Lustre Persistent Cache on Client: A client side cache that speeds up applications with certain I/O patterns

Li Xi

DDN Storage

NSCC-Wuxi and the Sunway Machine Family

Sunway-I:
- CMA service, 1998
- commercial chip
- 0.384 Tflops
- 48th of TOP500

Sunway BlueLight:
- NSCC-Jinan, 2011
- 16-core processor
- 1 Pflops
- 14th of TOP500

Sunway TaihuLight:
- NSCC-Wuxi, 2016
- 260-core processor
- 125 Pflops
- 1st of TOP500

PCC project is collaborated by NSCC-Wuxi and DDN

Sunway TaihuLight in NSCC-Wuxi: a 10M-Core System

163,840 processes — 65 threads

racks *chips* *core-groups* *cores* *total number of cores*

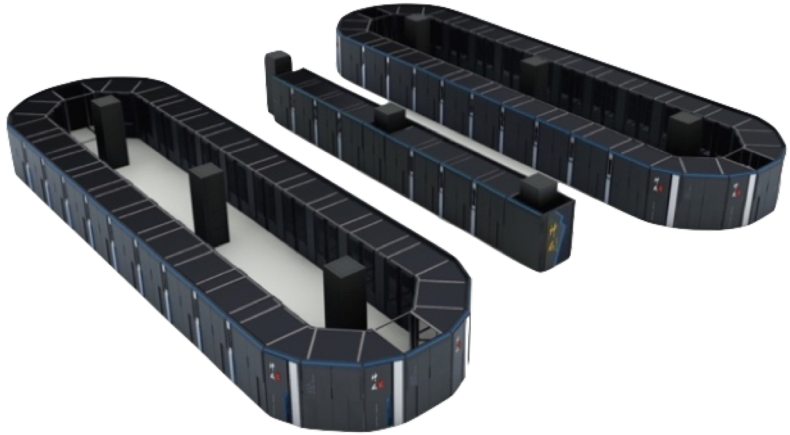
$$40 \times 1{,}024 \times 4 \times 65 = 10{,}649{,}600$$

