
whamcloud

whamcloud

SC11
Nov 15th, 2011

Parallel Directory Operations of Lustre

- Liang Zhen

  Whamcloud, Inc.

  liang@whamcloud.com

Why need PDO (Parallel directory operations)

- For many HPC applications, performance of single directory operations is critical
- Threads vs State machines
  - Threads based programming is much much easier than state machines based programming
  - Well designed multiple threads system has good performance on SMP system
- However
  - multiple threads system can kill performance if it's not well designed
    - Could even be a lot worse than single thread system
  - Overhead of thread context switch is very expensive
  - All Exclusive locks can't scale well for many threads
- Lustre has a lot of threads
  - Huge mount of thread context switches

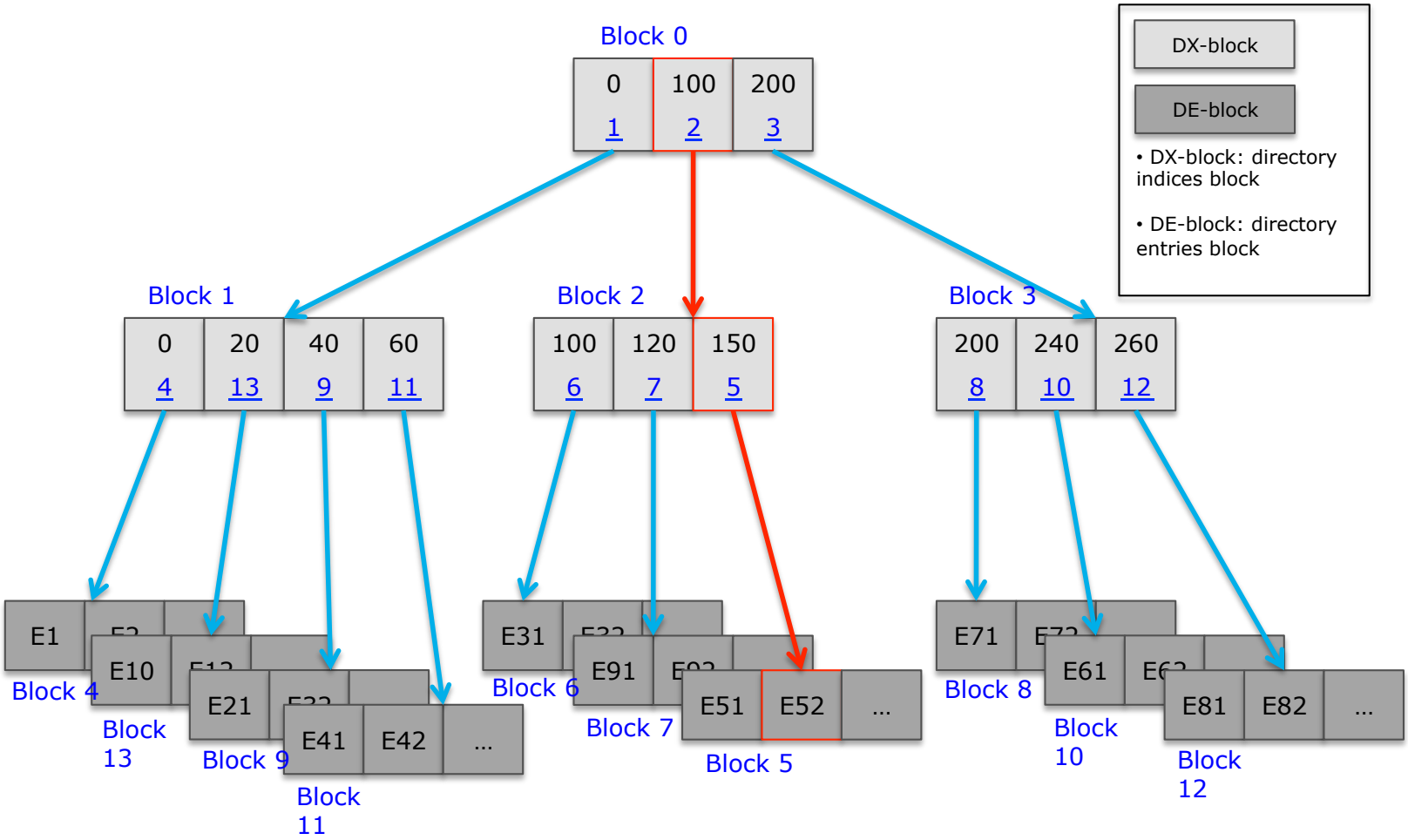
How Lustre protects directory on 1.8.x

- A directory is protected by a single LDLM lock
  - It works just like an expensive rw_semaphore for directory operations
  - By default we have max to 512 service threads to handle metadata requests, but some customers require more than 512 threads
  - Assume all threads are waiting on a single lock
- Using VFS interface to access backend filesystem (ldiskfs)
  - VFS APIs always take per-inode lock i_mutex to protect tree topology
  - On Lustre 1.8.* or earlier versions, directory tree topology is _not_ really protected by i_mutex because operations have already been serialized by LDLM lock

How Lustre protects directory on 2.x

- PDO ldlm lock
  - For example
    - create/unlink will take CW lock on directory, PW lock on name entry
  - Parallelized operations for file creation
    - Object creation on backend filesystem
    - Permission check
    - Name entry Lookup
    - OI (Object index) operations
    - Creation of OST objects
  - Performance increased
- No VFS on MDS stack
  - VFS is replaced by MDD/OSD
    - Directly access backend filesystem
  - Name entry operations are still serialized by rw_semaphore in OSD
    - Name entry insert
    - Name entry remove
    - Name entry lookup (READ)
    - They are expensive

