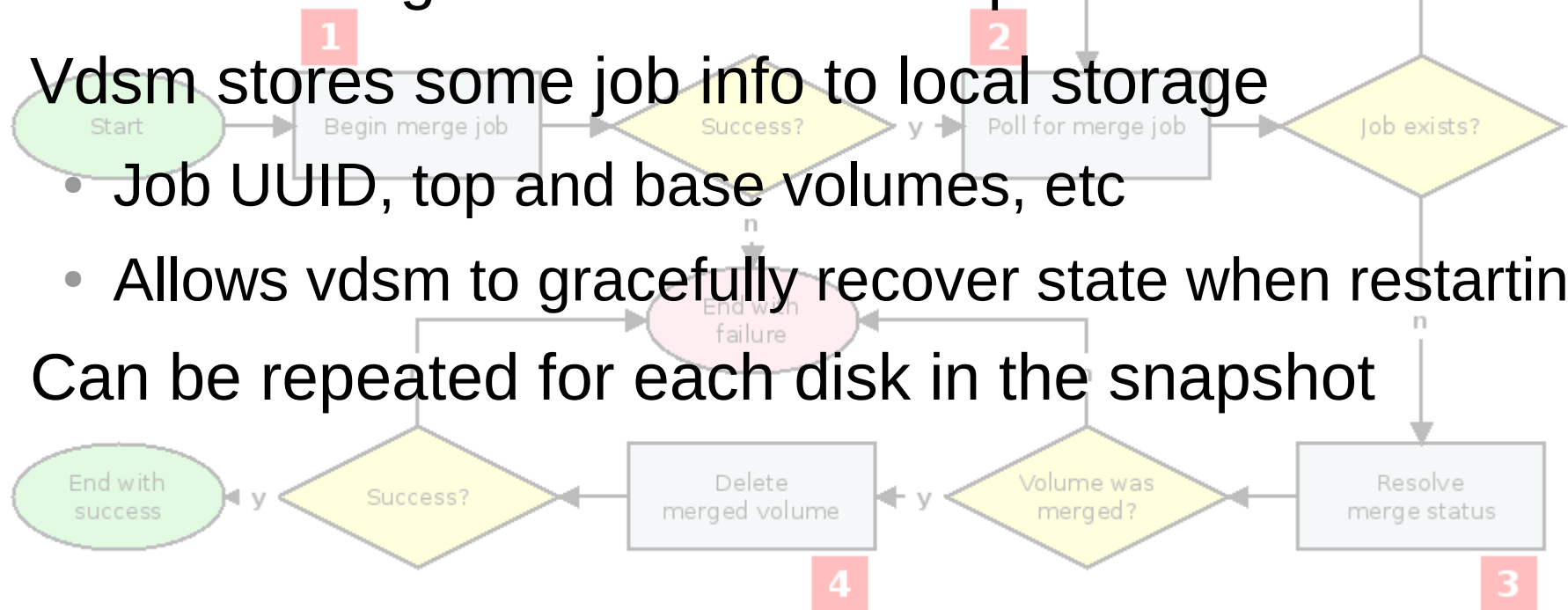
- Hide complexity behind a simple UI and API
- Snapshots involving multiple disks and memory
- Prevent invalid or potentially dangerous operations
  - Revert to a partially merged snapshot
  - Migration during live merge
  - Merge into a shared volume
- Fault tolerance and recovery

