Whamcloud

Lustre Client Metadata Writeback Caching: Design and Implementation

Yingjin Qian, Li Xi, Liu Ying

DDN® STORAGE

Outline

Whamcloud

- Why Lustre client **W**rite**b**ack **C**aching(WBC)
  - Better metadata performance
- Design and implementation
  - MemFS, caching mechanism, …
- Performance evaluation
- Conclusion and future work

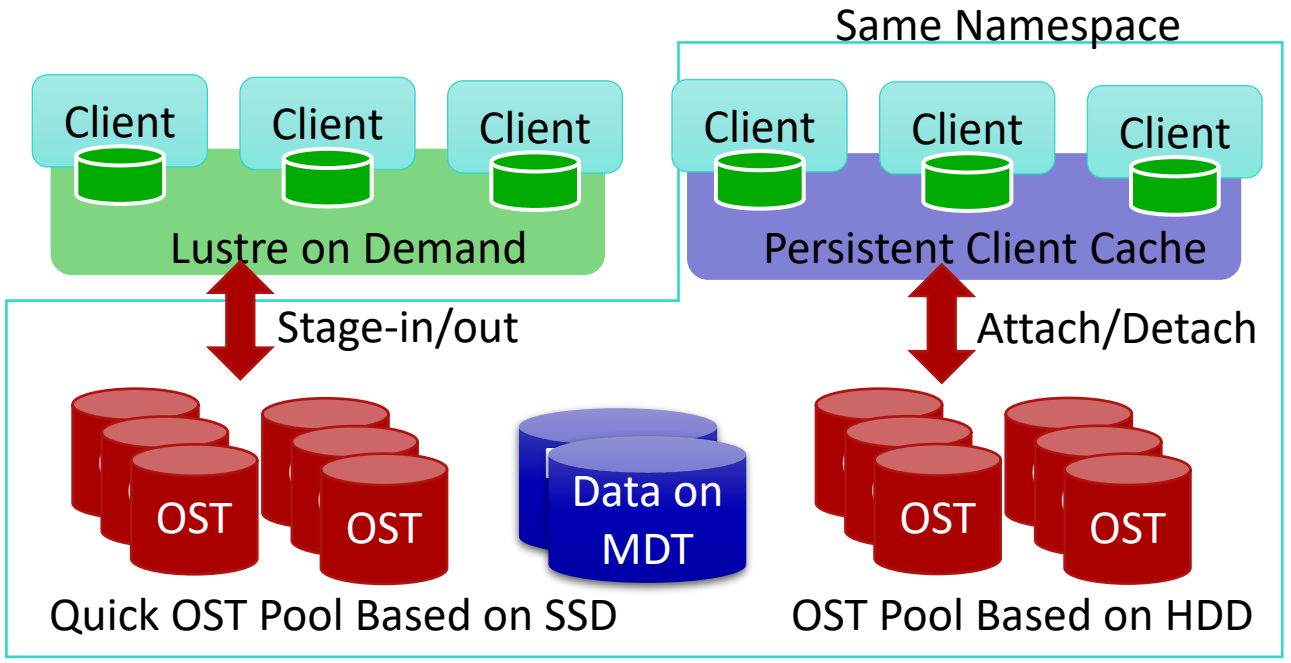
Whamcloud

Why Lustre client Writeback Caching ?

Current Data Cache/Acceleration Inside Lustre

Whamcloud

- Persistent Client Caching
- Data on MDT
- OST pool
- Lustre on Demand
- Data reads/writes are fully cached



