HTCondor

Tighter HTCondor and Kubernetes interplay for better glideins

Igor Sfiligoi and Jaime Frey
(UC San Diego and UW-Madison)

PATh

UC San Diego SDSC

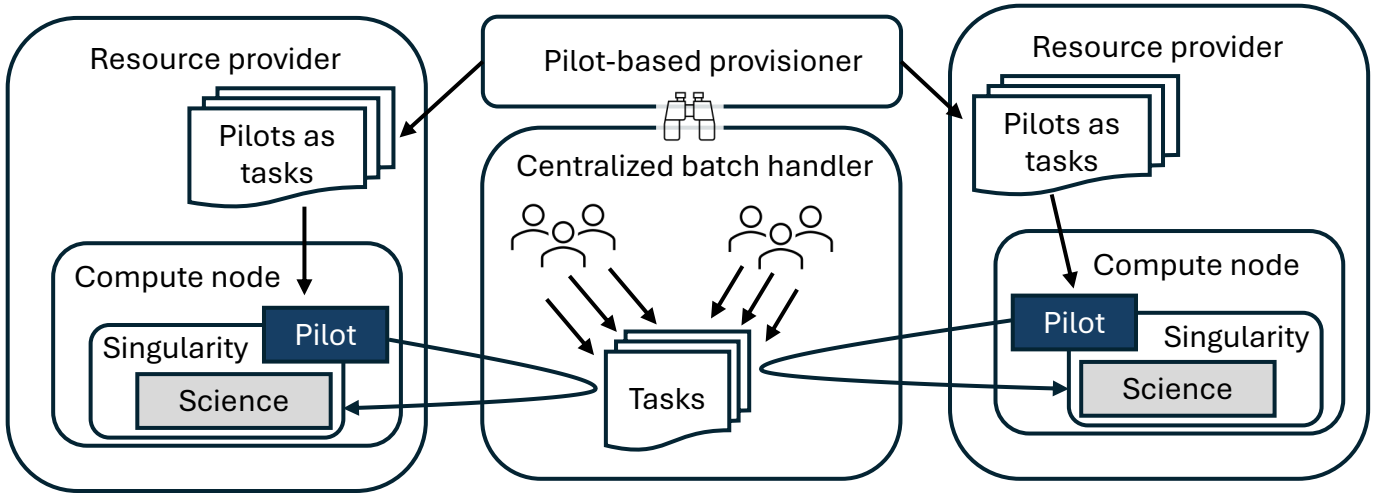
WISCONSIN UNIVERSITY OF WISCONSIN–MADISON CHTC

What are glideins and why do we care

- Glidein, a.k.a. pilot
  - Dynamically provisioned compute resource that joins a HTCondor pool
- Most of the resources in the OSPool are glideins
  - Modern dHTC would not exist without glideins/pilots

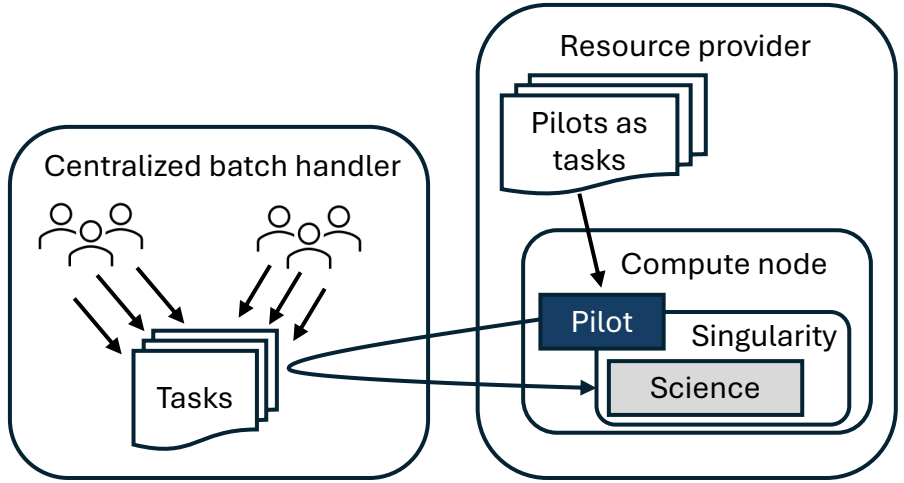
What are glideins/pilots and why do we care

