DADI Block-Level Image Service for Agile and Elastic Application Deployment

Huiba Li, Yifan Yuan, Rui Du, Kai Ma, Lanzheng Liu and Windsor Hsu
Alibaba Group

The Problem

- Container deployment (cold startup) is slow
  - *Long-tail latency reaches 10s of minutes*
- The essential reasons are image downloading and unpacking
  - *Only 6.4% [Slacker] of the image is used for startup*
  - *A regression to a decade ago, when VM images were also downloaded to hosts*
- P2P downloading [Dragonfly, Kraken, Borg, Tupperware, FID] is not enough
  - *Deals with only half the reason*
  - *Little effect for small clusters*
- Slimming the images [DockerSlim, Cntr] is not universal
  - *Hard to automatically find all dependencies for all applications*
  - *Hard to support ad-hoc operations*

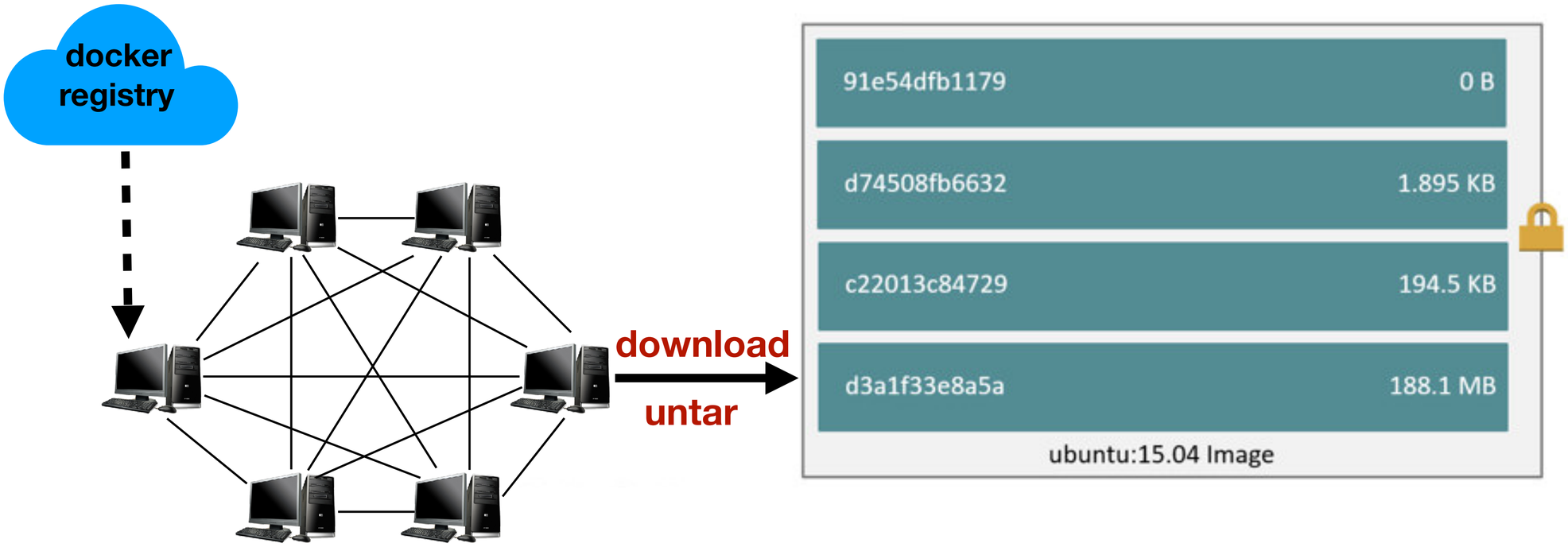
Remote Image

- is the trend
  - *[CRFS, Teleport, CernVM-FS, Slacker, Wharf, CFS, Cider]*
  - *Optionally with P2P transfer for large clusters*
- Container image (tarball) is, however, NOT viable for remote image
  - *Designed for unpacking, not seekable*
  - *Hard to support advanced features, such as xattr, cross-layer reference, etc.*
  - *We'd better to design a new one*
- Type of image
  - *File-system-based image?*
  - *Block-device-based image?*

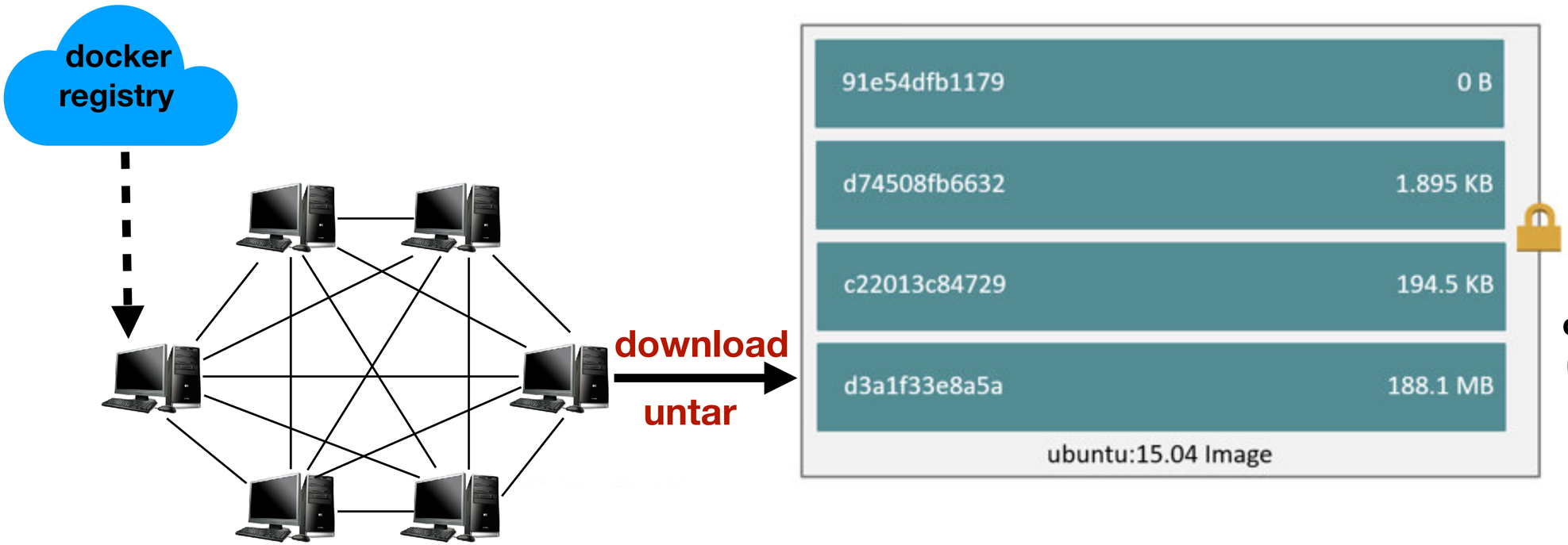
Type of Image: Block!

	Features	Existing Sys	Complexity	Universality	Security	Overall
Block-Device-Based	• Work together with a regular file system, such as ext4 • Viable for container, secure container and virtual machine	Cider (based on Ceph; no layering format.)	**Low** stability↑ optimization↑ advanced features↑	App can choose a best-match file system, e.g. NTFS, and pack it into the image as a dependence.	small attack surface	need the courage to walk alone (almost) TODO: layering
File-System-Based	• Provides a file-system interface directly • “Natural” extension of container image • Less mental friction (due to inertia and following the crowd)	CRFS, Teleport, CernVM-FS, Slacker, Wharf, CFS	**High** stability↓ optimization↓ advanced features↓	Fixed features; may not match all applications. (e.g. a Windows container on a Linux host)	large attack surface	Technical advantage is insignificant.

