# Logical Synchronous Replication in the Tintri VMstore File System

Gideon Glass
gxglass@gmail.com

joint work with Arjun Gopalan, Dattatraya Koujalagi, Abhinand Palicherla, and Sumedh Sakdeo

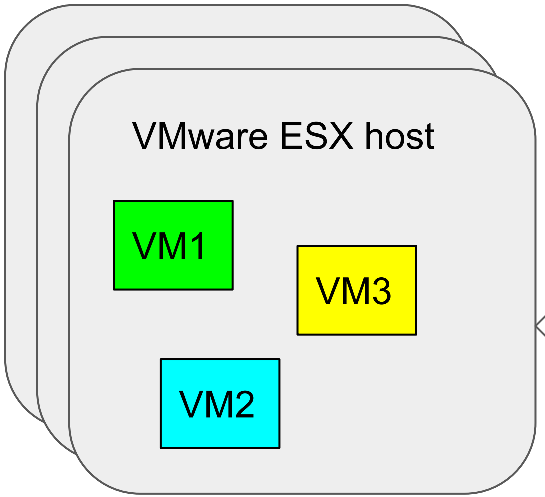
# Outline

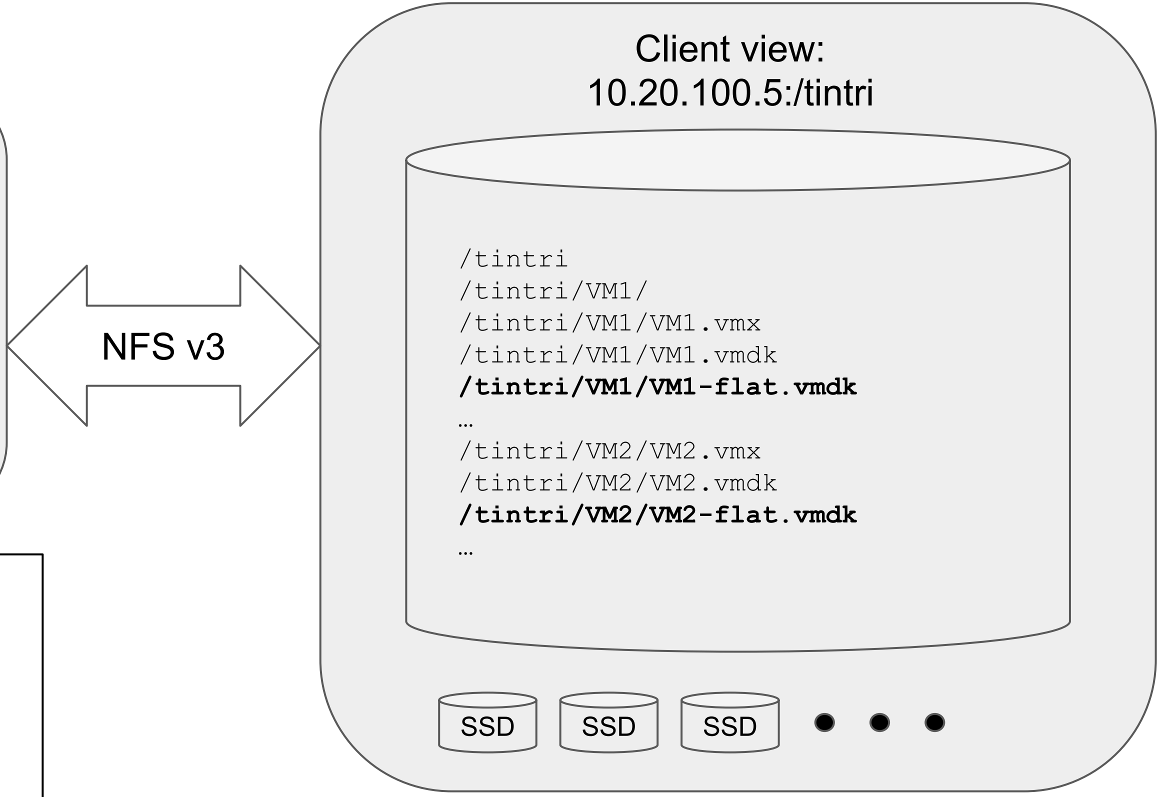
Background and Motivation (30%)

Major Technical Problems (60%)

Lessons Learned (10%)

# Tintri VMstore Ecosystem

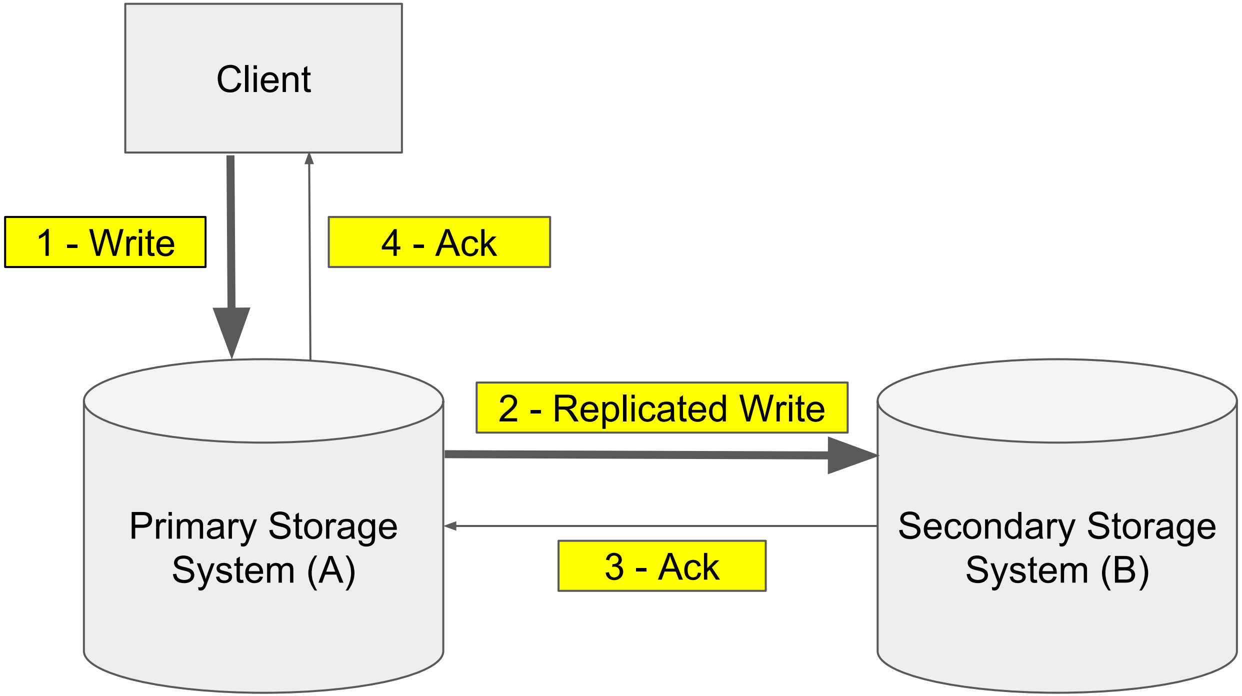


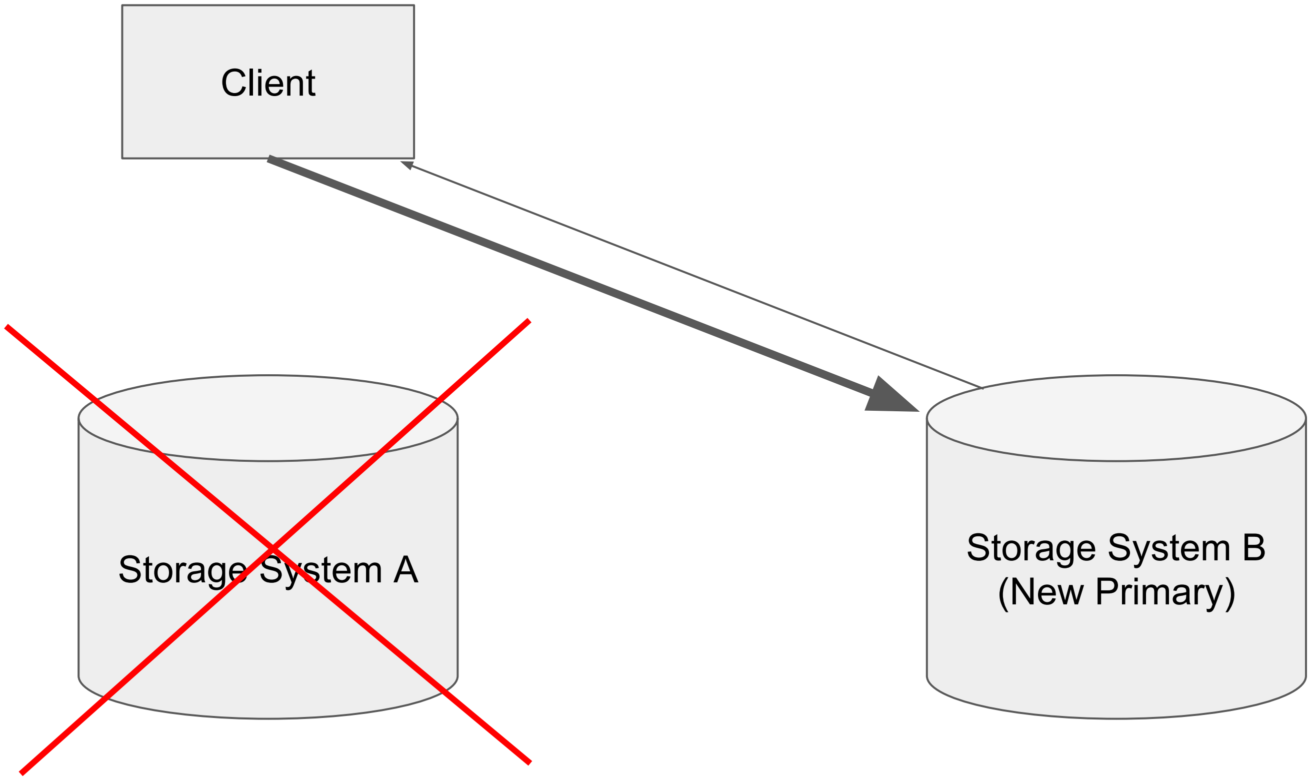

Typical VM: several dozen files
Dynamic operation mix:

