STORAGE DEVELOPER CONFERENCE

SNIA ■ SANTA CLARA, 2016

Improving Copy-on-Write Performance in Container Storage Drivers

Frank Zhao*, Kevin Xu, Randy Shain

EMC Corp

(Now Dell EMC)

Disclaimer

THE TECHNOLOGY CONCEPTS BEING DISCUSSED AND DEMONSTRATED ARE THE RESULT OF RESEARCH CONDUCTED BY THE ADVANCED RESEARCH & DEVELOPMENT (ARD) TEAM FROM THE EMC OFFICE OF THE CTO. ANY DEMONSTRATED CAPABILITY IS ONLY FOR RESEARCH PURPOSE AND AT A PROTOTYPE PHASE, THEREFORE : THERE ARE NO IMMEDIATE PLANS NOR INDICATION OF SUCH PLANS FOR PRODUCTIZATION OF THESE CAPABILITIES AT THE TIME OF PRESENTATION. THINGS MAY OR MAY NOT CHANGE IN THE FUTURE

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

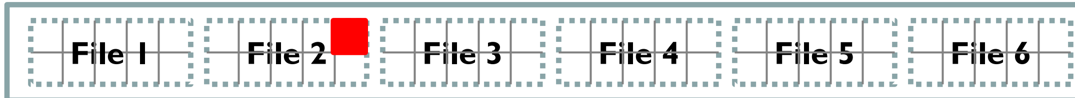
Outline

- Container (Docker) Storage Drivers
- Copy-on-Write Performance Drawback
- Our Solution: Data Relationship and Reference
- Design & Prototyping with DevMapper
- Test Results
  - Launch storm
  - Data access
  - IO heatmap
- Summary and Future Work

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Container/Docker Image and layers

Running container's logical view



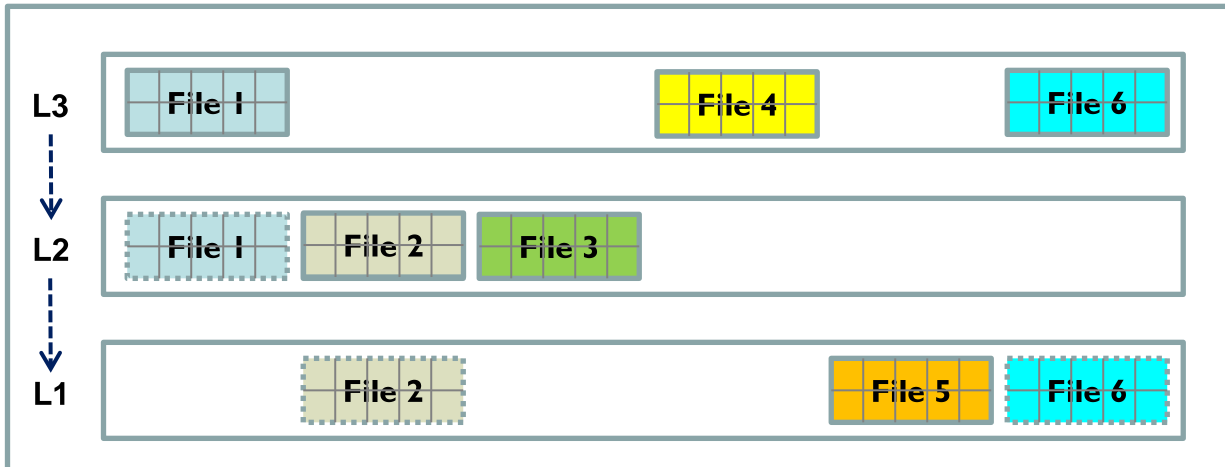


Image and its layers (e.g. Ubuntu 15.04)

- An image references a list of read-only layers or differences (deltas)
- A container begins as a R/W snapshot of the underlying image
- I/O in the container causes data to be copied up (CoW) from image
- CoW granularity may be file (e.g. AUFS), or block (e.g. DevMapper)
- CoW impacts I/O performance

SDC 16
2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Current Docker Storage Drivers

- Pluggable storage driver design
  - Configurable at start of daemon
  - Shared for all containers/images

```
[root@titan-03 ~]# docker info
Server Version: 1.10.3
Storage Driver: devicemapper
 Pool Name: docker--vg-thinpool
 Pool Blocksize: 65.54 kB
 Base Device Size: 21.47 GB
 Backing Filesystem: ext4
```

- Using snapshot and copy-on-write for space efficiency

Driver	Backing FS	Linux distribution
AUFS	Any (EXT4, XFS)	Ubuntu, Debian
Device Mapper	Any (EXT4, XFS)	CentOS, RHEL, Fedora
OverlayFS	Any(EXT4, XFS)	CoreOS
ZFS	ZFS (FS+vol, Block CoW)	Extra install
Btrfs	Btrfs (block CoW)	Not very stable, Docker has no commercial support

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Recommendations From Docker

https://docs.docker.com/engine/userguide/storagedriver/selectadriver/

- NO single driver is well suited to every use-case
- Considering: Stability, Expertise, Future-proofing
  - “Use the default driver for your distribution”

Driver	Has attribute	Good for use case	Bad for use case
AUFS	stable, production-ready, good memory use, smooth Docker experience	PaaS-type work	high write activity
Devicemapper (loop)	stable, in mainline kernel, smooth Docker experience	lab testing	production, performance
Devicemapper (direct-lvm)	stable, production-ready, in mainline kernel, smooth Docker experience		PaaS-type work
Btrfs	in mainline kernel	build pools	high write activity, container churn
Overlay	stable, good memory use, in mainline kernel	lab testing	container churn
ZFS native (ZoL)		PaaS-type work	

Legend: Has attribute — attribute (blue); If good for use case — use case (green); If bad for use case — use case (red)

SDC 16 2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Deployment in Practical Environments

- Likely DM & AUFS are the most widely used
  - Mature, stable
    - DM is in kernel
  - Expertise
  - Widely available
    - CentOS, RHEL, Fedora
    - Ubuntu, Debian
    - …