Background: Layered Image of Container

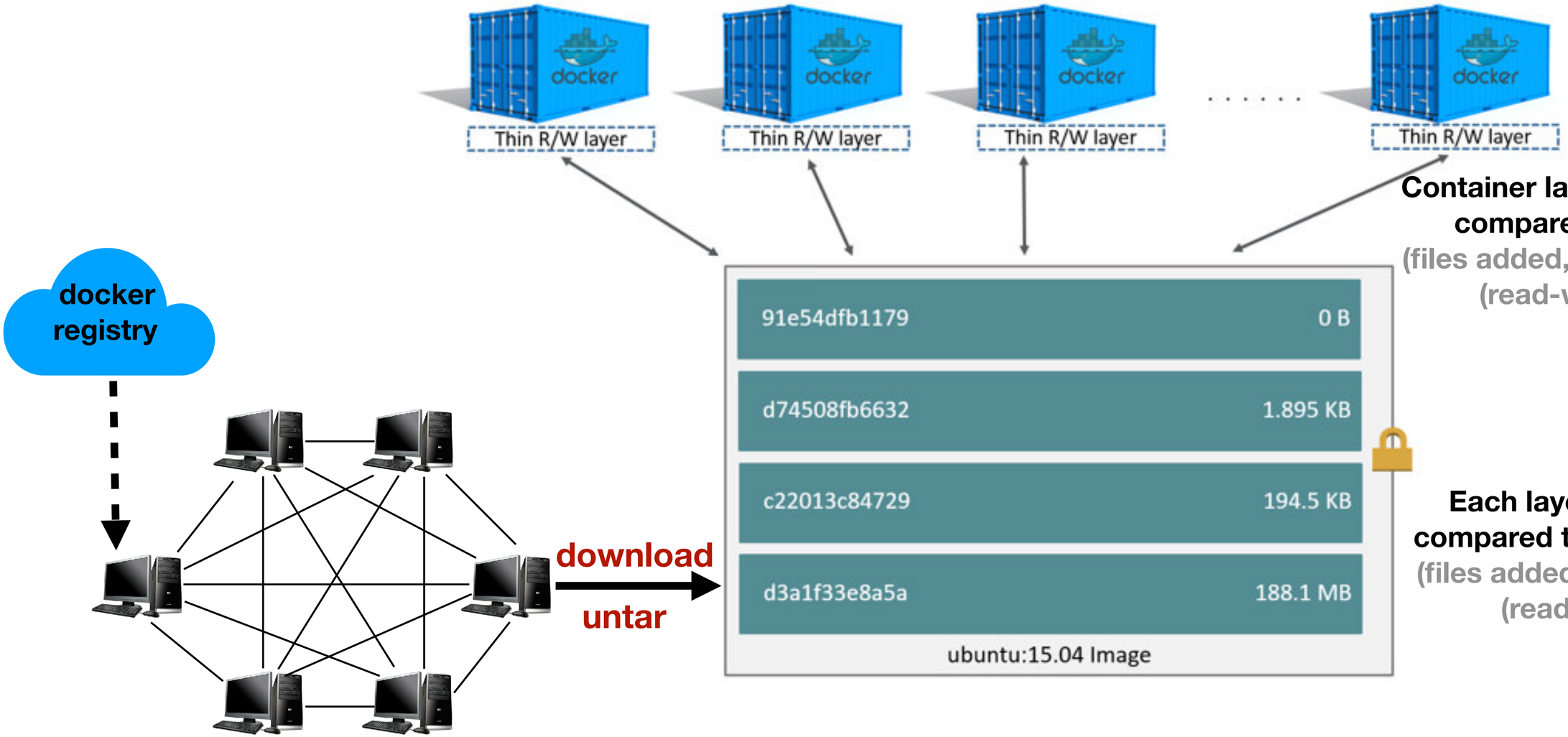


Background: Layered Image of Container



Each layer is a change set compared to the previous state
(files added, modified, deleted)
(read-only, shared)

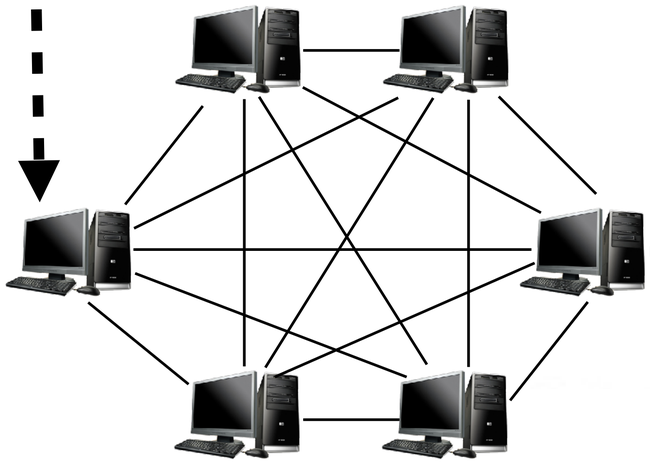
Background: Layered Image of Container



Background: Layered Image of Container

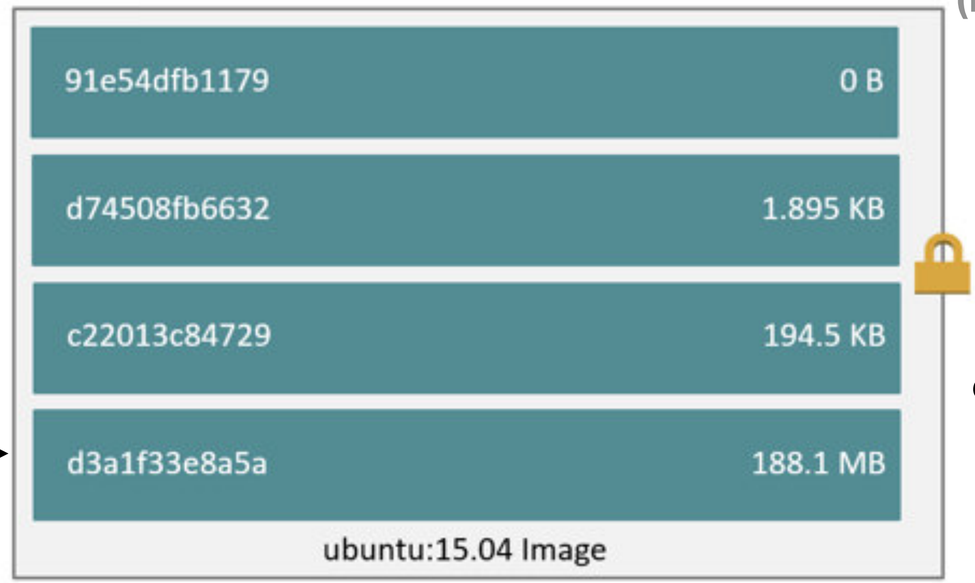
Usually the layers are stored in separate directories, and a merged view is created with a kernel module *overlayfs*.