Better metadata write performance

Whamcloud

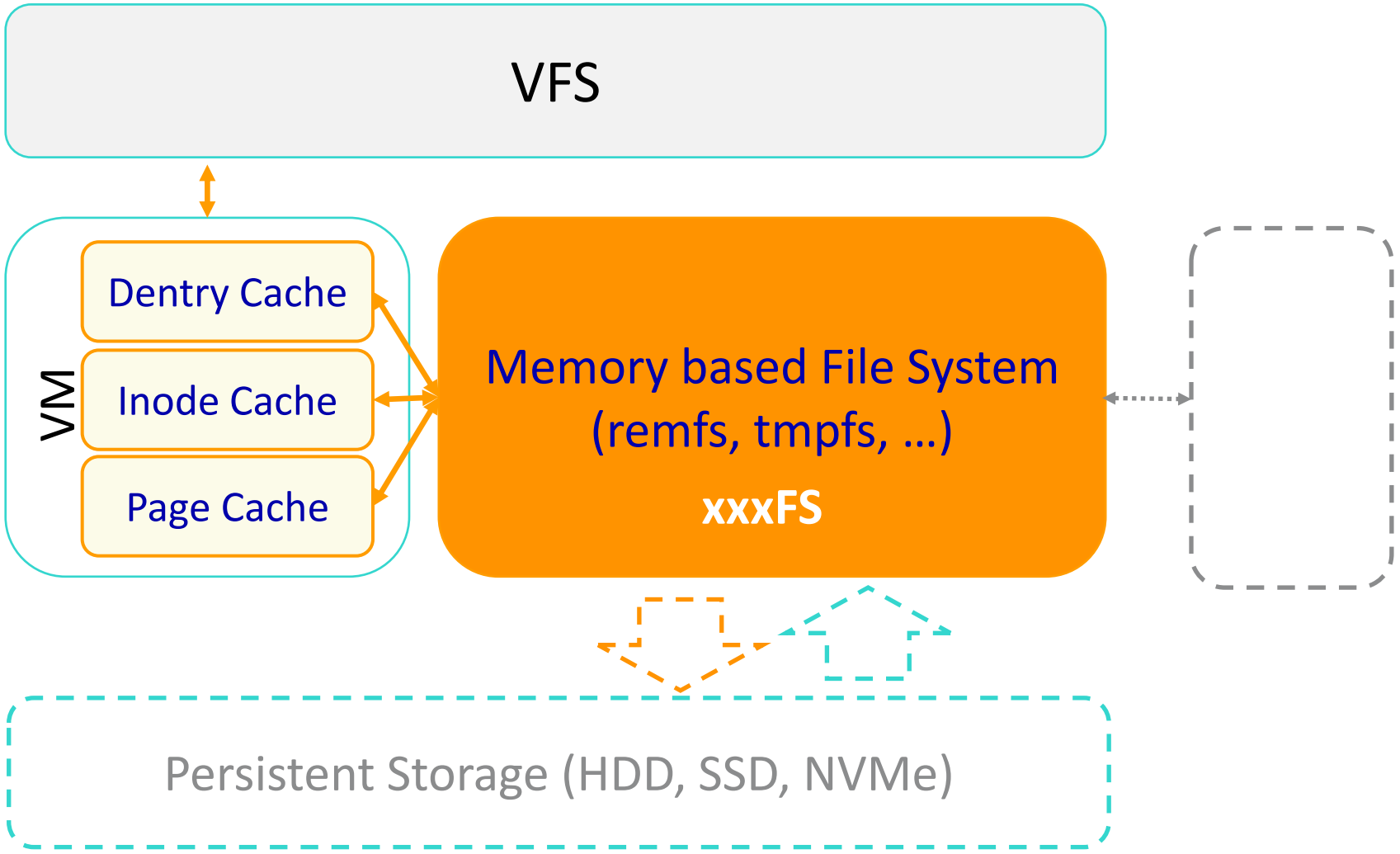
Design and Implementation

Main Targets of Lustre WBC

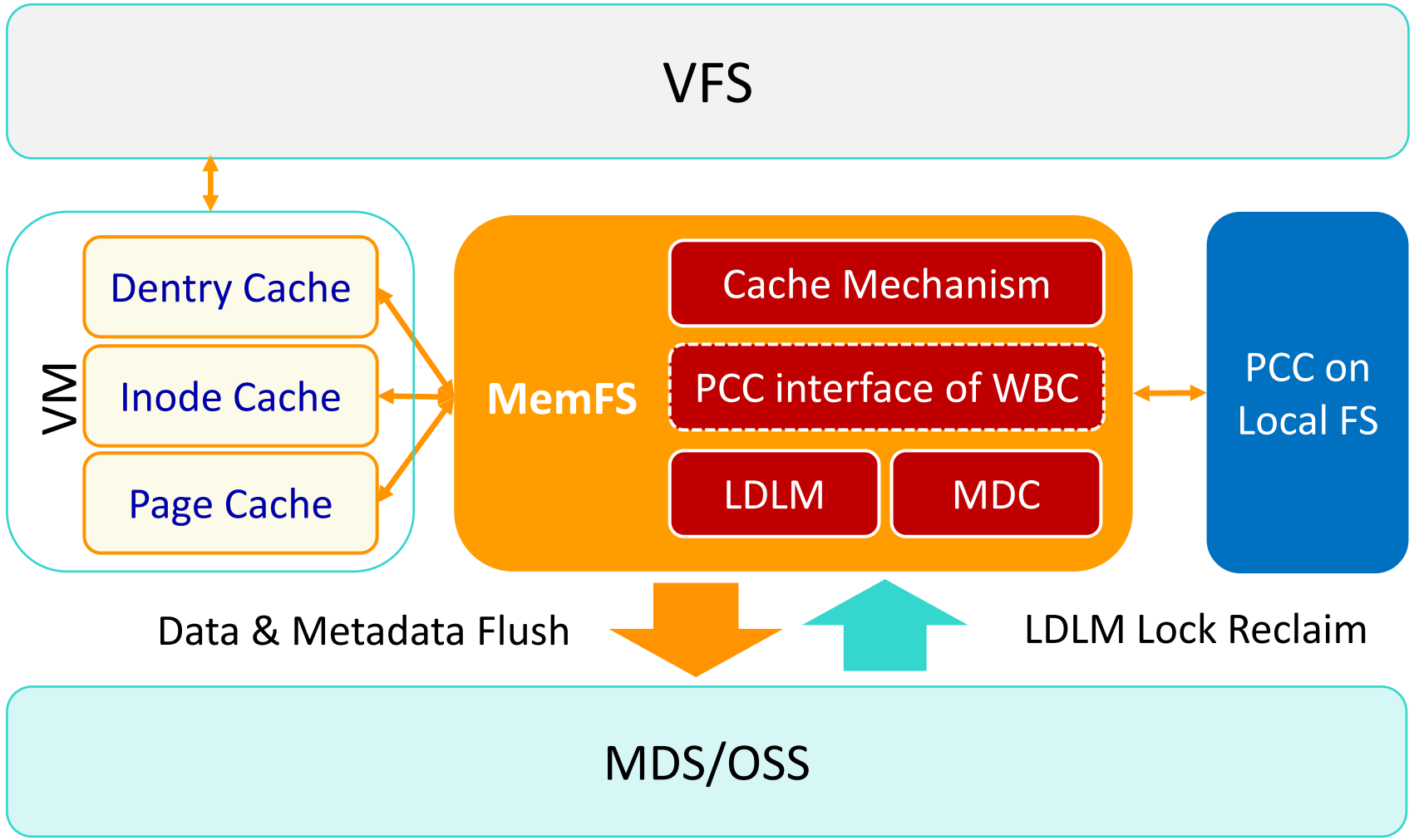
Whamcloud

- Better metadata write performance
  - How to store and access data on lustre client in high speed
  - How to keep data consistency
    - When to flush
    - How to flush
  - How to keep data persistent
- Generalization

General VFS


VFS
VM
Dentry Cache
Inode Cache
Page Cache
Memory based File System
(remfs, tmpfs, …)
xxxFS
Persistent Storage (HDD, SSD, NVMe)

