
whamcloud

whamcloud

Orlando Fl
April 2011

Lustre Metadata Performance Improvements

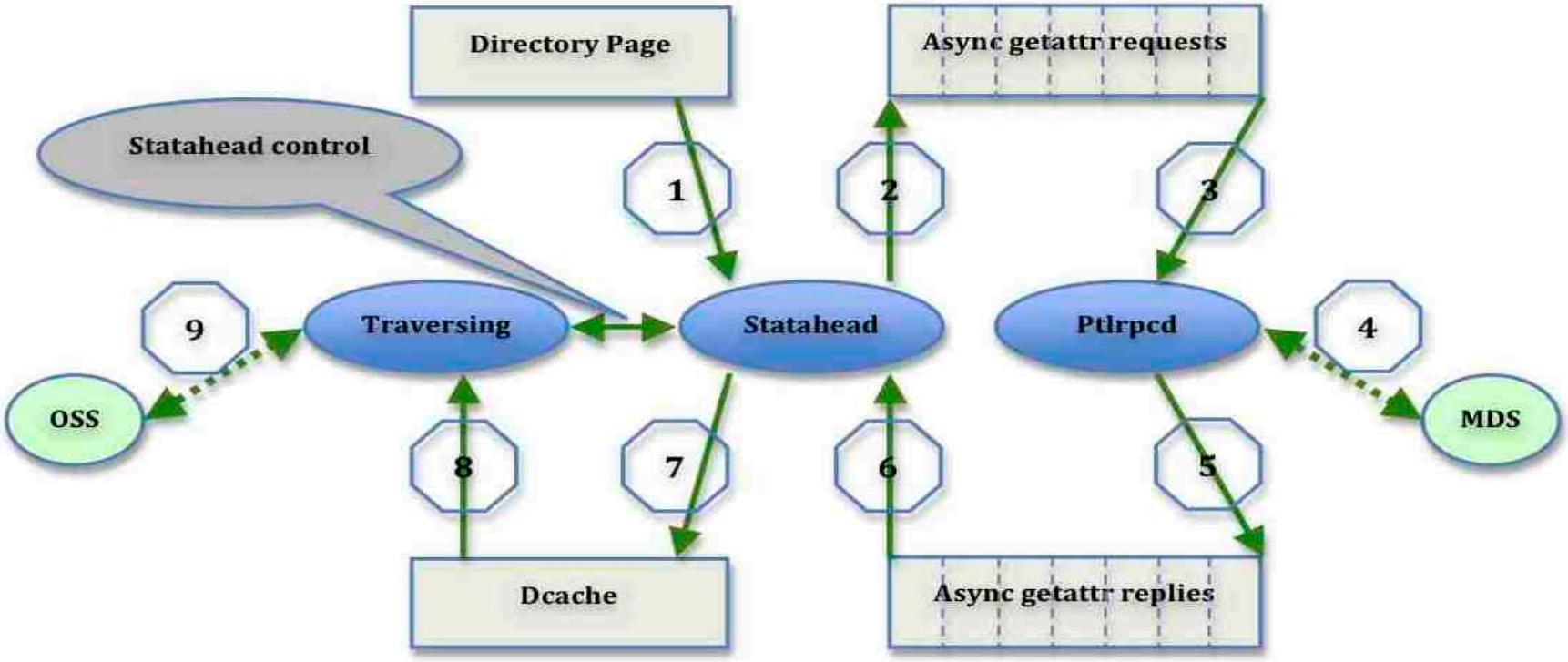
- Fan Yong
  Senior Engineer
  Whamcloud, Inc
  yong.fan@whamcloud.com
- Liang Zhen
  Senior Engineer
  Whamcloud, Inc
  liang@whamcloud.com

Overview

- What we focused on
  - Sequential traversing large directory on single client: ls/du/find
  - Metadata performance SMP scaling: opencreate/unlink
- Accelerate sequential traversing directory
  - Asynchronous glimpse lock
  - Statahead load balance
  - Aggregate attributes to save RPCs
  - Efficient ldlm/object hash
- New progress on Lustre SMP scalability
  - Flexible RPC scheduler
  - Parallel directory operations

Accelerate traversing directory (prolog)

- Existing statahead
  - Only prefetch attributes from MDS
  - Single pipeline for async getattr



Asynchronous glimpse lock

- Prefetch glimpse lock from OST
  - Async RPC without waiting
  - Cache file size/blocks if AGL granted
  - No glimpse callback if OST can't grant AGL
  - Re-enqueued by traversing thread if AGL non-granted/cancelled

