***Overview*:** This simple setup guide contains steps to build a custom XFS kernel module which is able to demonstrate layout control modification on pNFS under the given environment setup.

**Environment Setup:**

* pNFS: 1 MDS + some data servers + some clients
* OS: Fedora 23
* Kernel: (please use the latest kernel)
* Arch: x86\_64
* boot partition on MDS is not XFS (it can be other filesystems)

**Assumptions:**

* 2 disks from data servers form a logical volume on MDS by LVM:
  + HDD: 2 TB (1863 GiB)
  + SSD: 100 GB (93.16 GiB)
* The logical volume is at /dev/vg1/lvol0
* Check the physical extents and logical extents mapping via *lvdisplay -m*
  + In our testing environment, HDD is always in front of SDD.
* The virtual volume uses XFS with 4 KiB block size and 16 GiB AG size:
  + setup on MDS:   
    *mkfs.xfs -f /dev/vg1/lvol0 -b size=4k -d agsize=16g*
  + HDD: 117 AGs (0-116)
  + HDD & SSD mix: 1AG (117)
  + SSD: 5 AGs (118-122)

**Step 1: Install required software packages & the latest kernel**

Run the following commands to install required dependencies & the latest kernel:

*dnf install fedpkg fedora-packager rpmdevtools ncurses-devel numactl-devel pesign -y  
dnf groupinstall 'Development Tools' -y  
dnf update kernel -y  
reboot* # if kernel is updated

Also, make sure kernel on MDS is also updated to the latest one.

**Step 2: Install kernel source RPM & setup build environment**

**Note: Building packages as root is dangerous. For most of the following kernel building steps, we do not need to run as root!**

*rpmdev-setuptree  
dnf download --source kernel*

Assumes **kernel-4.2.5-300.fc23.src.rpm** is downloaded at home directory.  
(The version number is just an example)

*su -c 'dnf builddep kernel-4.2.5-300.fc23.src.rpm' # require root password  
rpm -Uvh kernel-4.2.5-300.fc23.src.rpm*

We can safely ignore all these warnings: *group \*\*\* does not exist - using root*

*cd ~/rpmbuild/SPECS  
rpmbuild -bp --target=$(uname -m) kernel.spec*

**Step 3: Configure kernel source building options**

*cd ~/rpmbuild/BUILD/kernel-4.2.fc23/linux-4.2.5-300.fc23.x86\_64/  
cp configs/kernel-4.2.5-x86\_64.config .config  
make oldconfig  
cp .config ~/rpmbuild/SOURCES/config-`uname -m`-generic  
vim Makefile*

*SUBLEVEL = 5-300  
EXTRAVERSION = .fc23.x86\_64*

*make kernelversion #* Make sure the version is **4.2.5-300.fc23.x86\_64**  
*make -j*

**Step 4: Modify the kernel source**

*cd fs/xfs/libxfs*

*vim xfs\_ialloc.c:  
 in function xfs\_dialloc()**near line 1698:* ***start\_agno = xfs\_ialloc\_ag\_select(tp, parent, mode, okalloc)***

This is where the AG number is decided, that is, to store data on HDD or SSD.

Currently, we hash parameters from *xfs\_dialloc* to randomly decide *start\_agno.*Therefore, we comment out the above line and add the following code:

***start\_agno = (hashvalue & 1) ? 0: 118; //0: HDD, 118: SSD***

In case an AG is full and unable to allocate more space, it will find the nearest available AG after it. *(around line 1724~1798)*

**Step 5: Build the modified XFS kernel module**

*cd ~/rpmbuild/BUILD/kernel-4.2.fc23/linux-4.2.5-300.fc23.x86\_64/  
make modules SUBDIRS=fs/xfs*

The kernel module is located at *~/rpmbuild/BUILD/kernel-4.2.fc23/linux-4.2.5-300.fc23.x86\_64/fs/xfs/xfs.ko*

**Step 6: Load the modified XFS kernel module (requires root)**

Now go to the MDS machine.  
Make sure NFS is stopped and all XFS disks are unmounted on MDS.  
Assume our new kernel module is placed at home directory.

*rmmod xfs # unload XFS module, if any  
insmod xfs.ko*

Now, we can mount the logical volume for pNFS and start the service to test it out.  
(steps to setup & start pNFS are omitted)

**Step 7: Test and verify the result**

Now go to one of the pNFS clients and mount to the MDS server.

Assume the mount point is at **/mnt/test**  
Randomly generate some files and see their inode numbers in XFS by this command  
*ll -lid ./\**

Check the inode number of files to confirm that files are placed inside the specified AG as we wish.

For example, in our case where files are randomly hashed to determine their location, you should see some files with inode number around 100 (on HDD) while others with numbers around 3959423075 (on SSD). Like shown below:

