FastBuild: Accelerating Docker Image Building for Efficient Development and Deployment of Container

Zhuo Huang,
Song Wu, Hai Jin

Song Jiang

May 23, 2019

Outline

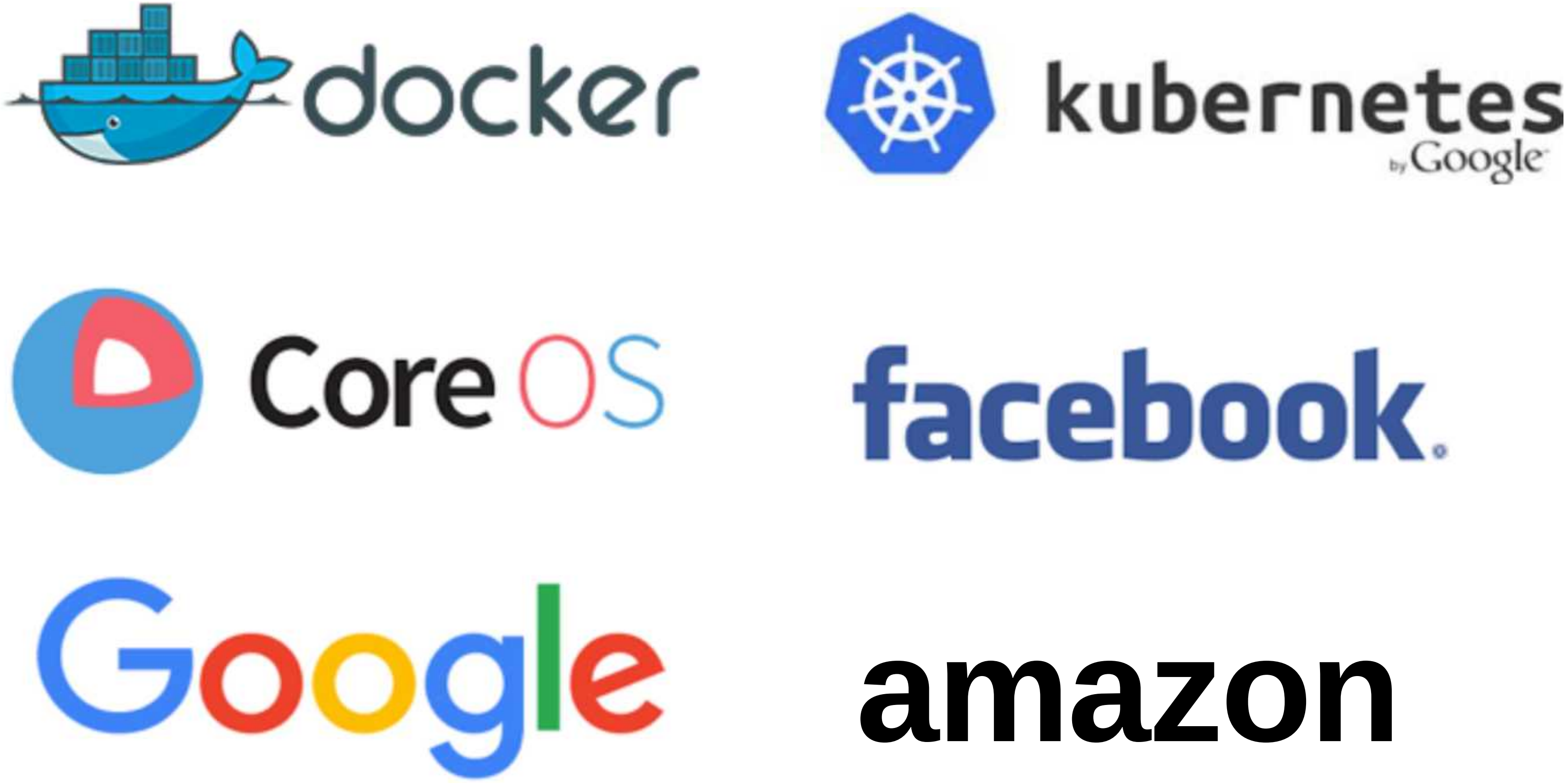
Background and Motivation

Design of FastBuild

Evaluations

Summary

Container is Popular



Starting a Container

By design: containers are lightweight