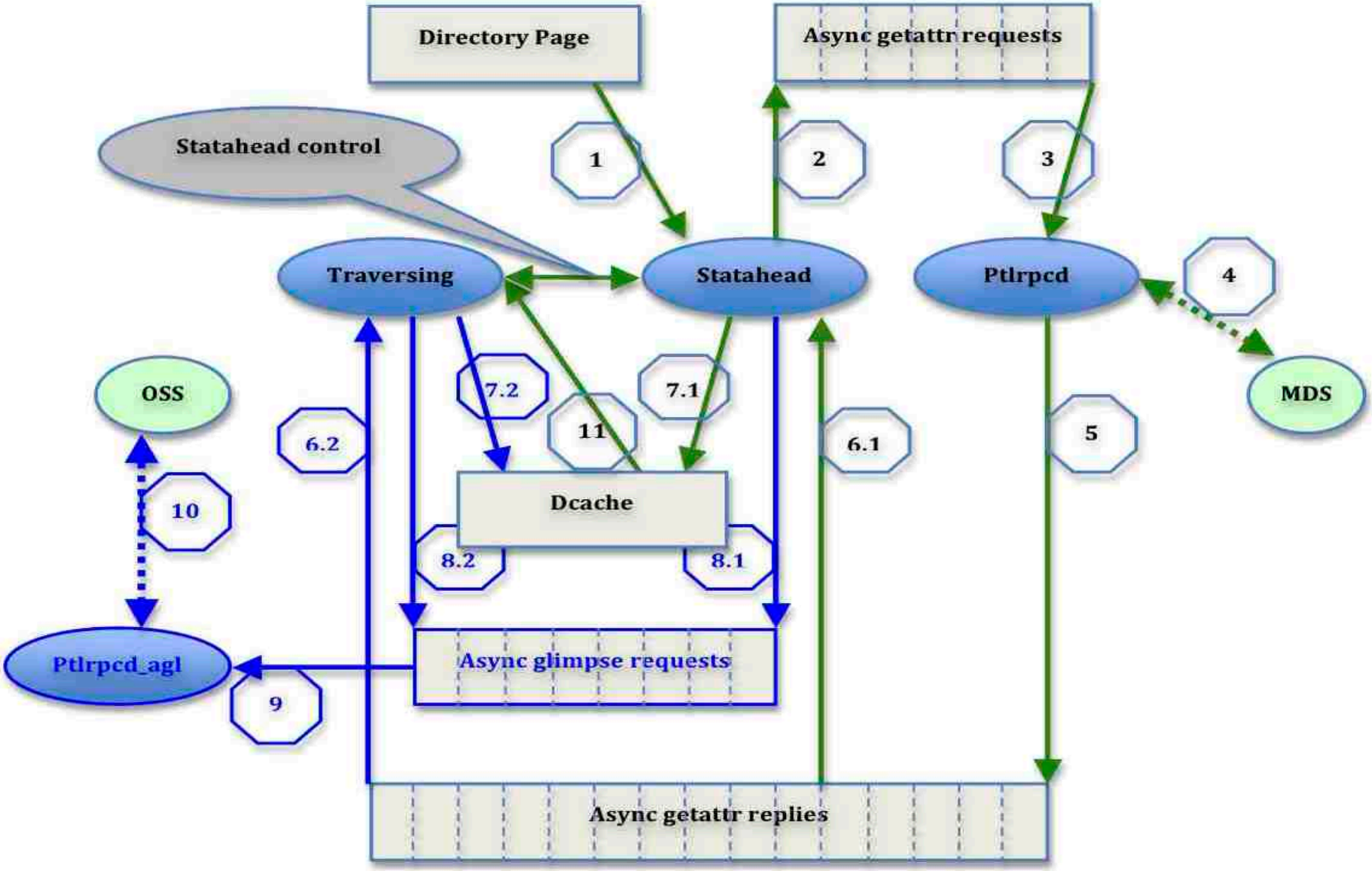
May cause additional RPCs if someone held conflicting locks on some items, but it works well for large directories, since stat/read cases are more common than writes.

Statahead load balance (1/2)

- Independent portal RPC service (ptlrpcd_agl)
  - Dedicated to processing AGL RPC
    - Single ptlrpcd on client may be overloaded to process all non-I/O async RPCs when traversing large directory
    - Unbalanced load, other CPU(s) on the same client maybe idle
  - Separate pipelines for async getattr and AGL
    - Async RPCs for getattr and glimpse can be processed in parallel, no contention between ptlrpcd and ptlrpcd_agl
    - Decrease the unnecessary dependence between MDS-side RPC and OSS-side RPC

Statahead load balance (2/2)

- Unbalanced load between statahead and traversing
  - Statahead thread is so slow that RPC processing time can't be hidden
  - Statahead thread
    - Async getattr pipeline Input & Output
    - AGL pipeline Input
  - Traversing thread
    - AGL pipeline Output
- Pool for async getattr replies to be interpreted
  - Traversing thread helps to process async getattr replies
  - Statahead thread
    - Async getattr pipeline Input & Output
    - AGL pipeline Input
  - Traversing thread
    - Async getattr pipeline Output
    - AGL pipeline Input & Output



Aggregate attributes to save RPCs

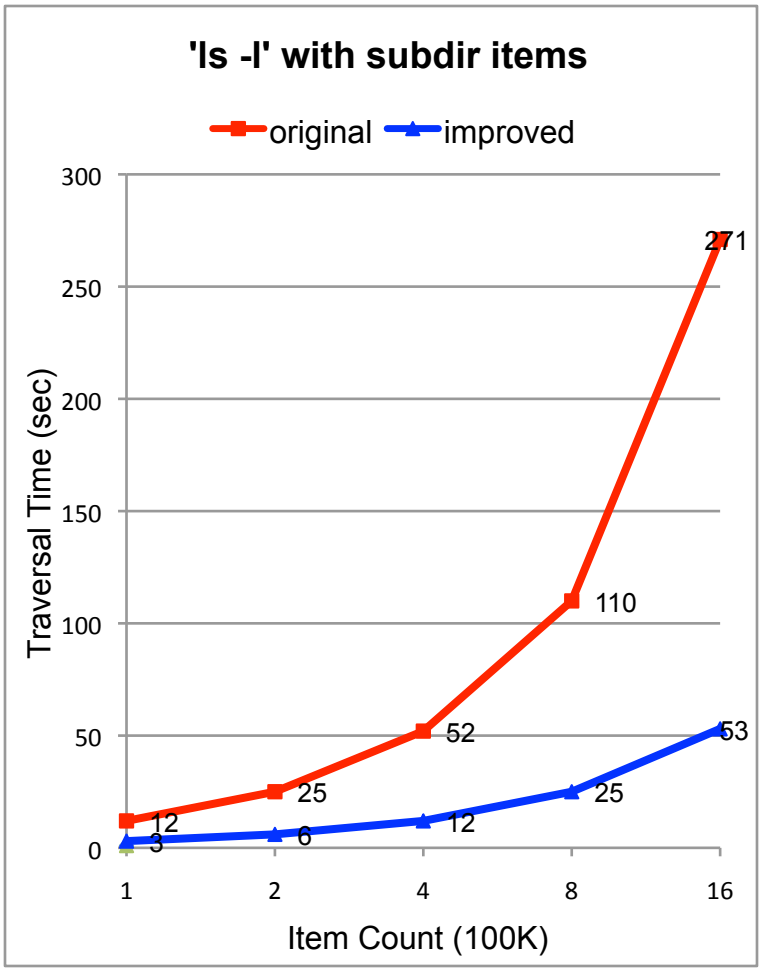
- MDS-side attributes for traversing directory:
  1) Basic attributes, like mode/owner/flags
  2) Stripe information
  3) Access Control List
  4) Default ACL for directory