260-core Chip

1,024x

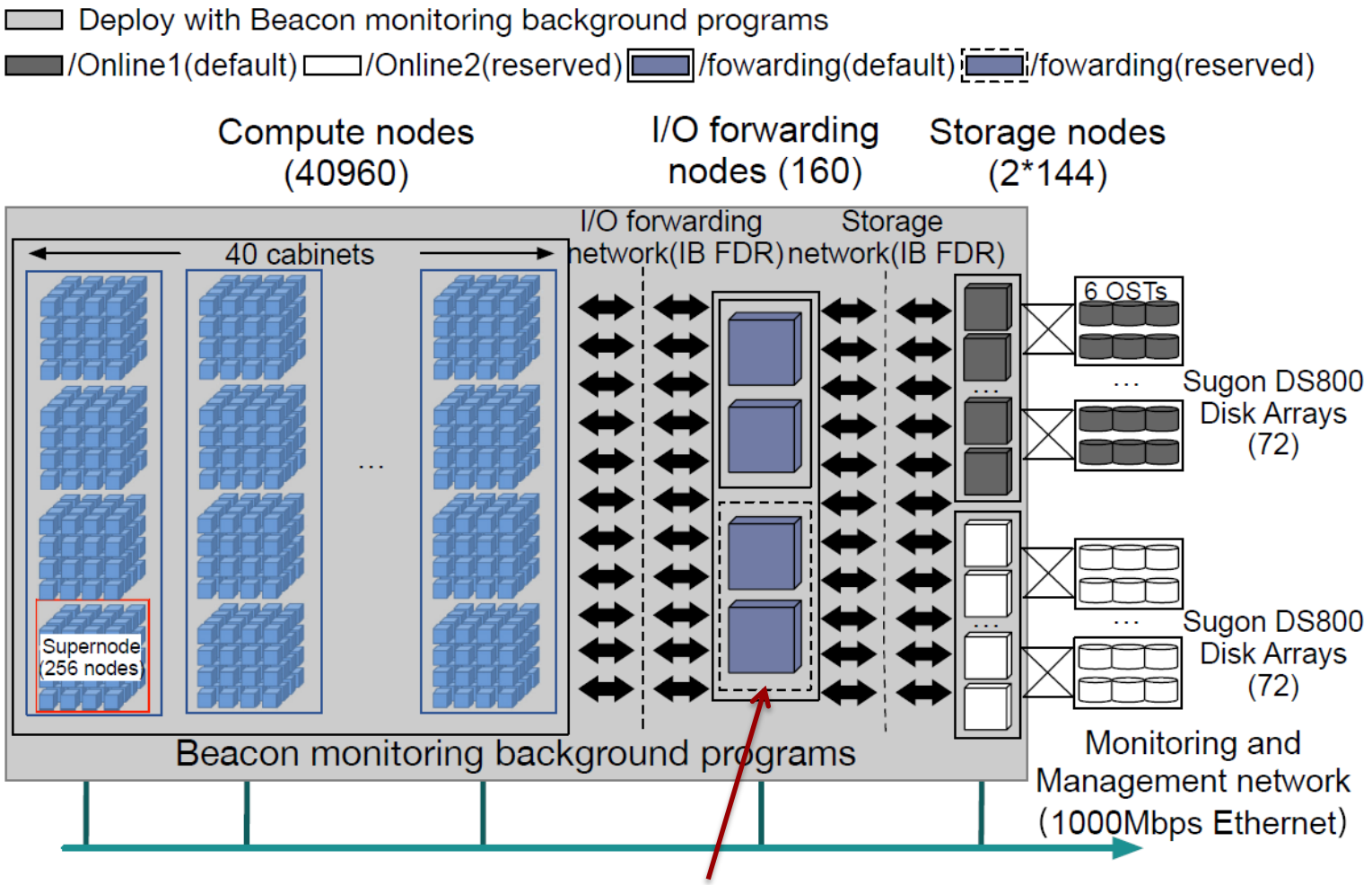
Rack

40 x



System

I/O Architecture of Sunway TaihuLight



Cache on I/O forwarding nodes (Lustre clients) should be helpful

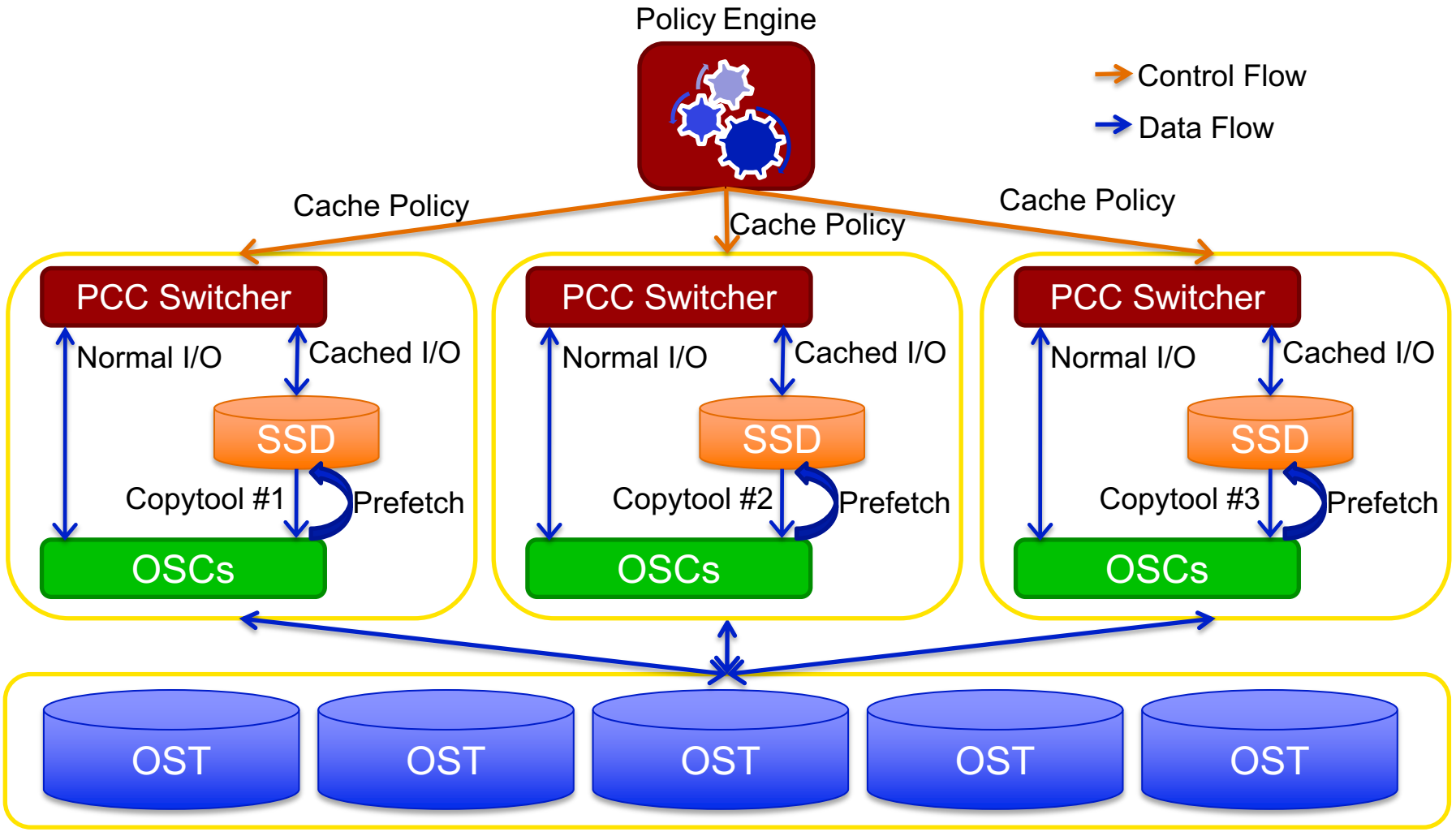
Why SSD cache on Lustre client?

