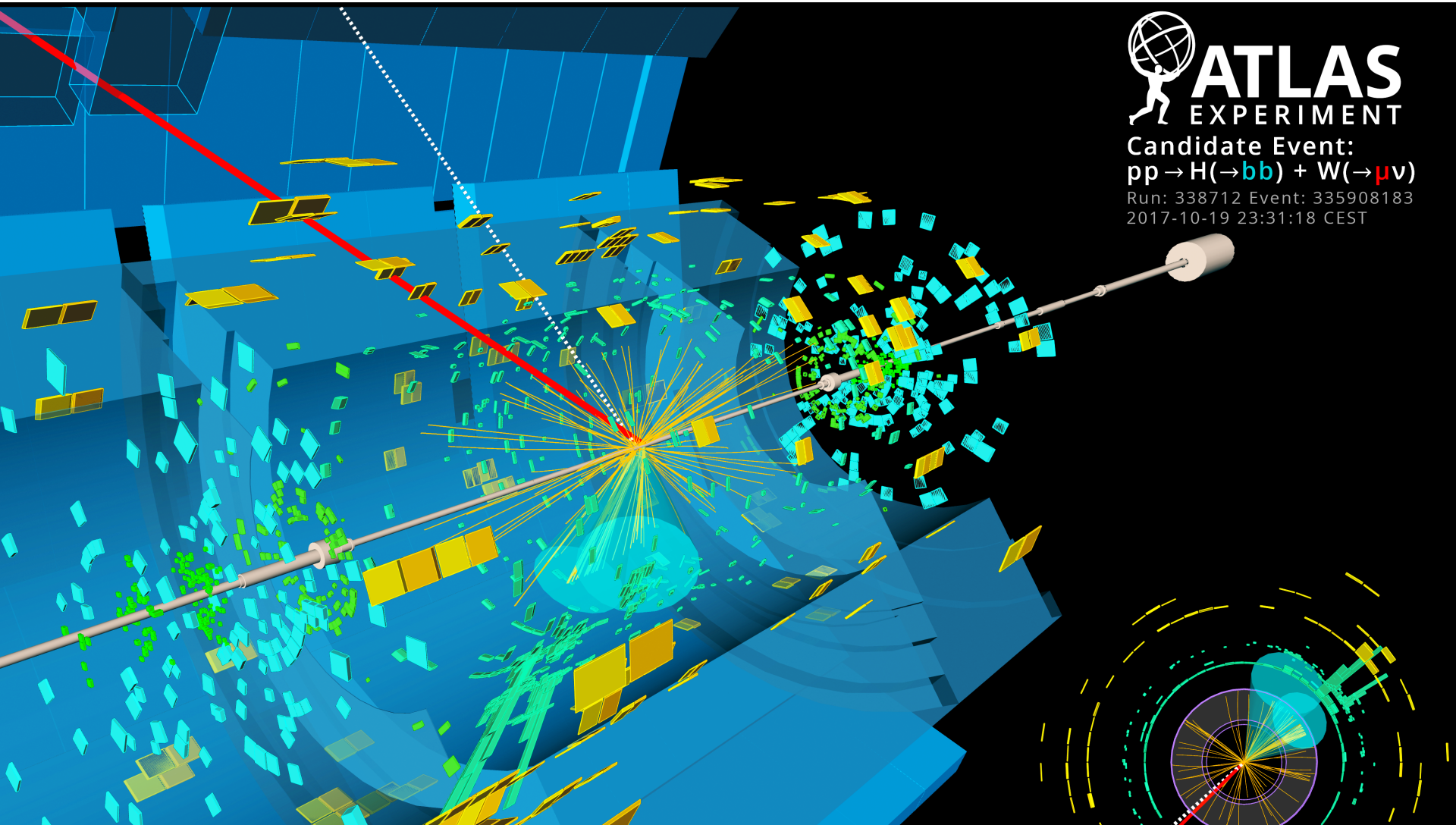
Containers for HEP

Lukas Heinrich 2019/05/17

HEP Computing — In a Nutshell

Foundational Principle: repeated experiment, i.e. proton collisions

- each event is independent of the other
- to zero-th order HEP computing is embarassingly parallel - great for distributed computing



HEP Computing — In a Nutshell

Three Scopes of Computing

1. Online Software:

Main Problem: too much data from collisions → real-time, distributed computed compute ("Trigger") to decide what to store persistently.

Software-based Compute Clusters need to be provisioned fast. Looking at Kubernetes, Mesos, etc...

