PACIFIC RESEARCH PLATFORM

Open Science Grid

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

UC San Diego

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

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

Open Science Grid

https://opensciencegrid.org

For the past year actively involved with Kubernetes
- As part of the Pacific Research Platform (PRP)

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

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

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

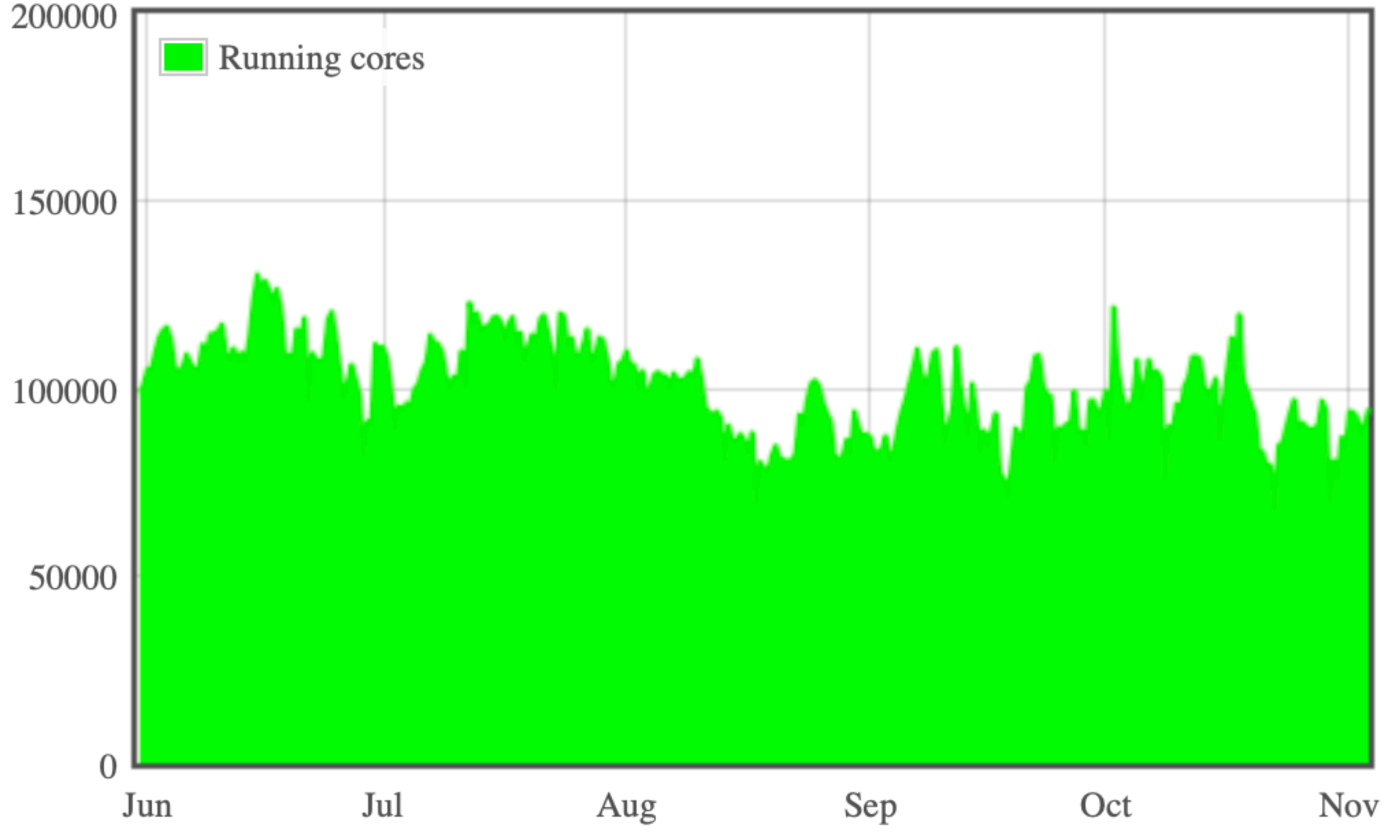
And many more problems can be cast in this paradigm.

