HiSMRfs – A File System for Shingled Storage Array

Presenter: WeiYa XI

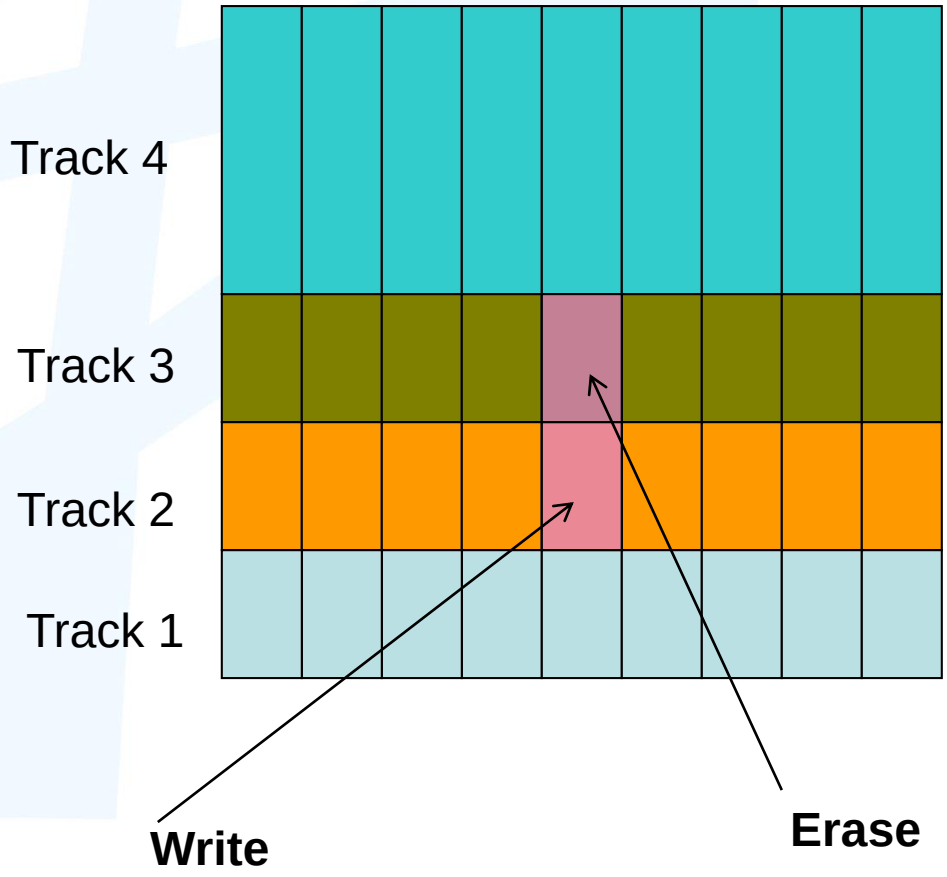
Chao JIN, WeiYa XI, Zhi Yong CHING, Feng HUO, Chun Teck LIM
Data Storage Institute
Agency of Science, Technology and Research (A*STAR)
Republic of Singapore

