Build and Run Containers With Lazy Pulling

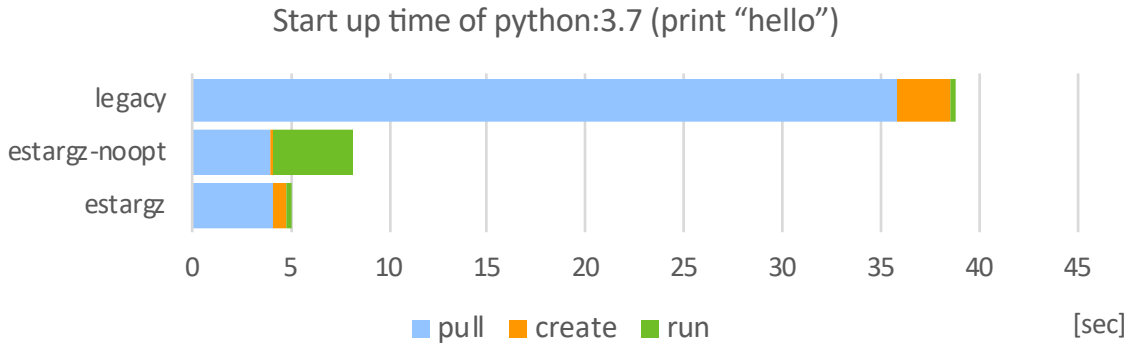
Adoption status of containerd Stargz Snapshotter and eStargz

FOSDEM 2021 (February 7)

Kohei Tokunaga, NTT Corporation

TL;DR

- 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
  - eStargz image based on Google stargz
  - Starndard compatibility, optimization and content verification
- **Collaboration in community**
  - eStargz is usable with: containerd, Kubernetes, BuildKit, Kaniko, go-containerregistry, ko, nerdctl
- **On-going in 2021:** Standardizing eStargz in OCI and improvements for stabilizing Stargz Snapshotter



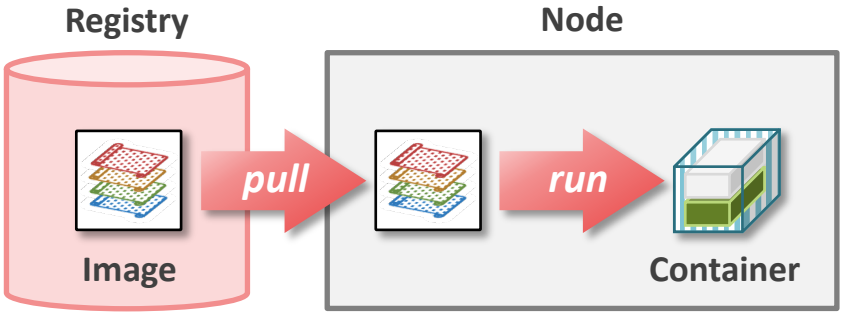
Host: EC2 Oregon (m5.2xlarge, Ubuntu 20.04)
Registry: GitHub Container Registry (ghcr.io)
Commit 7f45f74
(See detailed info in the later slides)

Pull and OCI/Docker image

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

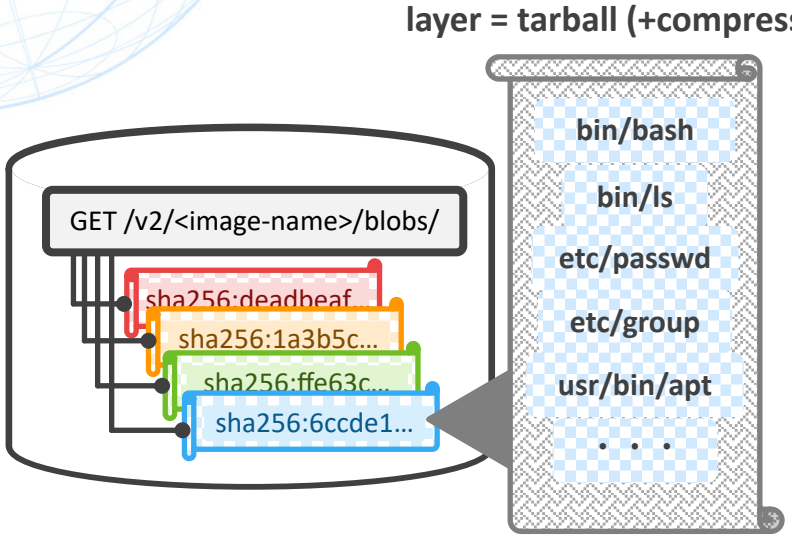


Workarounds are known but not enough

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

Problem on the current OCI/Docker image

A container is a set of ***tarball layers***
A container can't be started until the all layers become locally available