Lustre WBC

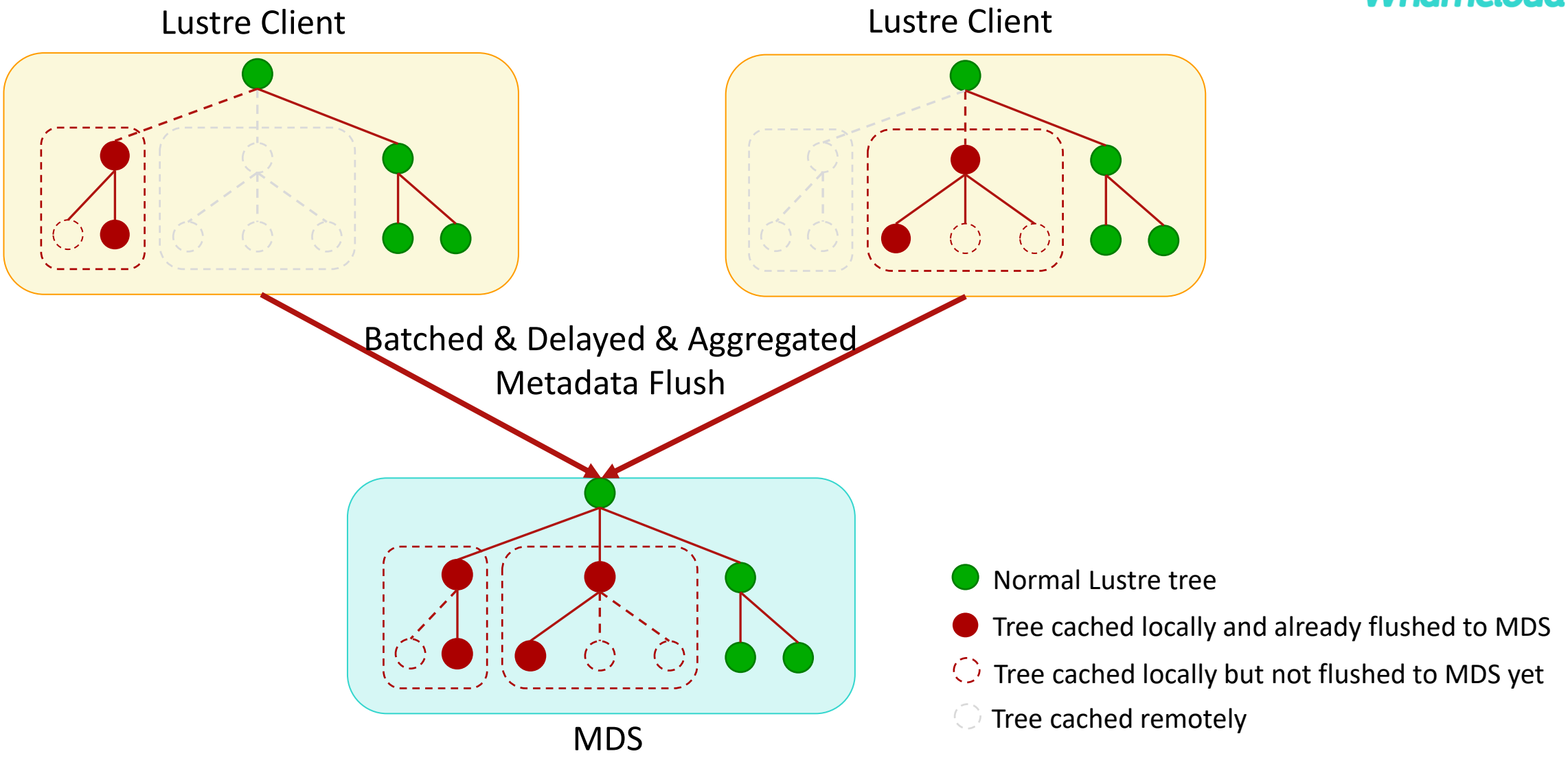

VFS
VM
Dentry Cache
Inode Cache
Page Cache
MemFS
Cache Mechanism
PCC interface of WBC
LDLM
MDC
PCC on Local FS
Data & Metadata Flush
LDLM Lock Reclaim
MDS/OSS

Design of Lustre WBC

Whamcloud

- MemFS
  - a POSIX-compatible memory file system, like tmpfs
  - Support persistent storage
- Exclusive LDLM lock to avoid access conflict
- Caching Mechanism
  - Time: Aging
  - Space: Memory size/inode limitation
  - Caching state machine

General Idea of Lustre WBC