- Glideins allow for creating an illusion of a common resource pool
  - By integrating resources from many independent resource providers
- Most of the time, they are provisioned just-in-time
  - i.e., when there are user tasks that can make use of them



The need for containerization in glideins

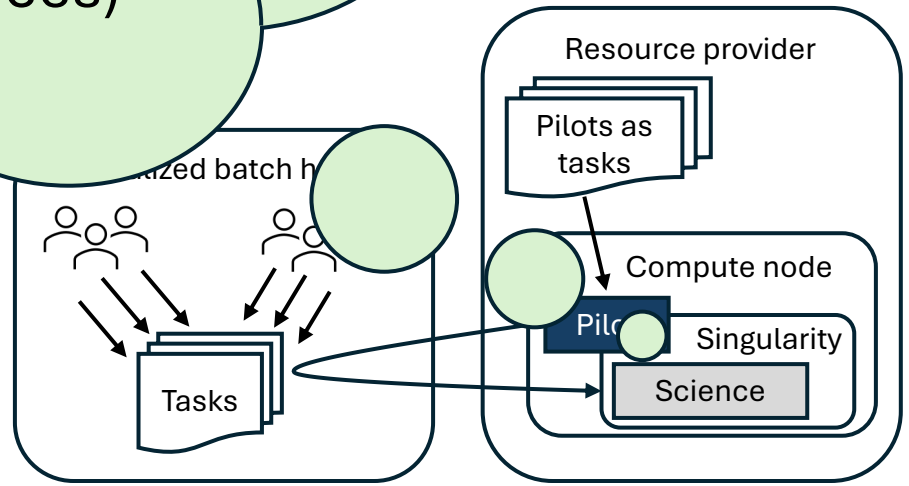
- In this talk, mostly ignore how the provisioning decisions are made
  - But want to stress that **glideins are not user-task-specific**
- A single glidein can serve many user tasks
  - Which may have different software needs
  - And may run concurrently
    e.g., using partitionable slots
- **Containerization** solves both problems
  - Task-specific container images for SW
  - Process isolation across cont. boundary, including protecting the pilot processes



The need for containerization in glideins

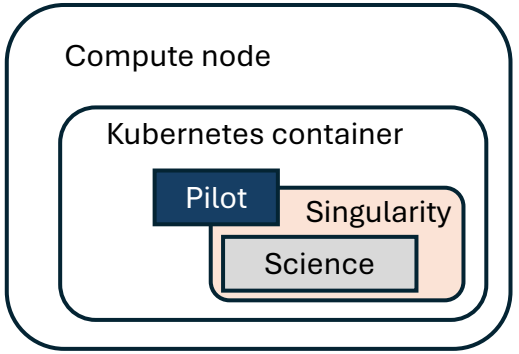
- In this talk, mo… …ns are made
  - But want to e… …cific
- A single gli…
  - Which ma…
  - And may r…
    e.g., using…
- **Containerizatio…**
  - Task-specific container images
  - Process isolation across cont. boundary, including protecting the pilot processes

Launching processes inside containers with singularity/apptainer allowed in all bare-metal setups (e.g. batch-managed resources)



Enter Kubernetes

- Kubernetes has become popular in some circles
  - If nothing else, the NSF-funded NRP platform is Kubernetes-based
  - We thus want to support glideins on Kubernetes-managed resources
- What's the problem with glideins on Kubernetes?
  - Everything in Kubernetes is container-based
  - The pilot itself starts inside a container
  - **Nested-containerization only allowed with elevated privileges**
    - Or non-standard setups

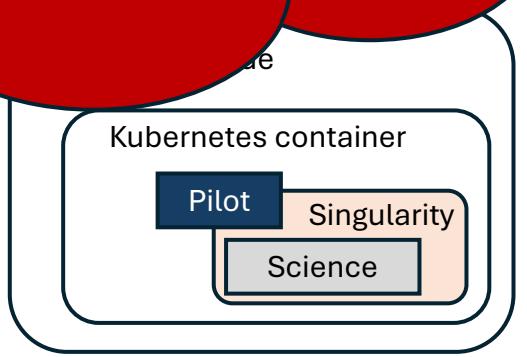


Enter Kubernetes

- Kubernetes has become popular in s
  - If nothing else, the NSF-funded NRP
  - We thus want to support glideins
- What's the problem with glideins
  - Everything in Kubernetes is containe
  - The pilot itself starts inside a contai
  - **Nested-containerization only allowed with elevated privileges**
    - Or non-standard setups

Not a sustainable solution in the long term.