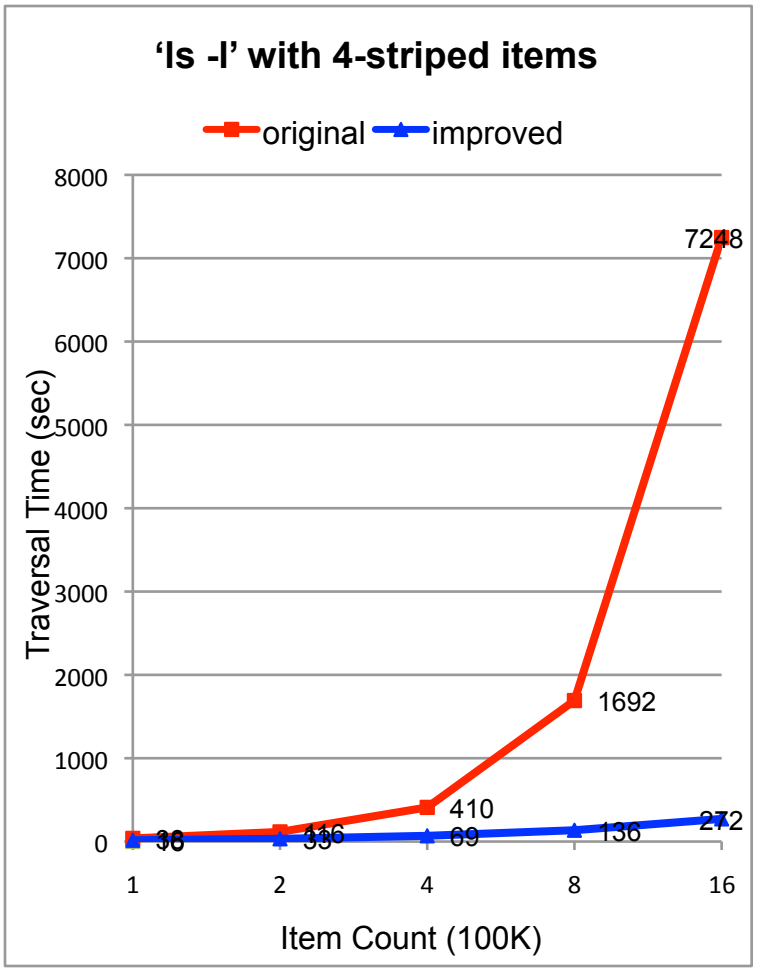
  Currently, 1) & 2) & 3) can be obtained through single RPC; and further combining 4) can save about 50% RPC for the best cases.
  - Maybe more in future

Efficient ldlm/object hash

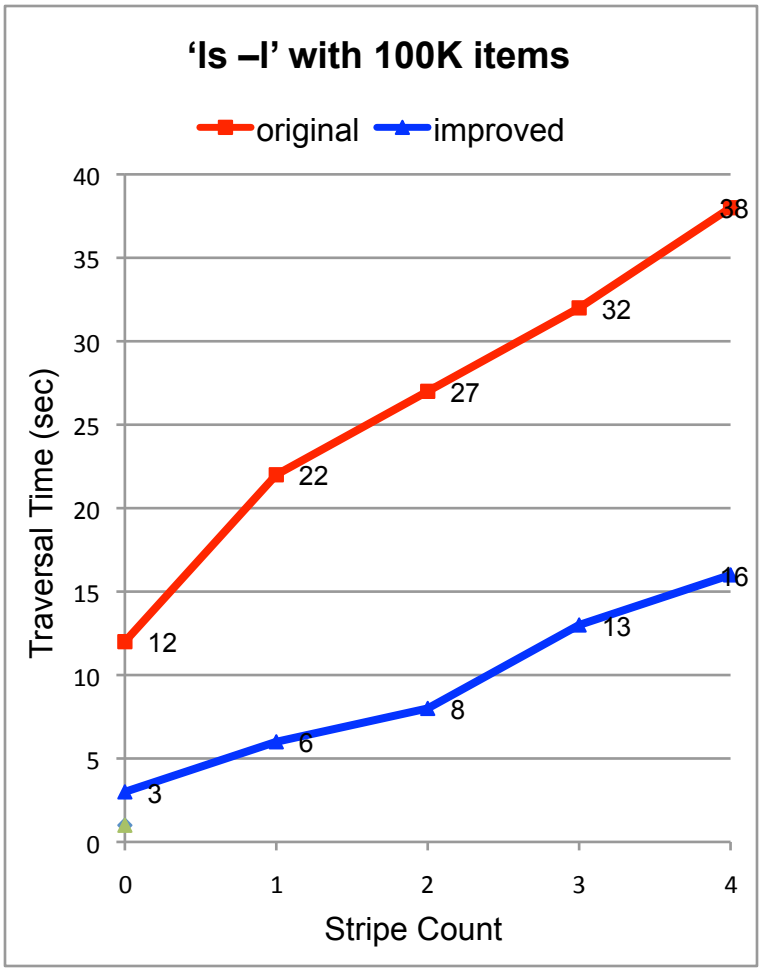
- Hash greatly affects traversing performance
  - MDS: object hash & ldlm hash
  - OSS: ldlm hash
  - Client: object hash & ldlm hash
- Hash bucket depth is quite important
  - Basically depends on hash function
    - Scatter ldlms/objects throughout buckets as evenly as possible
  - RAM size and filesystem size are associated
  - Experience: Max depth for millions of ldlms/objects hash
    - lustre-2.0 approaches 3K
    - no more than 50 is expected

