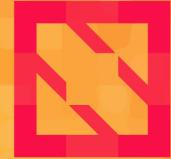




KubeCon



CloudNativeCon

North America 2019

Serving HTC Users in K8s by Leveraging HTCondor

Igor Sfiligoi, University of California San Diego (UCSD/SDSC)

UC San Diego



Who am I?

Name: Igor Sfiligoi

Employer: UC San Diego



Longtime HTC user

- Most recently as part of the Open Science Grid (OSG)

For the past year actively involved with Kubernetes

- As part of the Pacific Research Platform (PRP)

UC San Diego

SDSC
SAN DIEGO SUPERCOMPUTER CENTER



Open Science Grid

<https://opensciencegrid.org>

PRP
PACIFIC RESEARCH
PLATFORM

<http://pacificresearchplatform.org>

Let's define HTC

HTC = High Throughput Computing

Often also called Batch Computing
(although not all Batch Computing is HTC)

Let's define HTC



KubeCon



CloudNativeCon

North America 2019

HTC = High Throughput Computing

Often also called Batch Computing
(although not all Batch Computing is HTC)

The infrastructure for Ingenuously Parallel Computing

Ingenious Parallelism

- Restate a **big computing problem** as many **individually schedulable small problems**.
- Minimize your requirements in order to maximize the raw capacity that you can effectively use.

Ingenious Parallelism



KubeCon



CloudNativeCon

North America 2019



A large, light-yellow thought bubble with a blue outline and three smaller circles above it. Inside the bubble, the text reads:

Some call it
Embarrassingly Parallel Computing
but it really takes hard thinking!

- Restate a **big computing problem** as many **individually schedulable small problems**.
- Minimize your requirements in order to maximize the raw capacity that you can effectively use.

Example HTC problems



Monte Carlo Simulations
Parameter sweeps
Event processing
Feature extraction

And many more problems can be cast in this paradigm.