- 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

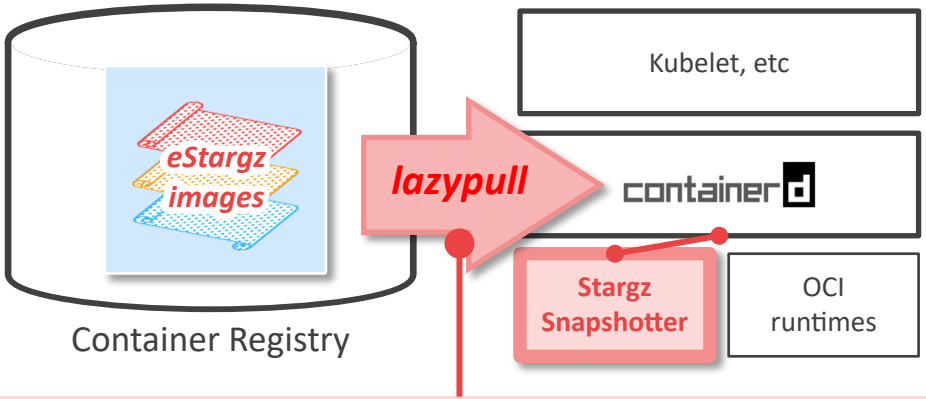
eStargz and Stargz Snapshotter

Standard-compatible lazy pulling with containerd

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

Stargz Snapshotter

- A plugin for containerd, developed in the non-core subproject
- Allows containerd to lazily pull **eStargz image from standard registry**
- eStargz comes with **workload-based optimization** and **content verification**