- **Less overhead visible for applications**
  - No network latency
  - No LDLM lock and other Lustre overhead
- **Easier to be optimized for the best performance**
  - I/O stack is much simpler
  - No interference I/Os from other clients
- **Relatively easier than server side implementations**
  - Write support for SSD cache on server side is very difficult
  - Problems for write cache on server side:
    - Visibility when failover happens
    - Consistency when corruption happens
- **Less requirement on hardware**
  - Any kind of SSD can be used as the cache device
- **Reduces the pressure of OSTs**
  - Small or random I/Os are regularized to big sequential I/Os
  - Temporary files do not need to be flushed to OSTs

Design of PCC

- **PCC provides a group of local caches**
  - Each client has its own local cache based on SSD/NVMe
  - No global namespace is provided by PCC
  - Local file system is used to manage the data on local caches
  - Cached I/O is directed to local file system while normal I/O is directed to OSTs
- **Two modes**
  - RW-PCC (LU-10092) keeps readwrite cache on local SSD/NVMe
  - of a single client
  - RO-PCC (LU-10499) keeps readonly cache on local SSDs/NVMe of multiple clients

Architecture of PCC



Design of RW-PCC (1)

- **RW-PCC uses HSM mechanism for data synchronization**
  - RW-PCC uses HSM copytool to restore file from local caches to Lustre OSTs
  - Remote access from another Lustre client will trigger the data synchronization
  - Each RW-PCC has a copytool instance running with unique archive number
  - If a client with PCC goes offline, the cached data becomes inaccessible for other client temporally
    - But this is fine, since it is “local ” cache
- **A policy is used to determine whether to cache a newly created file on local PCC**
  - RW-PCC currently check project ID of the parent directory
  - In the future rules based on UID/GID/project ID can be defined

Design of RW-PCC (2)

- **When file is being created on RW-PCC**
  - A normal file is created on MDT
  - An empty mirror file is created on local cache
  - The HSM status of the Lustre file will be set to archived and released
  - The archive number will be set to the proper value
- **When file is being fetched to RW-PCC**
  - An mirror file is copied to local cache
  - The HSM status of the Lustre file will be set to archived and released
  - The archive number will be set to the proper value
- **When file is being accessed from RW-PCC**
  - Data will be read/written directly from/to local cache
  - Metadata be read from MDT, except the file size
  - File size will be got from local cache