Can we do better?



Use Kubernetes to our advantage

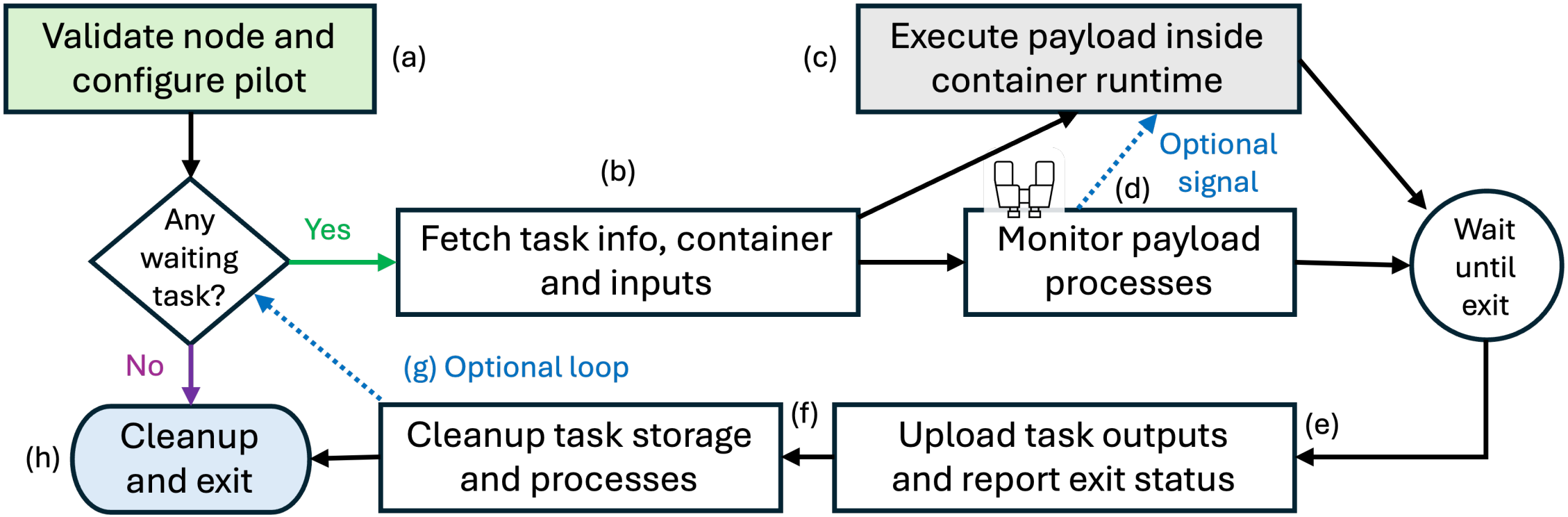
- A Kubernetes ”task” (aka ”pod”) can contain **multiple containers**
  - Unlike batch systems, **everything expects a ”set of container images”**

Use Kubernetes to our advantage

- A Kubernetes ”task” (aka ”pod”) can contain **multiple containers**
  - Unlike batch systems, **everything expects a ”set of container images”**
- Let’s use one for the pilot, and one (or more) for the user task(s)
  - Have not be done in the past
  - But seems the right way forward!
- It does need a rethink of the glidein operational logic
  - This talk is all about addressing these details

