Labs storage

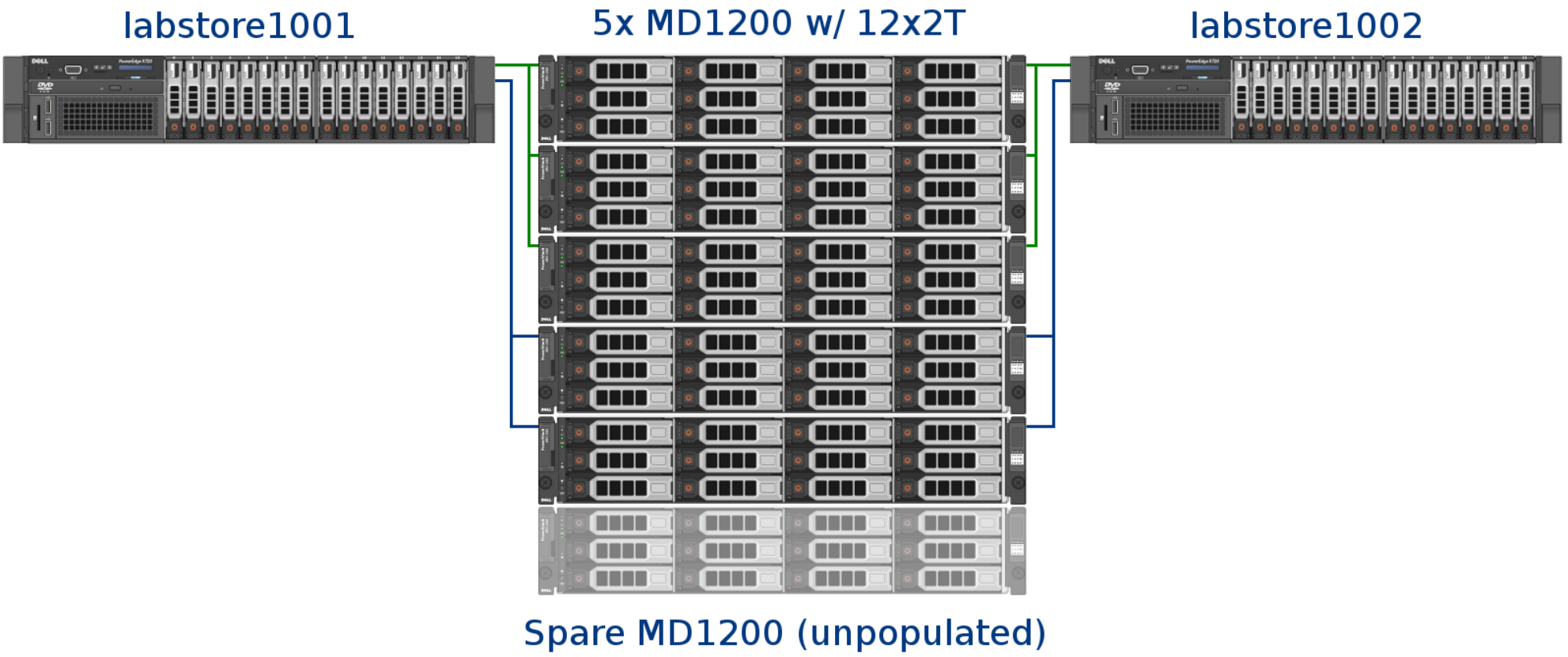
“Better than Gluster™”

Labs storage

- Serves filesystems to labs
  - /home
  - /data/project
  - /data/scratch
  - /public/keys (obsolescent)
  - /public/dumps
- Lots of interacting layers, but none of them complicated in isolation.

