KubeCon | CloudNativeCon

Europe 2020

Virtual

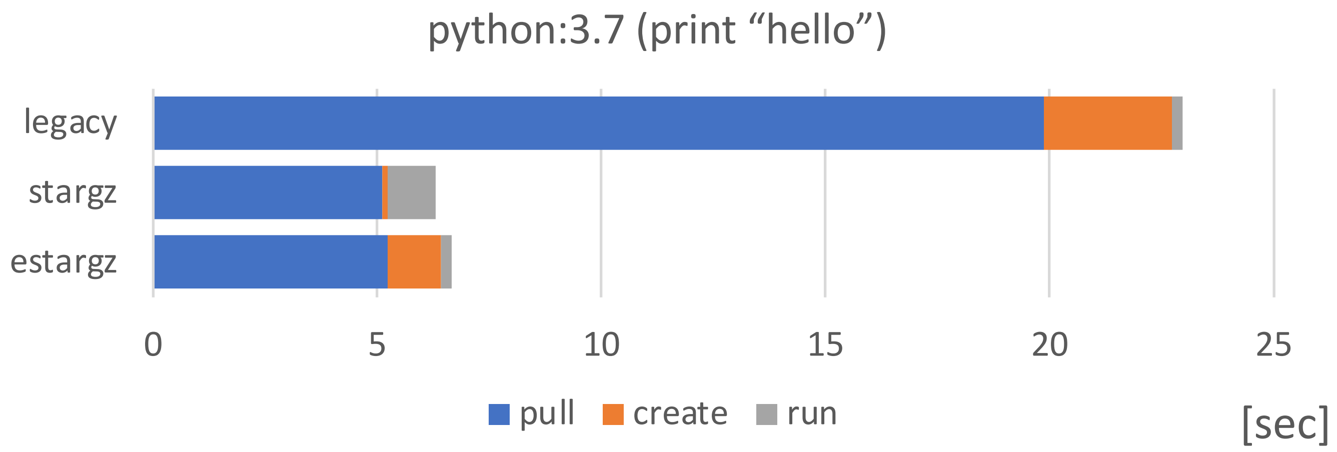
# Startup Containers in Lightning Speed with Lazy Image Distribution

*Kohei Tokunaga, NTT Corporation*

# Summary

- Pull is one of the time-consuming steps in container lifecycle

- Stargz Snapshotter, non-core subproject of containerd, is trying to solve it by lazy-pulling images leveraging stargz image by Google
  - Further runtime optimization is also held with an extended version of stargz (eStargz)



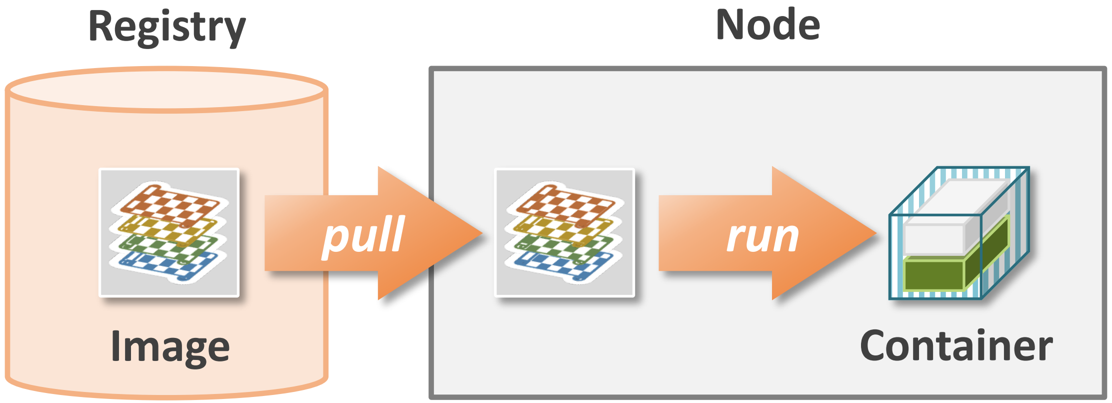
Host: EC2 Oregon (m5.2xlarge, Ubuntu 20.04)
Registry: Docker Hub (docker.io)
Commit b53e8fe
(See detailed info in the later slides)

- There are also other OCI-alternative image distribution strategies in container ecosystem

# Pull is time-consuming

pulling packages accounts for 76% of container start time,
but only 6.4% of that data is read [Harter et al. 2016]

[Harter et al. 2016] Tyler Harter, Brandon Salmon, Rose Liu, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau. "Slacker: Fast Distribution with Lazy Docker Containers". 14th USENIX Conference on File and Storage Technologies (FAST '16). February 22–25, 2016, Santa Clara, CA, USA

