Fermilab U.S. DEPARTMENT OF ENERGY Office of Science

Scientific Distributed Computing with HEPCloud, GlideinWMS and HTCondor

Marco Mambelli

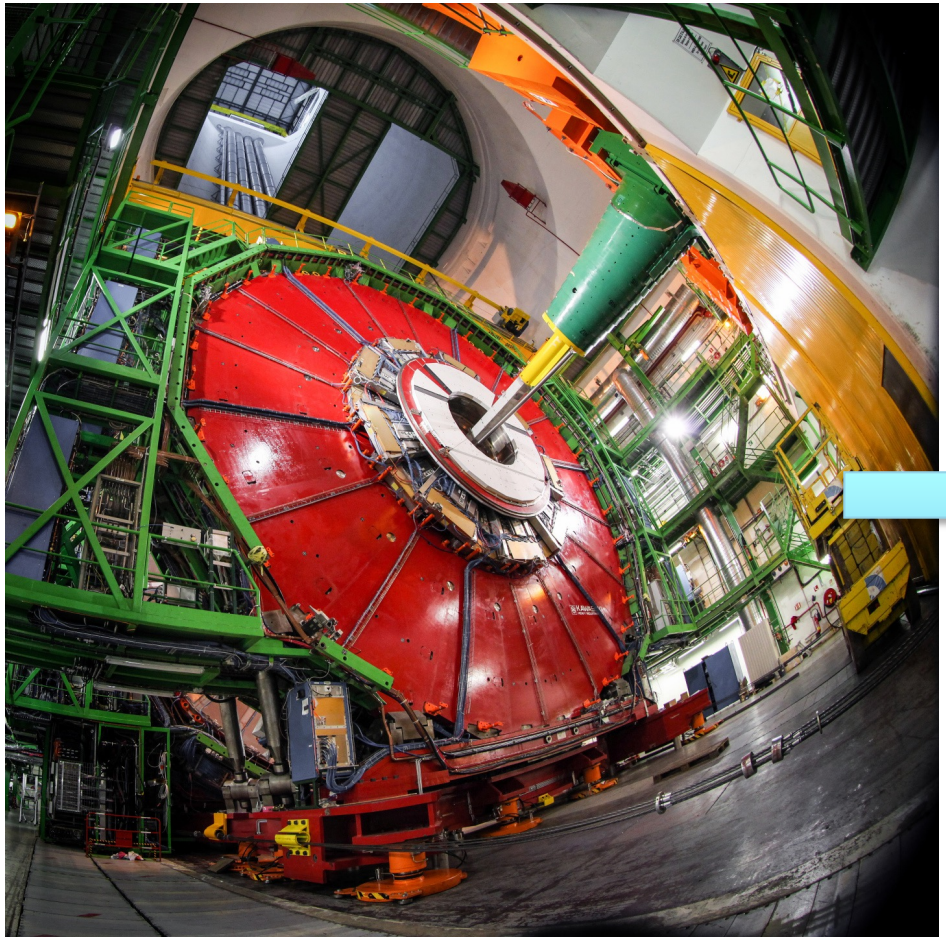
Computational HEP Traineeship Summer School 2024

