The ZIL,

The SLOG,

& Sync Writes

Balancing Performance & Protection

Asynchronous vs. Synchronous Writes

- Asynchronous Writes (or just "async writes"):
  - Application performing the write sends the data to the system and just waits for the data to be saved in system RAM before the application keeps going
  - System will eventually write that data to disks, but it can take a while (as we saw)
  - RAM is considered "**volatile**" storage: i.e, if power is lost, the data in RAM will be lost
  - HDDs and SSDs are considered "**stable**" or "**non-volatile**" storage-- i.e., if power is lost, the data is okay
  - Async writes can be lost if they haven't been written out to the HDD yet! Dangerous!

- Synchronous Writes (or just "sync writes"):
  - Application performing the write sends the data to the system and waits until the data is on stable storage (HDD or SSD) before it keeps going.
  - Since data has to hit the HDD, it can cause the application to sit and wait for a LONG time
  - Much safer than async writes because data will be protected even if there's a power outage

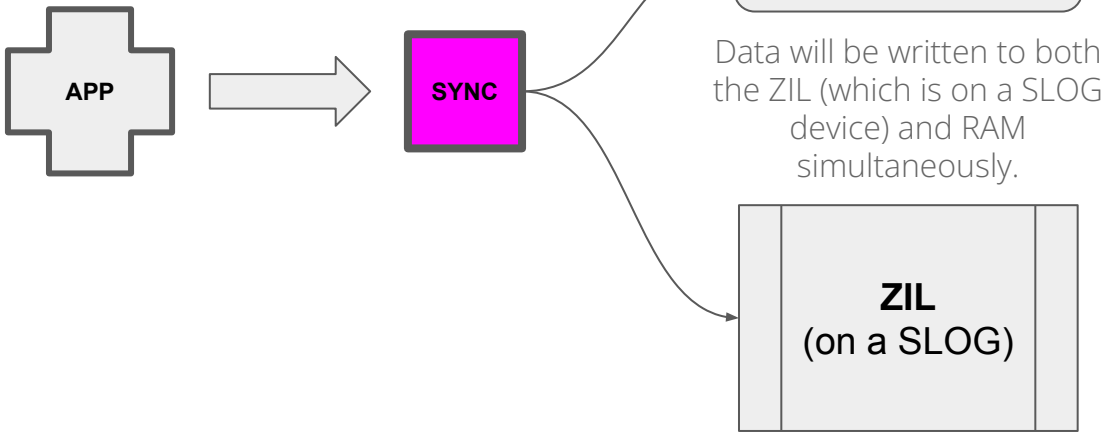
- Analogy: This is kind of like when you were a kid and your parents would drop you off at a friend's house and wait in the driveway for you to get safely inside before they drove away (that's a **sync write**). When your older brother dropped you off, he would just drive away as soon as you were out of the car (that's an **async write**).

Async & Sync Writes on ZFS

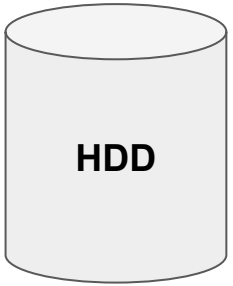
- **Async Writes** on ZFS get collected in RAM. ZFS collects ~5 seconds of data in RAM and then writes all the data to disk in one pass. Data in RAM but not yet on disk is called "**dirty data**", sometimes also referred to as "**in-flight**" data.
- **Sync Writes** on ZFS also get collected in RAM, but before the application sending the write data to ZFS can continue on its way, the data needs to be on stable storage! So in addition to sending data to RAM, ZFS writes the sync data to the **ZFS Intent Log (ZIL)**.
  - By default, the ZIL is kept on the disk pool. If the disks are HDDs, it's going to mean the application has to wait a LONG time before it can continue on its way.
  - Every ~5 seconds or so, the sync data is written from RAM to its permanent home on the pool
  - If system crashes or power is lost, we can recover the uncommitted sync writes from the ZIL!
  - We can put the ZFS Intent Log on a seperate device! Called a "**Separate Log Device**" or a "**SLOG device**" or just a "**SLOG**", this can MASSIVELY speed up sync writes to our pool!
  - We can also put **several SLOGs** in a system and "stripe" them together to handle larger workloads!
- The **SLOG** doesn't need to be big since it **only holds ~5s of writes**, but it MUST be power stable! Not all SSDs are power stable. Many consumer-grade SSDs will actually lose the last couple of seconds of writes if you cut power to them unexpectedly.