MSST 2014
June 2 ~ 6 2014

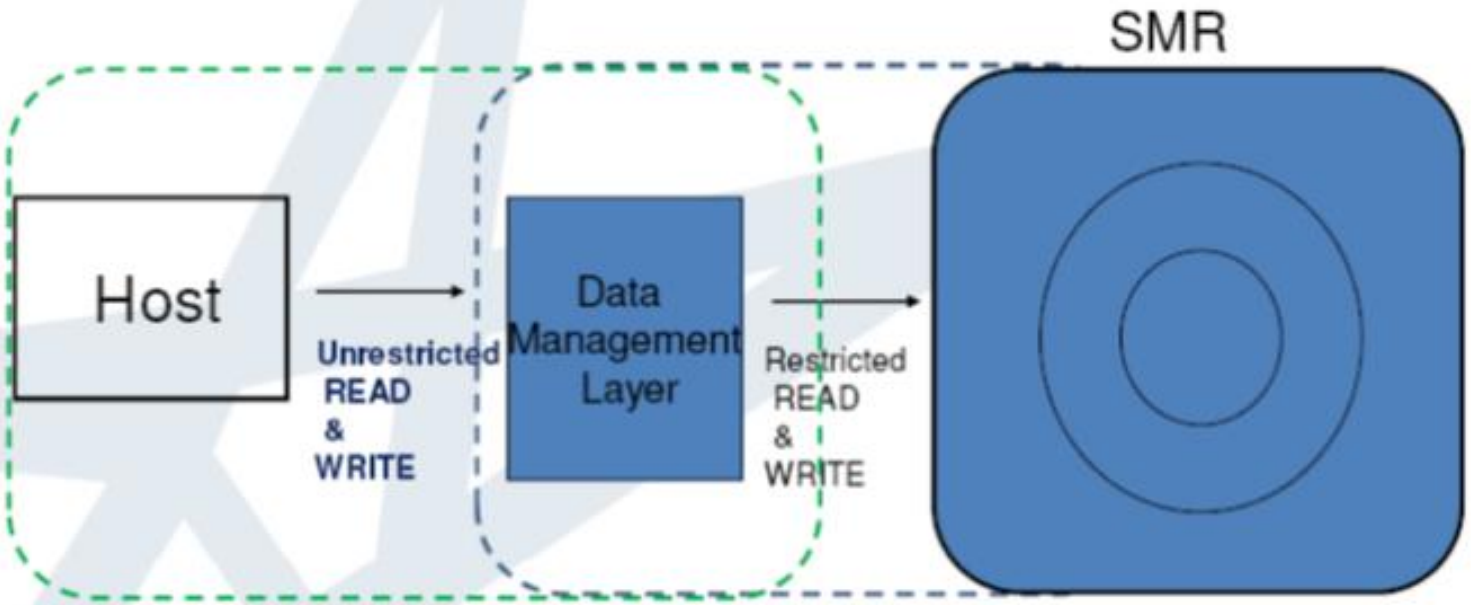
A*STAR DSI

Problem

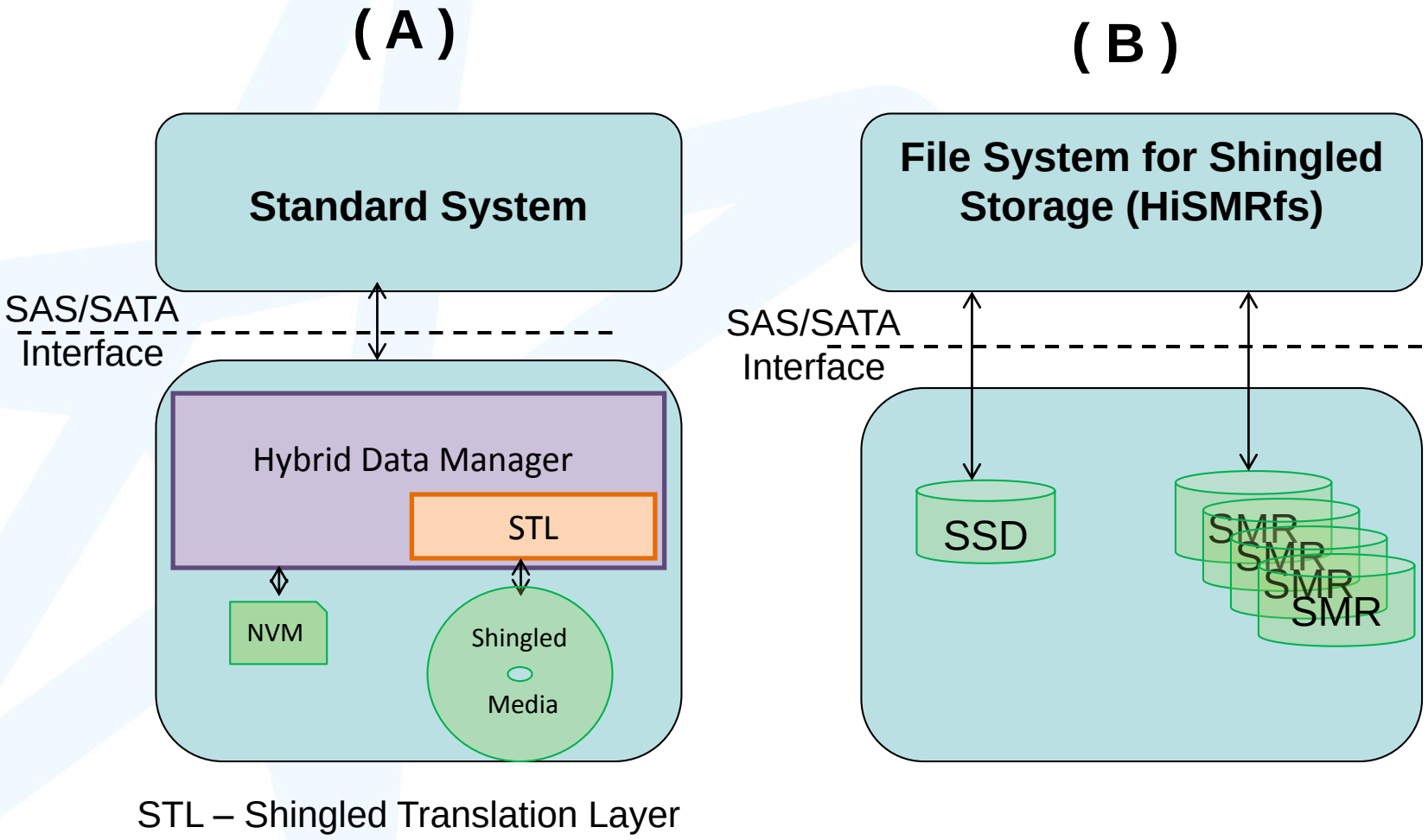
No Write-in-place/update



Approaches


SMR
Host
Unrestricted
READ
&
WRITE
Data
Management
Layer
Restricted
READ
&
WRITE

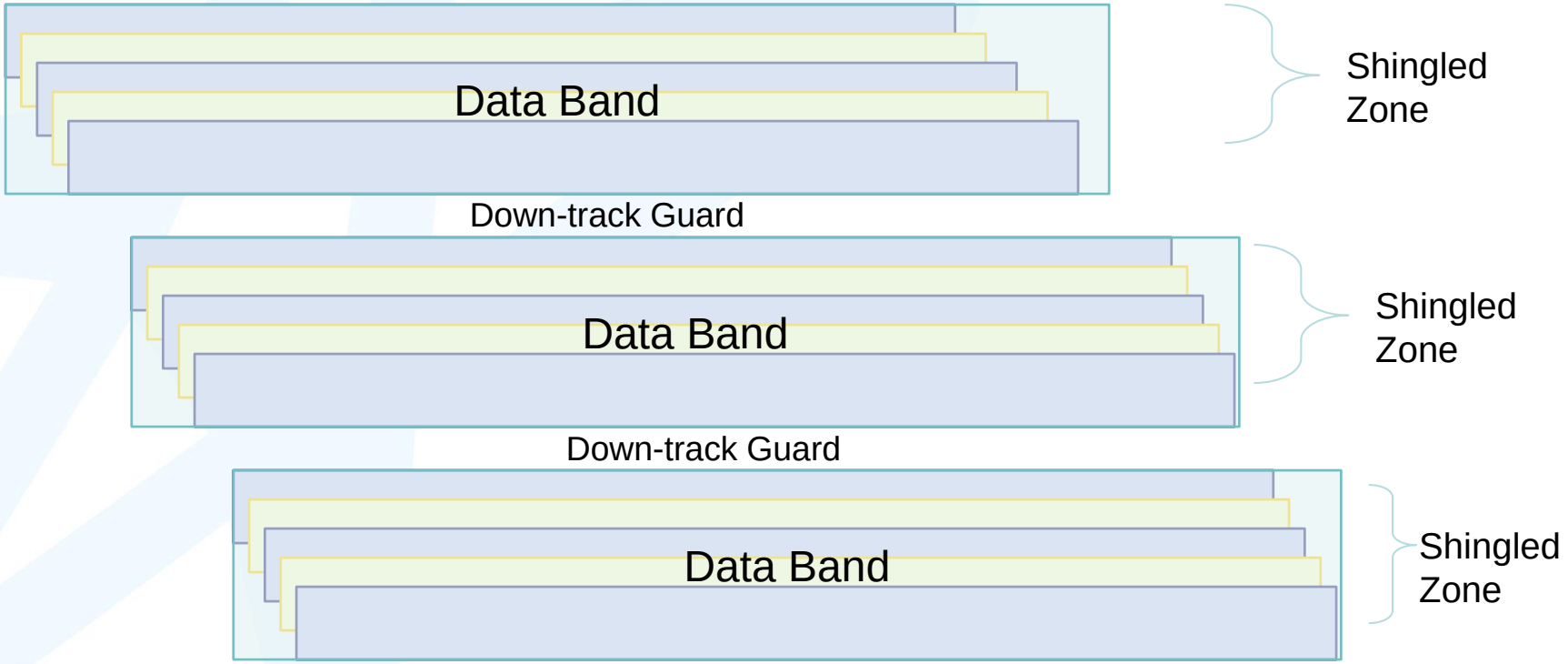
Our Approaches



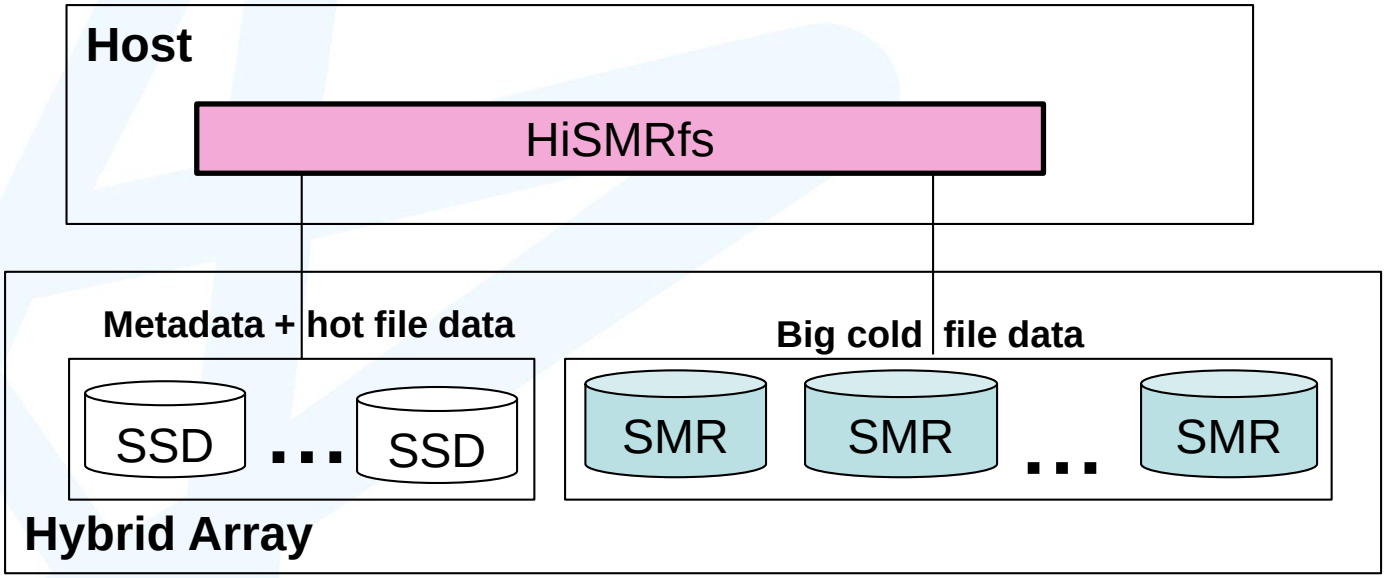
STL – Shingled Translation Layer

DSI A*STAR

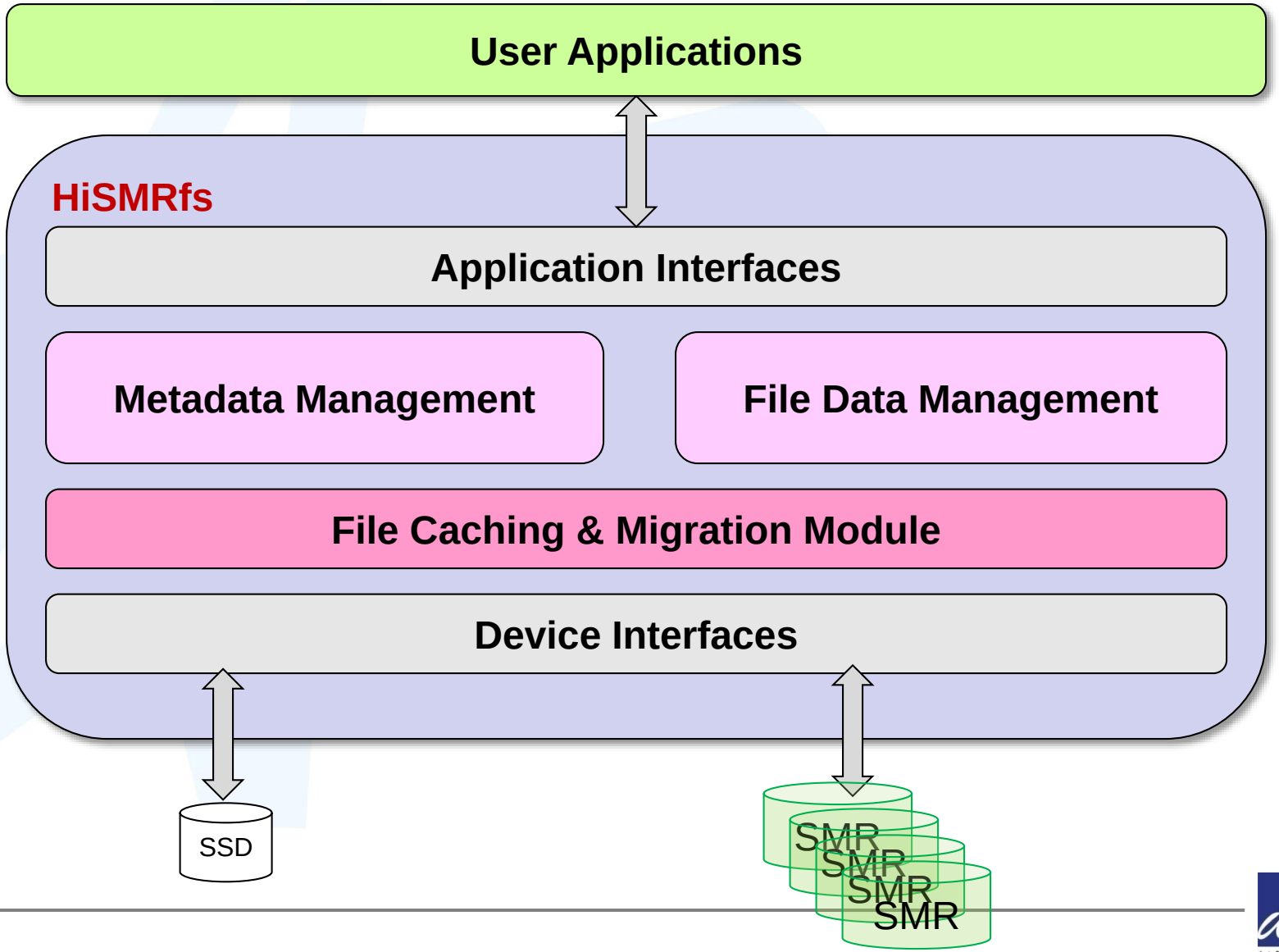
SMR Disk


Data Band
Shingled Zone
Down-track Guard
Data Band
Shingled Zone
Down-track Guard
Data Band
Shingled Zone

HiSMRfs


Host
HiSMRfs
Metadata + hot file data
Big cold file data
SSD
SSD
SMR
SMR
SMR
Hybrid Array

HiSMRfs - Architecture



DSI A*STAR

Application Interfaces

- HiSMRfs implements POSIX standard interfaces to file system clients.
- Existing applications require no modifications to run on top of HiSMRfs.

```
struct fuse_operations hybridfs_ops = {
    .init        = hybridfs_fuse_mount,
    .destroy     = hybridfs_fuse_umount,
    .getattr     = hybridfs_fuse_getattr,
    .fgetattr    = hybridfs_fuse_fgetattr,