# High level flow



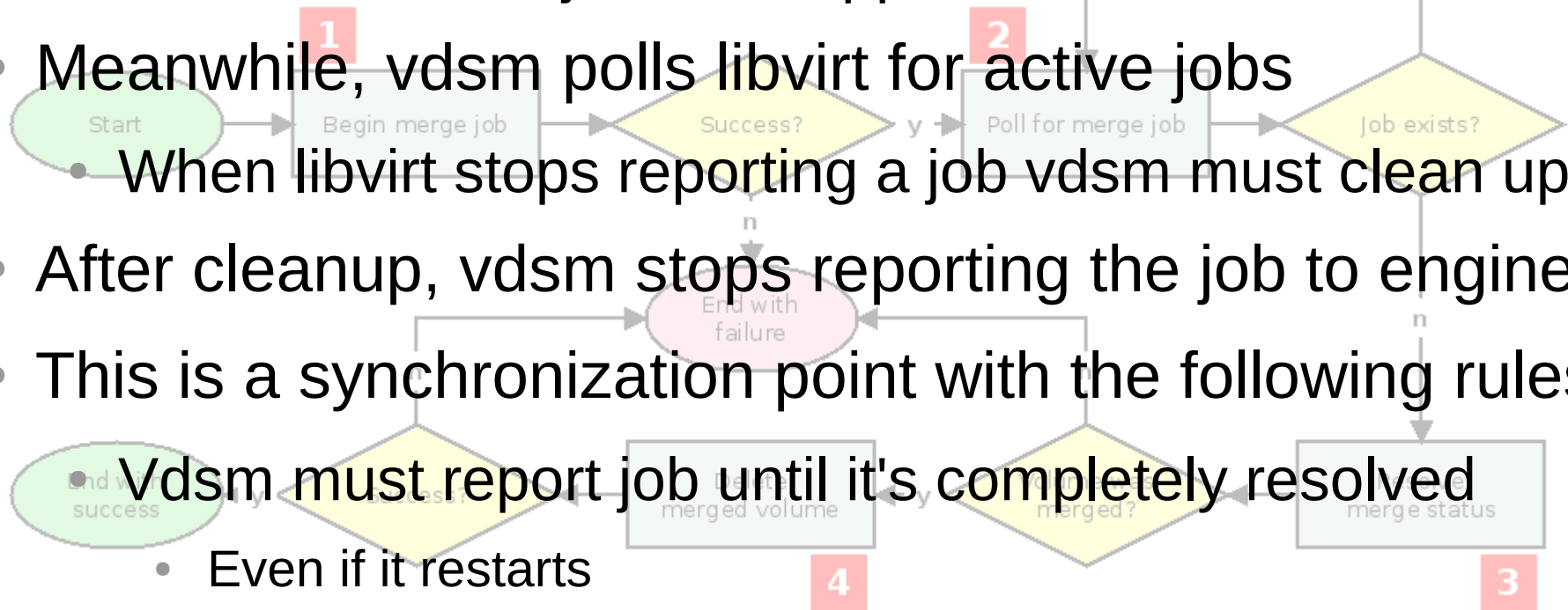
# 1. Begin merge job

- Ask vdsmd to start a live merge for a single vm disk
- Assume merge started unless explicit error is returned
- Vdsmd stores some job info to local storage
  - Job UUID, top and base volumes, etc
  - Allows vdsmd to gracefully recover state when restarting
- Can be repeated for each disk in the snapshot



## 2. Wait for merge job

- Engine periodically polls vdsmd to get active jobs
  - Just wait for the job to disappear from the results
- Meanwhile, vdsmd polls libvirt for active jobs
  - When libvirt stops reporting a job vdsmd must clean up
- After cleanup, vdsmd stops reporting the job to engine
- This is a synchronization point with the following rules
  - Vdsmd must report job until it's completely resolved
    - Even if it restarts
    - If engine does not receive any job info it must try again
    - When the job stops being reported advance to next step



