HPE POINTNEXT

Volume Shadowing Best Practices

OpenVMS Boot Camp 2017, SID 304

Keith Parris, Engineer

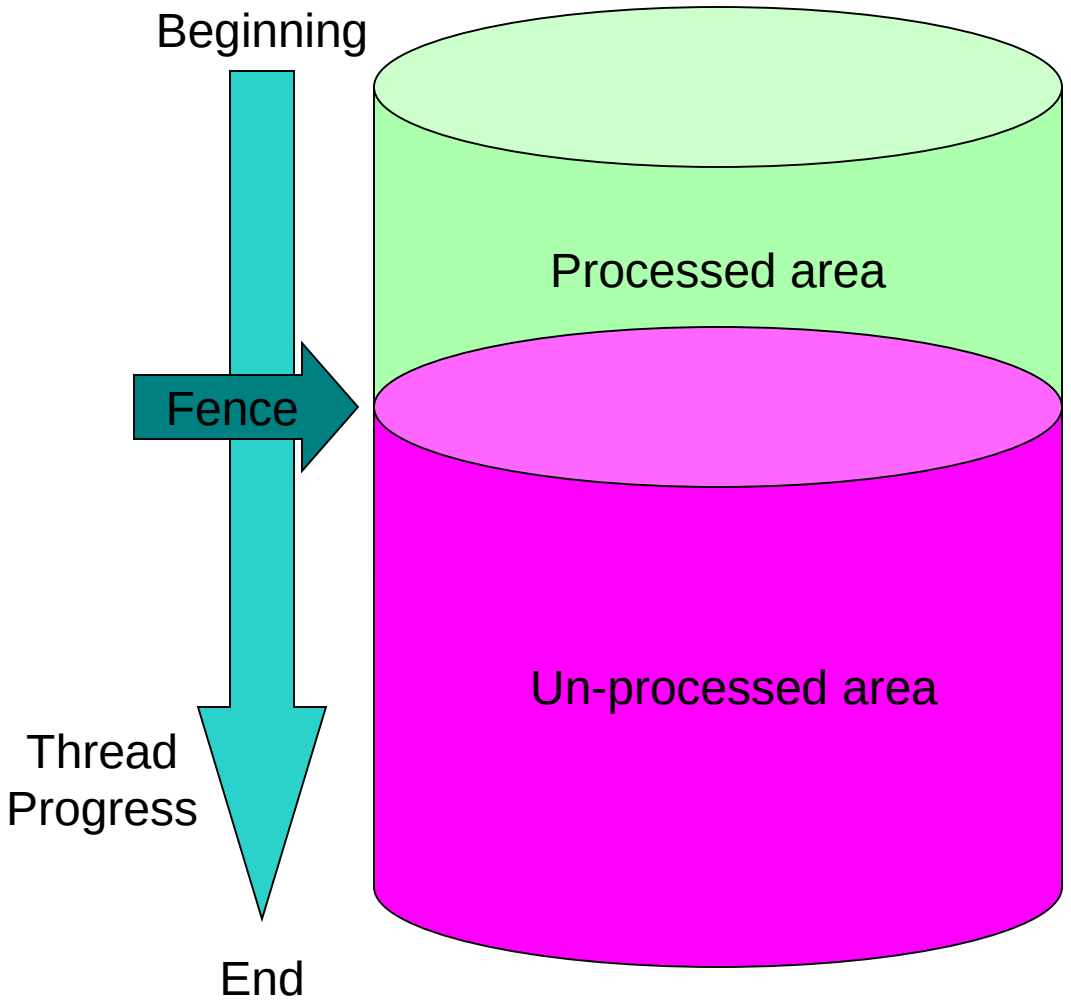
Hewlett Packard Enterprise

Shadowing Full-Copy and Full-Merge Algorithms

– Start at first Logical Block on disk (LBN zero)

