

# Dynamic Integration and Management of Opportunistic Resources for HEP

CHEP 2018 | 2018-07-12

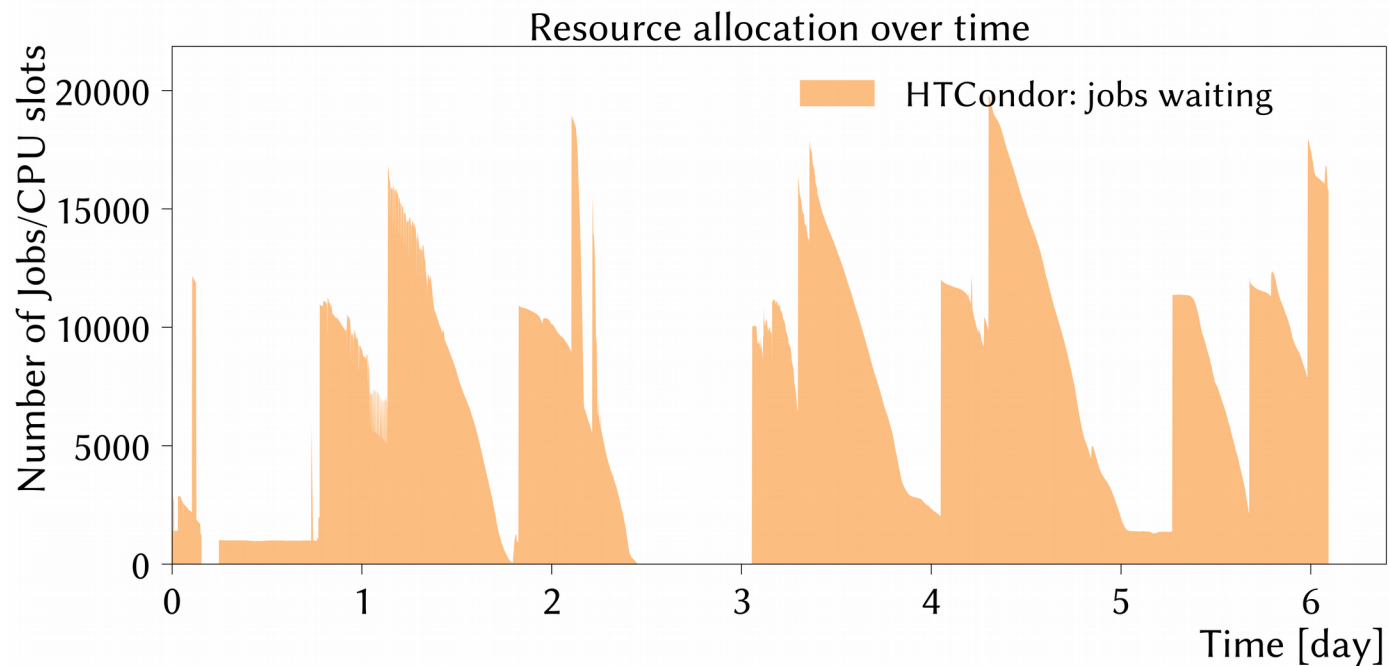
**Matthias J. Schnepf**, Max Fischer, Manuel Giffels, Christoph Heidecker, Andreas Heiss, Eileen Kuhn, Andreas Petzold, Günter Quast