Example HTC resource

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

Open Science Grid

Number of CPU cores in use by OSG HTC jobs

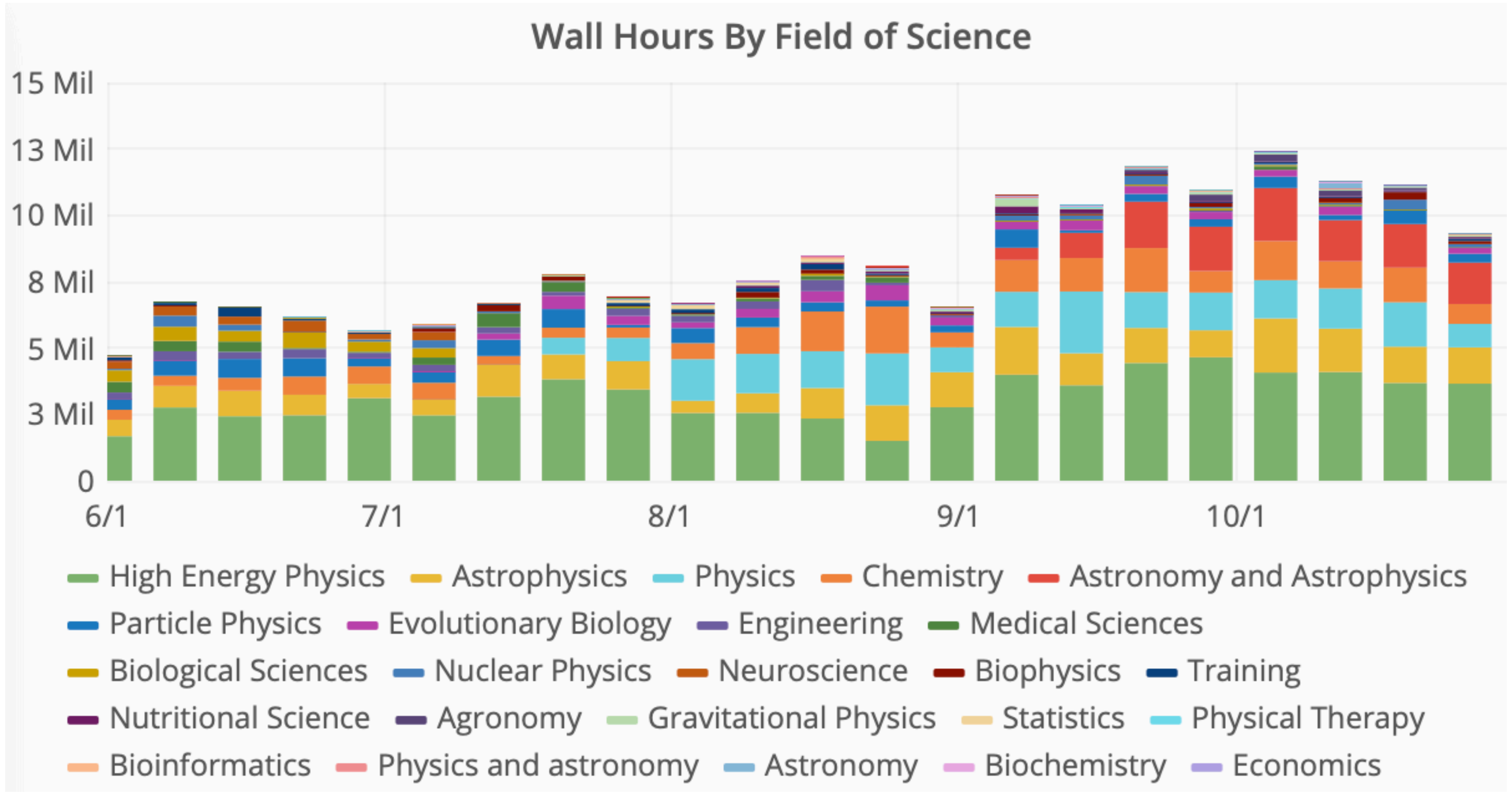


Example HTC users

OSG serving many different scientific domains

Open Science Grid

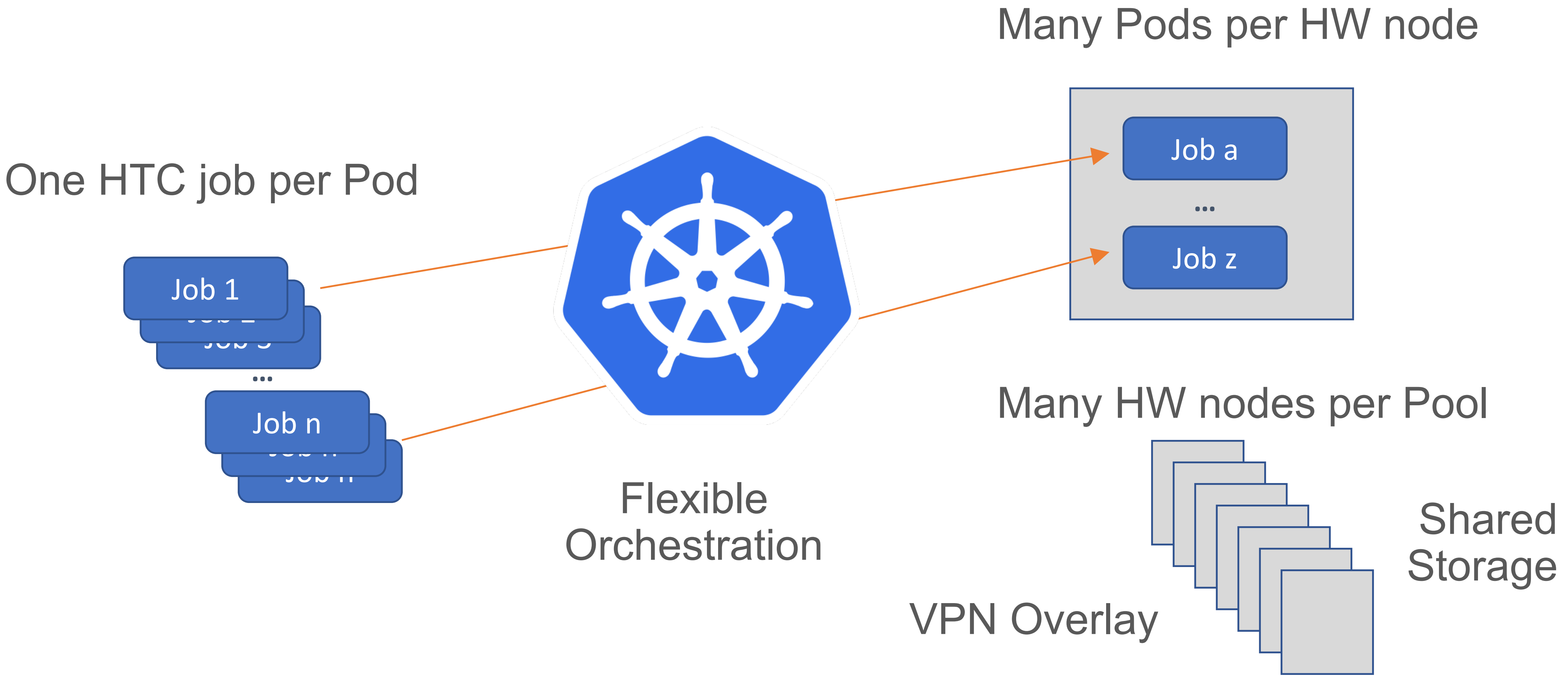
Weekly CPU hours used by OSG HTC jobs



HTC and Kubernetes

Can we use Kubernetes for HTC?

K8s in principle great for HTC



