UNIVERSITY OF NOTRE DAME

Wharf: Sharing Docker Image in a Distributed File System

Chao Zheng, Lukas Rupprecht, Vasily Tarasov,
Douglas Thain, Mohamed Mohamed, Dimitrios Skourtis,
Amit S. Warke and Dean Hildebrand

IBM Research

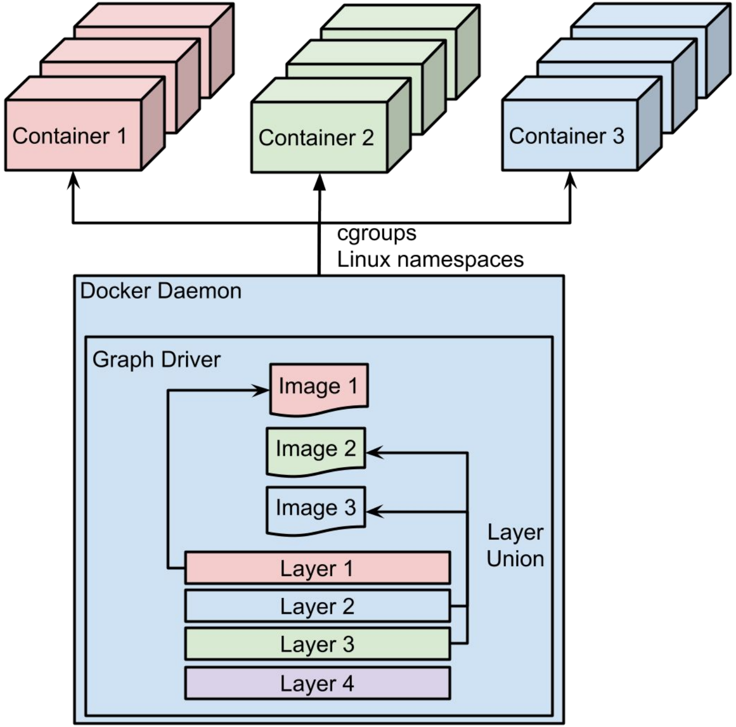
CCTools

1

Problem

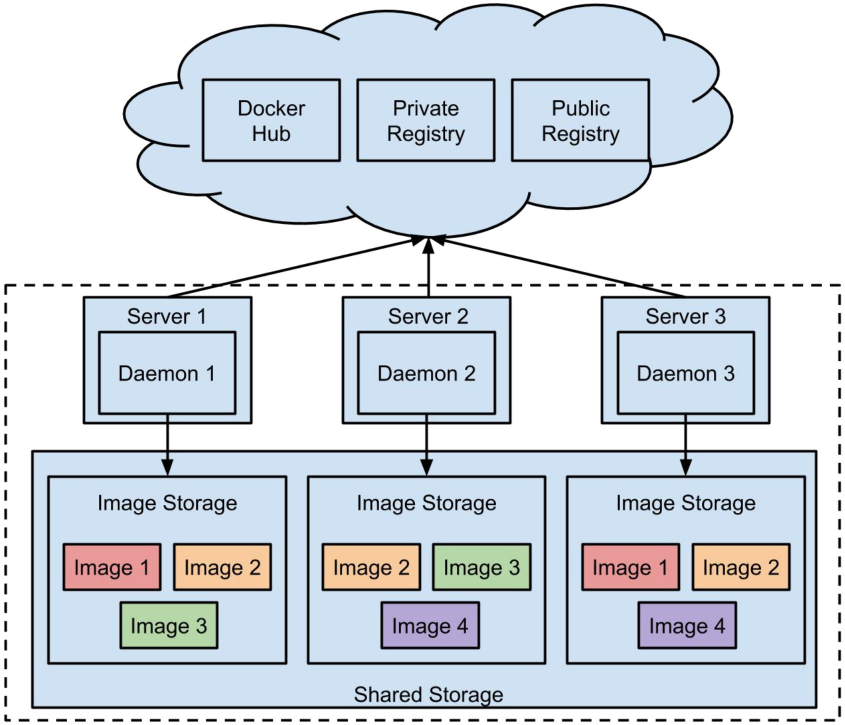
High Network and Storage Overheads

Docker Container Runtime



- Container Image & Layer
  - Multiple read-only layers and one writable layer
  - Different images may share layers
  - All changes are stored in writable layer (COW)
- Graph Driver
  - Overlay drivers: AUFS, OverlayFS/2
  - Specialized drivers: Devicemapper, btrfs