SCC / ETP KIT



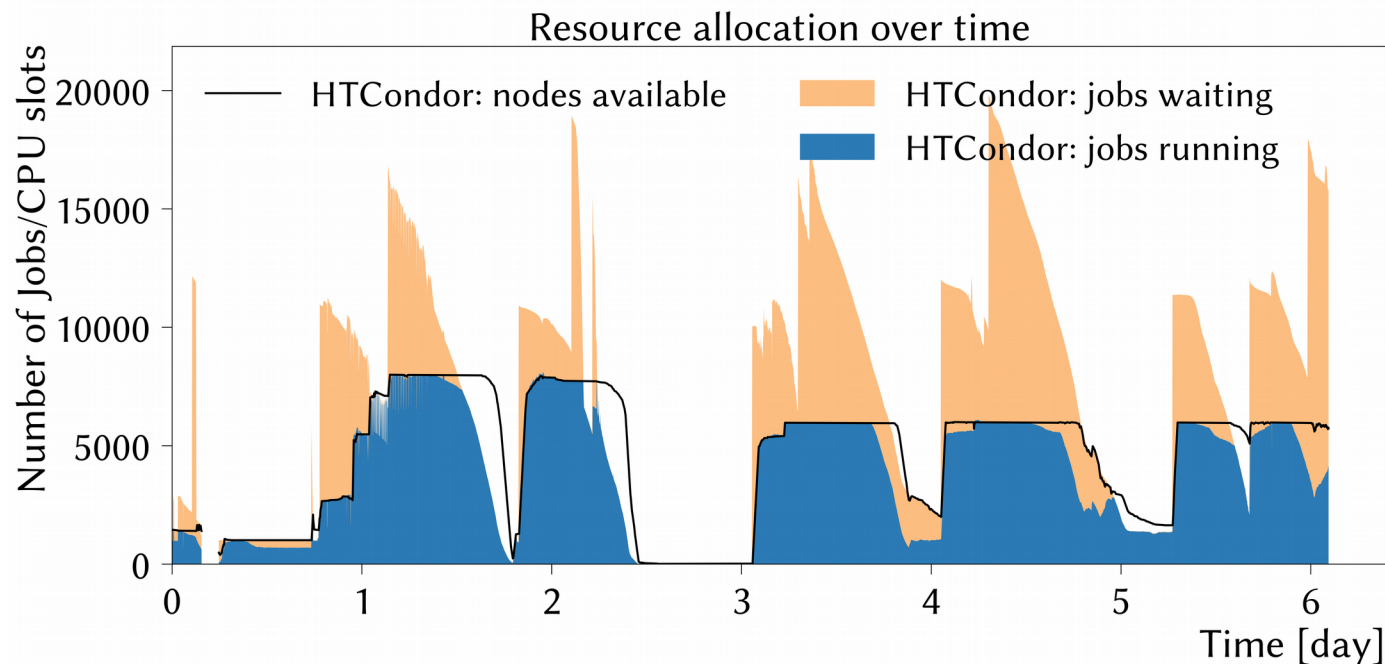
# Typical Situation at Institutes

- demand of resources has peak loads due to time schedules
- want to avoid long waiting times



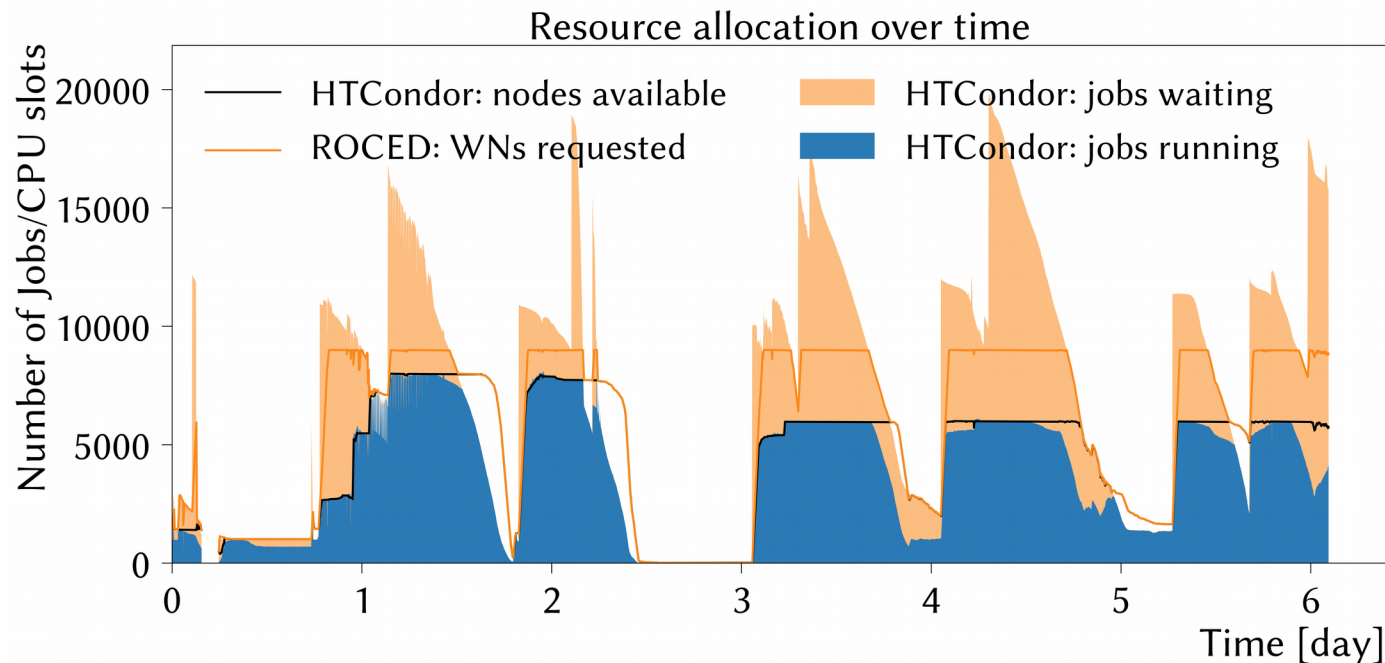
# Typical Situation at Institutes

- demand of resources has peak loads due to time schedules
- want to avoid long waiting times
- ⇒ integrate additional opportunistic (non-HEP) resources on-demand



# Typical Situation at Institutes

- demand of resources has peak loads due to time schedules
- want to avoid long waiting times
- ⇒ integrate additional opportunistic (non-HEP) resources on-demand
- resource allocation managed by ROCED (KIT development)



# Challenges Integrating Opportunistic Resources

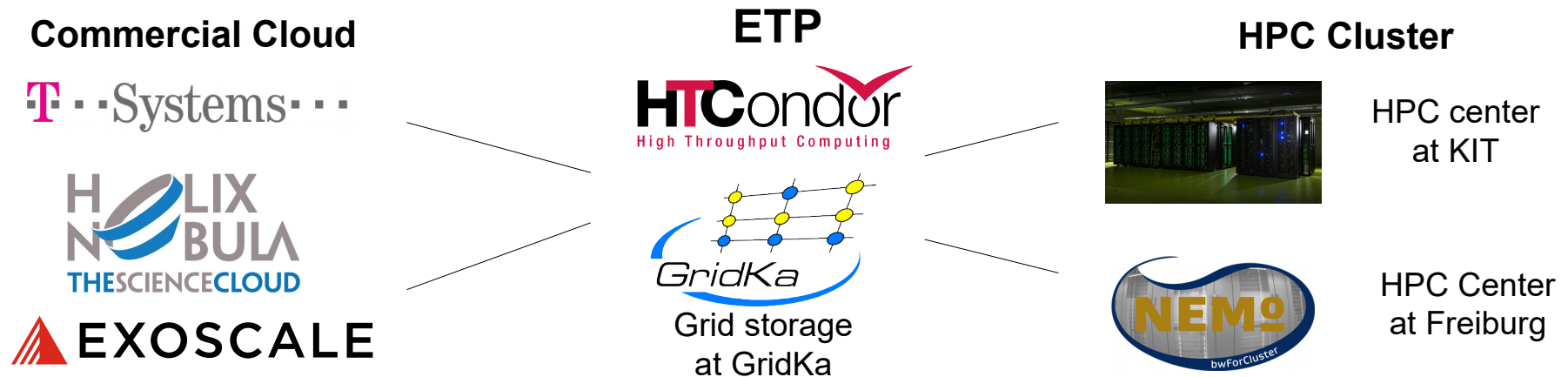
- allocation and integration of resources
- management of heterogeneous resources
- provisioning of software environment
- data transfers for jobs