- ➢ can be started as fast as a process

In practice: container startup is much slower

- ➢ 25 seconds startup time

"task startup latency (the time from job submission to a task running) is an area that has received and continues to receive significant attention. It is highly variable, with the median typically about 25 s. Package installation takes about 80% of the total"

—— Large-scale cluster management at Google with Borg (EuroSys'15)

Container Image

An application is packaged as a container image that includes:

- application binary
- shared libraries
- Linux distribution

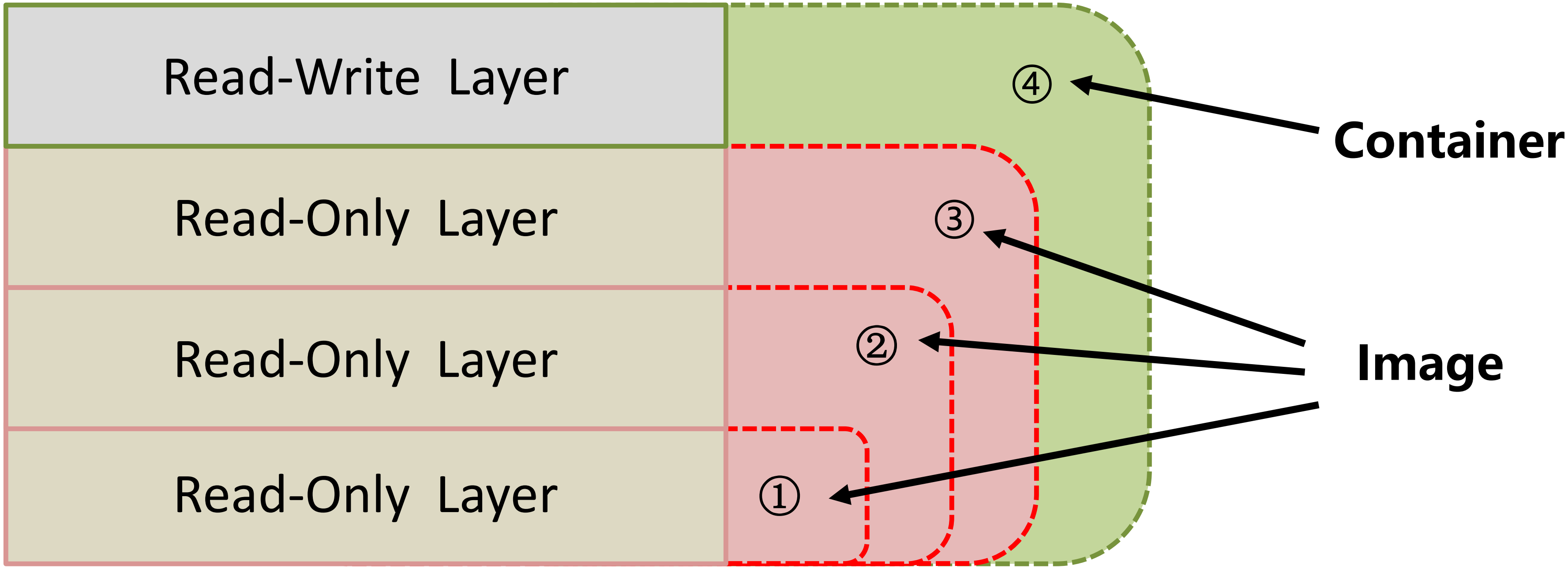
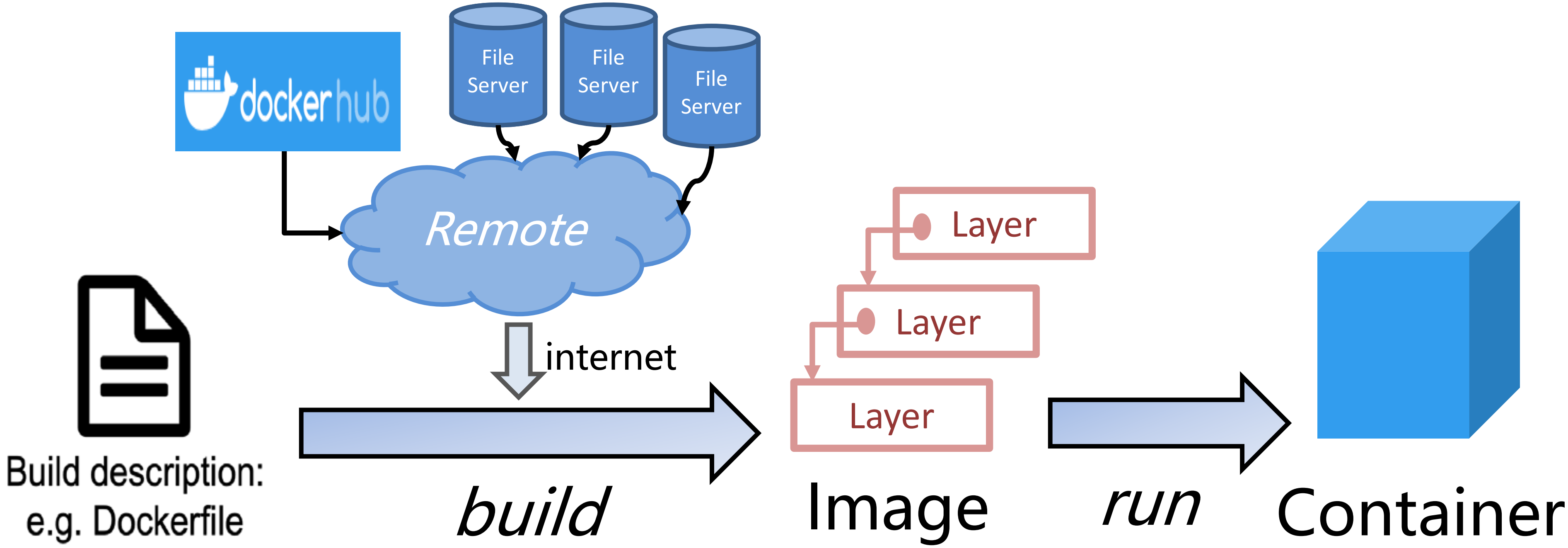