2. Offline Software

The data we do write out needs to be pre-processed. "Reconstruct" what happened in the event (which particles went in what direction?) from raw readout data

3. Analysis Sofware

code that looks at reconstructed data to derive physics results

HEP Computing — In a Nutshell

Three Scopes of Computing

1. Online Software:

Main Problem: too much data fro… …ne, distributed computed compute ("Trig… …store persistently.

Software-based C… …sters need to be provisioned fast. Looking at Kubernetes, Mes…, etc...

highly stable, mission-critical code
bugs = data forever lost

2. Offline Software

The data we do write out need… …essed. "Reconstruct" what happened in the event… …went in what direction?) from raw readout data

core software written
by physisict / sw-eng hybrids

3. Analysis Sofware

code that looks at reconstr… …tive physics results

grad student bashochism

HEP Computing — In a Nutshell

Idea: easier to send code to data than vice versa.

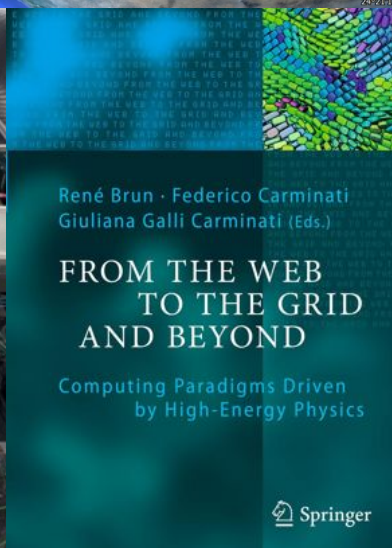
Worldwide LHC Computing Grid (WLCG)

Federation of Independent Compute Clusters from Universities, Research Labs, etc

- **necessarily heterogeneous**
  - **small univ. clusters**
  - **leadership class HPCs**

Misssion:

Keep it all working for all use-cases from well-oild s/w to one-off users

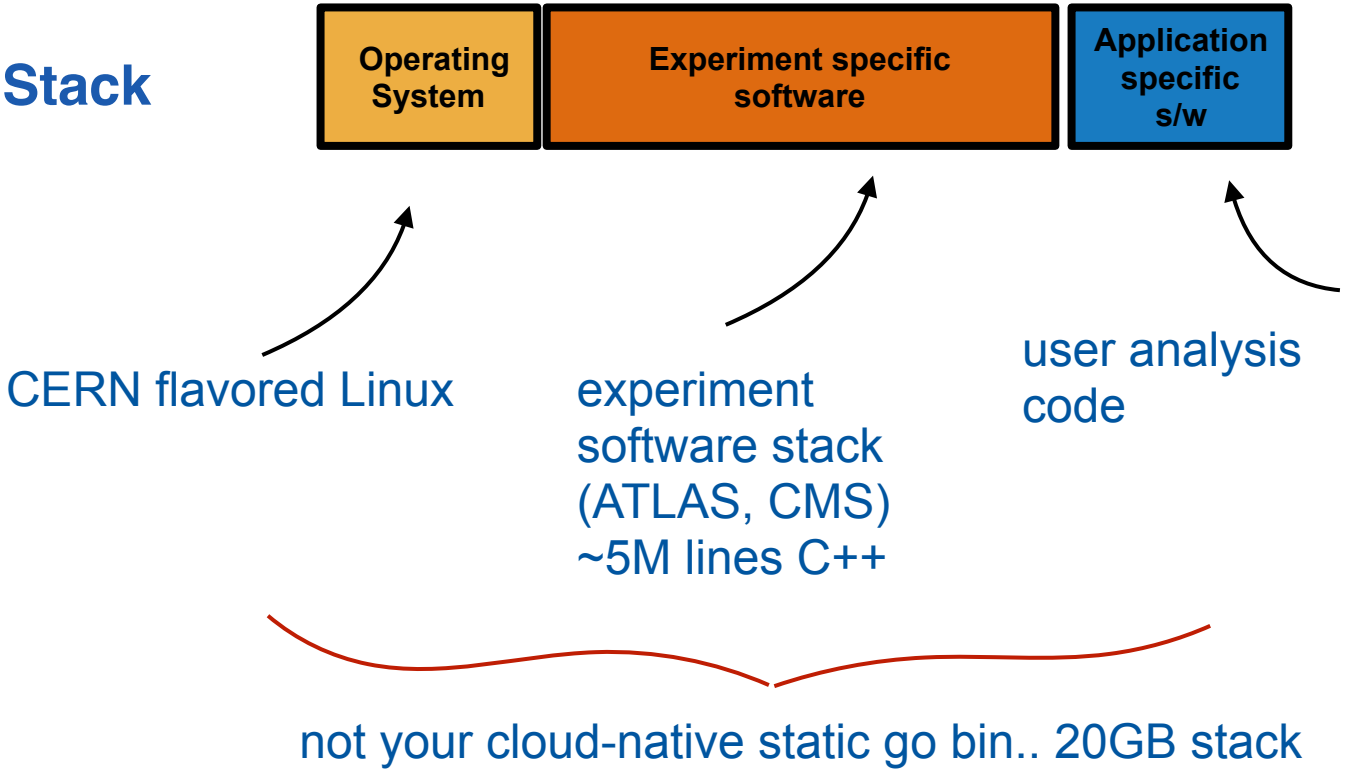


Distributed Computing ↔ Software Distribution

Idea: easier to **send code** to data than vice versa.

We need to materialize the software stack on the remote machines somehow.

Our traditional Stack



Distributed Computing ↔ Software Distribution

Idea: easier to **send code** to data than vice versa.

We need to materialize the software stack on the remote machines somehow.

On a compute node...

works very reliably for bulk workloads

immutable for provisioned node

global read-only filesystem (CVMFS) few have update rights

app layer downloaded on demand by payload

Distributed Computing ↔ Software Distribution

Idea: easier to **send code** to data than vice versa.

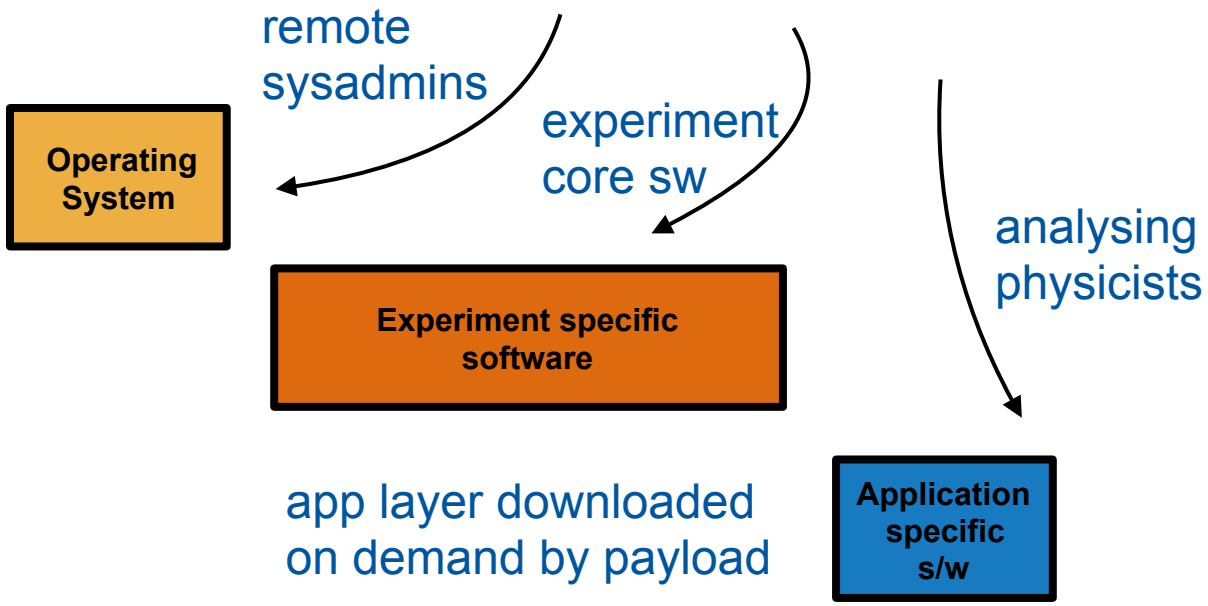
We need to materialize the software stack on the remote machines somehow.