Example HTC resource



KubeCon

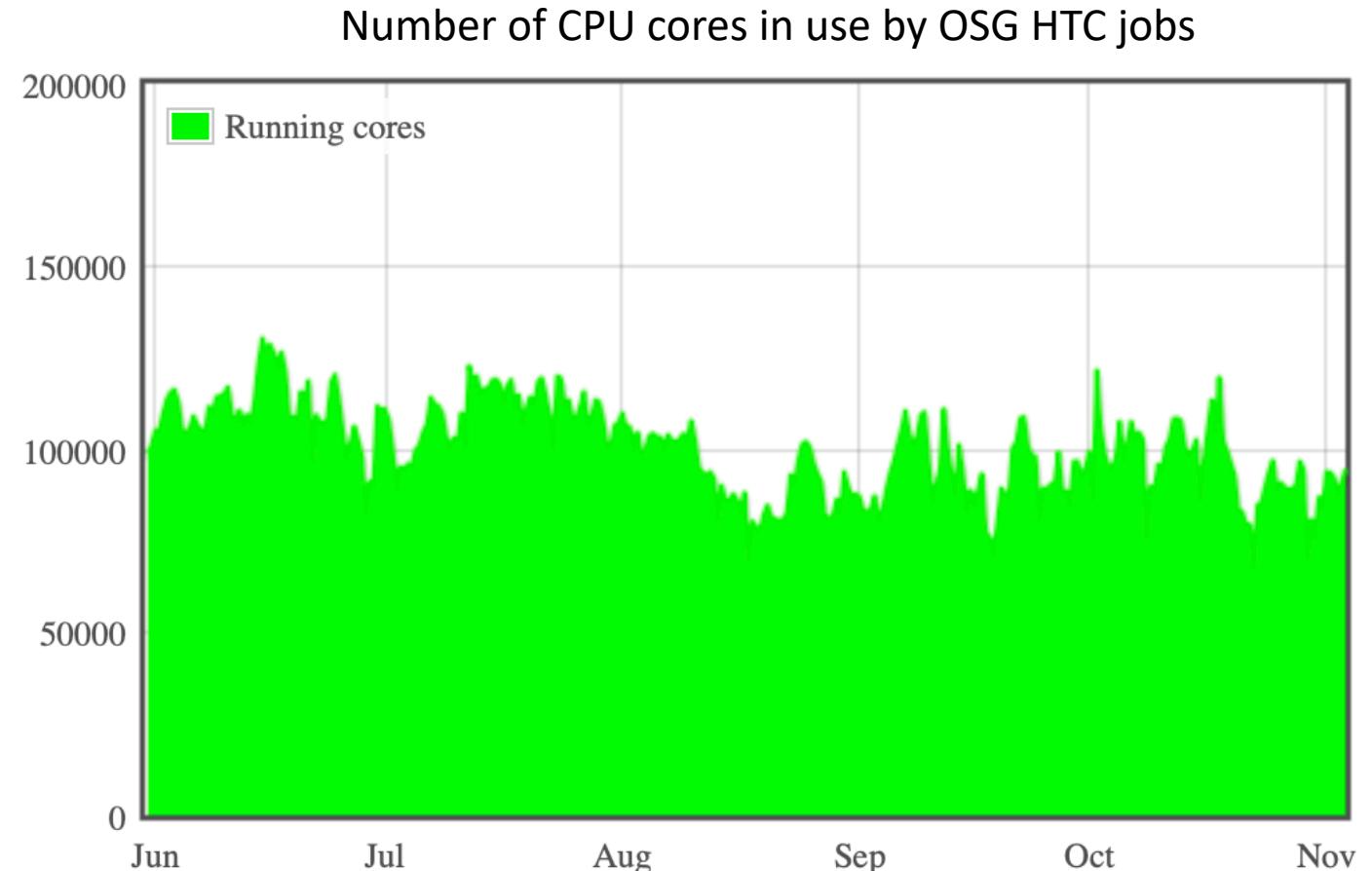
CloudNativeCon

North America 2019

Open Science Grid (OSG)
operates a large scale HTC pool



Open Science Grid



Example HTC users



KubeCon

CloudNativeCon

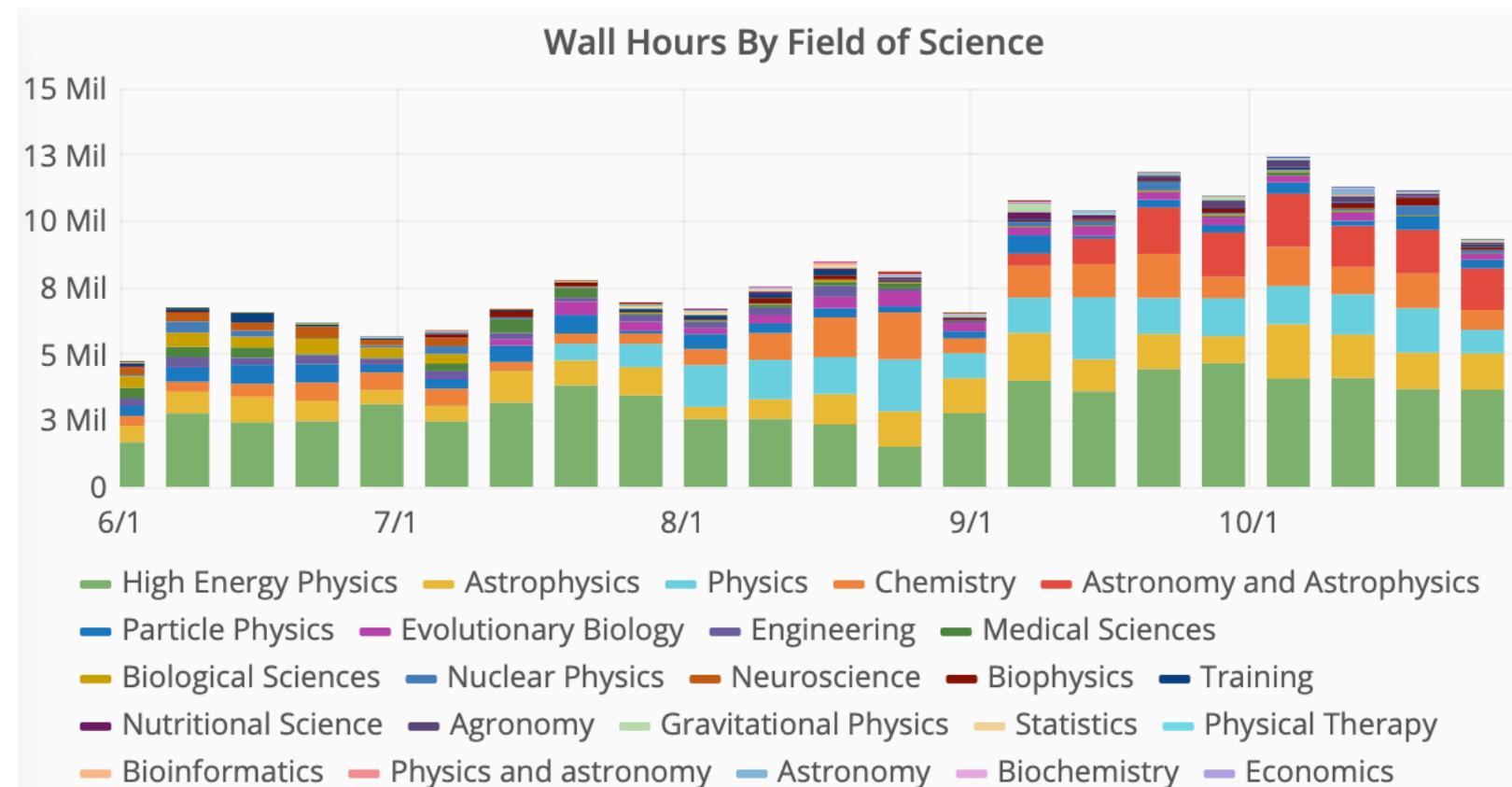
North America 2019

OSG serving
many different
scientific domains



Open Science Grid

Weekly CPU hours used by OSG HTC jobs





KubeCon



CloudNativeCon

North America 2019

Can we use Kubernetes for HTC?