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

CoW Performance Penalty

DM/AUFS as example

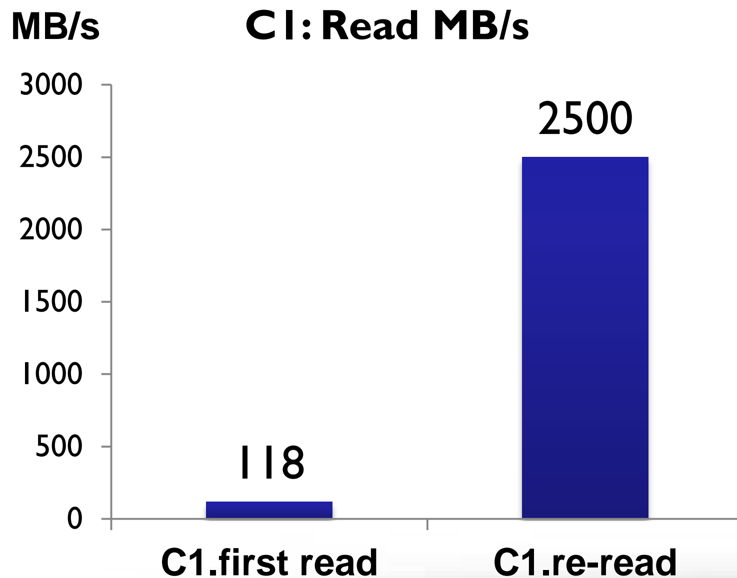
	DevMapper (dm-thin)	AUFS
Lookup	**Good:** Single layer (Btree split)	**Bad**: Cross-layer traverse
I/O	**Good**: Block level CoW **Bad**: Can't share (page) cache	**Good**: Share page cache **Bad**: File level CoW
Memory efficiency	**Bad**: Multiple data copies	**Good**: Single/shared data copy

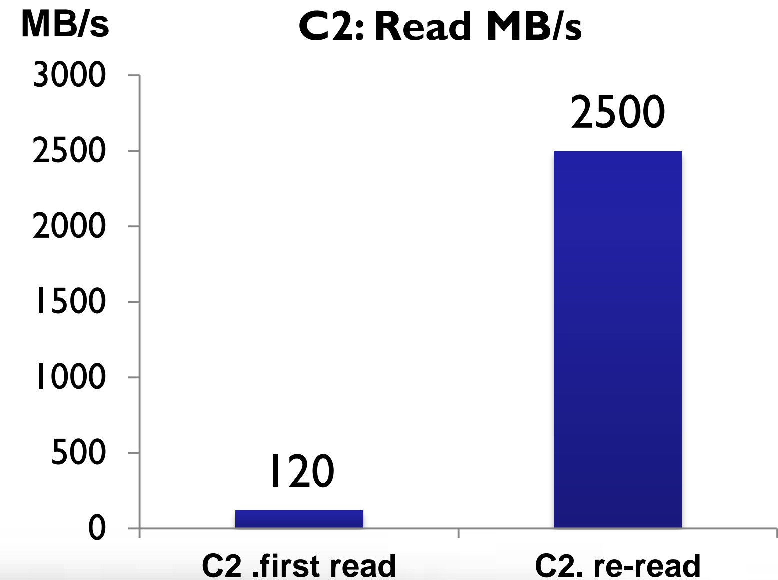
2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

CoW Performance Penalty Example

- Initial copy-up from image to container adds I/O latency
- Sibling containers suffers copy-up again even accessing the same data
  - First read by C1 penalized by CoW
  - Re-read of the same data by C1 satisfied from C1's page cache (**20X faster**)
  - **First read by C2 penalized again due to DM CoW**

(CentOS, HDD, **DM-thin**)





SDC 16

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Our Goals

- Target **dense containers(snaps)** environment
  - Multiple containers from same image, similar workload
- Improve **common CoW** performance
  - Speedup **cross-layer lookup** (akin to AUFS)
  - Reduce **disk IO** (akin to DM)
  - Future, facilitate single mem copy
- Software atop of existing CoW infrastructure
  - **NO extra data cache:** but efficient metadata
  - **Cross-container:** C1→C2, C2→C3, …
  - **Cross-image:** e.g. between Ubuntu and Tensorflow
    - As long as derived from the same base layer
  - Good scalability

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

DRR: Data Relationship & Reference

*Research project *(i.e. not mature enough for production)*

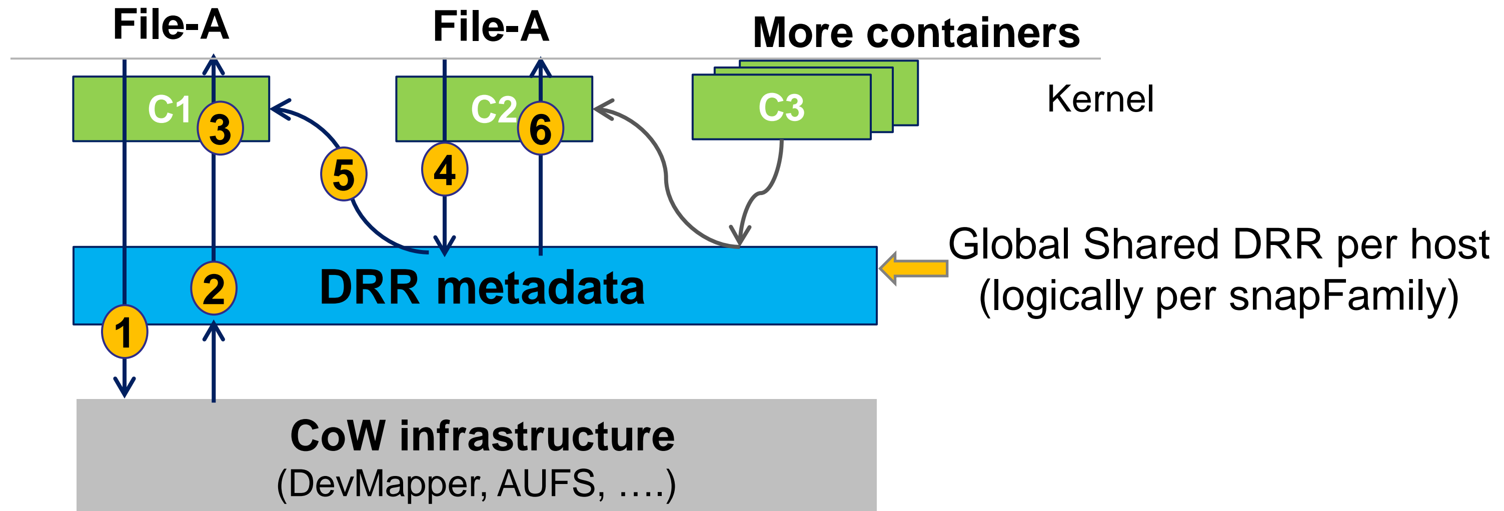
- A metadata describes latest ownership of data
- **Layer relationship**:
  - A tree for base/snap relationship, R/W containers reside at leaf
  - Lookup layer tree: when driver indicates data is shared
  - Update layer tree: when create/delete a layer
    - E.g. image pull, commit changes, or delete image
- **Data access record**: tracks latest IO on shared data
  - Record: {LBN, Len, srcLayer, targetLayer, pageBitmap}