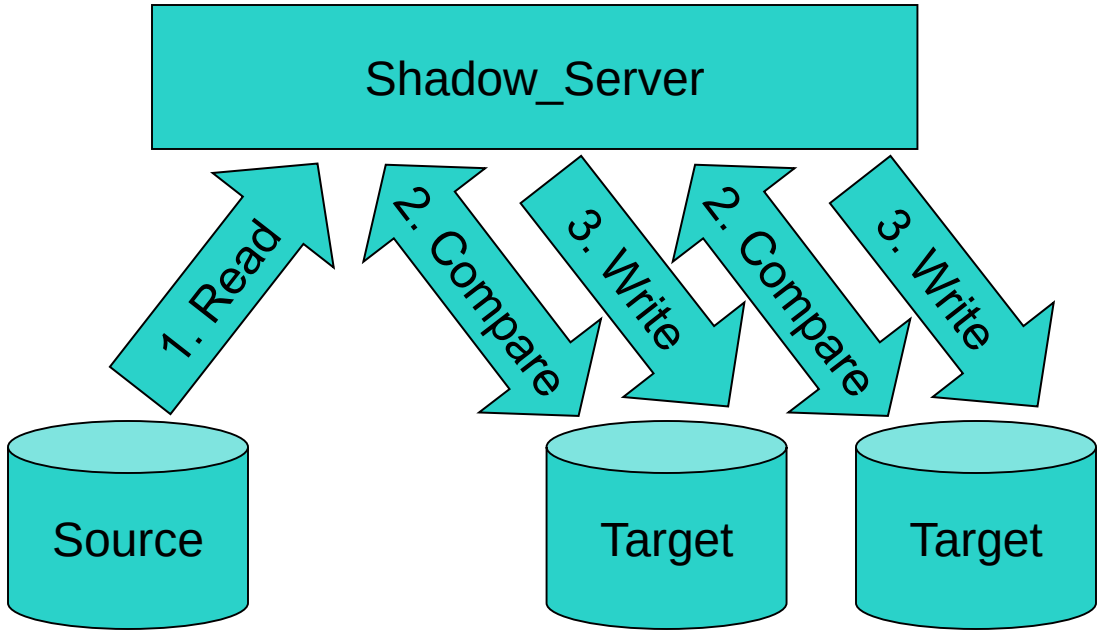
– Process 127 blocks at a time from beginning to end

– Symbolic “Fence” separates processed area from un-processed area



Shadowing Full-Copy Algorithm

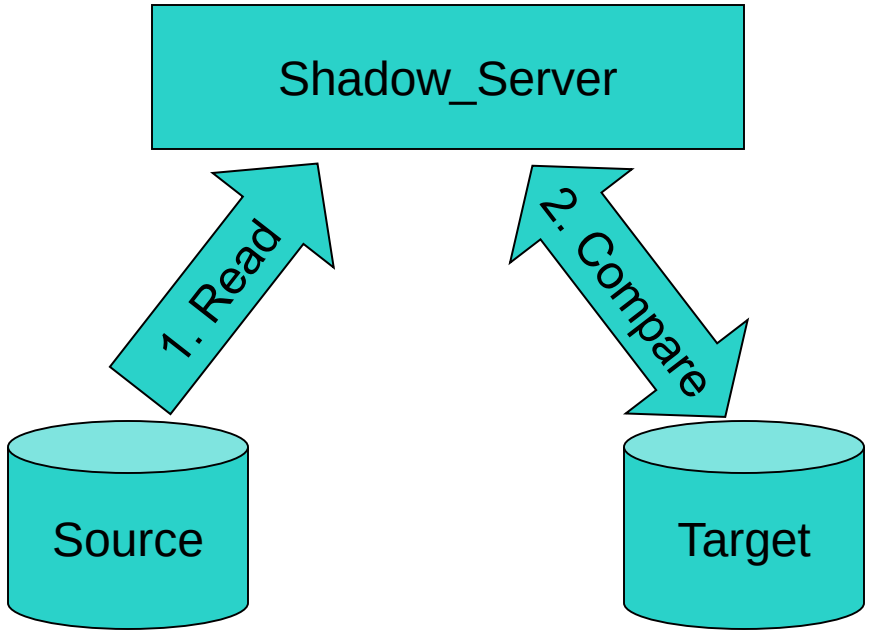
1. Read from (a) source member
2. Compare with target member(s)
3. If different, write data to target and start over at Step 1.