Design of RO-PCC

- **RO-PCC uses LDLM lock to protect file data from being modified**
  - Grouplock is currently used for protecting other clients from writing the file (unfortunately reading is prohibited too)
  - A new kind of LDLM lock (PCCRO) is being added, so that other clients can read the file normally
- **No data write is permitted to the RO-PCC cached file**
- **Multiple copies of the cache can be kept on multiple clients**
- **When file is being fetched to RO-PCC**
  - An mirror file is copied to local cache
  - A grouplock (or PCCRO lock) of the file will be hold by this client
- **When file is being read from RO-PCC**
  - Data will be read directly from local cache
  - Metadata will be read from MDT, except file size
  - File size will be got from local cache

Interfaces

- **Add a PCC storage to the client**
  - # echo -n 'add $PCC_ROOT $ID $PCC_PROJID' > /proc/fs/lustre/llite/$CLIENT/pcc
- **Delete a PCC storage from the client**
  - # echo -n ‘del $PCC_ROOT' > /proc/fs/lustre/llite/$CLIENT/pcc
- **List the existing PCC storages**
  - # cat /proc/fs/lustre/llite/$CLIENT/pcc
- **Fetch a file to RW-PCC**
  - # lfs pcc_fetch -a $ID $FPATH
- **Fetch a file to RO-PCC**
  - # lfs pcc_fetch -r -a $ID $FPATH
- **Evict a file from RO-PCC**
  - # lfs pcc_detach $FPATH
- **Check the PCC status of a file**
  - # lfs pcc_state $FPATH

Data management of PCC

- **Possible conditions to fetch a file:**
  - Suitable I/O patterns are detected for the application
  - High access heat (LU-10602) is being detected on that file
  - The file is going to be accessed soon (e.g. job is starting)
  - Explicit command from applications/users (lfs pcc_fetch)
- **Possible conditions to shrink a file from the cache:**
  - Cache is becoming full
  - The file size is growing too big to be cached
  - Low access heat is detected on the file in the cache
  - The file won't be accessed any more for some time (e.g. job is stopping)
  - Explicit command from applications/users (lfs pcc_detach)
- **Data movement between local caches and OSTs can be managed by policy engine**

What kind of I/O can be accelerated?

- **RW-PCC: The file should be read/written only from a single client**
  - But no inconsistency will happen even the application writes the cached file on a remote client
  - File creation performance on RW-PCC is slightly slower
    - Overhead of file creation on local file system
- **RO-PCC: Access to the cached files should be entirely readonly**
  - Write operation will get failure or be blocked
- **Stat performance of cached files will be accelerated**
- **Random & small I/O will be accelerated a lot**
  - Readahead doesn't help much in this circumstance
  - No LDLM overhead
  - No RPC overhead