# Challenges Integrating Opportunistic Resources

- allocation and integration of resources
- management of heterogeneous resources
- provisioning of software environment
- data transfers for jobs
- challenges depends on resource provider
  - HPC cluster
  - commercial cloud provider

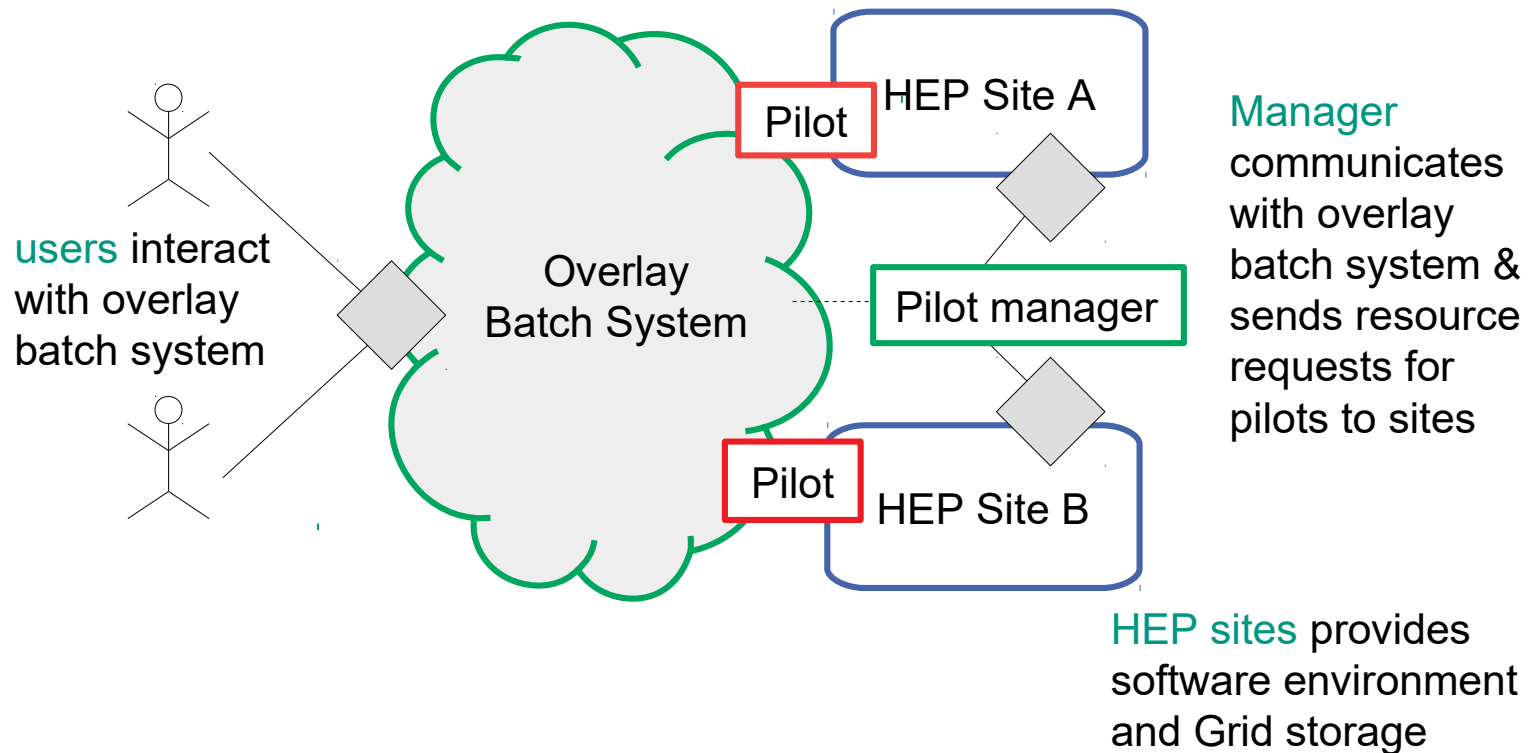
# Challenges Integrating Opportunistic Resources

- allocation and integration of resources
- management of heterogeneous resources
- provisioning of software environment
- data transfers for jobs
- challenges depends on resource provider
  - HPC cluster
  - commercial cloud provider