Container layer is a change set compared to the image
(files added, modified, deleted)
(read-write, private)



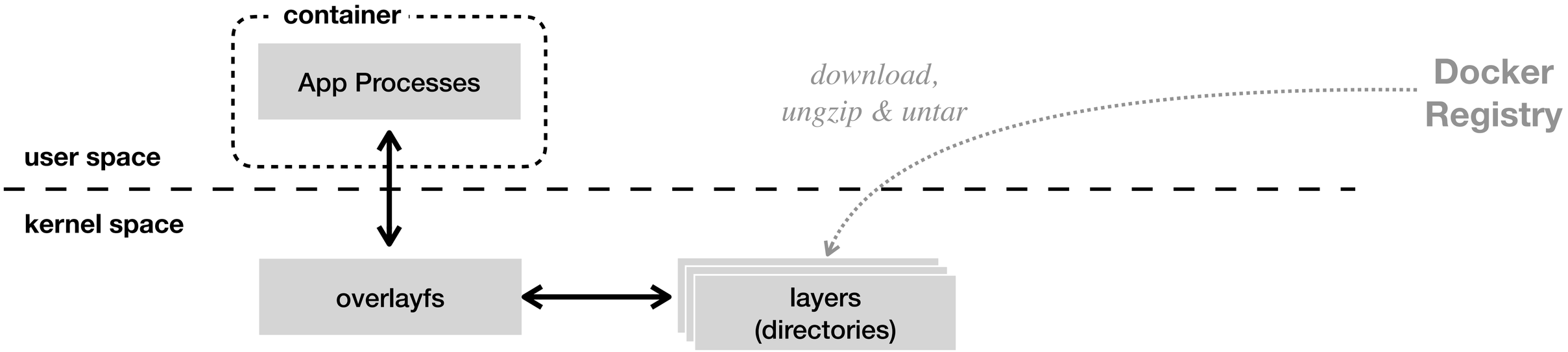
download

untar



Each layer is a change set compared to the previous state
(files added, modified, deleted)
(read-only, shared)

Background: I/O Path



DADI Remote Image

- A layered image format
  - *based on virtual block device*
  - *work together with a regular file system, e.g. ext4*
  - *a general solution for container ecology*
- Compression
  - *and seekable decompression (online)*
- Scalability
  - *peer-to-peer (P2P) transfer*

DADI Remote Image

- A layered image format - - - - - - - - - - - - - - - → Overlay Block Device
  - *based on virtual block device*
  - *work together with a regular file system, e.g. ext4*
  - *a general solution for container ecology*
- Compression
  - *and seekable decompression (online)*
- Scalability
  - *peer-to-peer (P2P) transfer*

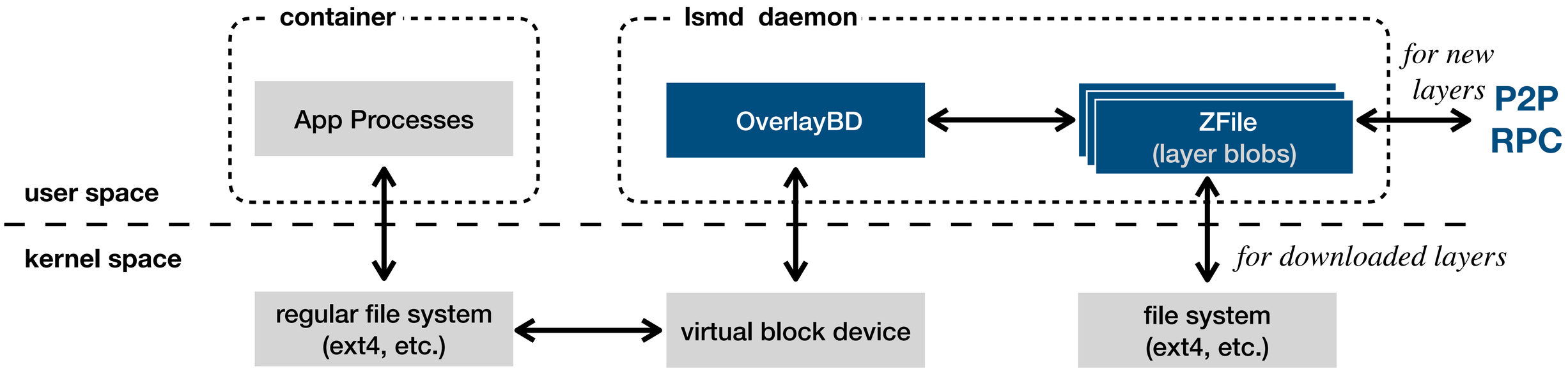
DADI Remote Image

- A layered image format ⇢ Overlay Block Device
  - *based on virtual block device*
  - *work together with a regular file system, e.g. ext4*
  - *a general solution for container ecology*
- Compression ⇢ ZFile
  - *and seekable decompression (online)*
- Scalability
  - *peer-to-peer (P2P) transfer*

DADI Remote Image

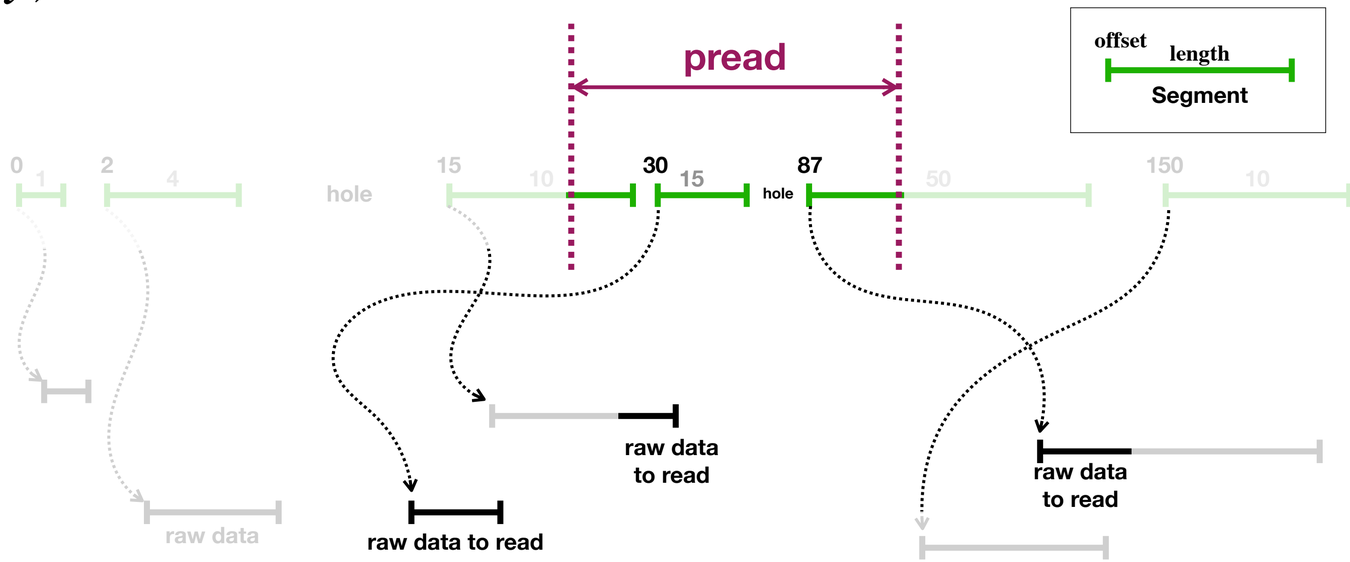
- A layered image format ⇢ Overlay Block Device
  - *based on virtual block device*
  - *work together with a regular file system, e.g. ext4*
  - *a general solution for container ecology*
- Compression ⇢ ZFile
  - *and seekable decompression (online)*
- Scalability ⇢ P2P on-demand read in a tree-structured topology
  - *peer-to-peer (P2P) transfer*