State Flags of Cached Files/Directories in WBC

Whamcloud

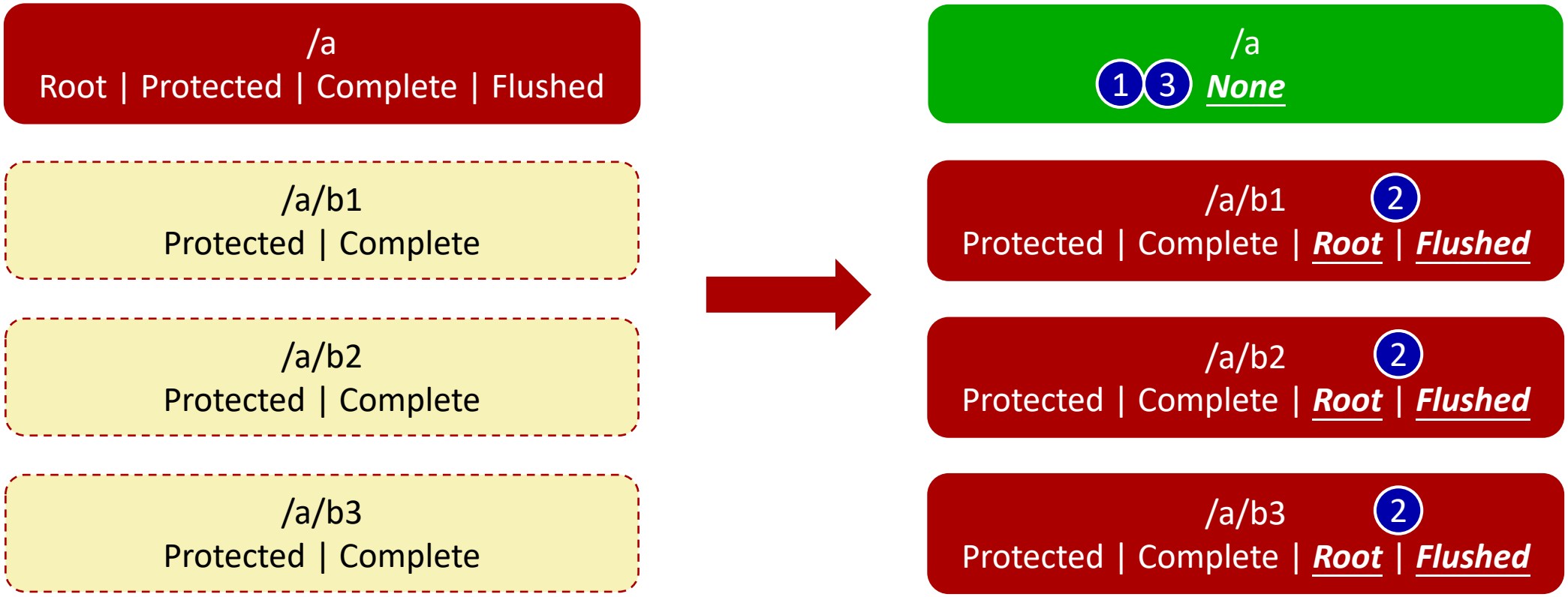
- **Root**: Root of the cached directory tree
- **Protected**: The directory/file is protected by an EX LDLM lock
- **Complete**: The children under this directory are fully cached in MemFS
- **Flushed**: Metadata has been flushed from MemFS to MDS
- **Assimilated:** Page cache of the file has been assimilated from MemFS to Lustre OST
- **None**: None of the above flags is set for normal Lustre files