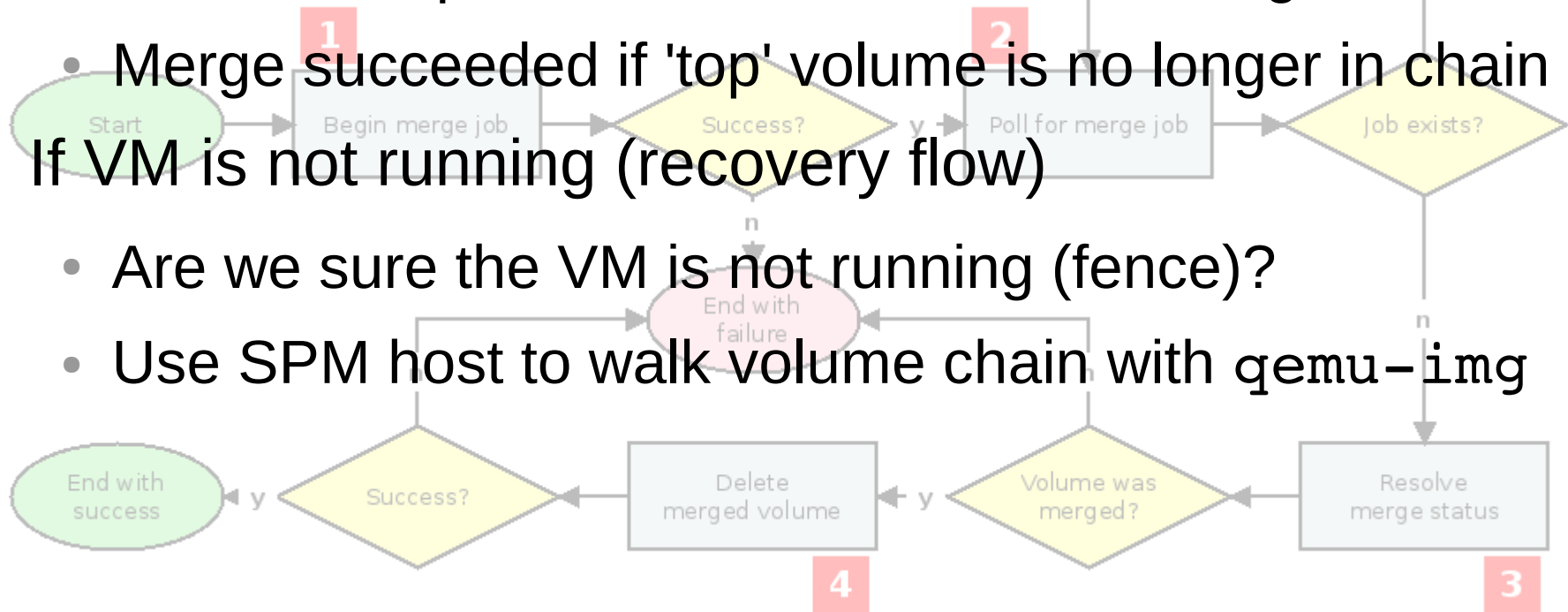


## 2. Wait for merge job (cleanup)

- Step 1: Manual pivot (active layer merge only)
  - Indicator: libvirt job info cursor is at 100%
  - Vdsm asks libvirt to pivot to the new active volume
- Step 2: Volume chain synchronization (all merges)
  - Indicator: libvirt job no longer reported
  - Vdsm gets new volume chain from libvirt XML
  - Vdsm syncs volume chain metadata on storage
  - Vdsm syncs in-memory volume chain info for VM
- Step 3: Stop reporting live merge job to engine

### 3. Resolve merge status

- Engine requests live VM configuration from vdsms
  - Vdsm has updated volume chain after merge finished
  - Merge succeeded if 'top' volume is no longer in chain
- If VM is not running (recovery flow)
  - Are we sure the VM is not running (fence)?
  - Use SPM host to walk volume chain with `qemu-img`