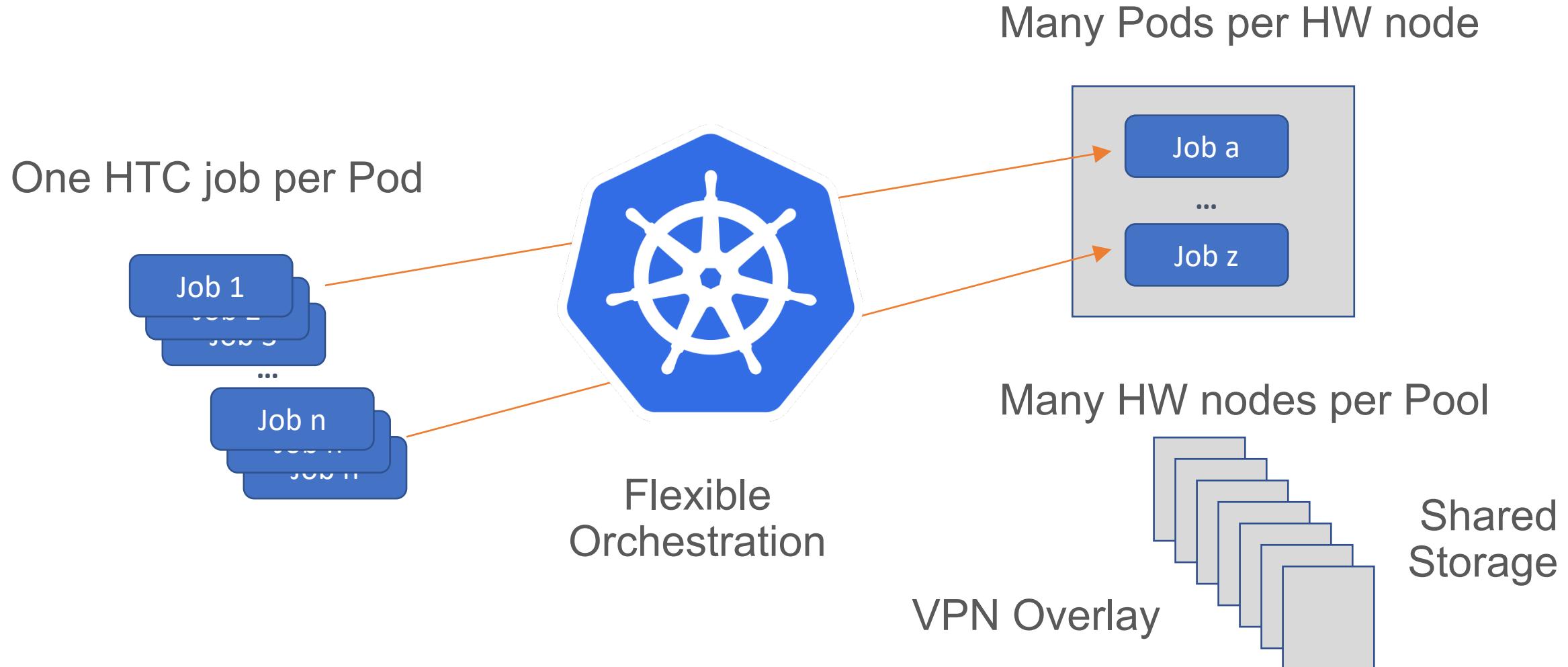
K8s in principle great for HTC



KubeCon

CloudNativeCon

North America 2019



K8s in practice not so great



KubeCon

CloudNativeCon

North America 2019

K8s missing a few features HTC users are used to

- Indexed parameter passing
- Automatic Input/Output handling
Note: HTC jobs typically do not require a shared FS
- Fair Share Scheduling policies
Essential for highly contested resources
- Can it scale to millions of queued Pods?

K8s in practice not so great



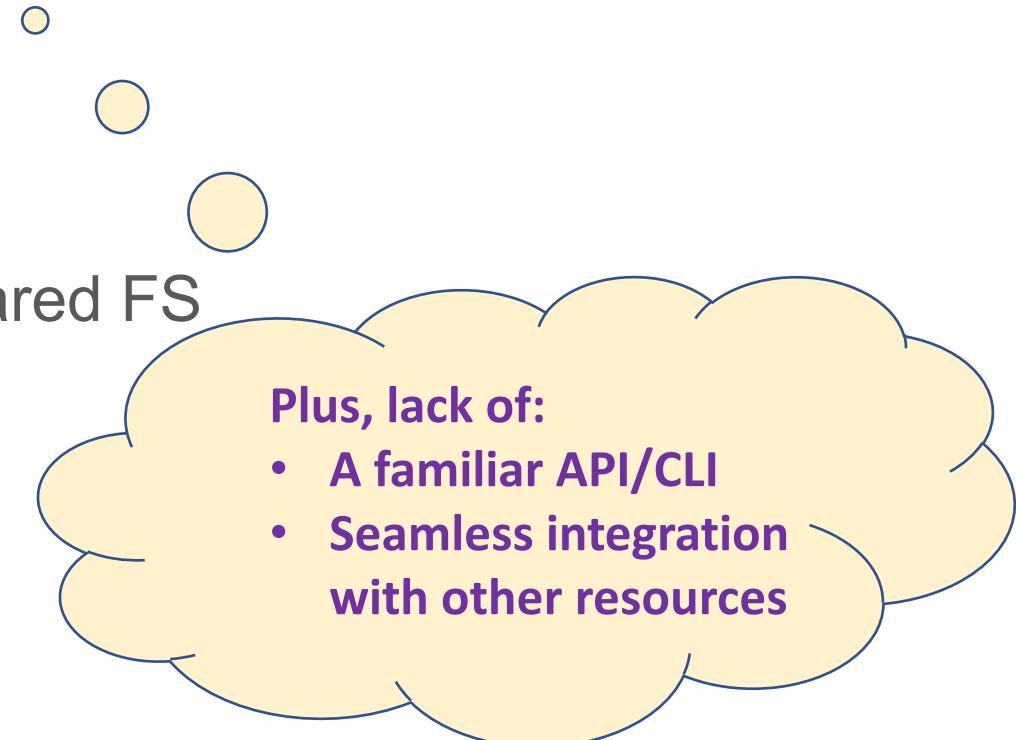
KubeCon

CloudNativeCon

North America 2019

K8s missing a few features HTC users are used to

- Indexed parameter passing
- Automatic Input/Output handling
Note: HTC jobs typically do not require a shared FS
- Fair Share Scheduling policies
Essential for highly contested resources
- Can it scale to millions of queued Pods?



Plus, lack of:

- A familiar API/CLI
- Seamless integration with other resources

How about leveraging
HTCondor with K8s?

Using HTCondor with Kubernetes



KubeCon



CloudNativeCon

North America 2019

Why HTCondor?