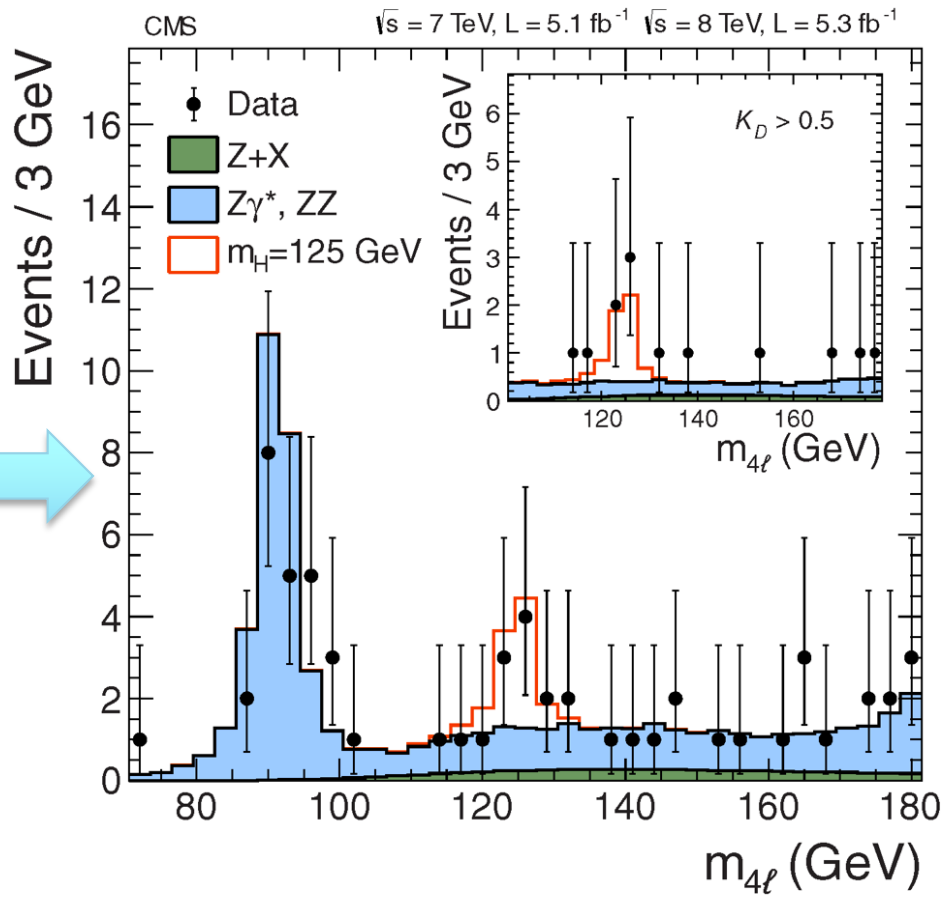
May 23 2024

Outline

- Scientific computing
- Workflows
- Distributed High Throughput Computing
- Pilot-based systems
- GlideinWMS and HEPCloud
- Storage and credentials
- HTCondor
- Resources and job requirements

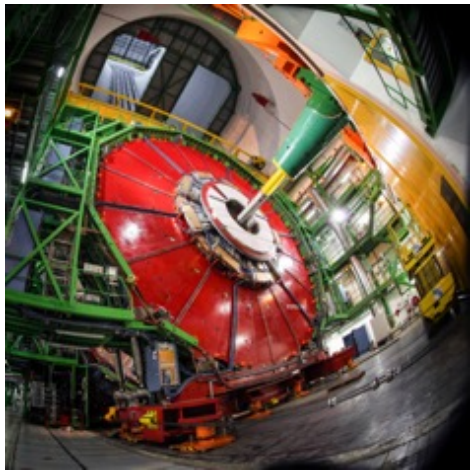
From here … to there





Fermilab

The process



DAQ, Triggering

"Raw" data

Dist. CPU

Monte Carlo simulation

Dist. CPU

reco. software

Dist. CPU

Reconstructed data

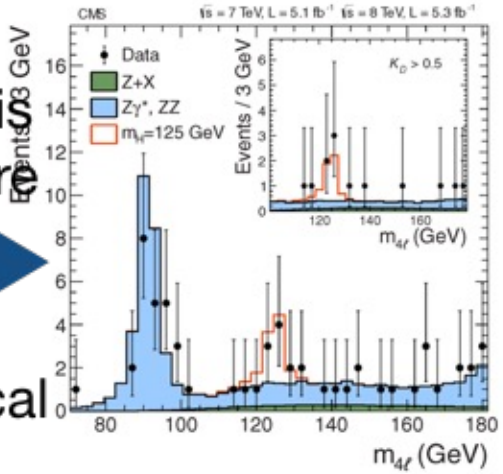
database

Reduced data

analysis software

(further) Reduced data

analysis software



Dist/local CPU

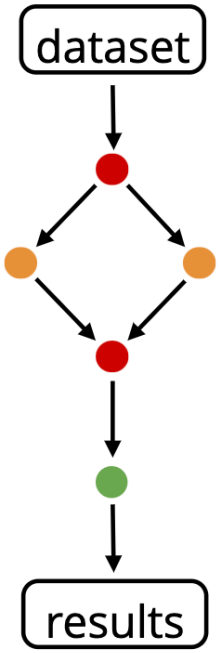
Dist/local CPU

Fermilab

Scientific computing system

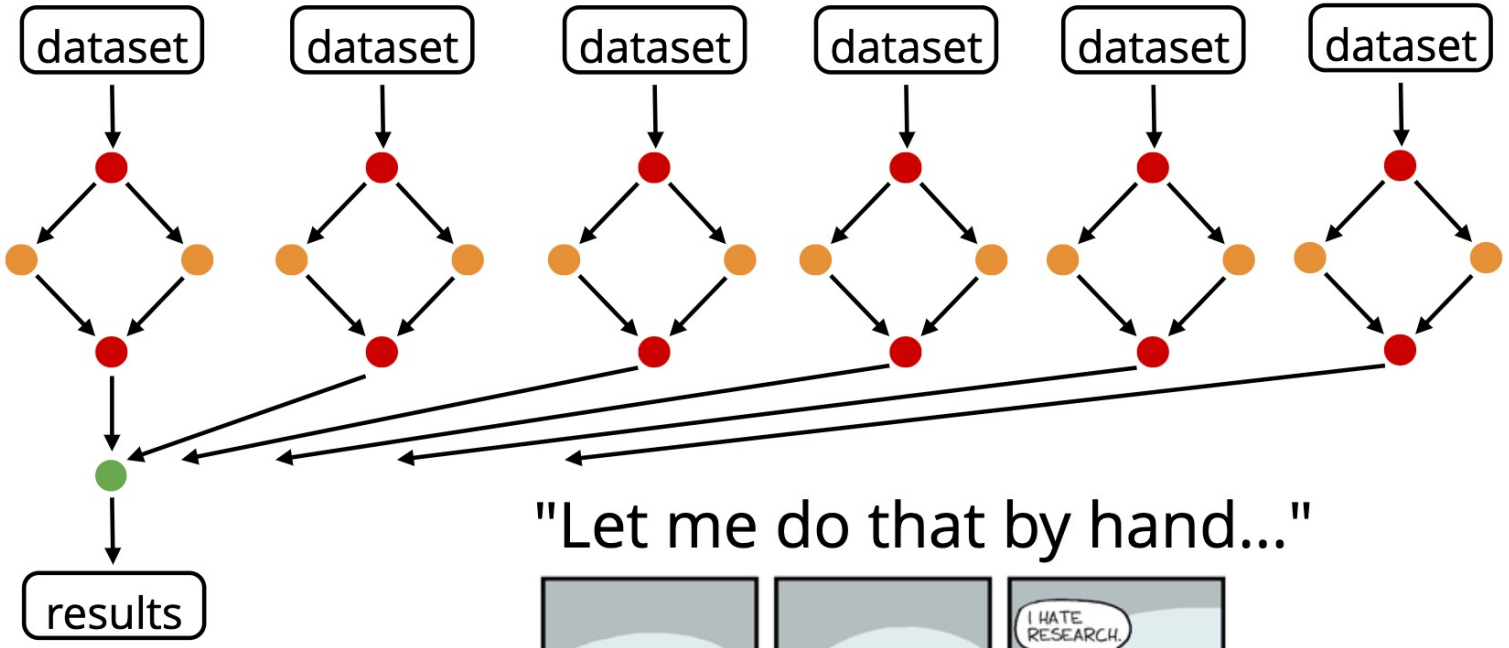
- UI
- Workflow manager
- Workload manager (Resource provisioner)
- Resources (Execution Points)