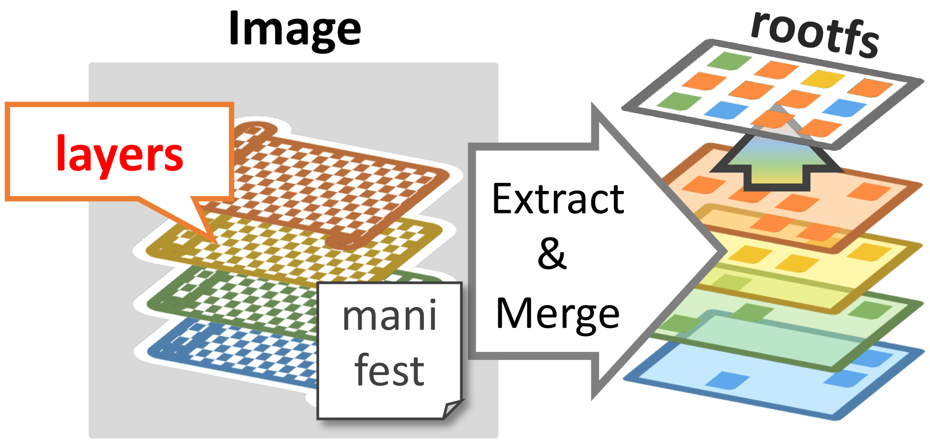


**Workarounds are known but not enough**

| | |
|---|---|
| Caching images | **Cold start is still slow** |
| Minimizing image size | **Not all images are minimizable**<br>Language runtimes, frameworks, etc. |

# OCI/Docker Specs for image distribution

A container is a set of ***layers***



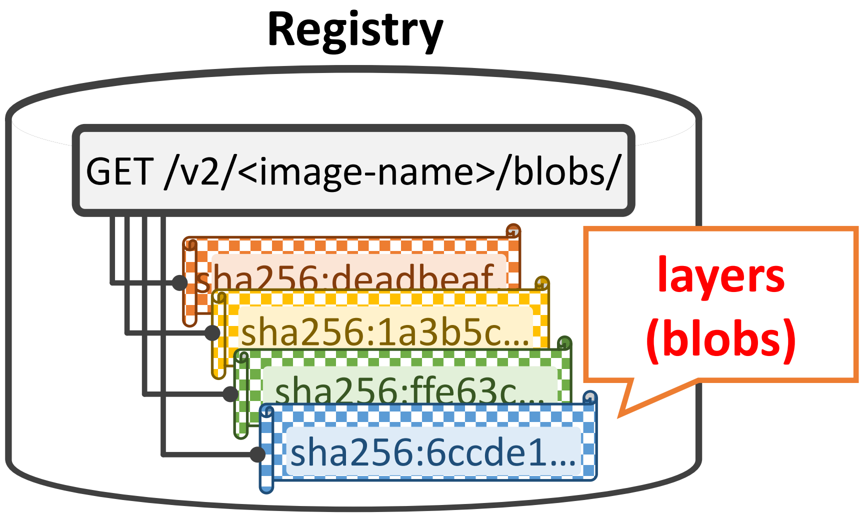
Image

layers

mani fest

Extract & Merge

rootfs



Registry

GET /v2/<image-name>/blobs/

sha256:deadbeaf

sha256:1a3b5c...

sha256:ffe63c...

sha256:6ccde1...

layers (blobs)

## Image Spec

- Defines layers and metadata (image manifest, etc.)
- Layer is defined as tar (+compression)
- Rootfs can be composed by merging layers

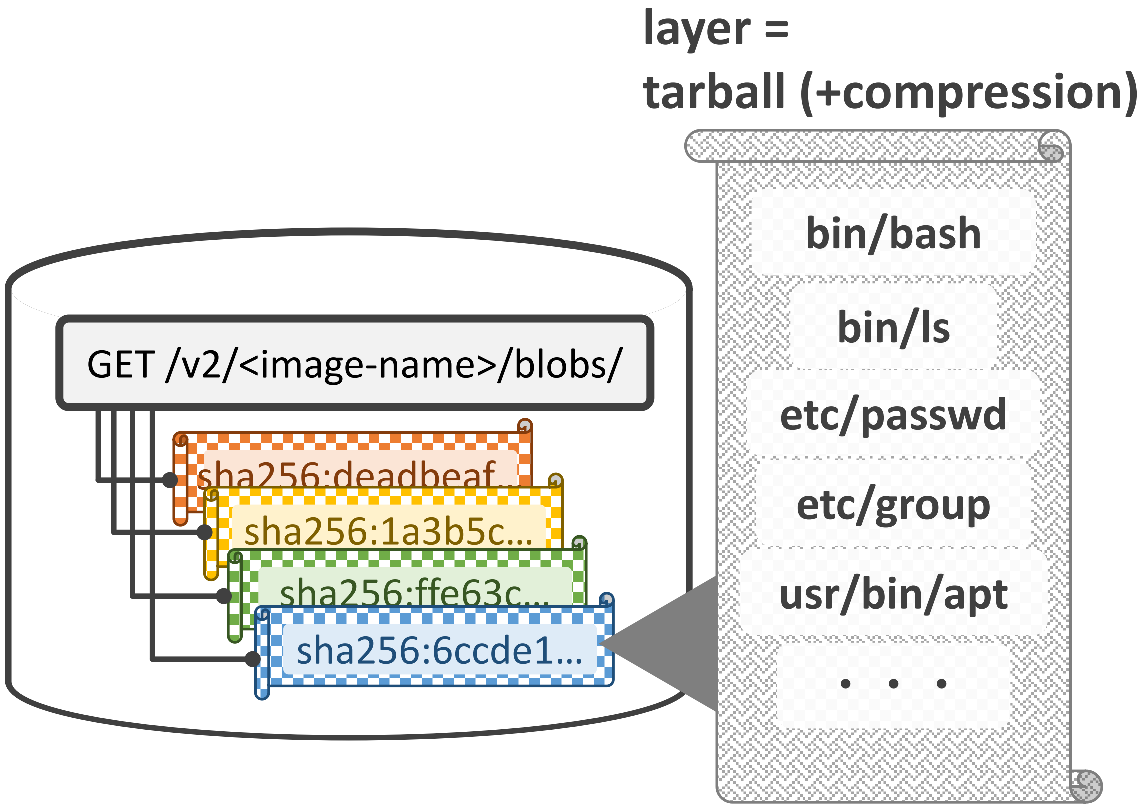
## Distribution Spec

- Defines HTTP API of registry
- Layer can be fetched as a “blob” named with a content-addressable digest
- Optional support for HTTP Range Request