- One of the major batch systems
- HTC-focused architecture
- Very flexible, often used in heterogeneous environments
- Native support for containers
- Highly scalable



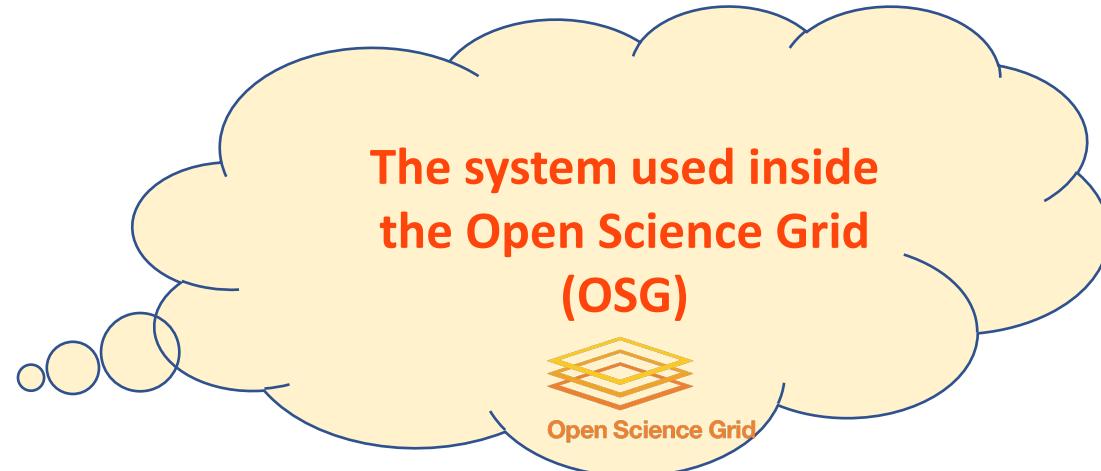
<https://research.cs.wisc.edu/htcondor/>

Using HTCondor with Kubernetes



Why HTCondor?

- One of the major batch systems
- HTC-focused architecture
- Very flexible, often used in heterogeneous environments
- Native support for containers
- Highly scalable



<https://research.cs.wisc.edu/htcondor/>

Using HTCondor with Kubernetes



KubeCon



CloudNativeCon

North America 2019

NVIDIA
Nov 17th, 2019

THE LARGEST CLOUD SIMULATION IN HISTORY

ICECUBE OBSERVATORY DETECTING NEUTRINOS

Events Processed Per GPU Type

GPU Type	Events Processed
K80	Red
M40	Blue
T4	Purple
P100	Grey
V100	Teal

MULTIPLE GENERATIONS, ONE APPLICATION

50K NVIDIA GPUs IN THE CLOUD
350 PF OF SIMULATION FOR 2 HOURS
PRODUCED 5% OF ANNUAL SIMULATION DATA
AWS, MICROSOFT AZURE, GOOGLE CLOUD PLATFORM
DISTRIBUTED ACROSS U.S., EUROPE, APAC

Wuerthwein, Ph.D.
Associate Director, Open Science Grid

Igor Shmelev
Lead Developer and Researcher

UCSD
Open Science Grid

HTCondor
High Throughput Computing

Inside Grid

A man in a black jacket is standing on a stage, pointing towards the large screen behind him. The screen displays a slide titled "THE LARGEST CLOUD SIMULATION IN HISTORY". The slide features a pie chart showing the distribution of events processed by different GPU types: K80 (red), M40 (blue), T4 (purple), P100 (grey), and V100 (teal). Below the chart, it says "MULTIPLE GENERATIONS, ONE APPLICATION". To the left of the chart, there's an image of the ICECUBE observatory detecting neutrinos. On the right side of the slide, there's text about the simulation: "50K NVIDIA GPUs IN THE CLOUD", "350 PF OF SIMULATION FOR 2 HOURS", "PRODUCED 5% OF ANNUAL SIMULATION DATA", "AWS, MICROSOFT AZURE, GOOGLE CLOUD PLATFORM", and "DISTRIBUTED ACROSS U.S., EUROPE, APAC". At the bottom of the slide, there are logos for UCSD and Open Science Grid, along with names of two speakers: Wuerthwein, Ph.D. and Igor Shmelev. A yellow cloud-like graphic with the text "Inside Grid" is overlaid on the right side of the slide.

HTCondor Architecture

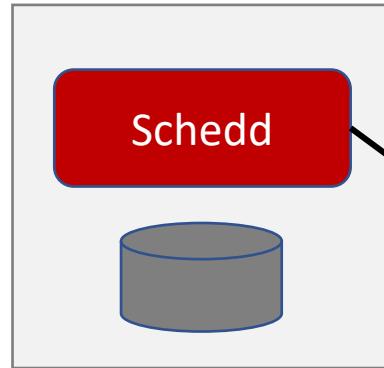


KubeCon

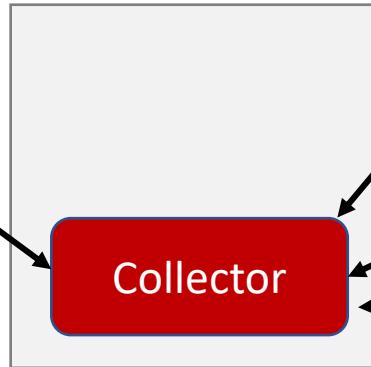
CloudNativeCon

North America 2019

Persistent Job Queue
(can be more than one, but all independent)