Image Building

Containers are becoming heavyweight:



Starting a container requires the image to be available.

Analysis

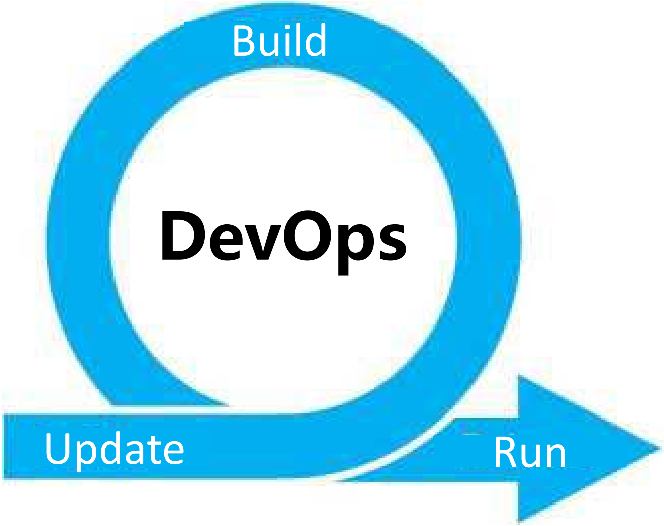
Objective:

Accelerate Docker image building

Method:

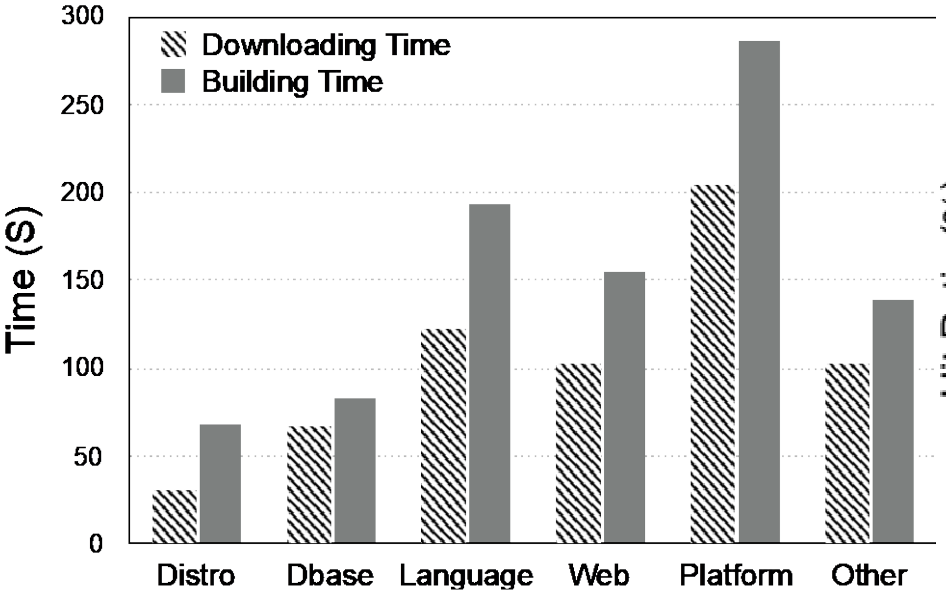
□ Build 2746 container images

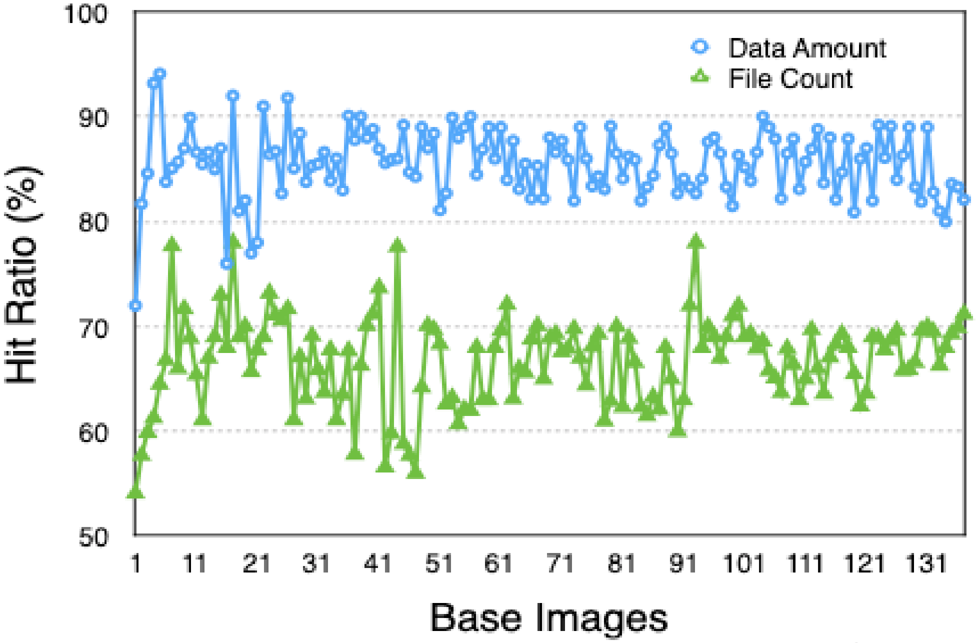
- ✓ base images are downloaded more than 100,000 times
- ✓ divided into 137 groups according to different repositories