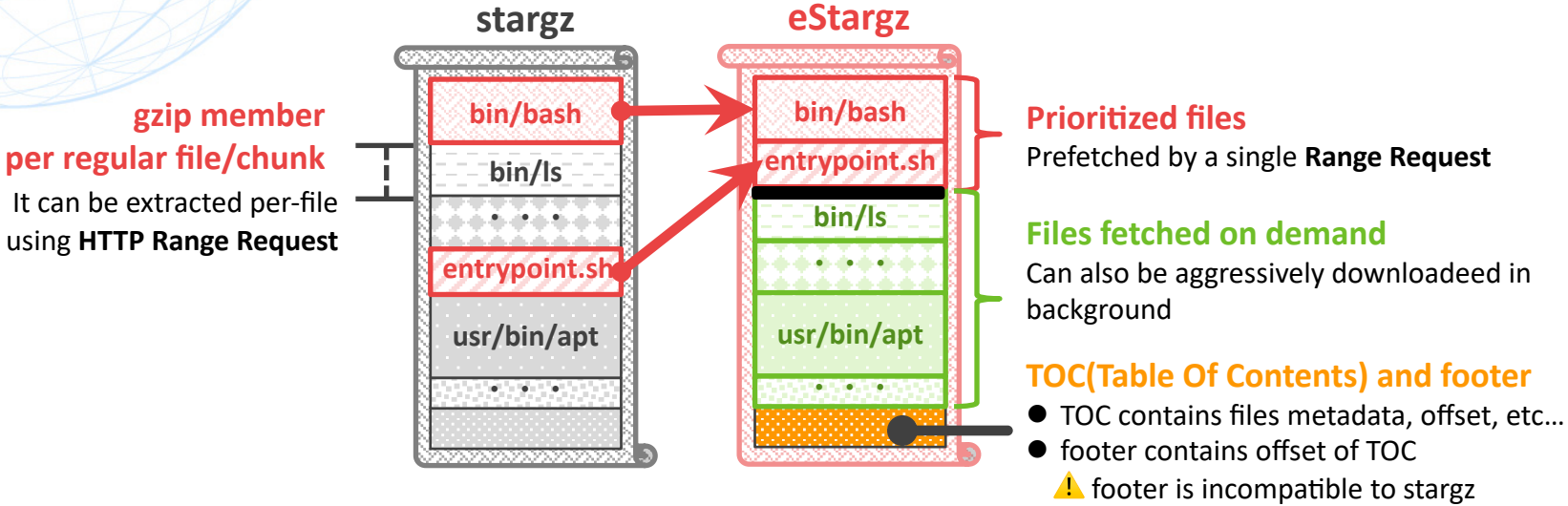


doesn't download the entire image on pull operation but fetches necessary chunks of contents on-demand

The structure of eStargz

- Seekable tar.gz and compatible to RFC 1952 (gzip) = **usable as a valid OCI/Docker image layer**
- Based on the **stargz** by Google CRFS (https://github.com/google/crfs)
- **eStargz** comes with performance optimization and content verification
- **Prioritized files** enables to prefetch and precache likely accessed files

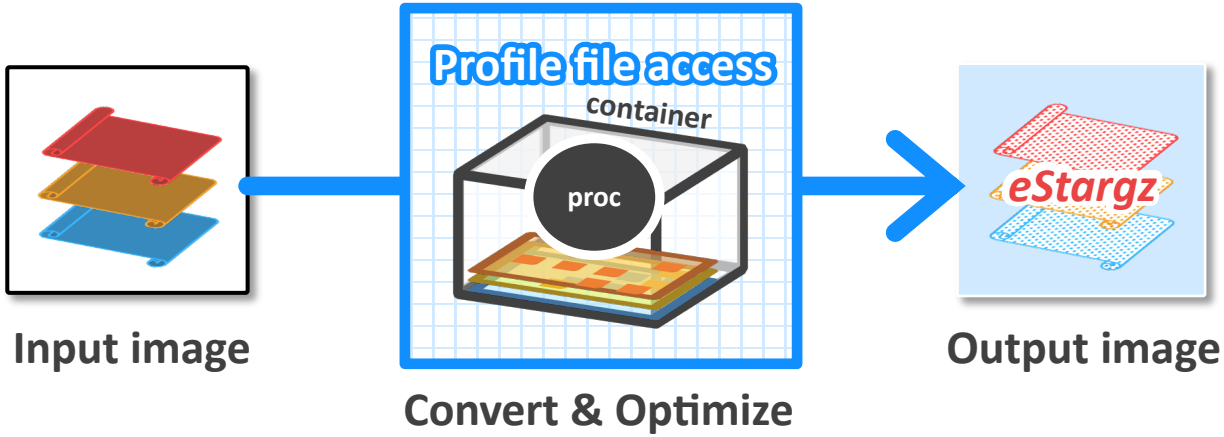
⚠ eStargz is incompatible to stargz: “footer” is changed to make eStargz compatible to RFC 1952



For more details: https://github.com/containerd/stargz-snapshotter/blob/master/docs/stargz-estargz.md