- ~70% reads
- ~30% writes
- ~0.1% other

# Generic Synchronous Replication



# Transparent Failover



# What to replicate? Alternatives considered

| **How user might configure replication** | **Reason for rejecting** |
|---|---|
| Selected LUNs | VMstore does not expose LUNs |
| Selected filesystem volumes | We only have one filesystem |
| Whole system | Forces users to replicate all data;<br>subtle complications to rest of system |
| Per-VM | Does not work well with transparent failover |
| Directory | This is what we did -- let users configure one or more directories for replication, and replicate all filesystem operations occurring on files/subdirectories within these directories. |

# Major data path design challenges

Replicate writes

Replicate arbitrary filesystem operations

Resync efficiently: handle extended outages

Primary/Secondary integrity check

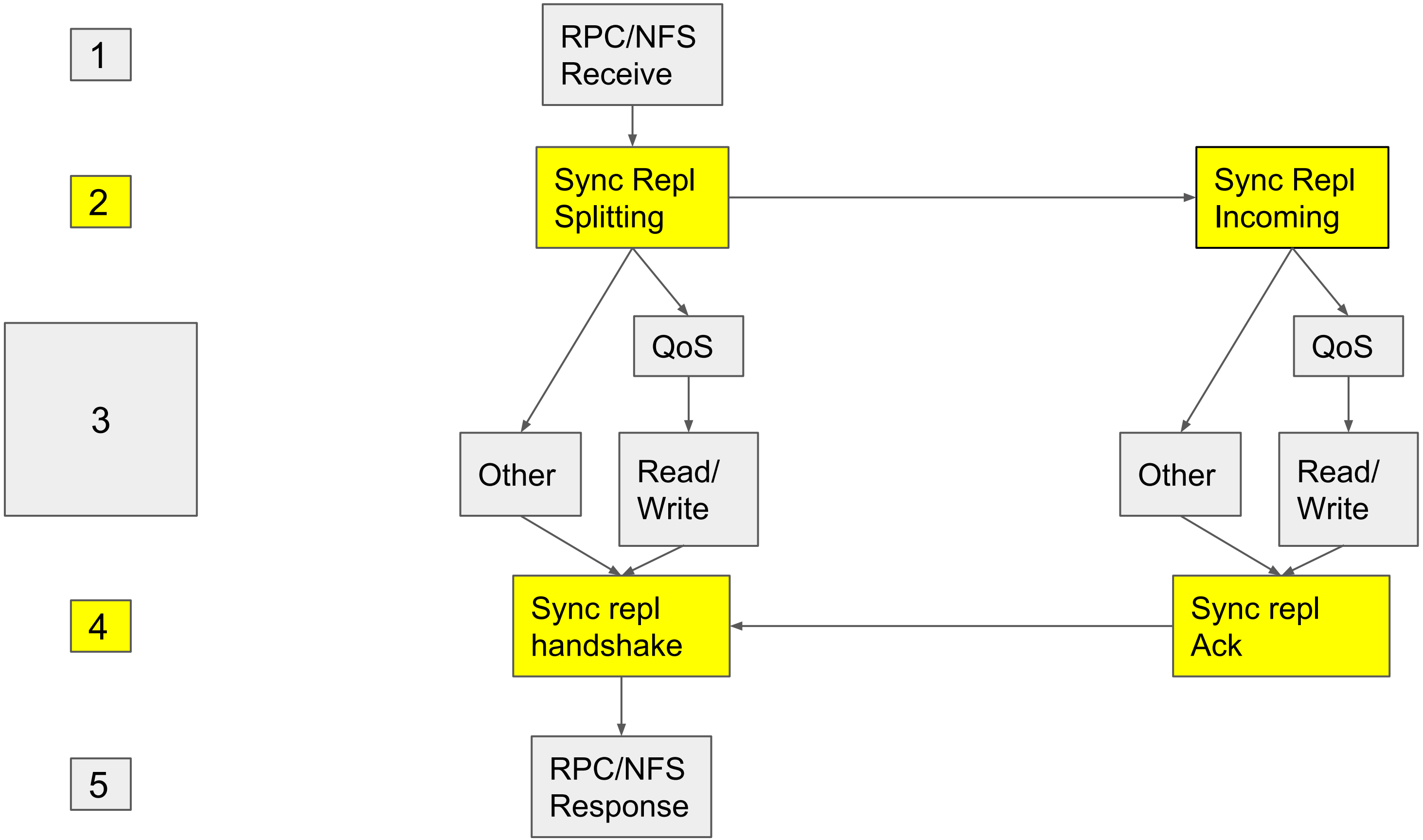
# Replicating writes

Objective: minimize added latency

Requirements:

1. Never lose acknowledged writes.
2. Recover from crashes, disconnects, etc and converge to an identical state.

# Filesystem Write Path



# Bookkeeping for in-flight writes

Objective: keep track of in-flight writes persistently for crash recovery

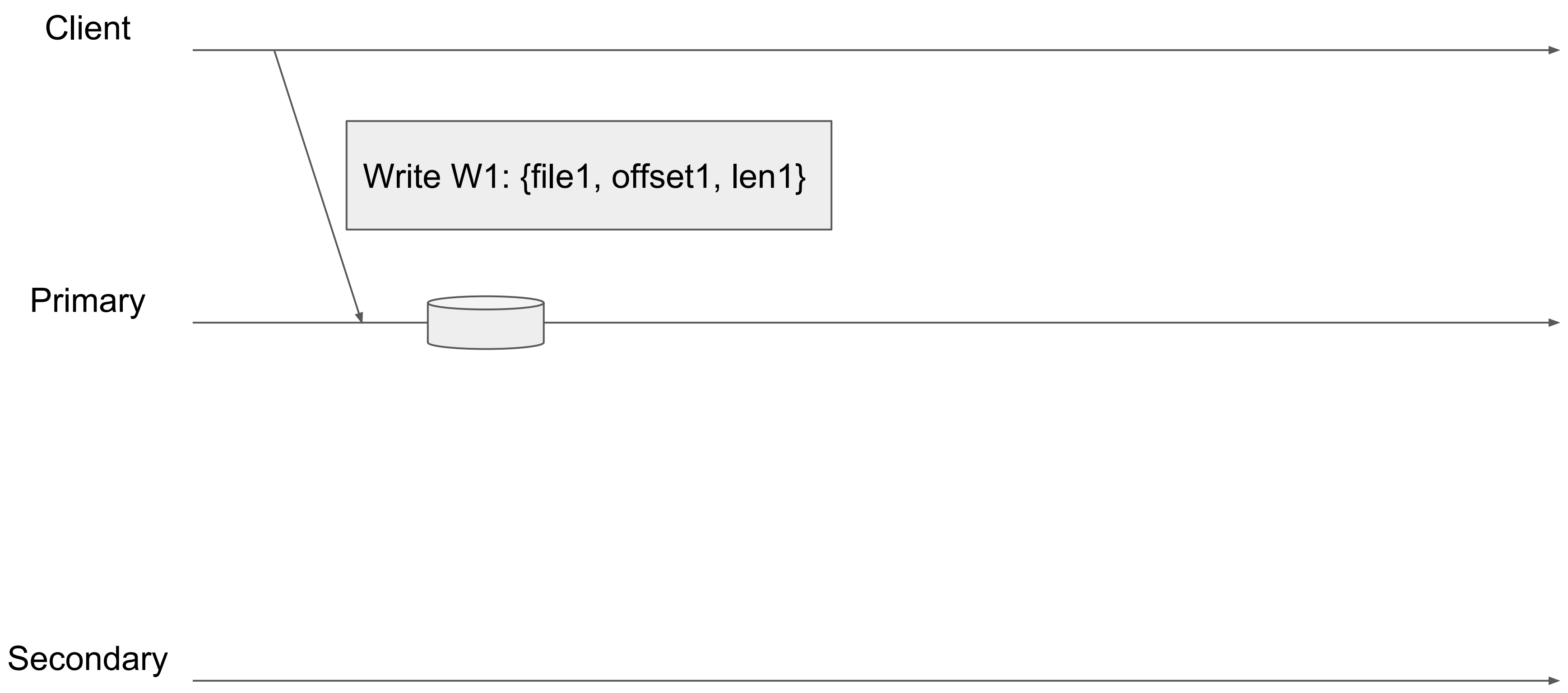
All operations tagged with sequence number (OSN)

