Can’t we all get along?

Redesigning Protection Storage for Modern Workloads

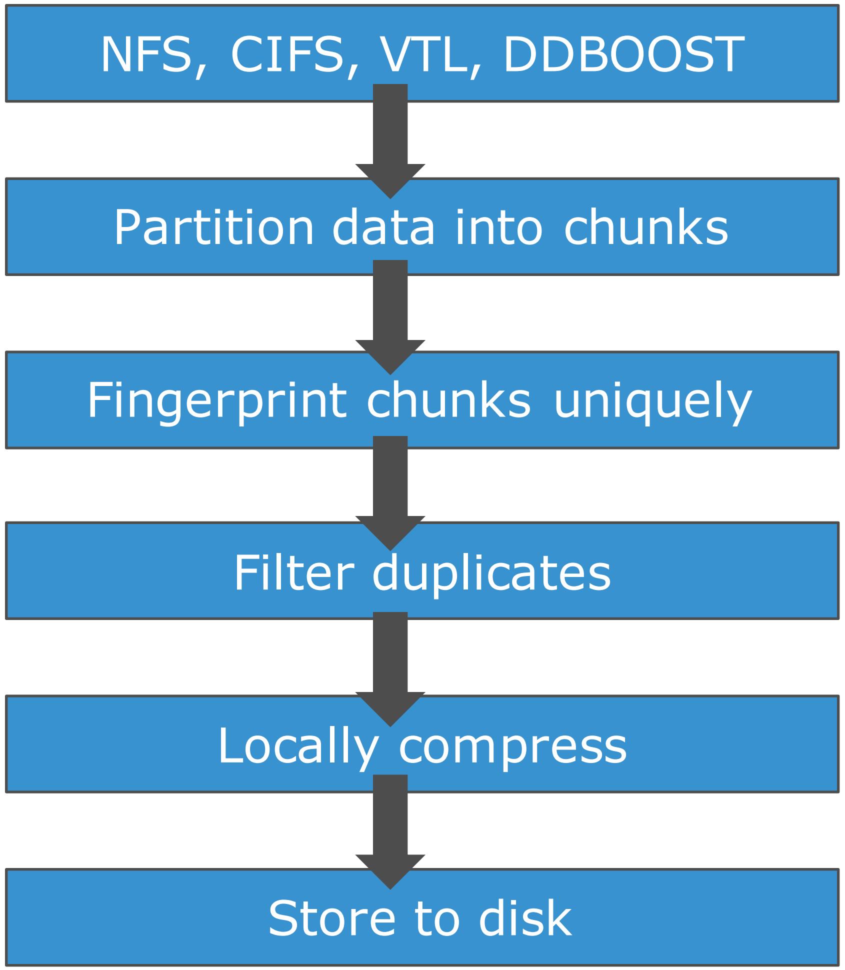
Yamini Allu
Fred Douglis*
Mahesh Kamat
Ramya Prabhakar
Philip Shilane
Rahul Ugale

* Perspecta Labs

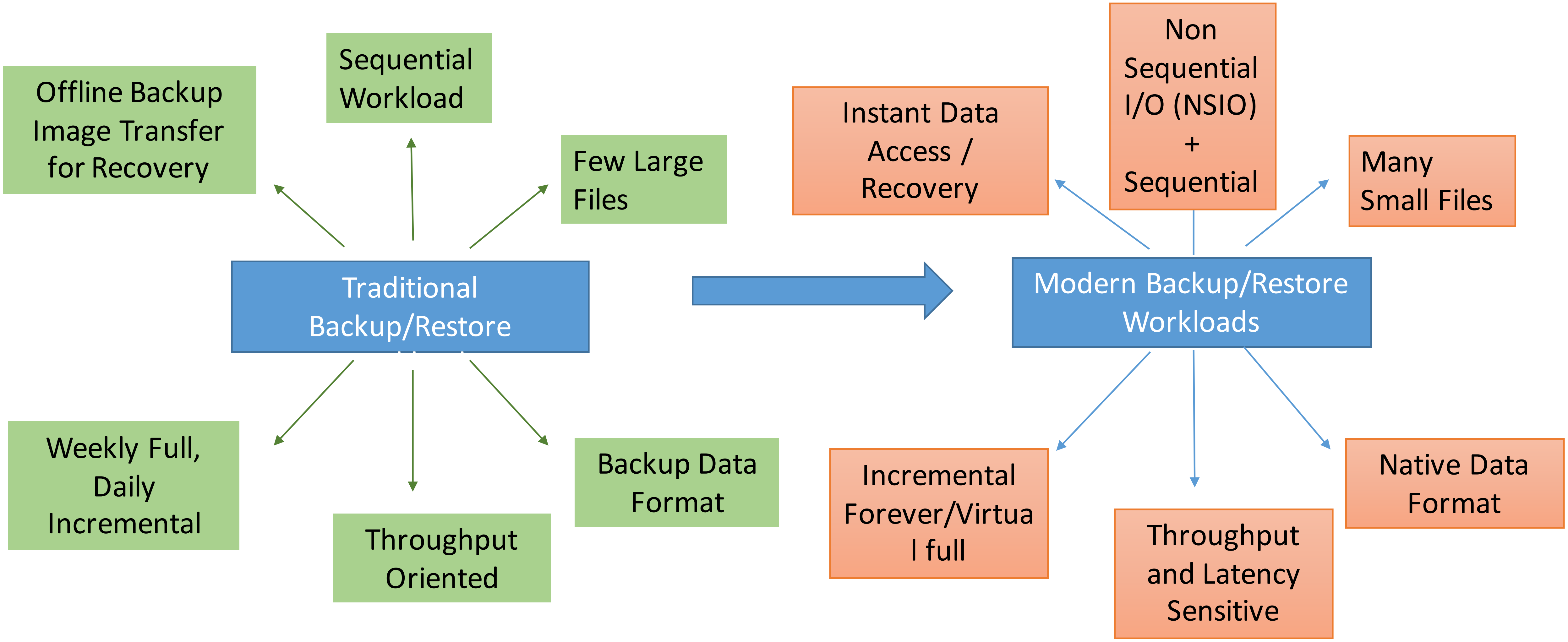
Data Domain File System

- Purpose-built backup appliance
  - Designed to identify duplicate regions of files and replace with references
  - Designed around backup workloads which is typically data written and read sequentially.
- De-duplication
  - Content-defined chunks, fingerprinted with secure hash
  - Generally claim 10-40x data reduction
- Avoiding disk bottlenecks in Data Domain[Zhu 08]
  - SISL: Stream informed segment layout
  - Locality preserved caching for segments