Sync Write Path in ZFS

An application sends some data to be written to ZFS & requests that it's written synchronously. Let's see what happens!

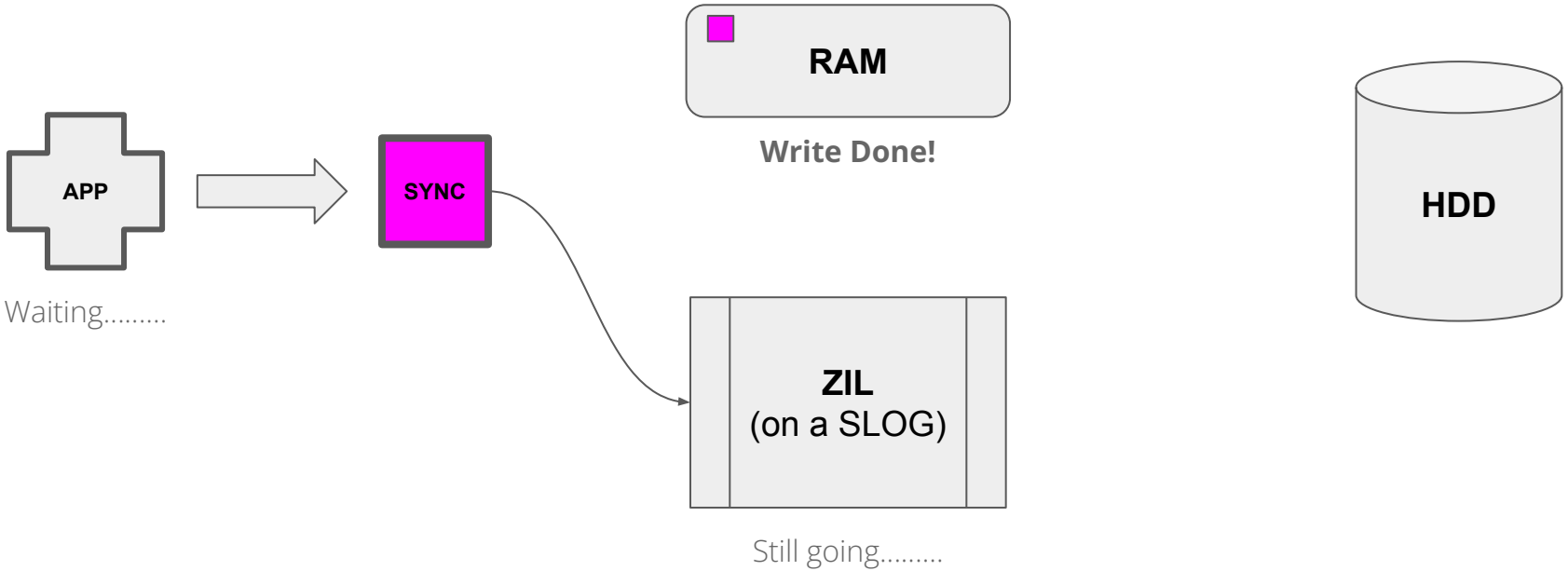


Data will be written to both the ZIL (which is on a SLOG device) and RAM simultaneously.



Write to RAM will always finish first, but application must wait for data to finish writing to ZIL before it continues