DADI I/O Path

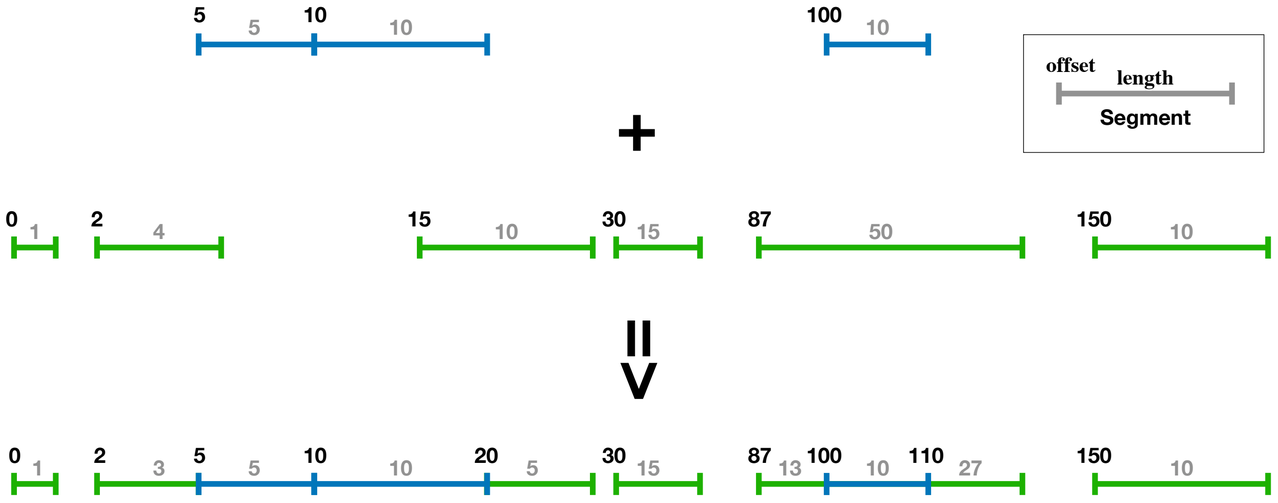

container
App Processes
lsmd daemon
OverlayBD
ZFile
(layer blobs)
for new
layers
P2P
RPC
user space
kernel space
for downloaded layers
regular file system
(ext4, etc.)
virtual block device
file system
(ext4, etc.)

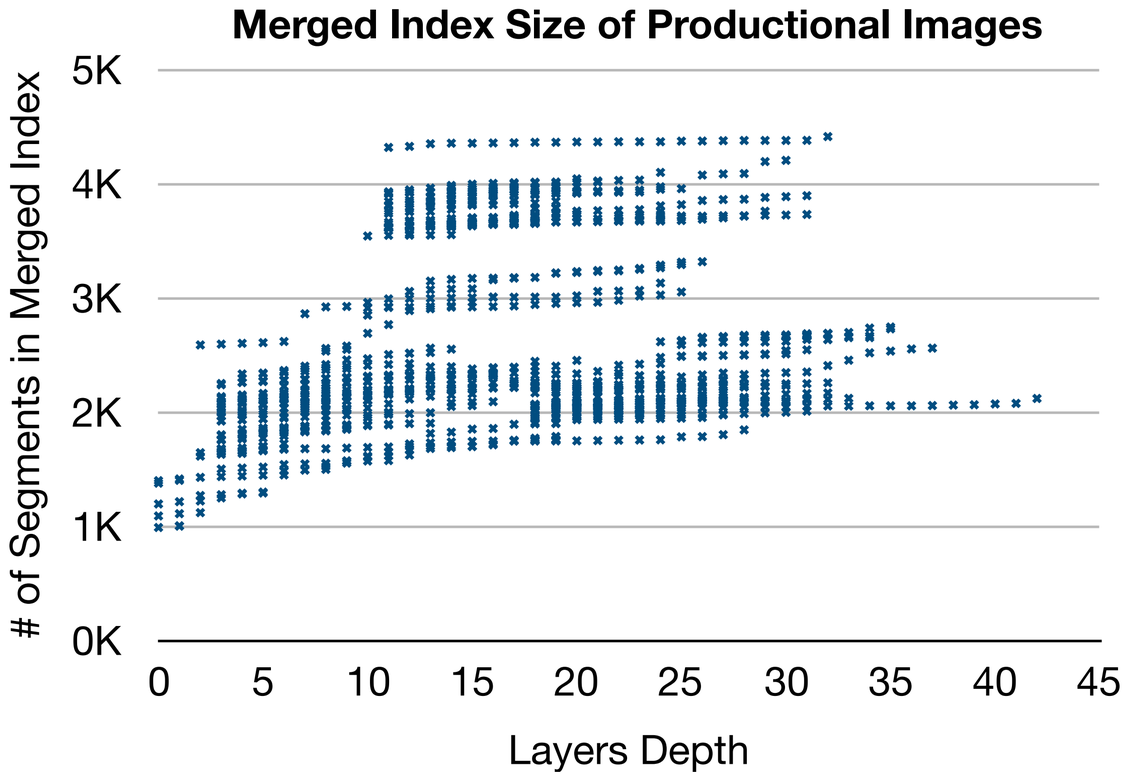
Overlay Block Device

- Each layer is a change set of overwritten blocks
  - *no concept of file or file system*
  - *512 bytes block size (granularity)*
- An index for fast reading
  - *variable-length entries to save memory by combining*
  - *non-overlapping entries sorted by logical offsets*
  - *range query by binary search*