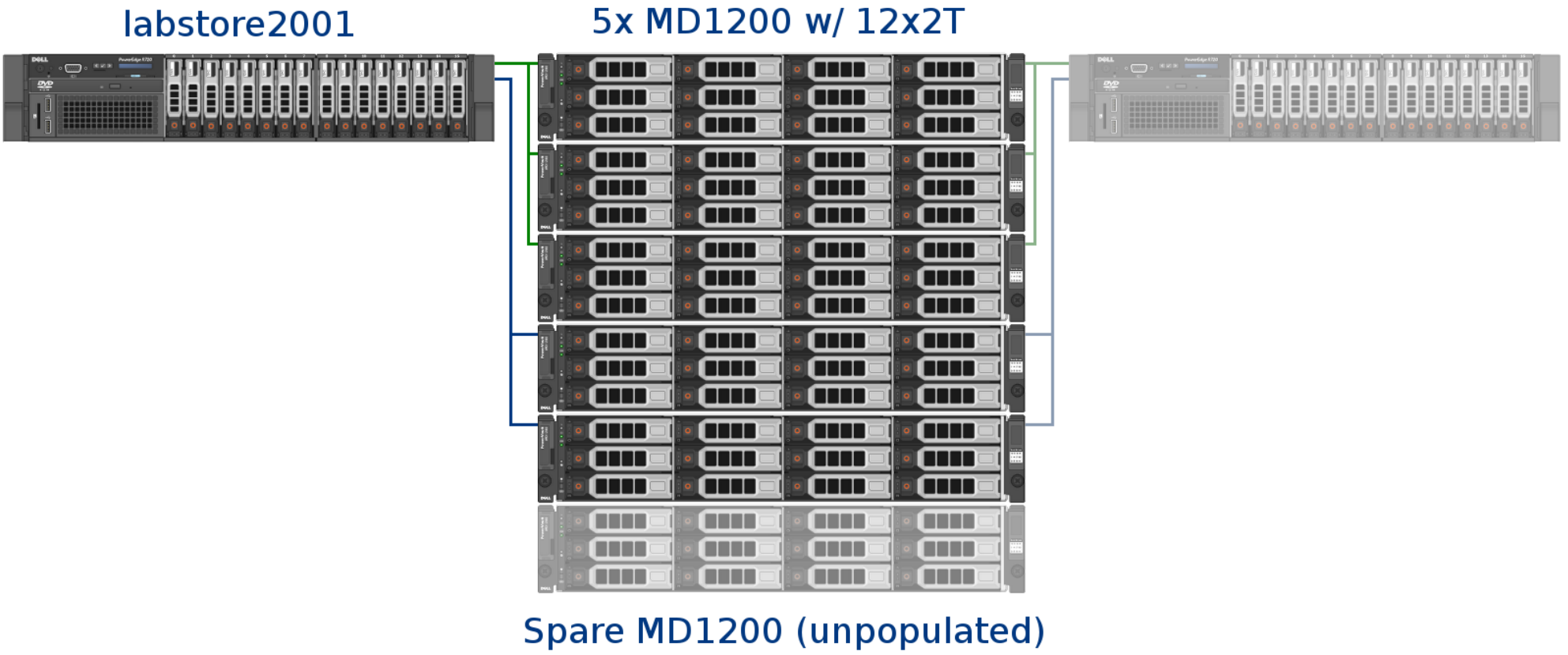
1. Hardware

eqiad



1. Hardware

codfw



2. RAID

- First two internal drives as RAID1 for OS
- Six internal drives as RAID5 (scratch)
- Four internal drives no longer used (was dumps)
- Each external shelf (12 disks) configured as a RAID6 array

2. RAID

- First two internal drives as RAID1 for OS
  - Boring mirror
  - Managed with LVM

2. RAID

- Six internal drives as RAID5 (scratch)
  - Explicitly publicised as ephemeral (/tmp-ish), good for caches, work directories, etc
  - Shared between projects
  - Different controller, so [less] contention with /data/project
  - Lost in case of switchover unless we take the time to make a copy (unlikely)

2. RAID

- Four internal drives no longer used (was dumps)
  - Dumps now live on labstore1003 (were growing too fast)
  - Was just striped, since the data can just be copied again.