Docker on Distributed Storage



- High Network Overheads
- Waste Disk Space
- Longer Workload Startup Time

2

Solution

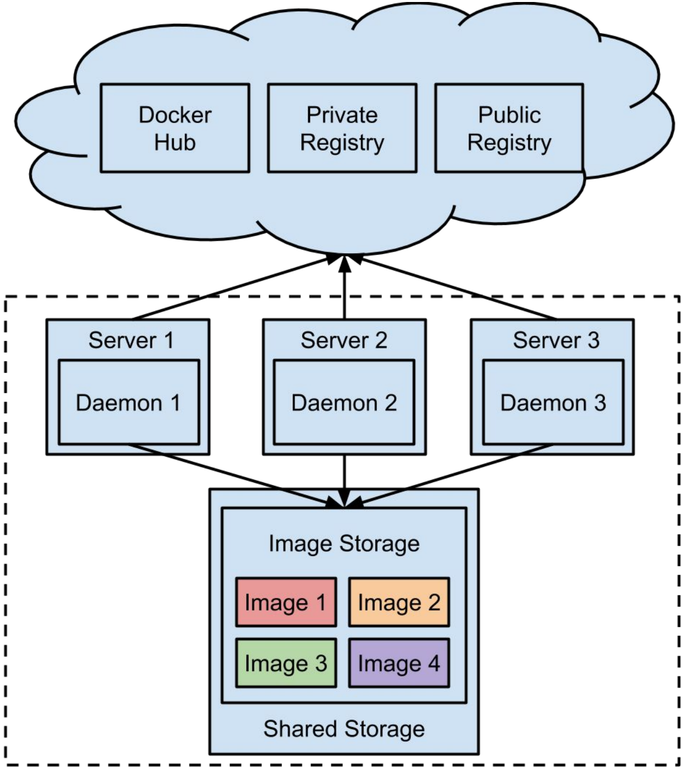
Share Images & Layers across Daemons

Share Layers Across Daemons

- Daemons share storage
  - Cluster often offer a shared storage layer to computing nodes
- Few data is read
  - Only **6.4%** of the image data is read by containers on average [1]

[1] Harter et al. Slacker: Fast Distribution with Lazy Docker Containers FAST'16

Challenges



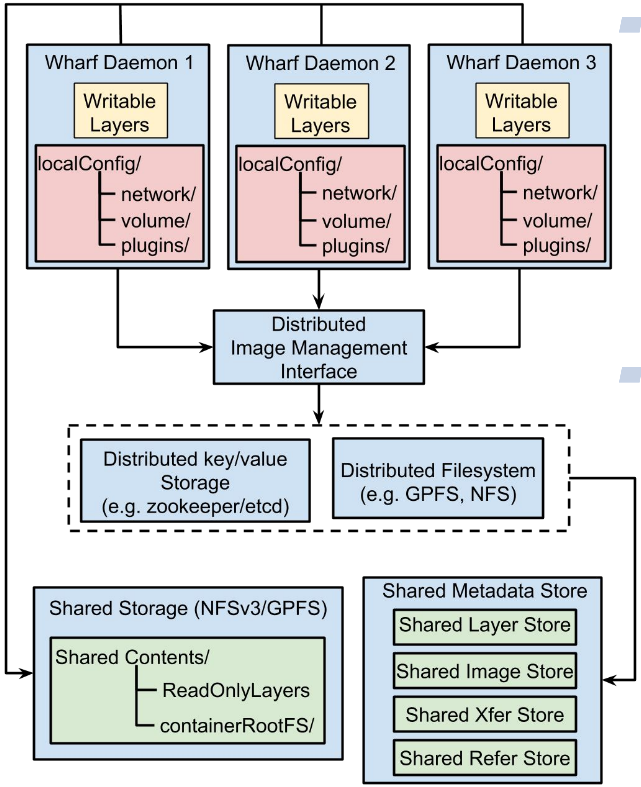
But, How to … ?

- Keep consistency between daemons
- Avoid potential performance degradation
- Avoid remote access to shared storage

Design Goals

- Avoid Redundancy
- Collaboration
- Efficient Synchronization
- Avoid Remote Access
- Fault Tolerance