- DRR common operations:
  - **Add/update**: add new record when IO done on shared data
  - **Lookup**: for new IO (read, small write) on shared data to reference from recent valid page cache, instead of disk
  - **Remove**: write IO invalidates and removes the record since data is now owned by that container

SDC 16

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

DRR I/O Diagram

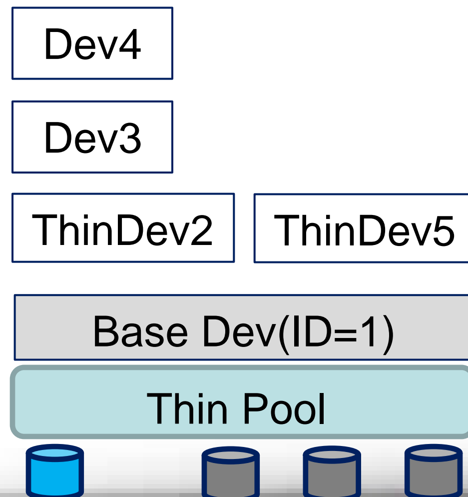


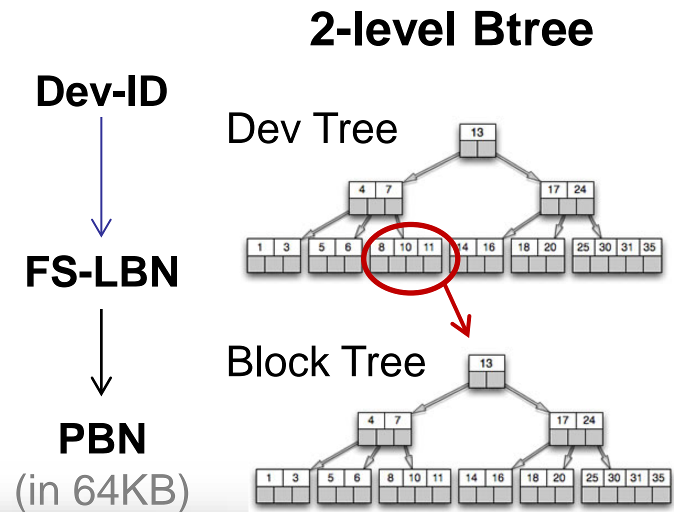
1. C1 first time accesses file data, loads from disk
2. Adds record to DRR metadata after IO
3. Return data to C1
4. C2 to access the same data
5. Look up DRR metadata for valid reference
6. Memory copy data from C1 page cache via normal file interface at corresponding offset, update DRR to reflect most recent reference

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Linux DM-thin

- One of DM target drivers, in kernel for 5Y, mature enough
  - Shared block pool for multiple thin devices
  - Internally single layer (metadata CoW) so no traverse
  - Nice support many snaps in depth! (snap-of-snap)
- Metadata (Btree) and data CoW
  - Granularity: default 64KB chunk