De-duplication storage pipeline

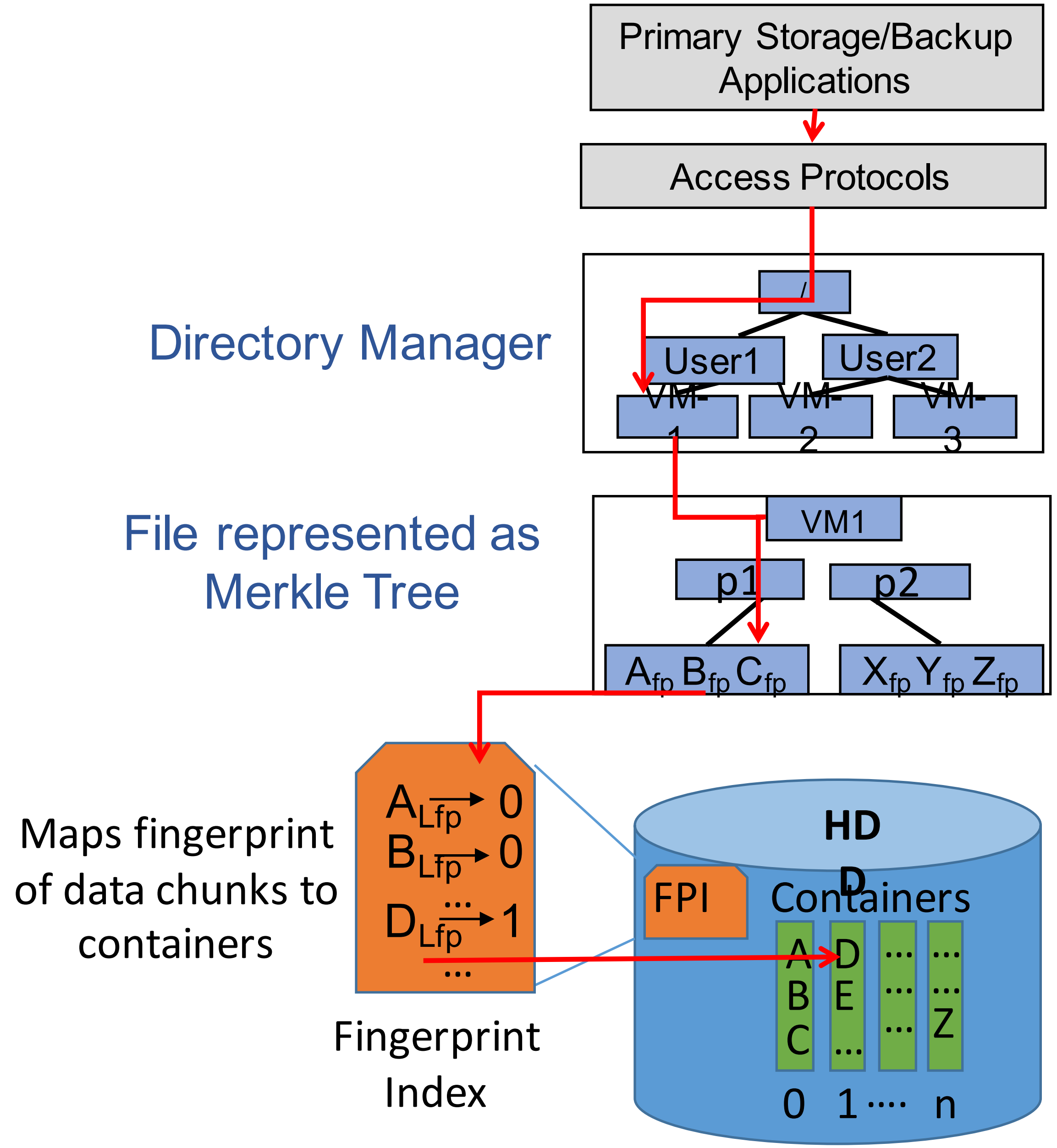


Shift in Data Protection Workloads



Challenge was to enhance our filesystem stack to support BOTH traditional and modern workloads

IO Profile for Traditional vs Modern



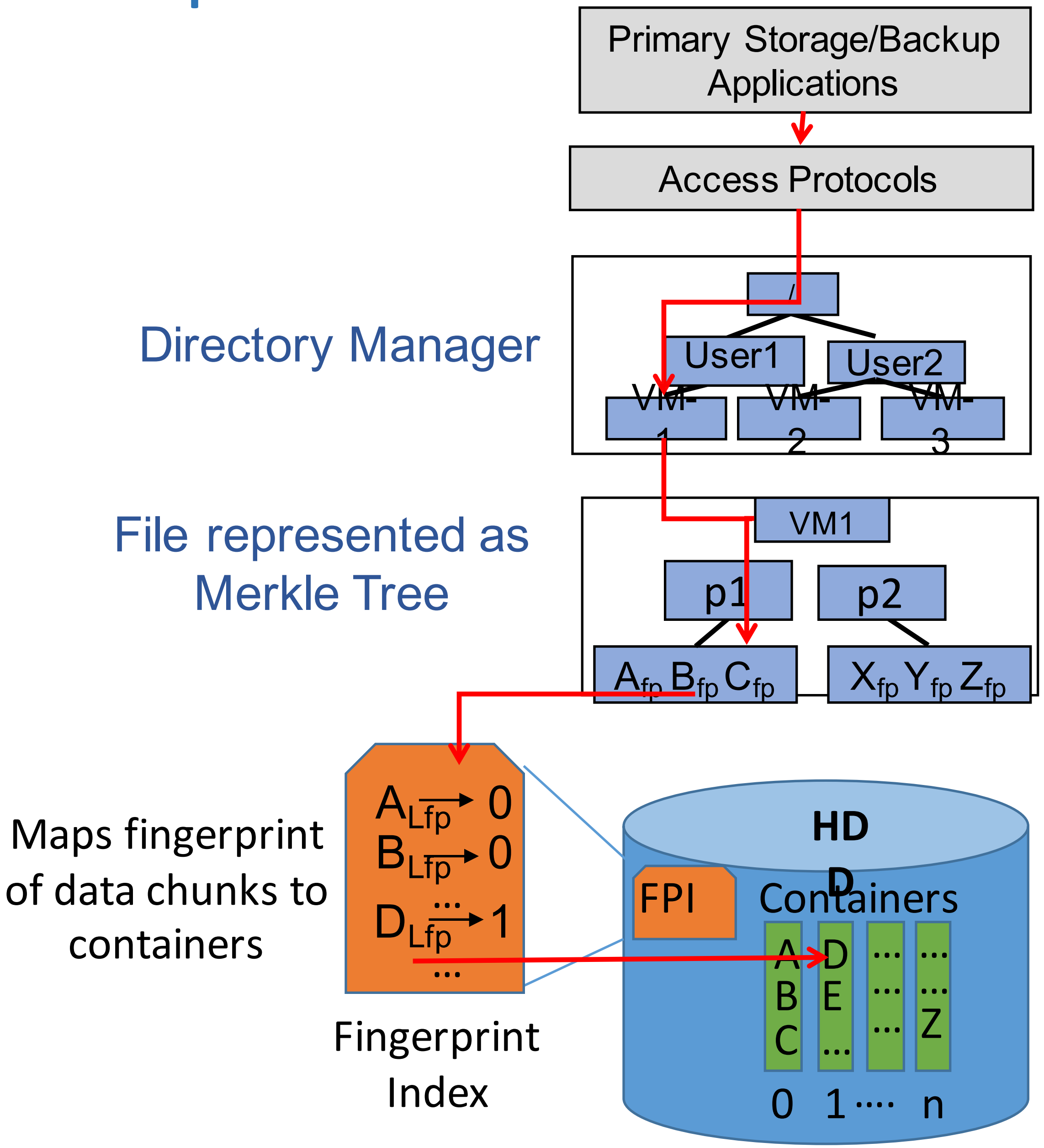
Traditional Workloads:

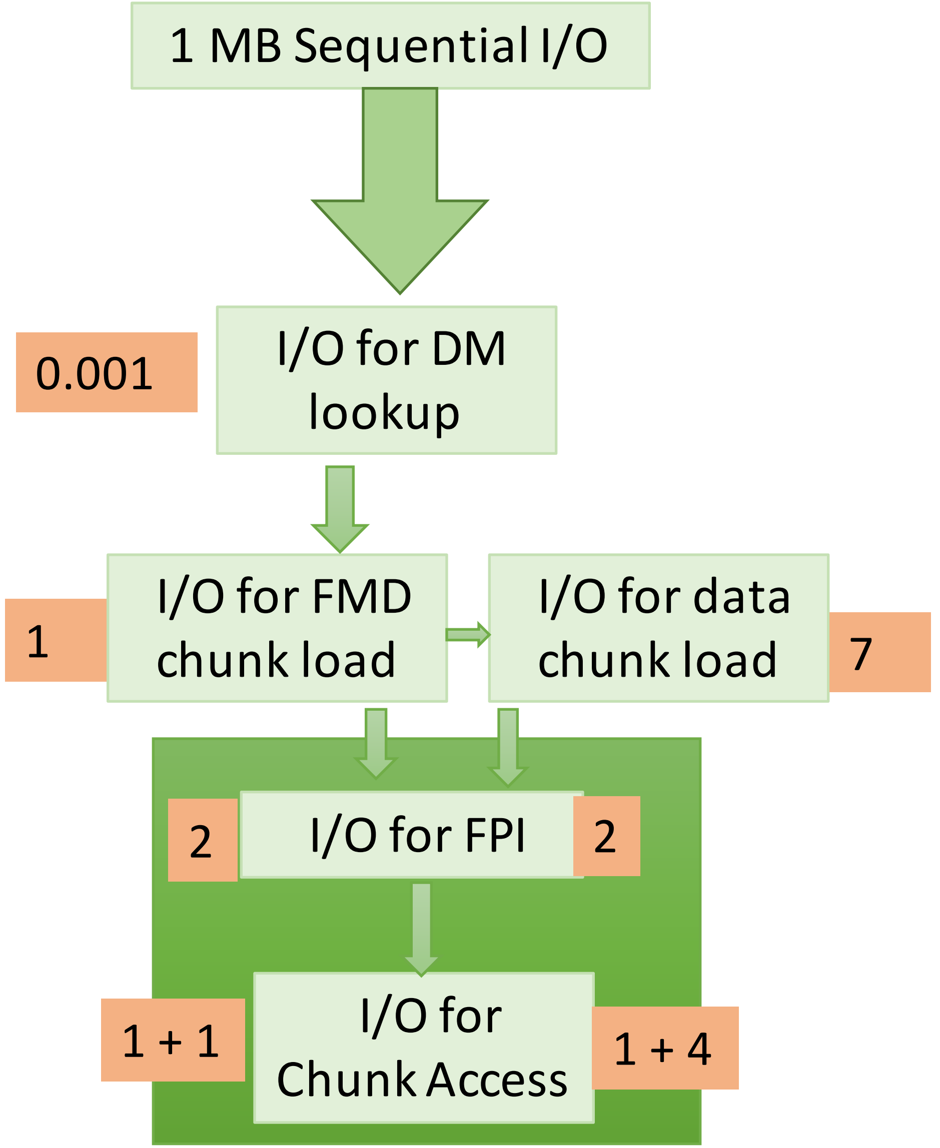
- Large files amortize Directory Manager lookup cost
- File Metadata (FMD) can be pre-fetched into memory for sequential I/O
- Good data locality – Only one FPI lookup for 1 MB
- Good data locality – fewer I/Os to fetch data chunks number of data can be Pre-fetched to memory

Modern Workloads:

- Increased Directory Manager lookups for native format backups (small files)
- FMD pre-fetch into memory is not efficient for NSIO
- Increased index lookups due to random locality (once every I/O)
- Random I/O to HDD for every request