NVRAM entries: data, plus <FileId, Offset, Length, OSN>

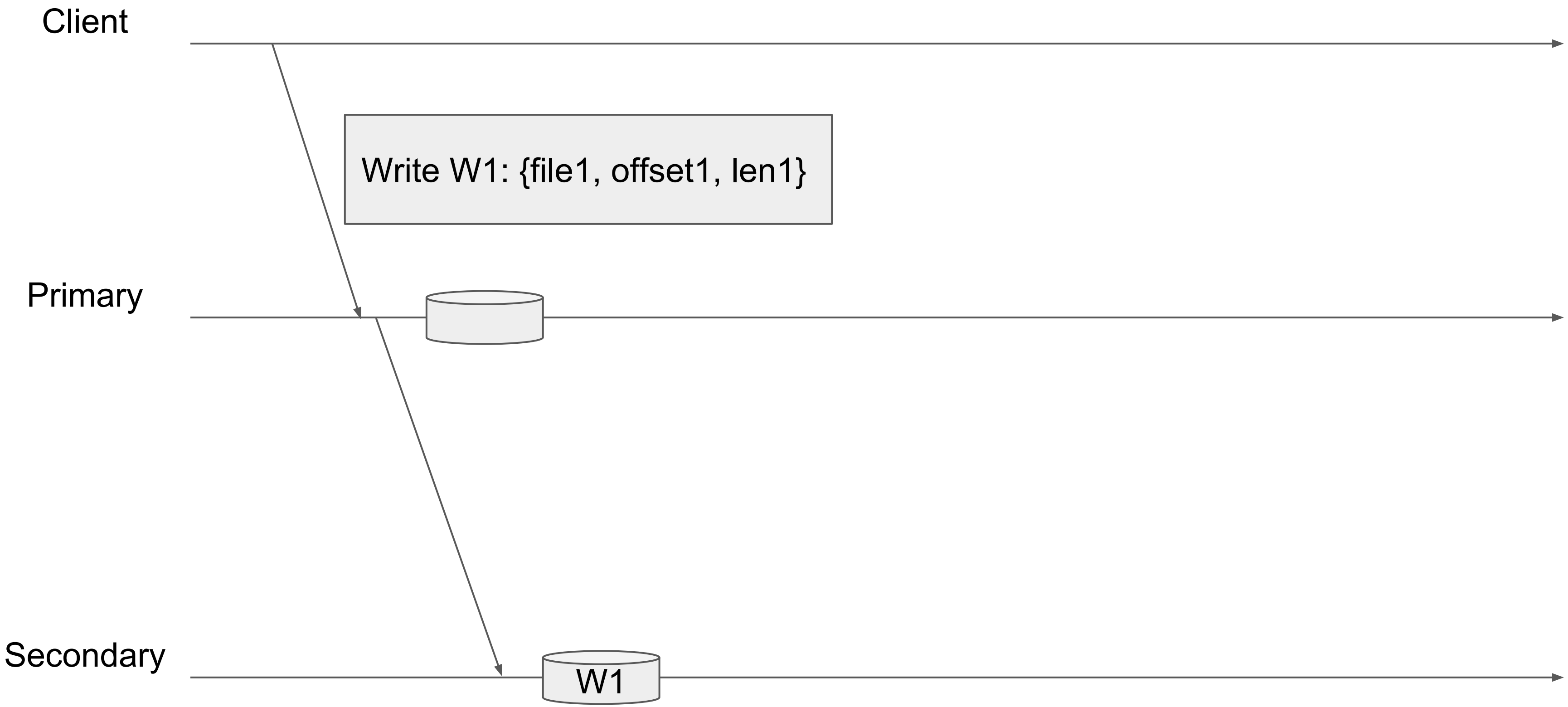
Persistent metadata.  For each replicated directory, keep a map of

OSN → <FileId, Offset, Length>

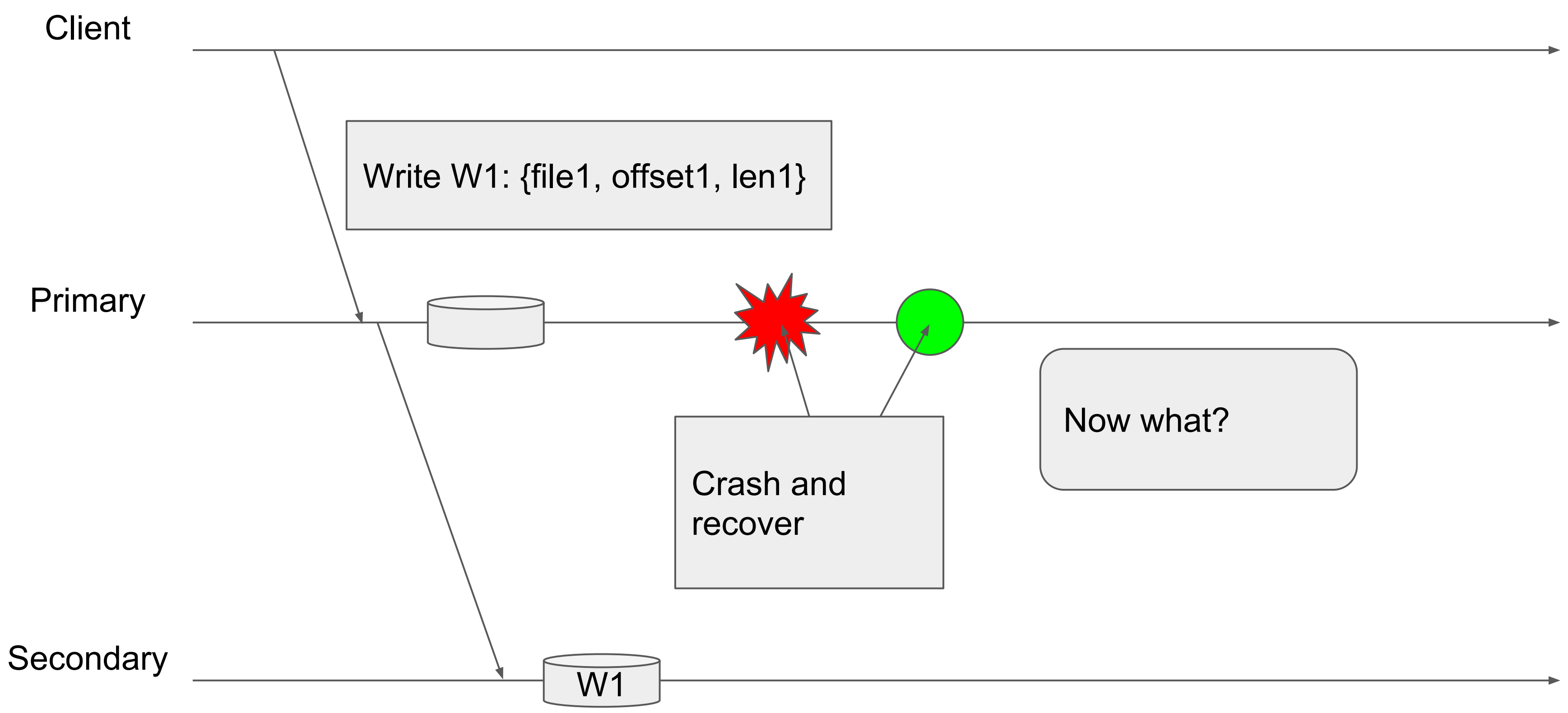
# Crash Recovery Example



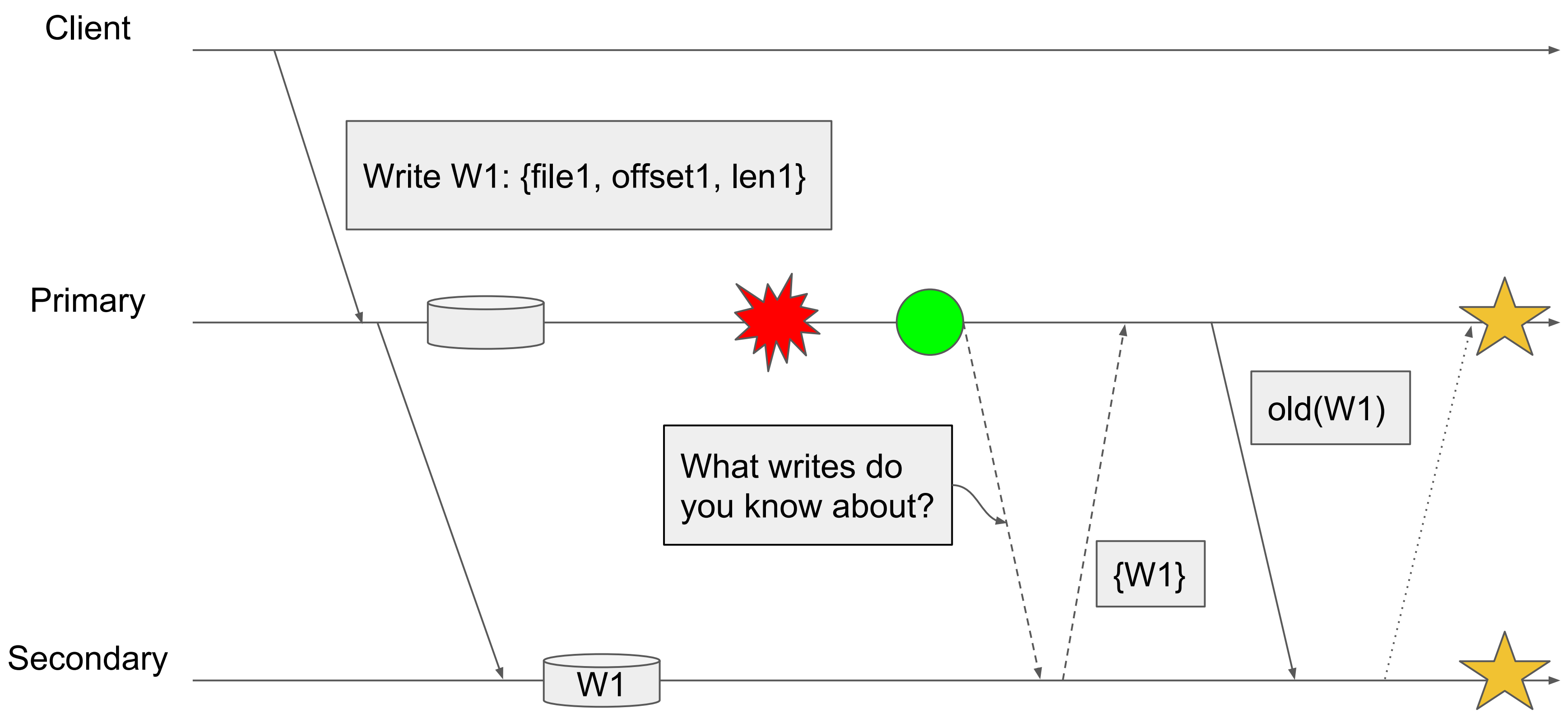
# Crash Recovery Example (2)