Wharf



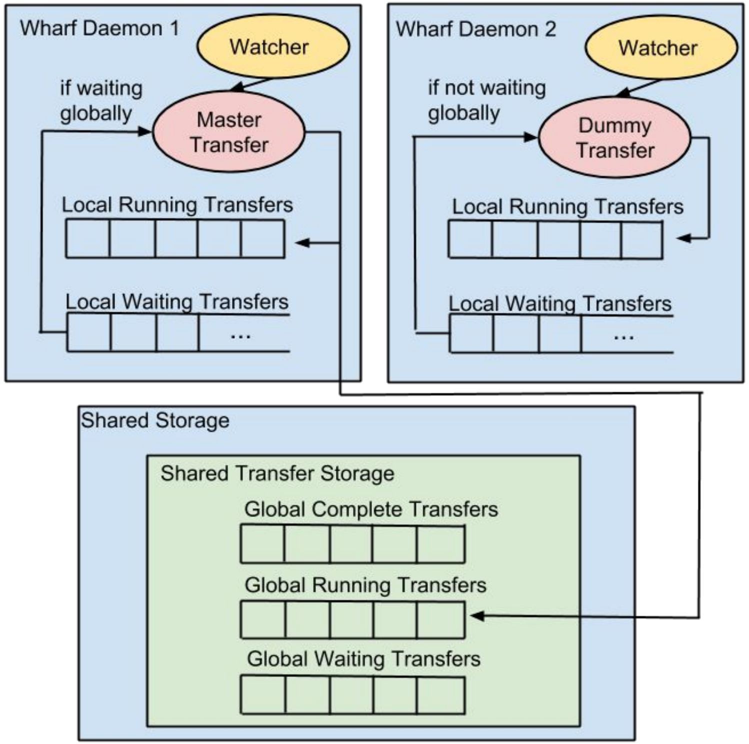
Global/Local State

- Global State: 1) ***Shared Data Store*** - image and layer data; 2) ***Shared Metadata Store*** - layer, image and xfering metadata
- Local State: 1) ***Metadata***: network, volume, container plugins, etc. 2) ***Container Data***: container writable layers

Read/Write Operations

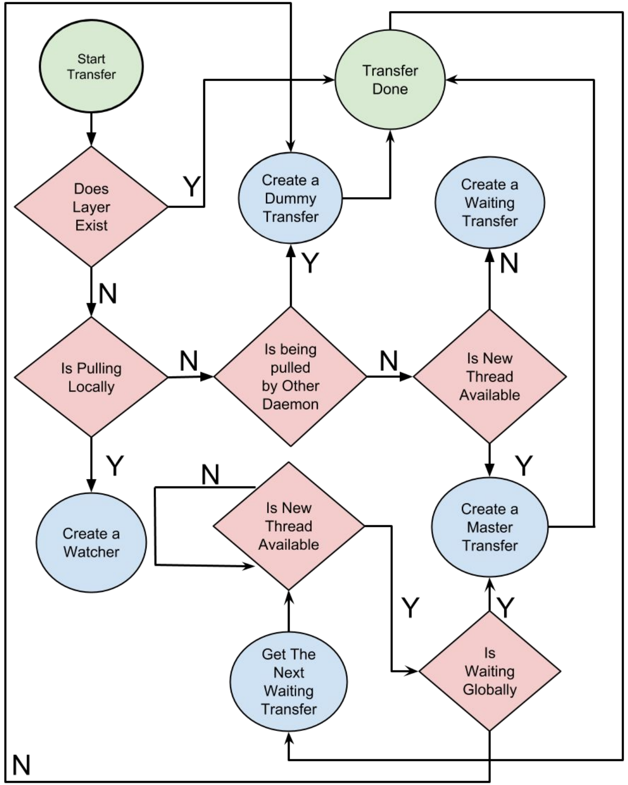
- All operations will access the shared metadata store, before the shared data store
- Read: read the global state. *eg.* list images
- Write: update the global state. *eg.* pull images

Fine-grained Locking



- Lock small portion of global state
  - Only lock the metadata related to the operation (list images, pull layer, …)
  - Operation can only be started after successfully accessing the metadata store
- Concurrent Read, Exclusive Write
- Extend the parallel model of Docker
  - Use watcher to watch the pulling of layers
  - Use dummy transfer to imitate real transfer