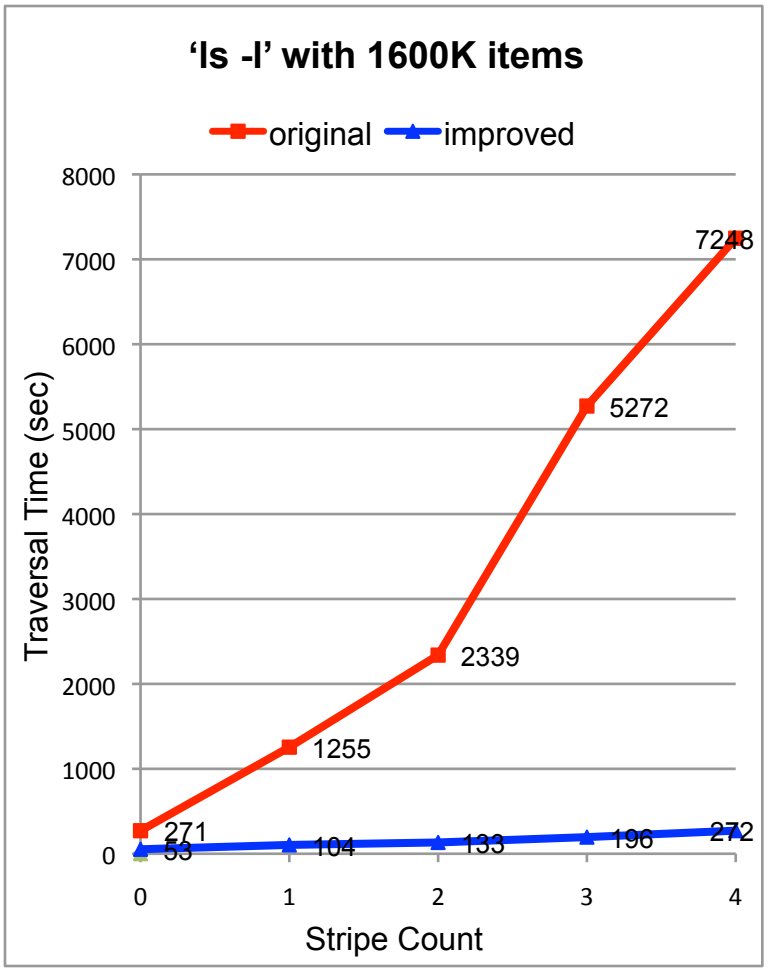
'ls –l' on single client (1/2)





'ls –l' on single client (2/2)

