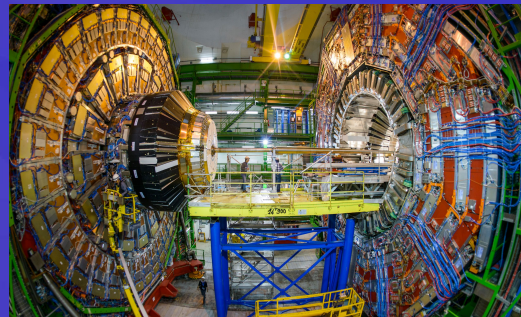
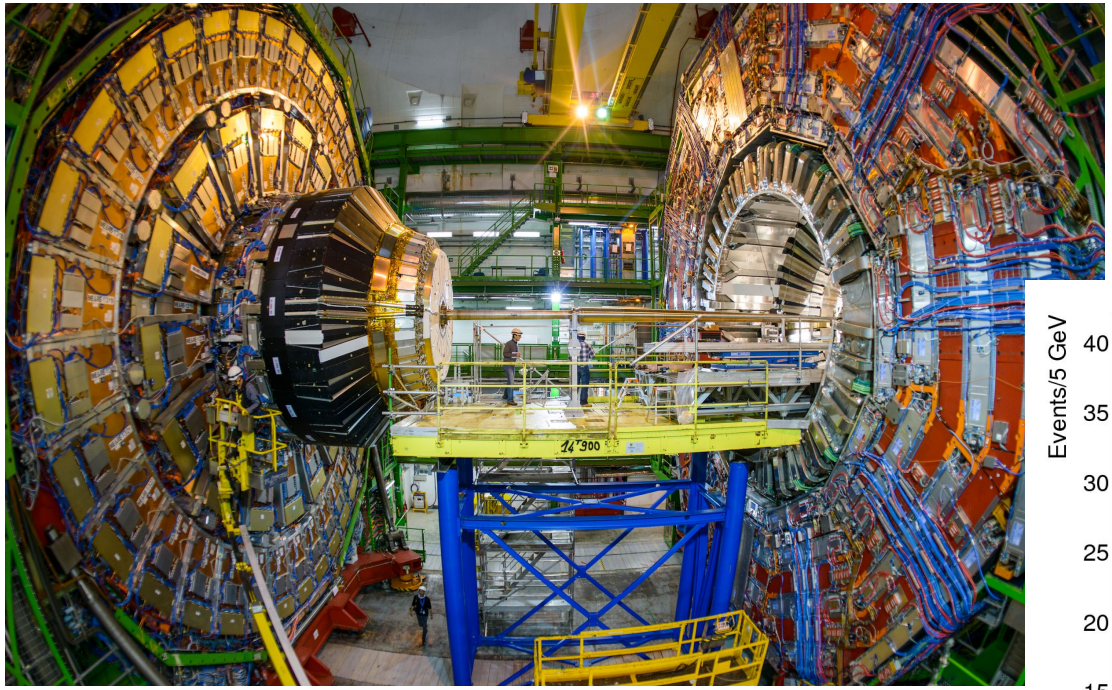
Speeding Up Analysis Pipelines with Remote Container Images

