LCOC
Lustre Cache on Client

Xue Wei
The National Supercomputing Center, Wuxi, China

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

LCOC project is collaborated by NSCC-Wuxi and DDN

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

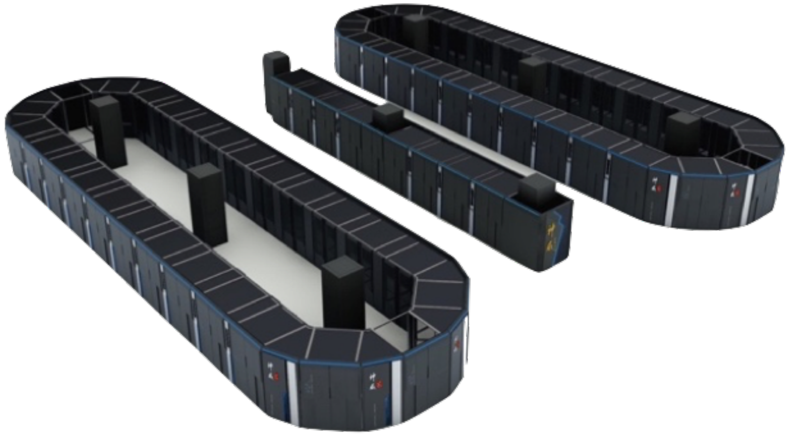
$$\overbrace{\underset{\text{racks}}{40} \times \underset{\text{chips}}{1{,}024} \times \underset{\text{core-groups}}{4}}^{\text{163,840 processes}} \times \overbrace{\underset{\text{cores}}{65}}^{\text{65 threads}} = \underset{\text{total number of cores}}{10{,}649{,}600}$$

260-core Chip

1, 024x

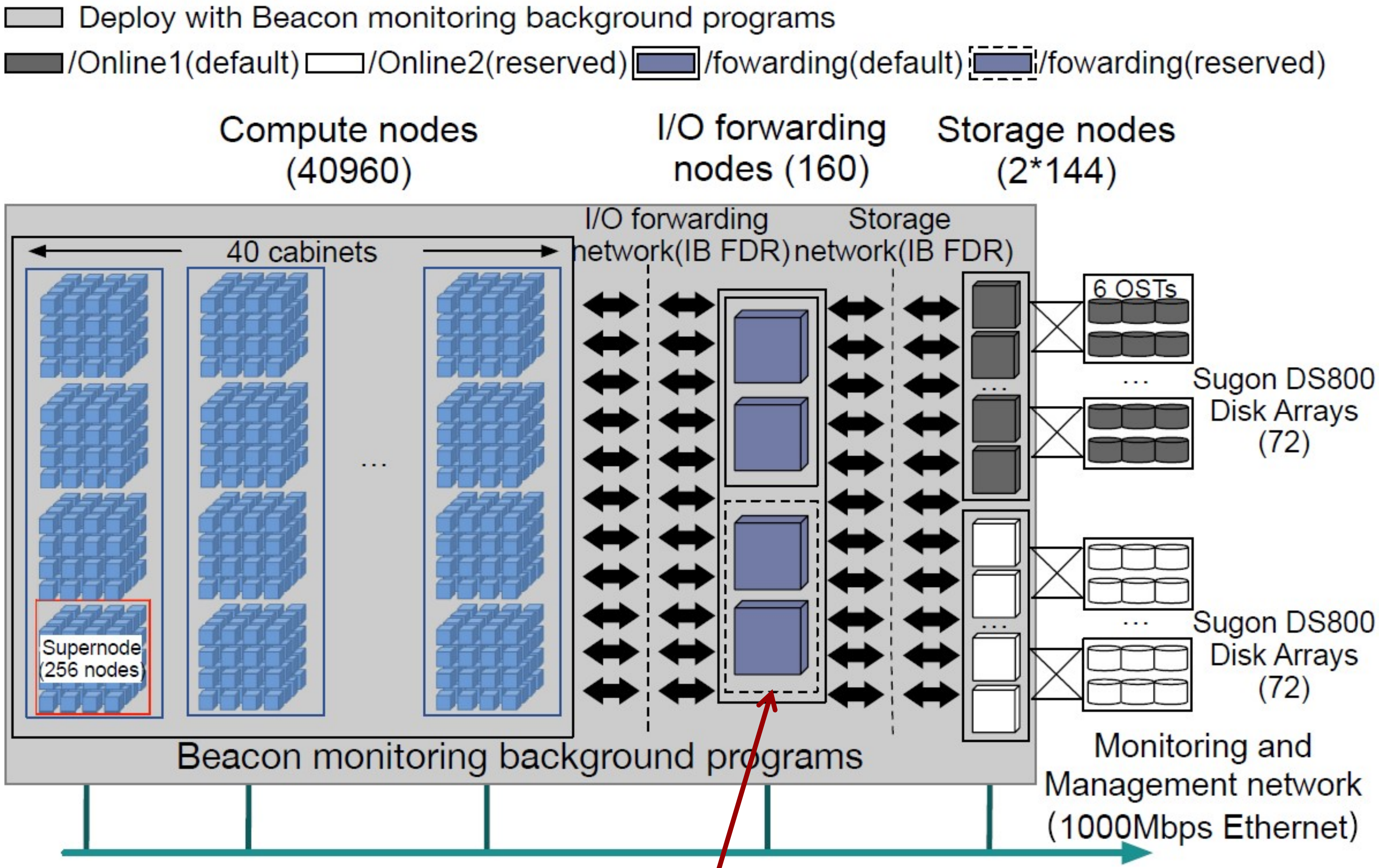
Rack

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**
  - Less 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
- **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
- **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

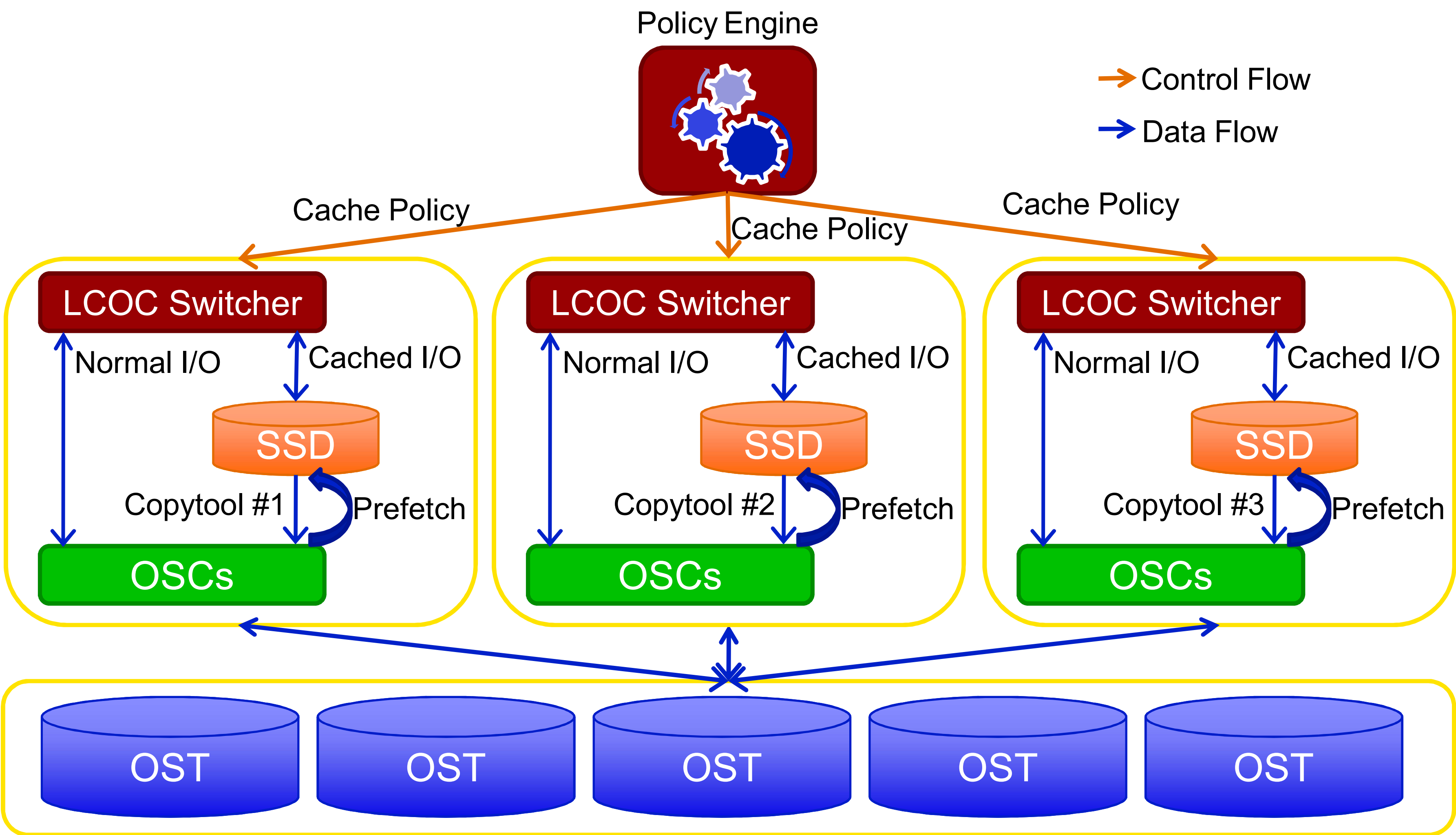
Design of LCOC (1)