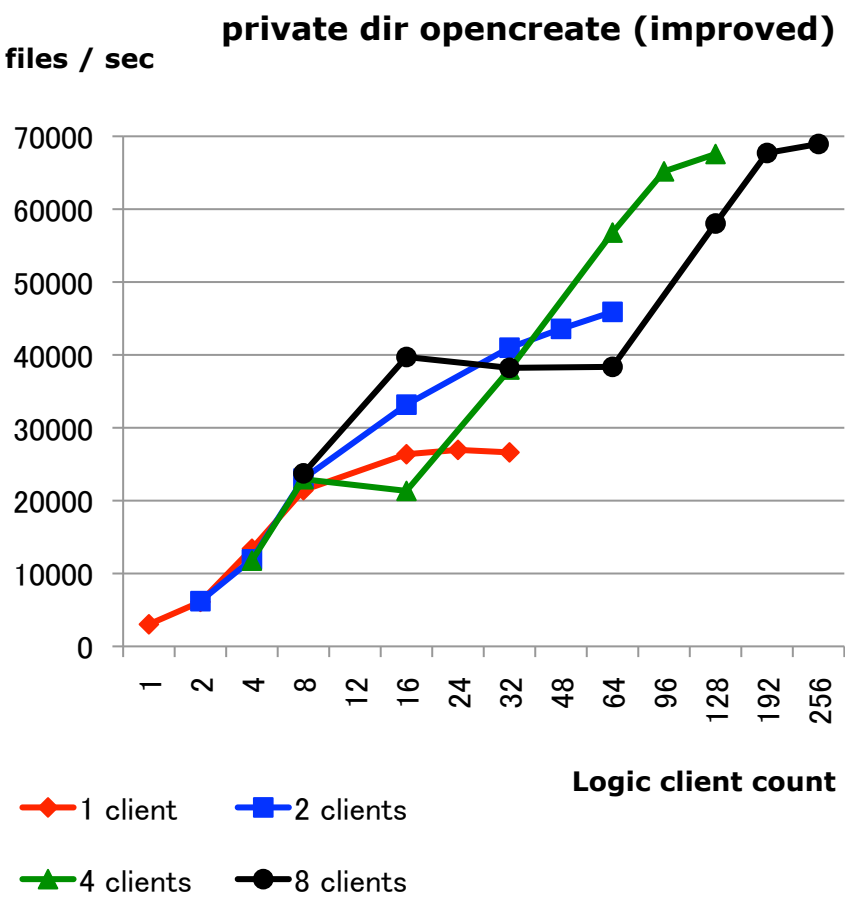


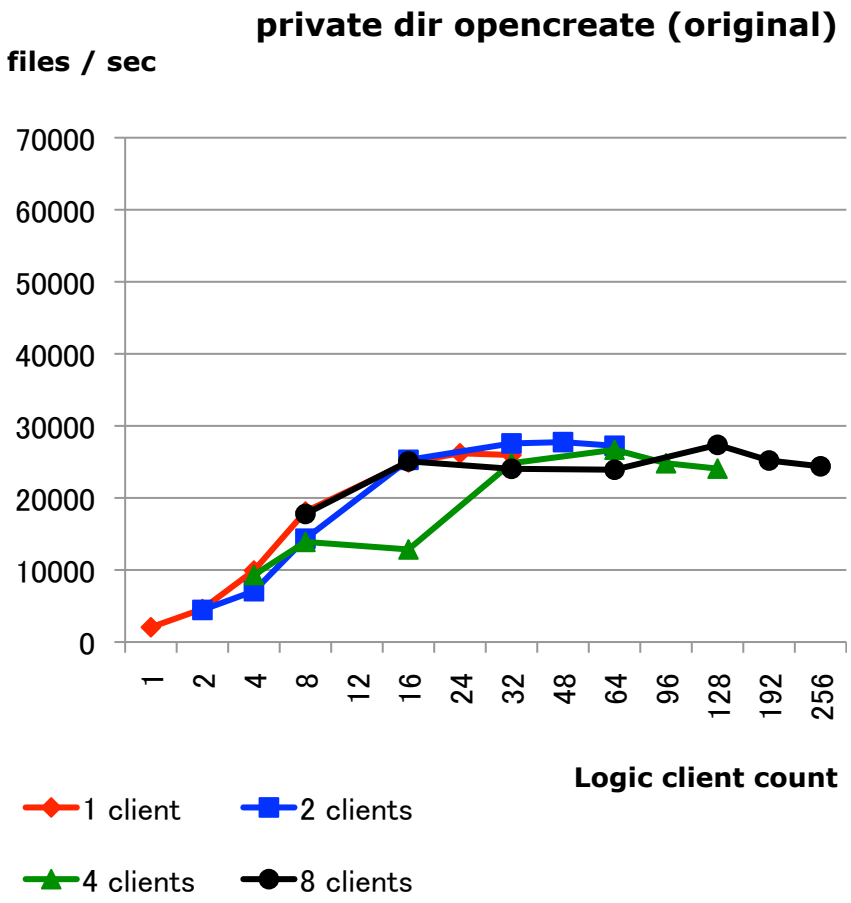


Flexible RPC scheduler – SMP scaling

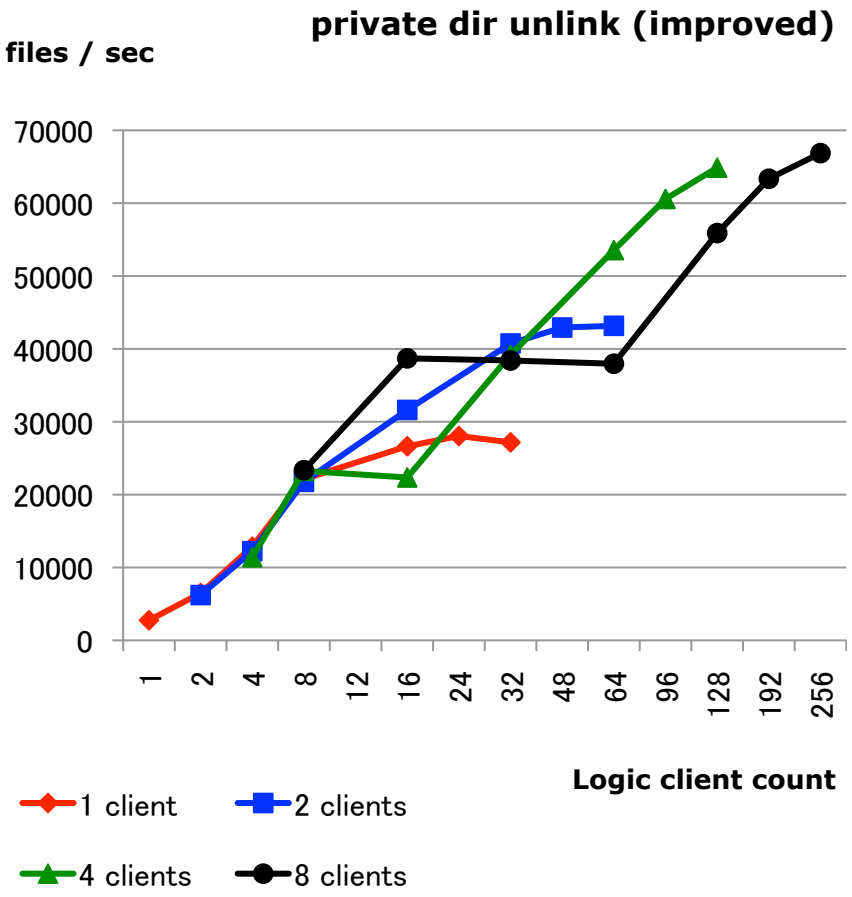
- Metadata performance is CPU bound for scaling
- Localize lifecycle of RPC request/reply on the same CPU unit all along
  - CPU unit: socket, core, processor (hyper-thread)
  - Separate RPC queue for each CPU unit
  - General optimization for service side RPC processing
    - RPC queue lock contention within CPU unit
    - No data migration between CPU units
  - Little help for shared directory operations
    - High contention on operation target(s)