Ext2/3/4/Ldiskfs directory

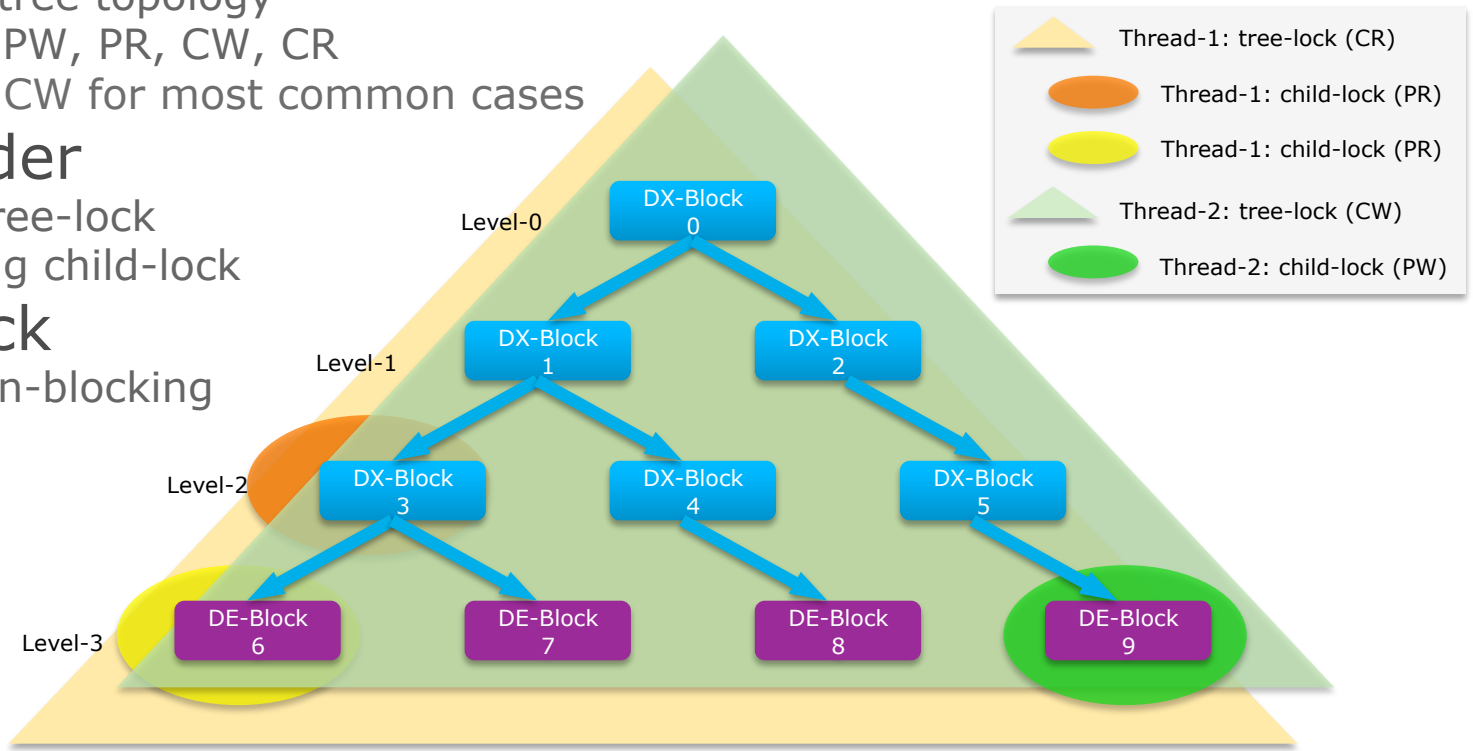


Operations on htree based directory

- probe htree-path
- Insert name-entry to DE-block
- Remove name-entry from DE-block
- Iterate over all DE-blocks
- Split DE-block
- Split DX-block
- Grow tree depth
  - Support N-level htree
- How to parallelize these operations?
  - No loss in performance of FFP
  - w/o rewriting htree directory of ldiskfs

Htree-lock