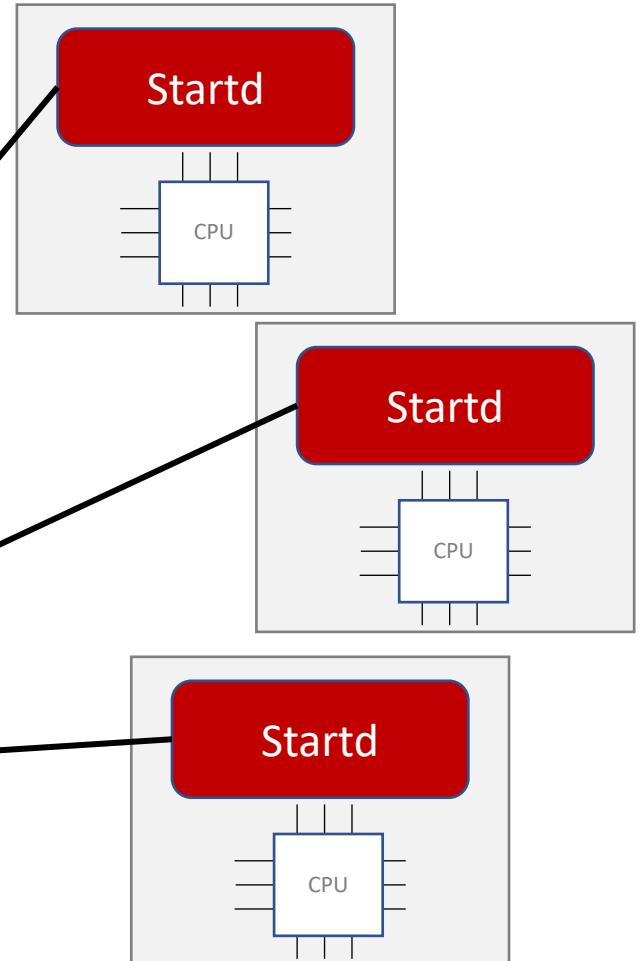


Submission typically local
(e.g. ssh)



Central manager for bookkeeping
(can have multiple for HA)