Shadowing Full-Copy Algorithm

Data identical

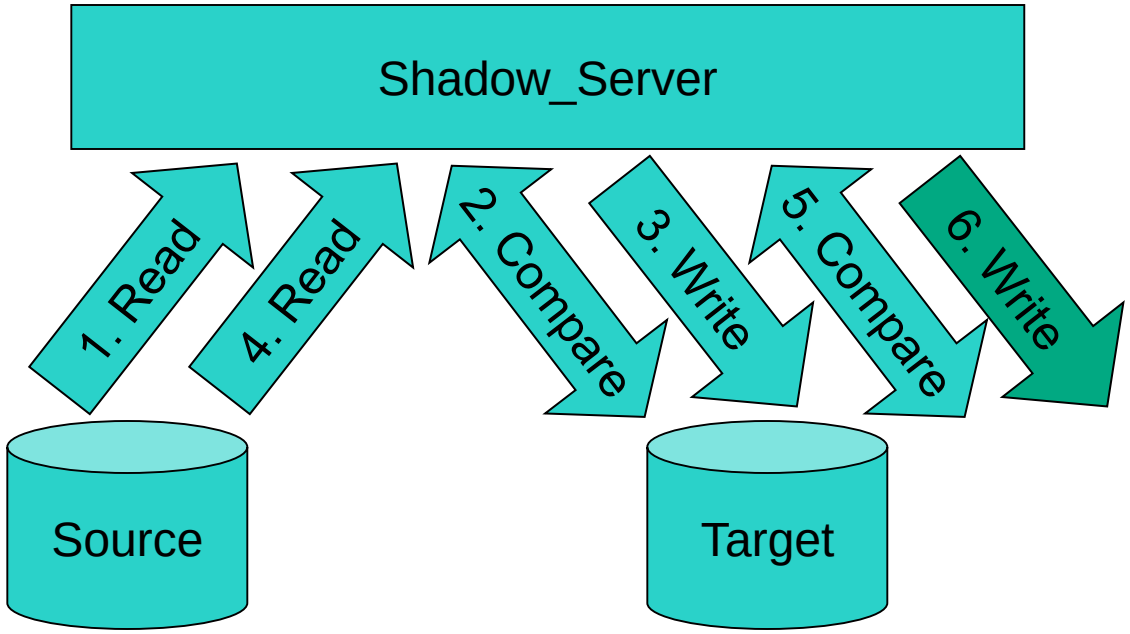
1. Read from source
2. Compare with target (matches, so done)



Shadowing Full-Copy Algorithm

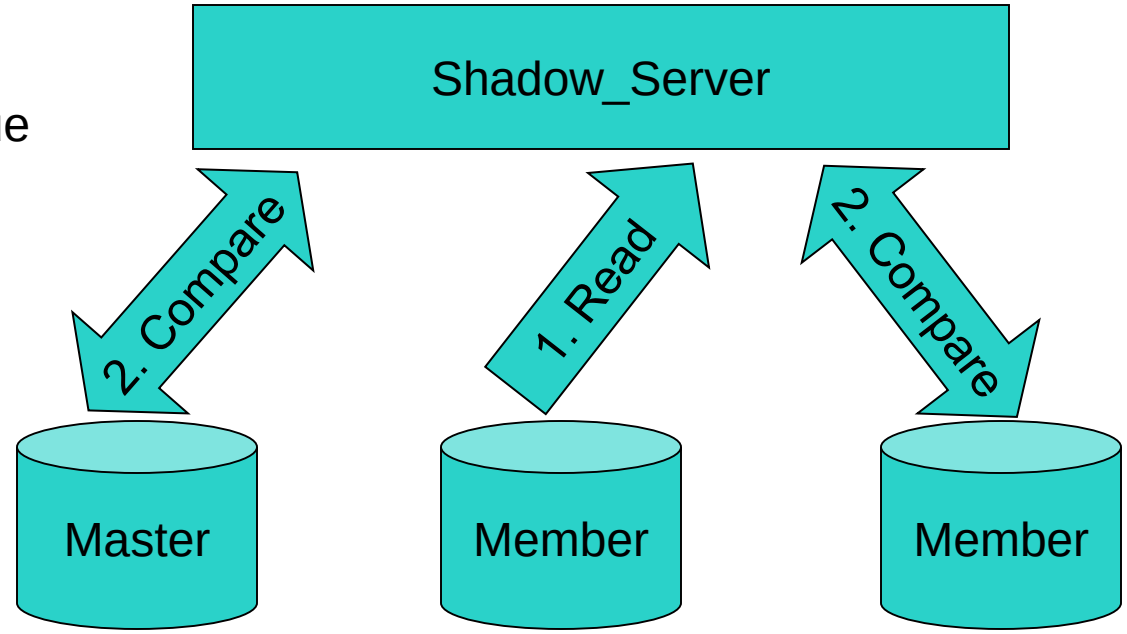
Data different

1. Read from source
2. Compare with target (difference found)
3. Write to target
4. Read from source
5. Compare with target (difference seldom found). If different, write to target and start over at Step 4.



Shadowing Full-Merge Algorithm

1. Read from any member
2. Compare with other member(s)
3. If different, do a Fix-Up: halt all I/Os to the shadowset, fix up differences using data from the Master member, then allow I/Os to continue



General Volume Shadowing Best Practices for Performance

- Because shadow copy/merge threads operate from the start to the end of the disk in 127-block increments:
  - If you want to protect a given amount of data using Shadowing, the shadow copies will tend to complete faster if you have more of smaller volumes than if you have fewer, larger volumes.
  - Data is not redundant until all shadow copies complete, so avoid a single or a few huge volumes and the test smaller
- Because copies and merges are faster when the data is identical:
  - If you have a new disk to add to a shadowset, it's actually faster in elapsed time to do a $ BACKUP /PHYSICAL from the DSAnnn: device to the target disk (mounted /FOREIGN) and then add it to the shadowset afterward, than to simply add it to the shadowset while it still contains mostly-different data