# Resource Allocation: Pilot Concept

**Pilot** allocates and manages a fix amount of resources for the overlay batch system





# More Generalized Approach: Drone Concept

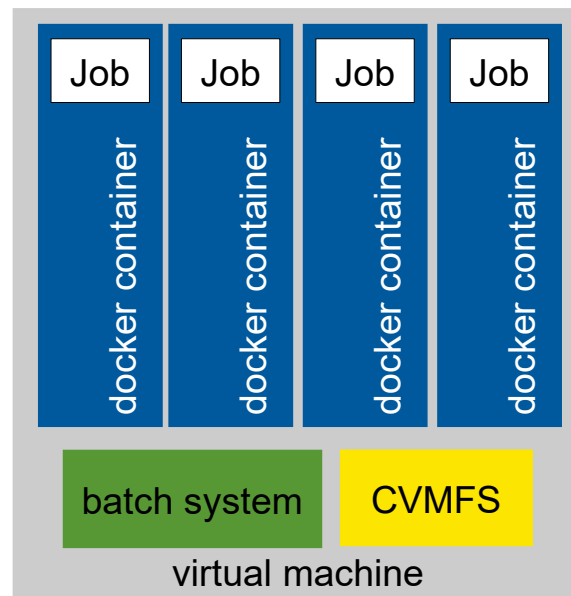
- pilots
  - allocate resources for overlay batch system
  - start overlay batch system worker
  - check environment
  - shutdown and release resources automatically

# More Generalized Approach: Drone Concept

- pilots
  - allocate resources for overlay batch system
  - start overlay batch system worker
  - check environment
  - shutdown and release resources automatically
- “drones”
  - our more generalized approach
  - include features of pilots
  - provide software environment
    - SLC6 / CC7
    - CVMFS
  - adapted for each resource provider

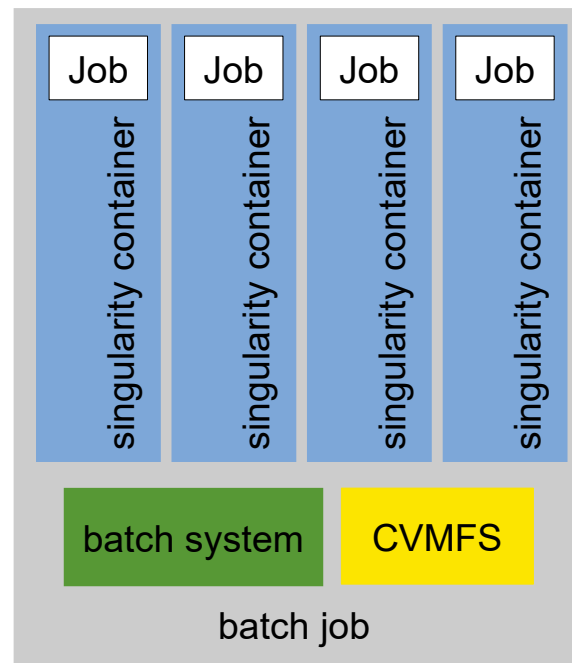
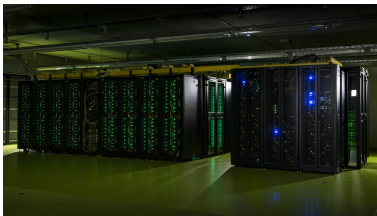
# Example Drones: OpenTelekomCloud

- virtual machine provides CVMFS and overlay batch system worker
- docker containers
  - provide different software environments
  - enable network monitoring per job
- all docker containers inside a VM use local CVMFS cache
- setup requires extra frontier squid for CVMFS



# Example Drones: HPC @ KIT

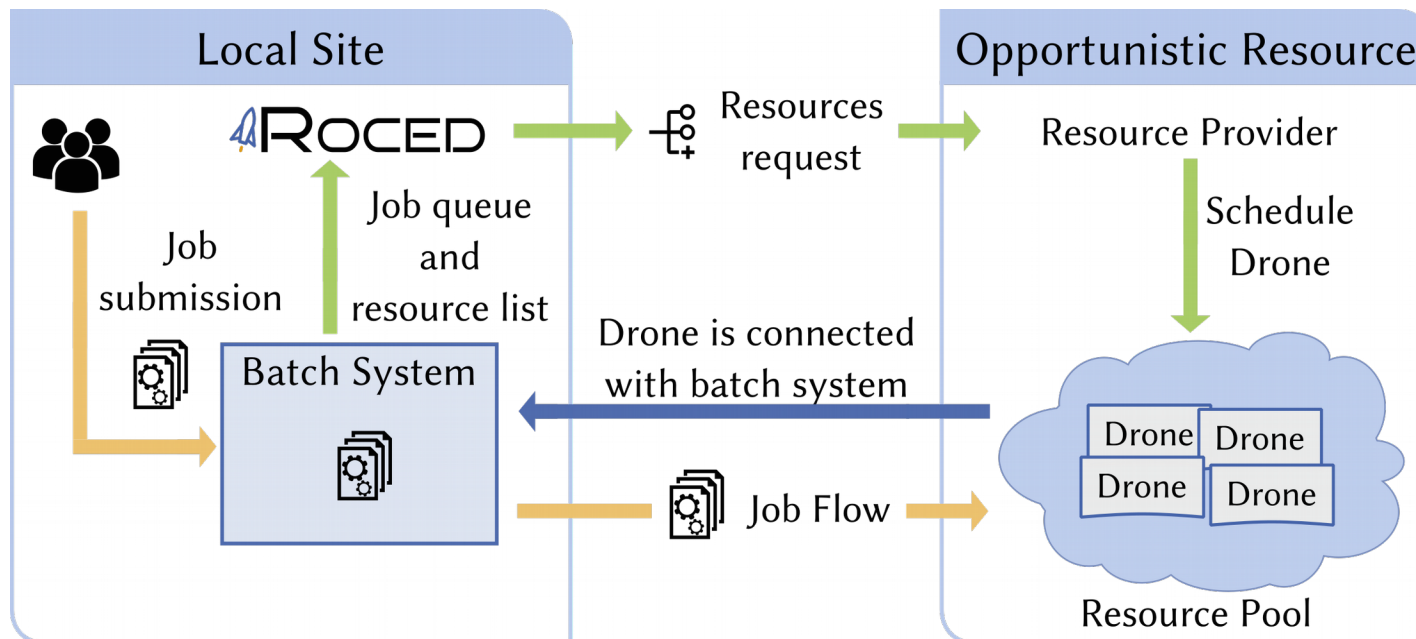
- batch job provide CVMFS and overlay batch system worker
- singularity containers provide different software environments
- use HPC file system for shared CVMFS cache