- **LCOC provides a group of local caches**
  - Each client has its own local cache based on SSD
  - No global namespace is provided by LCOC
  - Data on the local cache can not accessed by other clients directly
  - 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
- **LCOC uses HSM for data synchronization**
  - LCOC uses HSM copytool restore file from local caches to Lustre OSTs
  - Remote access from another Lustre client will trigger the data synchronization
  - Each LCOC has a copytool instance running with unique archive number
  - If a client with LCOC goes offline, the cached data becomes inaccessible for other client temporally
    - But this is fine, since it is “local ” cache

Design of LCOC (2)

- **When file is being created on LCOC**
  - 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 prefetched to LCOC**
  - 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 LCOC**
  - 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

Architecture of LCOC



Data management of LCOC

- **Policy engine manages the data movement from local caches to OSTs**
  - Policy engine will prefetch data if necessary
  - Possible conditions to prefetch a file:
    - High access heat is being detected on that file
    - The file is going to be accessed soon (e.g. job is starting)
    - Explicit hint is being given by applications/users (e.g. lfs ladvise)
  - Policy engine will do HSM restore to flush data according to the policies defined
  - 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 hint is being given by applications/users (e.g. lfs ladvise)

Limitations

- **Not all applications are able to be accelerated by LCOC**
  - Locality requirements of application I/Os
    - Applications shall not access the cached file through multiple clients
    - But no inconsistency will happen even the application writes the cached file on a remote client
  - 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 LCOC clients are limited to 32**
  - Only 32 different archive numbers are supported by Lustre
  - This upper limitation can be raised in the future