Wolrflow



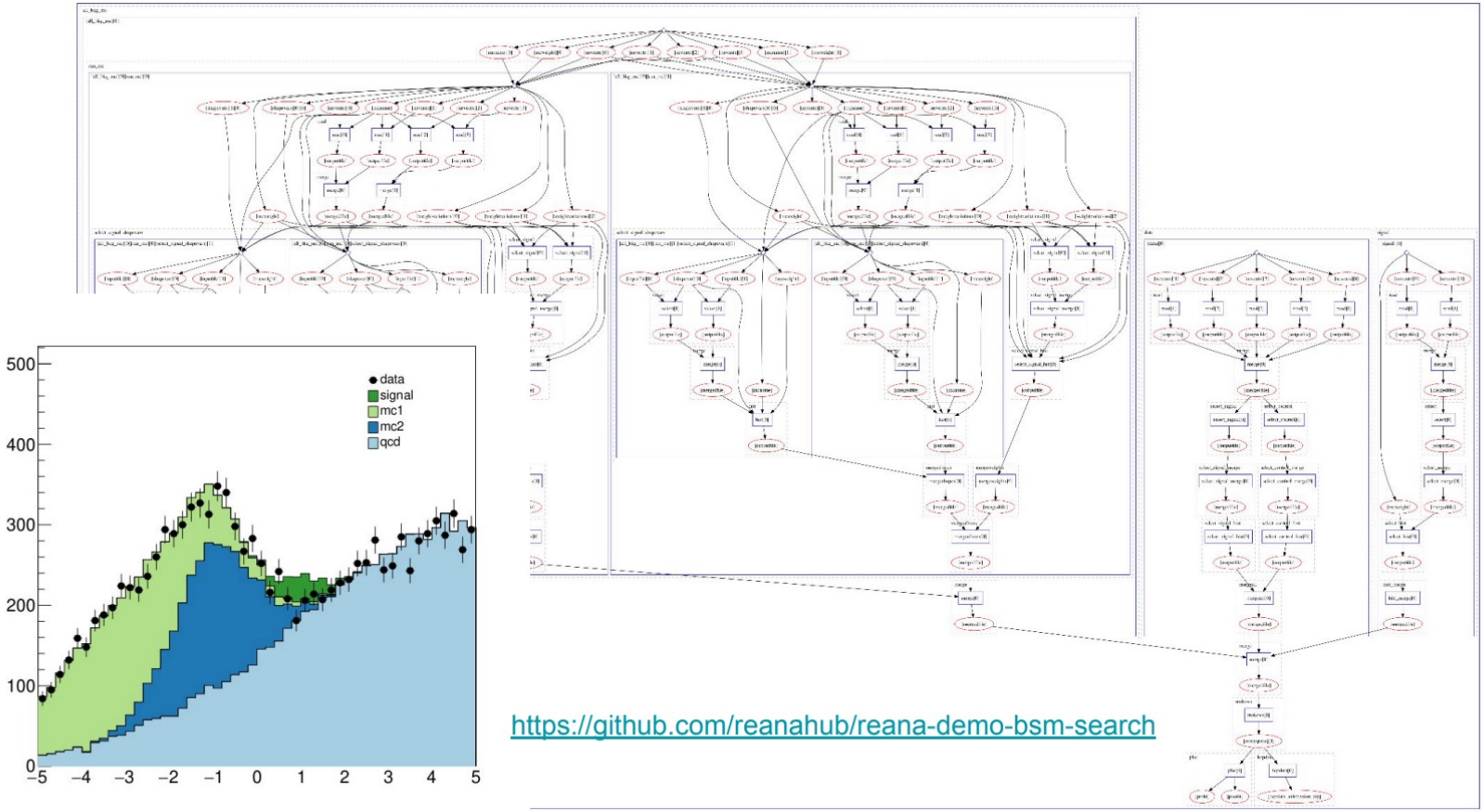
"Let me do that by hand..."

Workflow (cont)



"Let me do that by hand..."