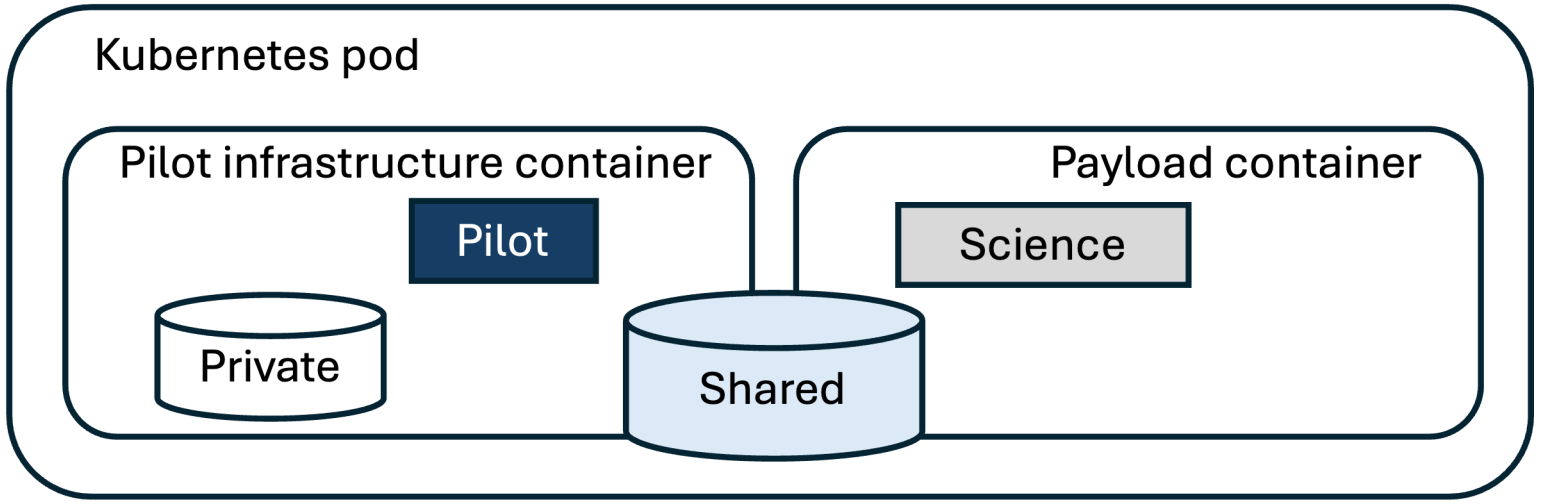
Lifetime of a glidein

- At a high level



Passing info between pilot and payload

- Just use a shared storage area
  - A fundamental Kubernetes property
- Pilot can also have a private storage area
  - Not accessible from the payload container