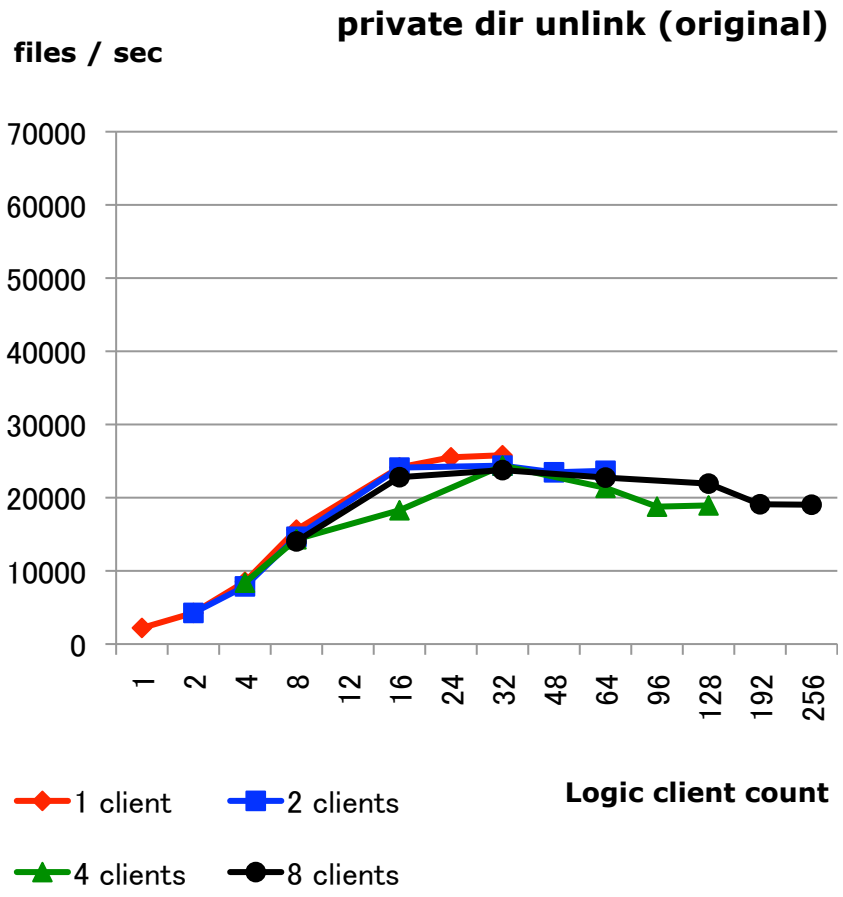
Opencreate under private dir





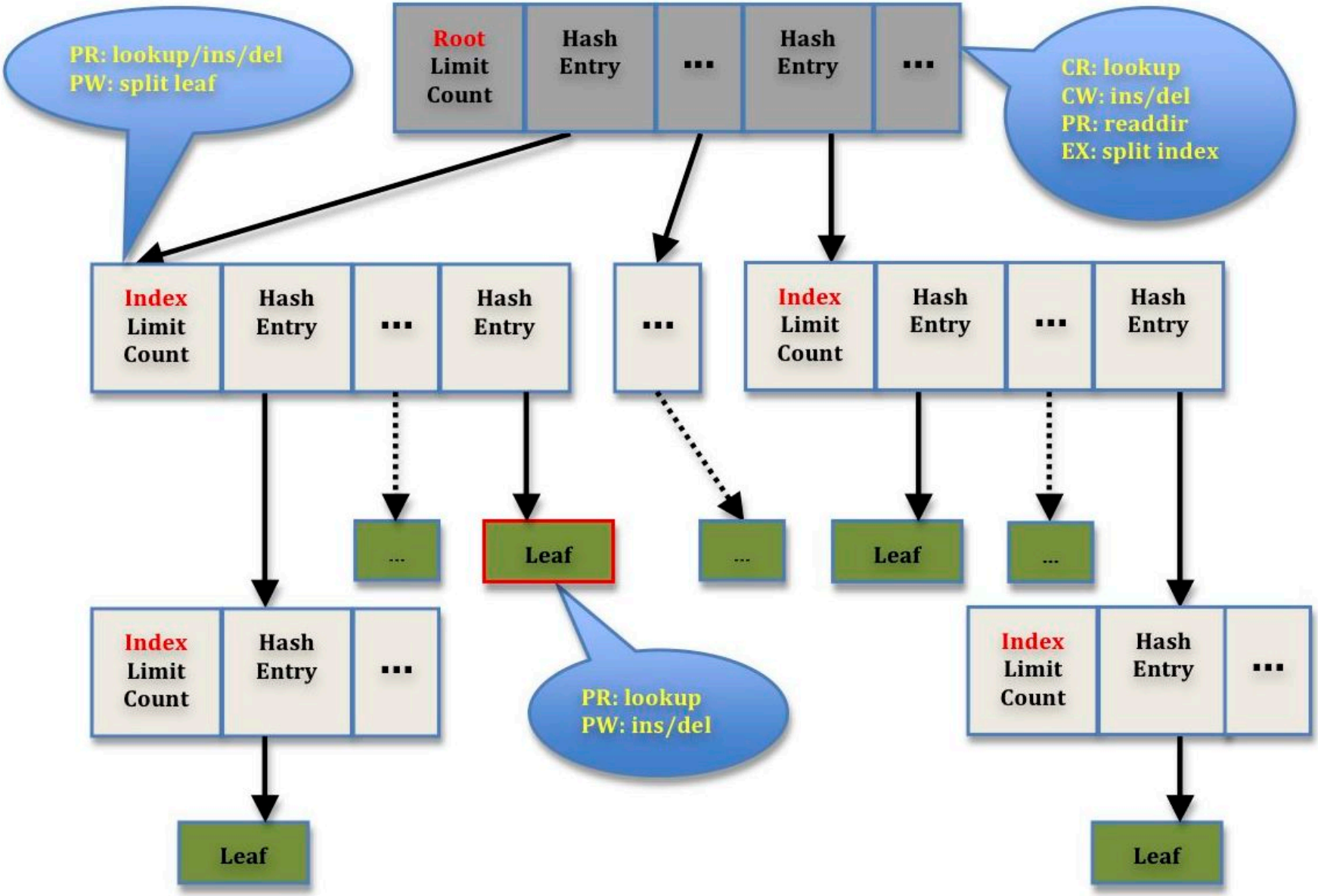
Unlink under private dir



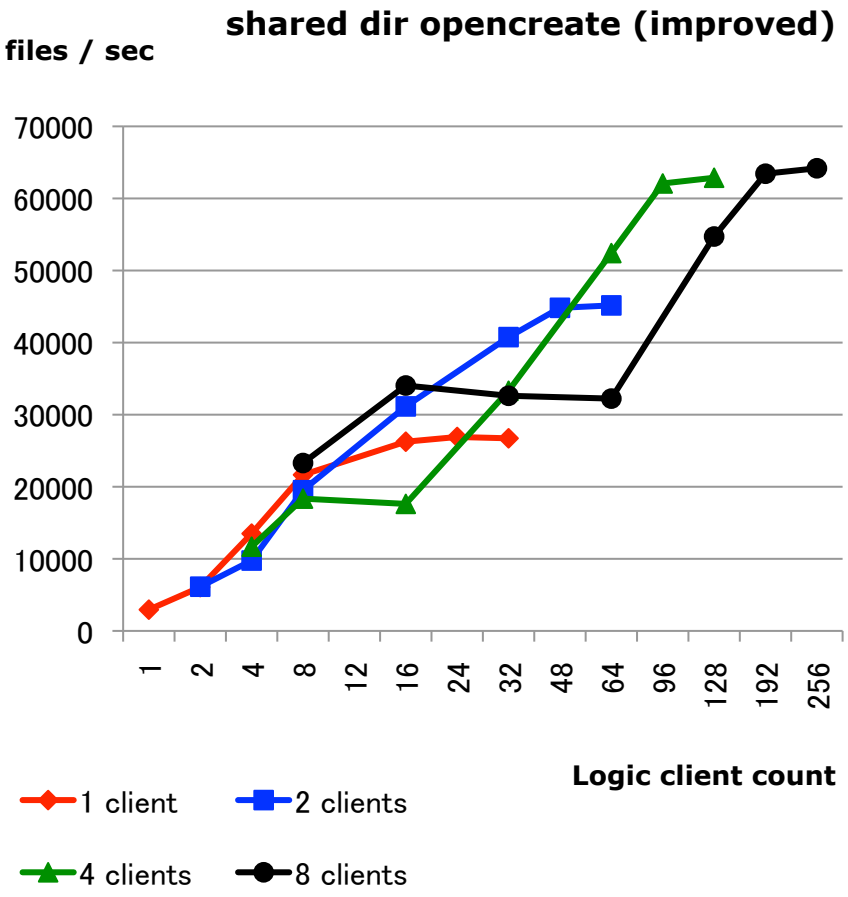


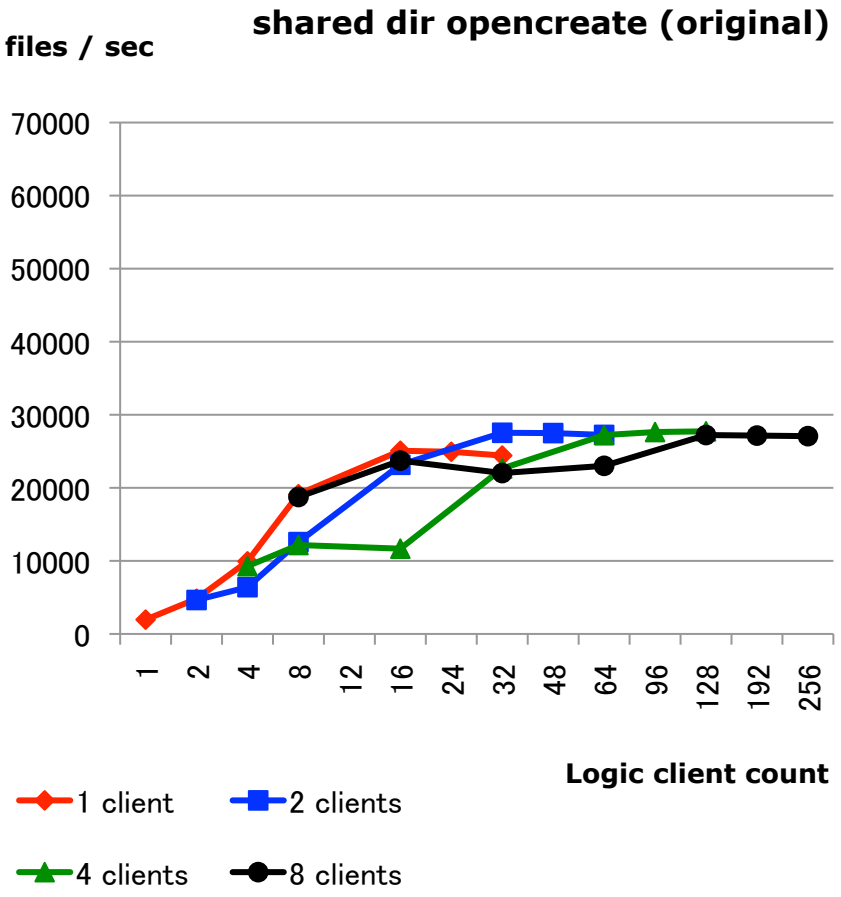
Parallel directory operations

- Policy: reduce contention range as far as possible
  - At least two levels contention: ldlm & ldiskfs
  - Already support ldlm level PDO
- IAM (Index Access Module) based ldiskfs
  - Contain ldiskfs level PDO control
  - Incompatible disk format with lustre-1.8
- EXT3/EXT4 based ldiskfs
  - Bottleneck: single read-write semaphore protects whole parent htree
    - Most concurrent operations at ldlm level are serialized here
  - Improvement:
    - Only lock leaf node if non-split
    - Backtrack to lock index node(s) when split
    - Local lock hash