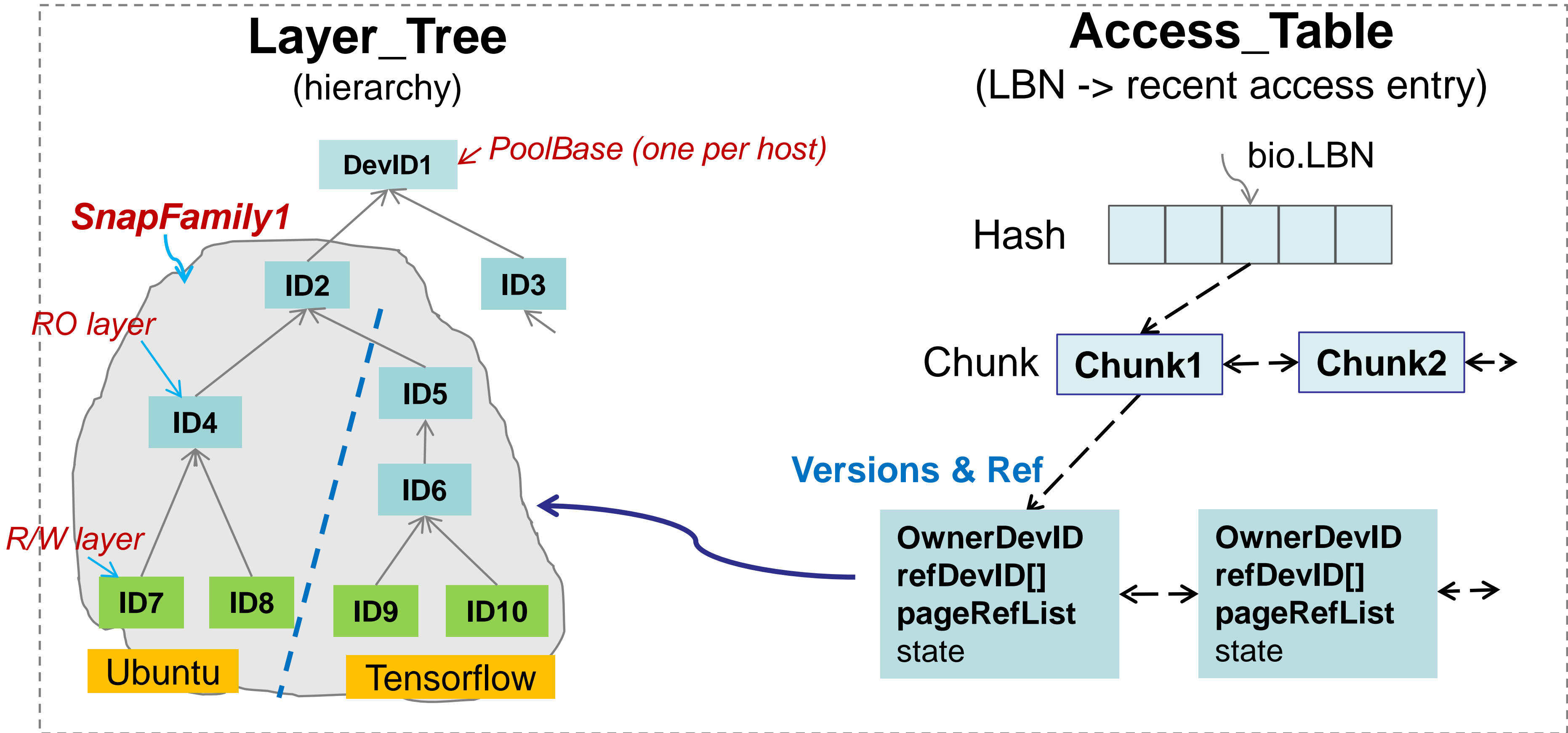




SDC 16

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

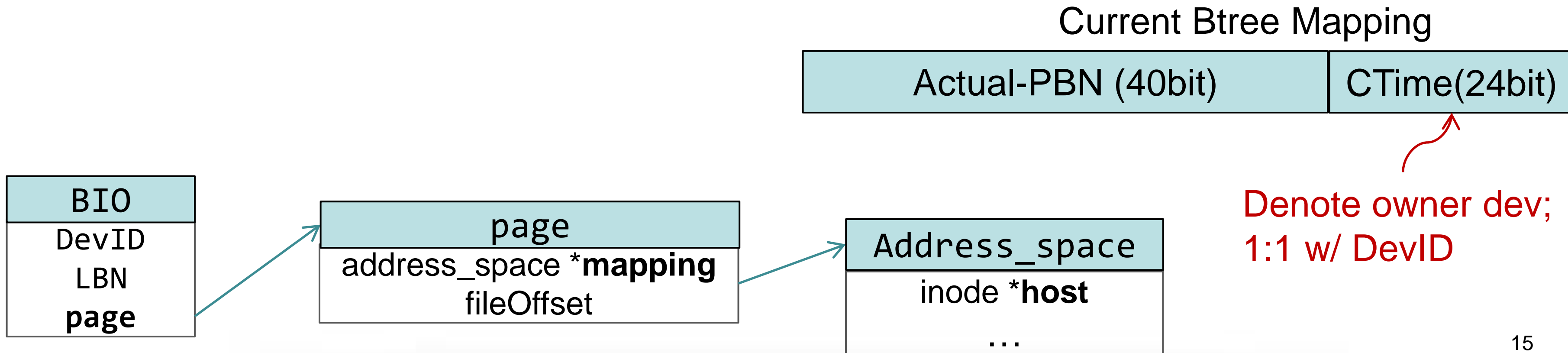
DRR Key Structures: LayerTree, AccessTable