I/O Pattern Detector based on Changelog

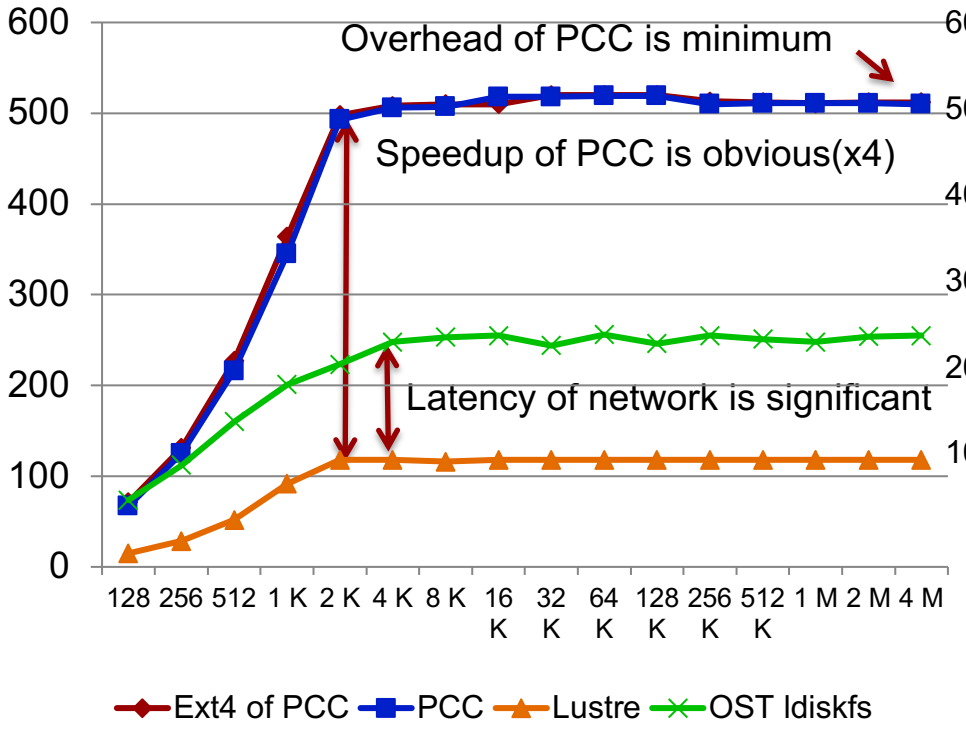
- **I/O pattern detector is needed to find proper application for PCC**
- **Lustre Audit (LU-9727) extends the capability of Changlog**
  - Client NID is added into the record
  - OPEN/CLOSE events can be recorded
- **I/O pattern detector can be implemented based on extended Changlog**
  - Analyze I/O patterns according the NIDs and mode of the OPEN event
  - If a file is only opened on a single NID, it might be a good candidate to be cached on PCC as readwrite mode
  - If a file is only opened with readonly mode by multiple NIDs, it might be a good candidate to be cached on PCC as readonly mode.

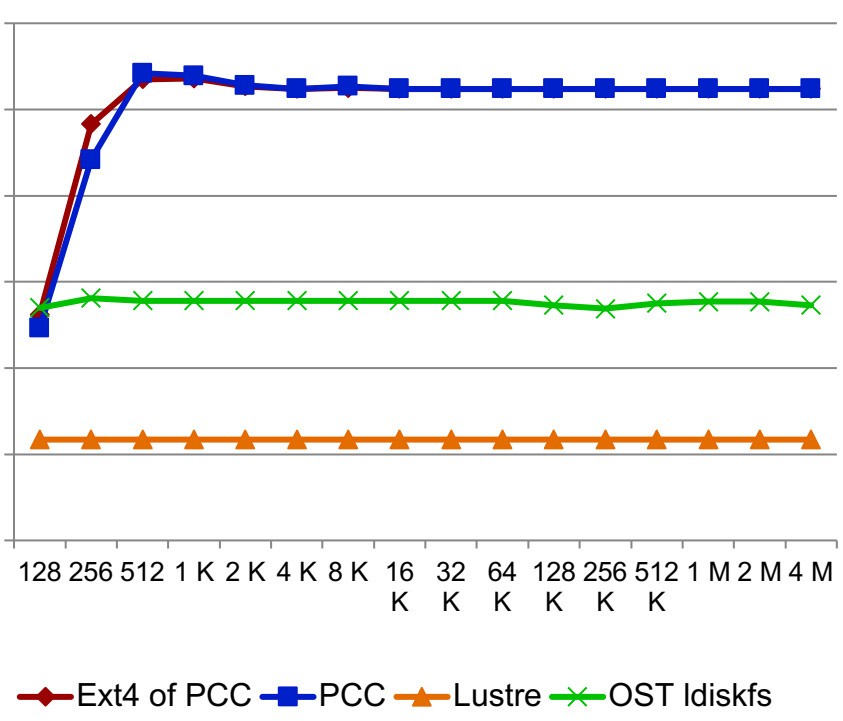
Limitations

- **Capacity of each local cache is limited**
  - Size of a cached file is limited to the available space of the local cache
  - The total cached data on a single client is limited
- **Files can not be partly cached**
  - Partial cache can be implemented if HSM supports partial archive/restore
- **The total PCC clients are limited to 32 (LU-10114)**
  - Only 32 different archive numbers are supported by Lustre
  - This upper limitation can be raised in the future