HEP Workflow

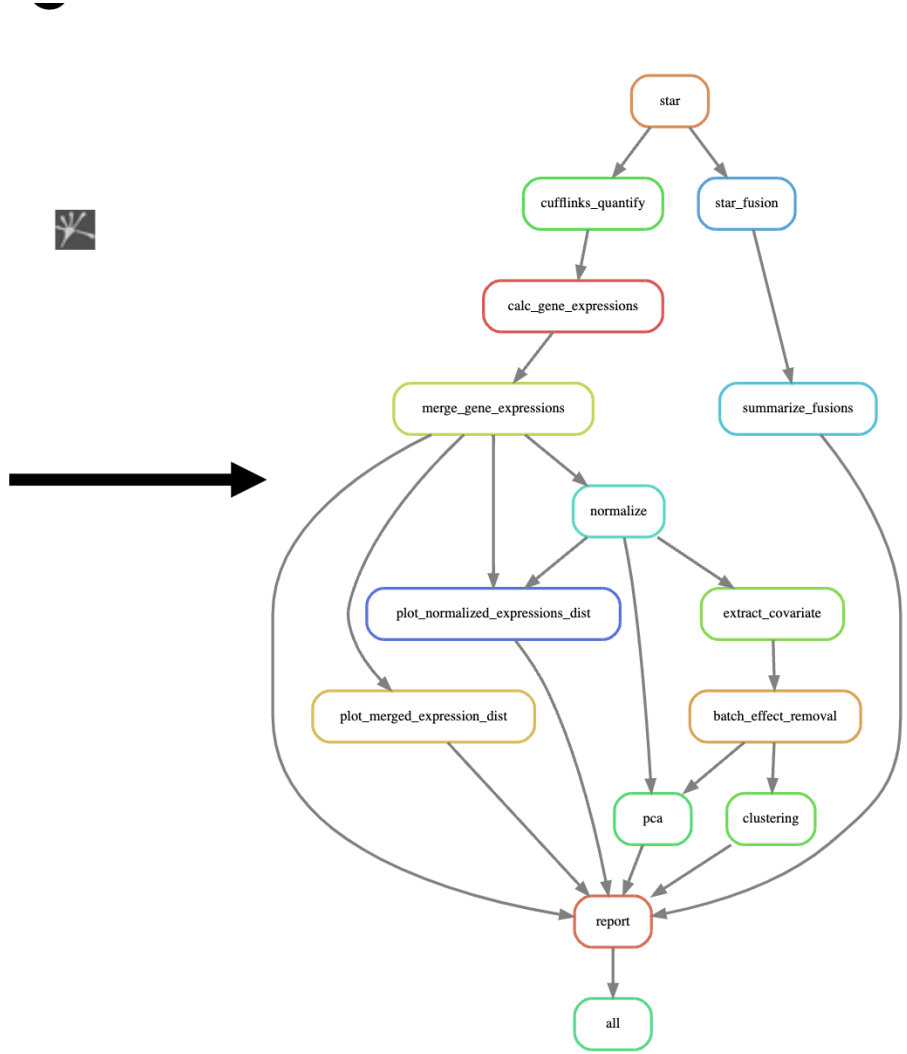


https://github.com/reanahub/reana-demo-bsm-search

Fermilab

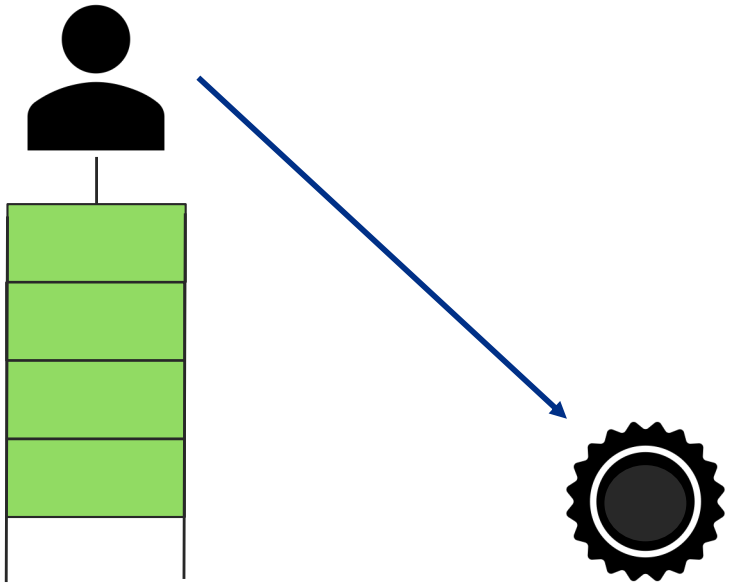
Snakemake

```
rule mytask:
    input:
        "path/to/{dataset}.txt"
    output:
        "result/{dataset}.txt"
    script:
        "scripts/myscript.R"

rule myfiltration:
     input:
        "result/{dataset}.txt"
     output:
        "result/{dataset}.filtered.txt"
     shell:
        "mycommand {input} > {output}"

rule aggregate:
    input:
        "results/dataset1.filtered.txt",
        "results/dataset2.filtered.txt"
    output:
        "plots/myplot.pdf"
    script:
        "scripts/myplot.R"
```