- Child-lock
  - may be used to protect any node in htree
    - Node == DX/DE-block
- Tree-lock
  - protect the tree topology
  - Modes: EX, PW, PR, CW, CR
    - CR and CW for most common cases
- Locking order
  - Must take tree-lock before taking child-lock
- scalable lock
  - Blocking/non-blocking
  - skiplist



Graph-2 : htree and htree-lock

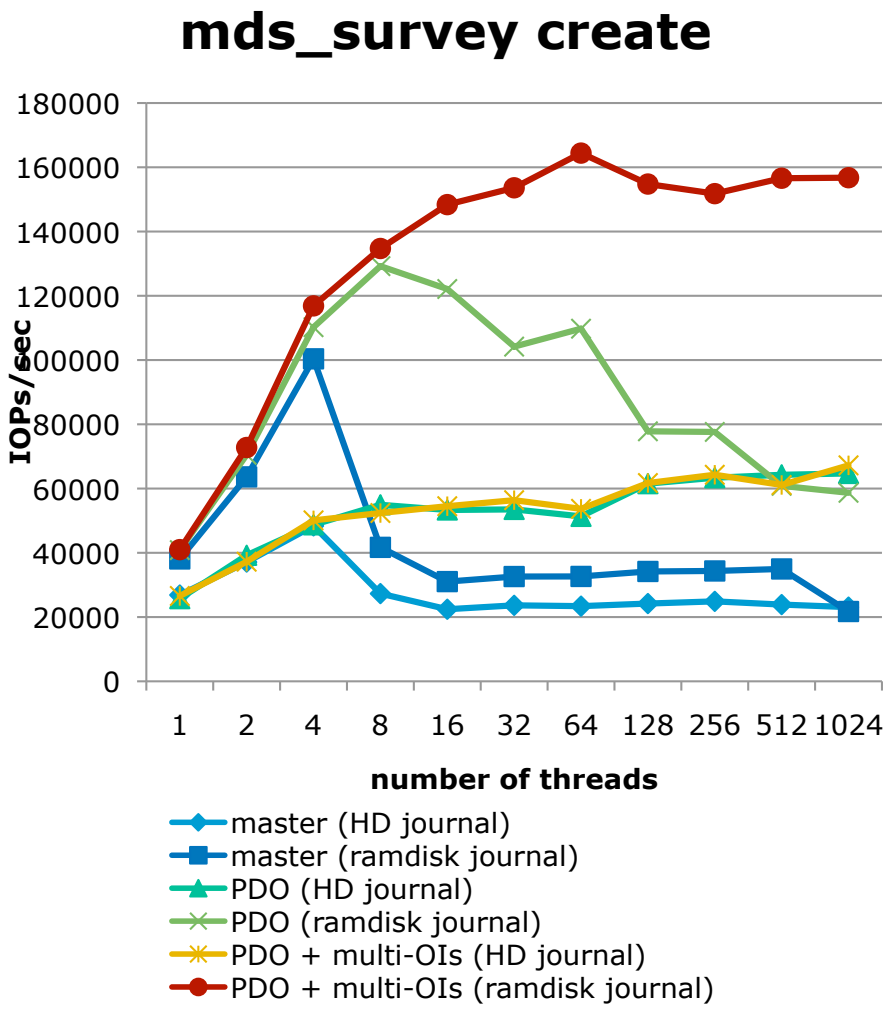
Protecting htree dir by htree-lock (1/2)