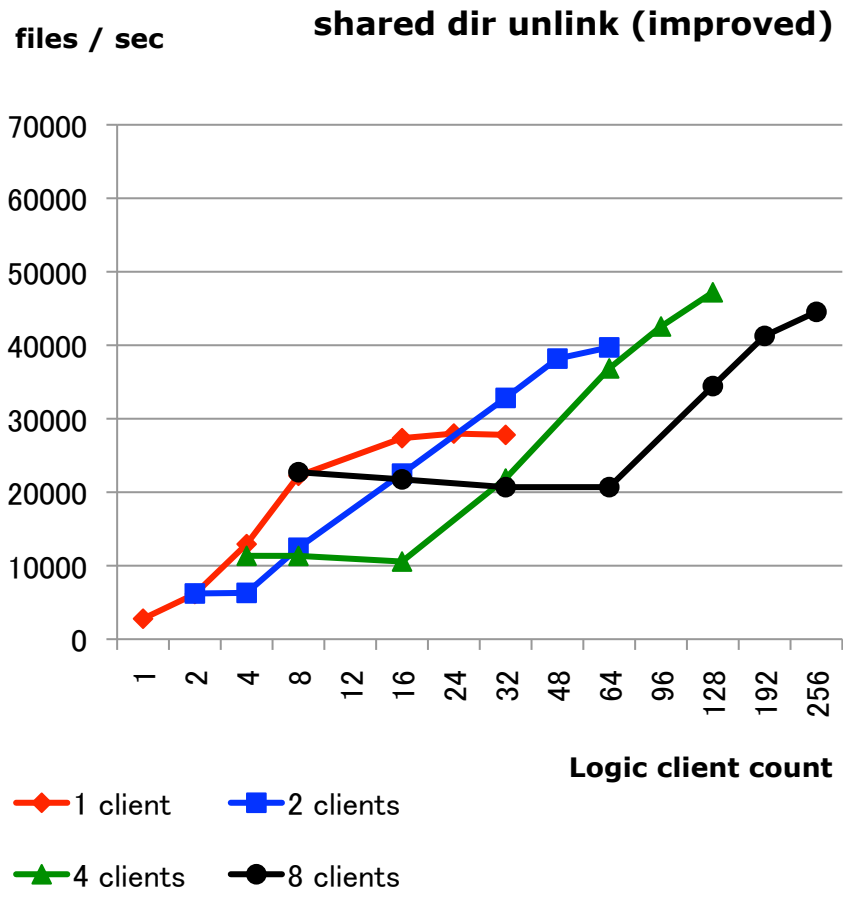


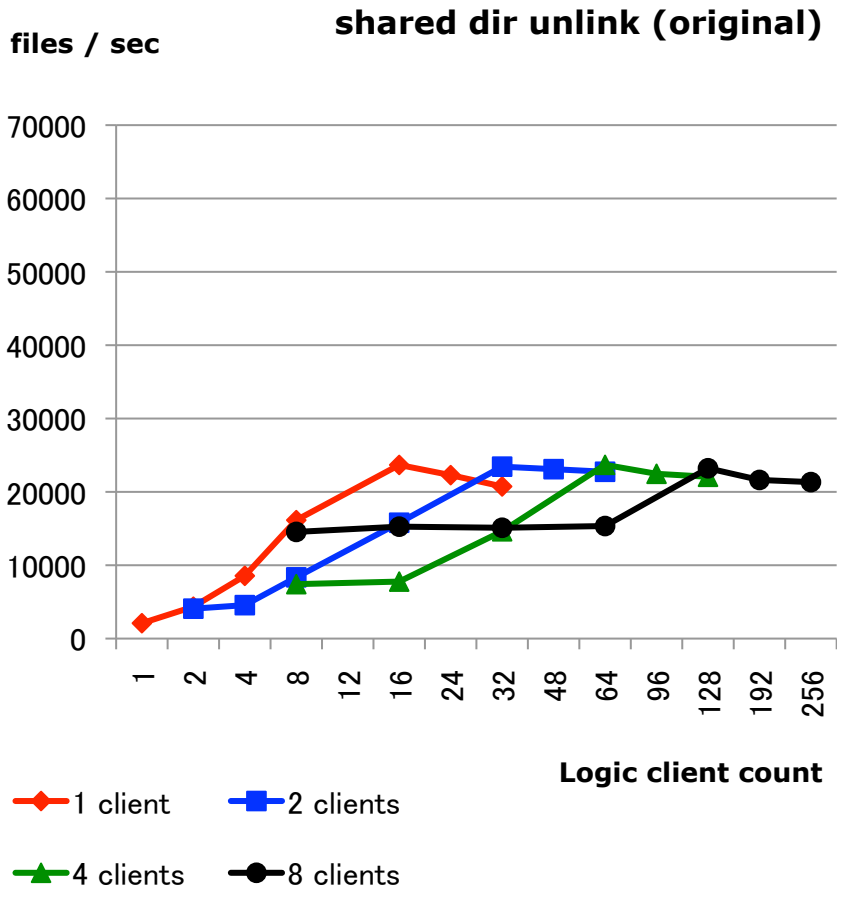
Opencreate under shared dir





Unlink under shared dir





Thank You

- Fan Yong
  Senior Engineer
  Whamcloud, Inc.
  yong.fan@whamcloud.com
- Liang Zhen
  Senior Engineer
  Whamcloud, Inc.
  liang@whamcloud.com