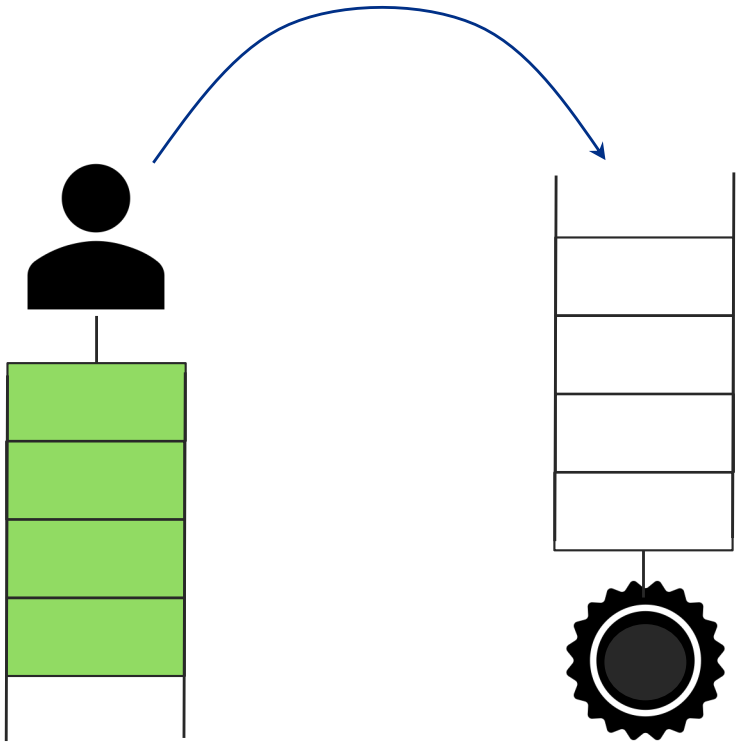


Fermilab

Job, queue

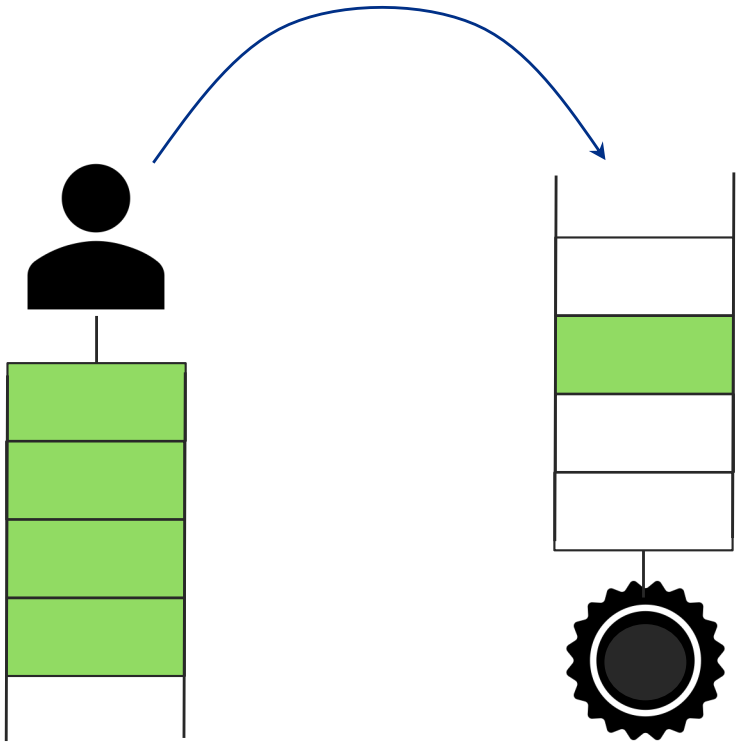


Queues



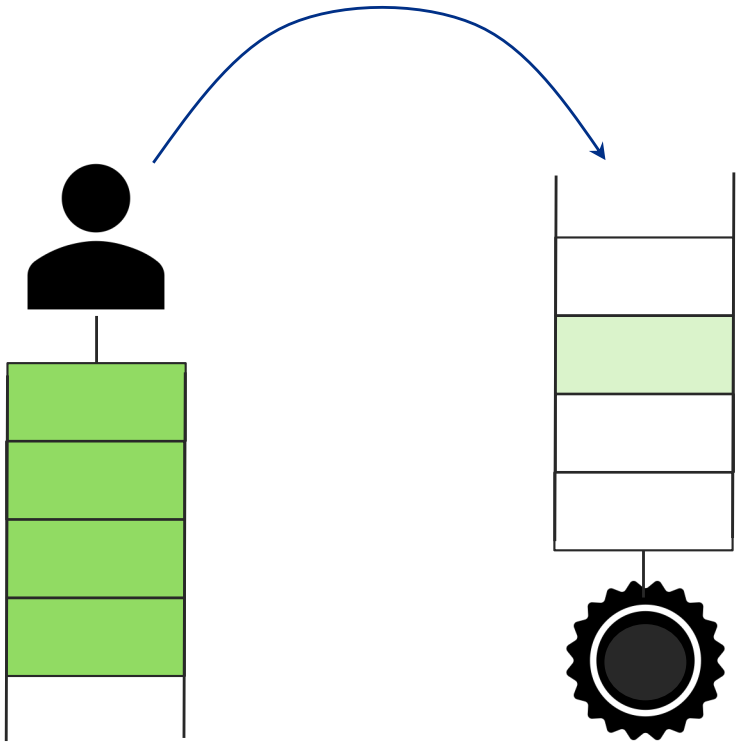
Efficiency

Queues



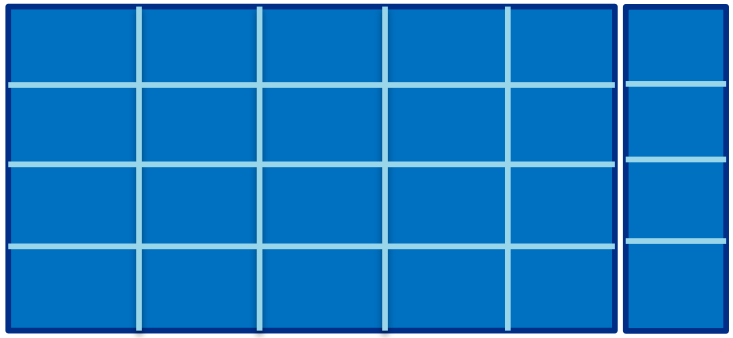
Latency vs Efficiency

Queues



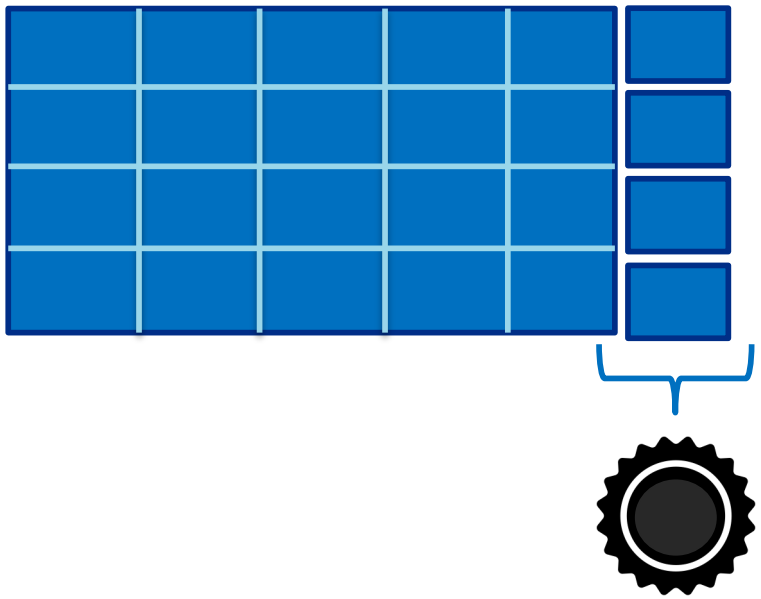
distributed High Throughput Computing (dHTC)

- Tasks split in small pieces (jobs)