Challenge: keep in sync across three parties to assemble filesystem view needed by analysis

On a compute node

immutable for provisioned node

global read-only filesystem (CVMFS) few have update rights



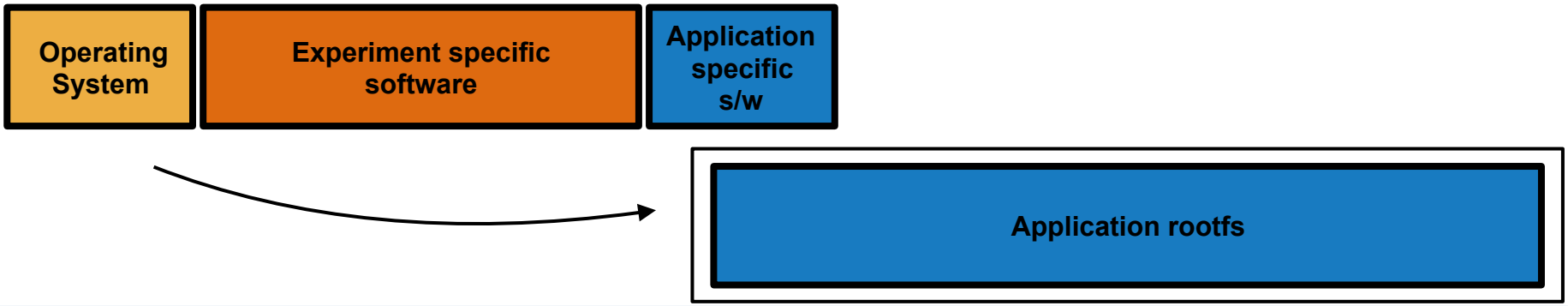
Distributed Computing ↔ Software Distribution

Idea: easier to **send code** to data than vice versa.

We need to materialize the software stack on the remote machines somehow.

Alternative of course: distribute software with as few assumptions as possible, i.e. kernel and just let user specify full rootfs

Sounds good .. but can we do it?



Menu

16443956 task: **user.lheinric.scweek.v1/** aipanda167 | 12-13 02:55:02, Reload

16443956 task: user.lheinric.scweek.v1/

Task ID	Jobset	Type	Working Group	User	Destination	Task status	Nevents | used	HS06*sec Expected Total done failed	Ninputfiles | finished | failed	Created	Modified	C
16443956	3800	analy		Lukas Alexander Heinrich		done	0 | 0 (%)	None 920 920 0	11 | 11 (100%)	2018-12-13 02:18:14	2018-12-13 02:36:07	1

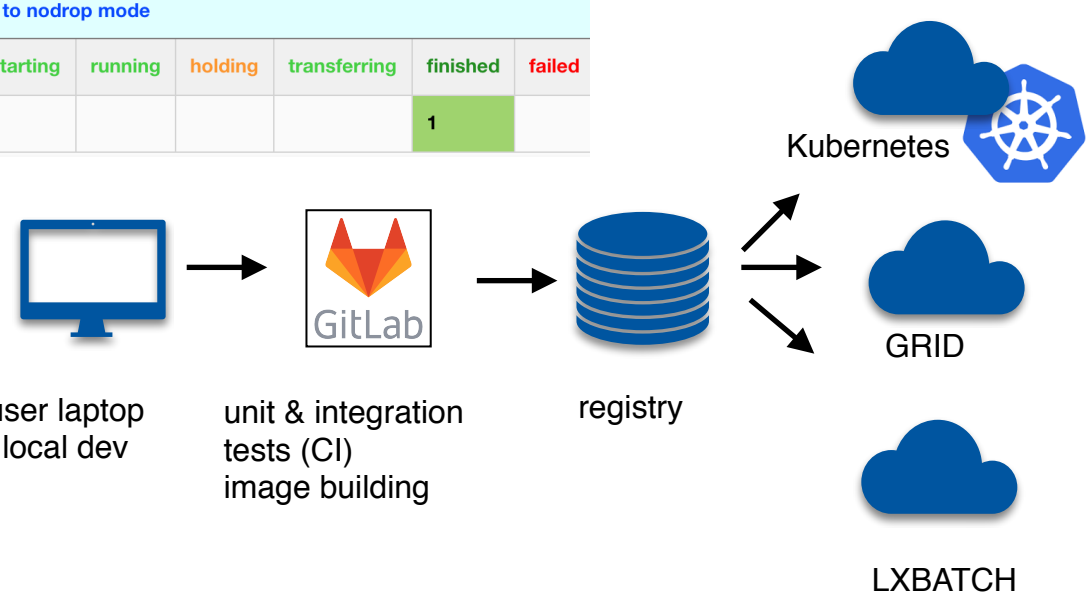
Job parameters

"
--containerImage docker://busybox
-a jobO.d4e92bca-3544-496a-b03a-bf2299a61d40.tar.gz
-j "" --sourceURL https://aipanda078.cern.ch:25443
-o "{'out.json': 'user.lheinric.$JEDITASKID._${SN/P}.out.json'}"

Jump to | Open plot

obs | Switch to nodrop mode

ted	sent	starting	running	holding	transferring	finished	failed
						1	



Distributed Computing ↔ Software Distribution

Benefits of Containers clear:

- **rootfs reproducibility**
- **simplified Ops on site admin site**
- **looser coupling between teams**
- **allow use of shared compute resources for non-traditional workloads. Prime ex: Machine Learning (incl hw acceleration)...**

Distributed Computing ↔ Software Distribution

... but we need a few ingredients to make this work within an academic setting.

