Update on ZeroWatt

Kevan Rehm
Principal Engineer, Storage Software
Greg Banks
Principal Engineer, Storage Software

sgi

Overview

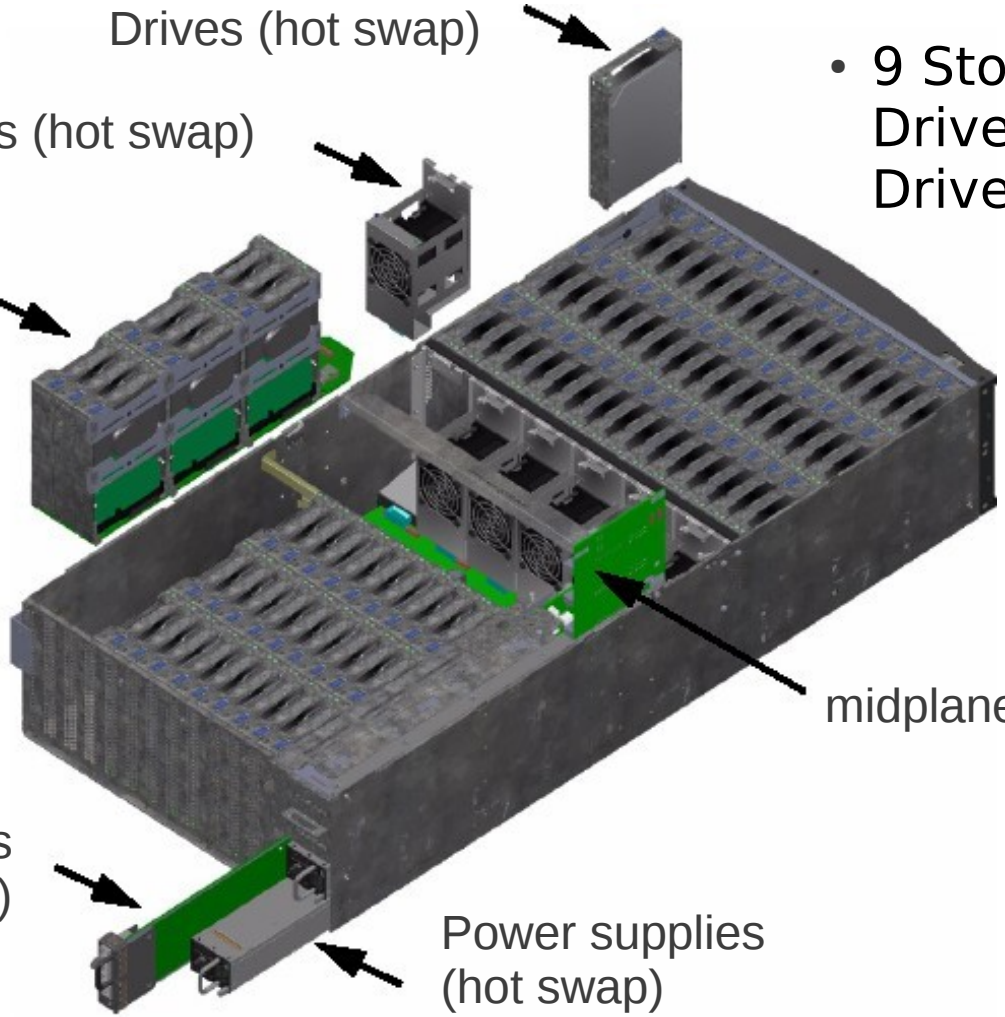
- About the MIS JBOD
- ZeroWatt™ on the MIS
- ZeroWatt with DMF/OV
- End Matter

About the MIS JBOD

High Storage Density

- MIS is a high density platform
- JBOD variant has no servers but even more disks
- 4U = 81 disks
- Up to 3.2 PB in a standard rack
  - raw capacity

Exploded View



- 9 StorBricks x 9 Drives = 81 3.5” Drives/chassis

Drives

- Minimum 3 drives per StorBrick
  - For airflow
  - 27 drives per chassis
- No *technical* limitations on adding drives
  - Any or all Storbricks
  - Could be different capacities
- Each drive has 2 SAS ports