2. RAID

- Each external shelf (12 disks) configured as a RAID6 array
  - /dev/md/slice1 to /dev/md/slice5
  - Used as physical volumes for LVM

3. LVM

- Organized in two volume groups
  - VG “os”
  - VG “store”

```
# vgs
  VG    #PV #LV #SN Attr   VSize  VFree
  os      1   2   0 wz--n-  1.82t  1.71t
  store   5   5   0 wz--n- 90.94t 18.15t
```

- “os” is just boring

3. LVM

- VG store is the interesting one

3. LVM

```
# lvs store
  LV      VG    Attr     LSize  Pool  Origin Data%  Move Log Copy%

  journal store -wi-ao-- 40.00g
  keys    store -wi-ao--  1.00g
  now     store Vwi-aotz 40.00t space         34.27
  project store -wi-a--- 30.00t
  space   store twi-a-tz 42.75t               32.10
```

- store-project is the previous /data/project – it's only kept as a paranoid backup until we need the space

3. LVM

```
# lvs store
  LV      VG    Attr     LSize  Pool  Origin Data%  Move Log Copy%

  journal store -wi-ao-- 40.00g
  keys    store -wi-ao--  1.00g
  now     store Vwi-aotz 40.00t space         34.27
  project store -wi-a--- 30.00t
  space   store twi-a-tz 42.75t               32.10
```

- store-keys is a normal flat volume for /public/keys

3. LVM

```
# lvs store
  LV      VG    Attr     LSize  Pool  Origin Data%  Move Log Copy%

  journal store -wi-ao-- 40.00g
  keys    store -wi-ao--  1.00g
  now     store Vwi-aotz 40.00t space         34.27
  project store -wi-a--- 30.00t
  space   store twi-a-tz 42.75t               32.10
```

- store-space is the pool for thin volumes

3. LVM

```
# lvs store
  LV      VG    Attr     LSize  Pool  Origin Data%  Move Log Copy%

  journal store -wi-ao-- 40.00g
  keys    store -wi-ao--  1.00g
  now     store Vwi-aotz 40.00t space         34.27
  project store -wi-a--- 30.00t
  space   store twi-a-tz 42.75t               32.10
```

- store-now is the current, live /data/project – it's a thin volume in the *space* pool

3. LVM

```
# lvs store
  LV      VG    Attr     LSize  Pool  Origin Data%  Move Log Copy%

  journal store -wi-ao-- 40.00g
  keys    store -wi-ao--  1.00g
  now     store Vwi-aotz 40.00t space         34.27
  project store -wi-a--- 30.00t
  space   store twi-a-tz 42.75t               32.10
```

- store-journal is the journal device for *store-now*

4. Thin volumes

- Thin volumes work by an added level of indirection, the extent table
  - Extents are a multiple of the underlying device stripe size (~4M)
  - Allocated on demand, obeys discards from the filesystem (so extents get released)
  - Ext4 sees the extents as stripes, so allocates intelligently along boundaries to avoid write amplification

4. Thin volumes

- Thin snapshots are just playing with extent tables:
  - Creating a snapshot just makes a copy of the extent table (~1-2s) and mark the extents on the original COW*
  - Small (~5-10s) extra cost because the snapshot is forced consistent by flushing journal to next barrier
  - Extents are reference-counted, so discarding a snapshot just discards extents that have since been changed
  - Because of the indirection, further snapshots have no duplicated extents

4. Thin volumes

- Thin snapshots are just playing with extent tables:
  - Creating a snapshot just makes a copy of the extent table (~1-2s) and mark the extents on the original COW*

    * not exactly true – there isn't an explicit COW flag, just anything with refcount over 1 is implicitly COW. This means that if the refcount goes back to 1 without a write then there will never be a copy.

5. Filesystems

- All filesystems ext4 with flex_bg, sparse_super
- / and /var come from /dev/mapper/os
- /srv has filesystems intended for export:
  - *store-now* on /srv/project
  - *store-keys* on /srv/keys
  - /dev/md/scratch on /srv/scratch

5. Filesystems

- There *is* one bit of trickery: *store-now* has an external journal device, *store-journal*
  - Not a thin volume to avoid write amplification (writes in units of RAID stripes, not extents)
  - Not part of snapshot since a journal is meaningless on a readonly time-consistent snapshot. Copying the extents it occupies would be pointless and guaranteed to happen every time.