- **Users must be able to build & run images in their normal env:**
  - **containers on multi-tenant systems (rootless!)**
  - **dito: image building**
- **We must find an efficient way to distribute out software stacks globally**

Rootless Containers

Vast Majority of HEP computing (interactive and batch) happens on shared resources:

- **shared clusters to which users have ssh-logins.**
- **either interactive work on "login nodes" or submit to batch systems (SLURM, HTCondor)**

Building Container Images

Currently Image Building is not supported on shared clusters

- **rely on users building images**
  - **in GItLab CI / their laptop**
- **also not solved in academic runtimes (singularity)**

- **RHEL7-based distro.**
  - **user-namespaces enabled**
  - **needs newuidmap / newguidmap / updated shadow-utils**

User Namespaces

Shared Systems have managed logins (LDAP, ...)

- **how can we assign uid ranges to users automatically?**
- **uid exhaustion?**

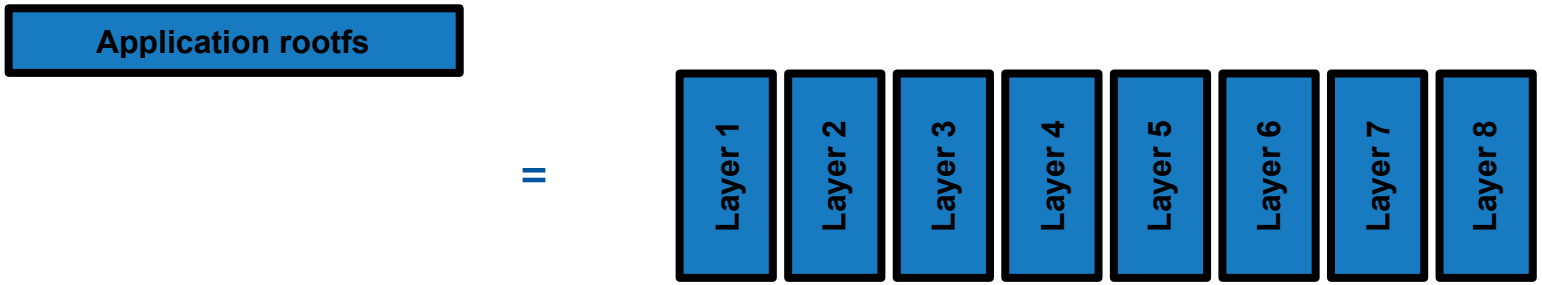
Running Containers

- Currently best bet for Container Runtime is Singularity
  - deep penetration / install base in academia
  - however rootless mode of singularity afaik
    not OCI compatible
  - from 3.0 requires expensive translation into SIF image format

- Container Execution ~same issues as img building
  - PoCs of rootless runc, containerd etc working but need to get into prod

- With current kernels in production we don't have overlay in userspace
  - what's needed for FUSE overlay (4.19? ...)