K8s in practice not so great

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

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**

HTC and Kubernetes

How about leveraging HTCondor with K8s?

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

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

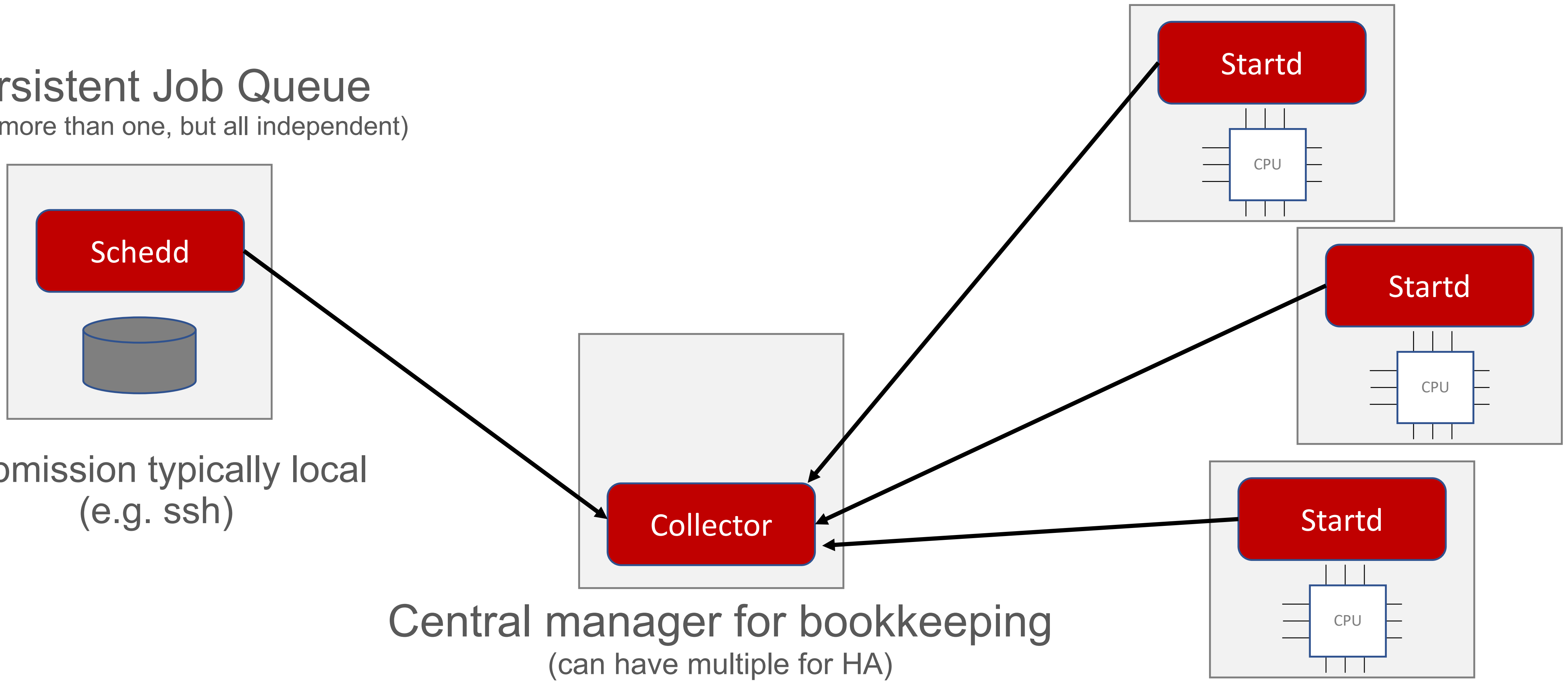
The system used inside the Open Science Grid (OSG)