# Crash Recovery Example (3)



# Crash Recovery Example (4): Distributed Recovery

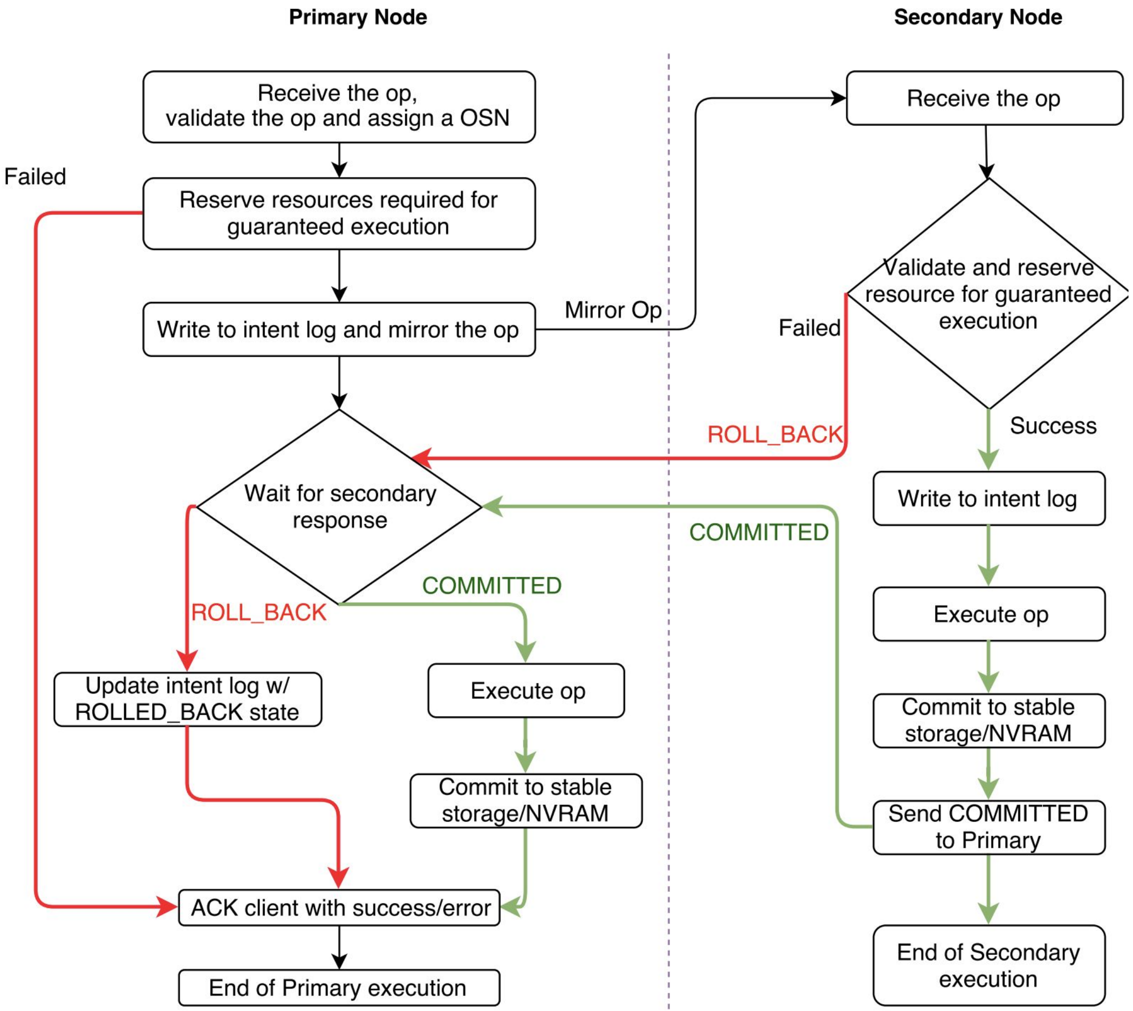


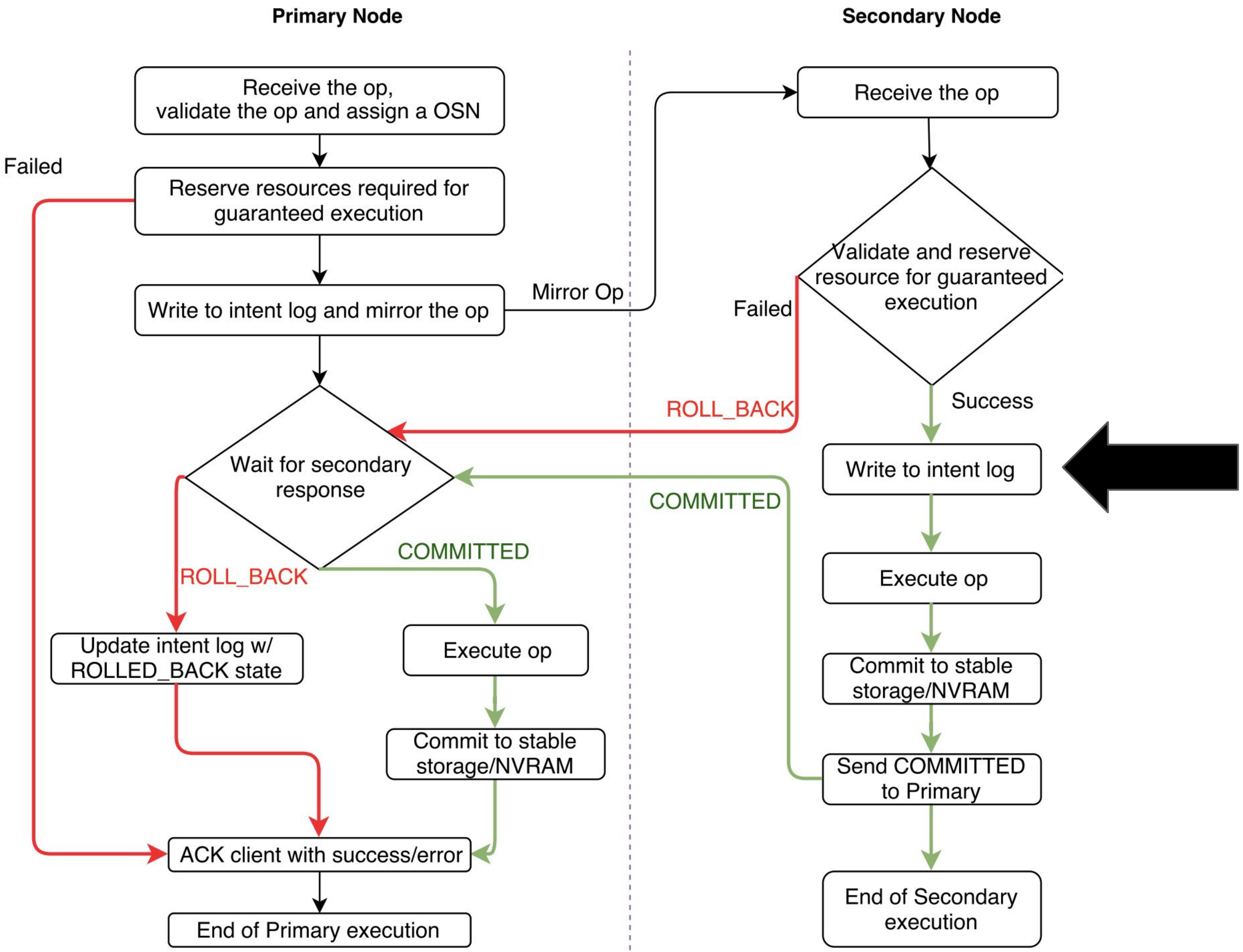
# Data Path 2: Operations Other Than Writes

Undo not possible (e.g., deletes & renames)

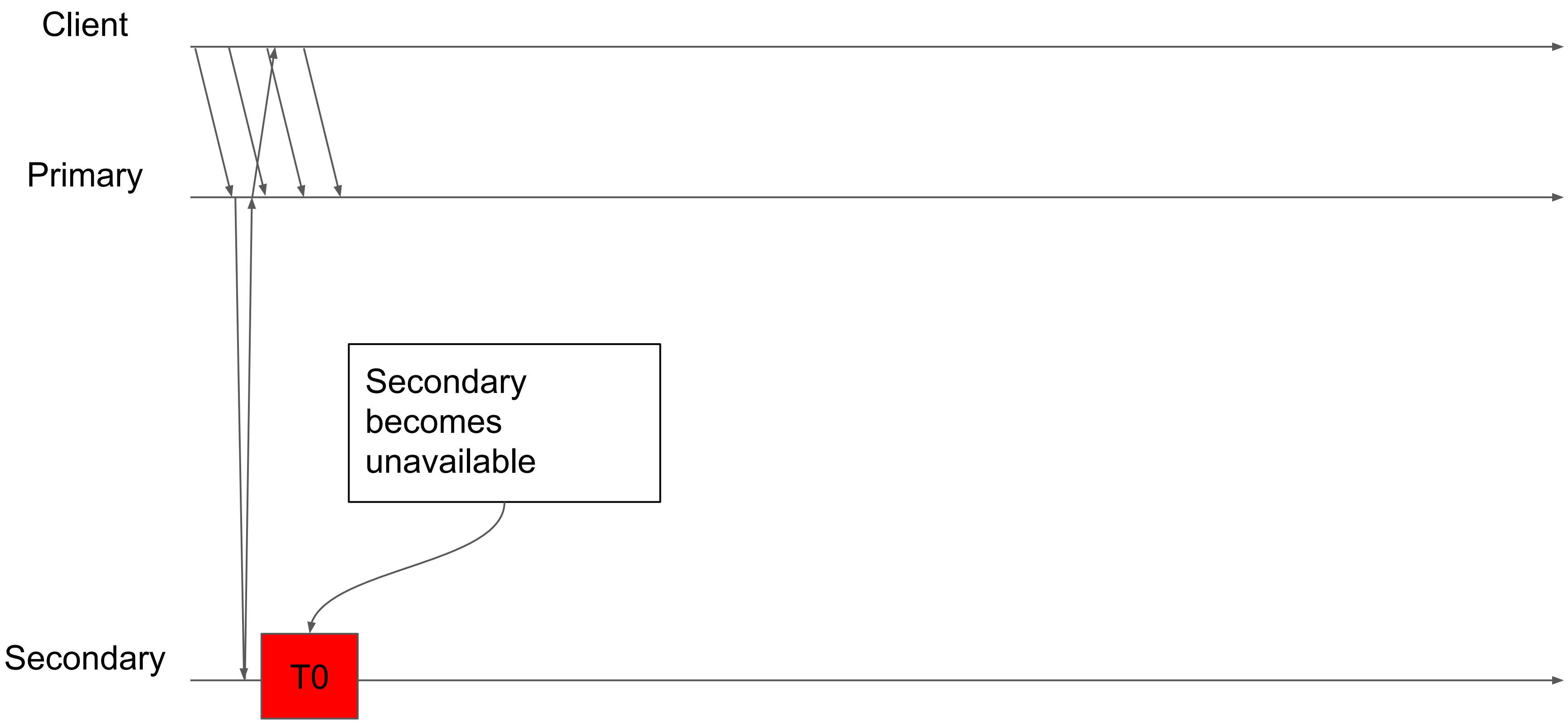
Less frequent: can afford higher overhead

Approach: two-phase commit

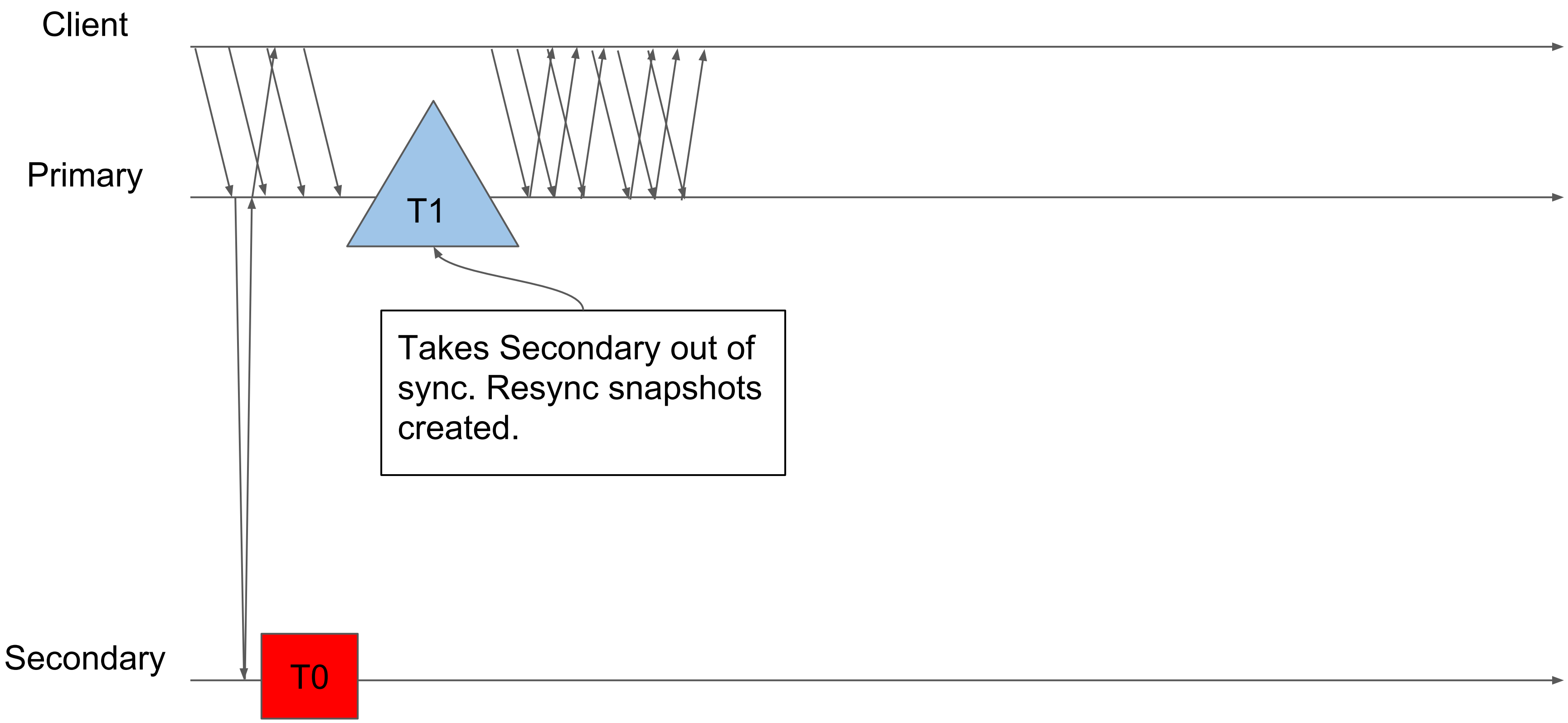

Primary Node
Secondary Node
Receive the op, validate the op and assign a OSN
Failed
Reserve resources required for guaranteed execution
Write to intent log and mirror the op
Mirror Op
Receive the op
Validate and reserve resource for guaranteed execution
Failed
Success
ROLL_BACK
Wait for secondary response
COMMITTED
Write to intent log
COMMITTED
ROLL_BACK
Execute op
Update intent log w/ ROLLED_BACK state
Execute op
Commit to stable storage/NVRAM
Commit to stable storage/NVRAM
Send COMMITTED to Primary
ACK client with success/error
End of Primary execution
End of Secondary execution


Primary Node
Secondary Node
Receive the op,
validate the op and assign a OSN
Failed
Reserve resources required for
guaranteed execution
Write to intent log and mirror the op
Mirror Op
Receive the op
Validate and reserve
resource for guaranteed
execution
Failed
ROLL_BACK
Success
Write to intent log
Wait for secondary
response
COMMITTED
COMMITTED
ROLL_BACK
Execute op
Update intent log w/
ROLLED_BACK state
Execute op
Commit to stable
storage/NVRAM
Commit to stable
storage/NVRAM
Send COMMITTED
to Primary
ACK client with success/error
End of Secondary
execution
End of Primary execution