# Problems on the OCI/Docker Specs

A container is a set of ***tarball layers***

A container can't be started until the all layers become locally available even if the most of the contents won't be used on container startup



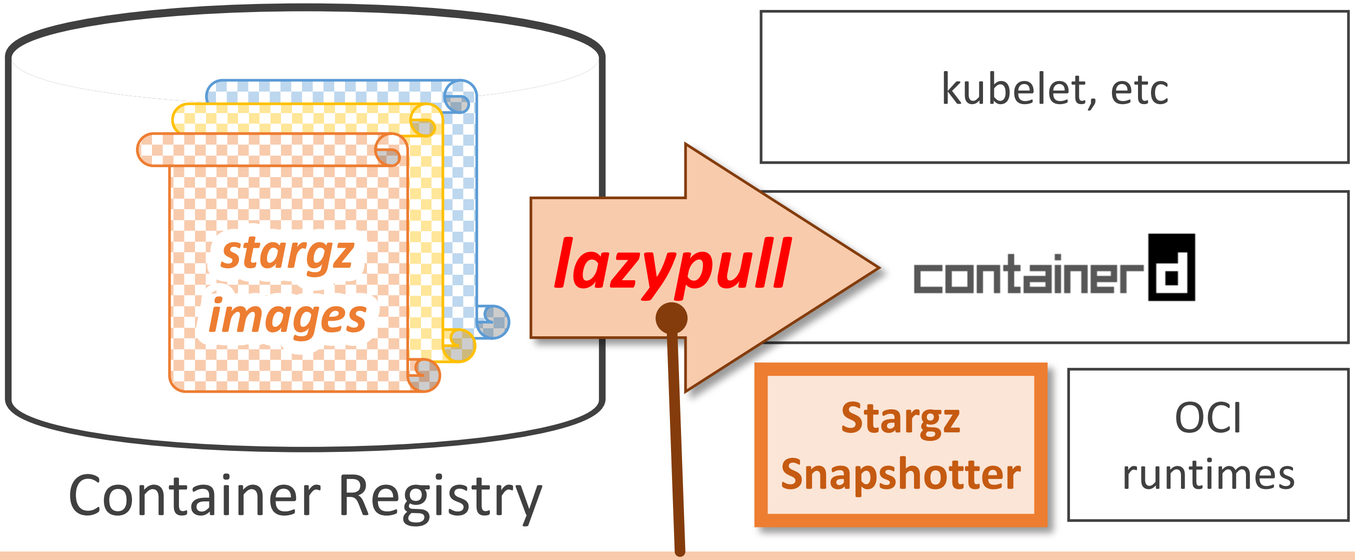
- Need to scan the entire blob even for extracting single file entry
  - If the blob is gzip-compressed, it's non-seekable anymore
- No parallel extraction
  - Need to scan the blob from the top, sequentially