Expanders

- A chassis contains either 1 or 2 JBODIO expanders
- Each JBODIO has 4 external SAS ports
- Each JBODIO connects through a midplane to all StorBricks
- Each JBODIO can see both drive SAS ports
  - Four paths to each drive
  - Redundancy in the face of failures of HBA port, cable, JBODIO, drive port

Cabling Configurations

- One config we've been using is a half-rack with
  - 1 MIS server
  - 4 MIS JBODs
  - one cable from each JBODIO going to the server (star config)
- It is also possible to daisy-chain JBOD chassis to a server
  - Slight performance loss

Weight Issues

- The weight of a populated chassis is 99kg
- The sliders are made so that you can only pull the chassis out halfway to prevent the rack from tipping over.
- It pulls out halfway both front and back, so you work on one half of the unit at a time.
- Cable arm on the back that contains power & SAS cabling, no strain when you pull the chassis out.
- You have to install the chassis in the rack first, then put the drives in after that.

Field Replaceable Units

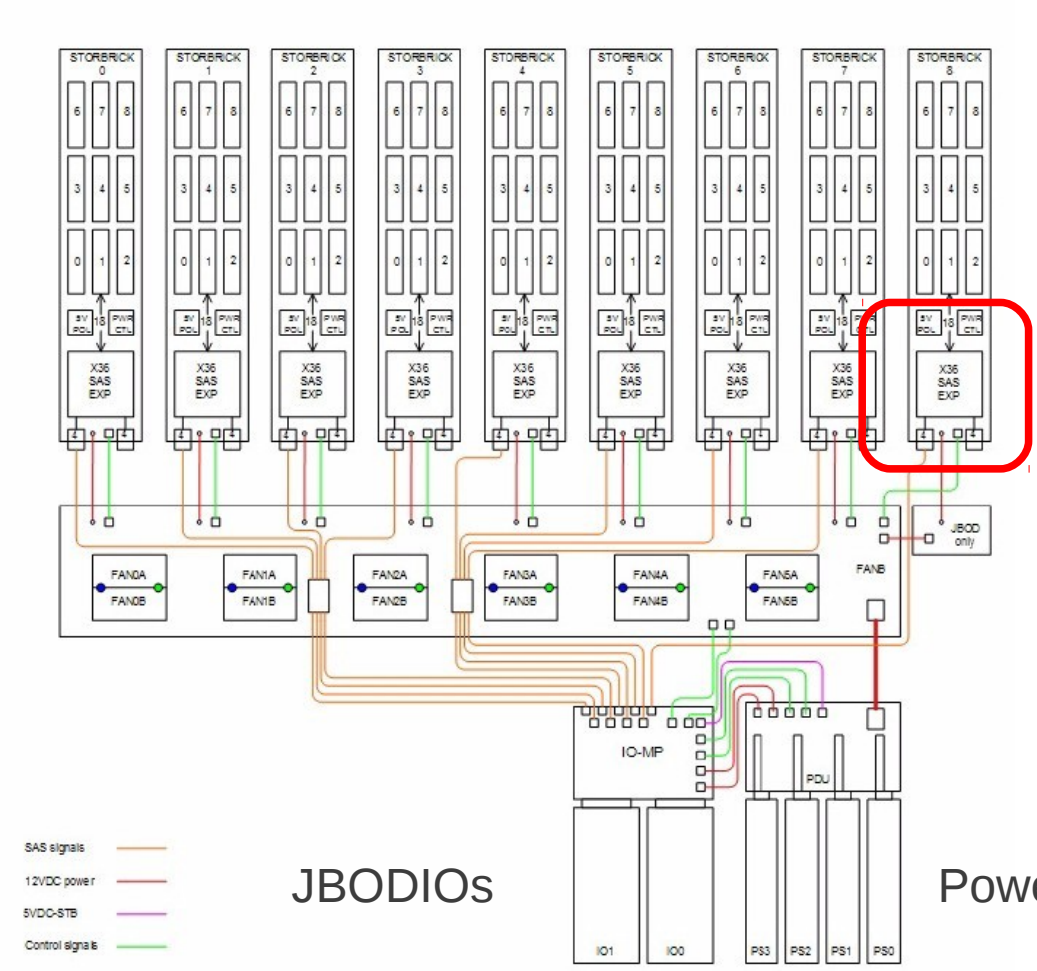
- The front half contains 5 StorBricks.
- The back half contains 4 StorBricks plus one or two JBODIOs.
- Drives are hot-swappable.
- JBODIO modules, fans, and power supplies are all modular hot-swappable.