    .access      = hybridfs_fuse_access,
    .statfs      = hybridfs_fuse_statfs,
    .open        = hybridfs_fuse_open,
    .release     = hybridfs_fuse_release,
    .fsync       = hybridfs_fuse_fsync,
    .flush       = hybridfs_fuse_flush,
    .utimens     = hybridfs_fuse_utimens,
    .chmod       = hybridfs_fuse_chmod,
    .chown       = hybridfs_fuse_chown,
    .create      = hybridfs_fuse_create,
    .truncate    = hybridfs_fuse_truncate,
    .ftruncate   = hybridfs_fuse_ftruncate,
    .unlink      = hybridfs_fuse_unlink,
    .rename      = hybridfs_fuse_rename,
    .mkdir       = hybridfs_fuse_mkdir,
    .rmdir       = hybridfs_fuse_rmdir,
    .opendir     = hybridfs_fuse_opendir,
    .readdir     = hybridfs_fuse_readdir,
    .releasedir  = hybridfs_fuse_releasedir,
    .fsyncdir    = hybridfs_fuse_fsyncdir,
    .write       = hybridfs_fuse_write,
    .read        = hybridfs_fuse_read,
    .setxattr    = hybridfs_fuse_setxattr,
    .getxattr    = hybridfs_fuse_getxattr,
    .listxattr   = hybridfs_fuse_listxattr,
    .removexattr = hybridfs_fuse_removexattr,
                   ......
```