# Lazypull with containerd Stargz Snapshotter

https://github.com/containerd/stargz-snapshotter

## Stargz Snapshotter

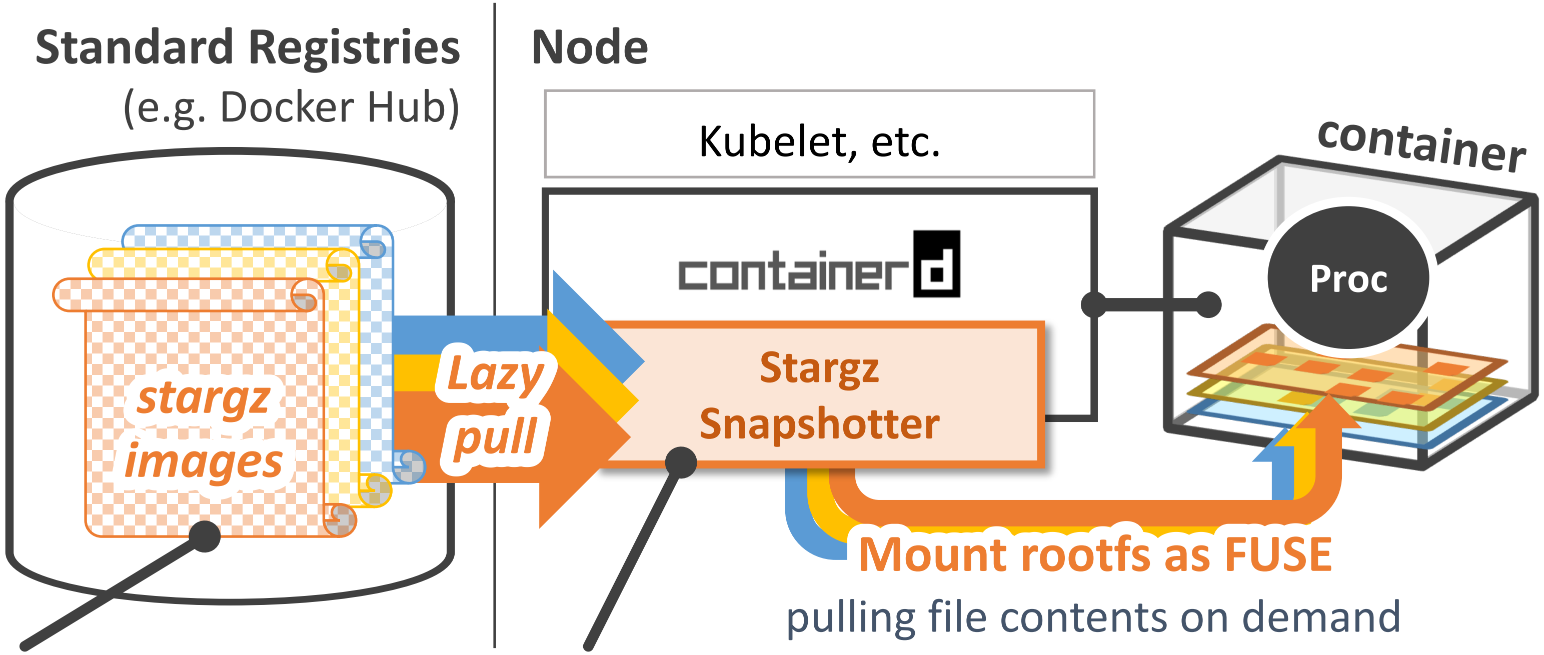
- **Non-core** subproject of containerd
- Works as a plugin of containerd
- Standard-compliant lazy pull leveraging **stargz image** by Google



doesn’t download the entire image on pull operation
but fetches necessary chunks of contents on-demand

# Standard-compliant lazypull

- Leverages OCI/Docker compatibility of stargz:
  - can be lazily pulled from standard registries
  - can also be run by legacy runtimes (but not lazily pulled)
- Mounts rootfs snapshots as FUSE and downloads accessed file contents on-demand



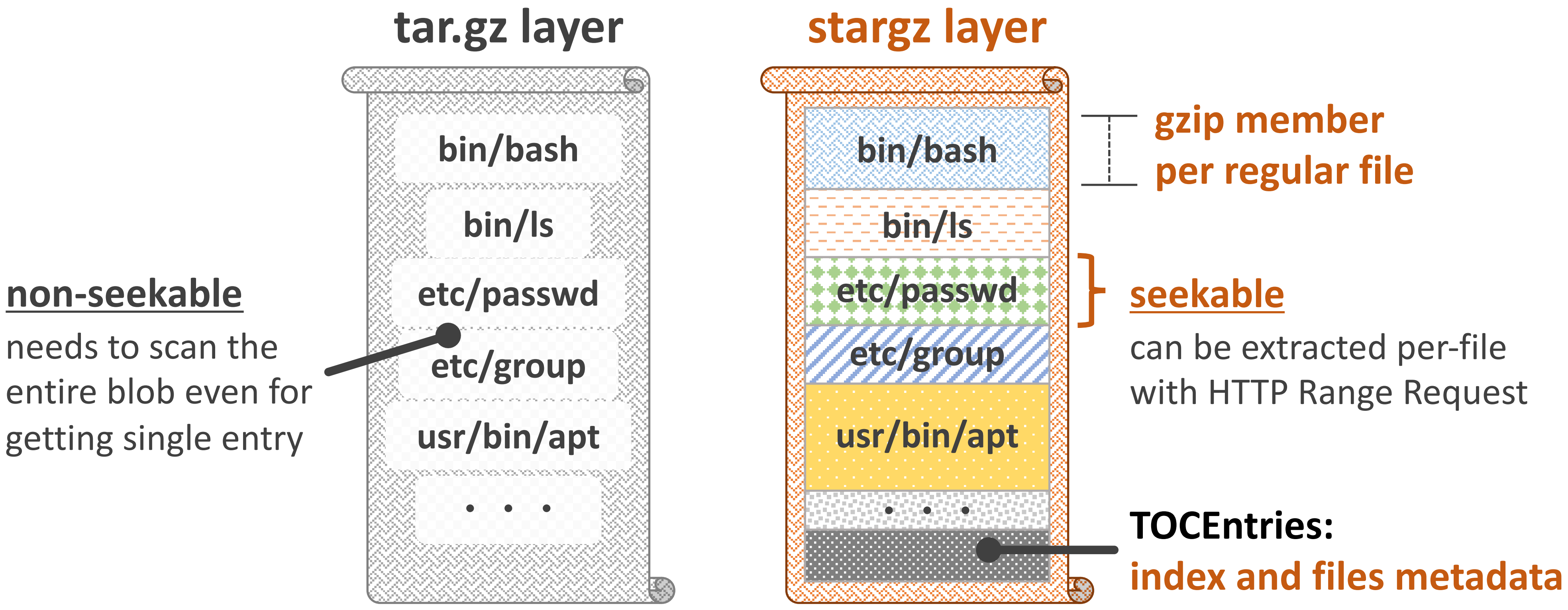
pulling file contents on demand

still pullable/runnable by legacy runtimes

implemented as a **remote snapshotter** plugin

# Stargz archive format

- Proposed by Google CRFS project: https://github.com/google/crfs
- Stands for ***Seekable targz*** so it's seekable but **still valid targz** = **usable as a valid OCI/Docker image layer**
- Entries can be extracted separately
  - Can be fetched separately from registries using HTTP Range Request