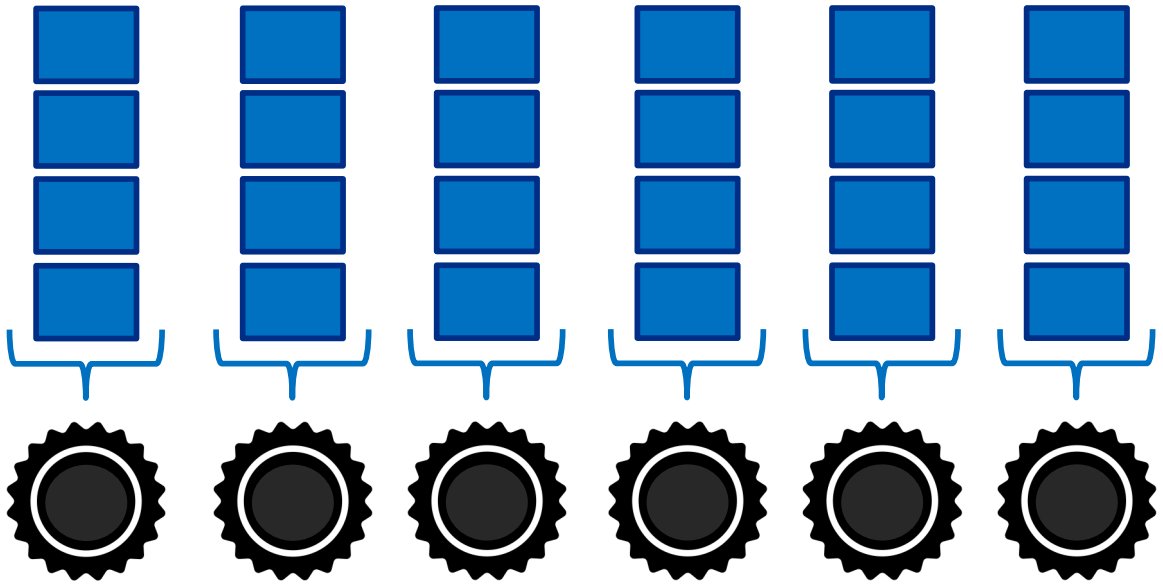
distributed High Throughput Computing (dHTC)

- Tasks split in small pieces (jobs)
- Resource processing queued jobs



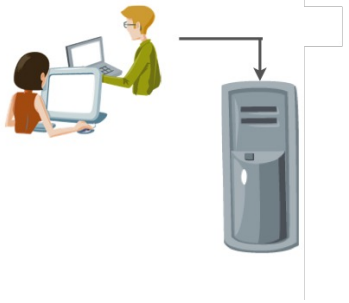
distributed High Throughput Computing (dHTC)

- Tasks split in small pieces (jobs)
- Resource processing queued jobs
- Run many jobs in parallel to shorten completion