Operations to Change WBC States

Whamcloud

- **Purge**: purge the **R**oot from the WBC
- **Flush:** flush the directory and file from WBC to MDS
- **Assimilate:** assimilate the data from WBC to Lustre OST

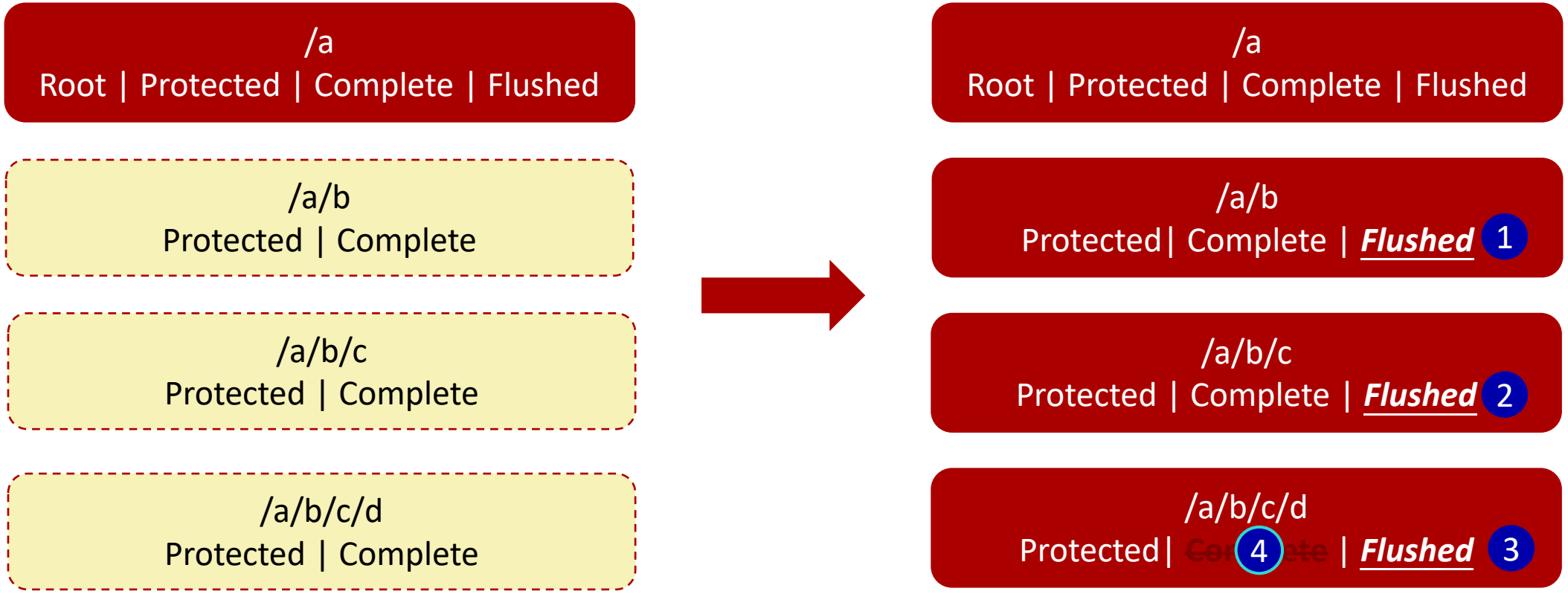
State Transition when Purging the Root



Remote access conflict of /a → Purge /a