**non-seekable**

needs to scan the entire blob even for getting single entry

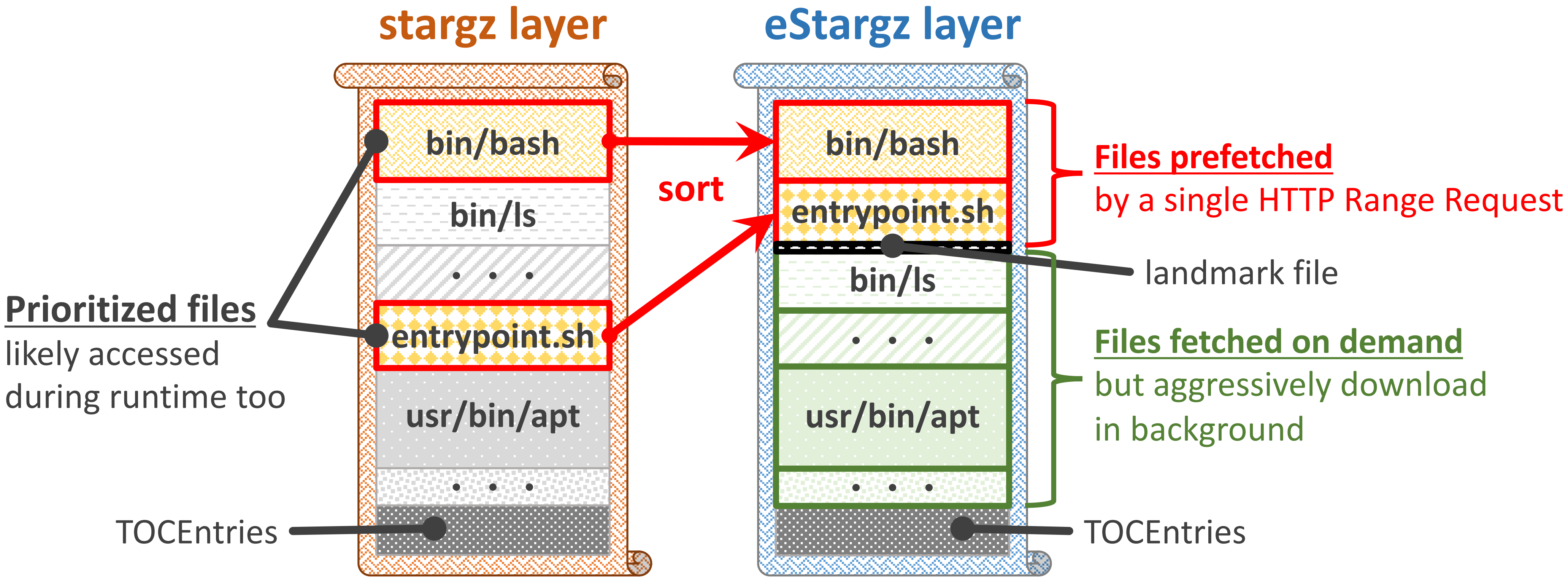
**seekable**

can be extracted per-file with HTTP Range Request

**TOCEntries:**
**index and files metadata**

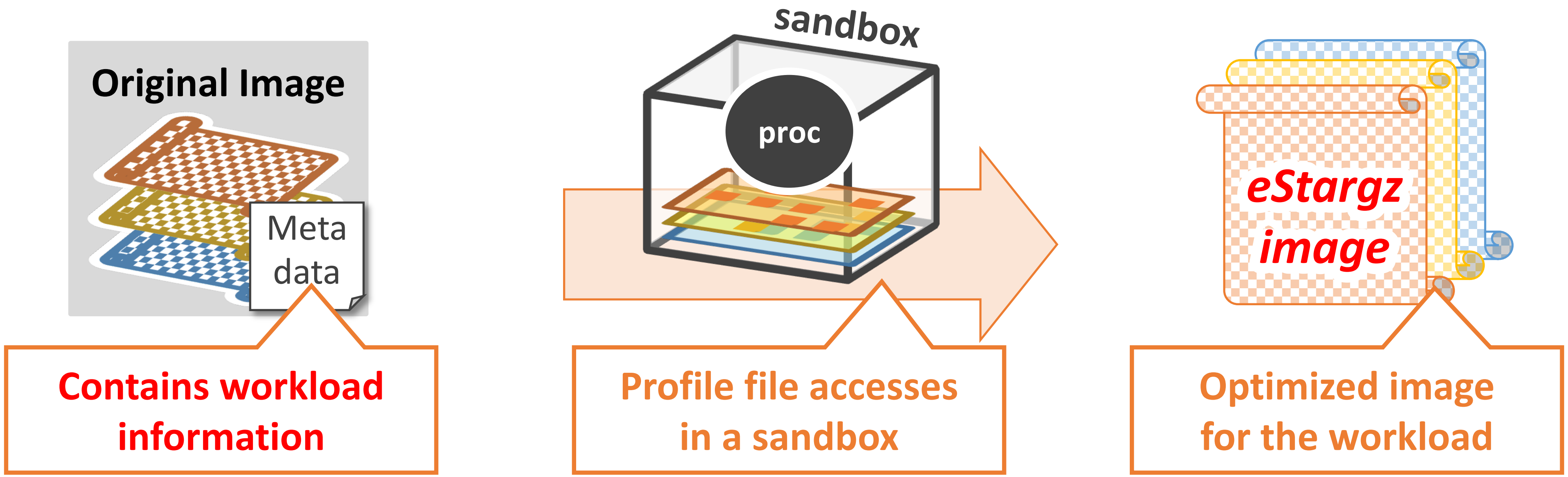
# eStargz archive for prefetch

- NW-related overheads can't be ignored for on-demand fetching with stargz
- **eStargz** enables to **prefetch** files that are llikey accessed during runtime (= **prioritized** files)
- Filesystem prefetches and pre-caches these files with a single HTTP Range Request on mount



# Workload-based runtime optimization with eStargz

- Leveraging eStargz, CLI converter command provides **workload-based optimization**
- Generally, containers are built with purpose
  - Workloads are defined in the Dockerfile, etc. (entrypoint, user, envvar, etc…) and stored in the image
- CLI converter runs provided image in a sandbox and profiles all file accesses