SDC 16
2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

DRR with DM-Thin

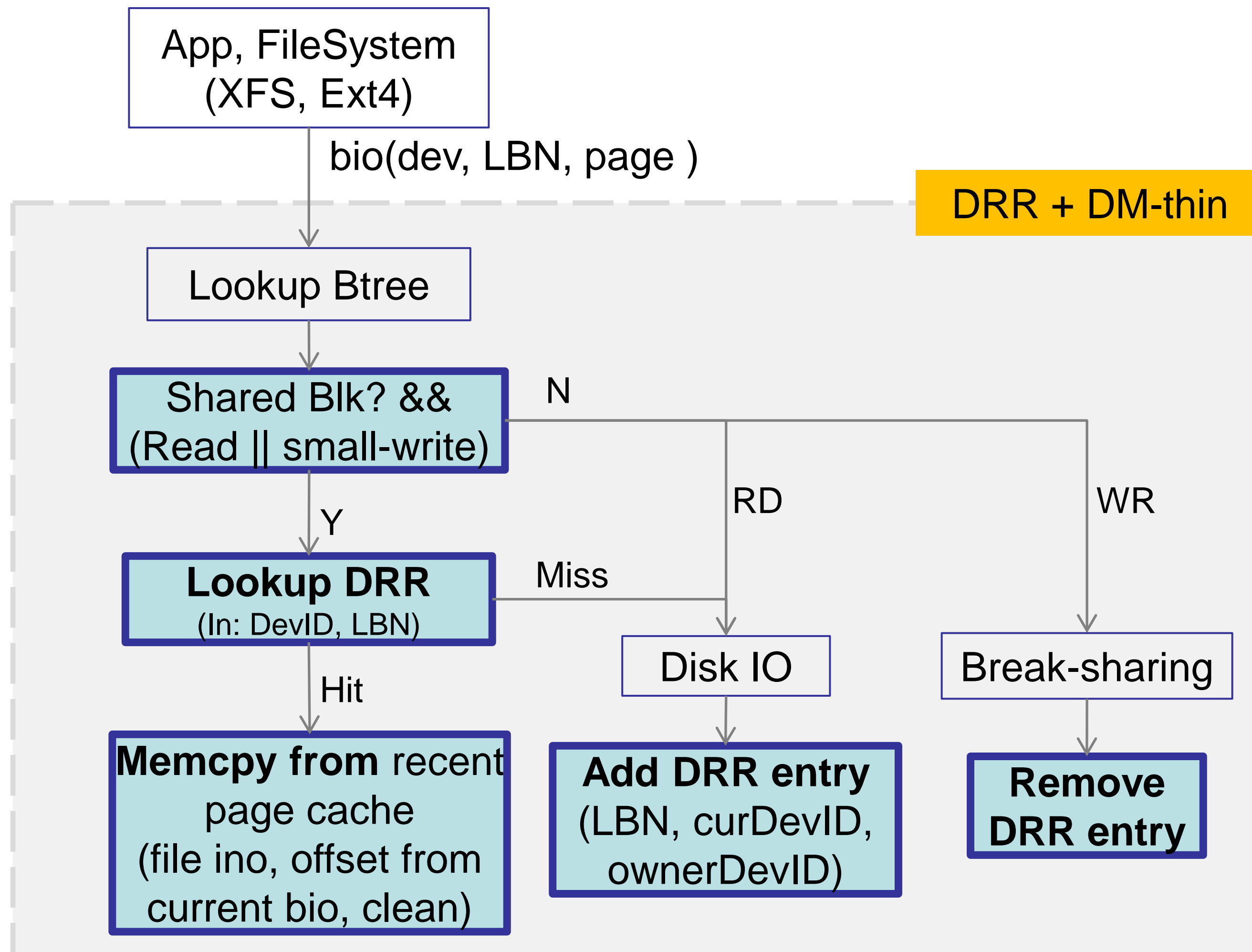
- Transparent to Docker/App
  - Kernel module, between Docker/FS and DM-thin
- No change to core DM-thin CoW essential
  - Leverage existing metadata, hierarchy, addressing …
  - Determine block is shared or not. ←*dm_thin_find_block()*
  - Break block sharing: ← *break_sharing()*
- File data, and shared block only



SDC 16

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

New I/O Flow with DRR



2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

DRR: Preliminary Benchmark of PoC

- Kernel driver integrated w/ DM-thin
  - 1800+ LOC, No Docker code change

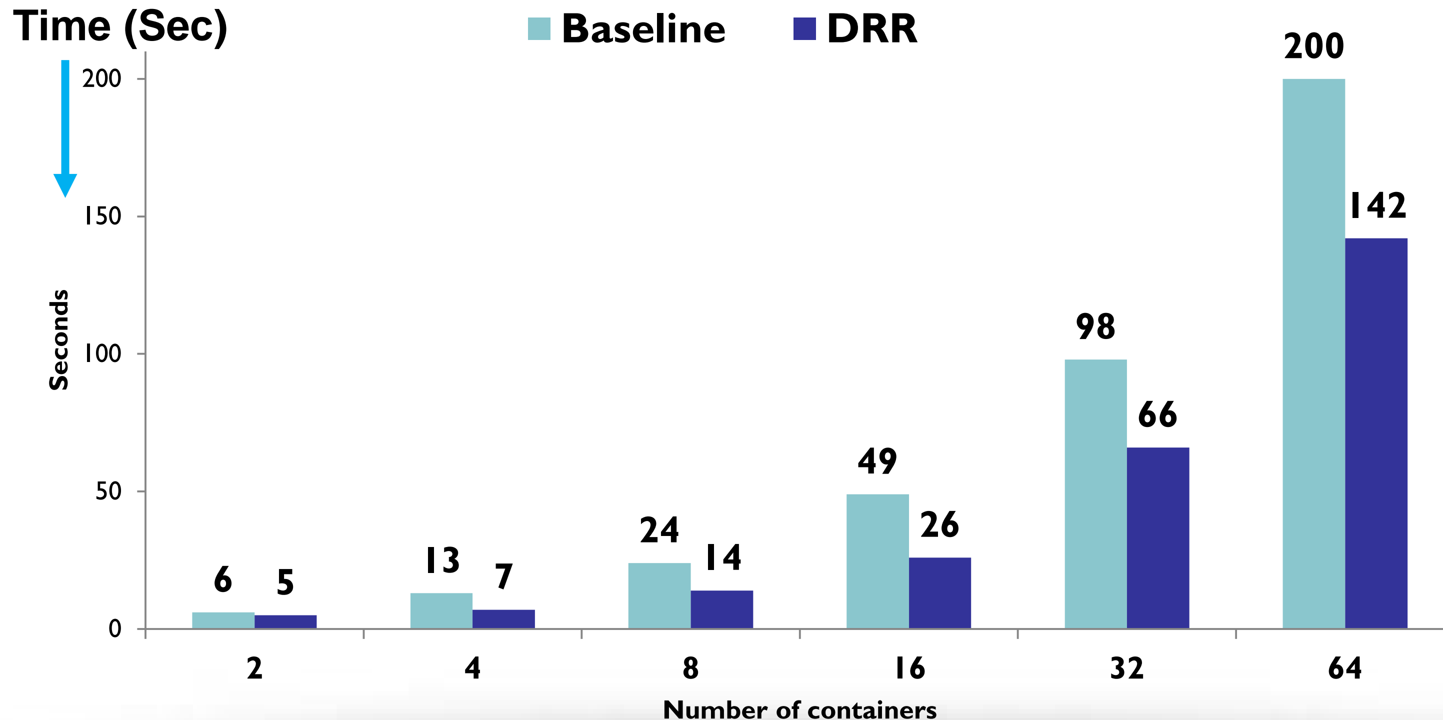
- Env: CentOS7, Docker1.10, DM-thin (64KB chunk), Ext4
  - **HDD** based system
    - E5410 @ 2,33GHz, 8 cores, 16GB DRAM
    - 600GB SATA disk configured as LVM-direct
    - CentOS7, kernel 3.10.0-327
  - **PCIe SSD** based system
    - E5-2665 @ 2.40GHz, 32 cores, 48 GB DRAM
    - OCZ PCIe SSD RM84 (1.2TB) as LVM-direct
    - CentOS7, Kernel 3.10.229