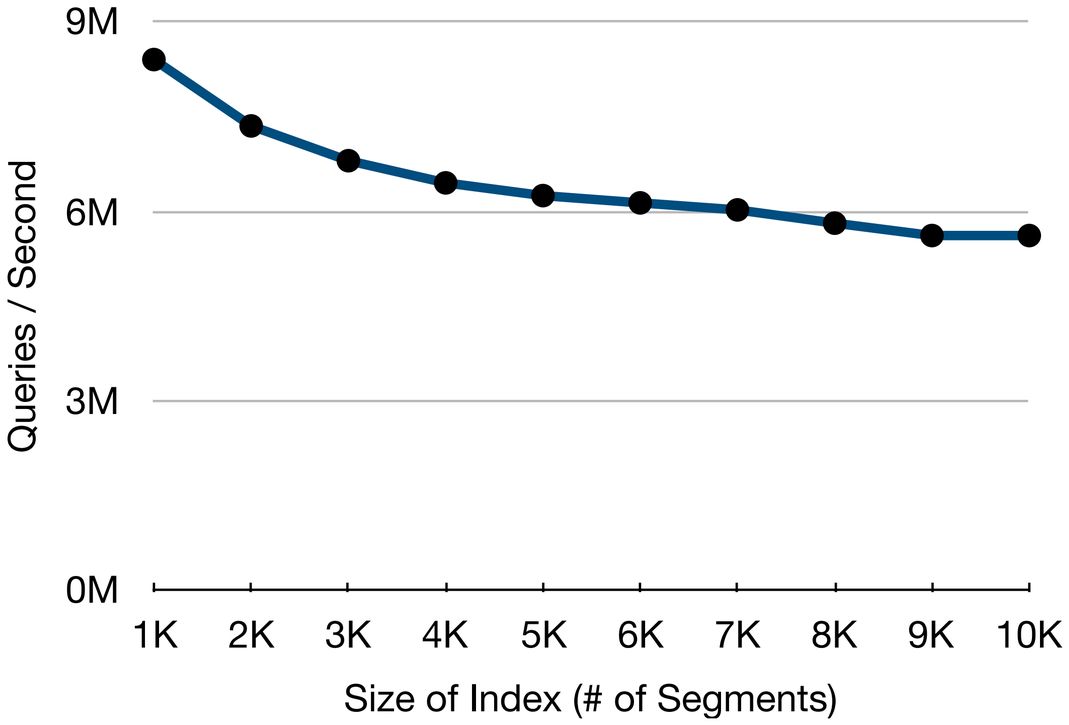
Index Merge

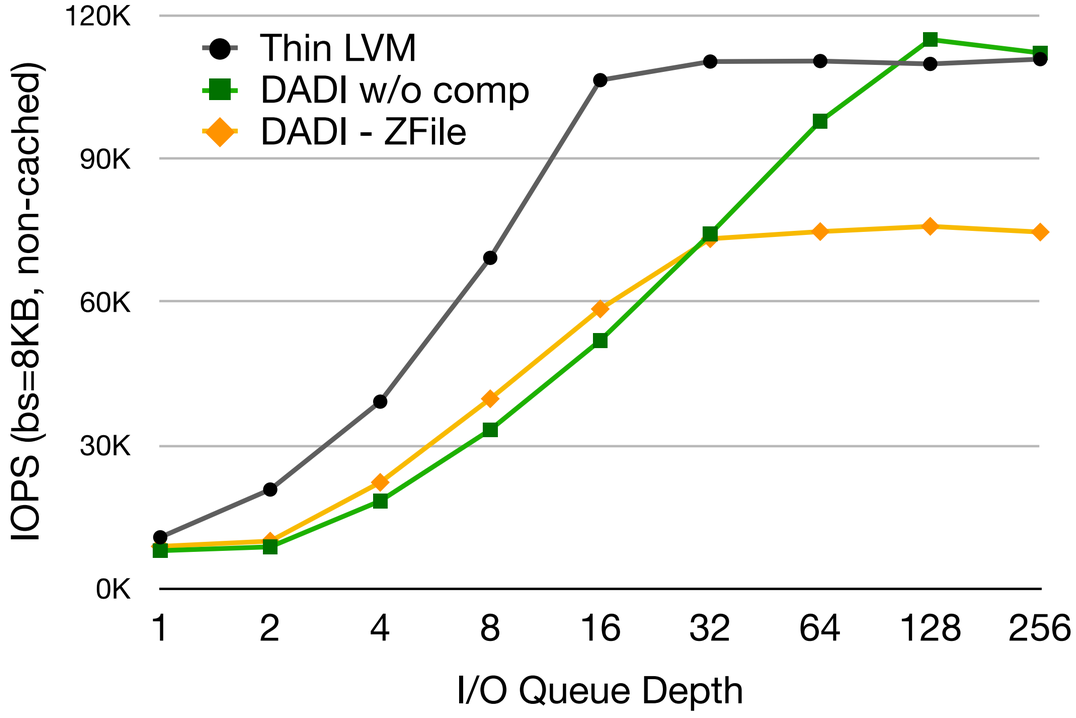




4.5K * 16 bytes = 72KB

Index Performance

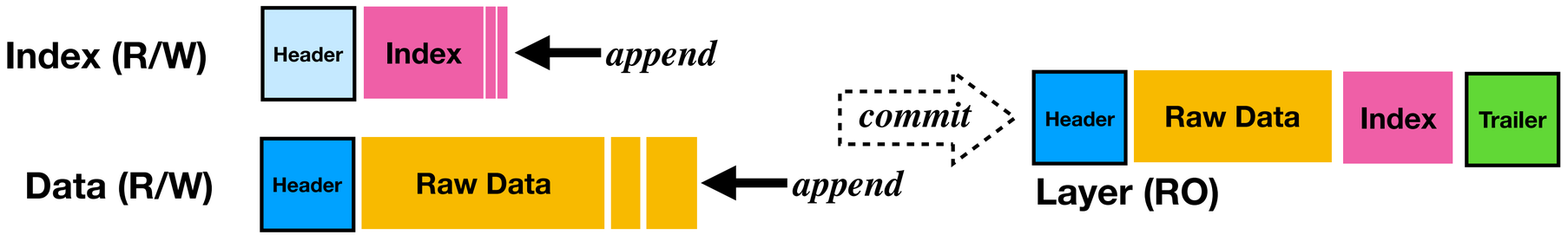




> 6M QPS for productional images

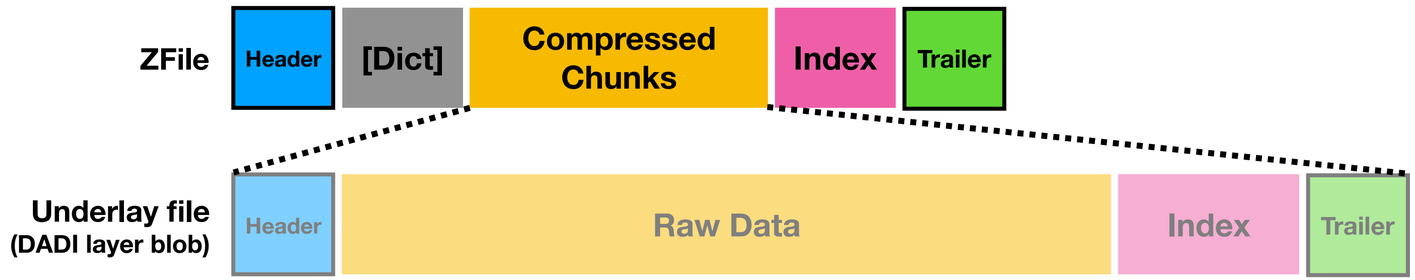
Writable Layer

- Log-structured design
  - *appending index and raw data to separate logs*
- Maintaining an in-memory index
  - *red-black-tree*
- Commit only useful data blocks (in offset order)
  - *combine index entries*



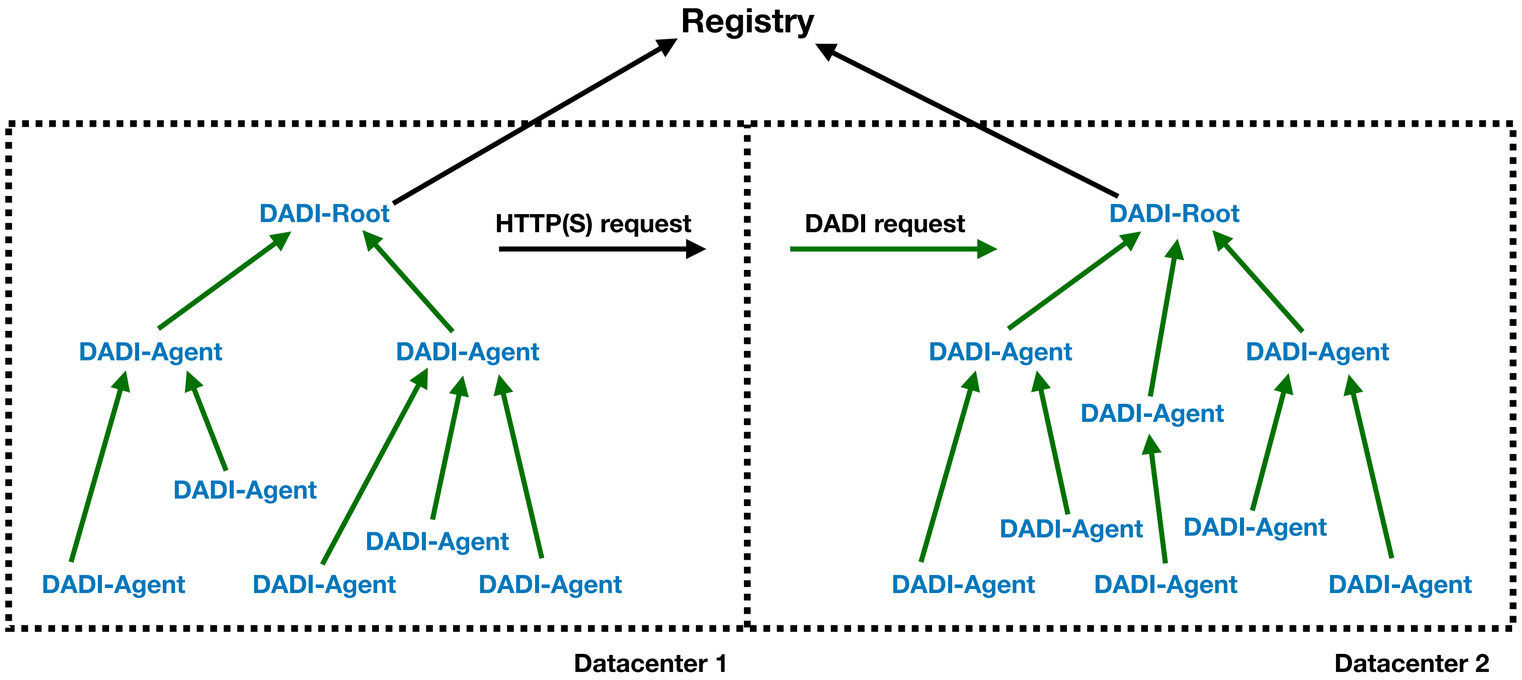
ZFile

- A seekable compression format
  - *random reading, and online decompression*
- Compressed by fixed-sized chunks
- Decompressed only needed chunks
- Not tied to DADI