ZeroWatt on the MIS

Hardware

StorBricks containing drives

midplane



Hardware here enables powering up or down individual disks

JBODIOs

Power supplies

Firmware & Software

- To power up or power down a drive, you have to issue a SES command to the StorBrick that contains the drive.
- No Always-On-Region (AOR) like the Copan had, drives do not spin up if accessed.
- Actually, when drives are off there are no disk paths present in /dev/sdXXX
- No ZeroWatt on the MIS Server
  - Incompatible with the internal MegaRAID HBA.

Timing

- Power down: the udev paths disappear immediately.
- Power up: up to 16 seconds before all the drive paths re-appear.
- Power draw limits require staggering power-up of multiple drives.
- An entire chassis power-up of 81 drives can take close to a minute.

ZeroWatt with DMF/OV

Layers

- DMF
- OpenVault
- JBFS
- Data path: XVM
- Data path: SD block device
- Control path: Python shim
- Control path: sg3_utils
- Control path: SG block device

Volume Management

- Using XVM to manage the drive paths.
- Each JBOD drive be a separate XVM volume
- We use XVM because it is able to
  - Automatically switch to other drive paths should one path fail.
  - Uses preferred paths (in daisy-chain configs).
  - In H/A configs prevents the backup H/A machine from touching the disk while the primary server is running.
  - Create larger block devices using stripes / concats
  - (should that provide a benefit in some particular config)
  - Provides a naming conventions for volumes.

JBFS

- JBFS = The JBOD File System
- A replacement for XFS for use in the context of the COPANSYS and ISG.
- JBFS sits on top of a block device
  - Could be disks, RAID LUNs, whatever
- JBFS divides a block device into one or more separate virtual volumes which are independent of each other.

NoXFS

- By comparison, the Copan used an XFS filesystem on the block device
- With JBFS we are not constantly mounting and unmounting XFS filesystems.
- XFS is very complex and powerful
- Virtual Tape requirements are limited: 3 files, 2 only used sequentially

JBFS - Virtual Volumes

- DMF MAID support currently maintains “tape data” on a disk system through the use of three files
  - The index file
  - The metadata file
  - The data file
- Together, these allow the disk to simulate a tape drive with very fast mount times.

JBFS - Blocks

- JBFS divides the block device into
  - Some metadata
  - Central data region
  - A redundant metadata copy
- Data region is divided into a relatively small number of large equal-sized “blocks”
- Number of blocks (maxblks) chosen at format time.
  - Default is 3600
  - Minimum is 1296

JBFS – Blocks & Files

- Blocks are initially a freelist.
- Blocks can be removed from the freelist and threaded together to create “files”
- A file is an arbitrary number of blocks
- File sizes are multiples of the total disk size divided by maxblks

OpenVault

- JBFS (like Copan) presents a "virtual tape" interface to DMF.
- JBFS is not visible to Linux as a file system (no mount/umount)
- A block device is treated as a "cartridge" in OV-speak.
- A cartridge can contain one or more "volumes" (VSNs).

Multiple Volumes/Cartridge

- OpenVault server doesn't yet support multiple volumes per cartridge.
- Since the LUNs are only 4 TB max in JBODs, we will only support a single volume per cartridge in the first release
- Will relax that restriction later for when we support much larger cartridges.

Powering Down

- The disk is left in a locked state when powered down, with a short expiration timeout.
- OpenVault can power up and use the disk immediately afterward for a new mount, since it already owns the lock.
- However, a subsequent **ov_jbfs** operation will have to wait for the lock to expire.

LCPs and DCPs

- In DMF, an MIS JBOD is managed using a Library Server.
  - Each JBOD disk is a volume record in the VOL database.
- In OpenVault, there will be a LCP for a JBOD chassis.
- And *some number* of DCPs.
- One CARTRIDGE and one VOLUME record for each disk.