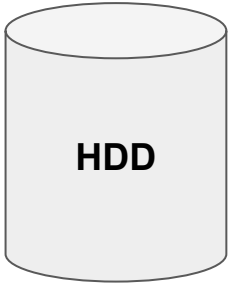
Sync Write Path in ZFS



Sync Write Path in ZFS

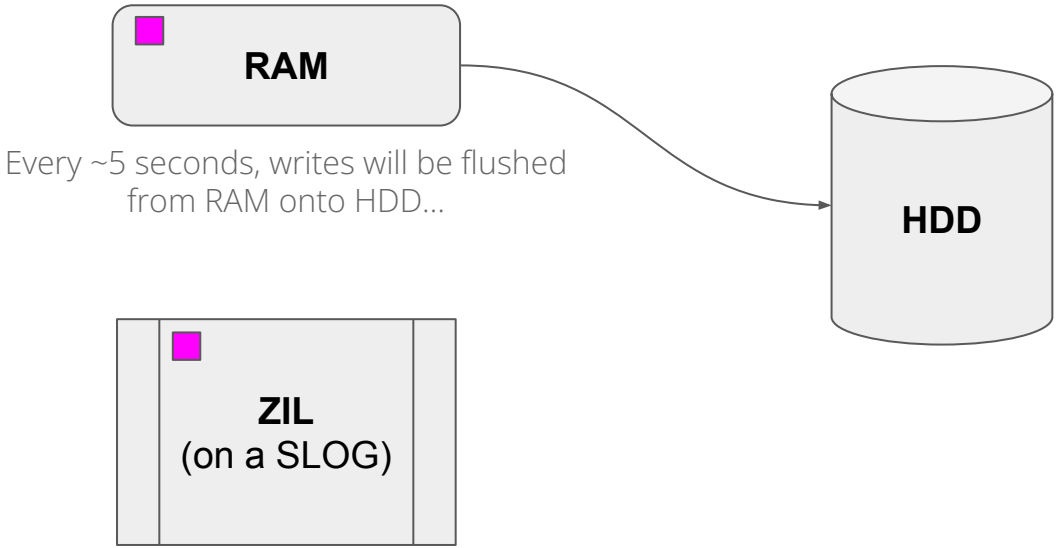
Finally Proceeding

Write Done!

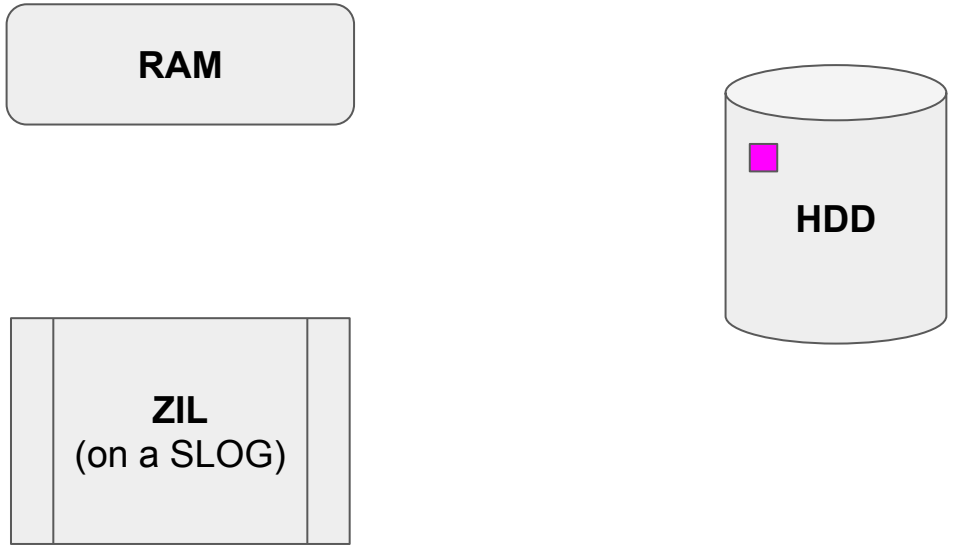


Write (Finally) Done!