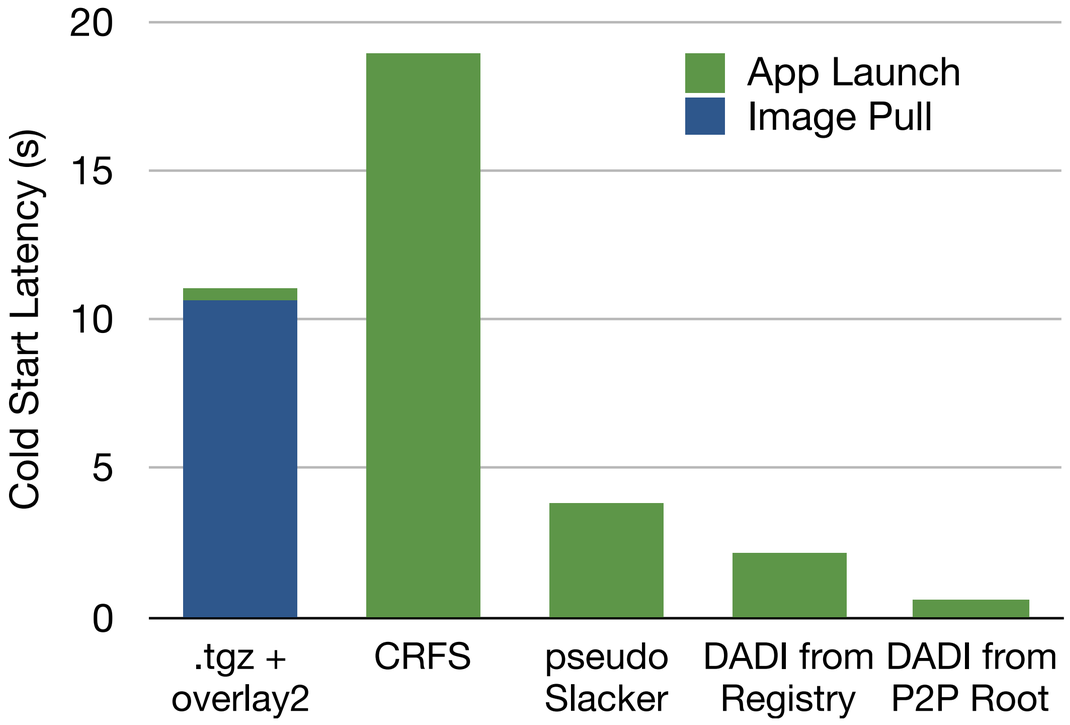
On-Demand P2P Transfer

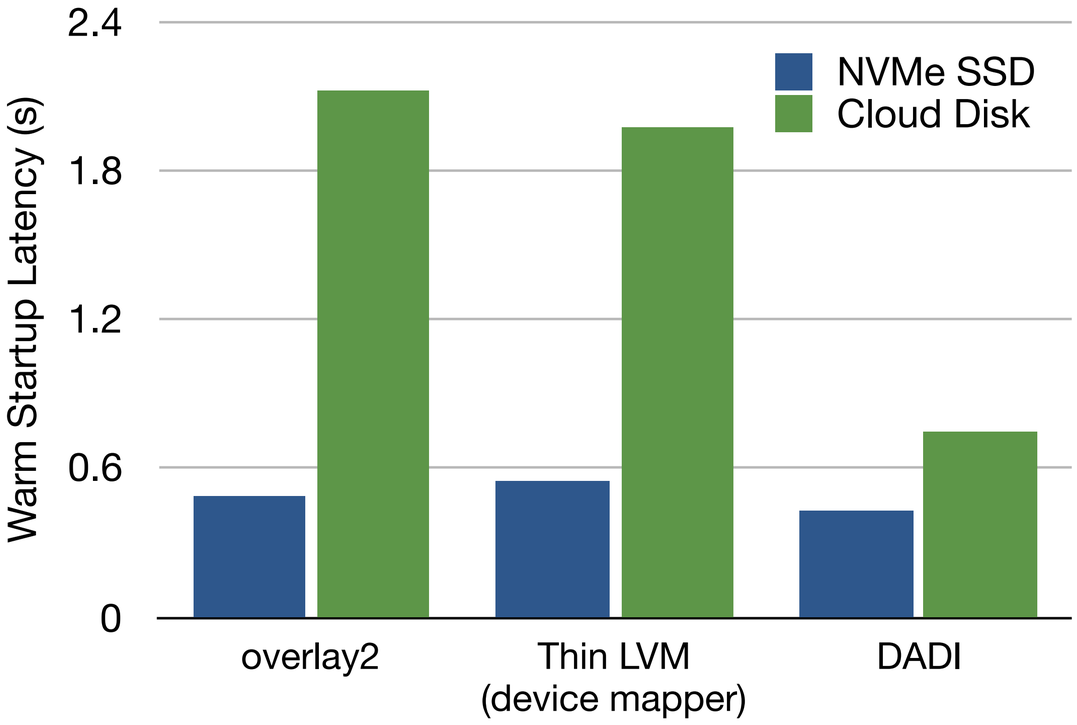
- In a tree-structured topology
  - *Each P2P node caches recently used data blocks.*
  - *A request is likely to hit parent's cache,*
  - *or the parent will forward the request upward, recursively.*