HTCondor
High Throughput Computing

HTCondor Architecture

Each execute resource has a control process

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)

HTCondor Architecture

Each execute resource has a control process

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

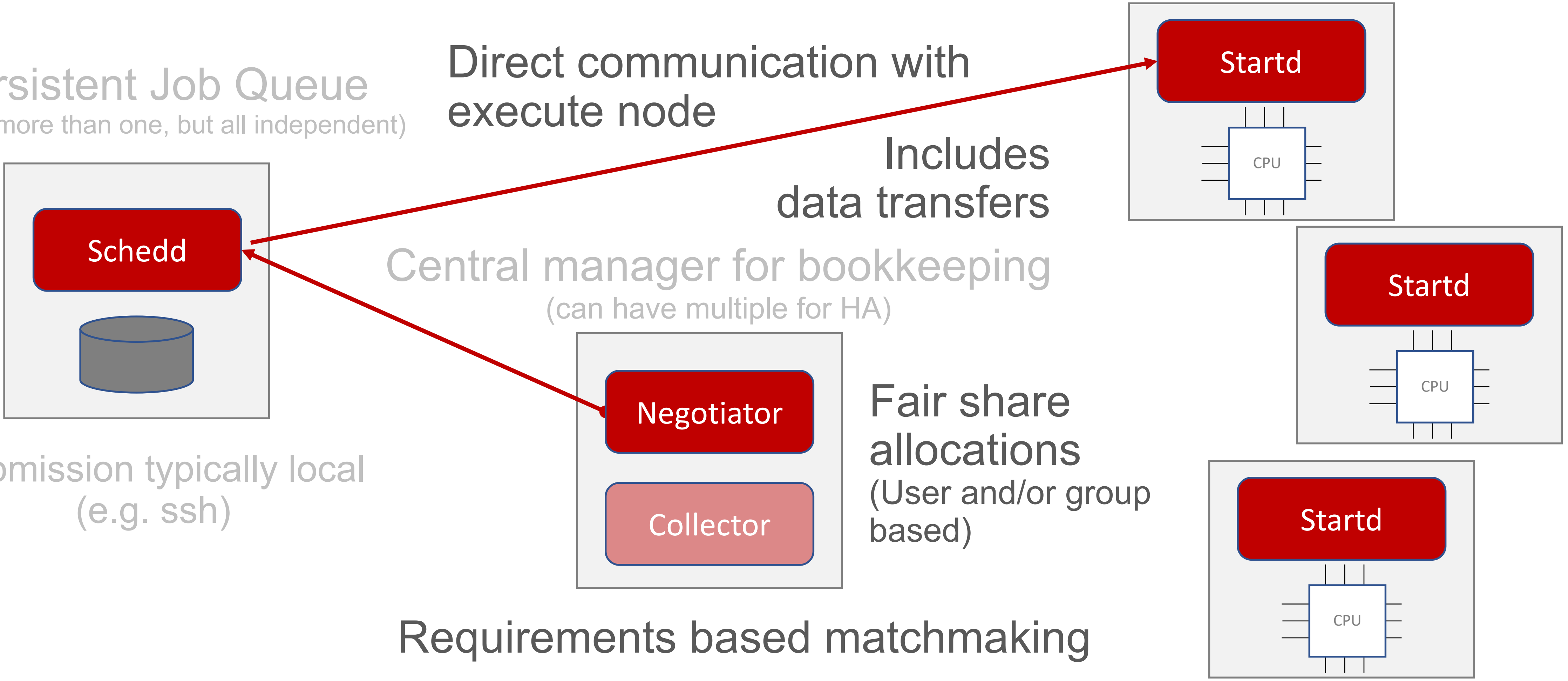
Direct communication with execute node

Includes data transfers

Central manager for bookkeeping
(can have multiple for HA)