Sync Write Path in ZFS



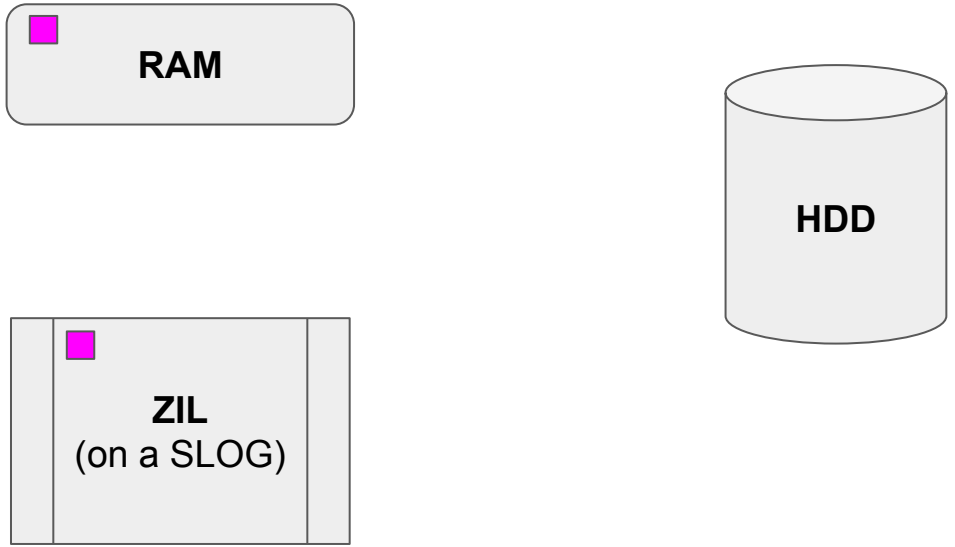
Sync Write Path in ZFS



After data is safely on HDD,
it can be deleted from ZIL

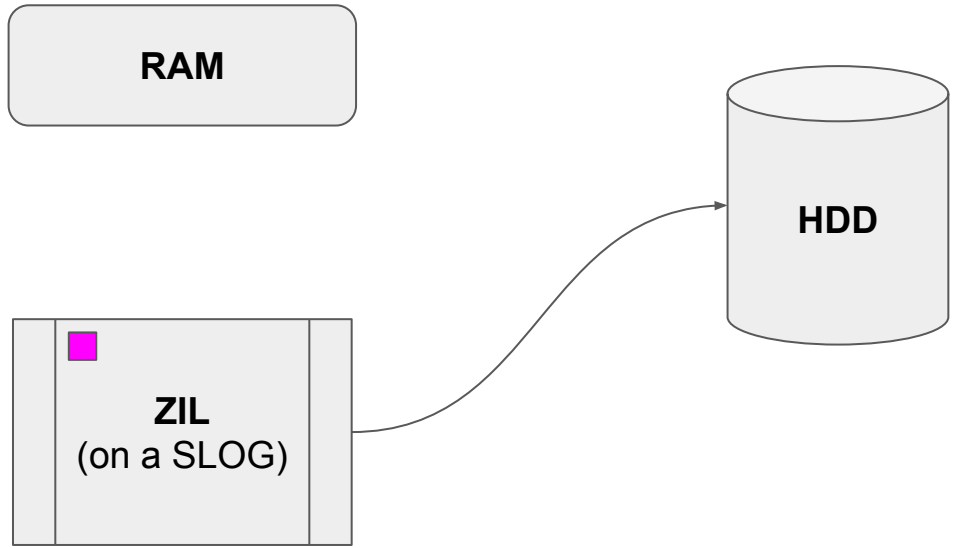
Sync Write Path in ZFS

But what if there is a power loss or system crash before data is moved onto HDD?



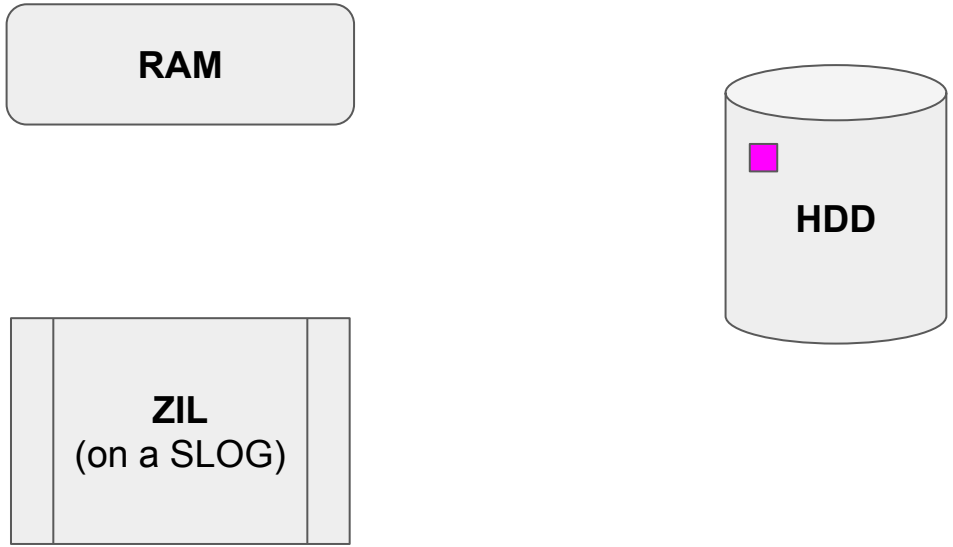
Sync Write Path in ZFS

On reboot, the write data will be gone from RAM, but it will safely be stored on ZIL. Can be recovered from ZIL and written out to HDD at that time.