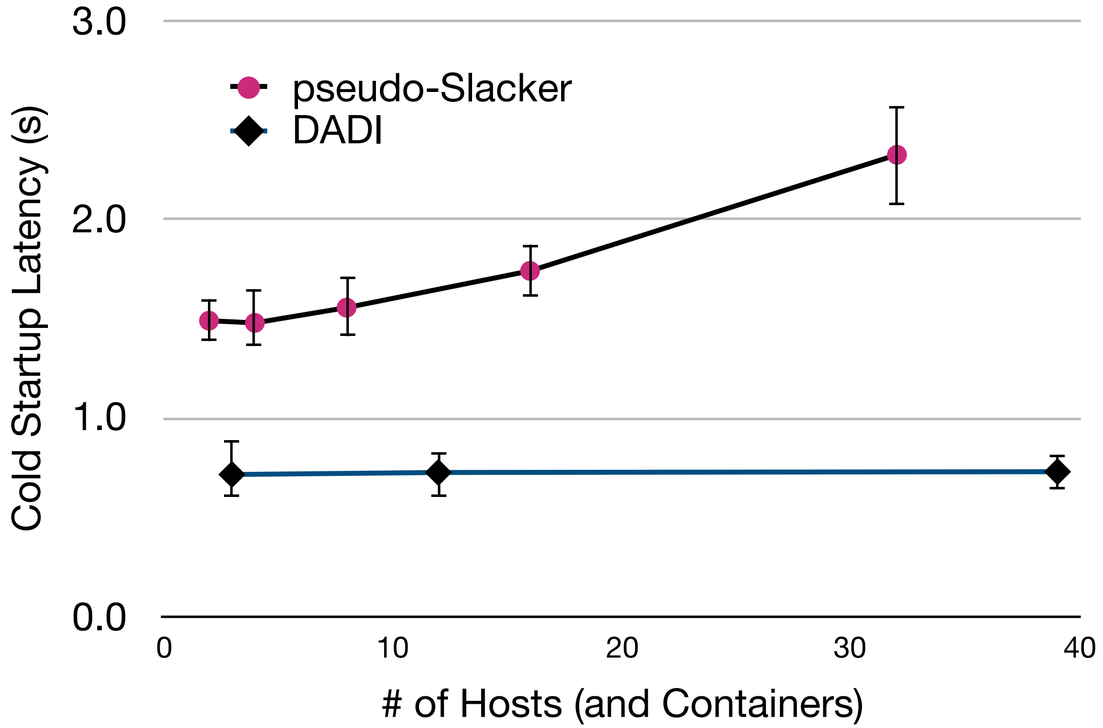
Evaluations

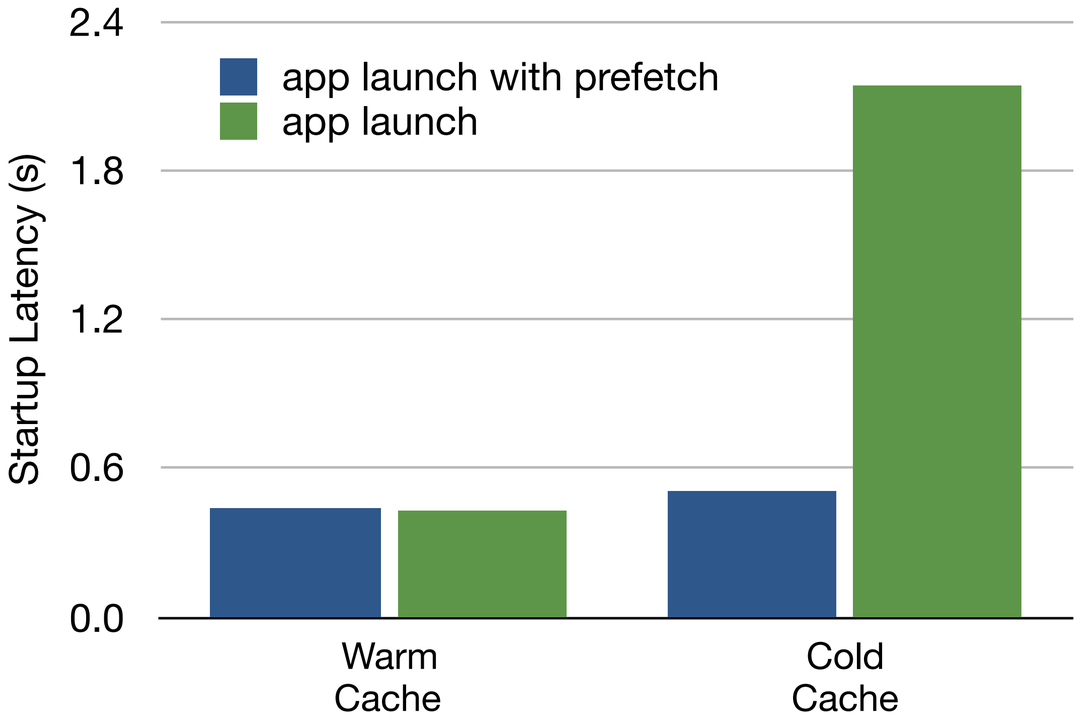
Startup Latency with DADI



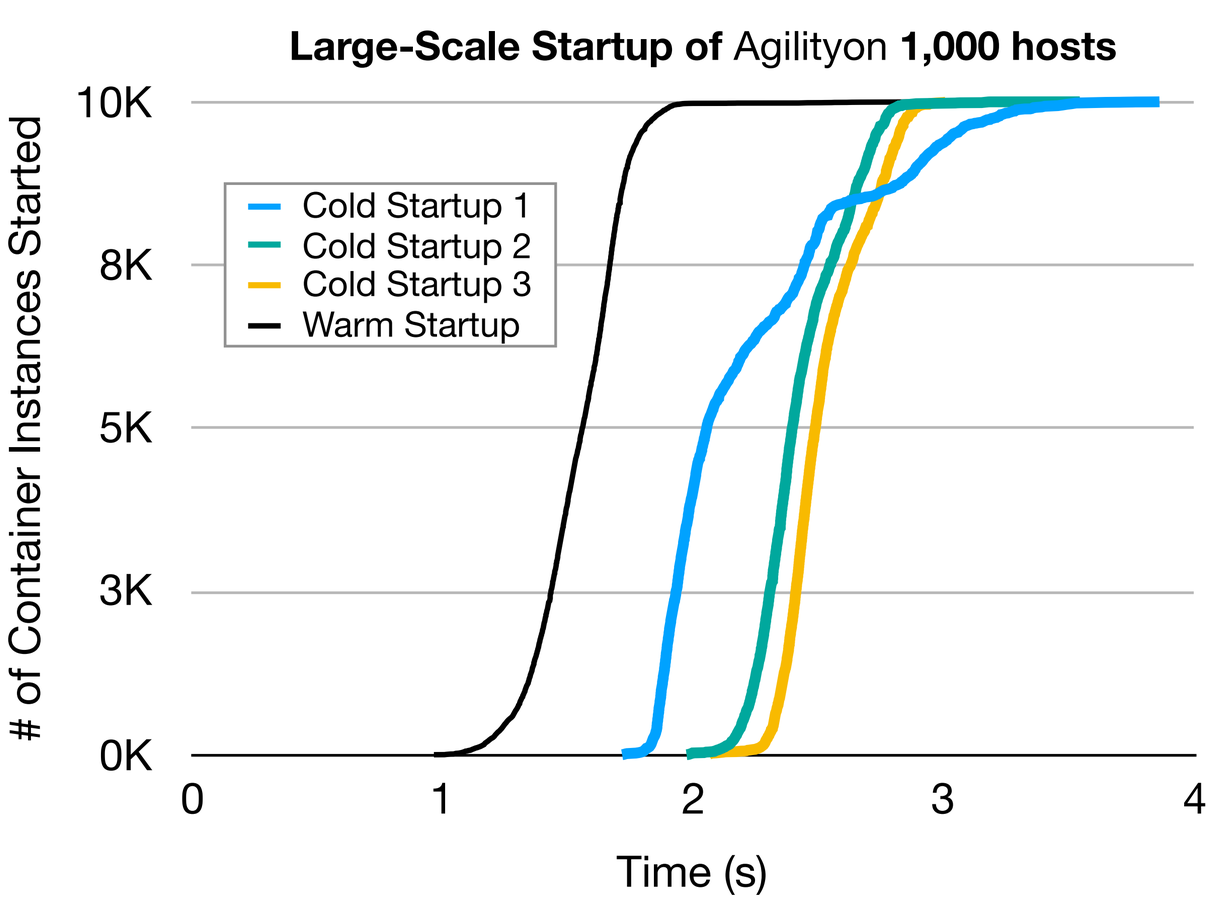


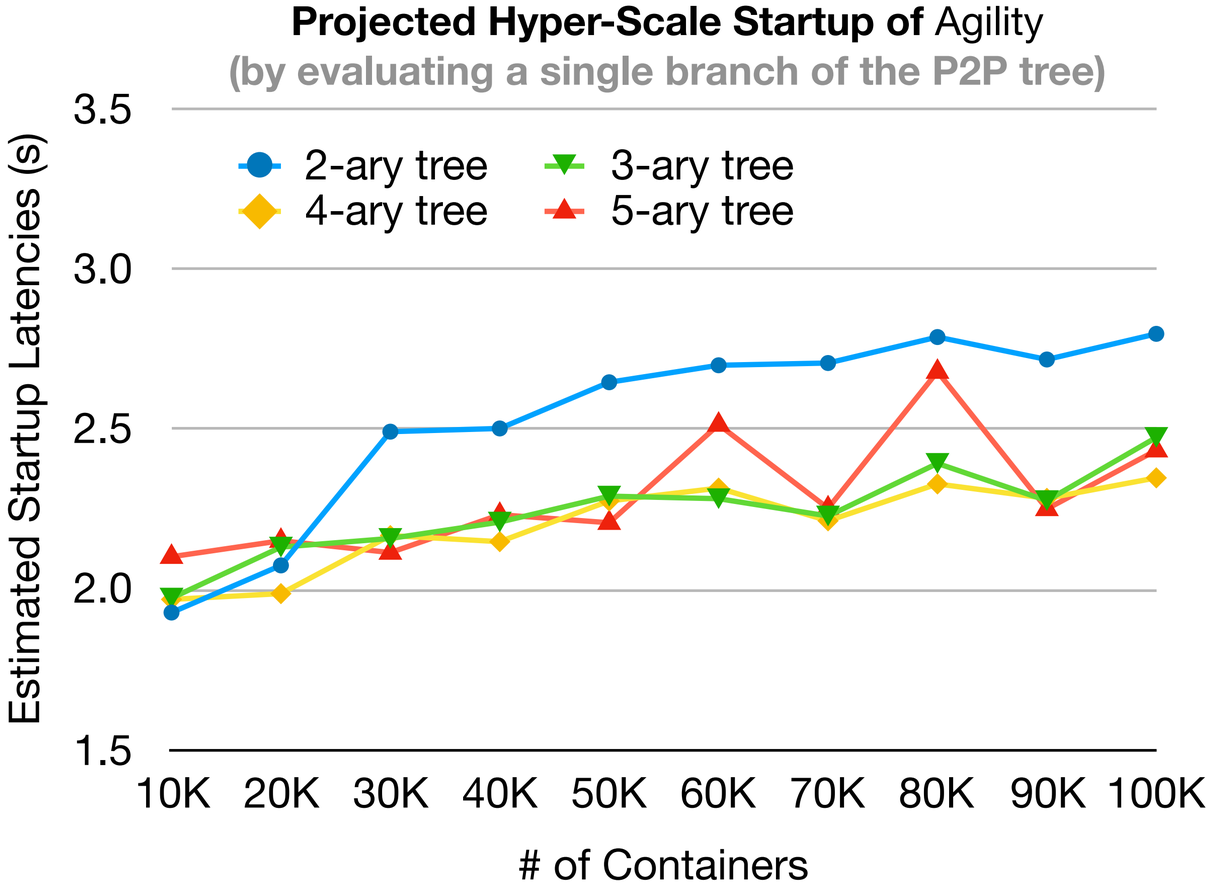
Startup Latency with DADI