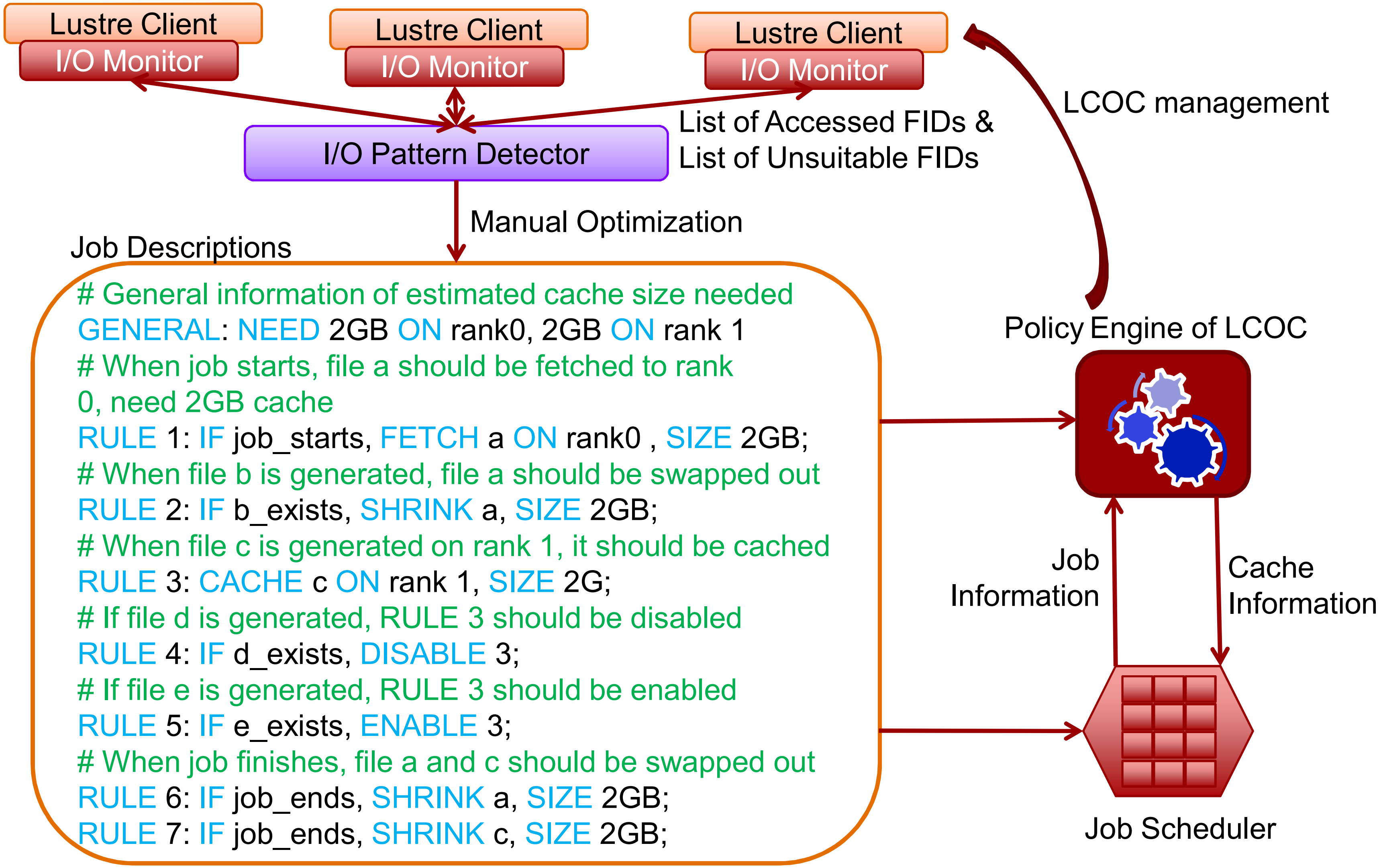
Extension: Read-only replications

- **Read-only replications are cached on multiple local caches**
  - The replications on LCOC are identical to the data on OSTs
  - A new global flag “lcoc_cached” is used to indicate whether any local replication exists for a file
  - Replications of files without “lcoc_cached” flag will be cleared
- **I/O on client with LCOC replication:**
  - Read:
    - The file data comes from cache if “lcoc_cached” is set
    - The file data comes from OSTs if “lcoc_cached” is cleared
  - Write:
    - Modification is applied directly to data on OSTs
    - The “lcoc_cached” flag is cleared
- **I/O on client without LCOC replication:**
  - Read:
    - Data are read from OSTs directly
  - Write:
    - The “lcoc_cached” flag is cleared

I/O Pattern Detector and Job Scheduler for LCOC

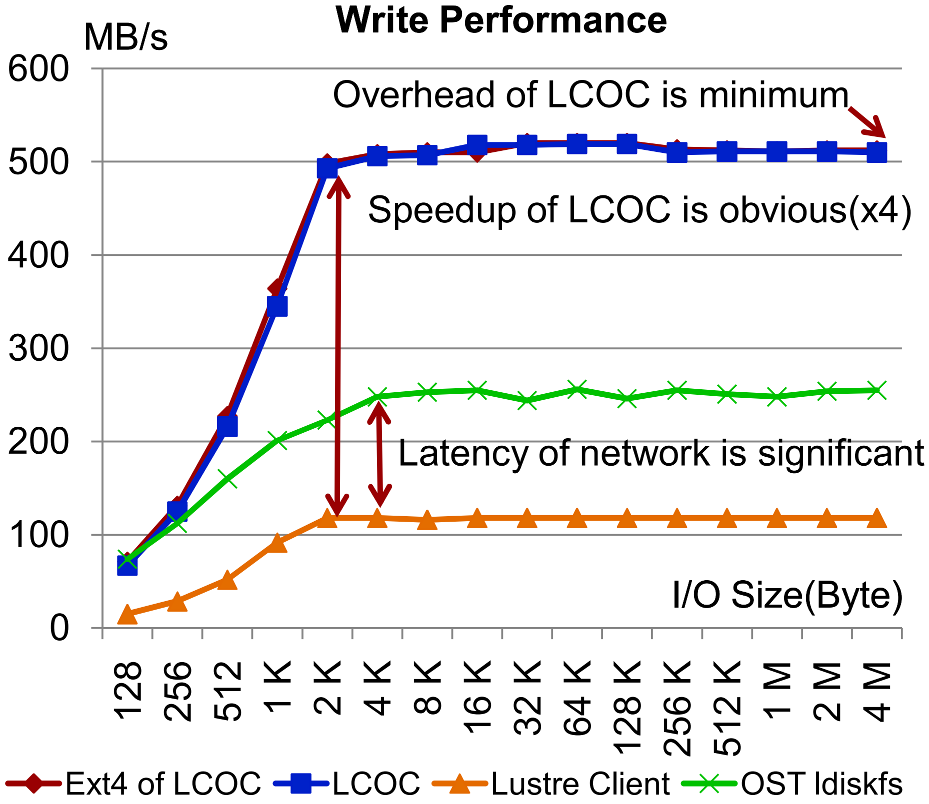
- **I/O pattern detector detects suitable applications for LCOC**
  - Jobstat ID is used to distinguish I/O from different jobs
  - The type, timestamp, size, offset, FID, job ID of I/Os are recorded on each client and sent to global detector
  - The global detector finds FIDs with cross-client I/O and send back to I/O monitors on all clients
  - A description about the I/O patterns on each job is generated by the detector
- **LCOC-ware scheduler**
  - The scheduler considers LCOC usage as part of the constraint when scheduling jobs
    - Concurrent jobs shall not cause contention or exhaustion of LCOC
  - The scheduler gives hints for LCOC cache management
    - Which files should be prefetched to cache
    - Whether a newly created file should be cached or not
    - Which client should cache the file
    - When should a file be swapped out of the cache