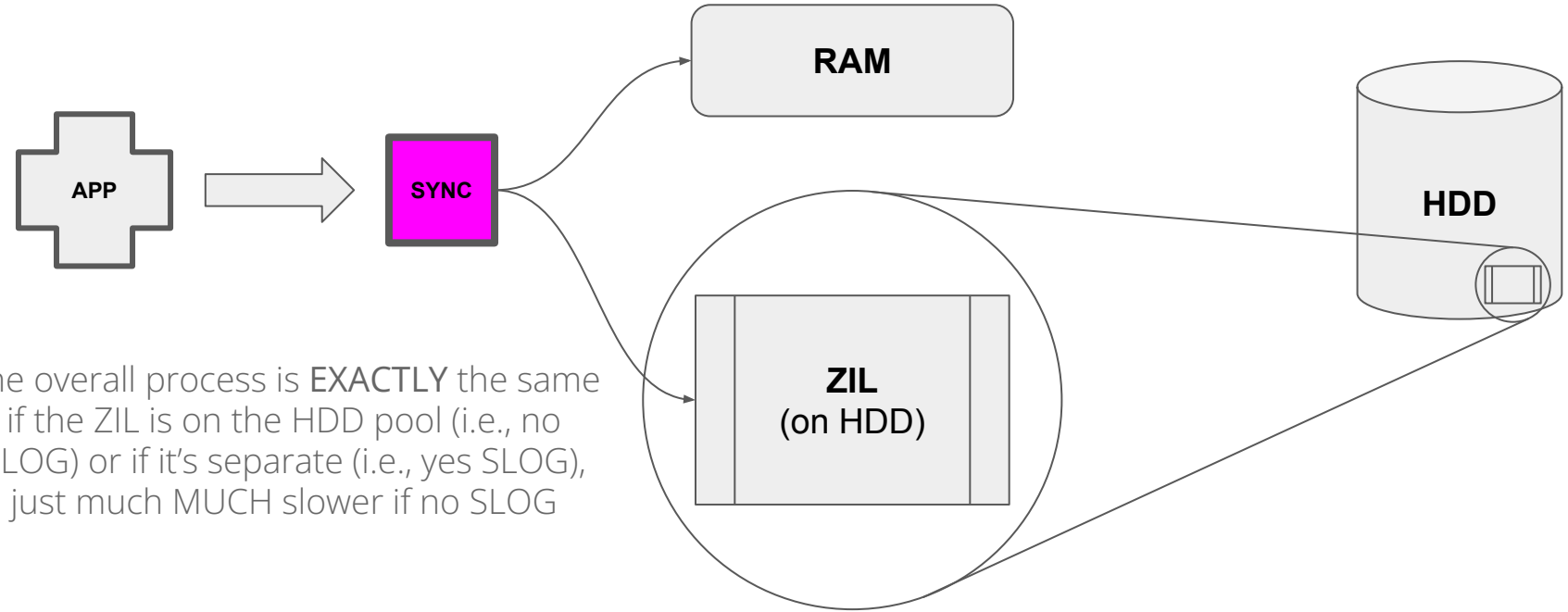


Sync Write Path in ZFS

Write data recovered!