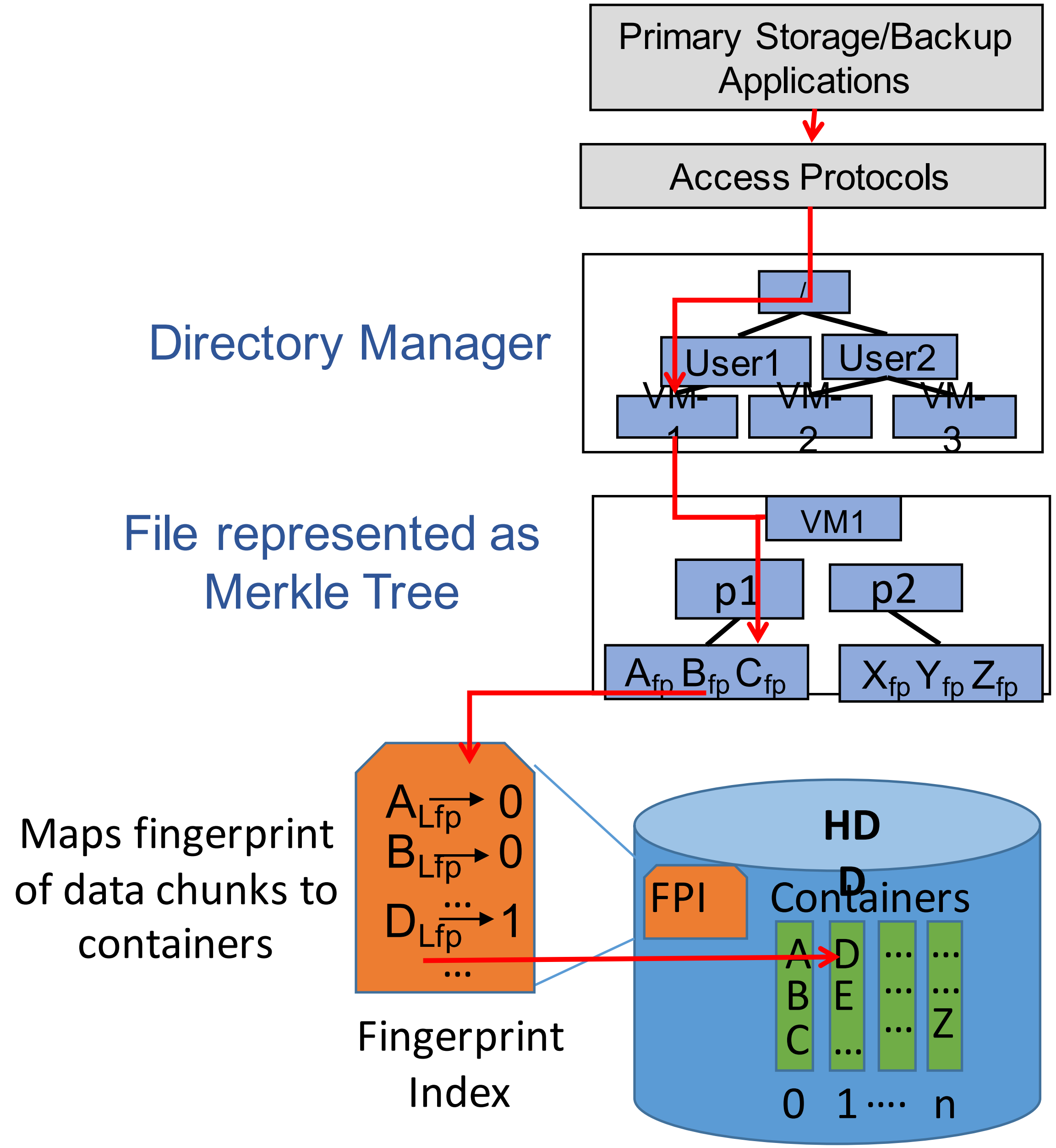
Example

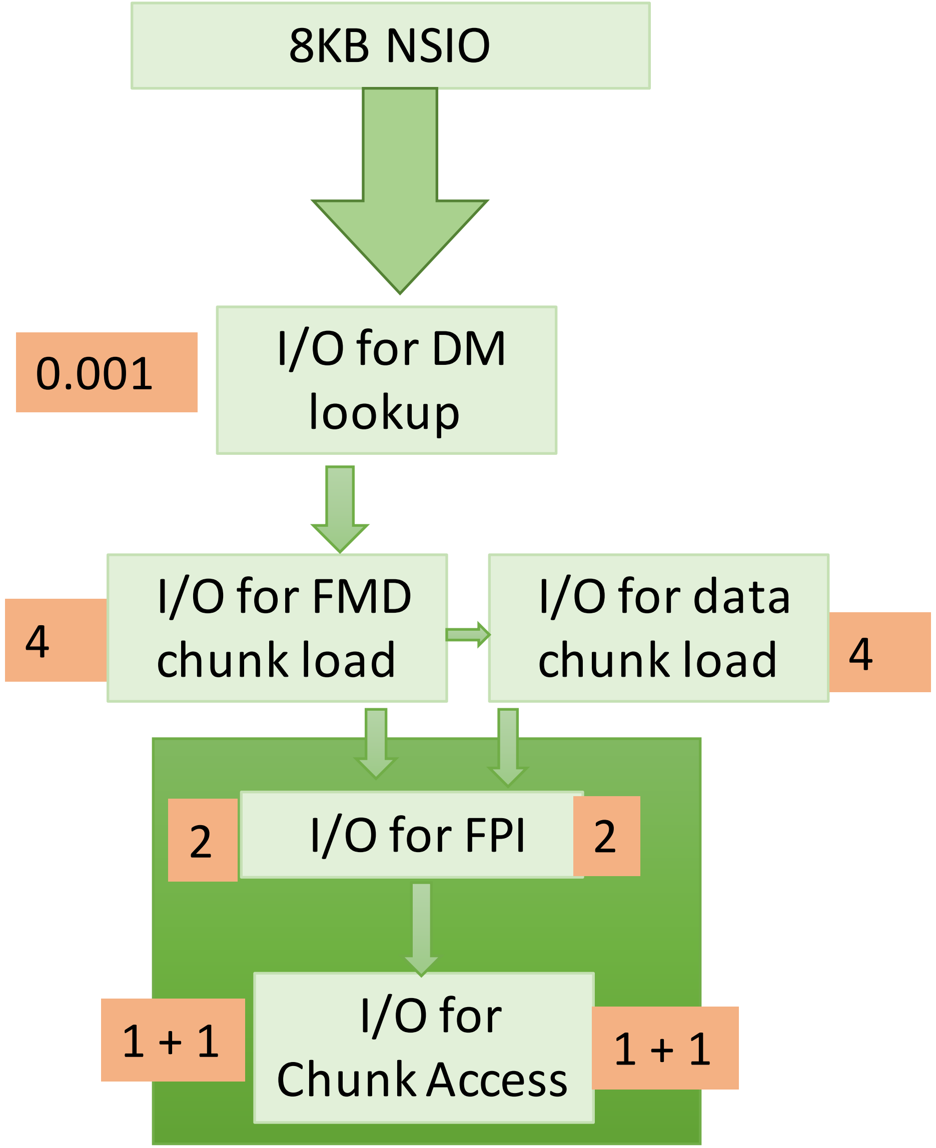




Total number of HDD I/Os = ~8 I/Os per 1MB

Example

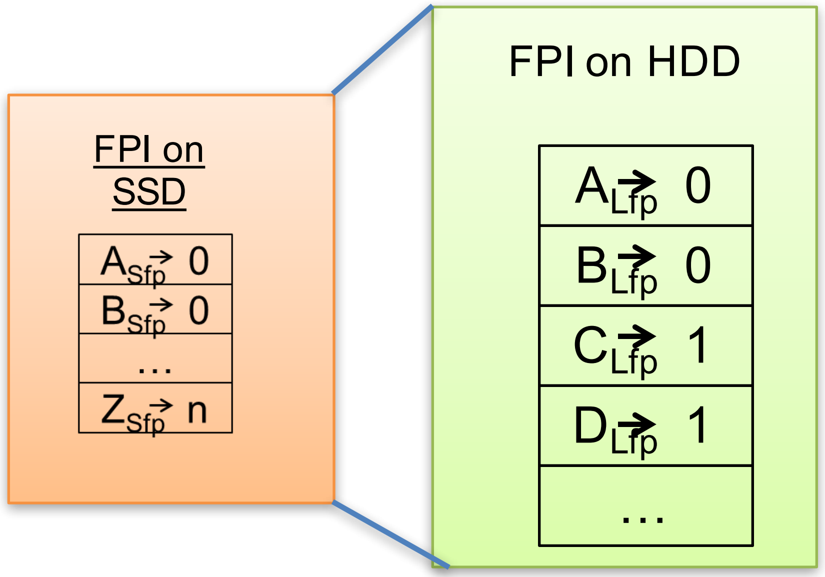


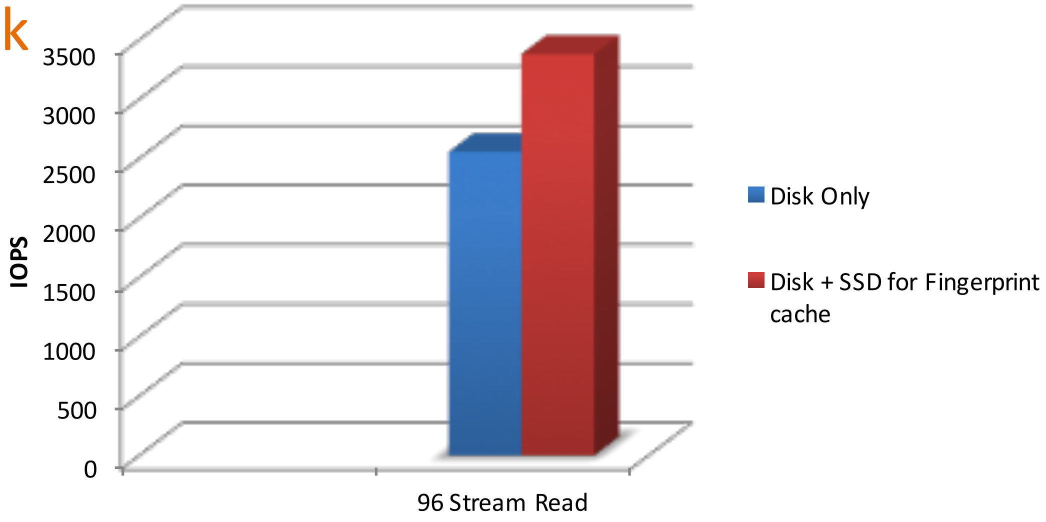


Total number of HDD I/Os = ~8 I/Os per 8KB
~1024 per 1MB

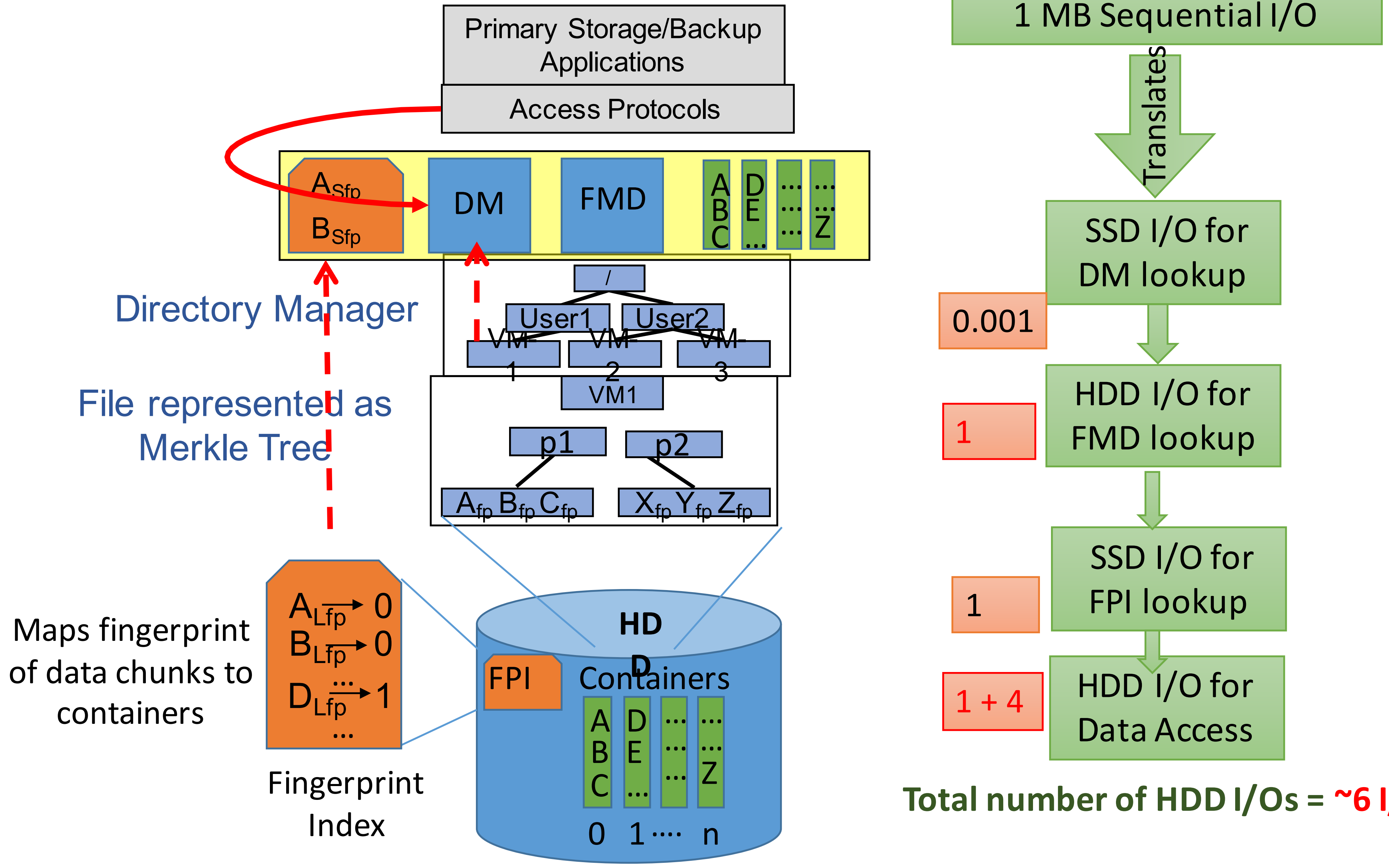
Fingerprint Index (FPI) Cache

- Two level index requires 2 HDD I/Os per chunk read
- For sequential restores, index I/Os are 25% of total I/Os, for NSIO, it is 50%
- Index metadata is 1.5% of total physical space
- Caching short fingerprints (4 bytes) in SSD reduces SSD space requirement to 0.4%, with a collision rate < 0.01%.
- Collisions resolved by comparing full FP on disk
- FPI cache improves sequential restore performance by up to 32% on I/O bound configurations