# Resource Manager: ROCED

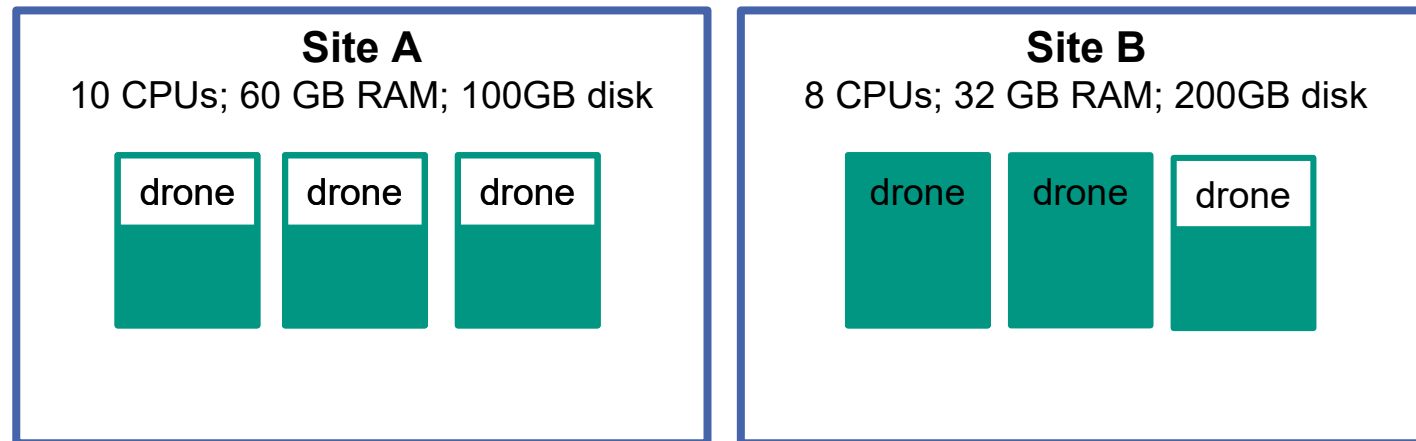
Responsive **O**n-Demand **C**loud-enabled **D**eployment

- management and integration of additional resources via drones on-demand
- support for multiple batch systems and resource providers
- simple scheduling of drones to resource providers based on demand of CPU cores



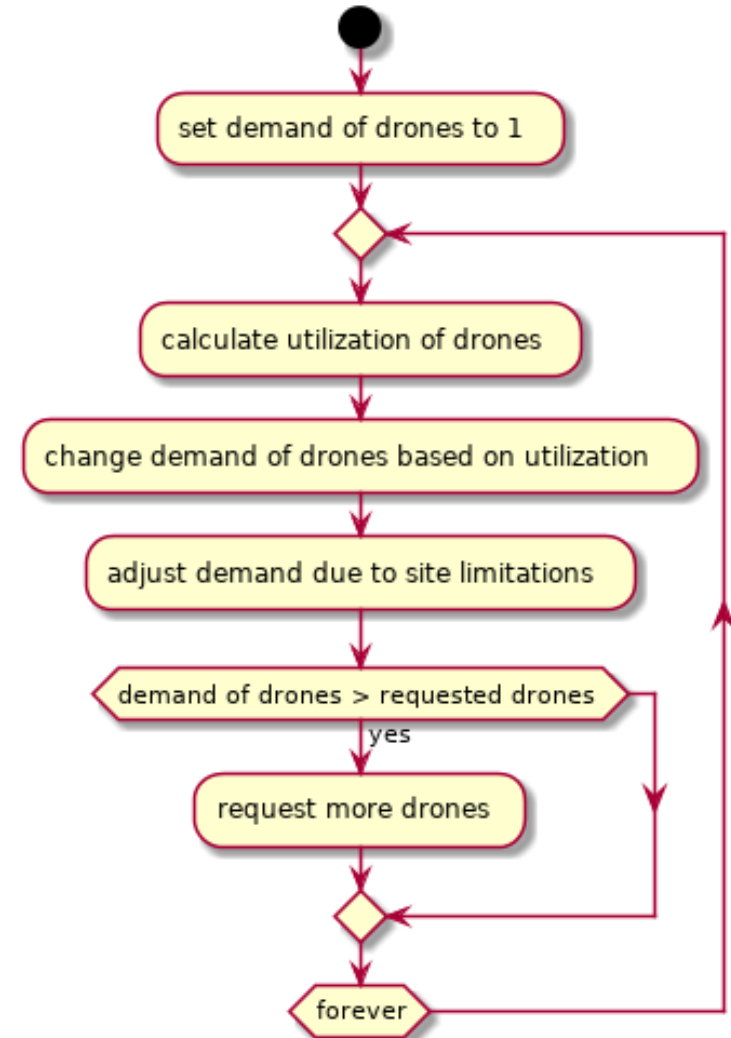
# Restrictions Caused by Simple Drone Scheduling