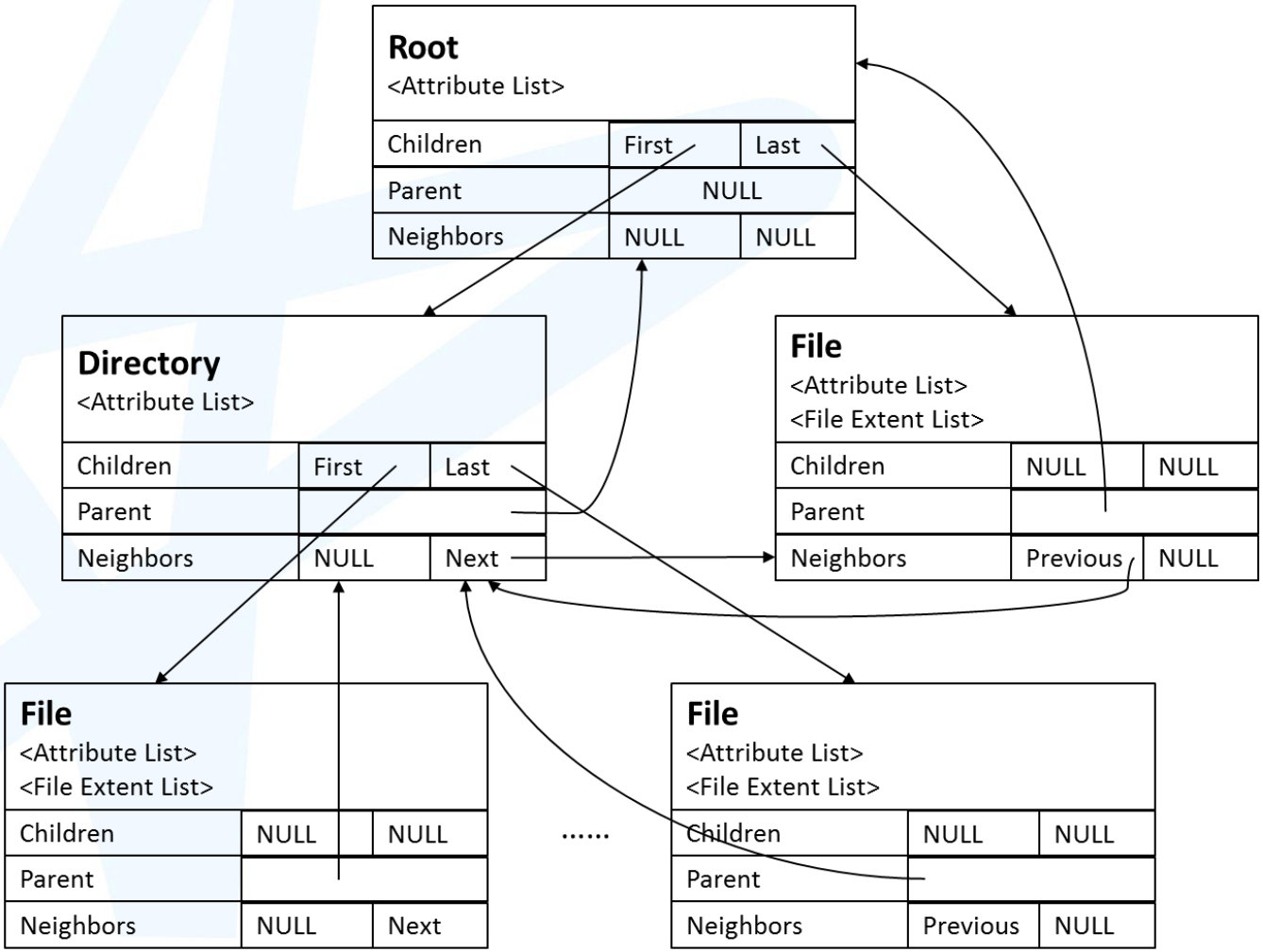
Metadata & File Data Management

- HiSMRfs separates metadata and file data in its data layout
  - Metadata are stored in higher performance & randomly accessible SSD; file data are stored in SMR disks
  - **Reason:** metadata are accessed more often with small random requests, file data are usually accessed sequentially.
- Different writing styles to the metadata partition and file data partition
  - **Metadata:** randomly written, can be overwritten, in-place update
  - **File data:** within each band, strictly written in order, cannot be overwritten

Metadata Management Schemes

- **Current implemented scheme:** in-memory metadata tree + on-disk metadata file
  - The directory hierarchy is stored as a tree structure in the DRAM
  - Root node or intermediate node represents a directory, leaf node represents a file
  - The in-memory metadata structure is periodically synchronized to a file on the disk

In-Memory Metadata Tree Structure



File Data Management Scheme

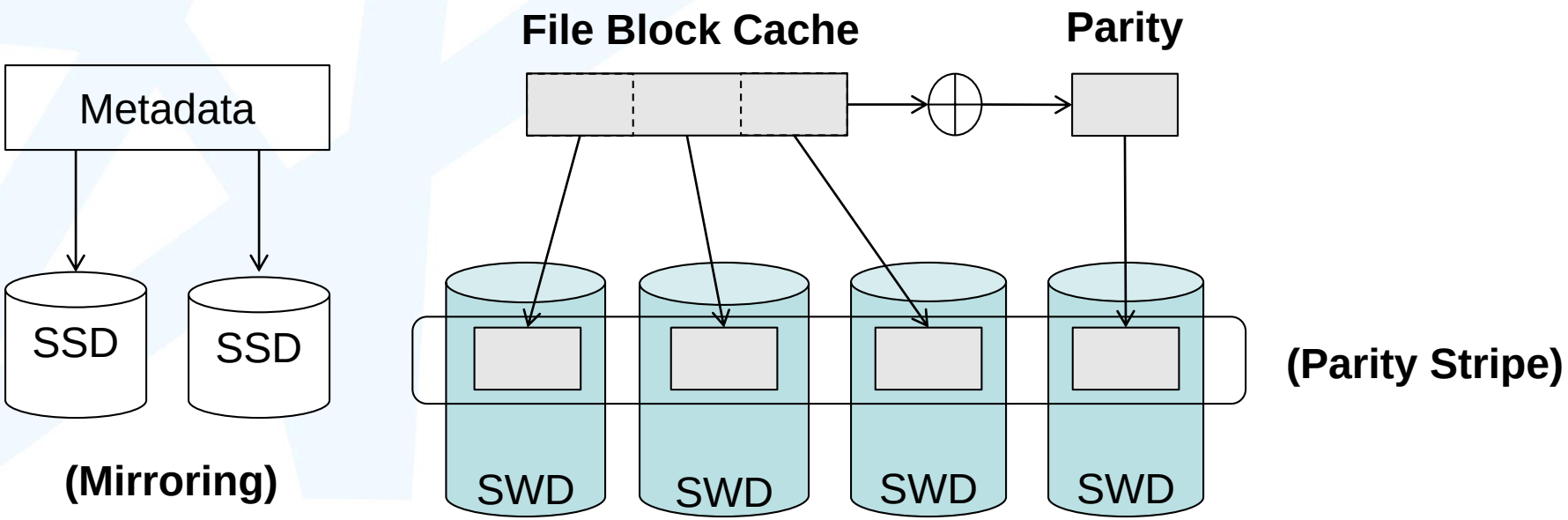
- File data are written sequentially into SMRs
- **Out-of-place update**: updated data is appended at the current sequential writing point, the original data blocks are marked as invalid
- Invalid data blocks are reclaimed by scheduled **space reclamation** process.

File Caching & Migration Policy

- New files are created & allocated in the SSD.
- When the size of a file grows and exceeds certain limit, the file is migrated to the SMRs.
- Cold files (rarely accessed files) are migrated to the SMRs; hot small files are cached in the SSD.