Workload-based Optimization of eStargz

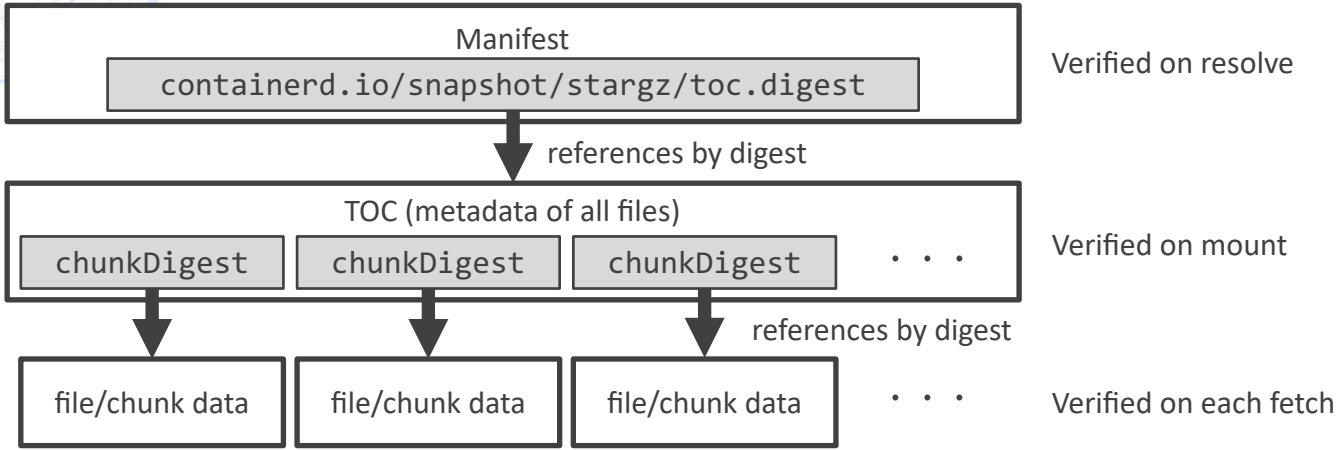
- Downloading each file/chunk on-demand costs extra overhead on each file access.
- Leveraging eStargz, CLI converter command **ctr-remote** provides **workload-based optimization**
  - Workload: entrypoint, envvar, etc… specified in Dockerfile (e.g. ENTRYPOINT)
- **ctr-remote** analyzes which files are likely accessed during runtime
  - Runs provided image and profiles all file accesses
  - Regards accessed files are also likely accessed during runtime (= **prioritized files**)
  - Stargz Snapshotter will prefetch these files when mounts this image