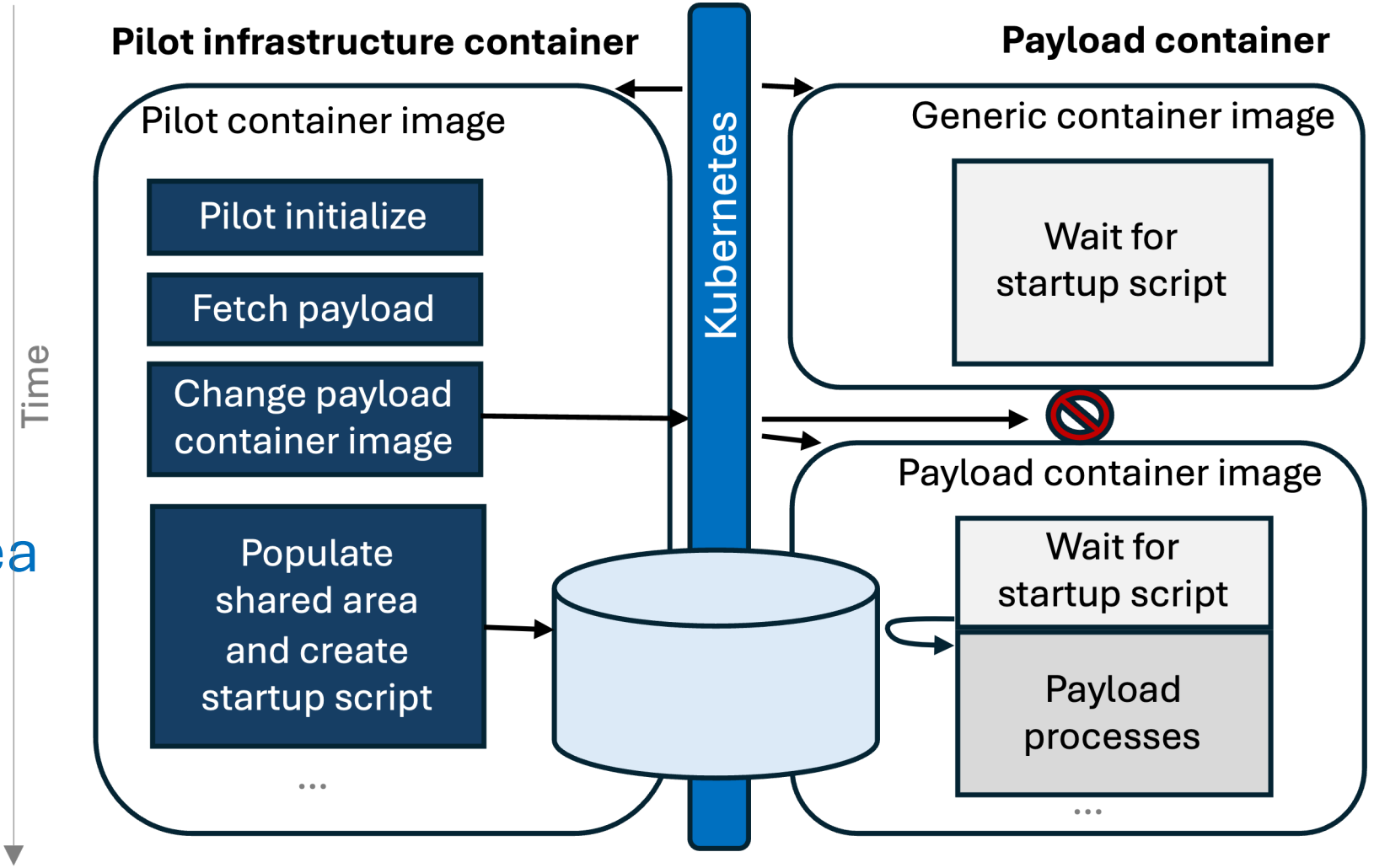


Using a user-task-specific container image

- Reminder: Pilot not task-specific
  - Task container image not known at pilot startup
- But Kubernetes needs the images of all the containers in a pod
  - **We thus start the pod with a “dummy” payload container image**
- Then **change the payload container image** once a user task is selected
  - **No special Kubernetes privileges needed**

Using a user-task-specific container image

- Start the pod with a “dummy” image
- Change the image once a user task is selected
- Use a lightweight script on shared area for starting the needed processes