Our Findings

- ✓ *70%* of the building time is spent on the remote file access
- ✓ *80%* of downloaded data are duplicated
- ✓ *30%* overlap of input data in different base images





Outline

Background and Motivation

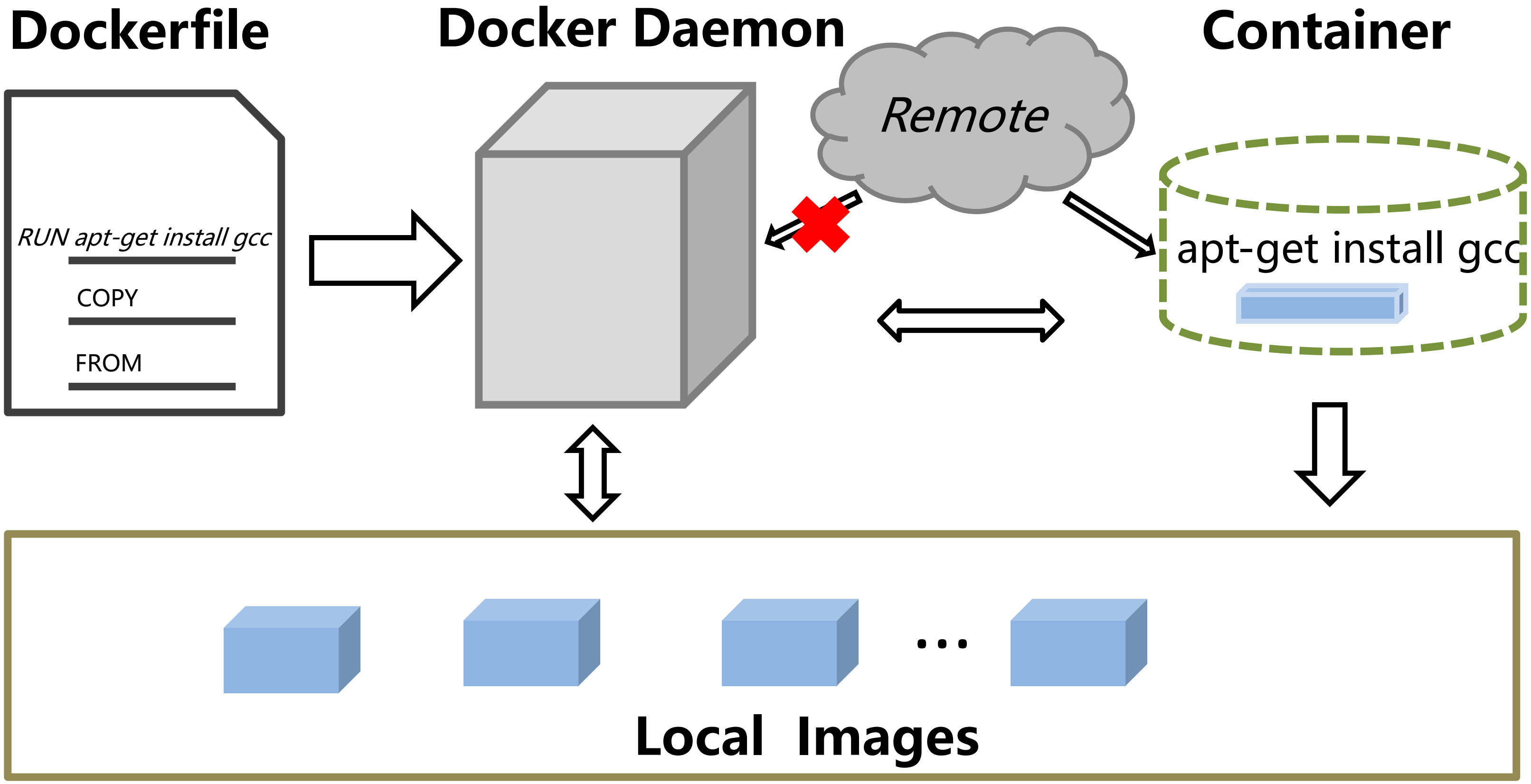
Design of FastBuild

Evaluations

Summary

Design Challenge

Obtain the requests for inputs without changing the image.