For more details: https://github.com/containerd/stargz-snapshotter/blob/master/docs/ctr-remote.md

Content Verification in eStargz

- Chunks are lazily pulled from registry on-demand
  - so they cannot verified when mounting the layer
- Chunks are “lazily” verified
  - TOC (metadata file) records digests per chunk
  - Each chunk can be verified when it’s fetched to the node
  - TOC itself is verified when mounting that layer using the digest written in the manifest



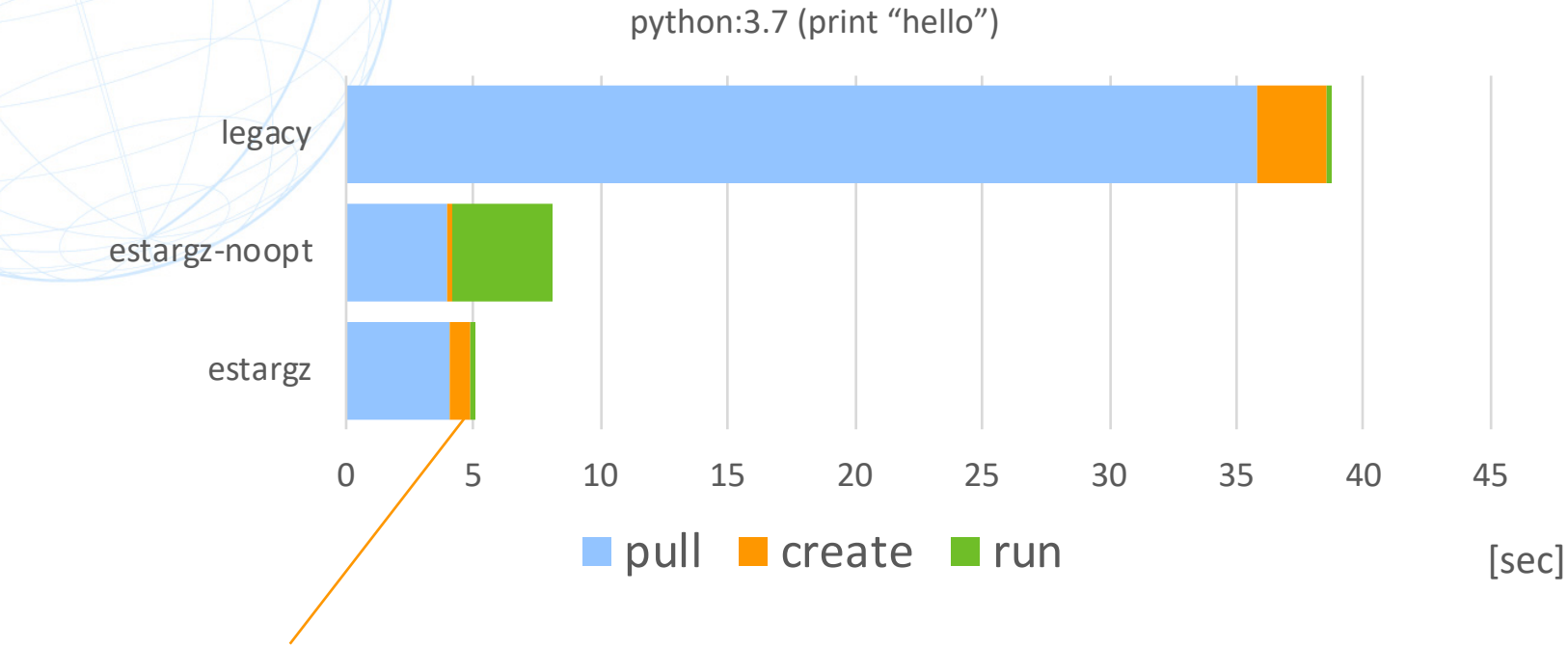
For more details: https://github.com/containerd/stargz-snapshotter/blob/master/docs/verification.md
(the above figure is from this doc)