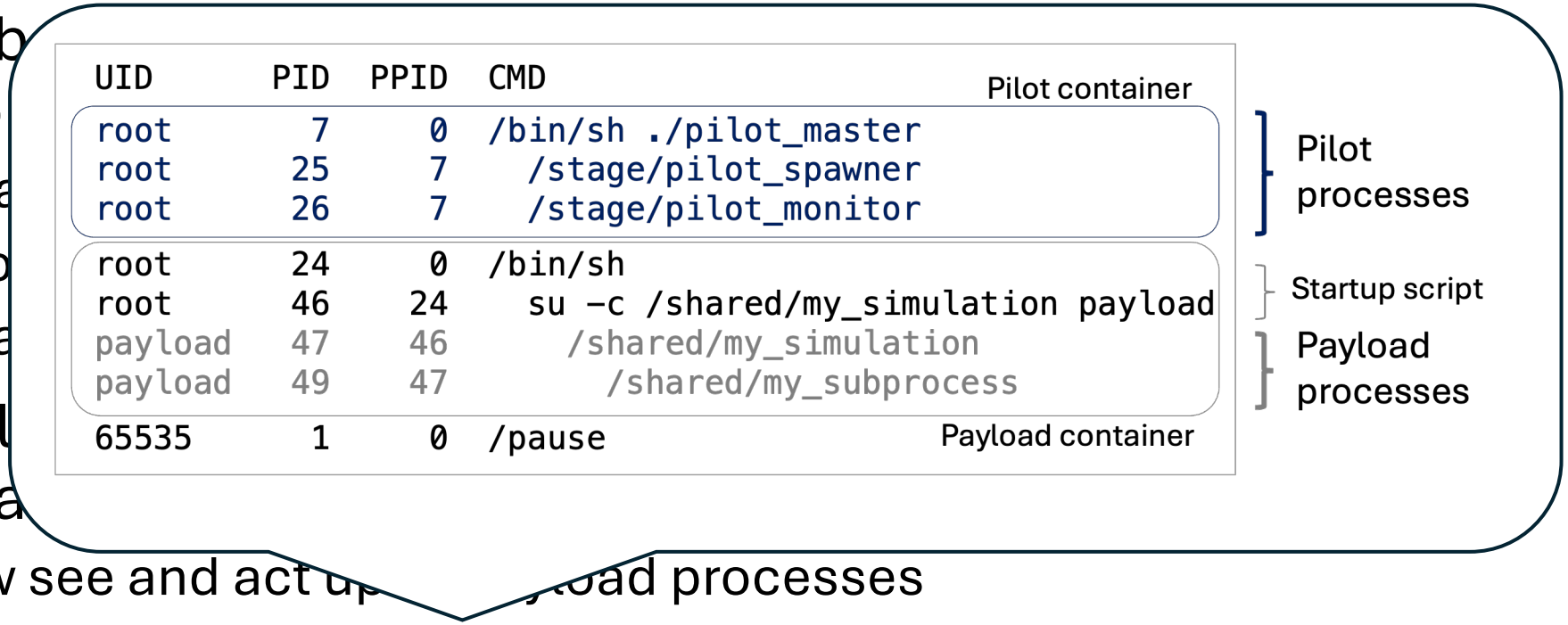


Limit container isolation

- By default, Kubernetes completely isolates processes in separate containers
  - Even in the same pod
  - Obviously, good for security
  - But problematic in a pseudo-parent-child relationship
- Kubernetes allows **for sharing of PID space** between containers in the same pod
  - Pilot can now see and act upon payload processes
  - Paired with **pseudo-root in pilot container**, can prevent payload to act upon pilot processes
    - **Assuming payload cannot gain pseudo-root privileges**