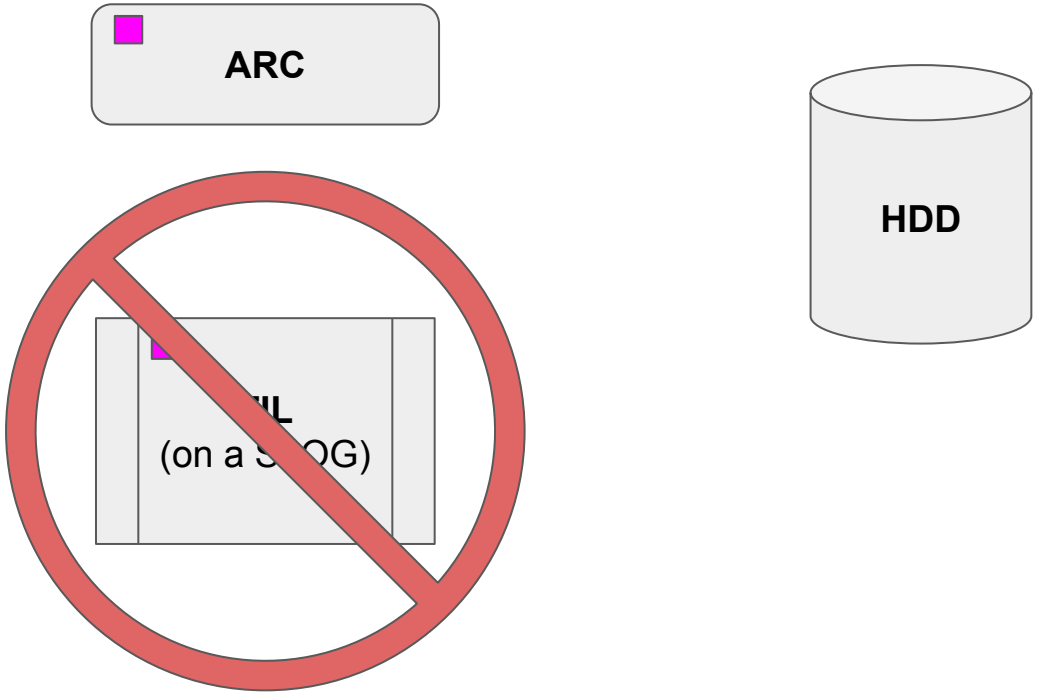
Sync Write Path in ZFS



The overall process is **EXACTLY** the same if the ZIL is on the HDD pool (i.e., no SLOG) or if it's separate (i.e., yes SLOG), just much MUCH slower if no SLOG

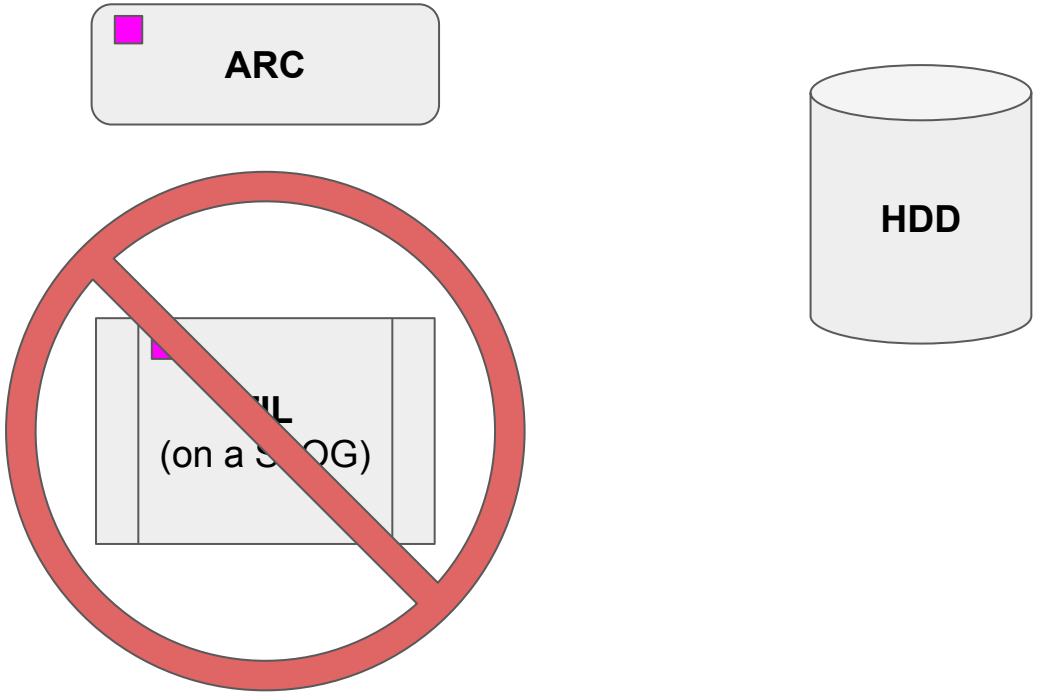
Sync Write Path in ZFS

What if the SLOG device dies?
Is the data lost?