Time to take for container startup

- Measures the container startup time which includes:
  - Pulling an image from GitHub Container Registry
  - For language container, running “print hello world” program in the container
  - For server container, 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: GitHub Container Registry (ghcr.io)
- Target commit: 7f45f7438617728dd06bc9853afb5e42c1d3d5a3

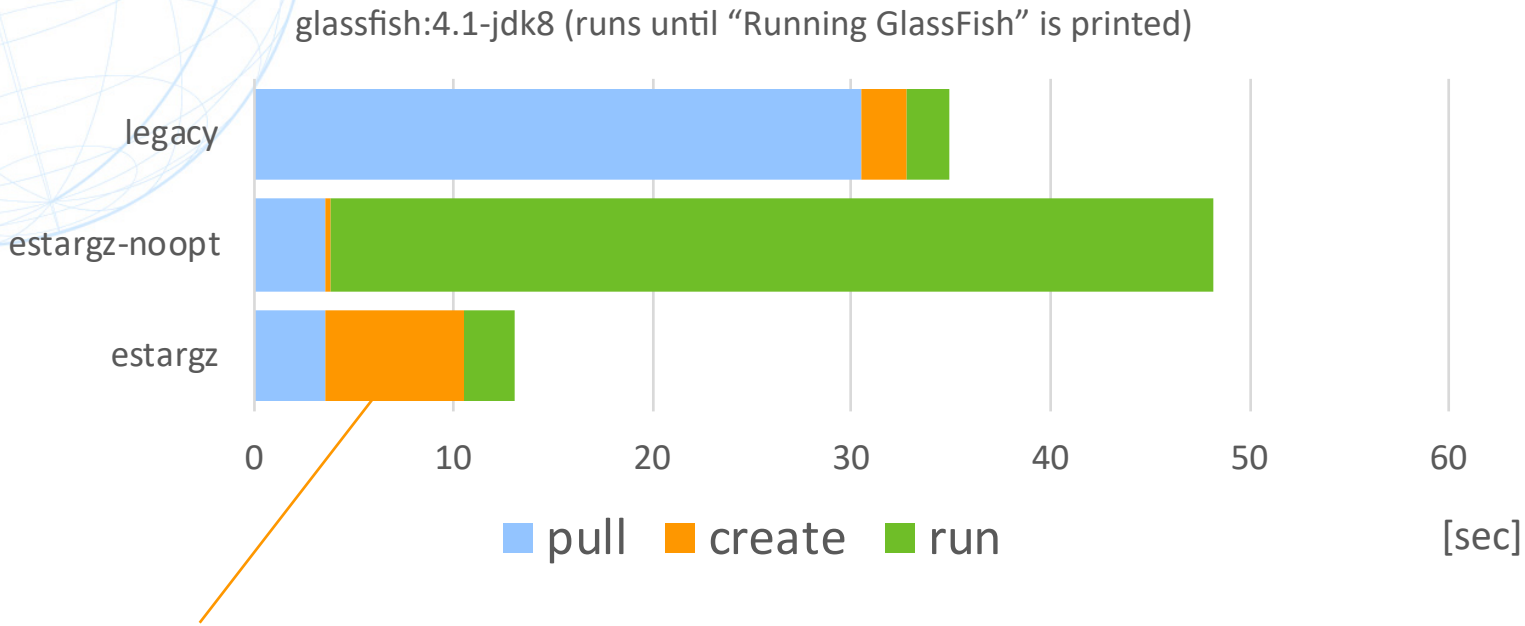
[Harter et al. 2016] Tyler Harter, Brandon Salmon, Rose Liu, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau. "Slacker: Fast Distribution with Lazy Docker Containers". 14th USENIX Conference on File and Storage Technologies (FAST ’16). February 22–25, 2016, Santa Clara, CA, USA