Redundancy Design

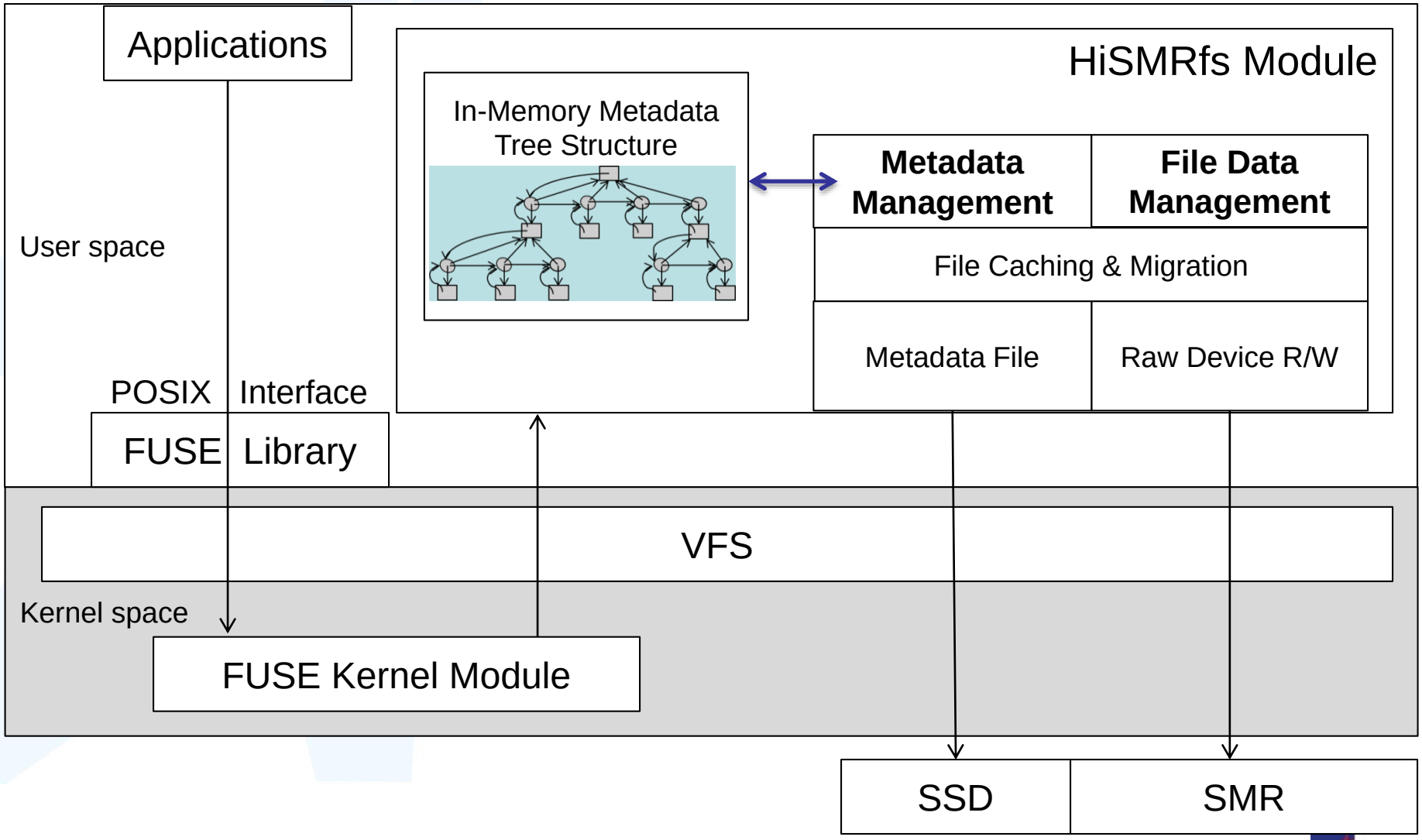
- Each file system block is divided into several sub-blocks, and the XOR sum of the sub-blocks is calculated as the parity sub-block.
- The whole parity stripe is written to the array via full-stripe write.
- Optimized Reconstruction Performance (skip invalid file blocks).



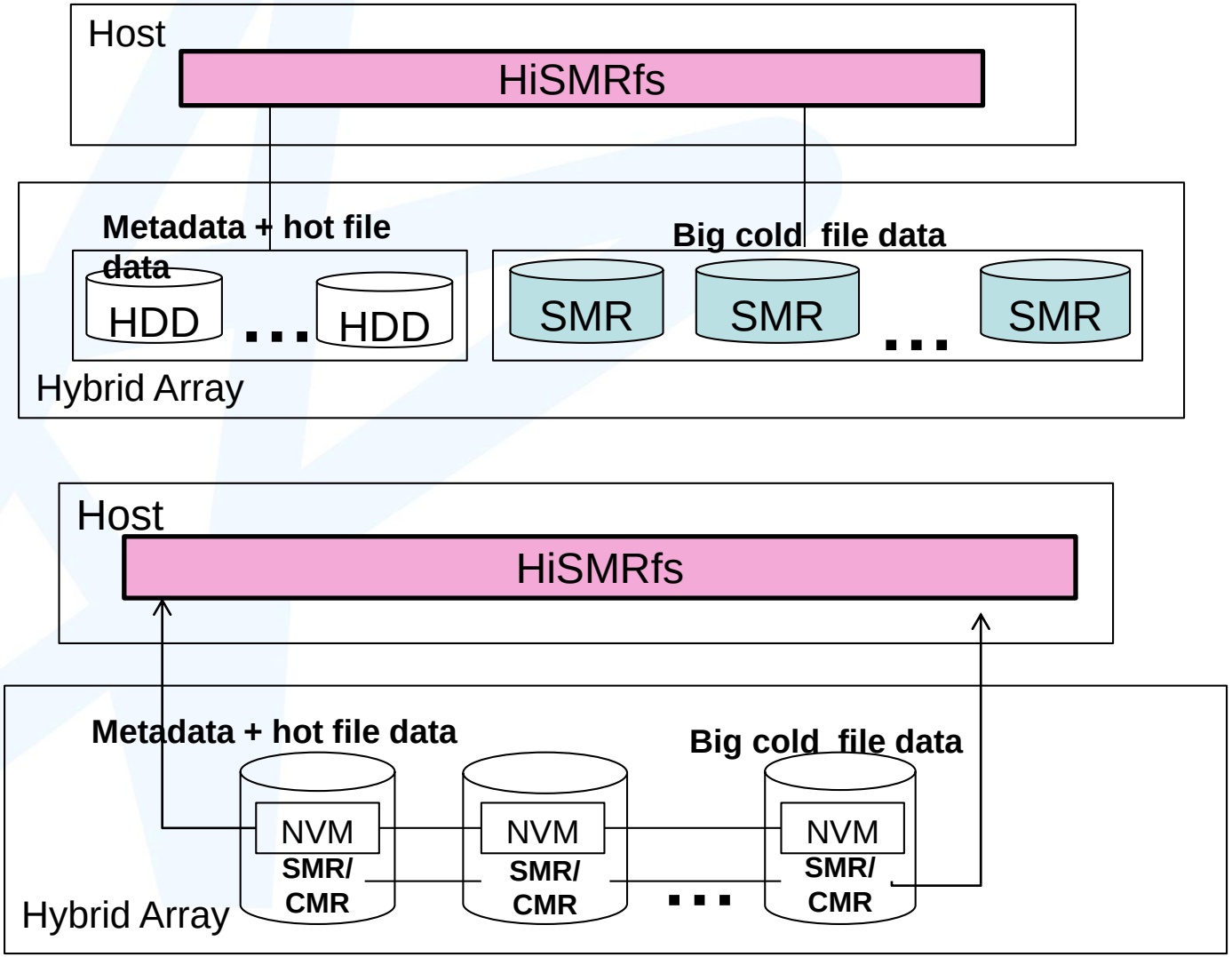
Prototype Implementation

- A workable HiSMRfs prototype is implemented under the Linux platform as a **user-level** file system based on the **FUSE** framework, and provides POSIX interfaces to the user applications.
- The metadata is organized as a hierarchical tree structure in the memory.
- The metadata tree is synchronized to file in the SSD.
- The file data is accessed from user space by calling the R/W interfaces directly to the disks. The current append position of the file partition is recorded in metadata.