Concurrent Image Retrieval Workflow



3

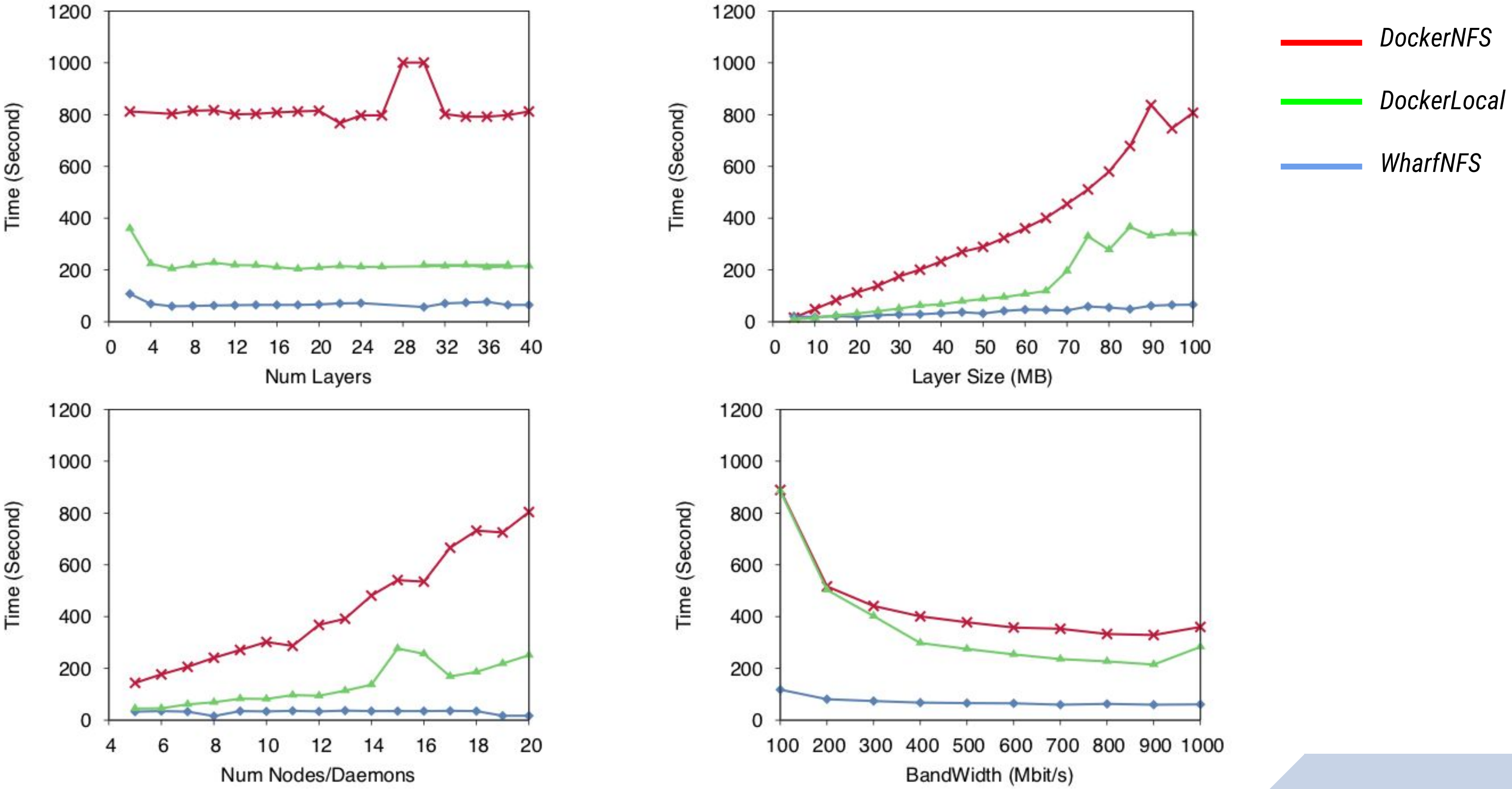
Evaluation

Wharf vs Docker

Experimental Setup

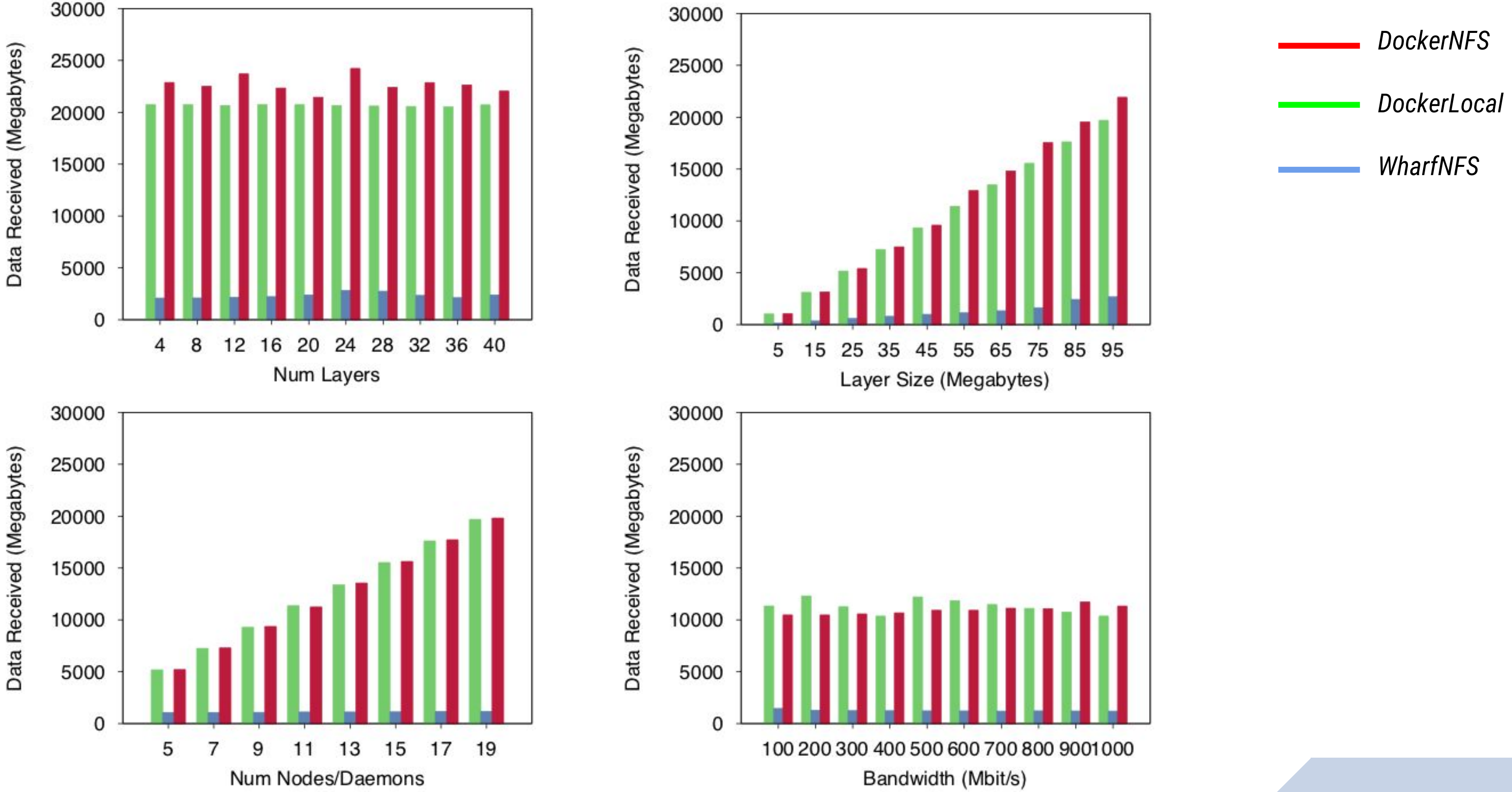
- Three configurations
  - Docker Local
  - Docker NFS
  - Wharf NFS
- Cluster Configuration
  - 5 - 20 aws t2.medium instances. .
  - Wharf is based on Docker CE17.05
  - Local image registry

Pull Latencies



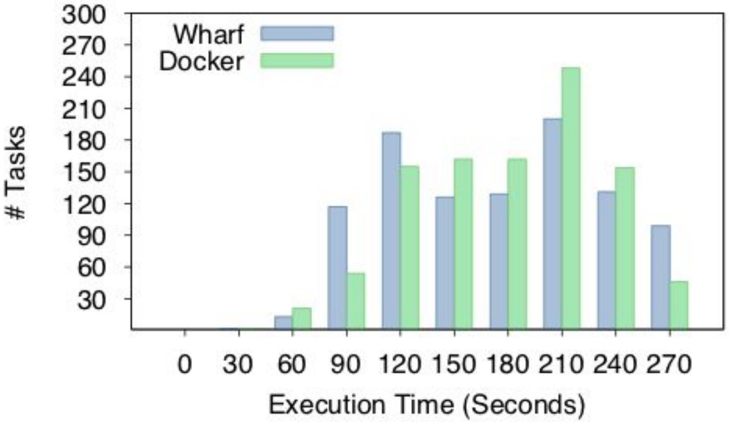
Network Overheads

Data From Registry



Runtime Overheads

	Docker	Wharf
Total Exec	**7 m 26 s**	**7 m 47 s**
Avg Exec (s)	**158**	**154**
Min Exec (s)	**31**	**46**
Max Exec (s)	**252**	**263**
Data Rev (MB)	**3227**	**354**
Data Sent (MB)	**50**	**768**



- Workload Spec
  - Bioinformatics Workflow
  - 1,000 parallel tasks
- Overhead mainly due to remote accesses

Summary

12 X

Faster pulling

9.1 X

Less data pulled, stored

2.6 - 4.7 %

Runtime degradation

THANKS!

Any questions?

You can find us at

czheng2@nd.edu