Each execute resource
has a control process



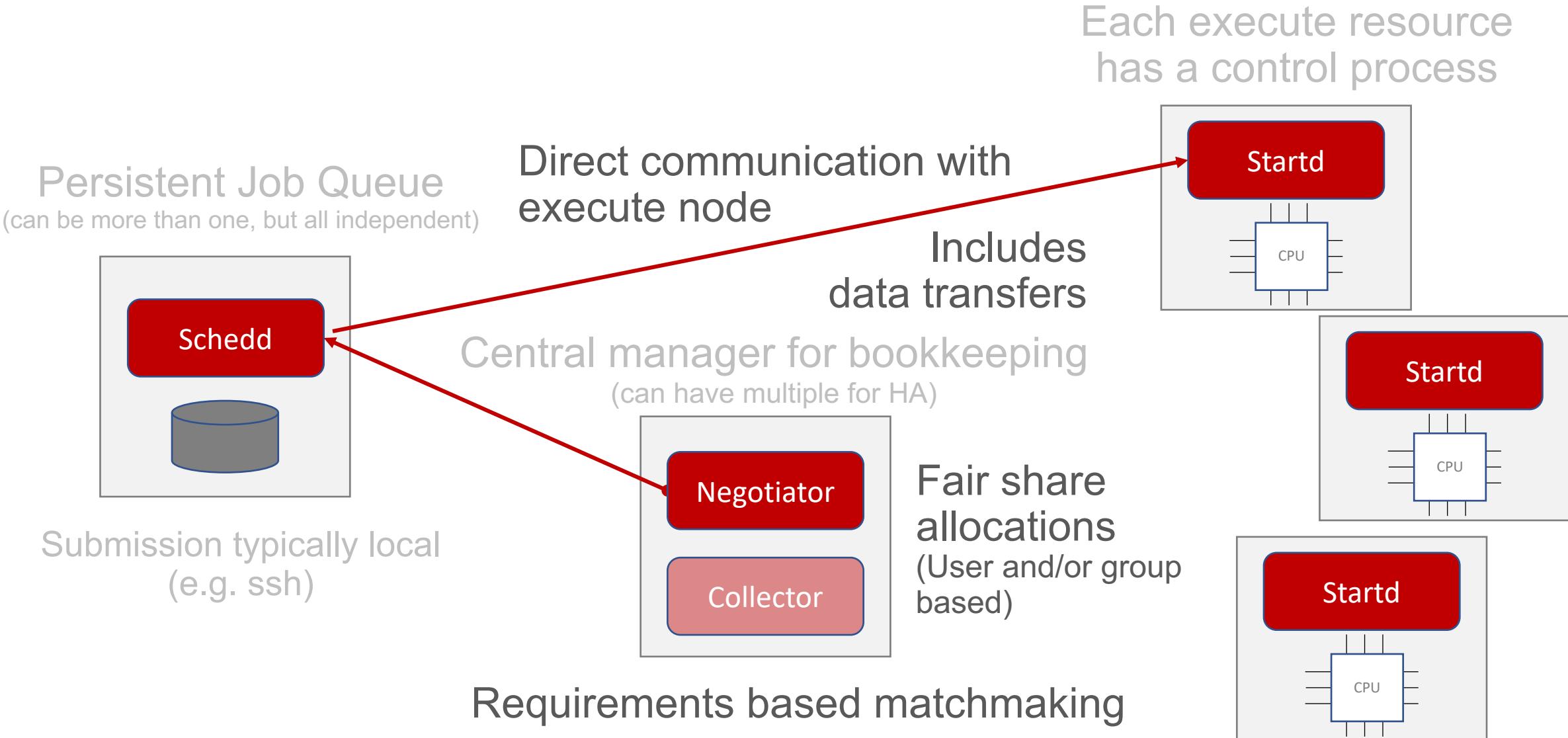
HTCondor Architecture



KubeCon

CloudNativeCon

North America 2019



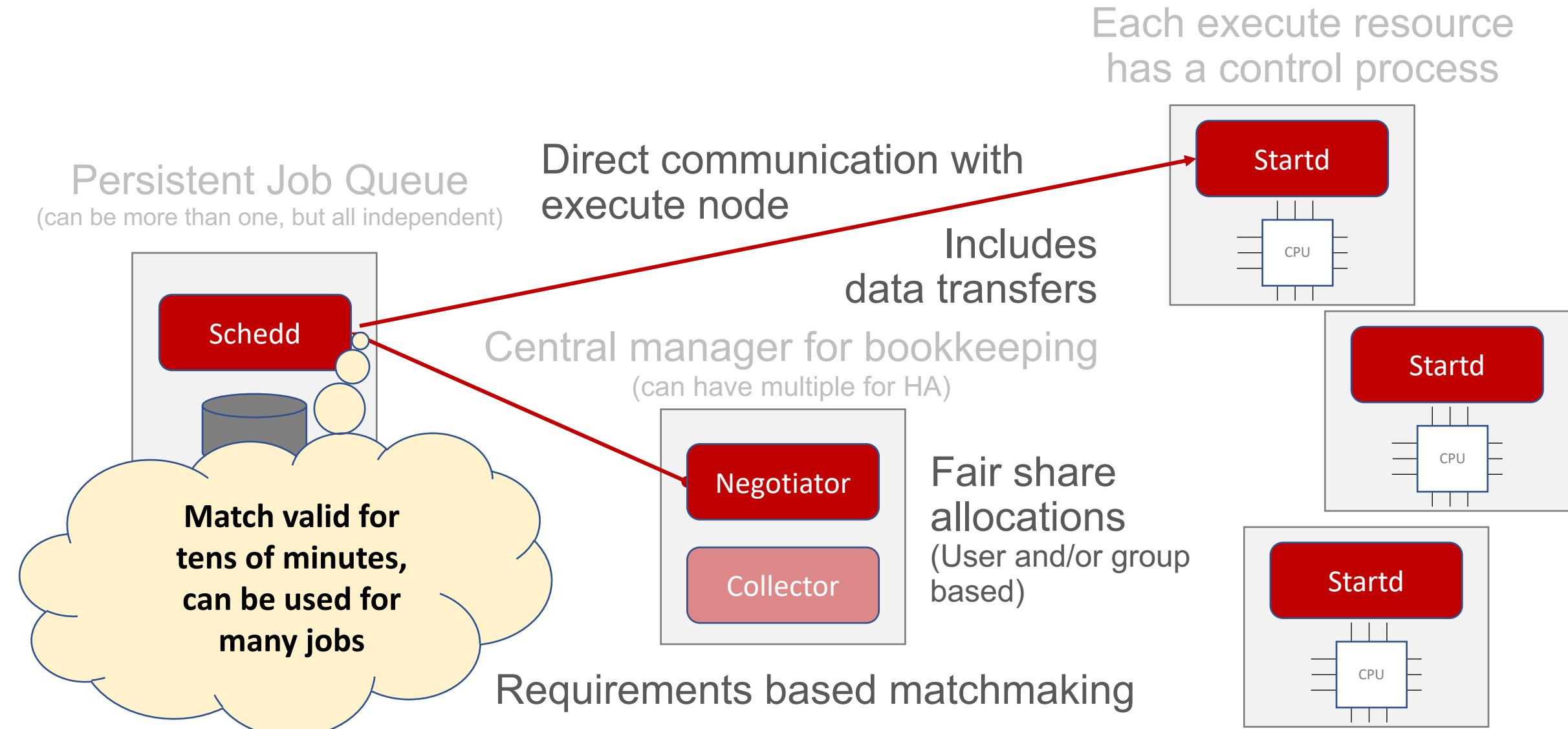
HTCondor Architecture



KubeCon

CloudNativeCon

North America 2019



How do we leverage
HTCondor with K8s?

Using HTCondor with Kubernetes



KubeCon

CloudNativeCon

North America 2019

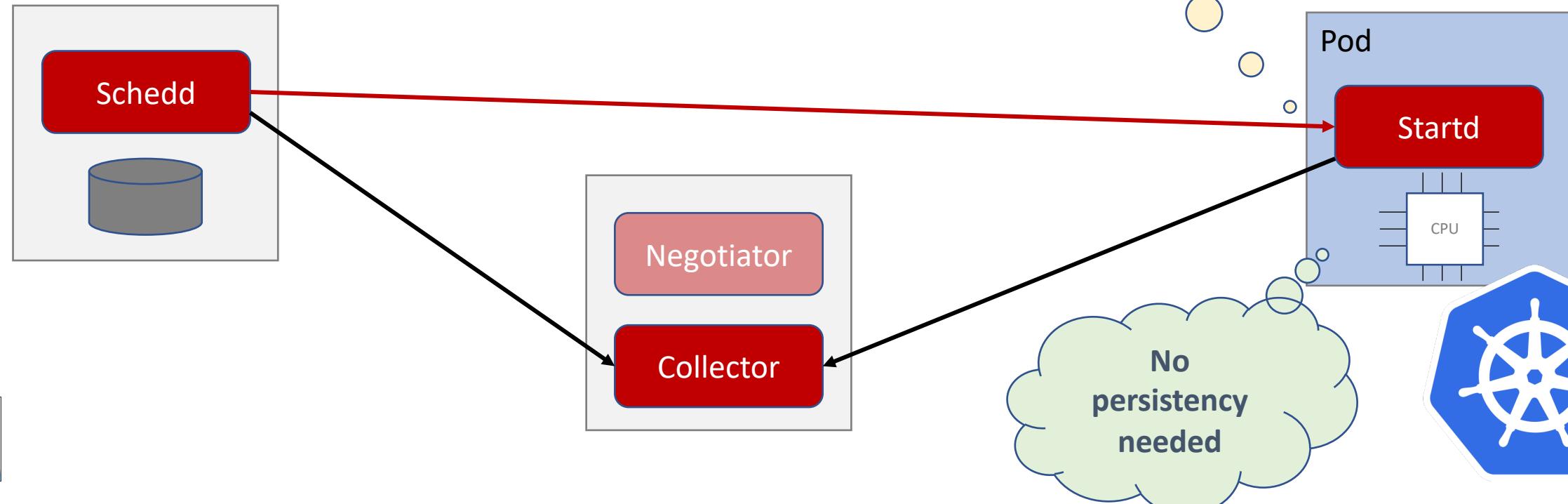
The Kubernetes resources can be joined to an existing HTCondor Pool