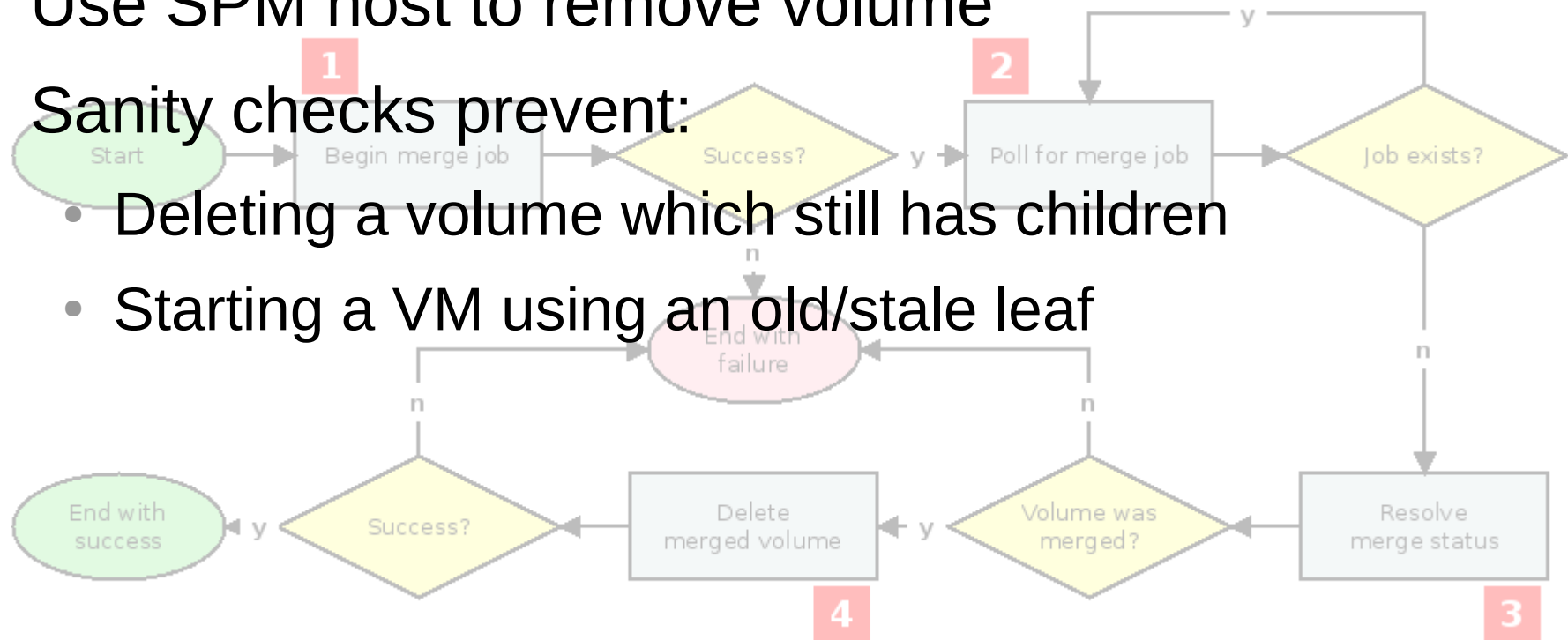


## 4. Delete merged volume

- Merge was successful and 'top' should be deleted
- Use SPM host to remove volume

- **Sanity checks prevent:**

- Deleting a volume which still has children
- Starting a VM using an old/stale leaf