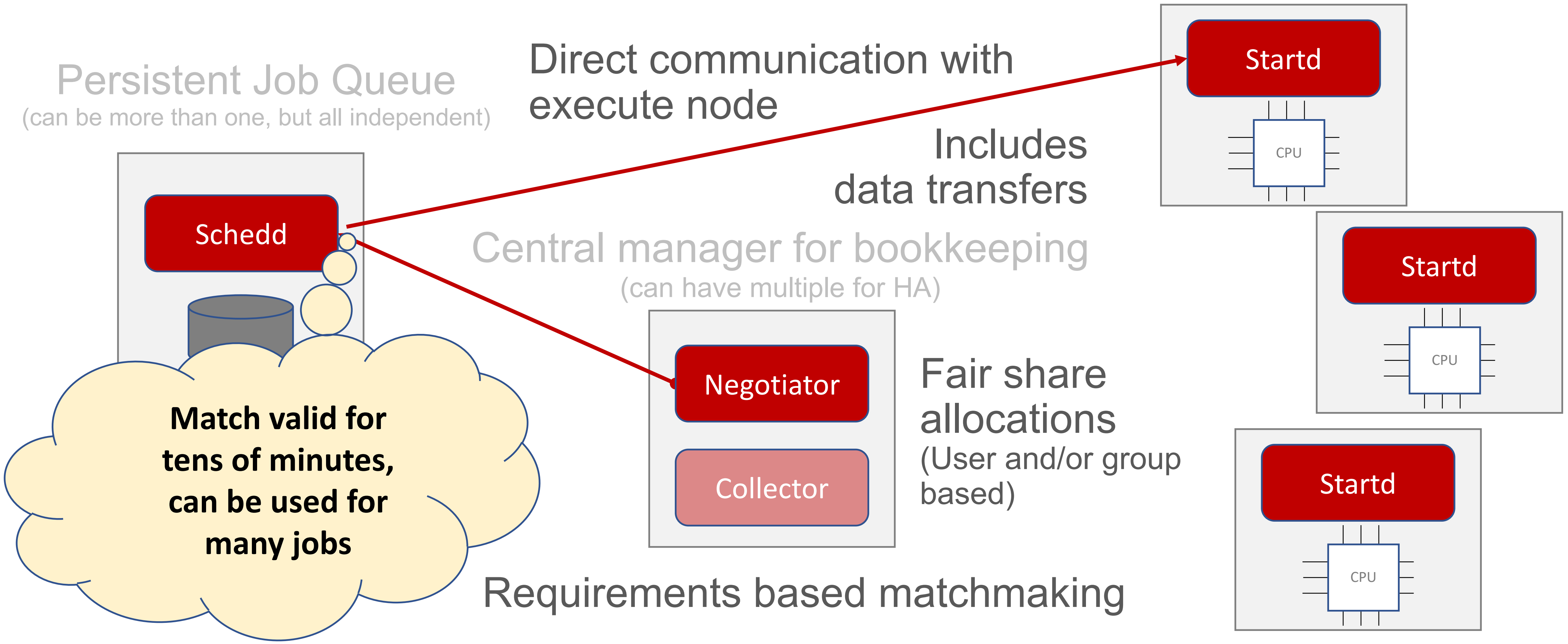
Fair share allocations
(User and/or group based)

Submission typically local
(e.g. ssh)