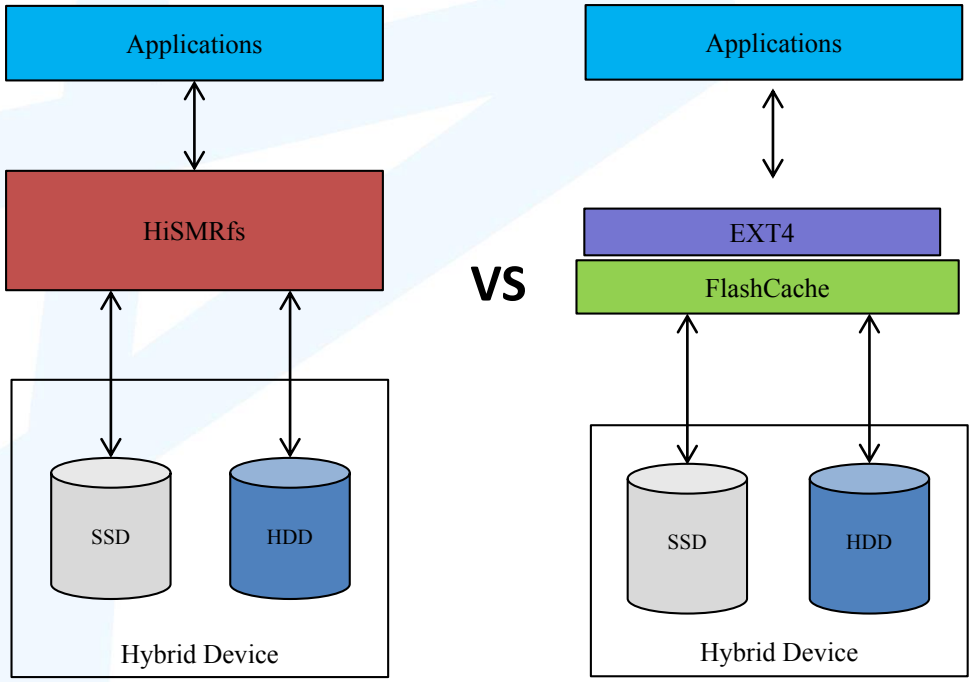
Implementation Architecture



HiSMRfs


Host
HiSMRfs
Metadata + hot file data
HDD
HDD
Big cold file data
SMR
SMR
SMR
Hybrid Array
Host
HiSMRfs
Metadata + hot file data
NVM
SMR/
CMR
NVM
SMR/
CMR
Big cold file data
NVM
SMR/
CMR
Hybrid Array

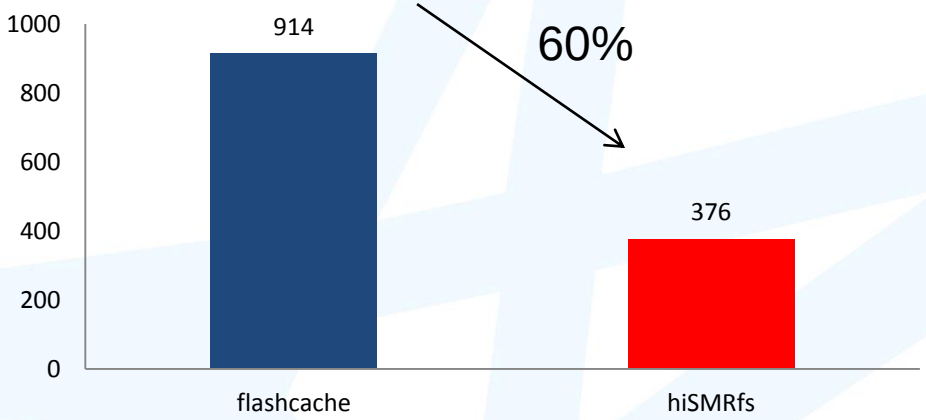
Prototype I – one SSD and one HDD



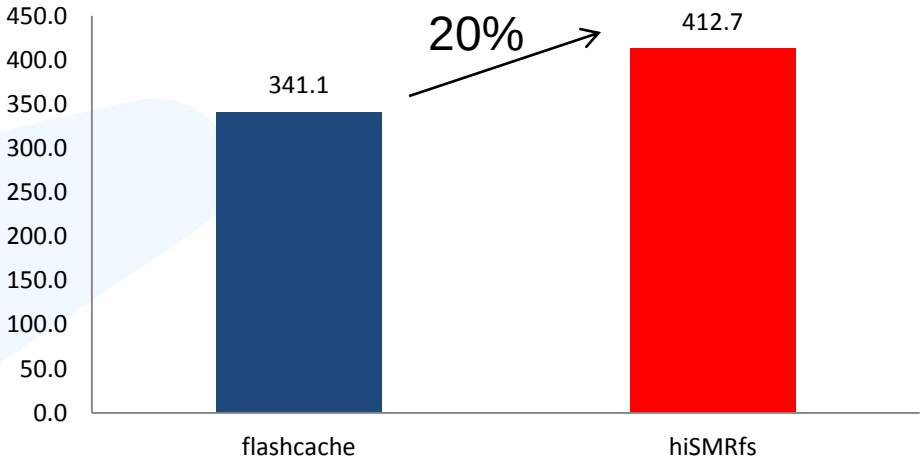
Fileserver Workload

Pre-Allocation time (second)

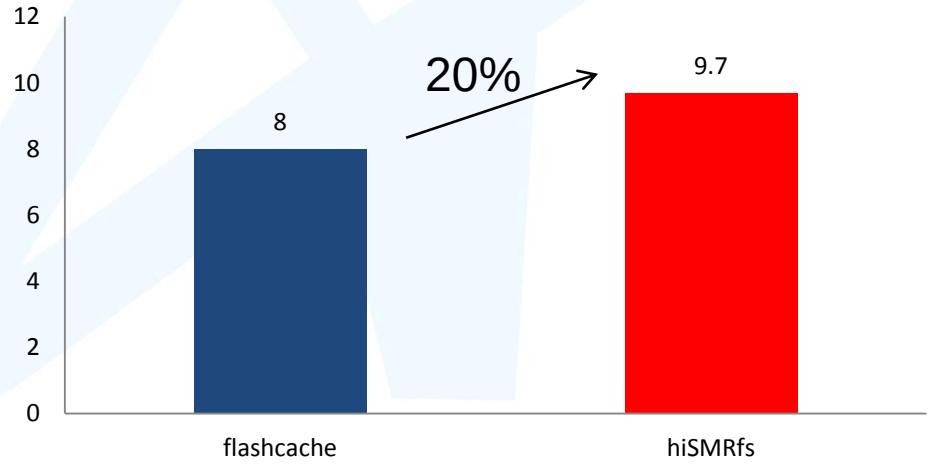
Pre allocate 200K files, avg file size 128KB



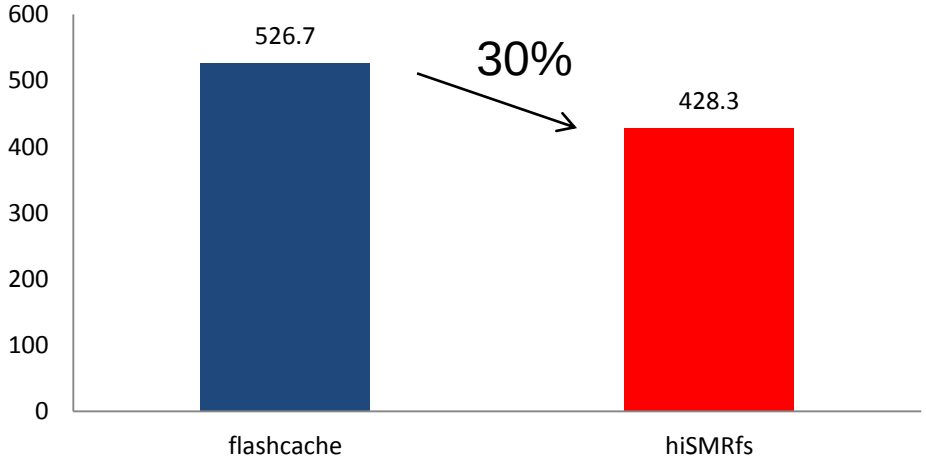
IOPS



Throughput (MB/s)



Latency(ms)