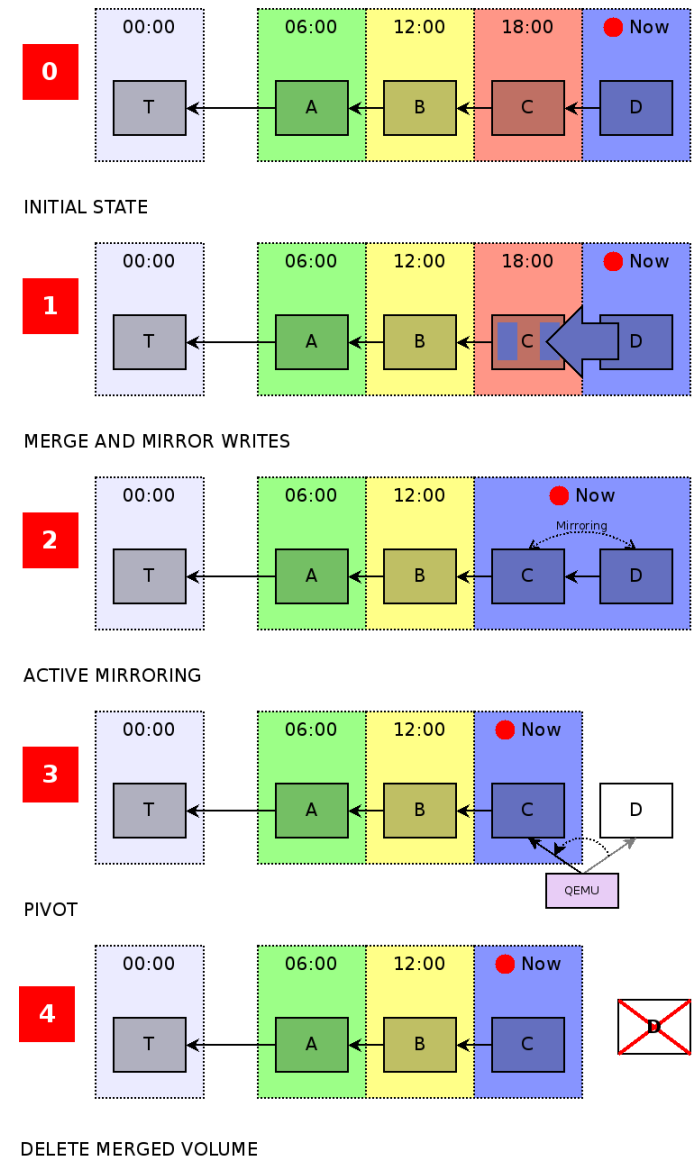
# Polling vs. event handling



- Events would be passed from  
qemu -> libvirt -> vdsms -> ovirt-engine
- If any component goes down we might miss an event
- To ensure recoverability, it must be possible to query the current state by asking two questions of the system
  - Is the job active?
  - If not, did it succeed or fail?
- Event handling could be added in the future as an optimization to eliminate polling delays for the common case.

# Scenario: qemu crash

- Is merge running? No.
- Use SPM host to run a recovery command
  - Run `qemu-img` to get current volume chain
  - Discard mirrored leaf if present
  - Correct vdsmd metadata and return new chain
- Mirrored leafs are flagged in vdsmd metadata before a pivot is requested



# Scenario: manager and vdsmd restart



- VDSM restart
  - Manager has a lapse in merge job info during restart and will continue to poll until reporting resumes
  - Upon restart vdsmd recovers the list of tracked jobs and handling will resume the next time libvirt is polled
- Manager restart
  - Live merge is a sequence of individual commands
  - Current progress is saved to the DB and the command state will be restored upon restart

# Scenario: manager / virt host destroyed



- Manager
  - In this case the DB is lost and it will be necessary to rebuild the oVirt environment
  - Data domain import allows you to recover VMs and templates from previous install
  - We must check all Vms on import as if qemu crashed
- Virtualization host
  - Admin must confirm that the host has been rebooted
  - Use qemu crash logic to synchronize and then the VM can be restarted on a different host

## Future Work



## Future work

- Automatic removal of temporary snapshots after live storage migration and VM backup
- Data domain import hook to sync vm volume chains
- Use forward merge if it would reduce I/O
- Improve responsiveness by using libvirt events and emitting vdsms events
- Delete multiple adjacent snapshots in one operation
- Rolling snapshots

# THANK YOU !

[http://www.ovirt.org/Features/Live\\_Merge](http://www.ovirt.org/Features/Live_Merge)  
devel@ovirt.org

#ovirt irc.oftc.net