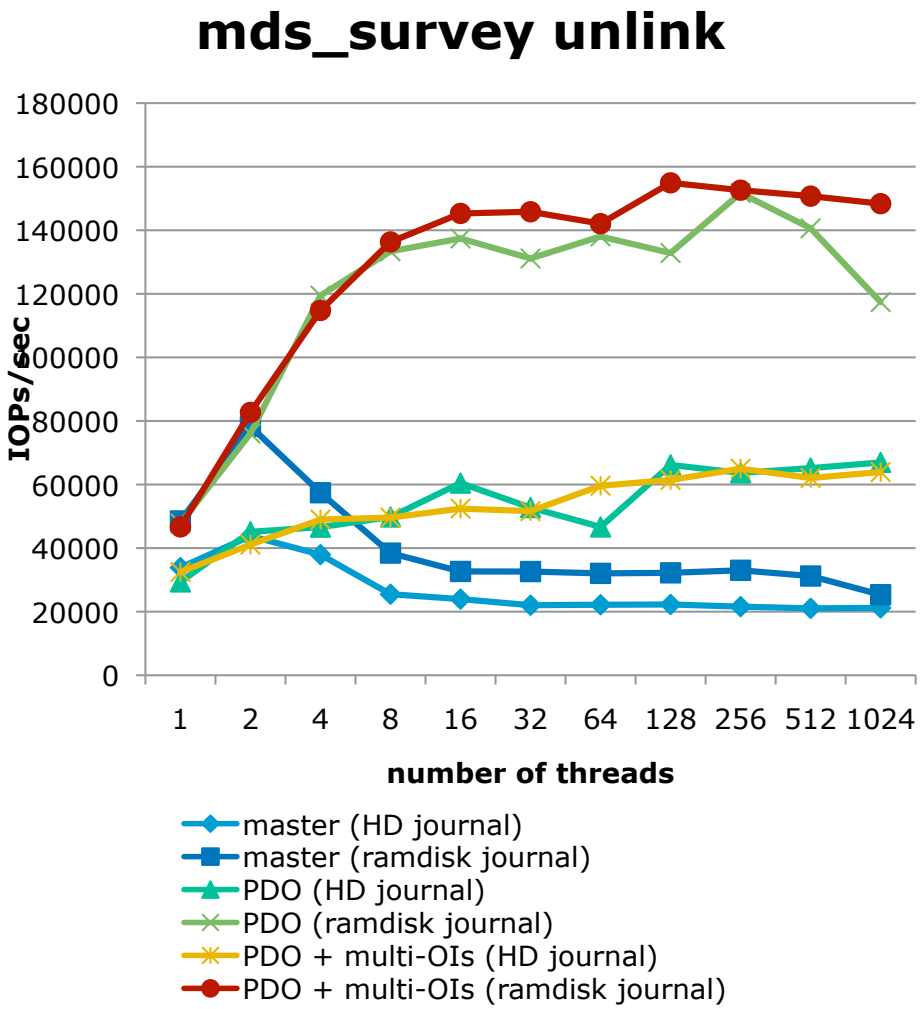
- preliminary idea
  - Child-lock only protects DE-block
    - Search/insert/remove entry from DE-block
  - Tree-lock protect all other operations
    - Probe htree-path
    - split DE-block
    - split DX-block
    - grow tree depth
  - However
    - split DE-block for each ~100 creation
      - Block size is 4K, each entry has name string + extra, so bytes of each entry ~= 40byets, and each DE-block can fit in ~100 entries
    - We have hundreds or thousands service threads
      - Always some threads want to exclusively lock the tree because they need to split DE-block
    - Performance results are not cool enough

Protecting htree dir by htree-lock (2/2)

- Improvement
  - Child-lock protect DE-blocks and the last level DX-blocks
    - Lock DE-block for search/add/remove name entry
    - Lock the last-level DX-block on DE-block splitting
  - Tree-lock
    - Tree-lock wouldn't protect tree topology change to last level nodes
      - Split DE-block (leaf node) is protected by child-lock
    - Take exclusive tree-lock for splitting DX-block (intermediate node)
      - Each DX-block can contain 512 pointers to DE-block, each DE-block can container ~100 entries
      - 512 * 100 = 51,200, chance to lock the whole tree is 1/51,200, which is small enough
    - Take exclusive tree-lock for growing htree
    - Other operations just take shared lock (CW/CR)

Graphs





Thank You

- Liang Zhen

  Whamcloud, Inc.

  liang@whamcloud.com