State Transition when Flushing a Directory

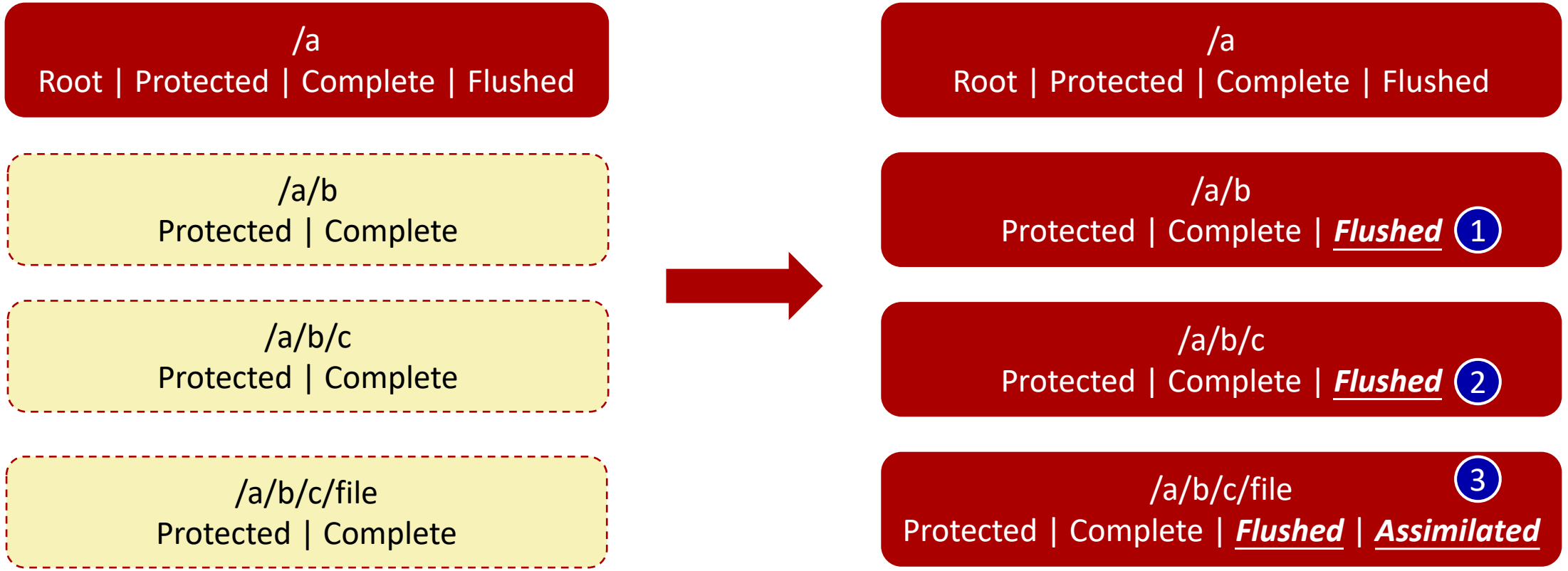
Whamcloud



/a/b/c/d/e

Reach inode limit when creating /a/b/c/d/e on MemFS → Flush b/c/d

State Transition when Assimilating File Data



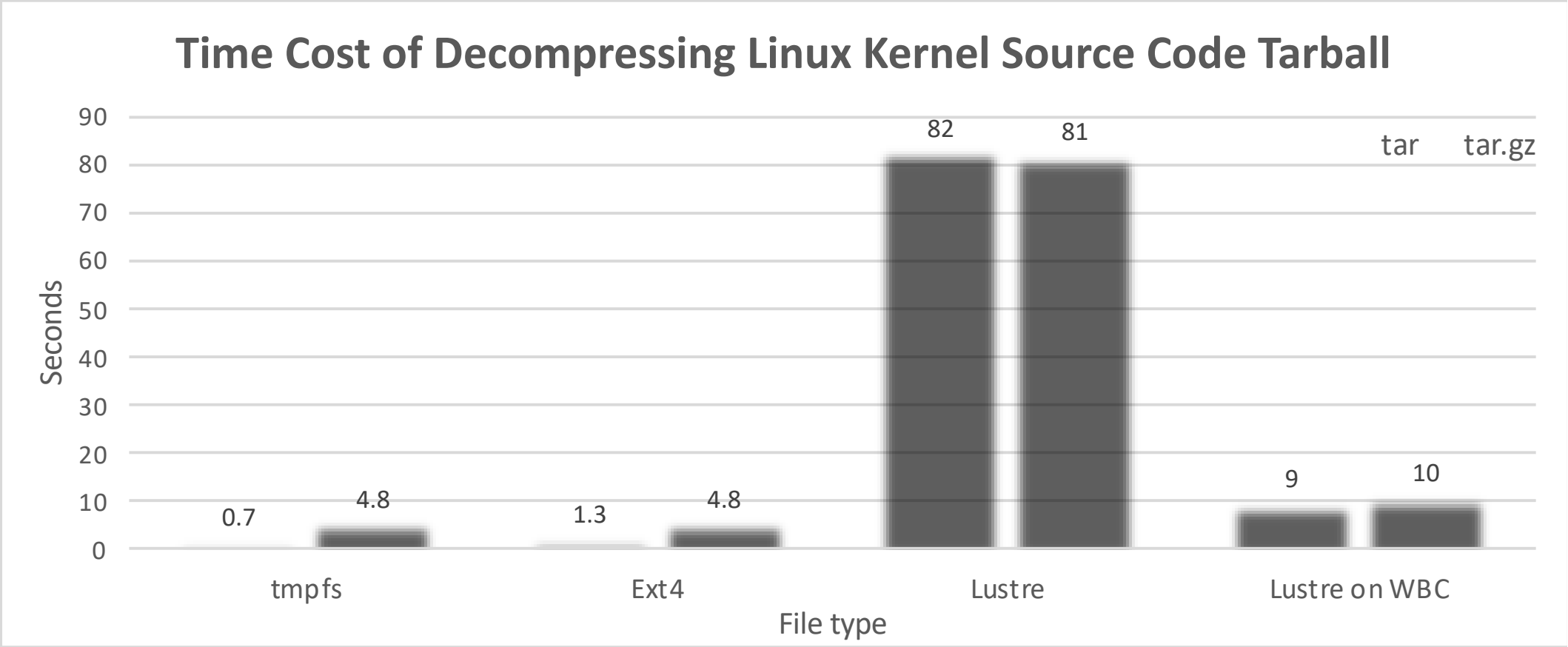
Reach memory size limit when writing /a/b/c/file → Assimilate Data of /a/b/c/file

