OpenZFS

`zfs diff' Optimizations

David Chen, Sanjeev Bagewadi
Nutanix

zfs diff

OpenZFS

- '**`zfs diff`**' shows path of files modified between snapshots
- We use it to allow incremental backup with 3rd party software
- Works very well for small set of changed files.

```
$ sudo zfs diff pp@001 pp@002
+         /pp/file.5
M         /pp/
M         /pp/file.2
M         /pp/file.3
M         /pp/file.4          (+1)
-         /pp/file.5
R         /pp/file.1 -> /pp/new.1
```

Problem

OpenZFS

- Very slow with large number of entries (eg. 1million changed files/directories)
  - E.g. zfs diff takes ~60min with 1 million changed files/directories
  - dnode to path translation is the bottleneck
- Hardlinks may break ‘zfs diff’
  - E.g. create hardlink for `file.6` in another dir and remove the new link.

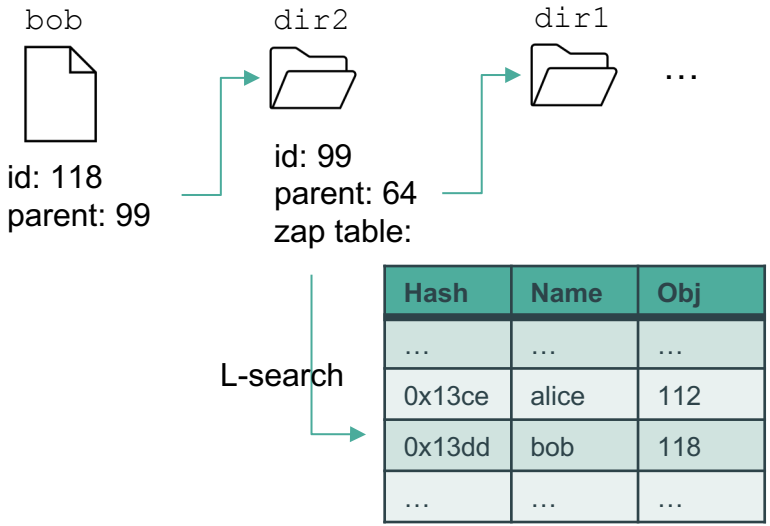
```
$ sudo zfs diff pp@002 pp@003
M           /pp/
+           /pp/dir0
-           /pp/file.6
```

  - Depending on version, may see error “`Unable to determine path or stats for object 107`” instead

Problem: path translation

OpenZFS

- dnode to path translation is slow:
  - Because dnode doesn't store file name, only store parent id in SA
  - Linear search on parent directory to find matching id
  - Repeat the process all the way to root



Hash	Name	Obj
…	…	…
0x13ce	alice	112
0x13dd	bob	118
…	…	…

Problem: path translation

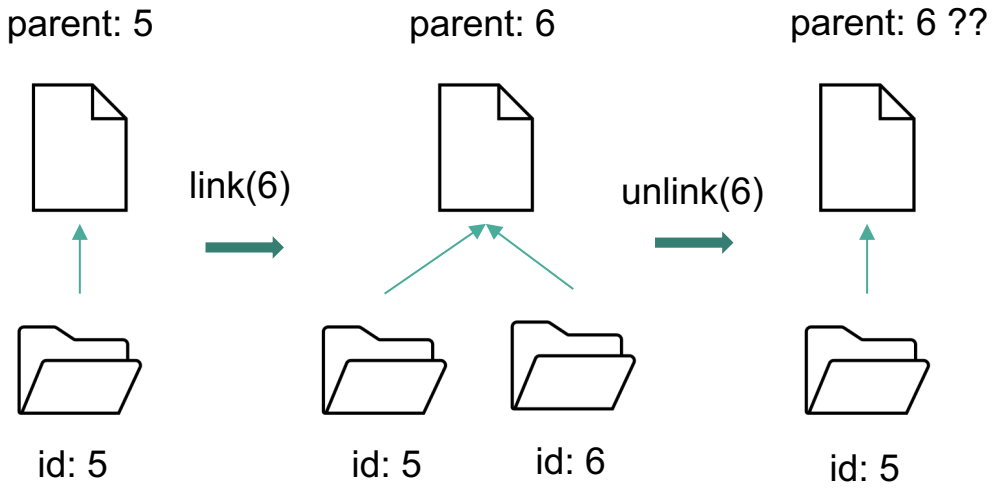
OpenZFS

- Each path component costs O(n)
- Common directories are translated repeatedly. Could benefit from caching.
- Could cause a huge unwanted churn in ARC
  - Due to sequential traversal of directory ZAP entries
- With a large number of changed entities, this becomes a serious issue.

