

OHSM RELOCATION – FREQUENTLY ASKED QUESTIONS

sandeepksinha@gmail.com

Jan 12th, 2009

1. What policies are supported by OHSM relocation?

Relocation Policies can be based on:

1. File Modification Age
2. File Access Age
3. File Size
4. File I/O Temperature
5. File Access Temperature

2. Can we have more than one relocation policy applied simultaneously to the FS?

Yes, we can have more than one policy simultaneously. The data structures are capable of capturing more than one relocation policy at the same time. This information will be present in the XML policy file and will be parsed by the OHSM XML parser. Each relocation policy will have a unique relocation policy ID called, RelPol-ID. While enforcing relocation use can specify the RelPol-ID to specifically trigger a particular policy. You can definitely trigger multiple relocation policies at the same time.

3. What happens if an object falls into relocation criterion under more than one policy?

In this case only one preferable policy will be enforced. Deciding which, initially would be decided by OHSM and would be predictable. So, that it can be used at the time of writing the XML policy file.

4. What are the data structures used for relocation?

Ext2 inode has been modified to contain ohsm structure that contains the following fields:

- | | |
|---------|-------------------------|
| 1. TID | (Tier ID) |
| 2. FMA | (File Modification Age) |
| 3. FAA | (File Access Age) |
| 4. FIOT | (File I/O Temp) |
| 5. FAT | (File Access Temp) |

Some of these are already present with different names.

Data structures used for relocation:

1. List of all the relocation policies applied to the FS.
2. Data block bitmap array.
3. Temporary inode.

These data structures are based on the initial design and goals of OHSM.

5. Where we store the data structures for relocation?

Some of these data structures will be stored in the FS as global, but most of them will reside in the OHSM itself.

6. Where do we extract the data from, which is used for relocation?

The data for relocation resides in inode itself (ohsm structure, added by OHSM), which is checked to qualify an inode for relocation.

OHSM scans the file system to search for qualified inodes for relocation. We allocate a ghost inode with empty data blocks from the target tier. Then we copy the contents of source inode data blocks to target data blocks. This inode is temporary and is freed as soon as the relocation of an object is completed. We also free the data blocks of the original inode object. Then we use the OHSM's very own 'Tricky Copy' algorithm. The algorithm reads buffer heads of source and target data blocks simultaneously and assigns the b_data pointer of source block to target block. And then marks the target buffer dirty to cleanly achieve its goal.

7. Why can't we have an analytical engine which analyses and runs relocation periodically?

We can have an analytical engine which will analyze and run relocation periodically. We should make sure that we don't call relocation too often, as that would become a bottleneck and surely can make the Filesystem slow.

8. Can we have unconditional movement of data across tiers, automatically whenever the relocation is triggered?

Yes we can have unconditional movement of data across tiers. It means that we can have a policy of moving everything from tier X to tier Y, whenever such a relocation policy is triggered.

9. Can we specify more than one condition together under one relocation policy? Something like, move mp3 files from tier X to tier Y, when size>10M && FIOT>100.

Yes, surely we can. We do support such constructs in XML policy file and also inside the OHSM.

10. Will the path of the file change after relocation?

No, the path of the file remains the same after relocation. This is what helps us keep the Filesystem live.

11. *Will the file be accessible when it is being relocated?*

Yes, the file system will be online and so the files will be accessible for both read and write. Even for user applications which have already opened the file for read and write, there will not be any thing visible to those applications. Relocation remains completely transparent to those applications.

12. *What if a file has already been opened by a user space application and this file becomes a candidate for relocation. Do we wait for the application to close that file? Or do we just wait for the file to complete its I/O?*

Refer answer 11.

13. *Do we check for space issues across tiers at the time of relocation? How does OHSM handle such situations?*

Of course we should check the space issues before relocating a file. If there is not sufficient space on the specified destination tier then it will be relocating across tiers as per tier priority, but that still remains as a future enhancement. Initially, we intend to warn the user of such a situation and abort relocation.

14. *How relocation handles symbolic and hard links.*

For this issue relocation doesn't have to care much, as we are not allocating new inodes, the inode remains intact. We are updating the existing inode only so the links remain valid and doesn't require any modification.

15. *What all changes in ext2 data structures will be required, does this affect ext2 in anyway?*

The only change OHSM requires in ext2 is the inode structure. It is already mentioned in answer 4. This increases the current ext2 inode size from 128 bytes to 256 bytes. But that doesn't affect ext2 working much. We are not using 128 bytes, but it always preferable to have inode size in powers of two.