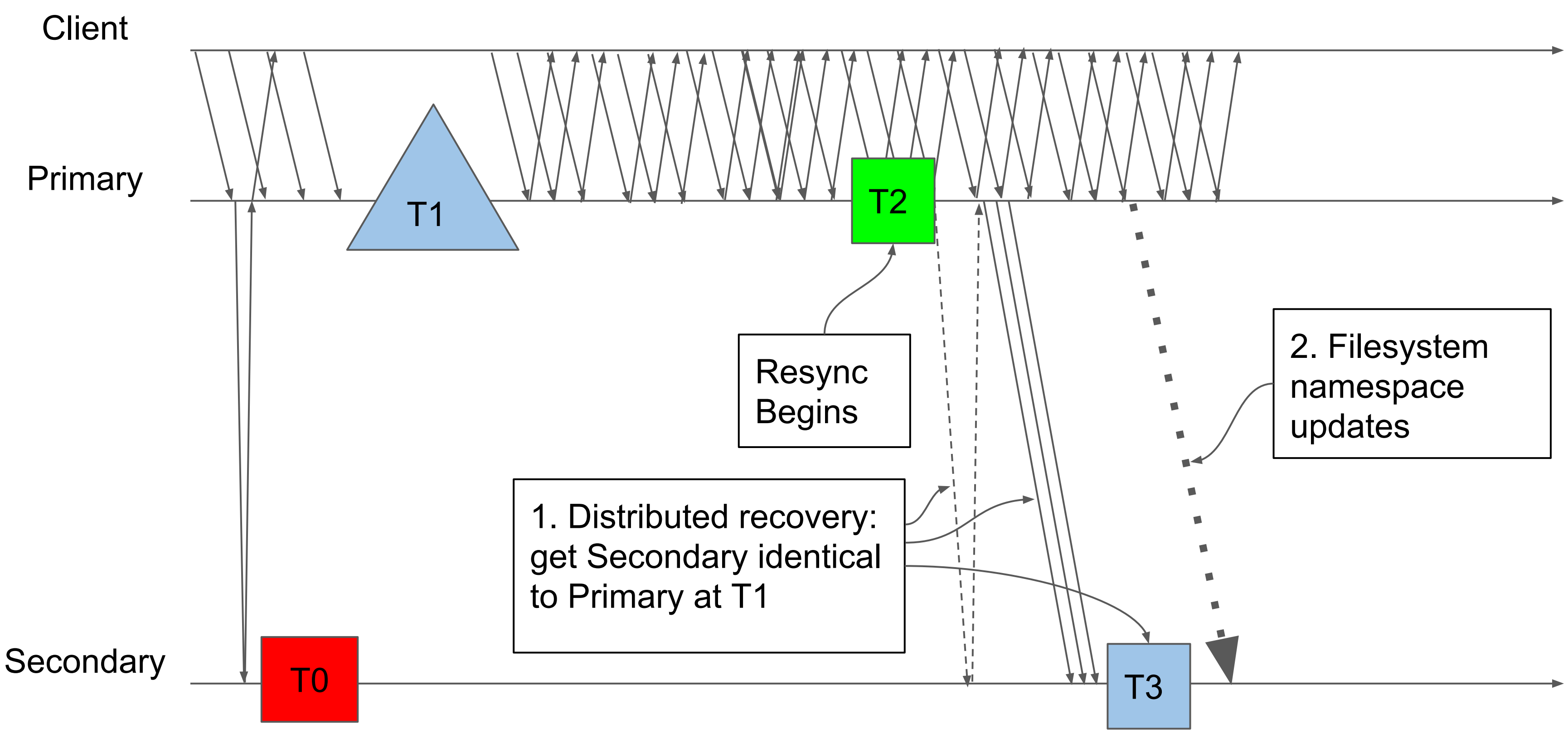
# Data Path 3: Resync



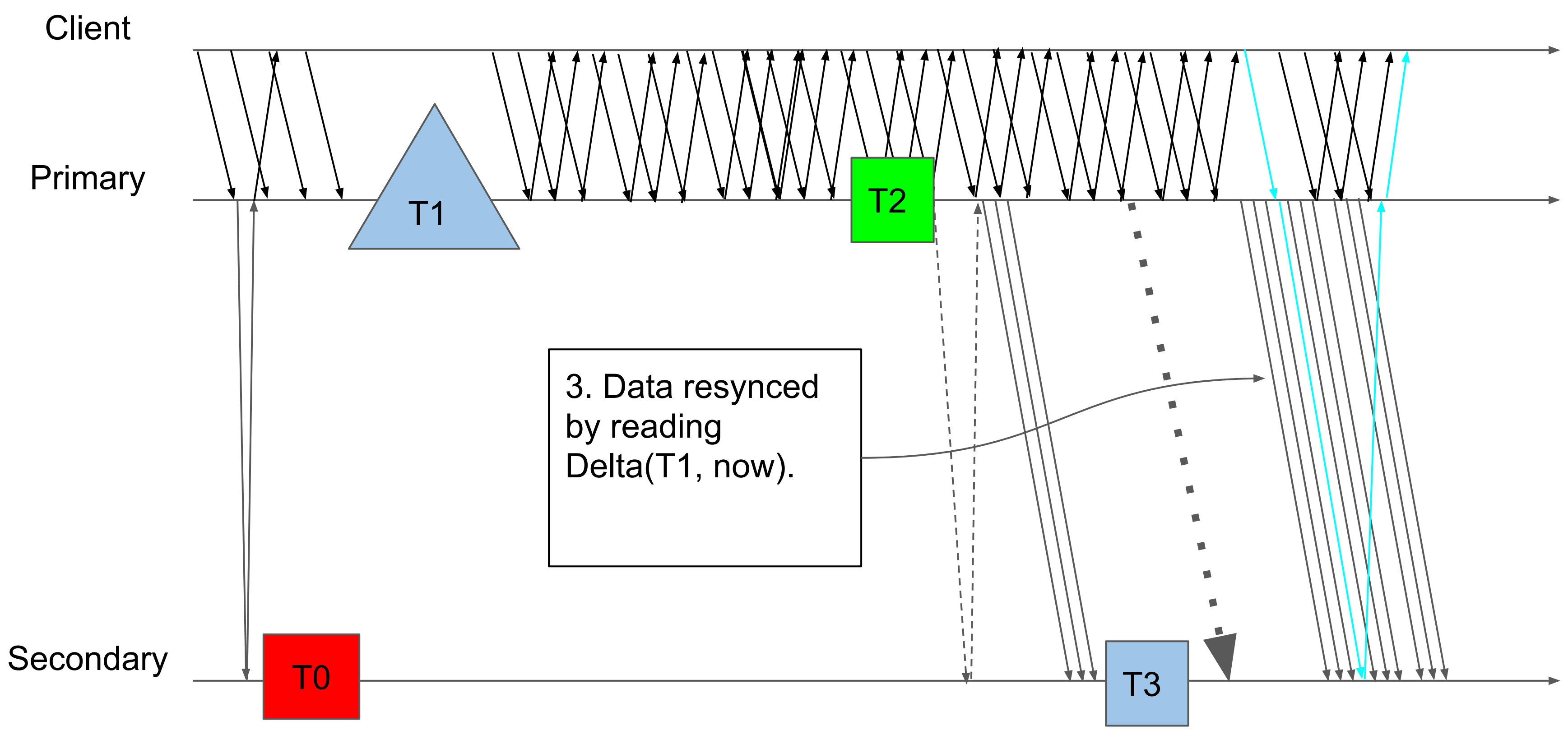
# Resync Timeline



# Resync Timeline



# Resync Timeline



# Data Path 4: Distributed integrity checking

Goal: verify that Primary & Secondary have identical content

Leverage existing per-file content checksum

Periodically and on demand (e.g. after resync):

1. Primary: temporarily pause incoming I/Os
2. Primary, Secondary: quiesce in-flight I/Os
3. For each file,
    a. Primary: extract logical file content checksum; send the checksum to Secondary
    b. Secondary: extract local checksum; compare with value from Primary
        i. If different, preserve state & take Secondary out of sync