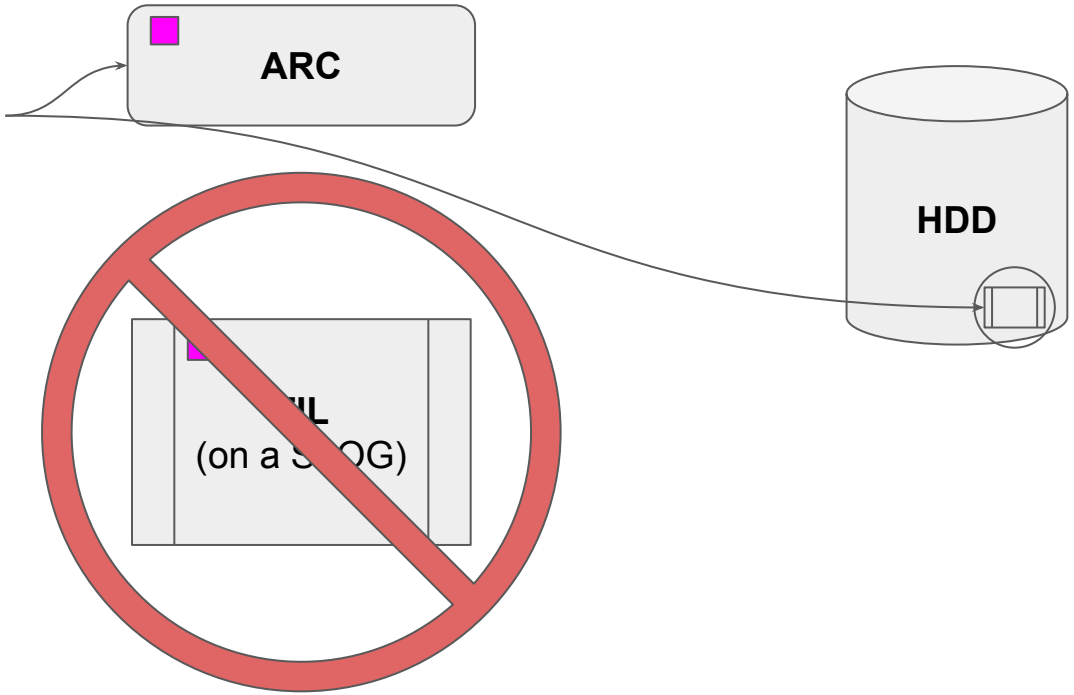
Sync Write Path in ZFS

What if the SLOG device dies?
Is the data lost?



Sync Write Path in ZFS

Data is still on the ARC, so it is not lost! System will continue to function as if there was no SLOG (ZIL will move to the pool).



What Applications Send Sync Writes?

- NFS does ALL of its writes as sync writes by default, making a fast SLOG essential!
- iSCSI and Fibre Channel will usually do some sync writes, but it depends on the client side. For example, the iSCSI client (called an "iSCSI initiator") on Windows sends all metadata as sync writes. A SLOG is typically recommended for iSCSI and FC unless workload is exceptionally light.
- Usually, SMB and AFP do not do any sync writes, however the SMB client on macOS DOES send sync writes!
- In ZFS, datasets and zvols have special properties that can be set to:
  - **`sync=standard`**: Respect the sync write settings of the application
  - **`sync=always`**: Ignore what the application says, treat ALL WRITES as if they were sync
  - **`sync=disabled`**: Ignore what the application says, treat ALL WRITES as if they were async
- In cases where data integrity is vital, users may set `sync=always` on their dataset. In that case, a SLOG will become essential.
- On an all-SSD pool, it usually does not make sense to use an SSD SLOG because it won't be any faster than the disks on the pool (i.e., moving the ZIL from one SSD to another doesn't help! Same thing for L2ARC.)