Write Bitmaps

– Mini-Merges and Mini-Copies take advantage of Write Bitmaps – data structures residing in memory
  – These bitmaps record writes to each 127-block increment of blocks on a disk
  – When a node leaves the cluster, it takes away with it any write bitmaps it had in its memory
  – If you need to use a Write Bitmap either for a Mini-Copy or a Mini-Merge, one of these needs to survive on a node which remains up in the cluster after a failure
  – In a multi-site disaster-tolerant cluster, you can lose an entire site, so you need Write Bitmaps to always survive on nodes at the surviving site

Write Bitmaps

- When Write Bitmaps are in use, before an actual write occurs to a shadowset, any associated Write Bitmap(s) are updated
  - This implies messages need to pass between nodes, with potential adverse performance impact
    - As a performance optimization, we keep a Local Bitmap for each remote Master Write Bitmap, and if we find a bit already sent, indicating we've already sent notification for a change to that 127-block segment, we don't need to send another notice
  - In 8.3-1H1 and earlier, these write bitmap updates occurred sequentially, one write bitmap at a time
  - In 8.4 these updates were done in parallel for increased performance, but obscure cluster hang problems caused this code to revert to the pre-8.4 code in post-8.4 patch kits VMS84A_SYS-V0600 and VMS84I_SYS-V0600. The release notes said:

    "An OpenVMS V8.4 cluster hang had been seen occasionally during system shutdown or boot while using host-based mini-merge (HBMM) for the system disk shadow set, shared by multiple systems. This was occurring during a bitmap update operation when a write bitmap message got lost and the device remained in a write locked state.

    In OpenVMS V8.3-1H1 and in previous versions, master bitmap update messages were sent one at a time, updating each remote master bitmap sequentially. In OpenVMS V8.4, these remote master bitmap update operations were made parallel. We have reverted back the code that performs parallel bitmap updates to sequential.

    This problem has been fixed."

Full Merges and Mini-Merges

Why do Full Merges hurt so much?

- When Full-Merges were first designed, developers thought the merge I/Os would be the primary bottleneck. In the real world, the bottleneck is reads ahead of the merge fence, where we must merge the area of the read before returning the data to the user, and do this for each and every I/O until the merge fence passes over the hot area. This is where the pain comes from.
- Conclusions:
  - Get Full Merges over as quickly as possible. Don't throttle back the merge thread – that just prolongs the pain.
  - Consider putting hot files at the beginning of the disk if possible, so the merge fence can get past them quicker.
- Set up a policy and use Mini-Merges instead of Full Merges whenever possible.

How many Mini-Merge Write Bitmaps should I have?

– Single cluster: Probably two. Probably not six.

– Two-site Cluster: Probably four (2 per site)

– Three-site Cluster with storage at all 3 sites: Probably six (2 per site)

Recently-added Volume Shadowing features

– Automatic Mini-Copy on Volume Processing (AMCVP) Feature:
  – MULTIUSE keyword in Mini-Merge policy: Automatically converts a Mini-Merge Write Bitmap to Multi-Use (both Mini-Merge and Mini-Copy) Write Bitmaps so the removed shadowset members can be re-added later with a Mini-Copy operation
– DISMOUNT keyword in Mini-Merge: Automatically convert the specified number of Mini-Merge Write Bitmaps to Multi-Use (both Mini-Merge and Mini-Copy) Write Bitmaps when a shadowset member is $DISMOUNTed, so it can be re-added later with a Mini-Copy operation

Site IDs and Read Costs

- In multi-site clusters, you can set different Site IDs for each site if you wish
  - Volume Shadowing will automatically select default Read Costs
    - MSCP Served disks: difference of 499 in Read Cost
    - Fibre Channel disk at different site: difference of 40 in Read Cost
  - Advantage: Biases read operations toward using disk at local site rather than remote disk, or both disks
  - Disadvantage: If you get a burst of reads from one local site, you won't start sending any reads to the opposite site until the queue length gets very big (40, or 499)

Checking integrity of shadowset data

- $ ANALYZE /DISK /SHADOW compares contents of member disks, and reports any differences
- If differences are found, next question is: Which member has the most valid data?
  - Tip: You can vary the Read Cost values between the members to select first one and then (each of) the other(s) for reads, and identify which is the best copy of the data, then remove the other(s) and start a Full Copy to fix the bad member(s).

Monitoring Shadowset Membership

– You don’t want a shadowset member to drop out unnoticed

– Console Management systems can scan for dismount messages for shadowset members

– DCL procedure can be used to track shadowset membership and send alert on any changes; see example SHADOW$TRACK.COM from http://encompasserve.org/~parris/

Questions?

HPE
POINTNEXT

Thank you

keith.parris@hpe.com

Hewlett Packard
Enterprise