SDC 16

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

DRR: Launch Storm Test (HDD)

- Launch TensorFlow containers
  - Each TensorFlow container needs to load ~104MB trained model & parameters
- Results: up to **47%** faster with DRR

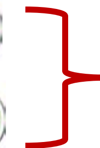


SDC 16

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Container Launch Internals

- Steps to start a container
  1. **Create new thin dev**: ← DM-thin locking/serialization
  2. **Mount Ext4 FS** ← Out of current DRR scope

     FS metadata blocks, not file data
  3. Binary file and libraries:
  4. Config file, parameters data …: 

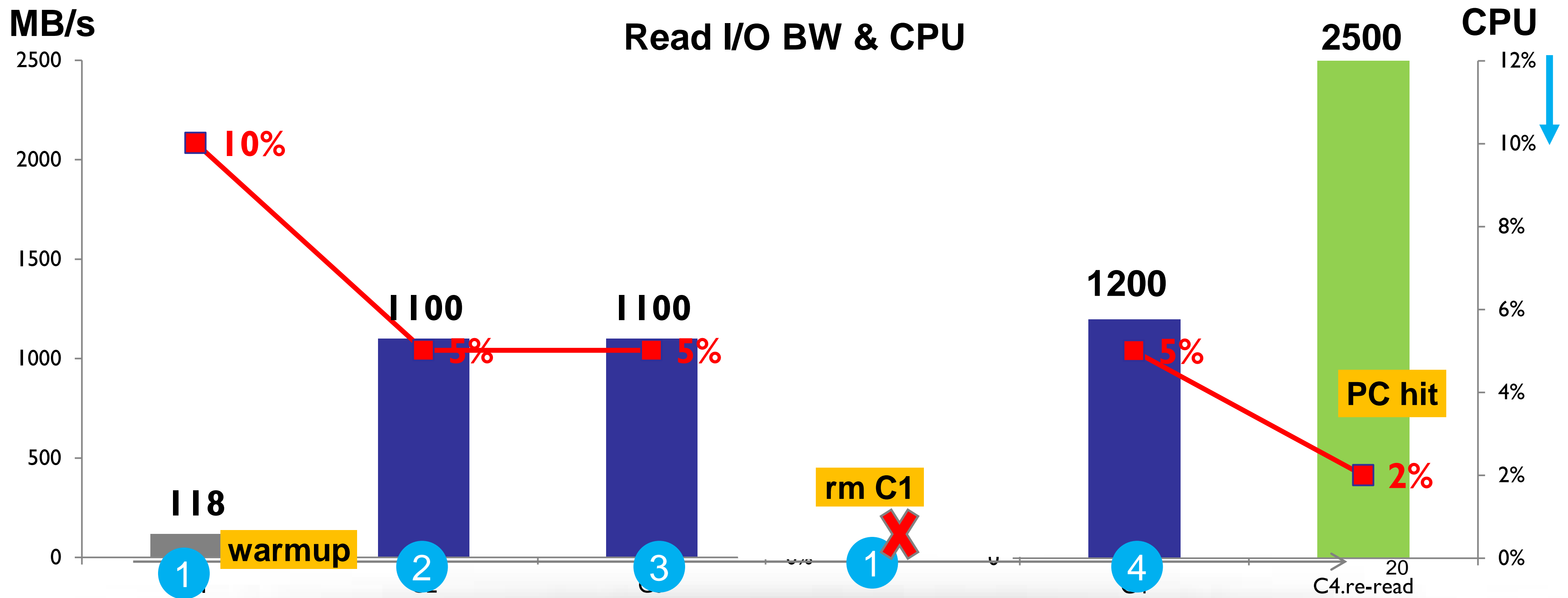
Handled through DRR

SDC 16

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

DRR: Container I/O Test (HDD)