Design of FastBuild

Interception of requests for input files

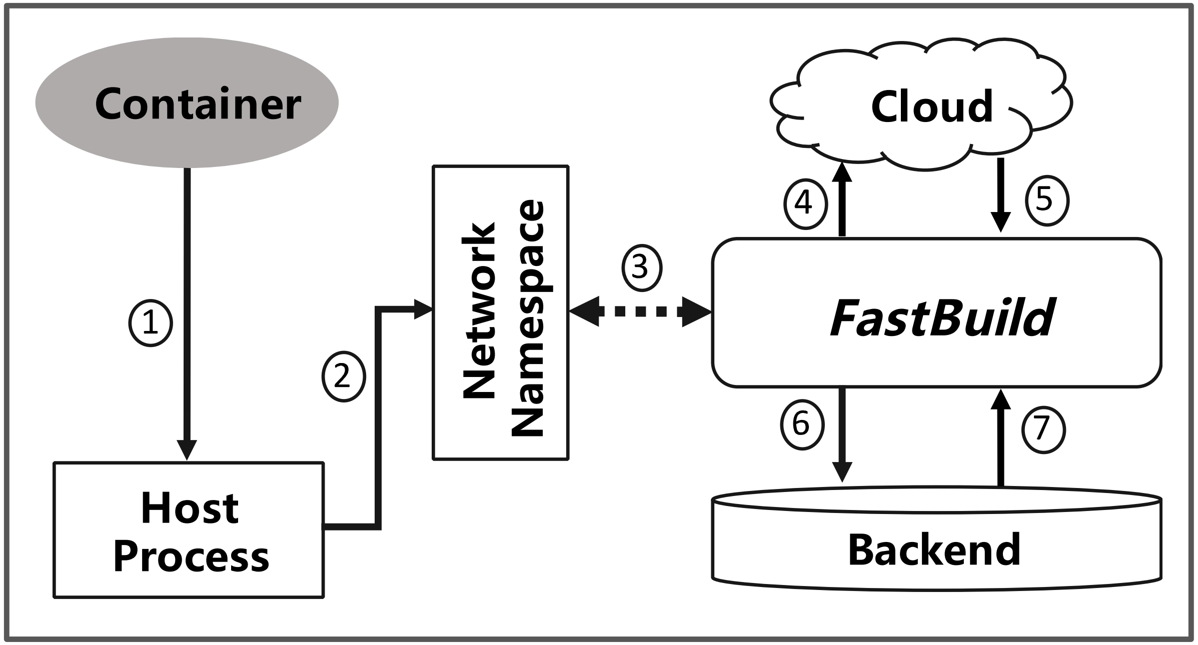
①： Resolve the container to the main process id;

②： Find out the network namespace by reading the */proc;*

③： Fork a child process to attach the namespace;

④⑤： Check the file timestamp;

⑥⑦： Search the local file cache.



Design FastBuild

- Instruction Overlapping
  - ✓ overlap instruction execution and layer commitment
  - ✓ build multiple layers of an image in one container instance
  - ✓ take a snapshot after executing each line instructions

- Quickly Obtaining Base Image
  - ✓ leverage the previous optimization to locally build base images.