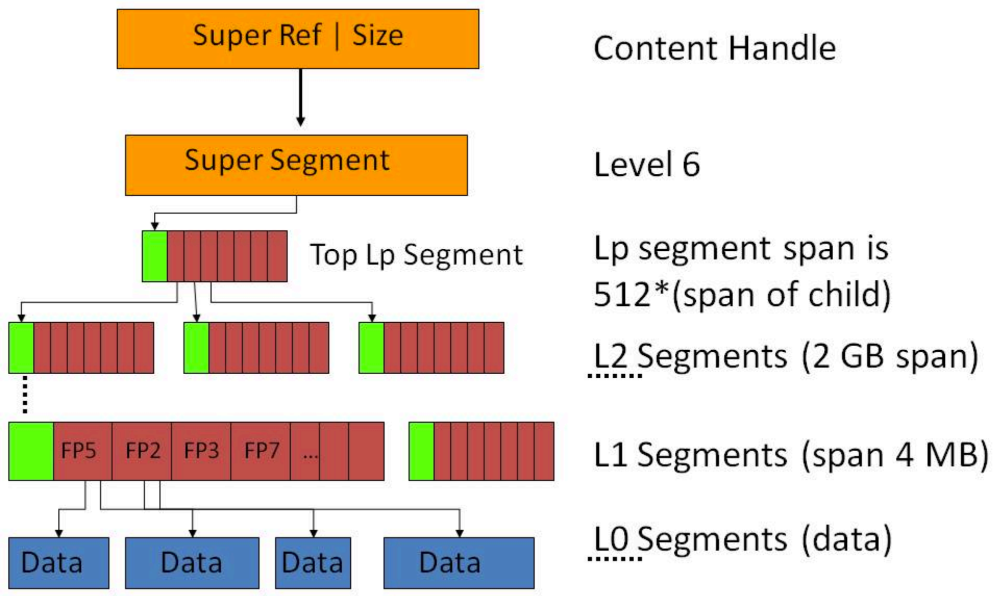
Addition of SSD Cache for PBBA (dense drive support)



Total number of HDD I/Os = ~6 I/Os per 1MB

File Metadata (FMD) Cache

- IOs for file metadata account to 50% of total I/Os for NSIO, FMD is cached only on NSIO accesses
- File metadata is variable sized (Average 16kb)
  - Accounts to 0.05% of logical backup size
- 10% of available SSD is reserved as FMD Cache to account for high metadata churn during NSIO
- FMD is packed into 1MB Write eviction units to SSD cache
- It is a **write through cache**, populated on FMD updates and read misses
- **Eviction is LRU** based, this keeps most relevant data in cache



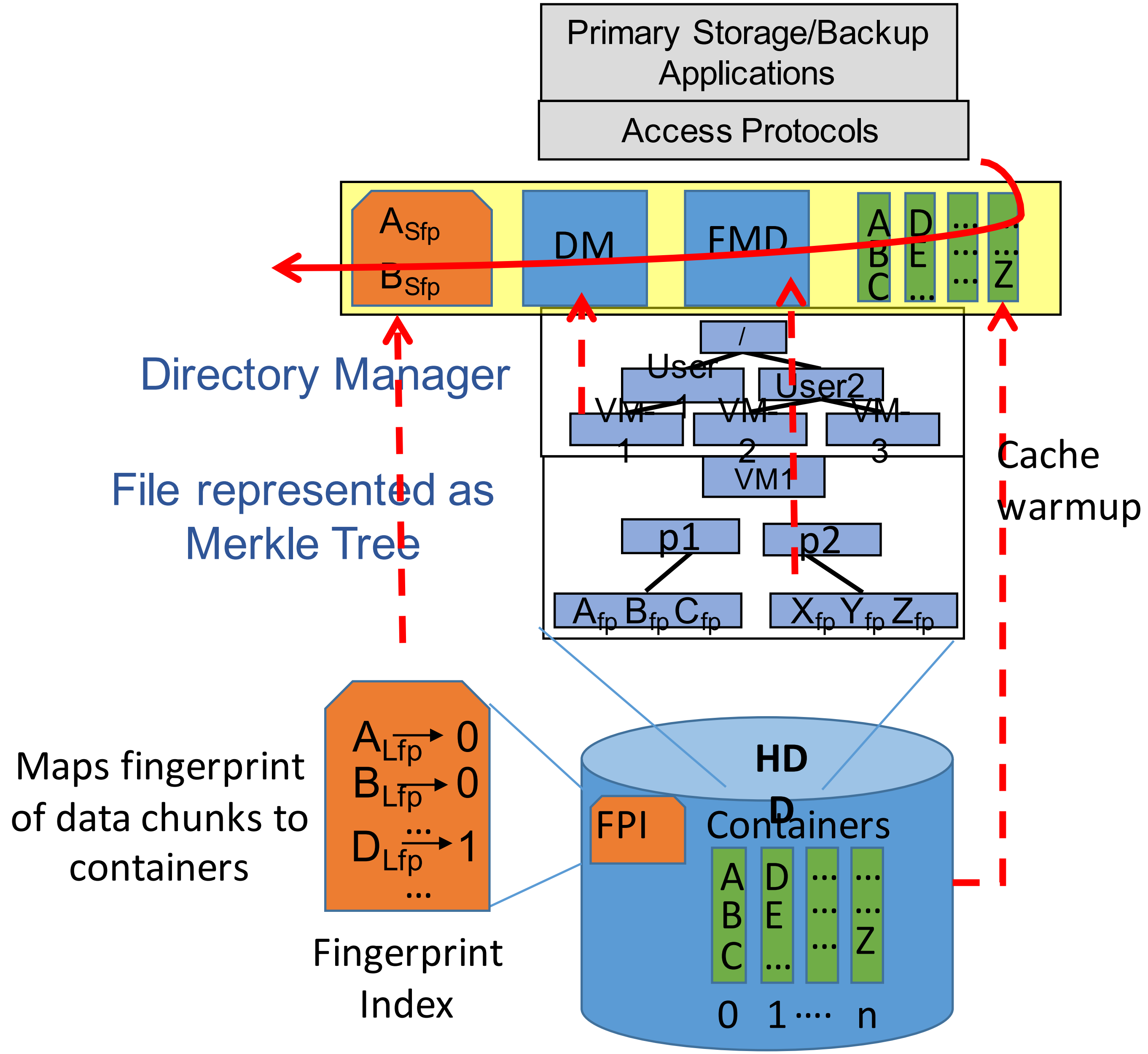
Just FMD Cache reduces HDD I/Os required for 8KB NSIO from 8 I/Os to 4 I/Os