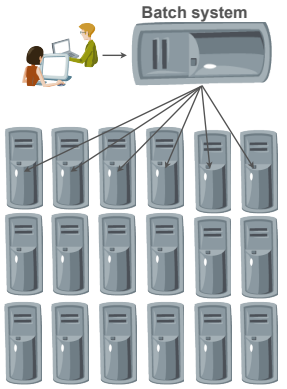


Where jobs run

- Your computer
  - Interactive
  - GUI
  - Your customization
  - Your software

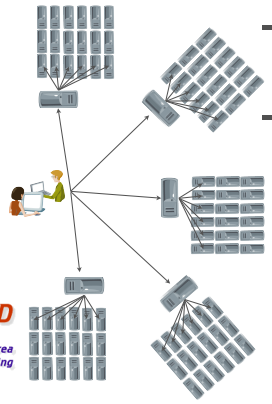


- Institutional cluster
  - Batch queue (SLURM, PBS, HTCondor, SGE, …)
  - Terminal
  - Network access
  - Familiar environment
  - Local support



Where jobs run (2)

- Grid clusters
  - Borrowed resources
  - Network reachable
  - Unknown environment
  - Multi-institution support system
- (Commercial) Cloud
  - Rented resources
  - Virtual machines
  - Cost optimization
  - On demand



Open Science Grid

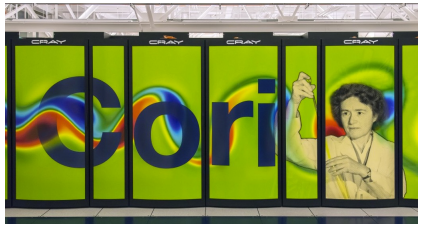
WLCG Worldwide LHC Computing Grid

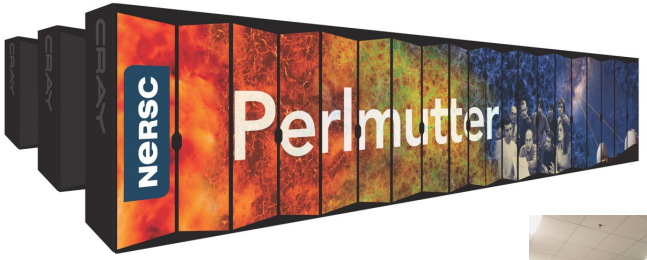
NORDUGRID Grid Solution for Wide Area Computing and Data Handling

Google Compute Engine

Amazon EC2

Where jobs run (3)