# Lessons Learned

# Lessons Learned

Integrity check: invaluable

# Lessons Learned

Integrity check: invaluable

Automatic cluster failover: more important to customers than anticipated

# Lessons Learned

Integrity check: invaluable

Automatic cluster failover: more important to customers than anticipated

Ease of use & flexibility: well received

# Questions

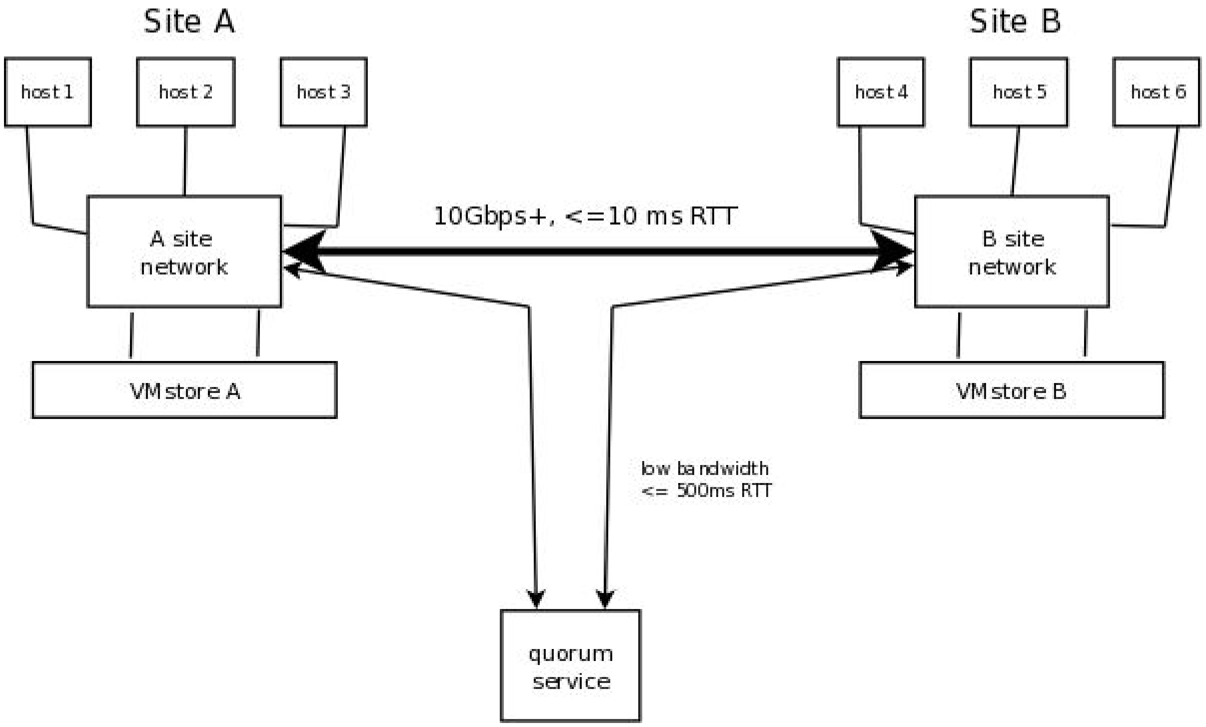
# Backup Slides

# Example VM

```
[it-ttha1#b]$ cd hg
[it-ttha1#b]$ ls -latr
total 139272640
-rw------- 1 root root           13 2014-12-10 16:51 hg-aux.xml
-rw------- 1 root root       145496 2014-12-10 16:51 vmware-23.log
-rw------- 1 root root      8545125 2014-12-10 16:51 vmware-22.log
-rw-r--r-- 1 root root          257 2014-12-10 16:51 hg.vmxf
-rw------- 1 root root  34359738368 2014-12-10 16:51 hg-a0cbeddf.vswp
-rw------- 1 root root   2147483648 2014-12-10 16:54 hg_2-flat.vmdk
-rw-r--r-- 1 root root       187947 2014-12-10 17:06 vmware-24.log
-rw-r--r-- 1 root root     17454622 2015-04-20 15:28 vmware-25.log
-rw-r--r-- 1 root root      8215263 2015-06-08 15:27 vmware-26.log
-rw------- 1 root root    117440512 2015-06-19 16:27 vmx-hg-2697719263-1.vswp
-rw------- 1 root root            0 2015-06-19 16:27 hg.vmx.lck
-rw-r--r-- 1 root root           73 2015-06-19 16:27 hg-a0cbeddf.hlog
-rw-r--r-- 1 root root      2767818 2015-06-19 16:27 vmware-27.log
-rw------- 1 root root         8684 2015-07-06 14:58 hg.nvram
-rw------- 1 root root          594 2015-12-22 09:54 hg_2.vmdk
-rwxr-xr-x 1 root root         4148 2015-12-22 09:54 hg.vmx
-rw-r--r-- 1 root root           45 2015-12-22 09:54 hg.vmsd
-rw------- 1 root root       983552 2015-12-22 09:54 hg-ctk.vmdk
-rw------- 1 root root      6554112 2015-12-22 09:54 hg_4-ctk.vmdk
-rw------- 1 root root      4260352 2015-12-22 09:54 hg_3-ctk.vmdk
-rw------- 1 root root       131584 2015-12-22 09:54 hg_2-ctk.vmdk
-rw------- 1 root root      2294272 2015-12-22 09:54 hg_1-ctk.vmdk
-rw------- 1 root root          592 2015-12-22 09:54 hg.vmdk
-rw------- 1 root root          596 2015-12-22 09:54 hg_1.vmdk
-rw------- 1 root root          597 2015-12-22 09:54 hg_3.vmdk
-rw------- 1 root root          572 2015-12-22 09:54 hg_4.vmdk
drwxr-xr-x 1 root root            0 2016-05-20 09:33 .
drwxrwxrwx 1 root root            0 2016-05-20 11:57 ..
-rw-r--r-- 1 root root     12546260 2016-05-27 11:05 vmware.log
-rwxrwxr-x 1 root root           84 2016-05-29 15:17 .lck-788c100000000000
-rwxrwxr-x 1 root root           84 2016-05-29 15:17 .lck-678c100000000000
-rwxrwxr-x 1 root root           84 2016-05-29 15:17 .lck-52df090000000000
-rwxrwxr-x 1 root root           84 2016-05-29 15:17 .lck-47df090000000000
-rwxrwxr-x 1 root root           84 2016-05-29 15:17 .lck-3cdf090000000000
-rwxrwxr-x 1 root root           84 2016-05-29 15:17 .lck-31df090000000000
-rwxrwxr-x 1 root root           84 2016-05-29 15:17 .lck-29600f0000000000
-rwxrwxr-x 1 root root           84 2016-05-29 15:17 .lck-26df090000000000
-rw------- 1 root root 214748364800 2016-05-29 15:17 hg_4-flat.vmdk
-rw------- 1 root root  69793218560 2016-05-29 15:17 hg_3-flat.vmdk
-rw------- 1 root root  37580963840 2016-05-29 15:17 hg_1-flat.vmdk
-rw------- 1 root root  16106127360 2016-05-29 15:17 hg-flat.vmdk
[it-ttha1#b]$
```

# Network topology

Customer must ensure that hosts at both sites can reach whichever VMstore is Primary, i.e. whichever one exposes the Cluster IP address for a given replicated directory.  Typically this is done via a stretched L2 or L3 network.



# Configuration Example

| **Datastores hosted by a given VMstore** | **Replicated?** |
|---|---|
| 10.10.10.5:/tintri | No |
| 10.10.10.6:/tintri/alpha | Yes |
| 10.10.10.7:/tintri/beta | Yes |

# Resync Semantics & Requirements

Primary takes Secondary out of sync if Secondary is down for >= ~30s

Primary continues to serve data and accept writes while out of sync

When Secondary comes back:

- Replicate only new/changed data
- Must discover & read this data efficiently
- Must converge reasonably quickly

# Resync algorithm

Primary decides to take Secondary out of sync, coordinates with quorum service to do this

Primary creates *resync snapshots* on all files in replicated directory

*time passes…*

When Secondary comes back,

1. Do distributed recovery; data comes from resync snapshots, not current content
2. Resync filesystem directory namespace, including deletions
3. Replicate changes to files that have resync snapshots
    a. Content between (resync snapshot creation, current time) can be efficiently obtained
    b. Do this in order of FileId's ordered by creation time
    c. increasing logical byte offset within each file
4. Simultaneously, synchronously replicate non-writes, and writes to files that are either
    a. already resynced, or
    b. created after resync began

# Performance (1)

| **Workload** | **8KiB Writes** | **64KiB Writes** | **256Kib Writes** |
|---|---|---|---|
| **Throughput reduction with sync repl** | 43% | 11% | 6% |

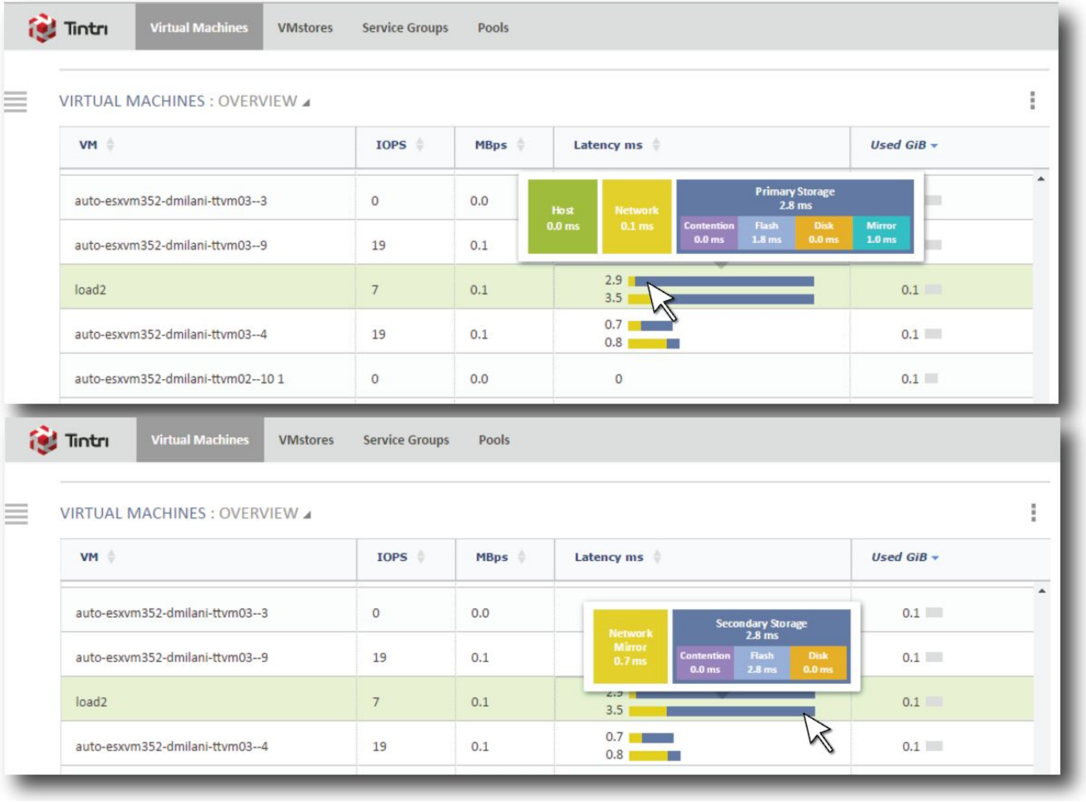
Disclaimer: old hardware; software may have improved subsequent to taking these measurements

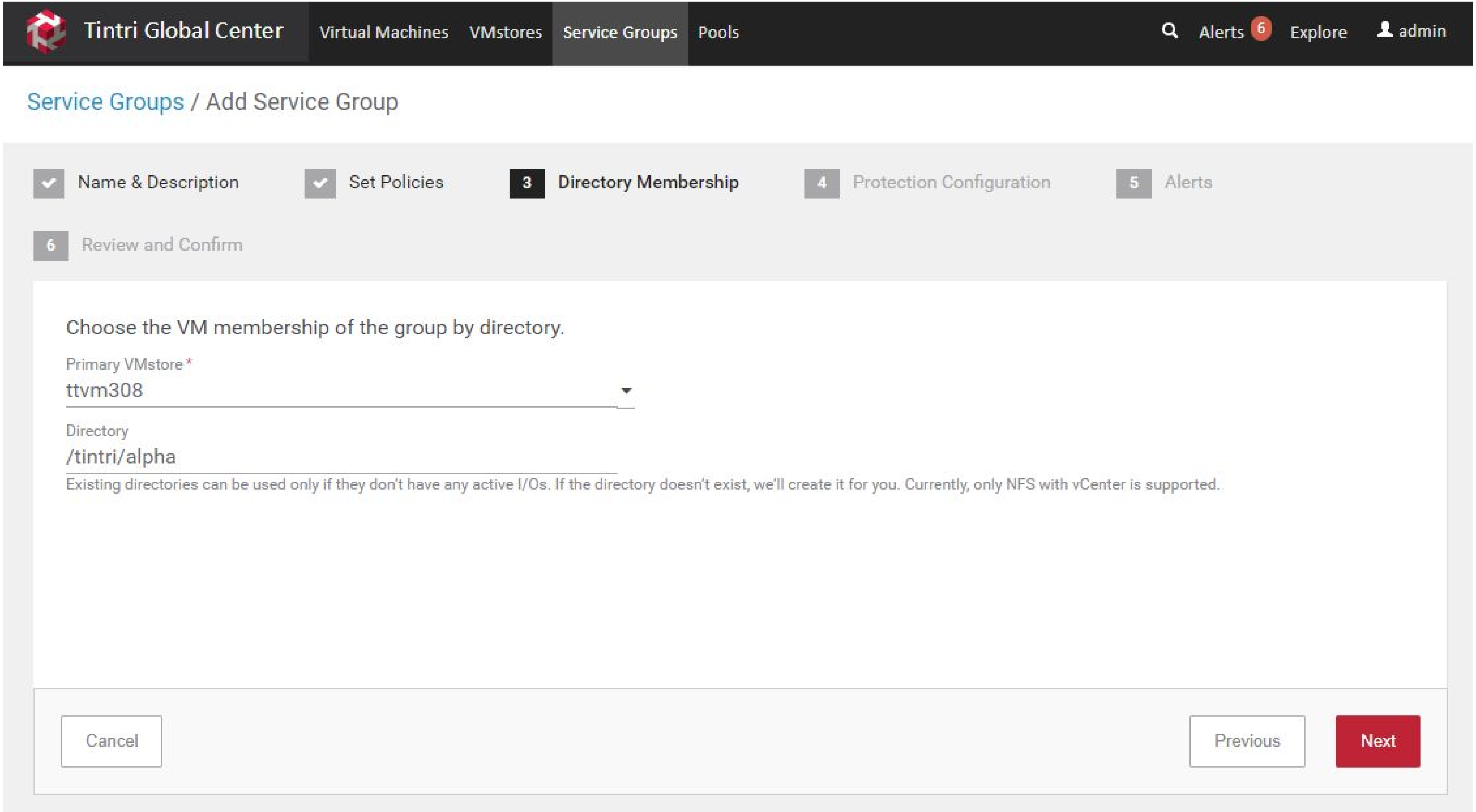
# Performance (2)