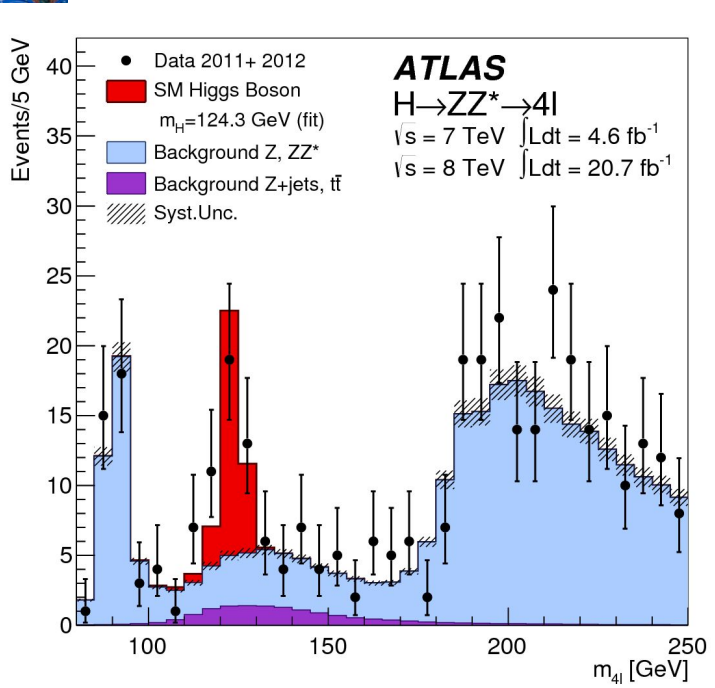
Ricardo Rocha @ahcorporto , CERN

Spyridon Trigazis @strigazi , CERN



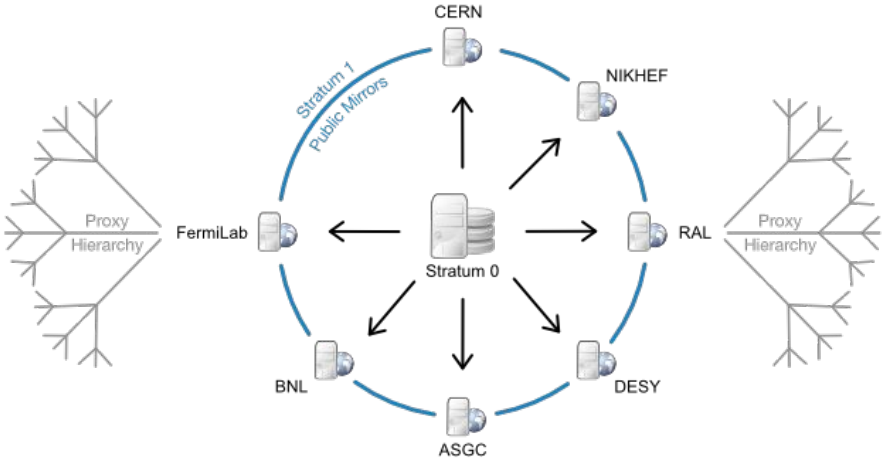

CMS
CERN
LHC
[Large Hadron Collider]
ALICE
LHCb
ATLAS




ATLAS
H→ZZ*→4l
√s = 7 TeV ∫Ldt = 4.6 fb-1
√s = 8 TeV ∫Ldt = 20.7 fb-1
Data 2011+ 2012
SM Higgs Boson
mH=124.3 GeV (fit)
Background Z, ZZ*
Background Z+jets, tt̄
Syst.Unc.
Events/5 GeV
0
5
10
15
20
25
30
35
40
100
150
200
250
m4l [GeV]


Running jobs: 965118
Active CPU cores: 795836
Transfer rate: 18.35 GiB/sec

Software Distribution



CernVM-FS (CVMFS)

https://cernvm.cern.ch/fs/

Scalable software distribution for the Grid

POSIX read-only filesystem in user space

Aggressive caching, HTTP based

Question

Can we achieve the same efficiency for containerized workloads?

More Questions

Software packaged in container images

- How can we speed up container creation and startup?
- Images of 10s of gigabytes!

How can we reduce / optimize network usage?

Cluster auto scaling is a major topic

- How can we properly handle this with huge images?

Some History

FAST 16, Slacker: Fast Distribution with Lazy Docker Containers

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

Docker CVMFS Graph Driver

https://github.com/cvmfs/docker-graphdriver

Lazy Pulling

Ongoing Work

Build on the existing CVMFS infrastructure for image distribution

https://github.com/cvmfs/containerd-remote-snapshotter

Today's presentation will focus on a more generic deployment

(e)stargz + distributed container registries

Stargz Remote Snapshotter

Containerd Remote Snapshotter

Based on seekable tar.gz (stargz) [0][1]

Proposed by Kohei Tokunaga, NTT [2]

Indexed files per image layer

Fuse mount per image layer

gRPC plugin for containerd

[0] https://github.com/containerd/stargz-snapshotter
[1] https://github.com/google/crfs
[2] https://kccnceu20.sched.com/event/ZepQ