  - Regards accessed files are also likely accessed during runtime (= **prioritized** files in eStargz)
  - Stargz Snapshotter will prefetch and pre-caches these files when mounts this eStargz image



**Contains workload information**

Specified by Dockerfile, etc. (entrypoint, user, envvar)

**Profile file accesses in a sandbox**

Custom workloads can be specified throught CLI options

**Optimized image for the workload**

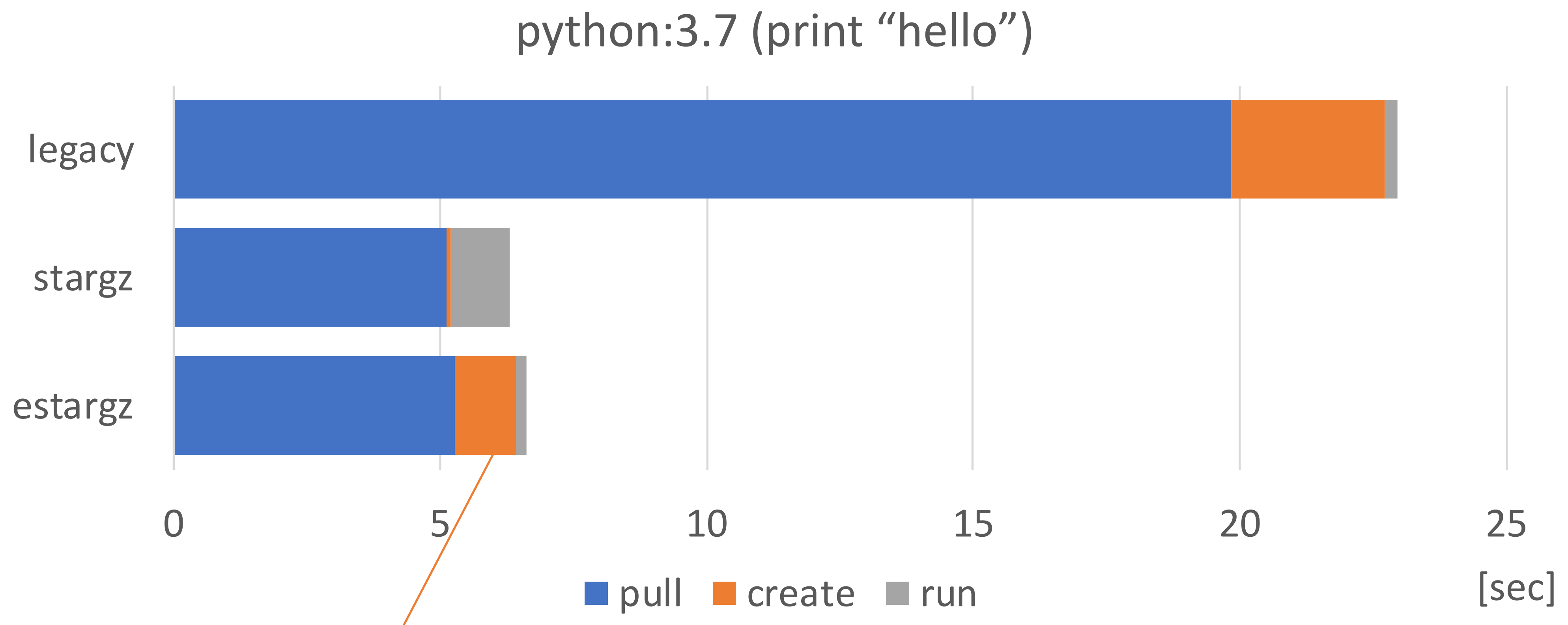
# Benchmarking results

- Measures the container startup time which includes:
    - Pulling an image from Docker Hub
    - For language containers, running “print hello world” program in the container
    - For server containers, waiting for the readiness (until “up and running” message is printed)
    - ➢ This method is based on Hello Bench [Harter, et al. 2016]
- Takes 95 percentile of 100 operations
- Host: EC2 Oregon (m5.2xlarge, Ubuntu 20.04)
- Registry: Docker Hub (docker.io)
- Target commit: b53e8fe8d37751753bc623b037729b6a6d9c1122