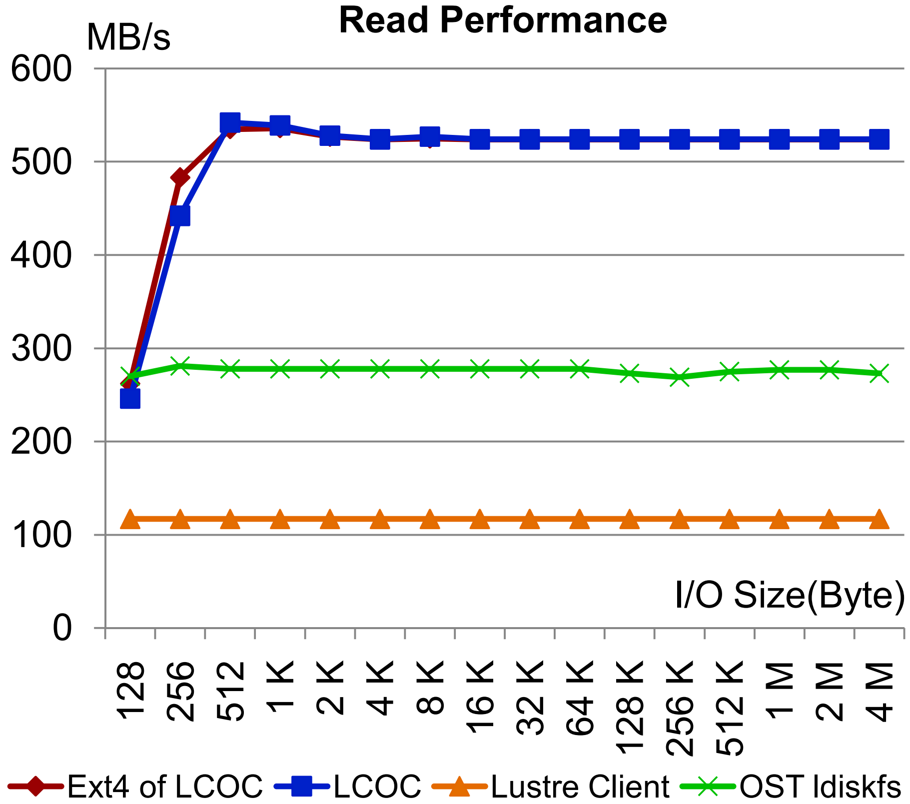
I/O Pattern Detector and Job Scheduler for LCOC



Benchmark results

- LCOC 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





Summary

- We designed and implemented a novel client side cache (LCOC) for Sunway TaihuLight
- Small scale benchmarks shows that LCOC is able to accelerate I/Os
- Large scale benchmarks and tests will be carried out in NSCC-Wuxi soon

Thank you!