Image Distribution

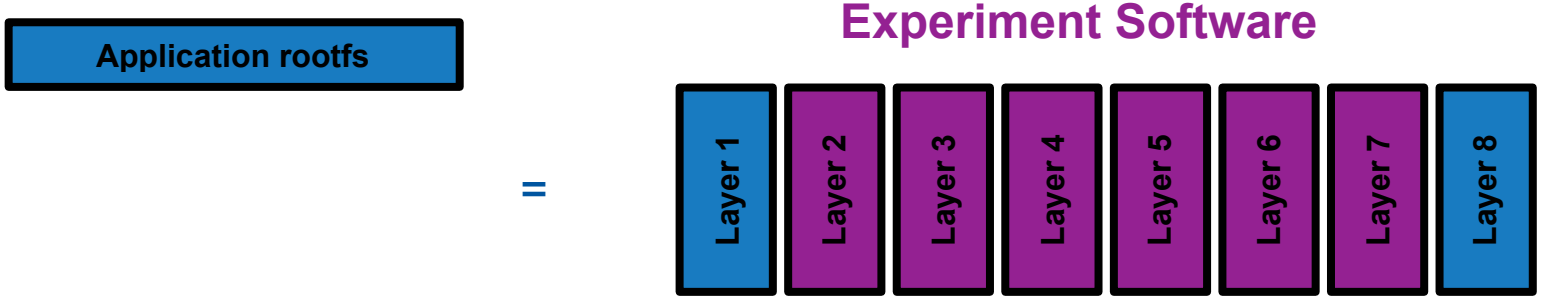


View the "image" not as a monolithic blob of layer data

- rather treat its manifest as a declaration of "intent" of what rootfs the user desires

- what's the best way to realize this filesystem on a remote host
- can we leverage the tech / experience we have?

Image Distribution



We know that images that users built will have significant overlap in the middle layers

- **90% of image size is in that middle part**
- **usually this layer is provided through a global read-only filesystem /cvmfs**