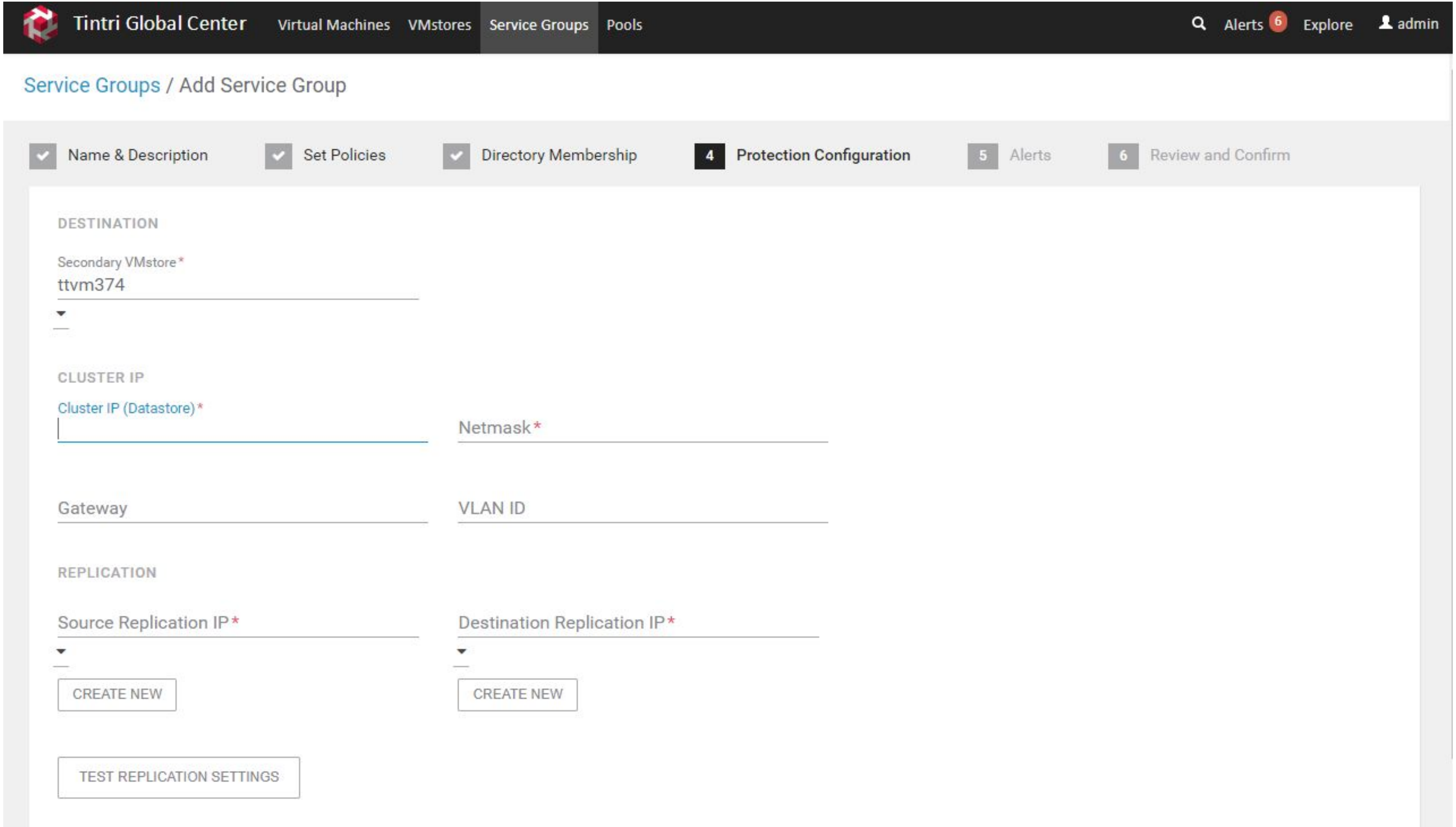
Incremental improvements already made (~ +60% improvement over unoptimized starting point for 8K writes):

- vectorized socket I/O (mainly write side)
- socket reads into large buffers, rather than per-message buffers.  Generally, remove mallocs on network input processing code paths
- increase thread priority of thread(s) reading from replication sockets.  [There are many threads in the system, around 1,000, and not many replication socket reading threads.]
- more could be done

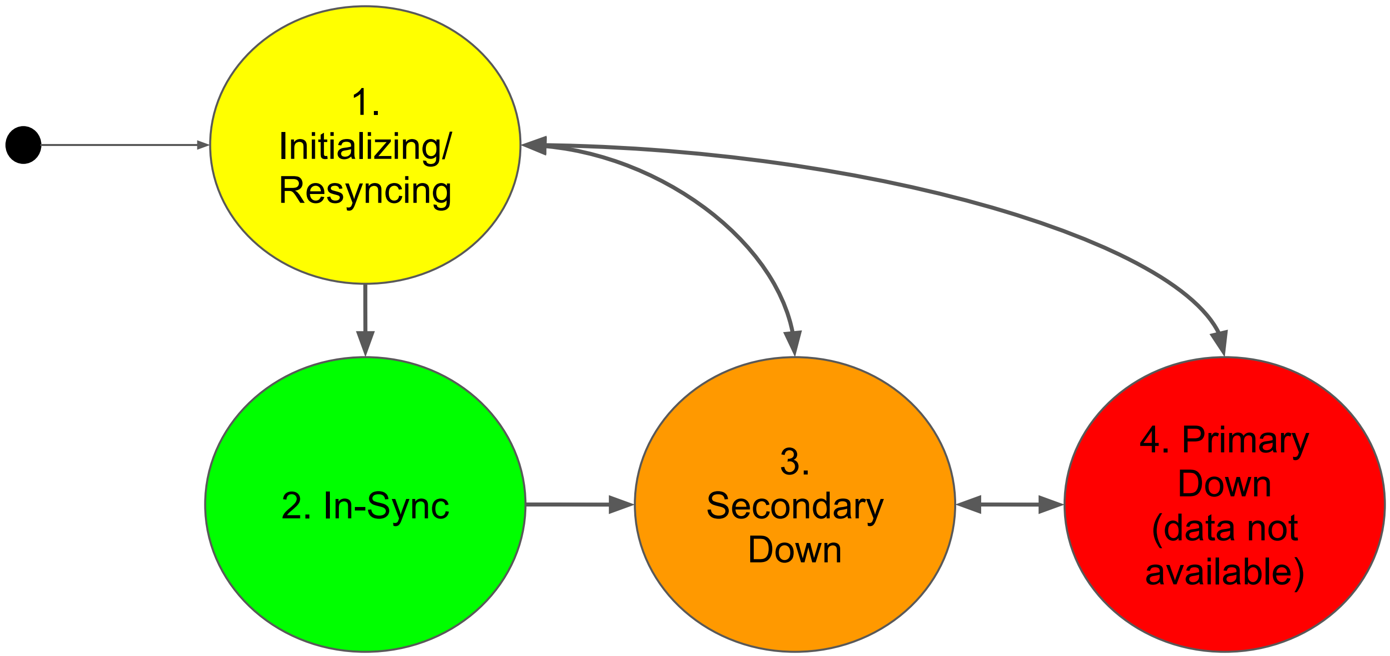
# Latency Visualization




Tintri Global Center
Virtual Machines
VMstores
Service Groups
Pools
Alerts 6
Explore
admin
Service Groups / Add Service Group
Name & Description
Set Policies
3 Directory Membership
4 Protection Configuration
5 Alerts
6 Review and Confirm
Choose the VM membership of the group by directory.
Primary VMstore *
ttvm308
Directory
/tintri/alpha
Existing directories can be used only if they don't have any active I/Os. If the directory doesn't exist, we'll create it for you. Currently, only NFS with vCenter is supported.
Cancel
Previous
Next


Tintri Global Center
Virtual Machines
VMstores
Service Groups
Pools
Alerts 6
Explore
admin
Service Groups / Add Service Group
Name & Description
Set Policies
Directory Membership
4 Protection Configuration
5 Alerts
6 Review and Confirm
DESTINATION
Secondary VMstore *
ttvm374
CLUSTER IP
Cluster IP (Datastore) *
Netmask *
Gateway
VLAN ID
REPLICATION
Source Replication IP *
Destination Replication IP *
CREATE NEW
CREATE NEW
TEST REPLICATION SETTINGS

# Conceptual State Machine: Data Availability



# VMstore file-level snapshots