PACIFIC RESEARCH
PLATFORM



No
persistency
needed



Using HTCondor with Kubernetes

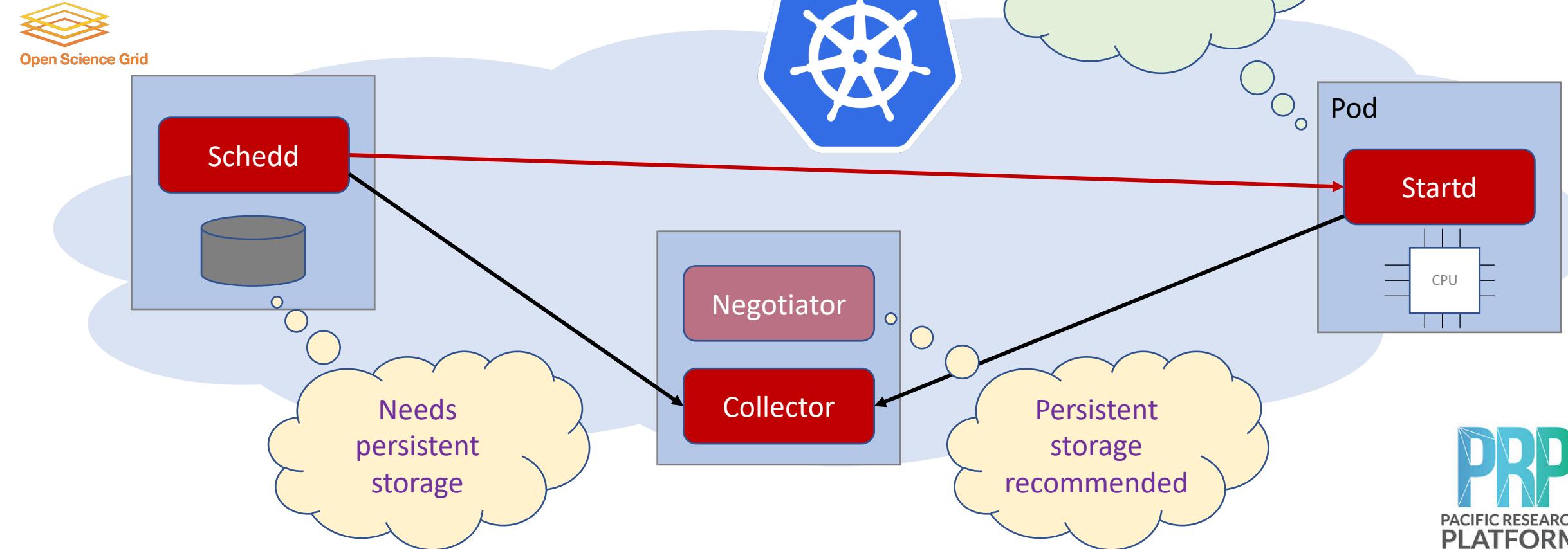


KubeCon

CloudNativeCon

North America 2019

Or a complete HTCondor Pool can be created inside Kubernetes



Using HTCondor with Kubernetes

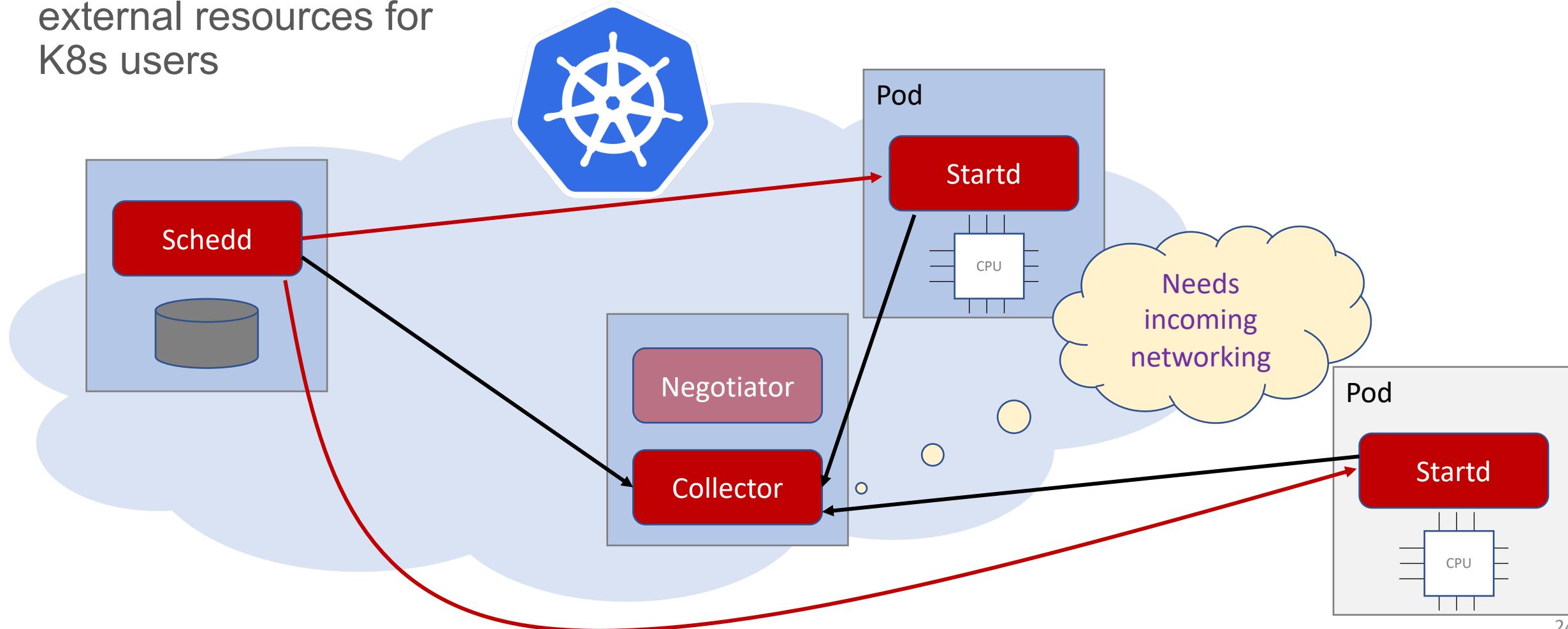


KubeCon

CloudNativeCon

North America 2019

Can be used to join
external resources for
K8s users



HTC Users and Containers



KubeCon



CloudNativeCon

North America 2019

Most HTC jobs are application + arguments + data

- Container just a convenient way to package the dependencies
- Usually a department/community maintained one