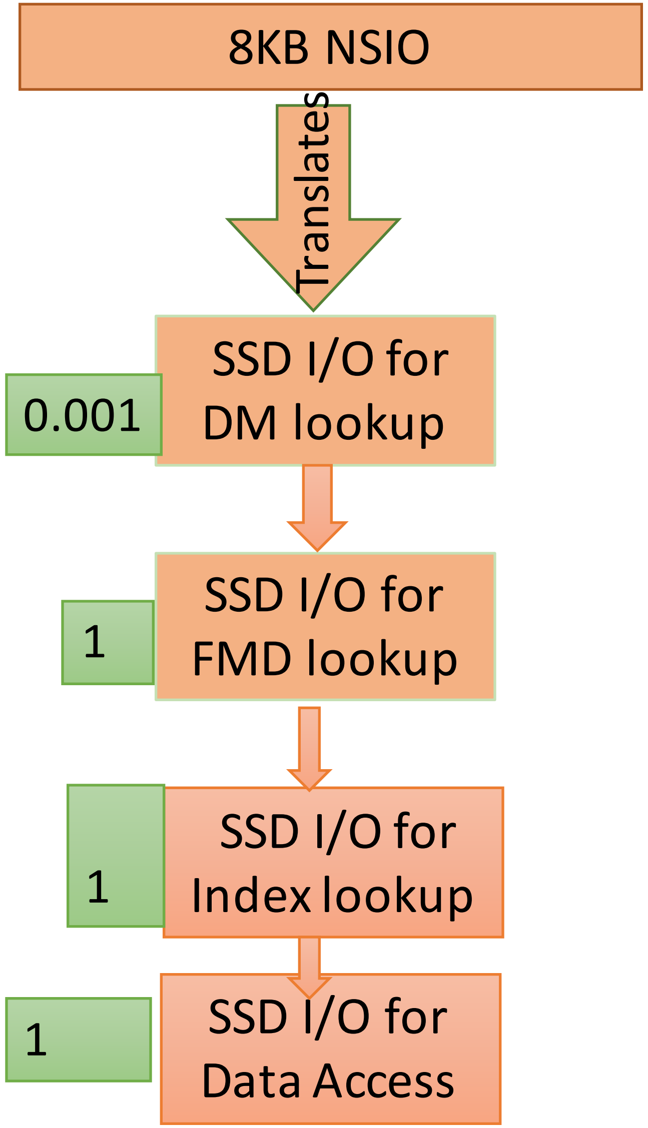
Data Cache on SSD

- Cache chunks of data that are randomly accessed within a file
- Data is cached on a read miss and on write
- Chunks are gathered and written to cache as 1MB Eviction unit. Eviction algorithm is LRU
- Larger chunks are read-ahead and cached until the cache is sufficiently warm
- 40% of SSD cache is allocated for Data, sufficient to instantly access up to 32 VMDKs and support up to 50k IOPs on our larger systems
- Data is not compressed on SSD to reduce CPU utilization

- Data Cache reduces NSIO I/Os to HDD to 0

Addition of SSD Cache for NSIO





Total number of HDD I/Os = 0* I/Os

*Assumes 100% cache hit, typical read/write workload sees >95% cache hits

File-system Modifications For NSIO Support

Access pattern detection

- Detect Sequential, NSIO Monotonic and NSIO patterns
- Detect multiple access patterns across different regions of a file
- Based on a sufficient history of past I/Os within a region of a file

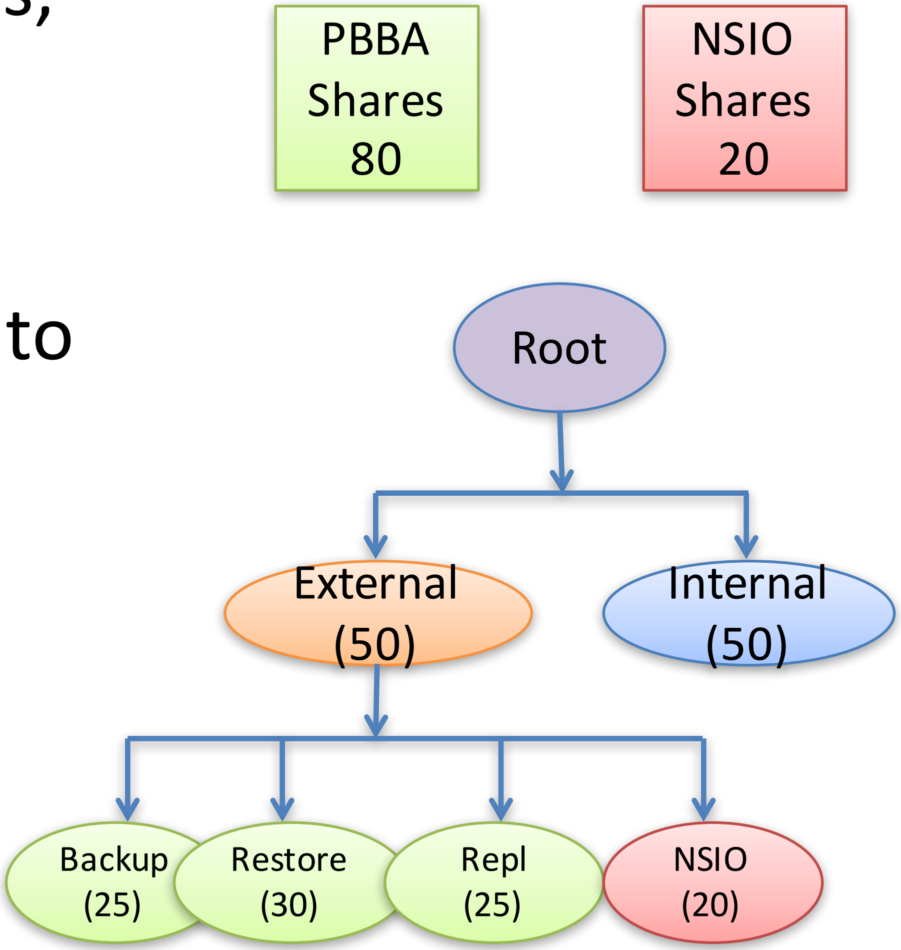
Access history per region

	0MB-1MB 1MB-2MB 2MB-3MB	2GB-2.1GB 2.8GB-3GB 3.1GB-3.2GB	4.2GB-4.3GB 4.0GB-4.1GB 4.9GB-5.0GB
Span:	0GB -2GB	2GB-4GB	4GB-6GB
label:	Sequential	NSIO-Monotonic	NSIO

Disable simple prefetch for NSIO	Enable parallel FMD and FPI lookups for NSIO	Fixed size chunking for image backups
Disable container metadata read for NSIO	Disable dedup for small NSIO	Delayed FMD updates for NSIO