RW-PCC test: sequential I/O

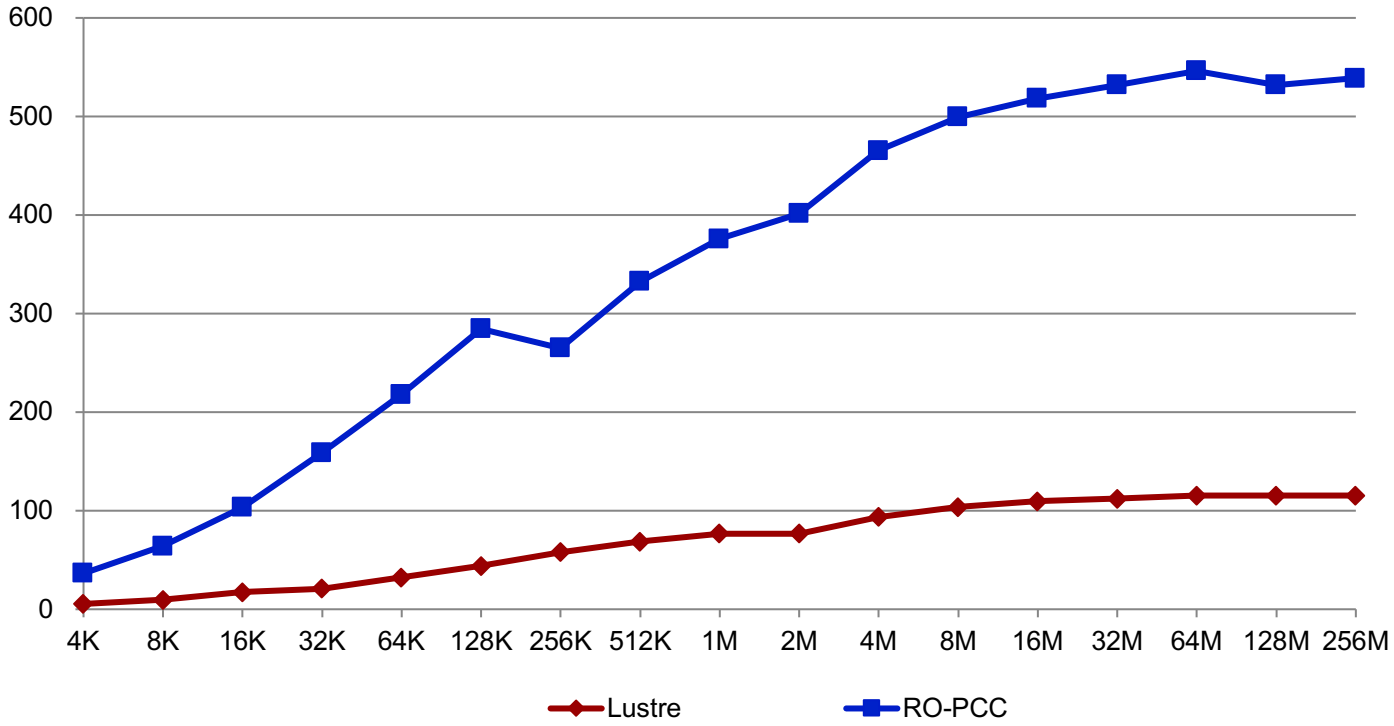
- PCC uses Ext4 (Samsung SSD 850 EVO 500GB) as local cache
- Lustre OST is based on a single SSD (Intel 535 Series)
- Network is Gigabit Ethernet
- Benchmark: use dd command to write/read 32GB data with different I/O sizes
- Run the same command on different levels of the storage





RO-PCC test: random I/O

- PCC uses Ext4 (Samsung SSD 850 EVO 500GB) as local cache
- Lustre OST is based on a single SSD (Samsung SSD 850 EVO 500GB)
- Network is Gigabit Ethernet
- Benchmark: read 40GB data at random offset with different I/O sizes



Summary

- We designed and implemented a novel persistent client side cache (PCC) for Sunway TaihuLight
- Small scale benchmarks shows that PCC is able to accelerate I/Os
- Large scale benchmarks and tests will be carried out soon
- Patches have been pushed to the community for review (LU-10092, LU-10499, LU-10602)
- Feature aimed at Lustre-2.13

Thank you!