- HTCondor batch systems reacts on current situation
- difficult to predict the scheduling decision of the batch system
- ROCED currently treats all sites equally
- inefficient utilization on drones which do not fit to current jobs



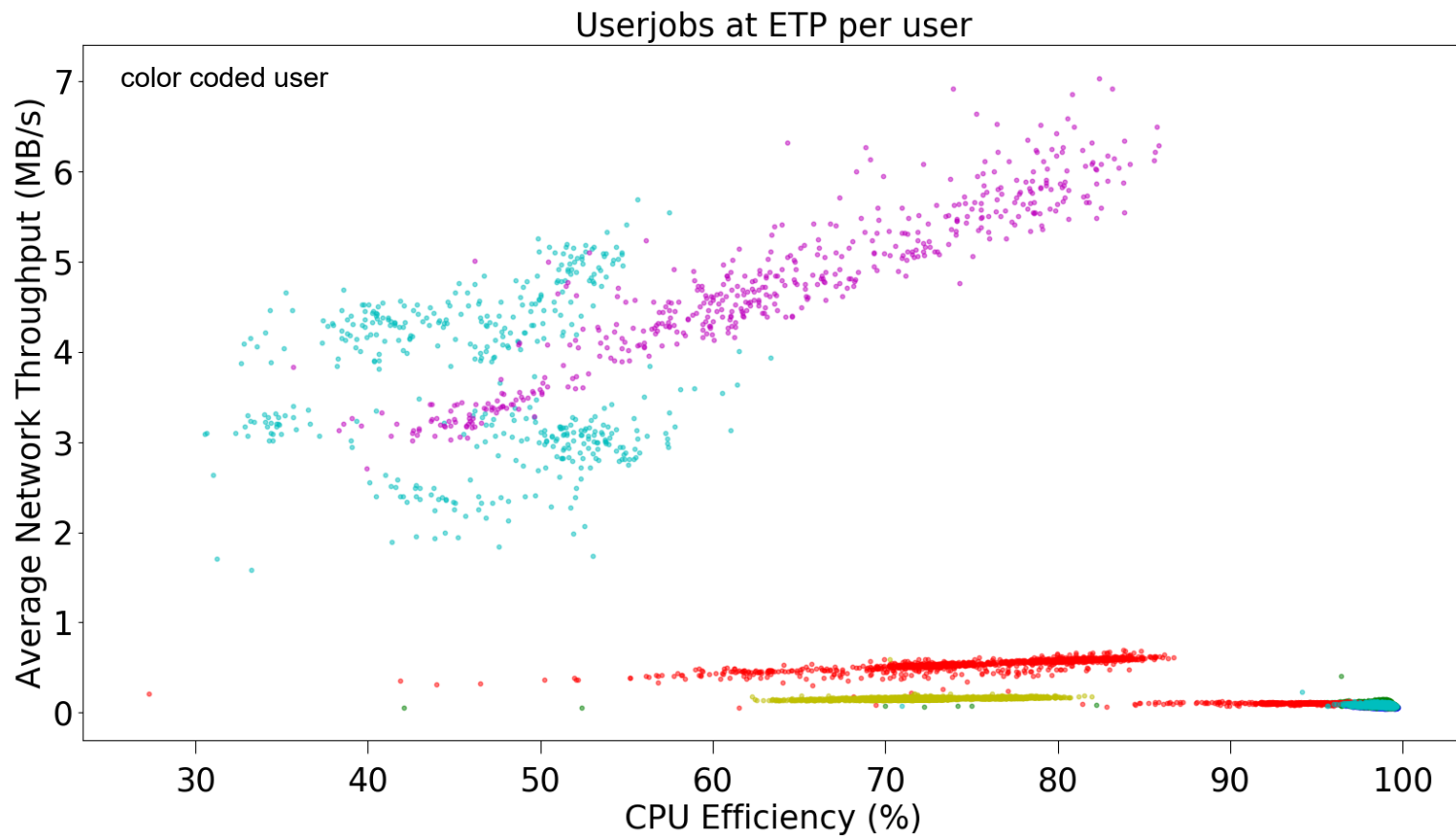
# Resource Scheduling via Feedback Loop

- our new approach: resource scheduler reacts on batch system scheduling via feedback
- currently in development
- scheduling decision based on utilization instead of demand per site
  - adopt scheduling of drones to resource restrictions
    - max amount of CPUs
    - costs
    - network limitations