QoS For Mixed Workloads

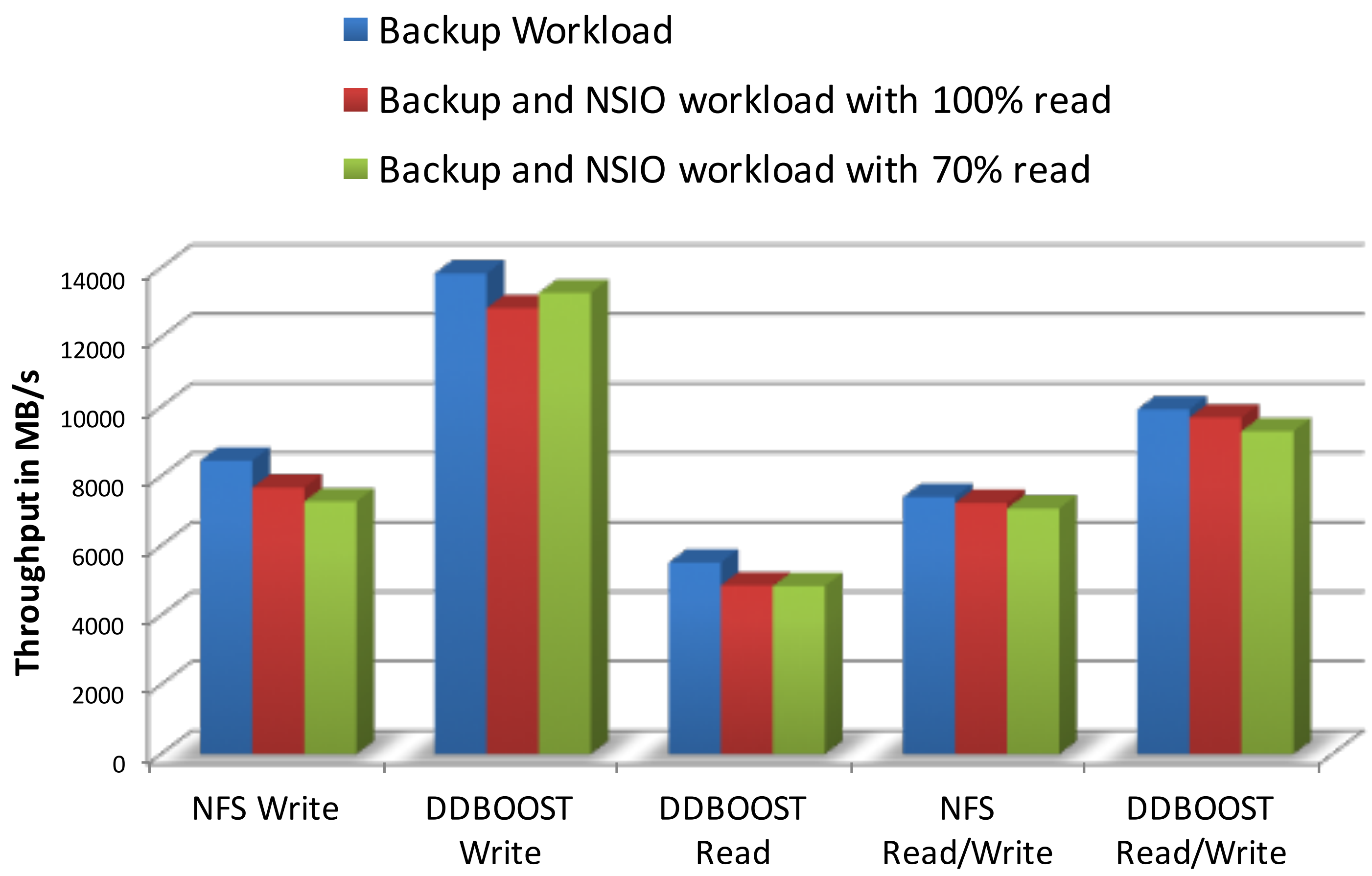
- Tunable shares for non sequential workloads, default is 20%
- CPU scheduling based on least loaded CPU, previously round robin
- Higher priority for random reads compared to writes
- Edge throttling based on feedback from different modules and subsystem health
- Increasing QoS share for NSIO workloads increased NSIO performance in our mixed workload experiments