Time to take for container startup



Waits for prefetch completion

Time to take for container startup



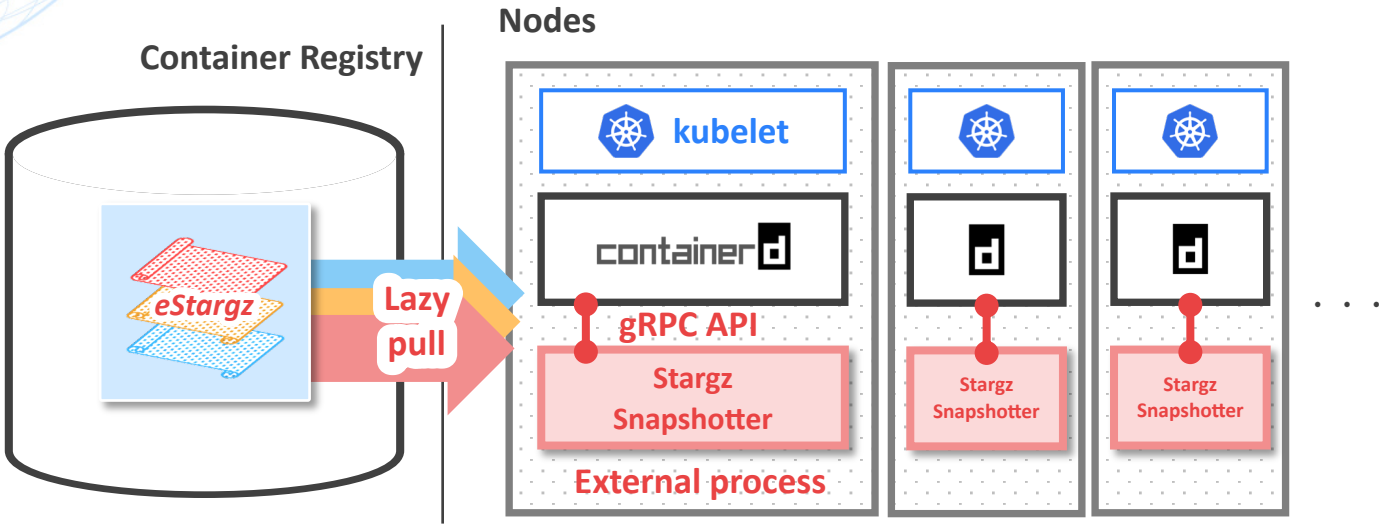
Waits for prefetch completion

Collaboration in community

eStargz on Kubernetes

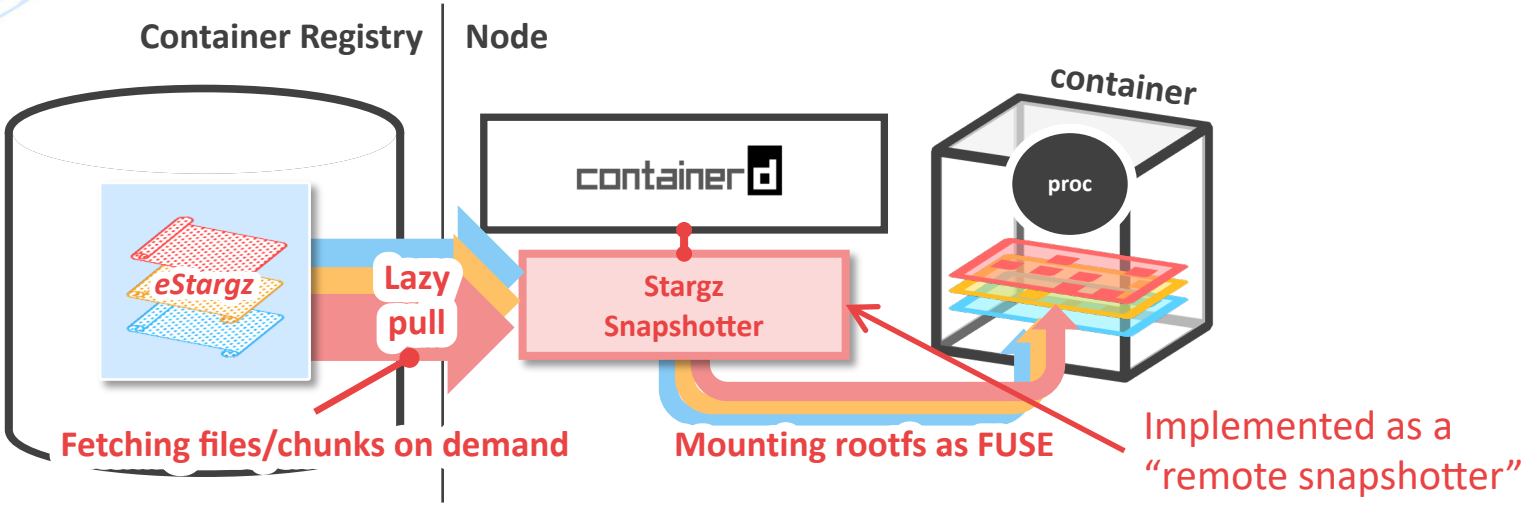
- Lazy pulling can be enabled on Kubernetes using Stargz Snapshotter, without patches
  - containerd is required as a CRI runtime
- Stargz Snapshotter needs to run on each node and containerd needs to be configured to recognize it
- Real-world use-case at CERN for speeding up analysis pipeline [1] (13x faster pull for 5GB image)

[1] Ricardo Rocha & Spyridon Trigazis, CERN. “Speeding Up Analysis Pipelines with Remote Container Images”. KubeCon+CloudNativeCon 2020 NA. https://sched.co/ekDj