- **instead of exposing /cvmfs directly to users, can we distribute image files through /cvmfs?**
  - **best of both worlds: if /cvmfs available, use it as a CDN**
  - **if not available, pull full image**

Image Distribution Using a Global Read-only FS

When constructing rootfs, container runtimes needs first acquire image data locally on the host and unpack

Idea: instead of downloading layer tarballs just use directories on global read-only filesystem

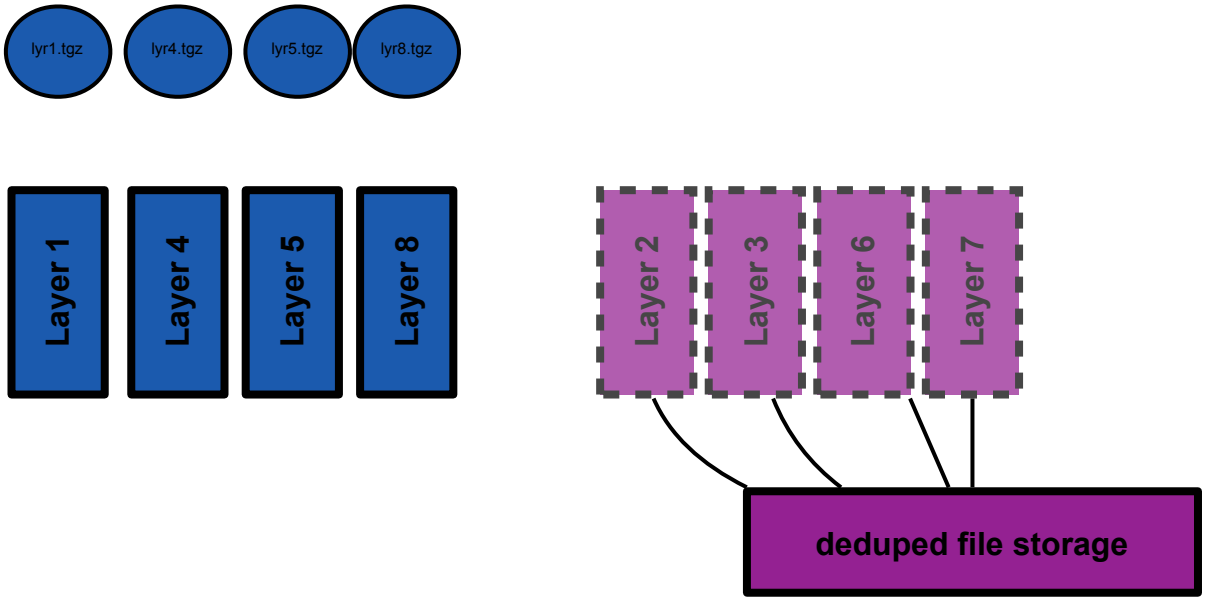
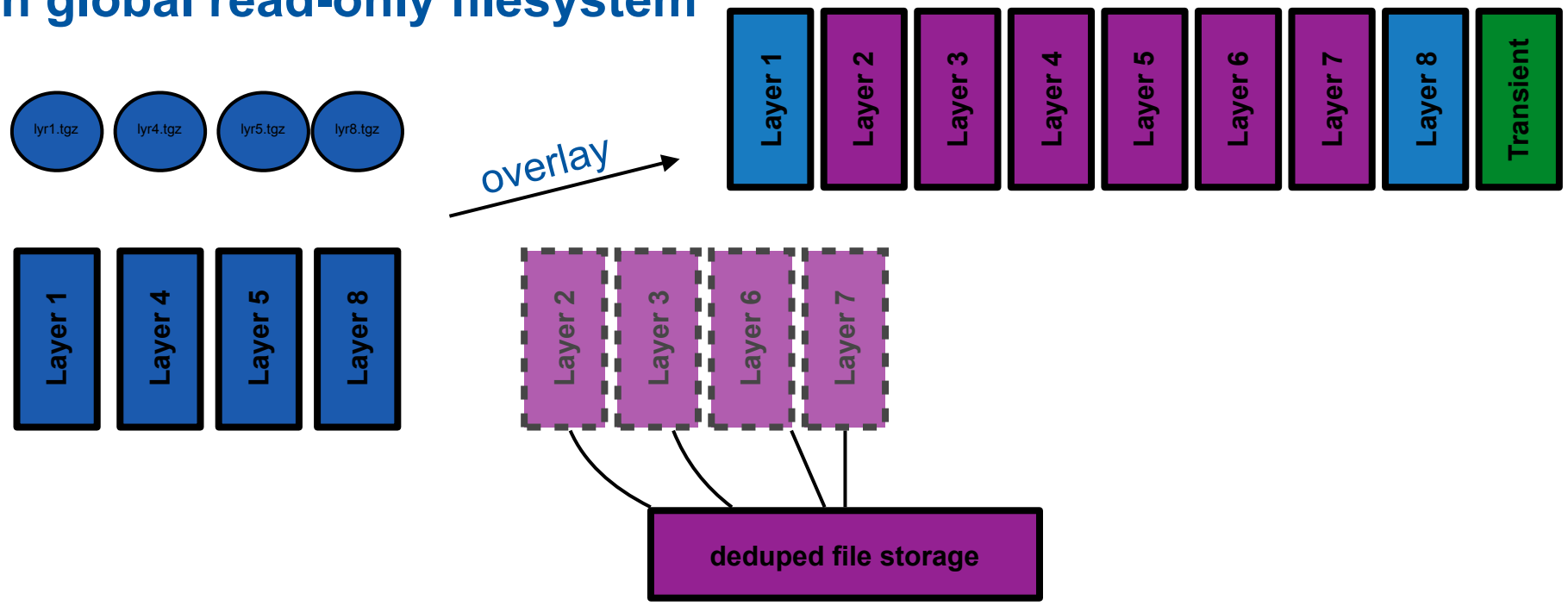