Mixed Workload Performance

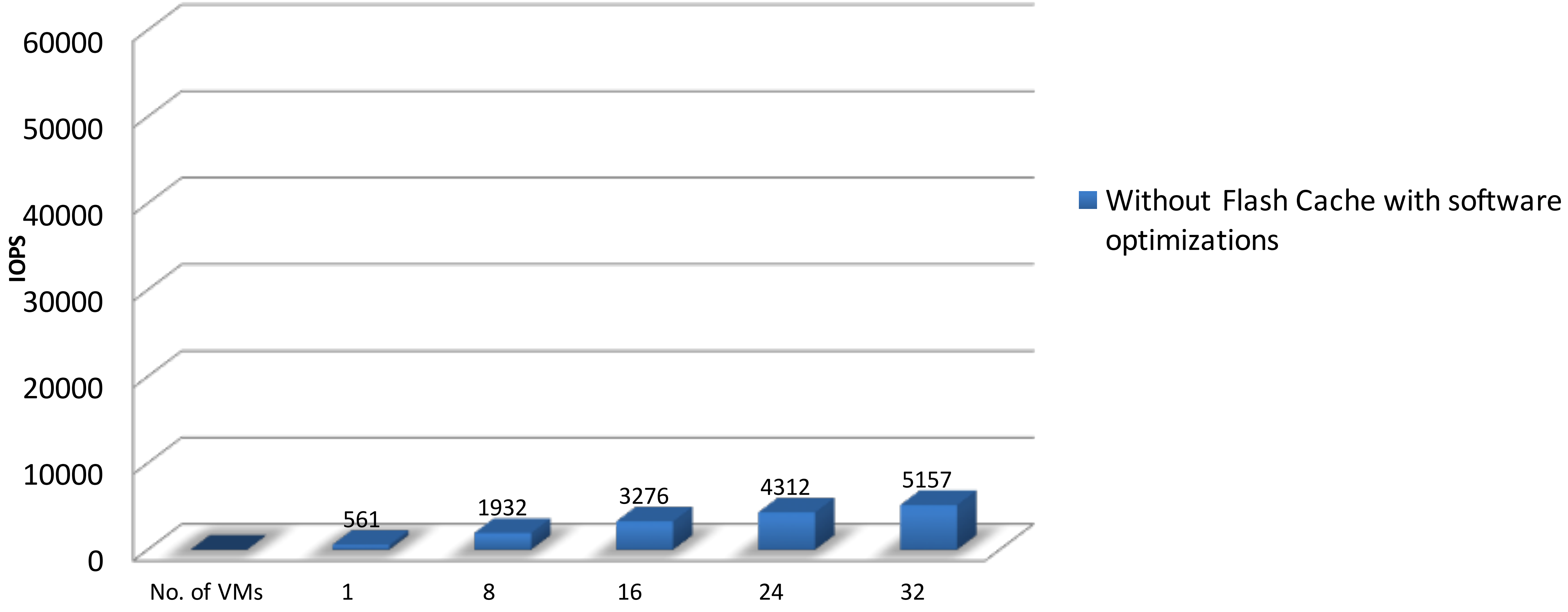
NSIO performance was capped at 10K IOPs

QOS throttle for NSIO set at 10%, Backup/Restore impacted by at-most 10%

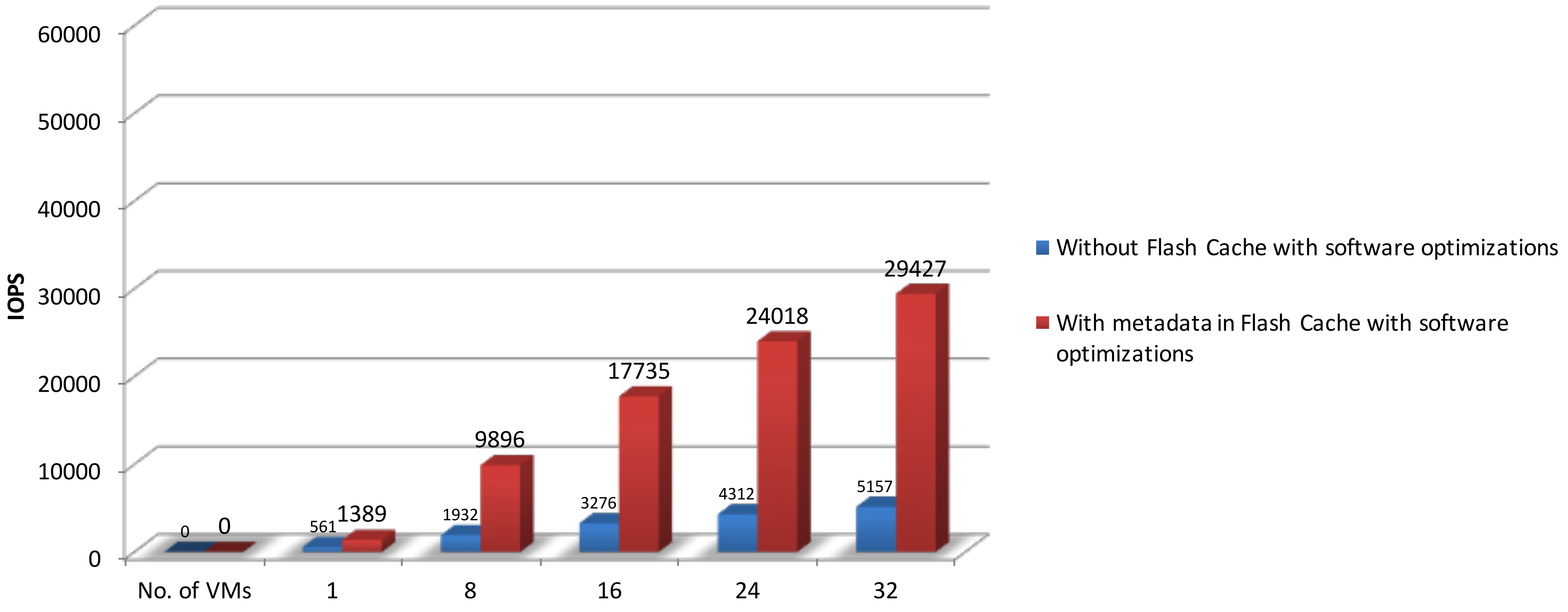


*DDBOOST – Bandwidth Optimized Open Storage Protocol

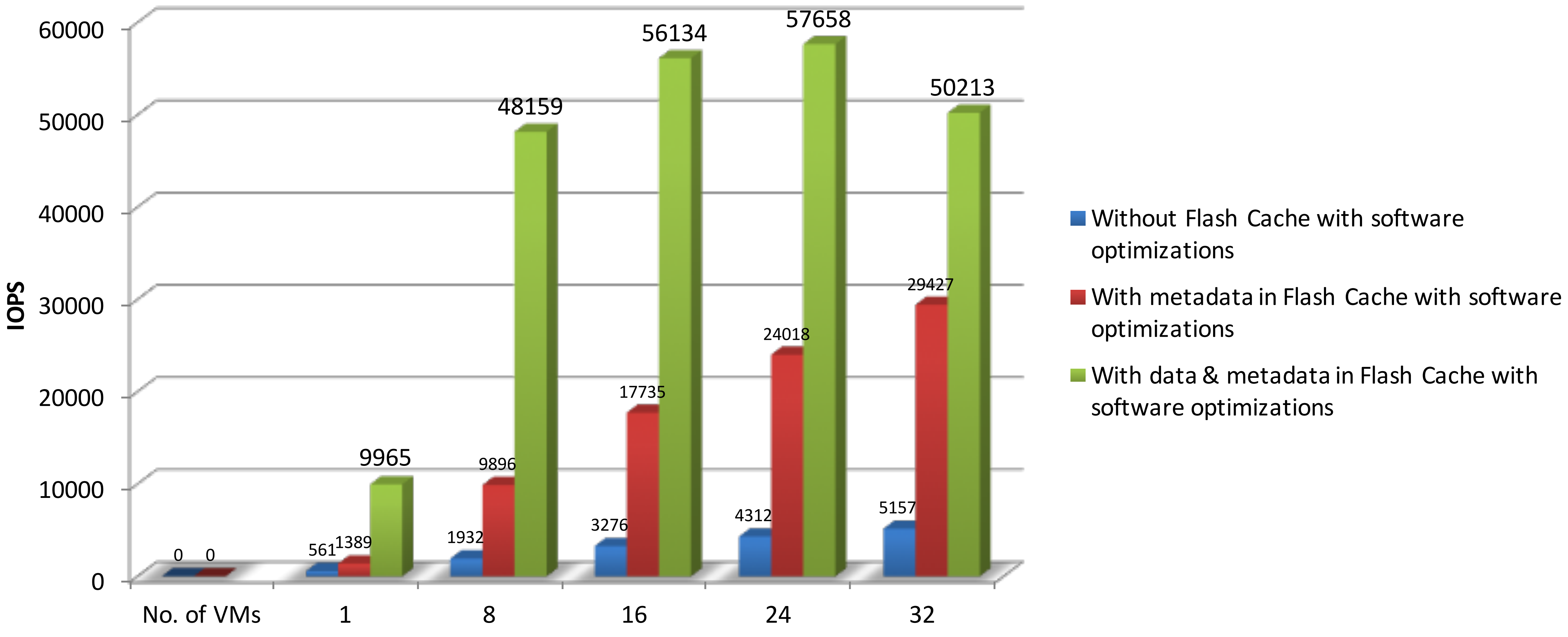
NSIO Performance Evaluation



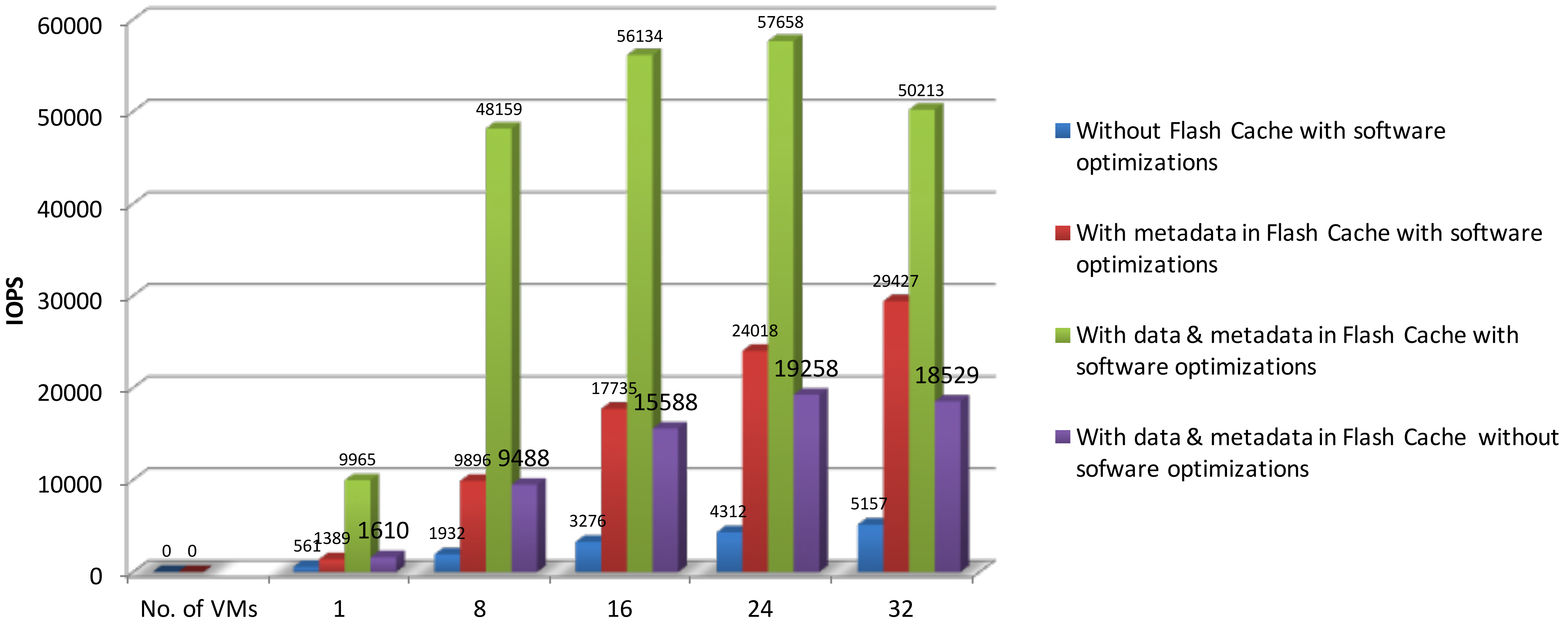
NSIO Performance Evaluation



NSIO Performance Evaluation



NSIO Performance Evaluation



Conclusion/Future Work

Improvements to our software and the addition of SSD caches allow Data Domain to support both new and traditional workloads

- With a NSIO workload, with SSDs for caching metadata, we measured a 5.7x IOPS improvement relative to a system without SSDs
- Adding data cache improved performance by further 1.7x
- Combining SSD caching with software optimizations throughout our system, added an additional 2.7x IOPs increase for NSIO workloads.
- Performance for traditional workloads did not see any degradation

Future work:

Additional SSDs and IOPs based on use-case

QOS/priorities within random workloads

Optimizations to improve CPU and disk utilization as we support higher IOPs

THANK YOU