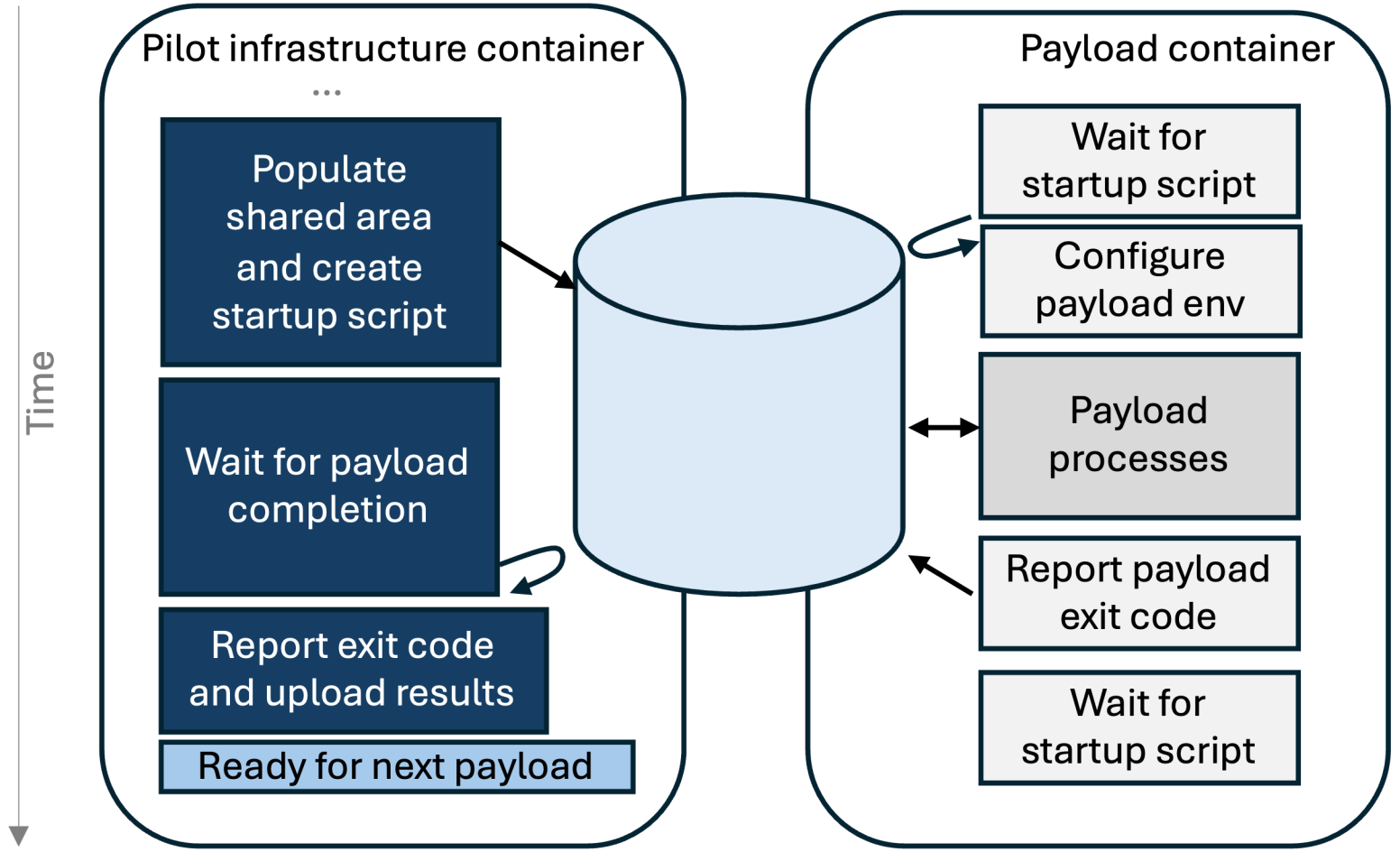
Limit container isolation - securely

- By default, Kub
  in separate co
  - Even in the sa
  - Obviously, go
  - But problema
- Kubernetes all
  between conta
  - Pilot can now see and act up oad processes
  - Paired with **pseudo-root in pilot container**, can prevent payload to act upon pilot processes
    - **Assuming payload cannot gain pseudo-root privileges**



A few more details on the glue-script

- The script that glues the pilot and payload container essential
- Does both the initial setup and final cleanup duties
- Also propagate the exit code
- Owned by pseudo-root for security



That’s it for the high-level overview

- We exercised the Kubernetes mechanisms with a PoC
  - Many thanks to UCSD student Grace (Yunjin) Zhu
- But needs changes to HTCondor to be put in production
  - As a reminder, a glidein pilot uses HTCondor for most of the serious work

Bonus material

(Not part of the original talk, dreamed it after seeing talks on Wed morning)

Can the same principles help outside K8S?

- E.g. in bare-metal docker installations?
  - Without granting any privileges to glideins

Can the same principles help outside K8S?

- E.g. in bare-metal docker installations?
- Many of the concepts are indeed not Kubernetes-specific
- But would need something at bare-metal layer to mimic that functionality
  - At the very least, to change the task container image

Can the same principles help outside K8S?

- E.g. in bare-metal docker installations?
- Many of the concepts are indeed not Kubernetes-specific
- But would need something at bare-metal layer
  to mimic that functionality
- Can only think of one functionality that may be hard to provide
  - “Limit container isolation”
  - Aka exposing PID namespaces across containers
  - Not essential, more a nice-to-have
  - One could inject something in user container to mimic that,
    but it is more work for HTCondor/glidein developers

Bonus material 2

(Not part of the original talk, dreamed it after seeing talks in late Wed morning)

How about pilot in pilot support?

- My proposal does not solve the nested container limitations
  - It just avoids them
- What is the task launched by a glidein is a pilot itself?
  - We saw plenty of examples Wed!

Credit: https://www.letstakeacloserlook.com/2020/04/27/turtles-all-the-way-down/

How about pilot in pilot support?

- My proposal does not solve the nested container limitations
  - It just avoids them
- What is the task launched by a glidein is a pilot itself?
  - Good news: **Most HTCondor jobs are not pilots**!
  - Bad news: A non-negligible fraction are.

How about pilot in pilot support?

- My proposal does not solve the nested container limitations
  - It just avoids them
- What is the task launched by a glidein is a pilot itself?
- One possible solution:
  - Do **not** run “tasks that are pilots” inside containers
  - HTCondor already has knobs to do this (during condor_submit)
  - No security risks, thanks to pseudo-root in containers
- Does need a mechanism to allow the “task that is a pilot” to launch its tasks in the “user payload” container

How about pilot in pilot support?