Requirements based matchmaking

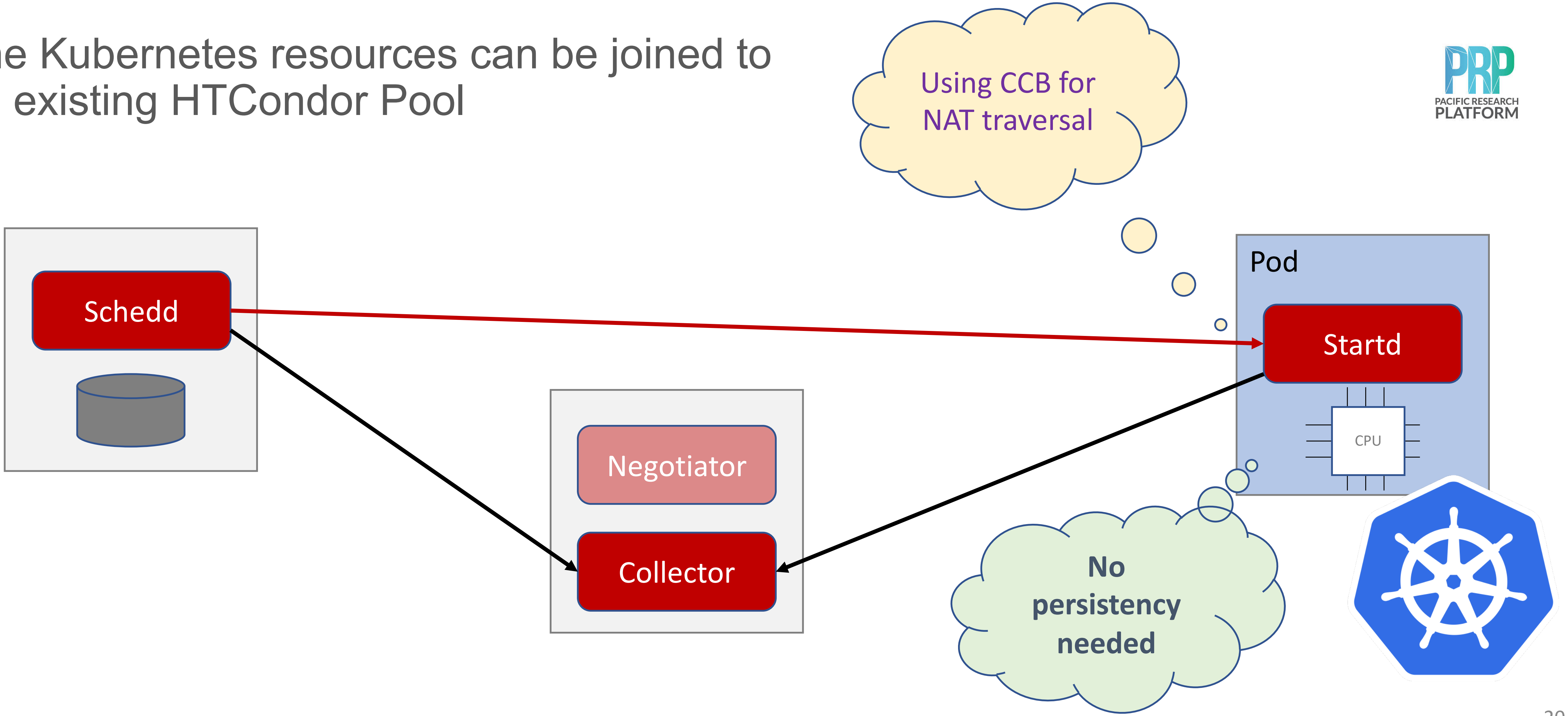
HTCondor Architecture


Each execute resource has a control process
Persistent Job Queue
(can be more than one, but all independent)
Direct communication with execute node
Includes data transfers
Central manager for bookkeeping
(can have multiple for HA)
Schedd
Negotiator
Collector
Startd
CPU
Startd
CPU
Startd
CPU
Match valid for tens of minutes, can be used for many jobs
Fair share allocations
(User and/or group based)