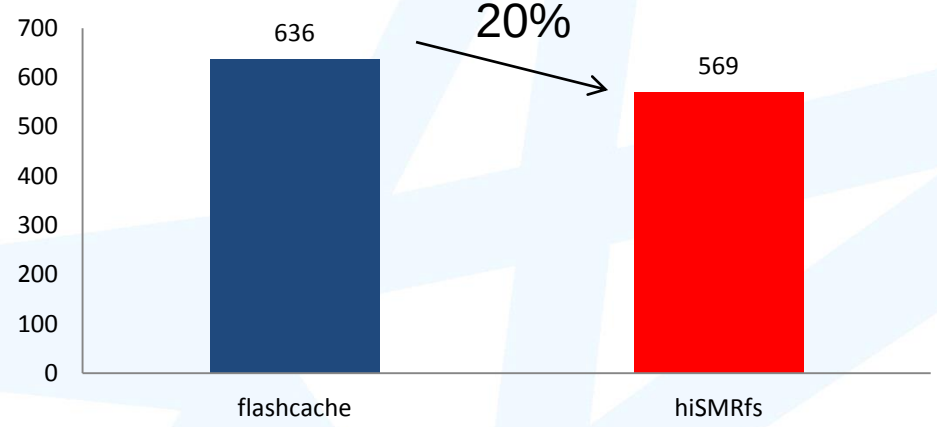


A*STAR DSI

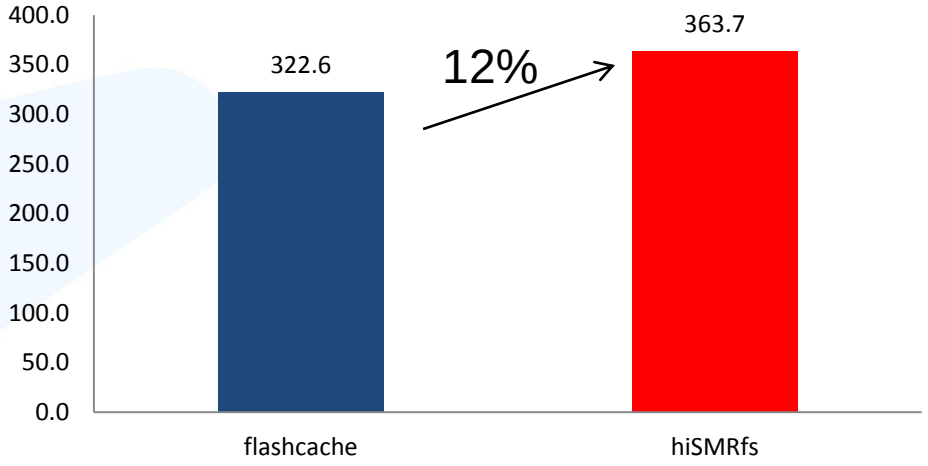
Mail Server Workload

Pre-Allocation Files (second)

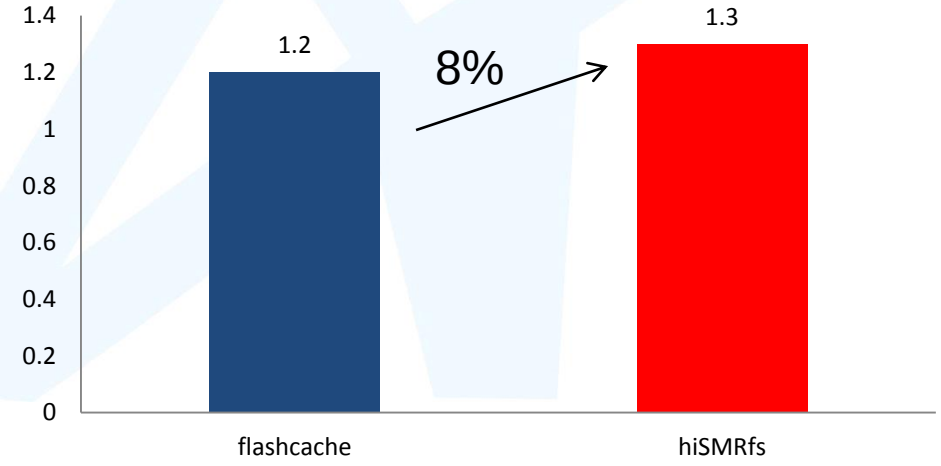
Pre allocate 1million files, ave file size 16KB



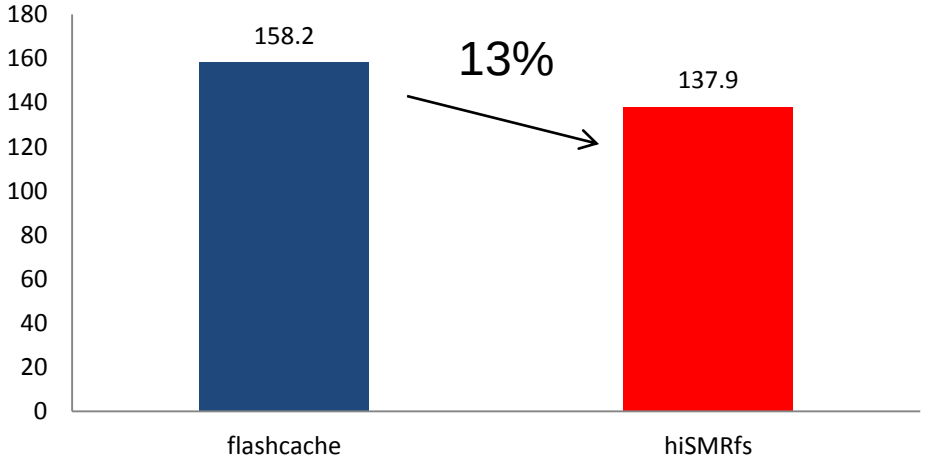
IOPS



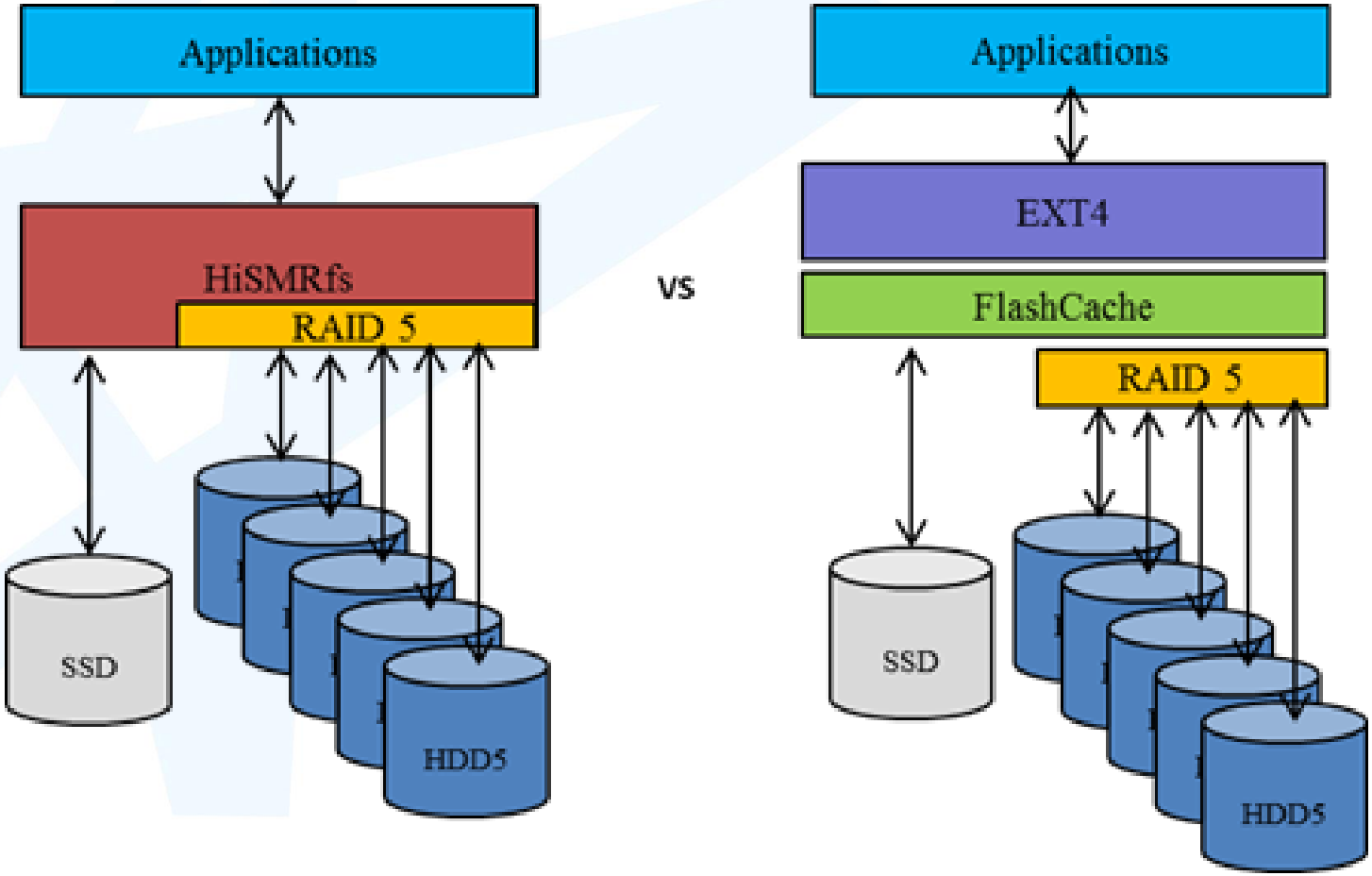
throughput (MB/s)