- My proposal does not a
  - It just avoids them
- What is the task laun
- One possible solutio
  - Do **not** run “tasks that are pilots” inside containers
  - HTCondor already has knobs to do this (during condor_submit)
  - No security risks, thanks to pseudo-root in containers
- Does need a mechanism to allow the “task that is a pilot” to launch its tasks in the “user payload” container

Such ”pilot tasks” are *usually* managed by professionals

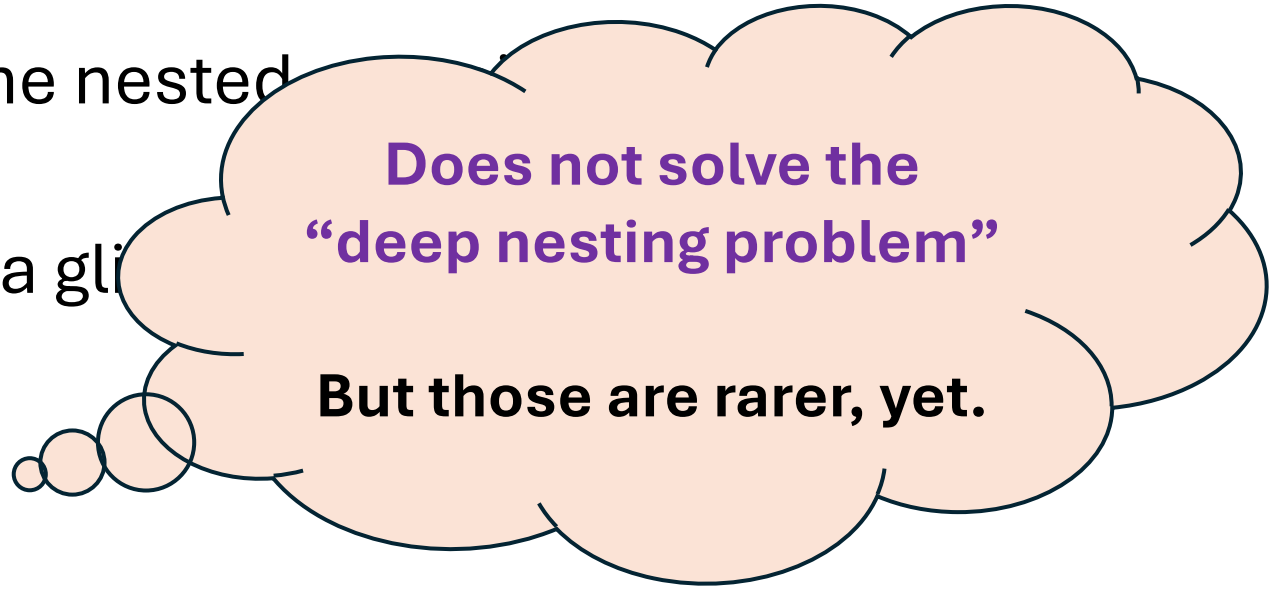
- **Coordinating with the glidein infrastructure ops regarding OS and installed libraries may be feasible**

How about pilot in pilot support?

- My proposal does not solve the nested container limitations
  - It just avoids them
- What is the task launched by a glidein is a pilot itself?
- Another possible solution:
  - **Use a 3-container k8s pod!**
- The 3rd container
  - Is allocated only minimal resources
  - Only used when a HTCondor job labels itself as a “pilot”
  - Is allowed to make changes to the “user payload” container

How about pilot in pilot support?

- My proposal does not solve the nested
  - It just avoids them
- What is the task launched by a gli
- Another possible solution:
  - **Use a 3-container k8s pod!**
- The 3rd container
  - Is allocated only minimal resources
  - Only used when a HTCondor job labels itself as a “pilot”
  - Is allowed to make changes to the “user payload” container



How about pilot in pilot support?

- My proposal does not solve the nested
  - It just avoids them
- What is the task launched by a gli
- Another possible solution
  - **Use a 3-co**
- The 3rd
  - Is alloc
  - Only u
  - Is allowed to ma container

Does not solve the "deep nesting problem"

rer, yet.

There are probably more edge cases I have not though about.

(Admittedly, very preliminary brainstorming)

Questions?

Acknowledgements

- This work was funded by the U.S. National Science Foundation (NSF) under grant OAC-2030508.
- Computing resources used were partially funded by NSF under grant OAC-2112167.