6. Exports

- NFS4 has a specific requirement: all exported filesystems must be from a single tree, which must have FSID 0
- We follow tradition and use /exp for this:

```
/exp *(ro,fsid=0,sec=sys,sync,subtree_check,root_squash,nocrossmnt)
```

6. Exports

- We populate /exp with bind mounts back into /srv
  - /exp/keys -> /srv/keys
  - /exp/scratch -> /srv/scratch
  - /exp/project -> *not* /srv/project
- /exp/project is a directory populated per-project
  - /exp/project/$project -> /srv/project/$project
- Why? Because you can't .. past a mountpoint so project exports are contained!

6. Exports

- Exports are all in /etc/exports.d
- Fixed exports have uppercased filenames
  - ROOT.exports
  - PUBLIC.exports

```
/exp/keys *(ro,sec=sys,sync,no_subtree_check,no_root_squash)
/exp/scratch *(rw,sec=sys,sync,no_subtree_check,root_squash)
```

6. Exports

- Per-project exports are named $project-home.export
- Predictable content:

```
/exp/project/tools  -rw,nohide,fsid=0000000000000000-50380-
0000000000,subtree_check,async,no_root_squash 10.68.16.10...
```

- fsid is derived from project group id
- All of this managed by manage-nfs-volumes

7. Software

- Not a lot running on labstore*
- Two daemons:
  - manage-nfs-volumes-daemon
  - replica-addusers.pl
- Plus a few utilities

7. Software

- manage-nfs-volumes-daemon
  - Polls LDAP for new projects and new instances
  - Creates their directory in /srv/project when needed
  - Maintains the exports to exports.d, so that new instances get access

7. Software

- replica-addusers.pl
  - Polls LDAP for new users
  - Creates database credentials for new users and add the grants to the replica databases and the user database
  - Stores those credentials in the users' homes in /srv/project
  - Why on labstore? Because this is the only server which has write access to *all* the project homes and access to the labs-support network to connect to the databases.

7. Software

- storage-snapshot
  - Creates hourly snapshots of the project filesystem and maintains a defined set for timetravel
  - Intended to be run by cron, but we're waiting to move to Jessie because kernel

7. Software

- storage-replicate
  - Takes the latest hourly snapshot available, mounts it, and rsyncs it to a destination of choice (codfw labstore2001)
  - Intended to be run daily by cron, but we're waiting to move to Jessie because kernel

7. Software

- sync-exports
  - Populates the /exp tree from the /srv tree
  - Automatically invoked by manage-nfs-volumes and start-nfs, but may be useful in case of trouble

7. Software

- start-nfs
  - This is what you run to start NFS service on a labstore.
  - *Not* invoked at boot – requires a human. Having two active servers would be a disaster
  - Mounts filesystems, syncs the exports, add the floating IP, and starts daemons.

8. Redundancy and DR

- Two servers attached to the shelves, either can be made the active server in case of server or controller failure
- Two controllers on each shelves, so if one fail we can switch servers or swap wiring
- A spare shelf in both DC so if the enclosure itself fails we can move the disks and recover

8. Redundancy and DR

- Two servers with kernels >= 3.9 (tested) can be on and running while attached to the same shelves provided the filesystems are mounted rw on only one of them.
- This is why the shared filesystems are noauto and only mounted by *start-nfs*

8. Redundancy and DR

- start-nfs can be invoked on the inactive server immediately upon shutdown of the active server (or as soon as it is known that it is dead and can no longer write to the drives
- Because the fsid are consistent, and the IP used by clients is floating, they won't even notice the switch...
- Except for /data/scratch which is not shared.

9. Notes

- Extra useful monitoring:

http://grafana.wikimedia.org/#/dashboard/db/labs-monitoring

- Known issue atm: CPU bottleneck on RAID6 parity calculation (Precise kernel binds checksumming to single core)
- Single client can consume enough IO bandwidth to impact all clients
- Won't scale forever. :-(

9. Notes

- labstore1003 serves dumps – it runs nothing else and only has the single, readonly, export to labs and readwrite to the dump servers.
- It has no attached shelves, just the internal bays.