Requirements based matchmaking

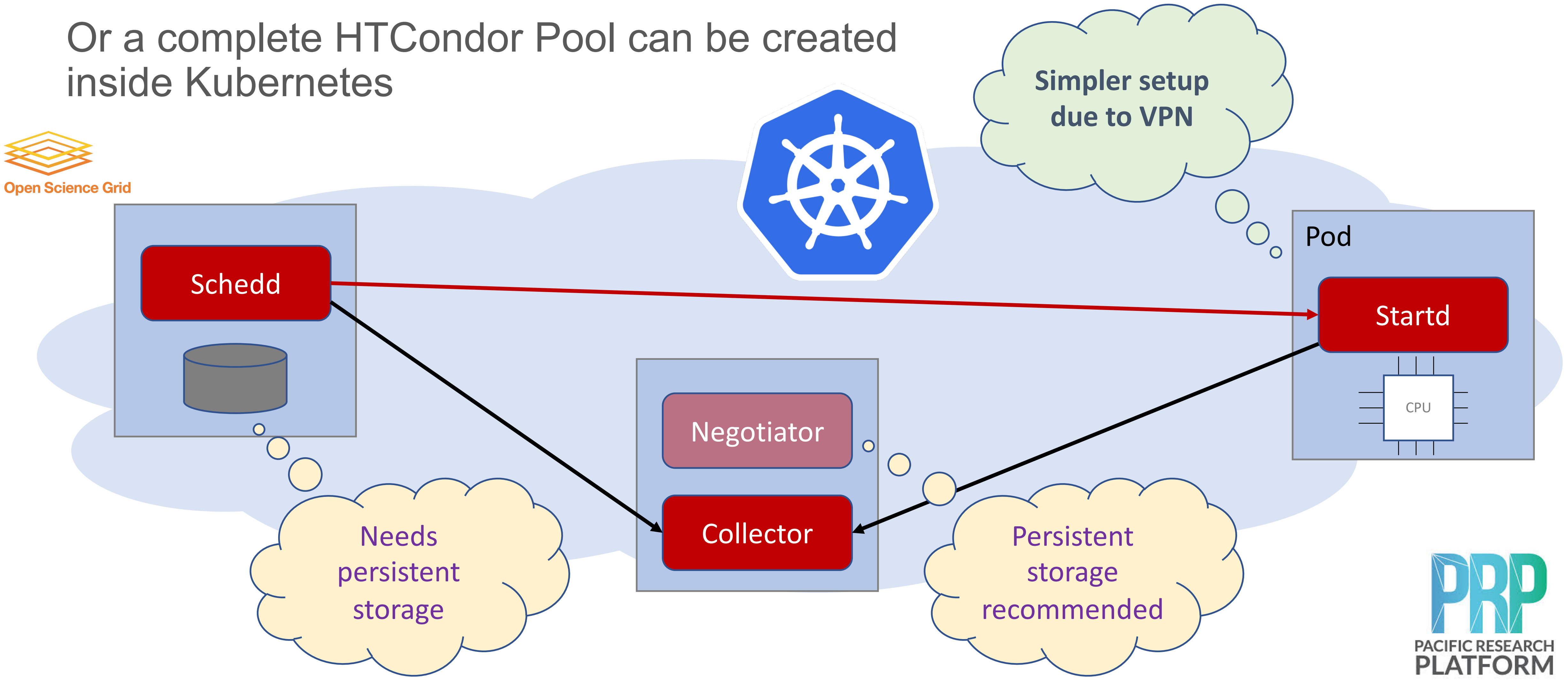
Using HTCondor with Kubernetes

The Kubernetes resources can be joined to an existing HTCondor Pool



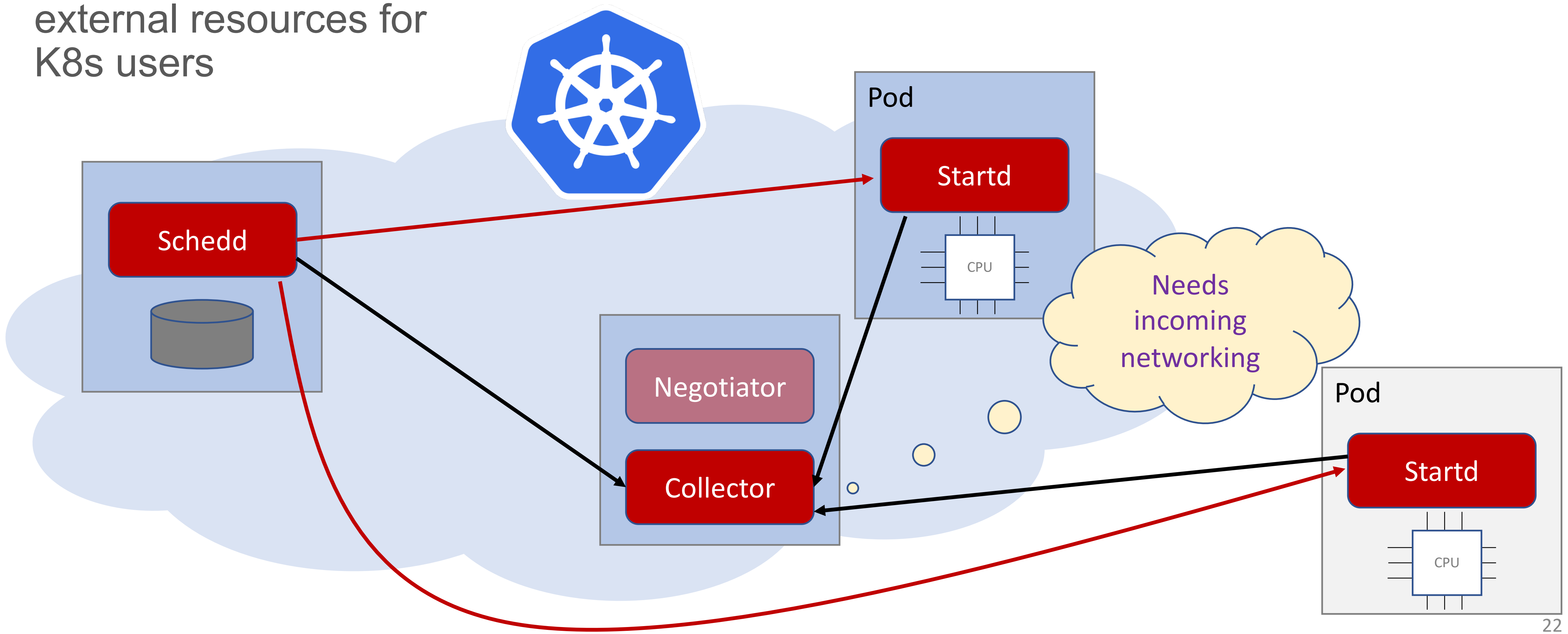
Using HTCondor with Kubernetes

Or a complete HTCondor Pool can be created inside Kubernetes



Using HTCondor with Kubernetes

Can be used to join external resources for K8s users



HTC Users 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 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

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

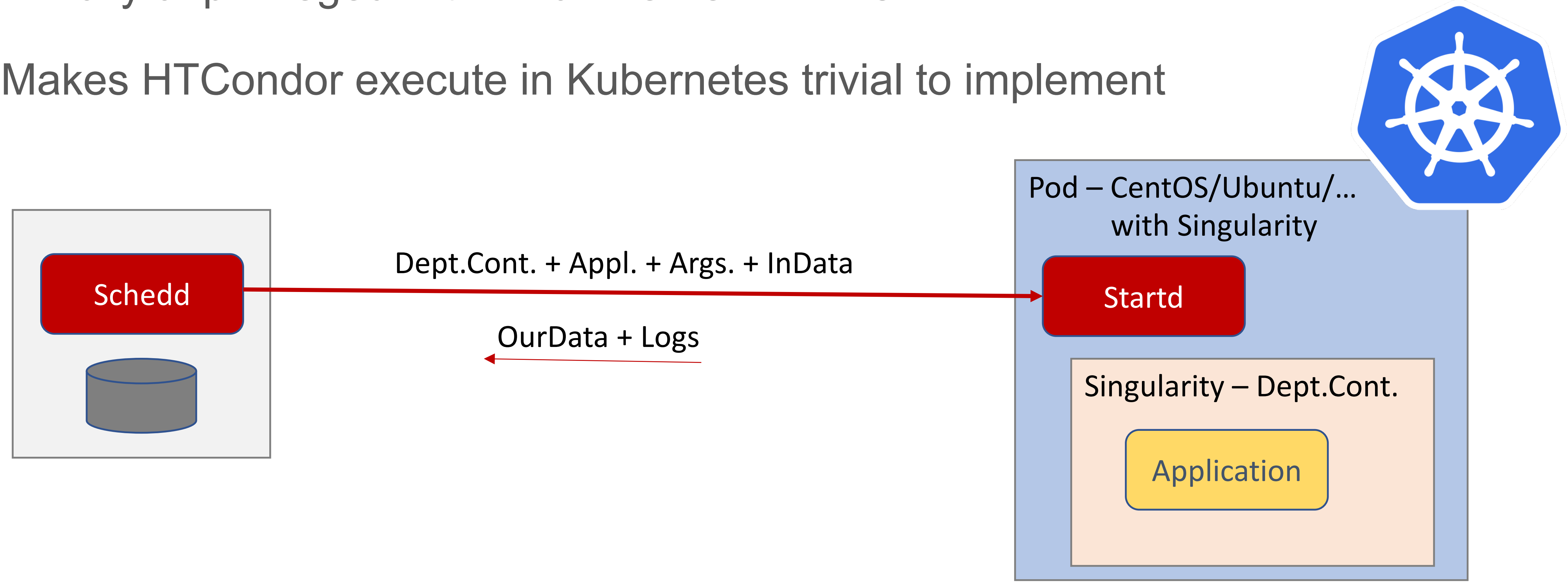
In principle Docker could be an option, but not currently supported

Nested containerization

Singularity can be invoked inside a Docker container

- Fully unprivileged with Linux Kernel >= 4.18

Makes HTCondor execute in Kubernetes trivial to implement


Schedd
Dept.Cont. + Appl. + Args. + InData
OurData + Logs
Pod – CentOS/Ubuntu/… with Singularity
Startd
Singularity – Dept.Cont.
Application

Explicit provisioning

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

Quite effective when only a few Container Images needed

Opportunistic use

Most HTC jobs tolerate preemption

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

Open Science Grid

Just launch HTCondor execute Pods with a very low K8s priority

Works best when you have a single backfill pool

PRP
PACIFIC RESEARCH PLATFORM

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.