# Snapshot of Typical End-User Jobs

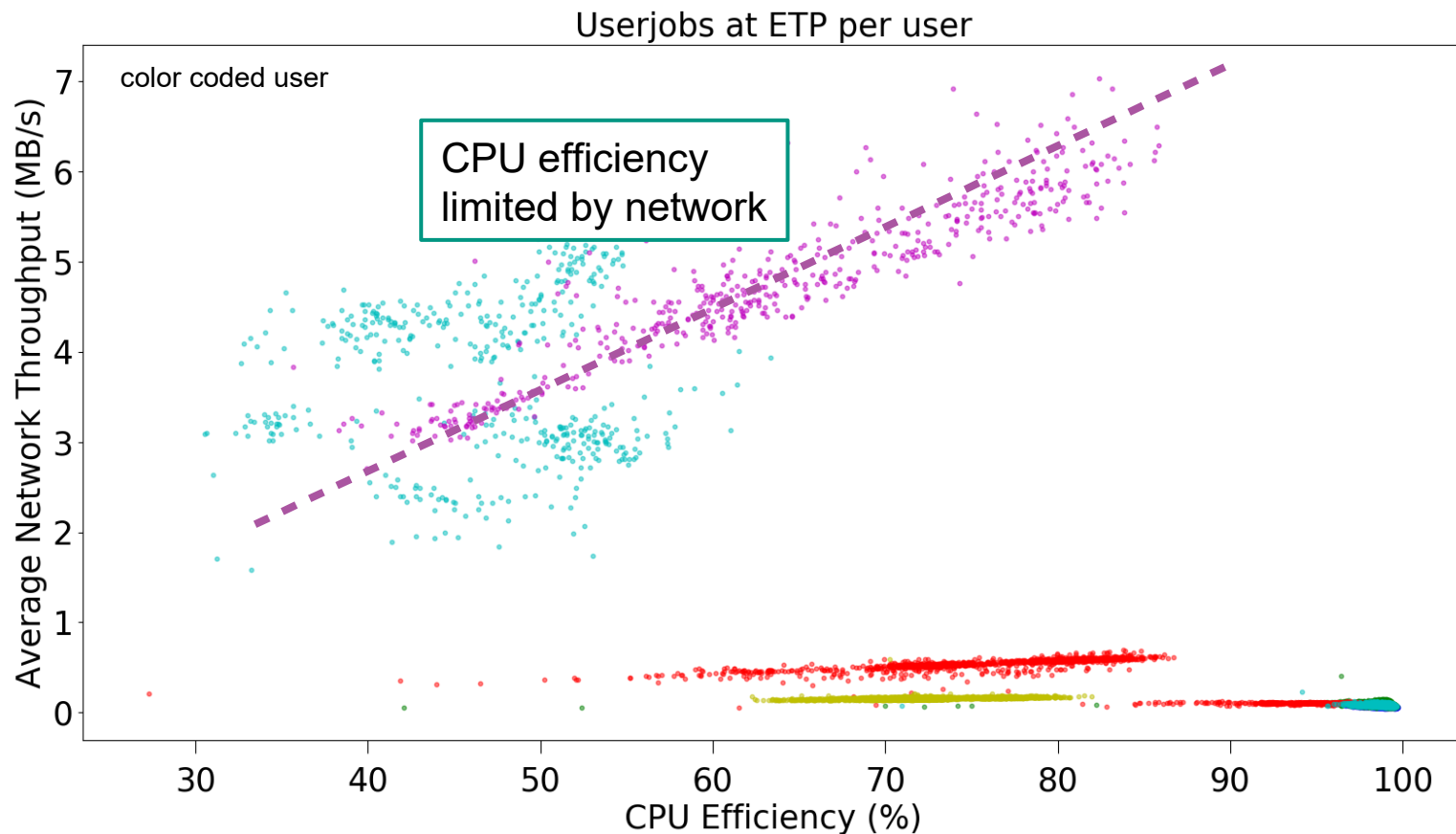
- HTCondor and docker enable network monitoring