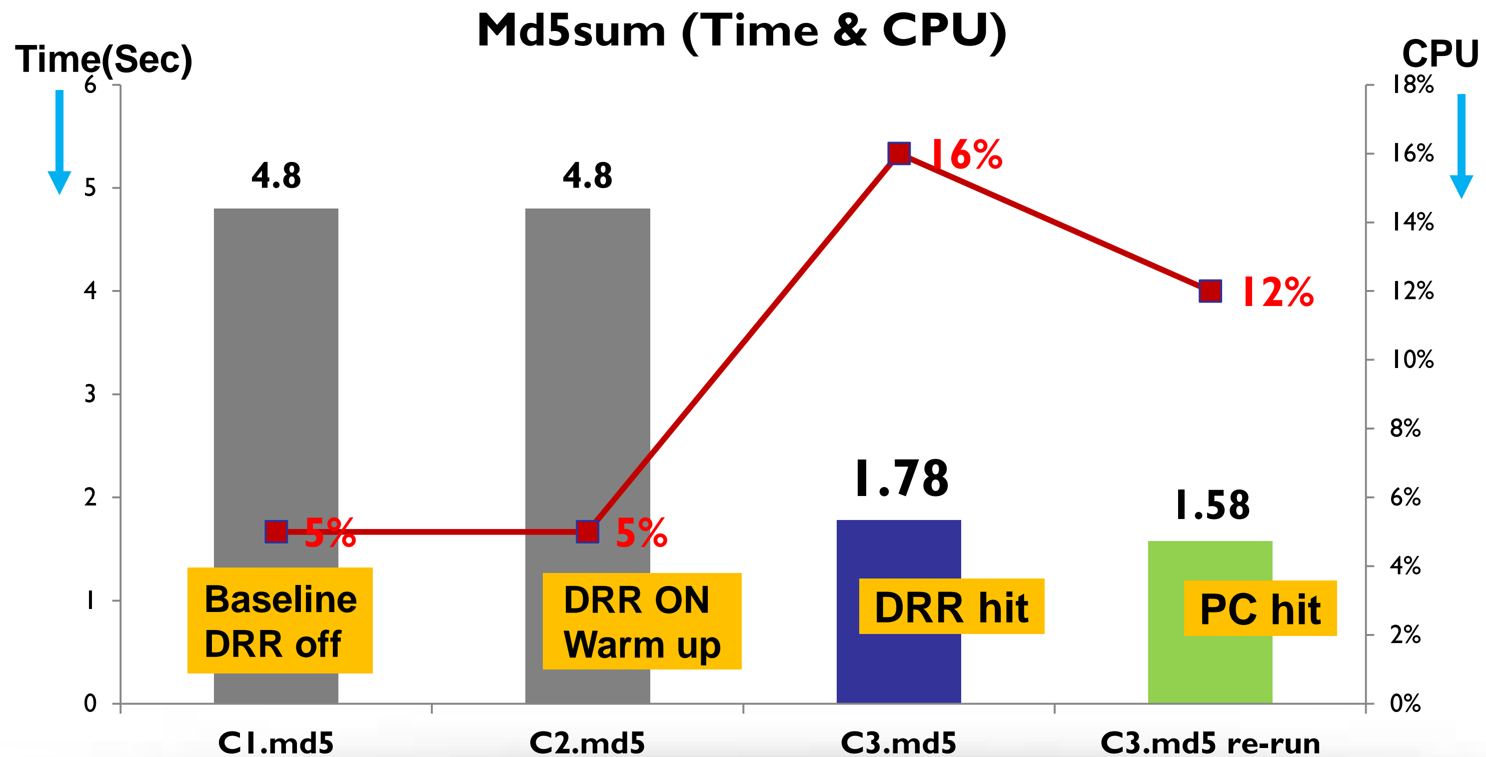
- **10**X faster (~120MB/s -> 1.2GB/s)
  - Launch C1, C2 & C3 in order; then rm C1 and launch C4
  - Issue same IO to read image big file on each container
    - local page cache hit gets 2.5GB/s (1.2/2.5 = 48%, CPU: 2%)



2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

DRR: Computing + IO test (HDD)

- **2.7**X faster in execution time: md5sum <big file>
  - Higher CPU (+4% vs. local PC hit, due to block bio + DRR)

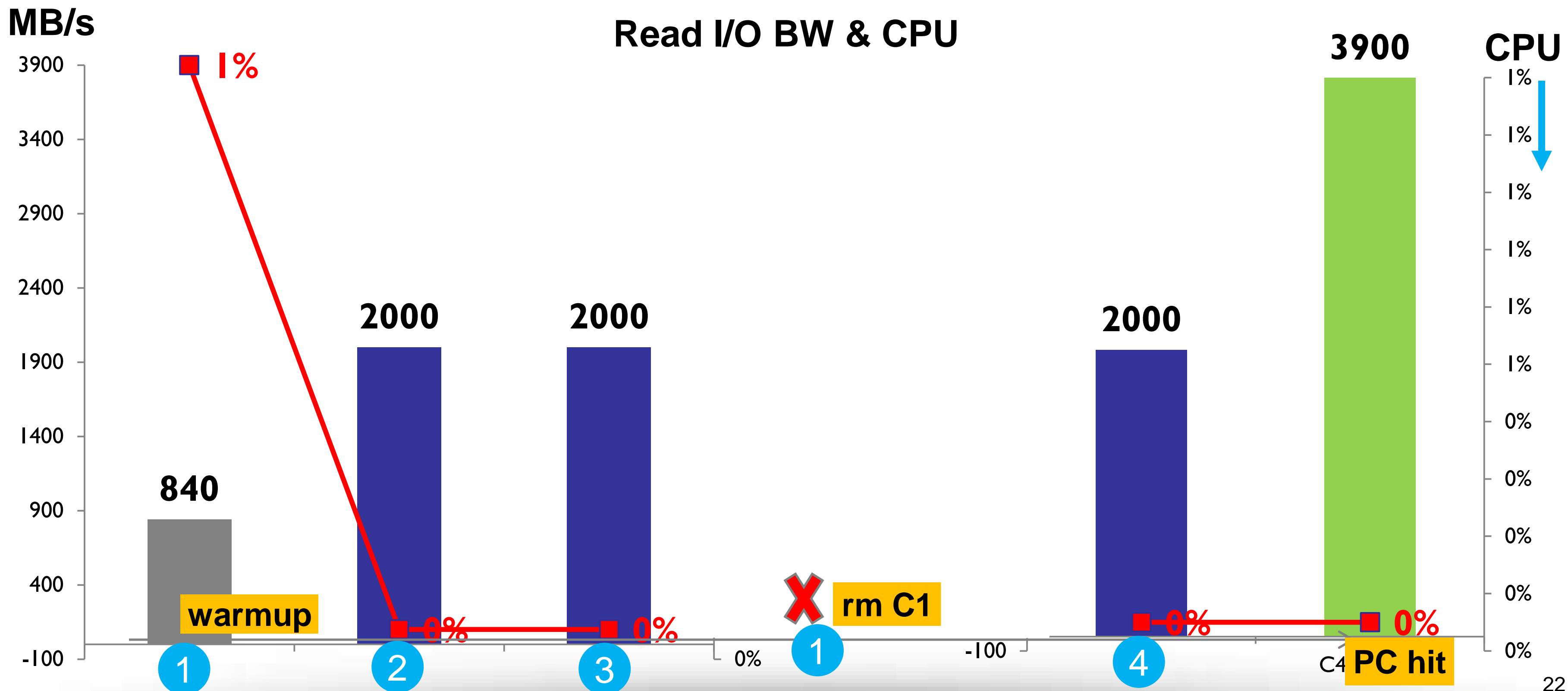


SDC 16

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

DRR: Container I/O Test (PCIe SSD)

- **238**% faster (840MB/s -> 2.0GB/s)
  - Same as HDD test steps; Local PC hit is 3.9GB/s (2.0/3.9= 51%)