Outline

Background and Motivation

Design of FastBuild

Evaluations

Summary

Evaluations

Experiment environment:

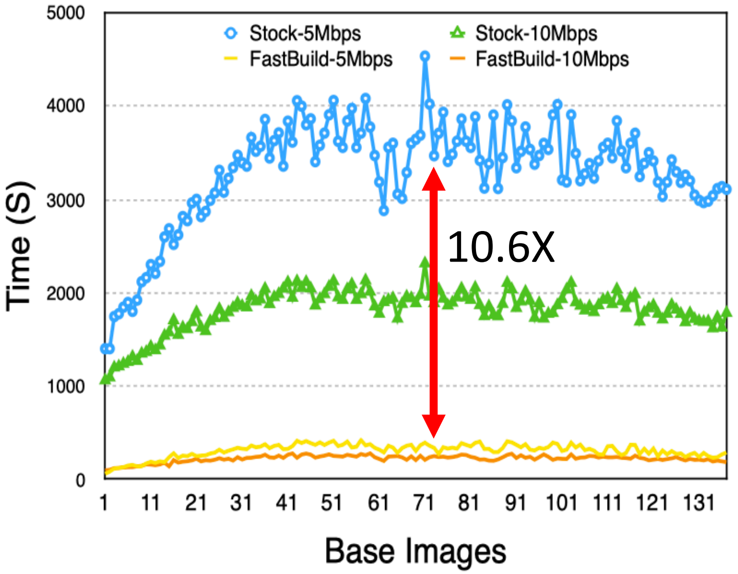
- ✓ 2.3 GHz Xeon CPUs(E5-2620)
- ✓ 64GB RAM
- ✓ Intel Gigabit CT PCIE Network Adapter
- ✓ West Digital WD60PURX hard disk
- ✓ China Education and Research Network

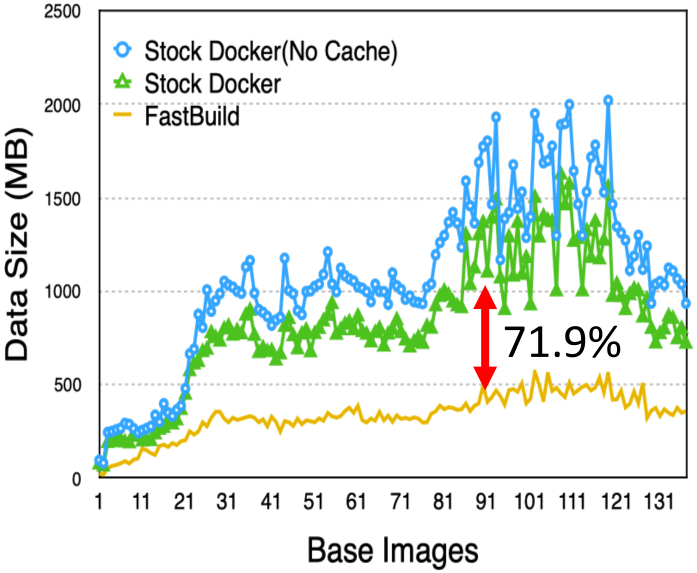
FastBuild prototype:

- ✓ 300 LoC for redirecting Dockerfile instruction
- ✓ 500 LoC for optimizing container runtime
- ✓ 1800 LoC for cache lookup

Evaluations

- ➢ FastBuild is about 4X faster than stock Docker for different image groups.
- ➢ FastBuild is 3.2X faster than stock Docker after execution of 6 Dockerfiles.
- ➢ FastBuild reduces 71.9% data downloaded.
- ➢ FastBuild can be 10.6X faster on the 5Mbps network.





Outline

- Background and Motivation
- Design of FastBuild
- Evaluations
- **Summary**

Summary

We extensively study how frequently input files are reaccessed in the building of Docker images and reveal opportunity of maintaining a local file cache for accelerating the process.

We propose and design FastBuild, a file caching function seamlessly integrated in Docker to transparently intercept requests for input files to minimize remote file access.

We prototype FastBuild in Docker 17.12 and extensively evaluate its impact on speed of Docker image building.

Thanks for your attention!

Any questions?

huangzhuo@hust.edu.cn