[Harter et al. 2016] Tyler Harter, Brandon Salmon, Rose Liu, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau. "Slacker: Fast Distribution with Lazy Docker Containers". 14th USENIX Conference on File and Storage Technologies (FAST ’16). February 22–25, 2016, Santa Clara, CA, USA

**Credit to Akihiro Suda (NTT) for discussion and experiment environment**

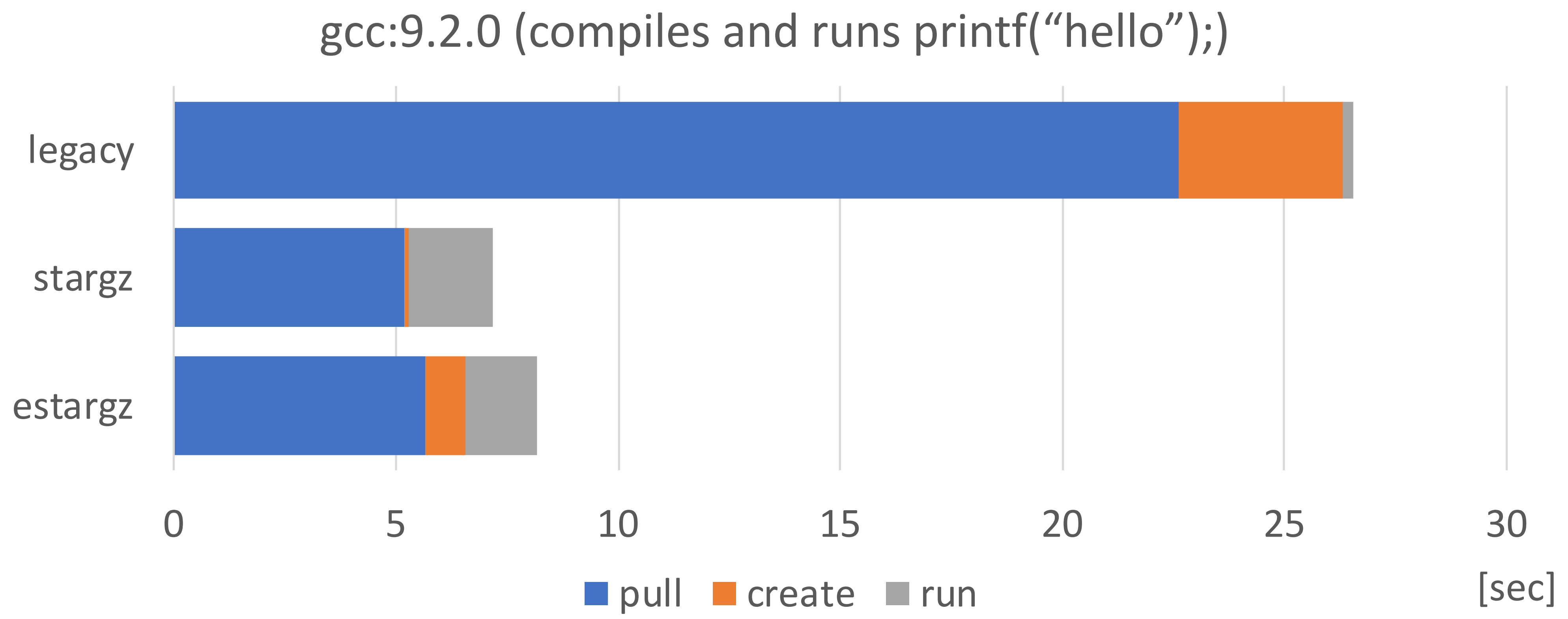
# Time to take for container startup



Waits for prefetch completion

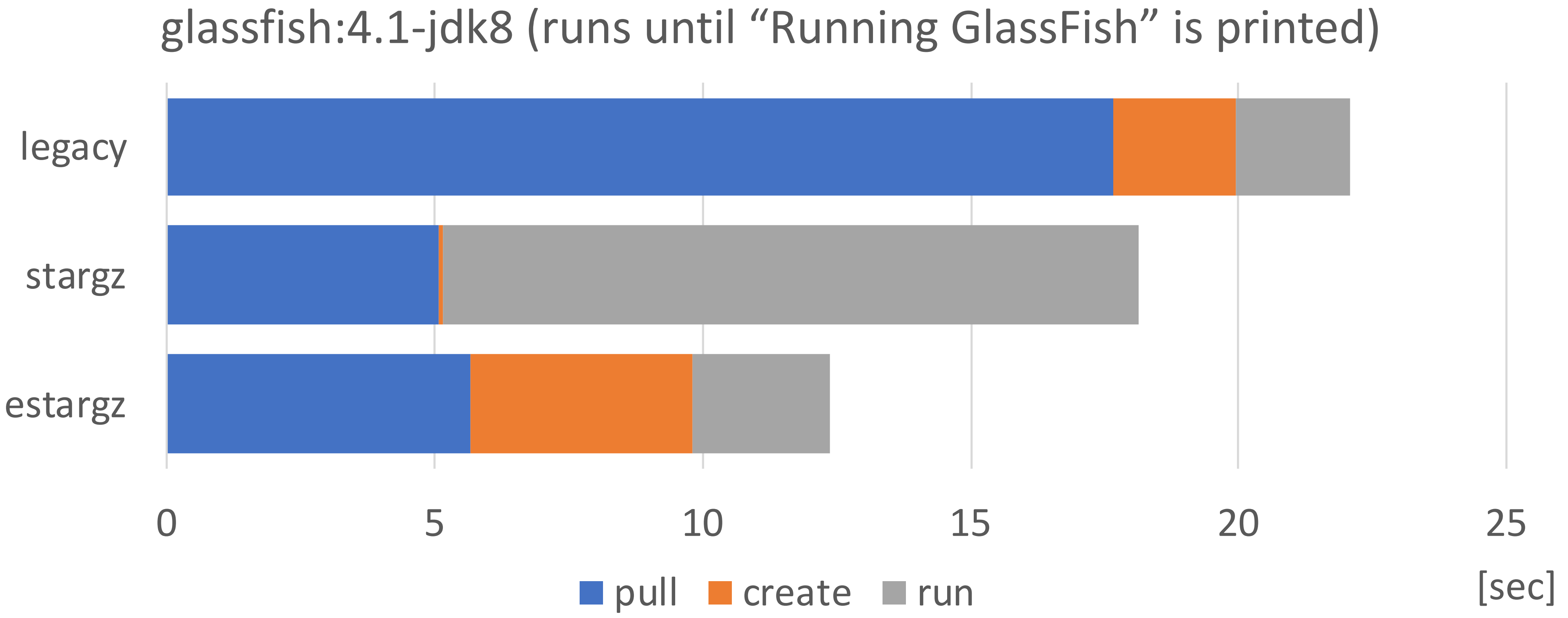
Credit to Akihiro Suda (NTT) for discussion and experiment environment

# Time to take for container startup



Credit to Akihiro Suda (NTT) for discussion and experiment environment

# Time to take for container startup



Credit to Akihiro Suda (NTT) for discussion and experiment environment

# Expected use-cases

**Speeding up base image distribution on image build**

- Especially for temporary base images of “dev” stages in multi-stage build
  - won’t be included in the result image
  - https://github.com/moby/buildkit/pull/1402

**Speeding up dev pipeline (or building/testing environment)**

- The initial motivation in Go community to invent stargz was to speed up the builder image distribution in their build system
  - https://github.com/golang/go/issues/30829

**Sharing large scientific software stack (e.g. ML frameworks)**

- For example, ML frameworks tend to be large (> 1GB)

**Improving cold start performance (e.g. Serverless)**

- But needs more investigation
  - https://github.com/knative/serving/issues/5913

**Stargz Snapshotter is still in early stage**

- ➢ Needs more performance improvements for the filesystem
- ➢ Lazy pull performance seems to be affected by the internet condition (e.g. CDN), etc.
- ➢ Be careful for the fault tolerance until the layer contents are fully cached
- ➢ …

**Feedbacks/comments are always welcome!**

# Other OCI-alternative lazy image distribution

**Slacker**: https://www.usenix.org/conference/fast16/technical-sessions/presentation/harter

- Uses NFS infra for the distribution of rootfs snapshots of containers
