We have a flash device on each controller and a shared volume on the drive array. My assumption is that the flash device will provide a partition for the SVC VM that we can use as any Linux block device. For now, I’m assuming there will be enough space in the flash partition to hold any persistent data. If this proves to be incorrect as we determine the volume of data to be stored, we can adjust the solution. The shared drive volume will be accessible from either controller, presumably with the primary controller (as determined by the failover package) owning the volume. The primary controller is assumed to be hosting the active SVC VM also.

We want to have persistent data stored in both the local flash and on the shared volume. A mirrored pair of devices, one being the flash and one being the shared volume, would provide the needed redundancy. This will allow for recovery from the shared volume if the flash device on either or both controllers needs to be restored. In the event that the shared volume data is lost, the data could be recovered from the primary controller’s flash. Since the Linux LVM provides for mirroring, this would seem like the first place to look for a solution. The ability to determine the consistency of the mirrored devices would need more investigation than I have done to this point. During a failover sequence, the startup on the standby controller would require taking ownership of the shared volume and refreshing the flash with the contents of the shared volume (essentially resyncing the mirror).

With mirroring in place, one could argue that we have a solution. But I am concerned about the scenario where the IOVM or drive array interface on the primary controller fails. We end up with no method to log errors to the shared volume or store any other persistent data. If it can’t be written to the shared volume, it won’t be available to refresh the flash on the standby controller. Most system configurations that require HA storage rely on dual paths to the storage to protect against this type of failure.

Since the network connection between the controllers is the only alternate path we have, it seemed obvious that we should explore the options for using it as the secondary path. I had investigated DRBD on a previous project and thought that it would be a potential package to provide what we need. But as I investigated further, I realized that we don’t need a mirror of the shared volume, only a redundant path. I didn’t see any way to get DRBD to provide that. But there is another open source package, NBD (Network Block Device) that can provide a path to a remote volume over Ethernet. This would give us two paths to the shared volume from the primary controller, one through the IOVM, the other through NBD. Control of the path could be 1) a manual switch provided by the failover package, 2) the Linux MPP driver or 3) another multi-path driver.

I haven’t investigated all of the issues in depth, and I know some exist. My goal was to propose a solution that would require minimal coding of features that were already available and allow us to build on it for any additional capabilities.