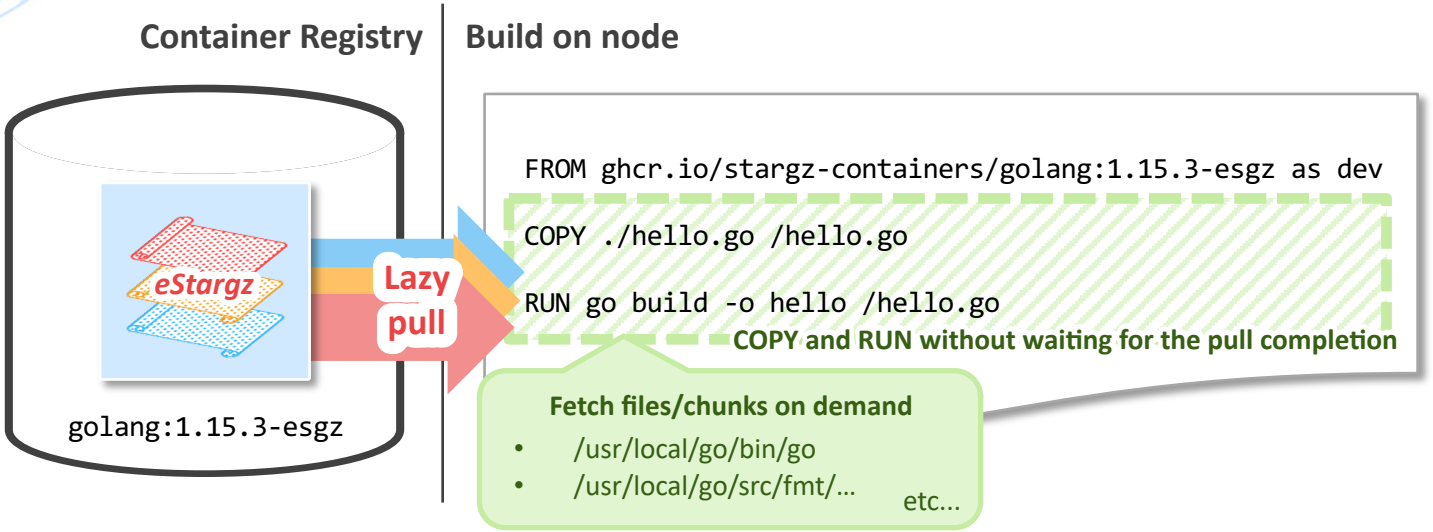
eStargz on containerd

- Stargz Snapshotter enables lazy pulling of eStargz on containerd
  - Implemented as a “remote snapshotter” plugin
- Mounts rootfs snapshots as FUSE and downloads accessed file contents on-demand
- **nerdctl** (Docker-compatible CLI for containerd; https://github.com/AkihiroSuda/nerdctl) supports lazy pulling of eStargz on containerd



eStargz on BuildKit

- BuildKit > v0.8.0 experimentally supports lazy pulling of eStargz base images during build
  - FROM instruction is skipped and chunks are lazily pulled on-demand during COPY/RUN/etc.
- Can shorten the time of build e.g. on temporary (and fresh) CI instances with big base images.
- More details at blog: https://medium.com/nttlabs/buildkit-lazypull-66c37690963f
  - speeding up building ”hello world” image from tens of seconds to a few seconds at the best



Tools start to support eStargz creation (1/2)

ctr-remote https://github.com/containerd/stargz-snapshotter/tree/master/cmd/ctr-remote

- Image converter developed in Stargz Snapshotter project
- Converts image to eStargz
- Comes with workload-based optimization

kaniko https://github.com/GoogleContainerTools/kaniko

- Container image builder by Google
- Builds eStargz image (no optimization)
- Base images need to be pre-converted to eStargz
- GGCR_EXPERIMENT_ESTARGZ=1 is needed

nerdctl https://github.com/AkihiroSuda/nerdctl

- Docker-compatible CLI for containerd by Akihiro Suda, NTT
- Converts image to eStargz
- Comes with manual optimization (i.e. manually specifying prioritized files)

Tools start to support eStargz creation (2/2)

go-containerregistry and crane CLI https://github.com/google/go-containerregistry

- Container registry client library and CLI by Google
- Converts image to eStargz
- Comes with manual optimization (i.e. manually specifying prioritized files)
- GGCR_EXPERIMENT_ESTARGZ=1 is needed

ko https://github.com/google/ko

- Build & Deployment tool of Go application on Kubernetes, by Google
- Builds eStargz image (no optimization)
- Base images need to be pre-converted to eStargz
- GGCR_EXPERIMENT_ESTARGZ=1 is needed

eStargz in 2021

Updates will come in 2021

NTT

Standardizing eStargz

https://github.com/opencontainers/image-spec/issues/815

- eStargz is proposed to OCI Image Spec
- Discussion is on-going
- Backward-compatible extensions
  - Optional extension to application/vnd.oci.image.layer.v1.tar+gz
  - Optional annotation for content verification

Features and improvements for stabilizing Stargz Snapshotter

- Higher availability of Stargz Snapshotter (mounting images from multiple backend registries)
- Improvements on memory consumption of Stargz Snapshotter
- Speeding up image conversion
- Static optimization of images
- etc…

Summary

- 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
  - eStargz image based on Google stargz
  - Starndard compatibility, optimization and content verification

- **Collaboration in community**
  - eStargz on various platforms: Kubernetes, containerd and BuildKit
  - go-containerregistry, ko, kaniko and nerdctl start to support eStargz creation

- **On-going in 2021:** Standardizing eStargz in OCI and improvements for stabilizing Stargz Snapshotter

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