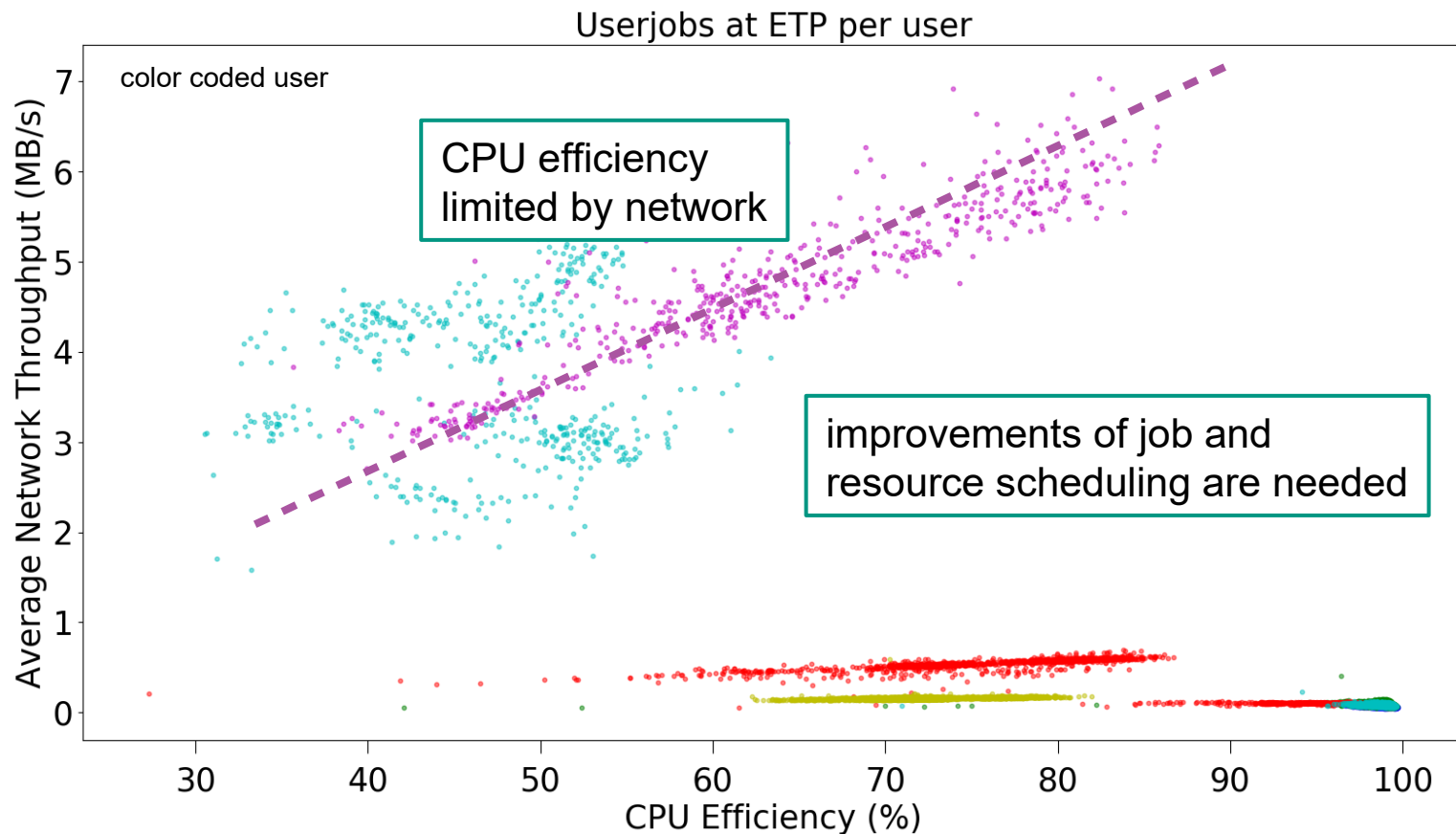
# Snapshot of Typical End-User Jobs

- HTCondor and docker enable network monitoring



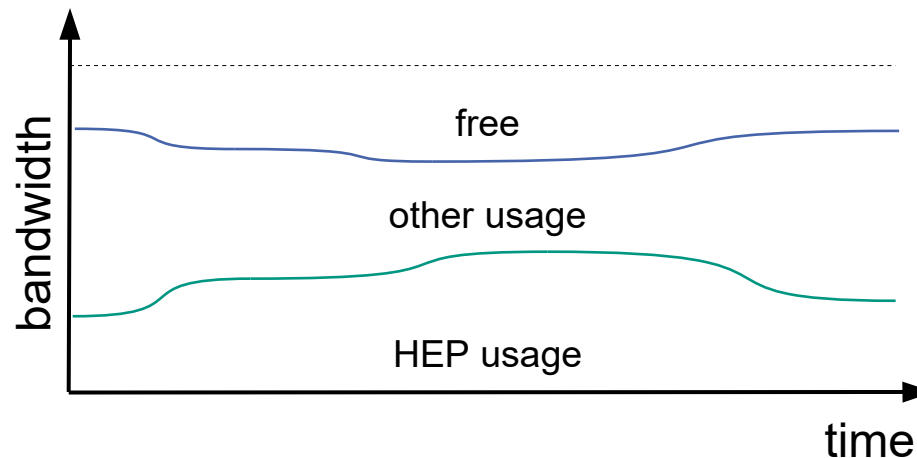
# Snapshot of Typical End-User Jobs

- HTCondor and docker enable network monitoring



# Scheduling using Network Information

- own network usage known by job monitoring
- schedule data intensive jobs to resources which have enough bandwidth to Grid storage
  - sample jobs to estimate network usage for workflow
  - readjust estimated network usage while jobs are running
- available bandwidth not constant
- free bandwidth measurable via benchmark
- use network information to limit amount of allocated resources



# Summary

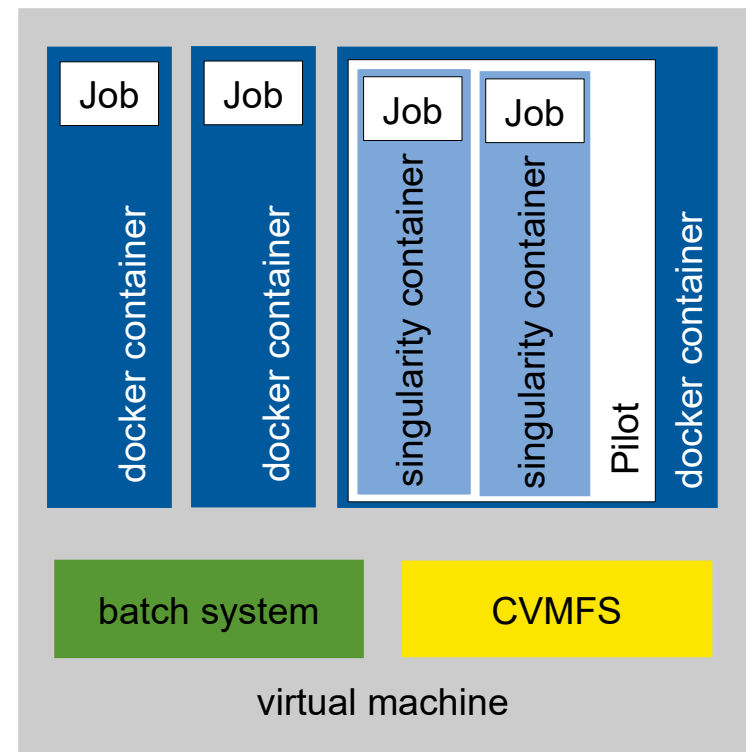
- Drone concept
  - enables a wider range of usable resources compared to pilots
  - uses container and virtualization technologies to provide HEP software environment
- resource scheduling via feedback loop
  - react on drone utilization
  - use information about site for scheduling
  - provides several interfaces for different resource providers
- planned to use network information of sites and jobs for scheduling
  - available bandwidth
  - estimate network throughput for jobs

backup



# Preferred Drone

- virtual machine
  - current standard at cloud provider
  - provide CVMFS and batch system worker node
- docker container
  - provides different software environments
  - enables additional monitoring features
- singularity container
  - provides different software environments
  - required by some VOs



# Job Monitoring

- HTCondor and docker enable network monitoring
- in usage at the physics institute (ETP)