- Registries are used for sharing snapshot IDs among hosts

**CernVM-FS:** https://cvmfs.readthedocs.io/en/stable/

- FUSE Filesystem by CERN for sharing High Energy Physics (HEP) software on worldwide infrastructure
- Software stack can be mounted and lazily downloaded from CernVM-FS “repository” via HTTP
- Remote Snapshotter implementation for containerd
  - https://github.com/cvmfs/containerd-remote-snapshotter
- On-going discussion towards integration with Podman
  - https://github.com/containers/storage/issues/383

# Other OCI-alternative lazy image distribution

**Filegrain**: https://github.com/akihirosuda/filegrain

- Proposed by Akihiro Suda (NTT)
- OCI compliant image format but uses continuity manifests as layers
- An image can be mounted and files are pulled lazily
- Each file is treated as a content-addressable blob => de-duplication in file granuality

**On-going discussion towards "OCIv2"**: https://hackmd.io/@cyphar/ociv2-brainstorm

- Proposed by Aleksa Sarai (SUSE)
- Brainstorm is in progress (2020/07)
- Lazy fetch support, mountable filesystem are also in the scope

**crfs-plugin for fuse-overlayfs**: https://github.com/giuseppe/crfs-plugin

- Proposed by Giuseppe Scrivano (Red Hat)
- Plugin of fuse-overlayfs for mounting stargz layer

# Recap

- Pull is one of the time-consuming steps in the container lifecycle.

- Stargz Snapshotter, non-core subproject in containerd, is trying to solve it by lazy-pulling images leveraging stargz image by Google.
  - Standard compliant so can be pushed to and lazily pulled from standard registries
  - Workload-based runtime optimization is also held with eStargz

- There are also other OCI-alternative image distribution strategies in container ecosystem

**Feedbacks and suggestions are always welcome!**
https://github.com/containerd/stargz-snapshotter

KubeCon | CloudNativeCon

Europe 2020

*Virtual*

KEEP CLOUD NATIVE

# CONNECTED