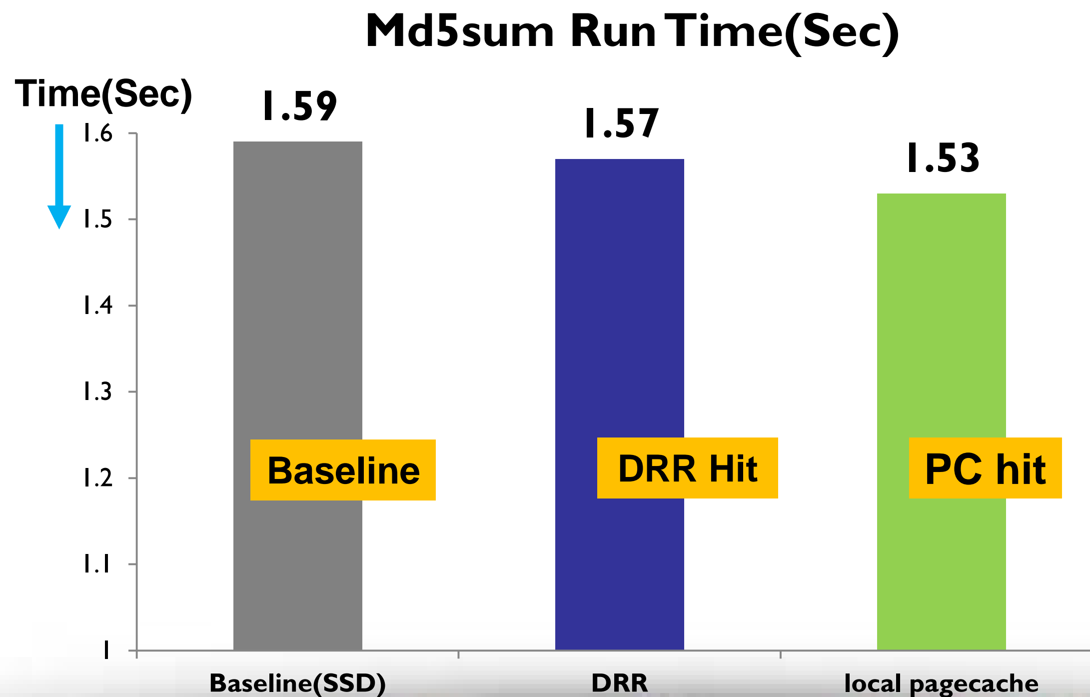


SDC 16

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

DRR: launch storm & md5sum (PCIe SSD)

- Slight improvement
  - Non-IO factors i.e., locking, processing etc
  - PCIe SSD read is fast!



SDC 16

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

DRR: I/O Heatmap utility

- **Heatmap to trace I/O cross-layer,** w/ and w/o DRR
  - Stats collected during TensorFlow launches on SSD system

Src -> Tgt	Total IO	Read
[271 -> 1]	113	110
[271 -> 6]	3	3
[271 -> 9]	26	10
[271 -> 11]	3	0
[271 -> 12]	8	1
[271 -> 13]	2484	2440
[271 -> 14]	252	233
[273 -> 1]	44	42
[273 -> 6]	5	5
[273 -> 9]	17	8
[273 -> 11]	3	0
[273 -> 12]	5	1
[273 -> 13]	818	807
[273 -> 14]	116	100
[273 -> 271]	28	6

Without DRR

2/3 disk IO reduction

With DRR

SDC 16
2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Discussion-1

- DRR CPU footprint: -5% ~ +10% vs. disk I/O
  - Or +3~+4% vs. page cache hit
- DRR memory footprint
  - Not related to number of containers (they're leaf)
    - but is related to **layer depth (shared block versions)**
  - Estimation: hot chunk * versions(layer-depth) * ref# (layer-depth)
    - 50GB image,10% hot; 5~10 depth w/ containers, then 100~400MB memory
  - If DRR 100% miss: perf -2.5% (HDD) ~ -6.7% (PCIe)

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Discussion-2

- Benefits from **dense container locality**
  - Many duplicate containers, similar workload/pattern
  - The more running containers, the less **shared footprint**
  - Auto warm up **after reboot, no big difference**
  - Access from most recent one(s), **balanced and recursive**
    - C1→C2, C2→C3, C3→C4, …
- Benefits small-writes (the “Copy” op)
  - Skip adding DRR entry since data to dirty soon

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Discussion-3

- I/O path
  - Before: C1→FS1: page-cache miss ------------->disk
  - Now: C2→FS2: page-cache miss ----DRR--->FS1
  - May lookup DRR before issue BIO to shorten I/O path
- DRR Current Implementation:
  - For file data only, page-aligned IO
  - In-mem only; not persistent
  - Next: LRU replacement? Part hit + part miss?
- Modify Docker driver API?
  - To facility control and management

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Outlook

- More comprehensive tests and tuning
- Prototyping with AUFS
  - Reduce cross-layer lookup
- Think about memory deduplication with DM?
  - Offline & Brute force way: **KSM** (Kernel Same-page Merge, for VM/VDI)
    - DRR to provide fine-grained source/target memory deduplication info
  - Inline & graceful fashion: Global FS data cache
    - Gap to enterprise/commercial FS (global data cache/dedup)

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Summary

- Targeted for dense container deployment environment
- DRR: a scalable solution to address common CoW drawback
  - Layer hierarchy, Access history
  - No extra data cache, cross-container locality even cross-image
    - Good scalability (with layer depth rather than container#)
  - Transparent, integrates w/ various CoW w/o significant change
  - Promising especially for HDD and IO heavy case
- PoC with Linux DM
  - Up to 10X in HDD, 2+X in PCIe SSD
  - Open new possibility
    - VM/VDI solutions → Container era?

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.

Thank you!

Any further feedbacks, please contact:

Junping.zhao@emc.com or ZhaoJP@gmail.com

SDC 16

2016 Storage Developer Conference. © EMC Corp All Rights Reserved.