HTCondor and Containers

Most HTC jobs are application + arguments + data

- Container just a convenient way to package the dependencies
- Usually a department/community maintained one

HTCondor allows for a container to be attached to a job

- Will use singularity to invoke it
- After binding the application and data



HTCondor and Containers



KubeCon

CloudNativeCon

North America 2019

Most HTC jobs are application + arguments + data

- Container just a convenient way to package the dependencies
- Usually a department/community maintained one

HTCondor allows for a container to be attached to a job

- Will use singularity to invoke it
- After binding the application and data



In principle Docker could be an option, but not currently supported

Nested containerization



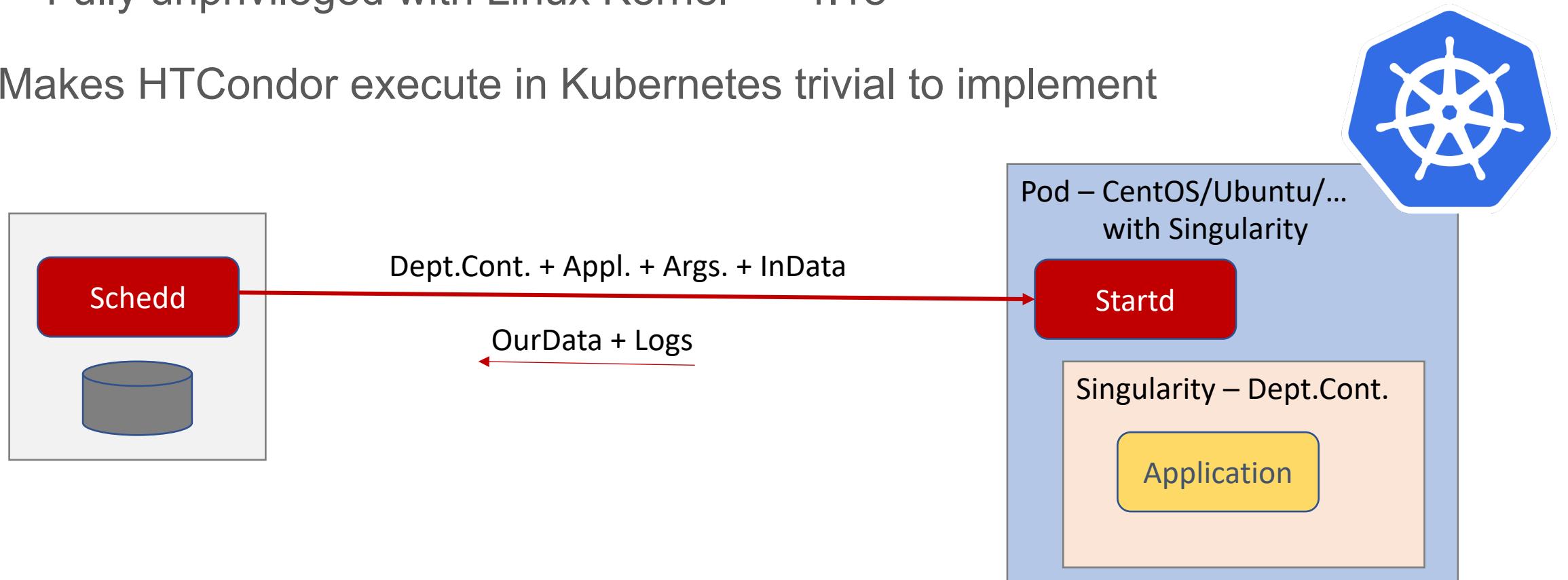
KubeCon

CloudNativeCon

North America 2019

- Singularity can be invoked inside a Docker container
- Fully unprivileged with Linux Kernel ≥ 4.18

Makes HTCondor execute in Kubernetes trivial to implement



Explicit provisioning



KubeCon

CloudNativeCon

North America 2019

Many systems still on older Linux Kernel Versions (e.g. CentOS 7)

- Unprivileged nested containerization not an option there

Some users also do not like singularity

- It does have some differences from Docker
- e.g. The root partition is always Read-Only

Kubernetes Pod can be launched with Container needed by User jobs

- Only jobs needing that Container will match
- Asking users to create a HTCondor-specific Container usually a non-starter
- Better to inject HTCondor bins and config at Pod startup

A ready-to-use template available at:

<https://github.com/sfiligoi/prp-htcondor-pool>



Opportunistic use

Most HTC jobs tolerate preemption

- HTC Pods great backfill option for keeping your Kubernetes resources fully utilized



Just launch HTCondor execute Pods with a very low K8s priority

Works best when you have a single backfill pool



To conclude

Kubernetes is a great foundation platform for HTC jobs

- But a bit hard to use by itself

HTCondor can add the needed glue to make it easy to use

- Data handling
- Parametrized argument passing
- Robust, contention-optimized and scalable policy manager

OSG and PRP have been successfully using this combination for awhile

Acknowledgments

This work was partially funded by
US National Science Foundation (NSF) awards
CNS-1456638, CNS-1730158,
ACI-1540112, ACI-1541349,
MPS-1148698,
OAC-1826967, OAC 1450871,
OAC-1659169 and OAC-1841530.