Problem: hardlinks

OpenZFS

- When creating hardlink, SA parent points to parent of new link
- If we remove new link, SA parent becomes stale



Solution: Regular File/Directory

OpenZFS

- Add a **`linkname-hash`** in the dnode
  - `linkname_hash` : hash value of the ZAP entry for the dnode in parent-directory
  - Added a new SA : `SA_ZPL_LINKNAME_HASH`
  - Using `linkname_hash`, search in parent-directory reduced to O(1).
  - linkname_hash is updated during rename(2)

Sample output : linkname_hash

OpenZFS

```
Object  lvl   iblk dblk dsize dnsize lsize lused %full type
 128     1    128K 512    0    512    512     0   0.00  ZFS plain file(K=inherit) (Z=inherit)
                                                 184   bonus  System attributes
        dnode flags: USERUSED_ACCOUNTED USEROBJUSED_ACCOUNTED
        dnode maxblkid: 0
        path      /alice/bob
        linkname_hash 9d23b52
        uid     0
        gid     0
        atime     Sun Sep 27 06:23:51 2020
        mtime     Sun Sep 27 06:23:51 2020
        ctime     Sun Sep 27 06:23:51 2020
        gen       39
        mode      100600
        size      0
        parent    2
        links     1
        pflags    40080000004
        xattr     129
```

Solution: Hardlinks

OpenZFS

- For Hardlinks :
  - Add a ZAP dnode which holds one entry for each hardlink :
    - Each entry is a pair of : `<parent_dnode:linkname_hash>`
    - The entries are updated during rename(2) and unlink(2)
  - ‘zfs diff’ lists the first entry found.

Solution: Hardlinks

OpenZFS

```
Object  lvl   iblk   dblk  dsize  dnsize  lsize   lused  %full  type
        5    1   128K    512      0     512    512       0    0.00  ZFS plain file
                                           192   bonus  System attributes
       dnode flags: USERUSED_ACCOUNTED USEROBJUSED_ACCOUNTED
       dnode maxblkid: 0
       path    /dir1/file1
       linkname_hash ae8f549
       linkzap 384
       parent  256
       links   2


Object  lvl   iblk   dblk  dsize  dnsize  lsize   lused  %full  type
      384    1   128K     4K     2K     512     8K      2K  100.00  zap
                                          64   bonus  uint64
       dnode flags: USED_BYTES USERUSED_ACCOUNTED USEROBJUSED_ACCOUNTED
       dnode maxblkid: 1

       linkzap info:
               object = 5

               259 238918162 =
```

```
bash# linkzap list zpool1/fs1 5
Parent  Hash    Linkname
256     ae8f549 file1
259     e3d9a12 file2
```

Additional Optimizations

OpenZFS

- Userland changes :
  - Top level directories have higher number references.
  - Cache the mapping (dnode -> name) for directories
  - Modified name-lookup to just fetch the name of the dnode, instead of full-path.
  - Rest of the path can be derived from the cached entries.

  Eg : If following are changed files :

  ```
  /zpool/fs1/a/b/c/file100.txt - file199.txt
  /zpool/fs1/a/b/d/file200.txt - file299.txt
  ```

  Entries for ‘a’, ‘b’ are will be looked-up for all 200 changed files.
  Caching them helps.

Summary of changes

OpenZFS

- Ondisk changes :
  - New SA properties added : `SA_ZPL_LINKNAME_HASH, SA_ZPL_LINKZAP`
  - An extra dnode consumed for each hardlinked file.
  - Add linkzap obj variant for TX_LINK and TX_RENAME in ZIL
- Changes to ioctl `ZFS_IOC_OBJ_TO_PATH` to refer to above SA properties during lookup. Fall back to regular method in its absence.
- Should be portable to community code
  - Should be able to add code to populate `SA_ZPL_LINKNAME_HASH`
  - *Fixing existing hardlinks might not be feasible.*