How Many DCPs?

- Each disk could be simultaneously powered
- So we could have one DCP per disk in the chassis (81)
- But one DCP per disk in unmanageable in OV
- So we choose to configure a lesser number of DCPs, enough to support a reasonable number of simultaneously spinning disks

How Many LCPs?

- It's hard to define what to use for the boundaries of an LCP.
- If you have two separate RAID units, each with a bunch of LUNs, do you put them together into one "library" or two?
- H/A issue
  - If one of those RAID units will fail over to another host but the other one won't, then you don't want them to be in the same LCP library.
- Redundancy issue: you don't want to create two LCPs to manage disks in the same RAID controller
  - That will look to DMF like two independent pieces of hardware.
  - Should the RAID controller fail, then both LCPs will go offline.
  - We don't want an admin to configure his two copies of each file and have them both end up on the same piece of hardware with a single point of failure.

When To Power Off/On Drives

- The LCP will be the gatekeeper in terms of deciding when to power off drives
- If a "cartridge" hasn't been mounted in some period of time, the LCP will call out to some Python code to power off the drive
- **ov_jbfs** is capable of spinning up drives
- Sets a hint on the disk as to how long the drive should stay spinning
- After that time the LCP will spin the disk down again
- Analogy for this is house lights
  - Kids can go from room to room turning on the lights and leaving the room.
  - At some point Mom (LCP) will go around, see there is noone in the room, and turn the lights off again.

Generic Code

- 99% of the LCP/DCP/DMF code for JBFS is generic, it will work with any block device
- It's essentially a distributed disk MSP
  - Except that the media format is virtual tape instead of separate disk files.
- In the case of JBOD there is a thin shim layer of functions that JBFS calls that interfaces with Python disk power code.

Python Disk Power API

- There are Python functions for
  - Turning on the disk write cache
  - Powering up a drive
  - Powering down a drive
  - Seeing if a drive is capable of power-down
  - Query to retrieve the list of LUNs that make up the "library"

LUN Naming Convention

- The LUNs that the LCP is responsible for managing.
- We use an XVM naming convention for this.
- Any LUN named **JBFS_**$*libname_$vsn* is deemed to be owned by the LCP which controls the OV library $*libname*.

New OpenVault commands

- **ov_jbfs** is a low-level command for formatting a block device to act as one or more virtual tapes.
- Its Copan equivalent is **ov_copan**.
- For Copan there is also a higher-level command called **ov_shelf** which understands OV configuration and is able to auto-configure a Copan shelf formatted with **ov_copan** for use with OV.
- I will need provide a similar command for JBFS block devices.

End Matter

Supported Configurations

- Not sure what sort of unbundled configs we will allow yet
- For ISG we were going to support a max of 4 JBOD chassis per MIS server
  - 1.2 PB raw in a half-rack.

Availablity

- ISSP 3.2
  - Possibly patched back to ISSP 3.1
- For unbundled DMF
  - Not for ISG
  - No Big Easy Button

Future Directions

- In the next release, we will add some software that will monitor the JBOD disks and report things like
  - Recovered/unrecovered errors
  - Fan temperature swings
  - Power supply hiccups

Future Directions: Drive Failures

- In some future release, hope to have DMF code which recognizes when a drive has failed, and automatically schedules dmmoves of other file copies to replace the lost copies
- For now we will need to do this manually should a LUN fail.

Possible Future Directions?

- Big Easy Button configuration of MIS JBODs
  - The look and feel would be similar to how Copan is done
  - The admin would be shown the list of available unconfigured JBOD chassis
  - He would be able to select one, give it a name (e.g. J00), then click on "configure it" and the entire chassis would get configured as a single library with 81 cartridges, both in OV and DMF
- GUI button which turns on the blue ident light on the front of the chassis.

More Future Directions??

- A way for the admin to assign a human name to a JBOD chassis
  - "second JBOD from the bottom in rack 37"
  - Not "chassis 0x49a8b8x" or whatever
  - Stored in the chassis NVRAM
  - Python functions to retrieve the name
  - So that error messages could contain meaningful names
  - Room for about 32 bytes
-

sgi