Whamcloud

Performance Evaluation

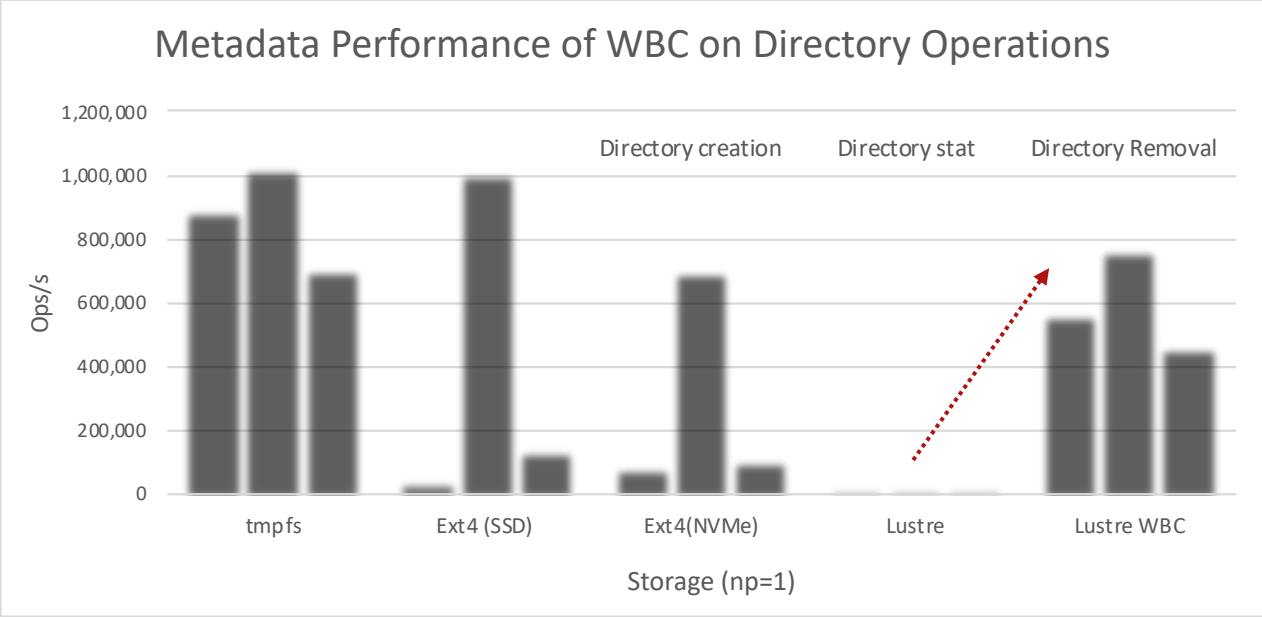
Untar Performance

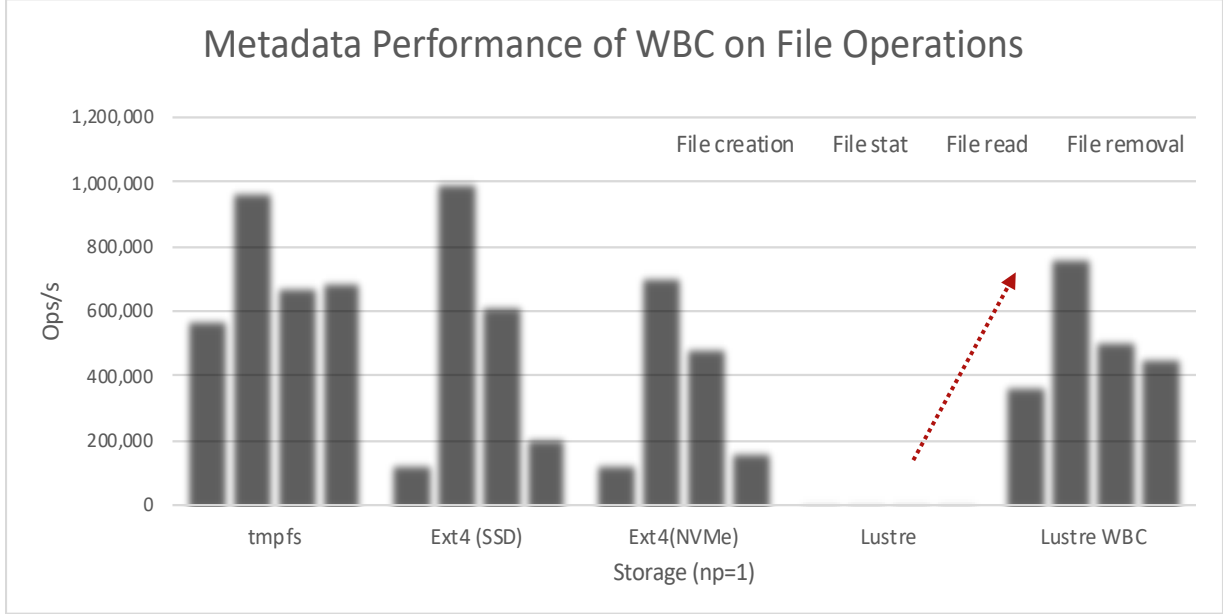
Lustre: DDN AI400X Appliance (20 X SAMSUNG 3.84TB NVMe, 4X IB-HDR100)
Lustre client: Intel Gold 5218 processor, 96 GB DDR4 RAM, CentOS 8.1
Local File System on SSD: Intel SSDSC2KB240G8



Metadata Performance

Lustre: DDN AI400X Appliance (20 X SAMSUNG 3.84TB NVMe, 4X IB-HDR100)
Lustre client: Intel Gold 5218 processor, 96 GB DDR4 RAM, CentOS 8.1
Local File System on SSD: Intel SSDSC2KB240G8
Benchmark: mdtest





x about 70 ~ 100 times

Whamcloud

Conclusion and Future Work

Conclusion

Whamcloud

- Lustre WBC for better metadata write performance
  - Store and access data in high speed by a new defined MemFS
  - Keep data consistency by
    - Exclusive LDLM lock
    - Caching Mechanism
    - Caching with complete state machine
  - Keep data persistent by persistent storage
  - Generalization

Future work

Whamcloud

- WBC with PCC
- WBC with DNE
- WBC with PMEM
- Possible offline/disconnected operations on Lustre client
  - Mobile computing?

Whamcloud

Thank you!

DDN® STORAGE