Stargz Remote Snapshotter

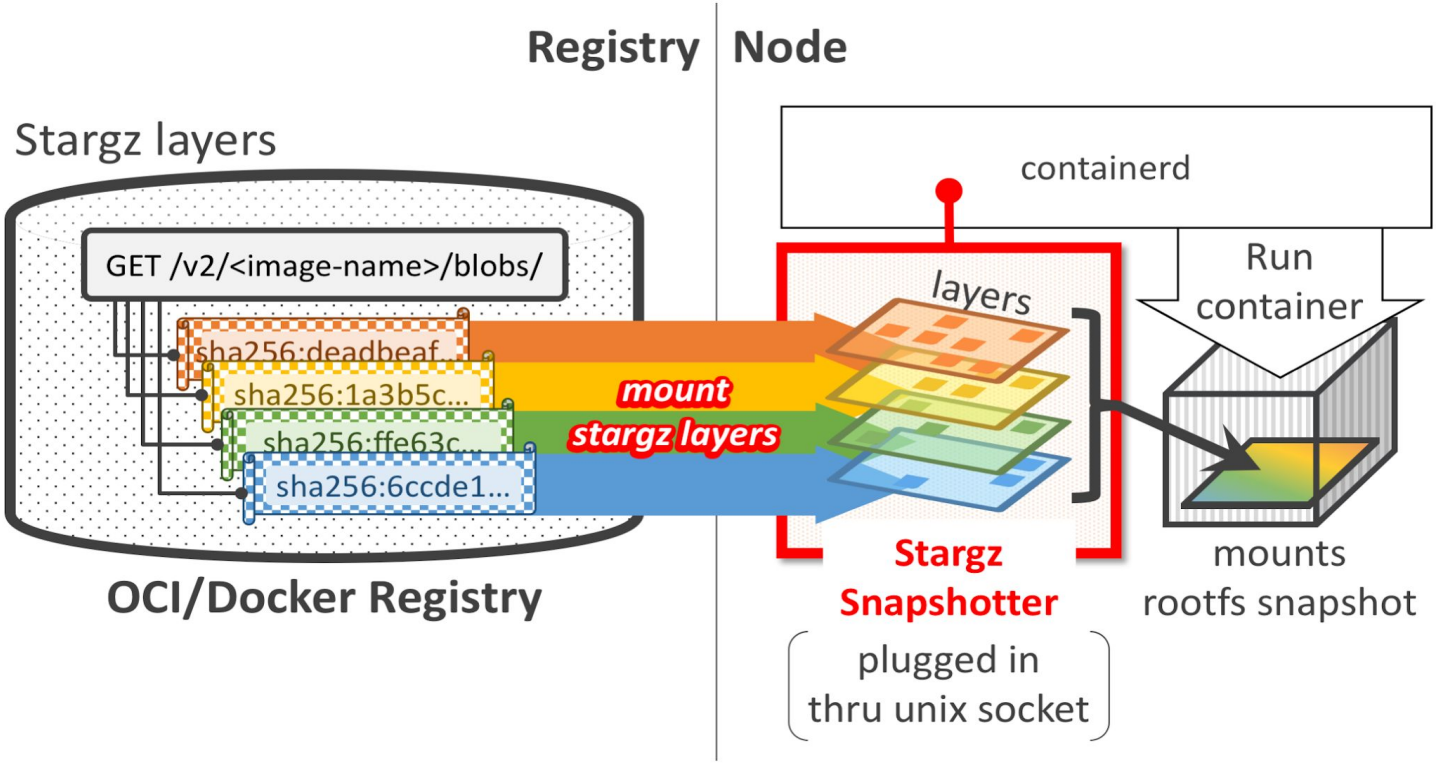


Image credit: The containerd Authors

Runtime statistics

Exec time, RAM, Network ingress

atlas/athena:21.0.15_100.0.2
17.2GB / 5.43GB

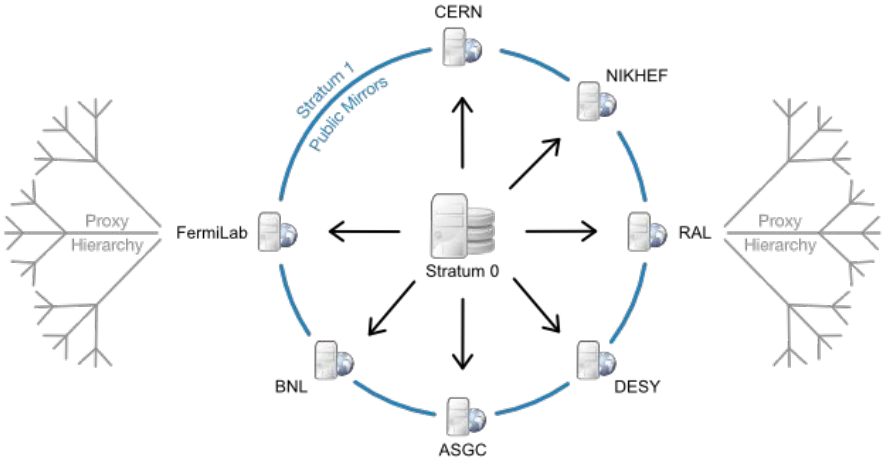
strigazi/athena:21.0.15_100.0.2-esgz-bash-version
17.2GB 5.56GB

mode	pulling time	RAM Containerd/ stanpshotter	Ingress on node	execution time workload
native	3m37s	257MB	5.84GB	7m15s
esgz	16s	1360MB	0.84GB	8m14s

- Fast startup time
- low network traffic (workload dependent)
- Memory consumption to be investigated
- 45m to convert to esgz

Demo

Software Distribution



CernVM-FS (CVMFS)

https://cernvm.cern.ch/fs/

Scalable software distribution for the Grid

POSIX read-only filesystem in user space

Aggressive caching, HTTP based

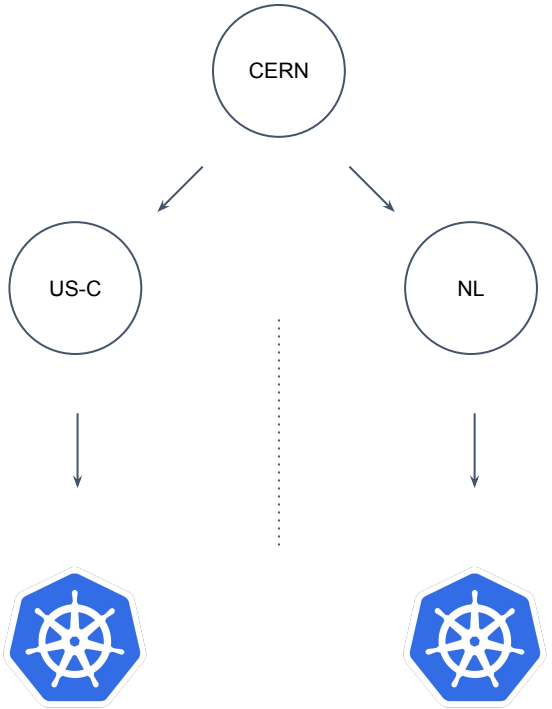
Registry Distribution

Registry implementation choice

Proxy cache and/or Replication

(e)stargz support

OCI Artifact support is a plus



Status

Stargz remote snapshotter is already functional

- Super fast startup times
- Reduce network usage
- Low cpu overhead

Some registries do not support HTTP range queries RFC7233 [0][1]

- Gitlab Registry which we use extensively at CERN

[0] https://docs.docker.com/registry/spec/api/#fetch-blob-part

[1] https://tools.ietf.org/html/rfc7233#section-2.3

Improvements

Speed up image optimization, currently single core / serial

Allow (e)stargz builds with optimized base images

Mounted / external data during optimization step

Smaller Issues

- containerd fallback when remote snapshotter is down
- Further investigating needed for Harbor handling of large layers

Thanks

Akihiro Suda and Kohei Tokunaga, NTT

CVMFS Team at CERN

CERN Cloud Team

Participants of the May 2019 CERN Workshop on SW Distribution
https://indico.cern.ch/event/788994


HELM
KV
KEEP CLOUD NATIVE
KubeCon
CloudNativeCon
North America 2020
Virtual
EVERYWHERE