latency (ms)



Prototype II – Storage Array

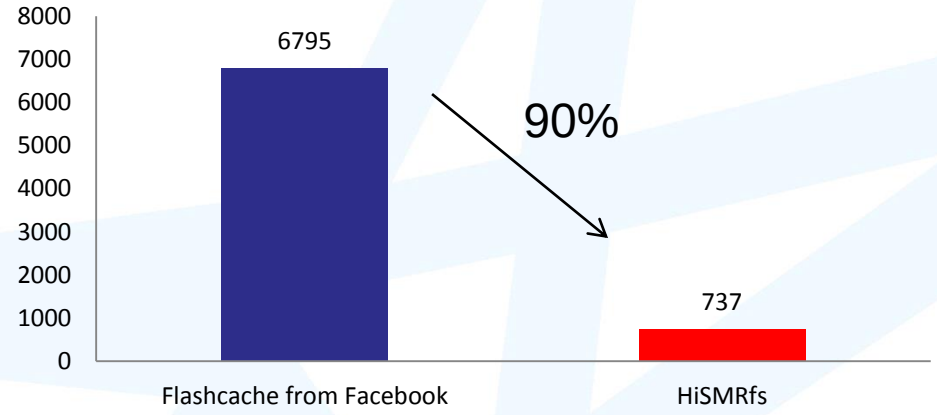


DSI A*STAR

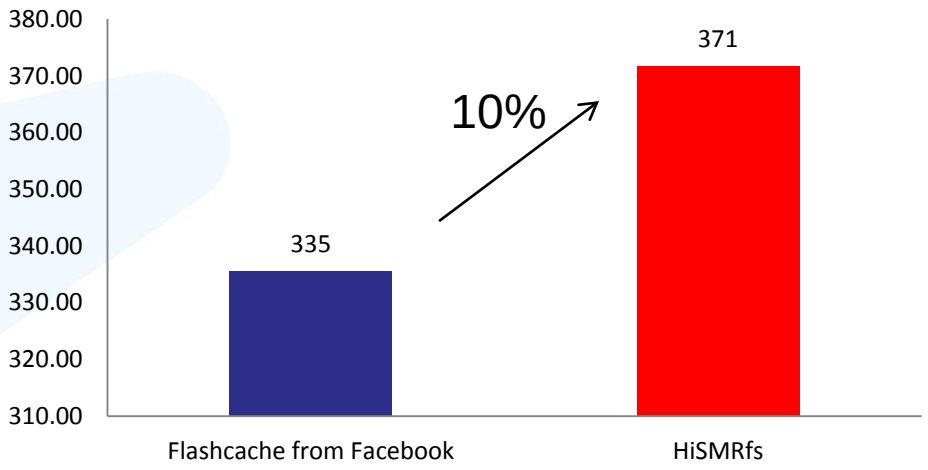
File Server Workload

Pre-Allocation Time (second)

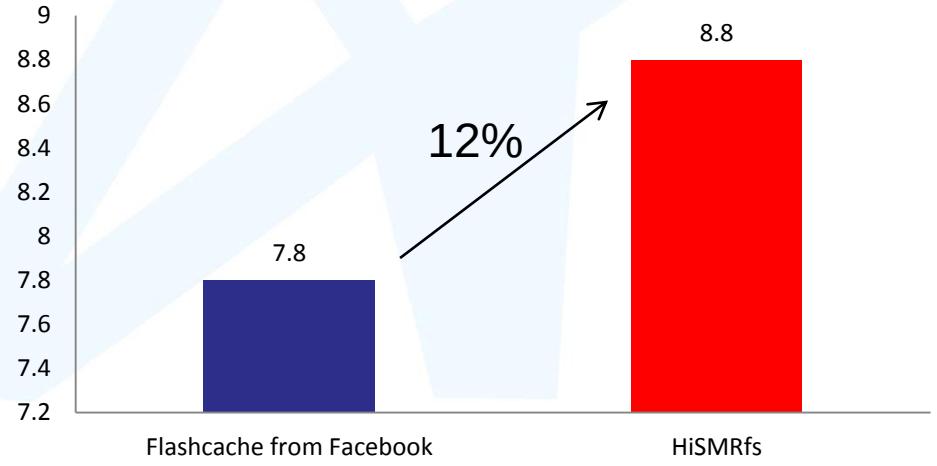
Pre allocate 200K files, avg file size 128KB



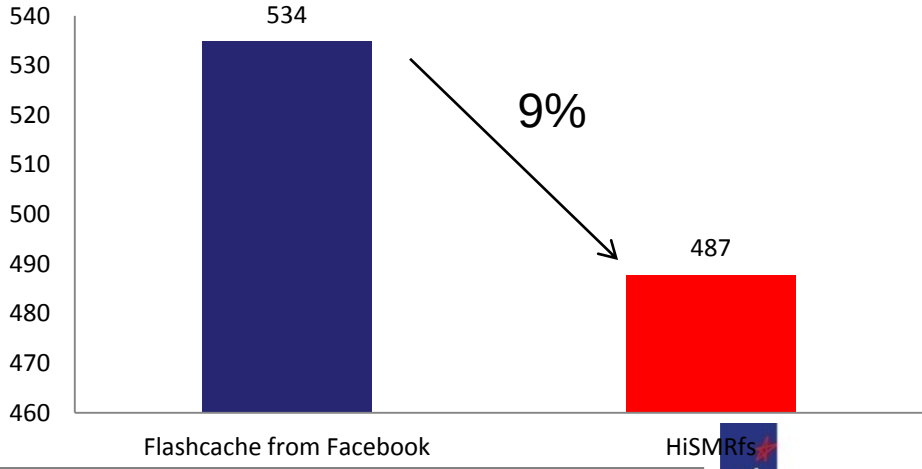
IOPS



Throughput (MB/s)



Latency(ms)



DSI A*STAR

Thank You!