Image Distribution Using a Global Read-only FS

When constructing rootfs, container runtimes needs first acquire image data locally on the host and unpack

Idea: instead of downloading layer tarballs just use directories on global read-only filesystem



containerd / containerd

Watch 234 | Unstar 4,095 | Fork 808

Code | Issues 125 | Pull requests 32 | Projects 0 | Insights

remote filesystem snapshotter #2943

Edit | New issue

Open **lukasheinrich** opened this issue on 22 Jan · 18 comments

lukasheinrich commented on 22 Jan • edited

This is an issue to track / follow up on a call re: file-level image distribution through remote filesystems.

Slides for CERN use-case shown during meeting:
https://docs.google.com/presentation/d/1DJlRV9a445567EyRa265uemWv5zoDQ4o1CK-ZszpFLE/edit?usp=sharing

The goal is to support exploiting the existence of unpacked layers on remote filesystems (FUSE mounted, possibly read-only) to reduce the amount of data transferred during image pull. A candidate filesystem could be CVMFS (CERN VM Filesystem: https://github.com/cvmfs/cvmfs)

The current approach in containerd has an ordering where

Assignees

No one assigned

Labels

None yet

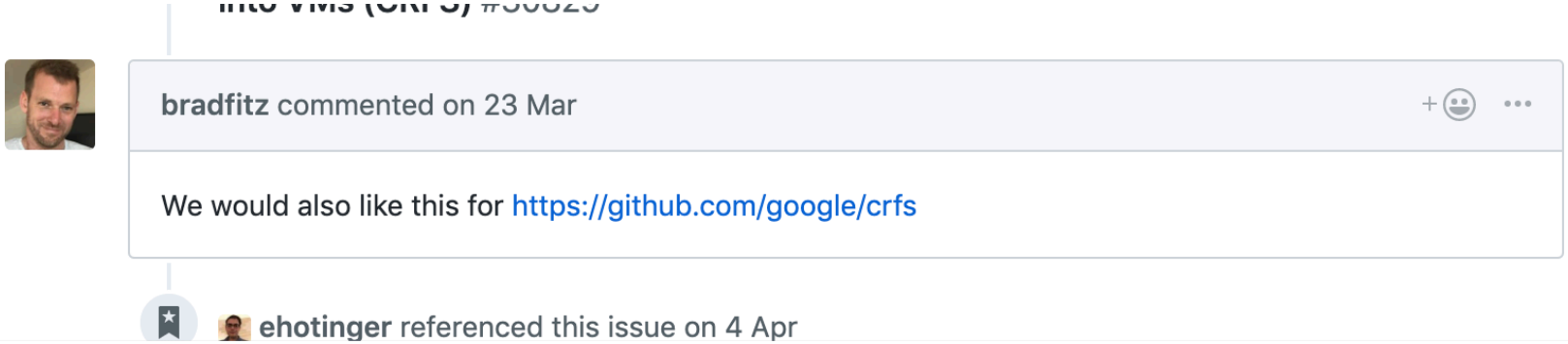
Projects

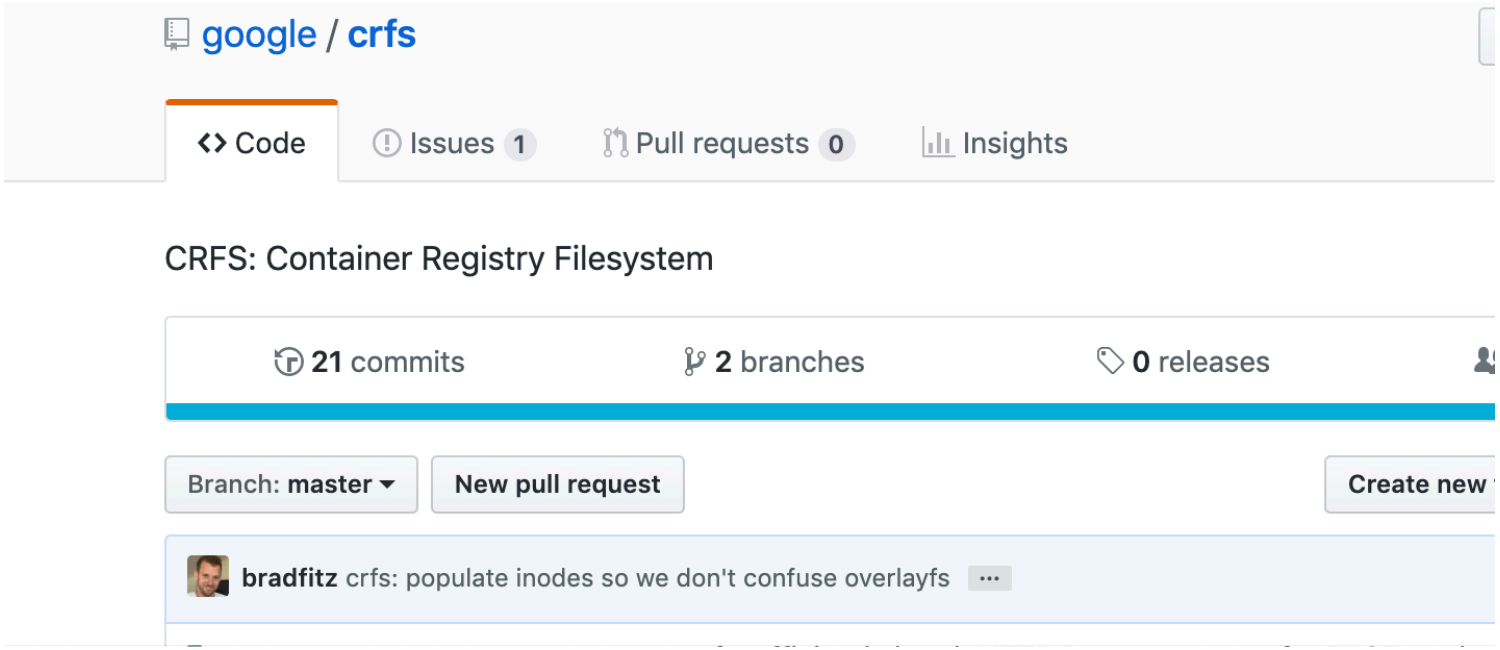
None yet

Milestone

No milestone

We're not alone





Conclusions

- Containers huge opportunity for HEP / Academia
- just a few bits missing to get broad adoption