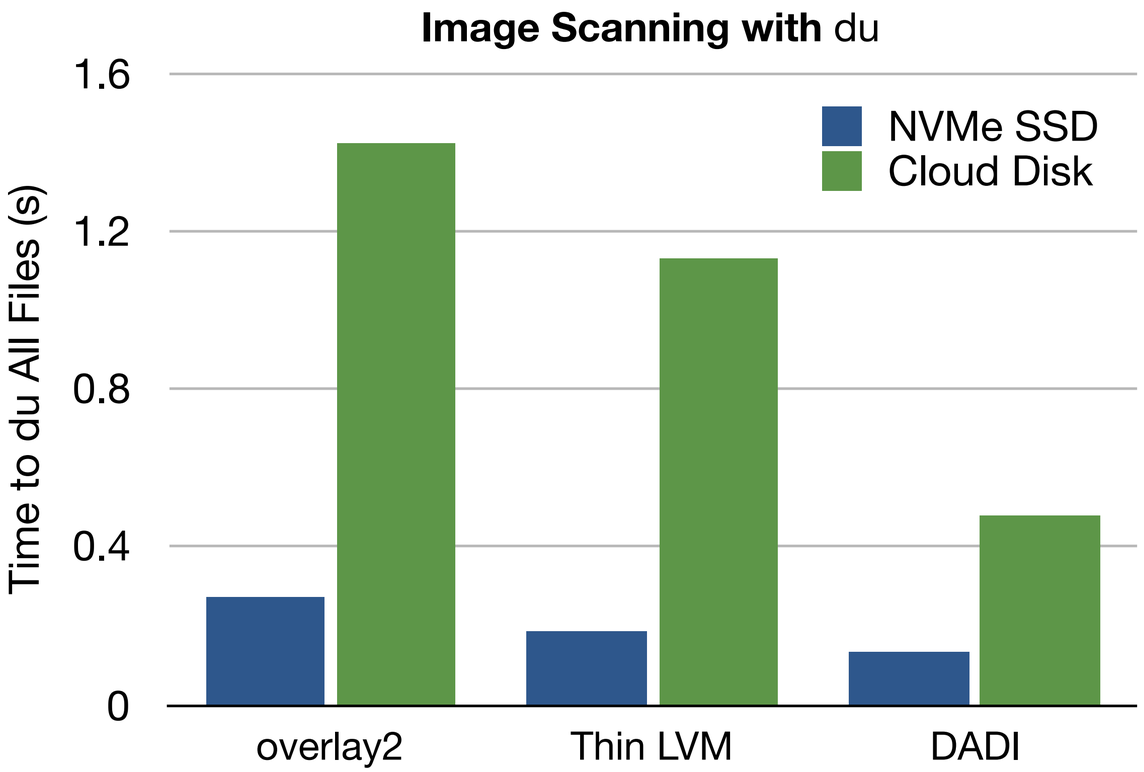
Scalability with DADI

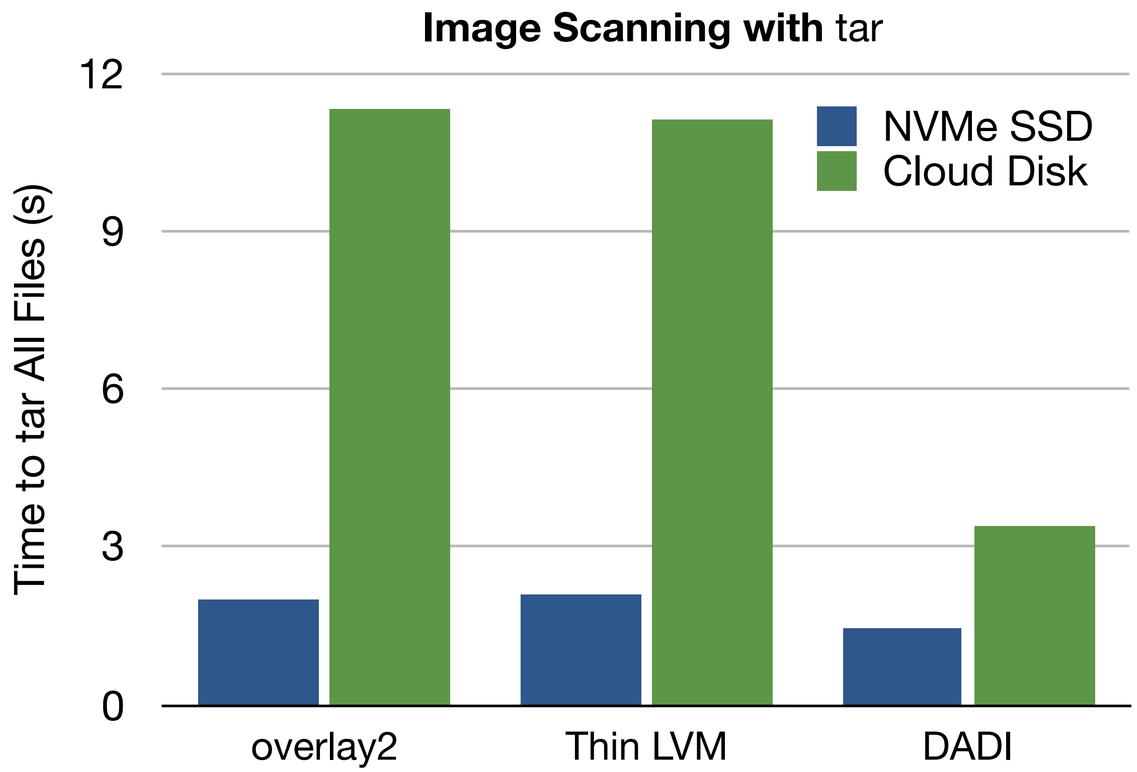




(Agility **is a small application specifically written in Python to assist the test)**

I/O Performance





Thanks!