Topics

- Building
- Configuring
Topics

> Building
> Configuring
Pre-built rpms

• We provide pre-built rpms
  > For 1.6, RHEL4/5 & SLES9/10
  > For 1.8, RHEL5 & SLES10/11
  > For 2.0 (alpha version) RHEL5, SLES10, SLES11 (soon) & RHEL6 (when available)
    – subject to change

• Include OFED & TCP support

• Rebuilding rpms is needed if:
  > Need support for another interconnect (Myrinet, ...)
  > Need to apply kernel or lustre patches
Building lustre (server side)

• Kernel patches needed
  > Re-add journal callback support in jbd
  > Jbd fixes & statistics
  > scsi disk statistics
    – could be removed if blktrace enabled
  > Export some symbols used by lustre
  > API for setting block device read-only
  > ...

• First step is to apply those patches & build the patched kernel
  > Use quilt to manage patches
  > Patch series available in lustre/kernel_patches/series
  > Quilt setup /path/to/series, quilt push -a
  > kernel config files in lustre/kernel_patches/kernel_configs
Building lustre (server side)

- Once the kernel is built, we are ready to build the lustre rpms:
  - Get the lustre source
  - `./configure --with-linux=/path/to/kernel ..`
  - `make rpms`

- This produces several rpms:
  - lustre-modules: the lustre kernel module
  - lustre-ldiskfs: ext3+patches
  - lustre-$version: utils (mkfs.lustre, mount.lustre, ...)

- Install the patched kernel + lustre/ldiskfs rpms on the servers (OSSs/MDSs)
Building lustre (client side)

- No kernel patches needed
  - except for RHEL4/SLES9
  - You can run the patched kernel on the clients if you wish
- Get the lustre source
  - make
    - configure --with-linux=/path/to/kernel --disable-server ..
    - make rpms
- Build the lustre rpms as previously:
  - configure --with-linux=/path/to/kernel
    --disable-server ..
  - Generate rpms with client only support
- Install the lustre rpms on the client nodes
Building lustre with DMU support

• No change
• ldiskfs rpm replaces by kDMU rpm
• kDMU integrated in lustre source
  > built as part of lustre, like ldiskfs today
  > only needed on OSS/MDS (again as ldiskfs)
Topics

> Building
> Configuring
Supported Networks

- Any network running TCP
- Quadrics
  > Qsnet I & II
- Myrinet
  > gm & mx
- Infiniband
  > Old stacks: Topspin, ininiserv, OpenIB gen 1, Voltaire
  > OFED
- Seastar network
- Routing is supported via Inet gateway
- Multiple NIC support
Management node (MGS)

- By default just uses the MDT
  - Can be a different node
  - Can be failover
- Functionality
  - Supply configuration information to clients
  - Receive new nodes into the cluster
  - Notify other nodes that some configuration has changed
    - E.g. Dynamic addition of servers
Setting up a lustre filesystem

- mkfs.lustre & mount
- MDS – on mds1 node format
  
  \texttt{mkfs.lustre --mdt --mgs --fsname=scratch } \$dev
  \texttt{mount -t lustre } \$dev \ /mnt/mdt

- OSS
  
  \texttt{mkfs.lustre --ost --fsname=scratch --mgsnid=mds1@vib } \$dev
  \texttt{mount -t lustre } \$dev \ /mnt/ost1

- Clients
  
  \texttt{mount -t lustre mds1@vib:/scratch } \ /scratch
Example: failover

- **MDS** – on mds1 node format:
  
  ```
  mkfs.lustre --mdt --mgs --fsname=swgfs --param
  lov.stripecount=4 \ 
  --param lov.stripesize=4194304 --failnode=mds2@vib $dev
  ```

  The failover software will execute a mount command like:
  ```
  mount -t lustre $dev /mnt/mdt
  ```

- **OSS**
  
  ```
  mkfs.lustre --ost --fsname=swgfs --failnode=ossY@vib \ 
  --mgsnid=mds[1-2]@vib $dev
  ```

  mount -t lustre $dev /mnt/ost1

- **Clients**
  
  ```
  mount -t lustre mds[1-2]@vib:/swgfs /mnt/lustre
  ```
**Lfs stripe commands**

- Default striping policy set with `mkfs.lustre`
- Files are striped at *creation time*
  > Until we have a data migrator, you can’t change it post-creation
- `lfs setstripe` can adjust it on a per-directory basis
  > That becomes the default for new files in that directory
  > Some admins create `/lustre/parallel` for massive single-file I/O
    - With a stripe count of -1(all OSTs)
- `lfs setstripe` can also create individual files
- Can specify a specific OST pool
- Query with `lfs getstripe`
Adding an OST

• Format & mount
• QoS introduced in 1.6
  > Qos will fill empty OST's first
  > We will stripe cleverly, avoiding multiple stripes on one OSS etc.
• No automatic rebalancing yet
Dealing with a failed OST

```bash
mount -v -o exclude=fsname-OST000N -t lustre mgs:/fsname /mnt/lustre
```

- This uses the server partition label
- The client will immediately return errors when trying to contact this OST
- This has the same effect as issuing `lctl --device N deactivate`

- Here N is the OSC device on the client find it with `lctl dl`
Stopping servers

- **Umount**
  - Mount, keep failover state
  - Client can reconnect and continue after a remount

- **Umount -f**
  - Unmount, disconnect clients, cleanup failover state
  - Reconnecting clients are evicted
  - This is like NFS server reboots
  - Can sometimes give application errors
I/O Performance Pipeline

• Keep the entire pipeline balanced:

• Network BW should be equal to disk BW
• We reach benchmark rates of 85-90% of raw I/O
• Bus transfers – don't forget about them!
  > The client has one – memory to network
  > The OSS has two – network to memory, memory to SAN/controller