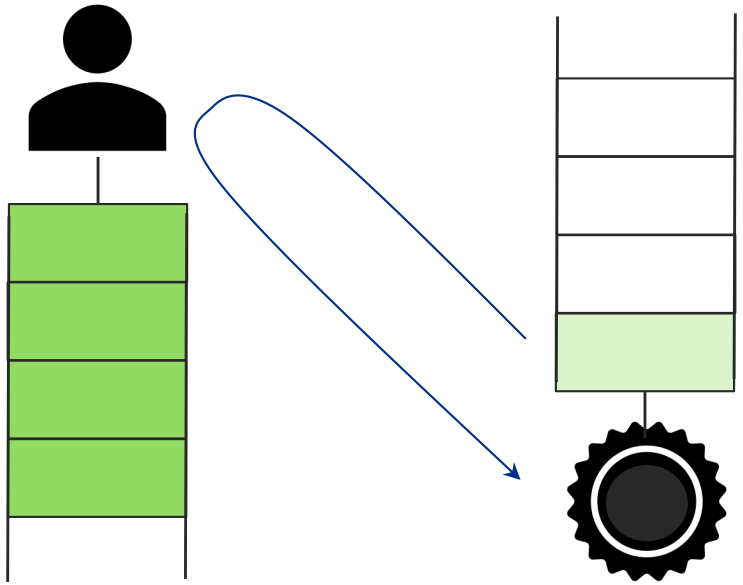




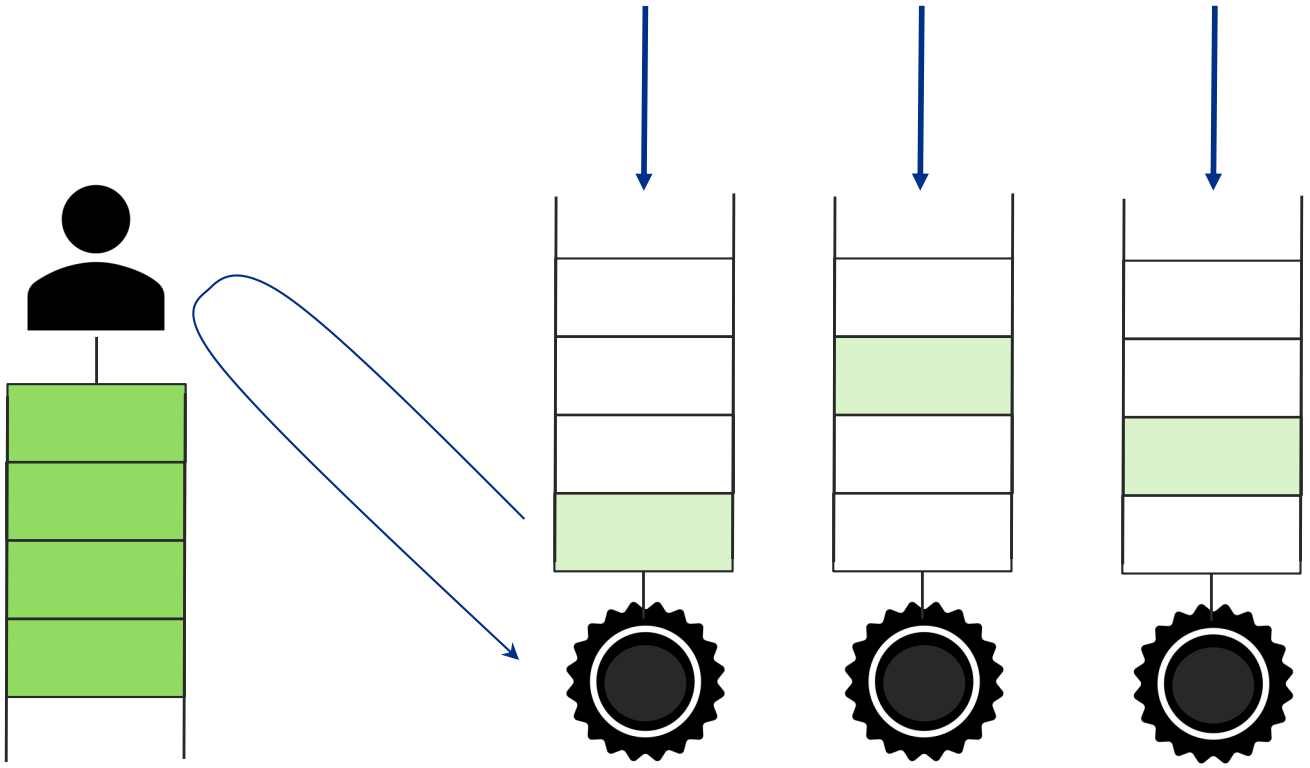
- High Performance Computers (HPC)
  - Each is unique
    - Architecture
    - Network topology
  - Parallel and coupled jobs (MPI)
  - Allocations and long queue times

Pilot jobs (Glideins)

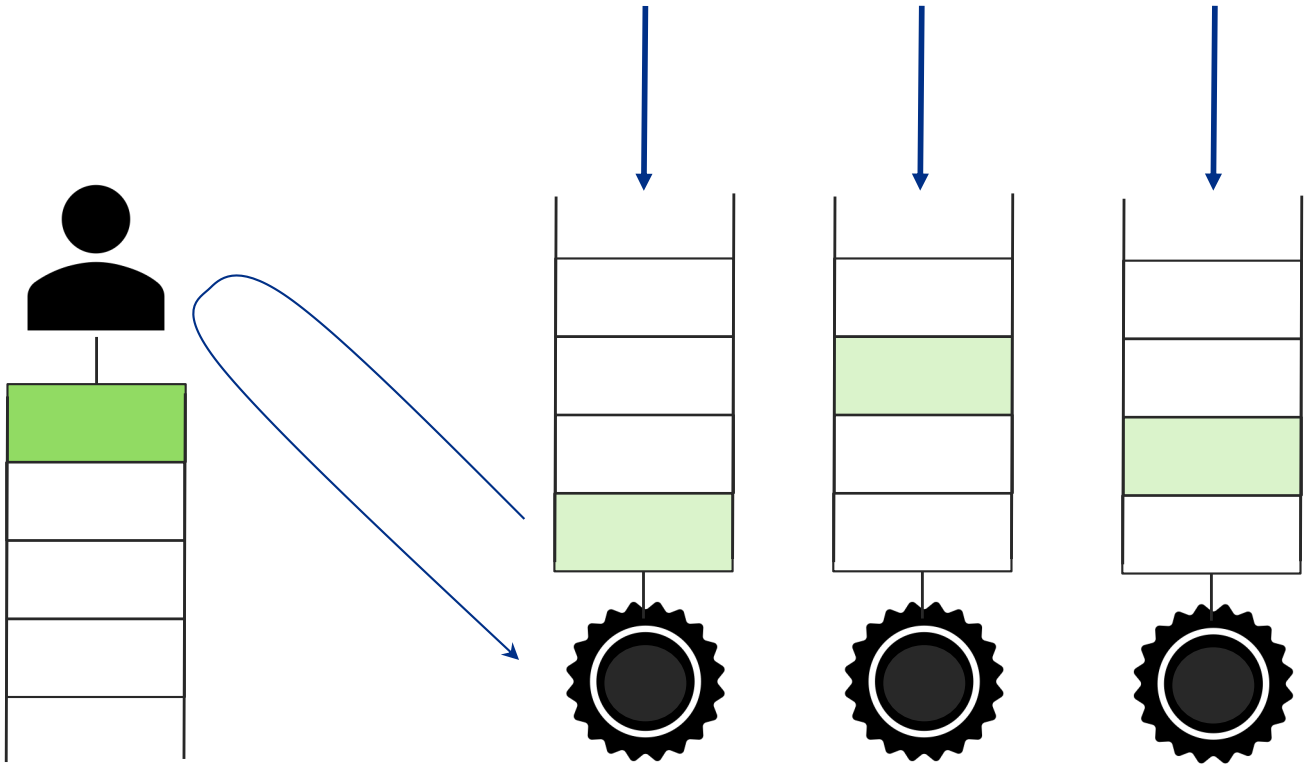
- Separation of tasks
  - Pilot job
    - Test
    - Set up
    - “Expendable”
  - User/real job
    - Science
- Late binding
- Flexible use of multiple resources



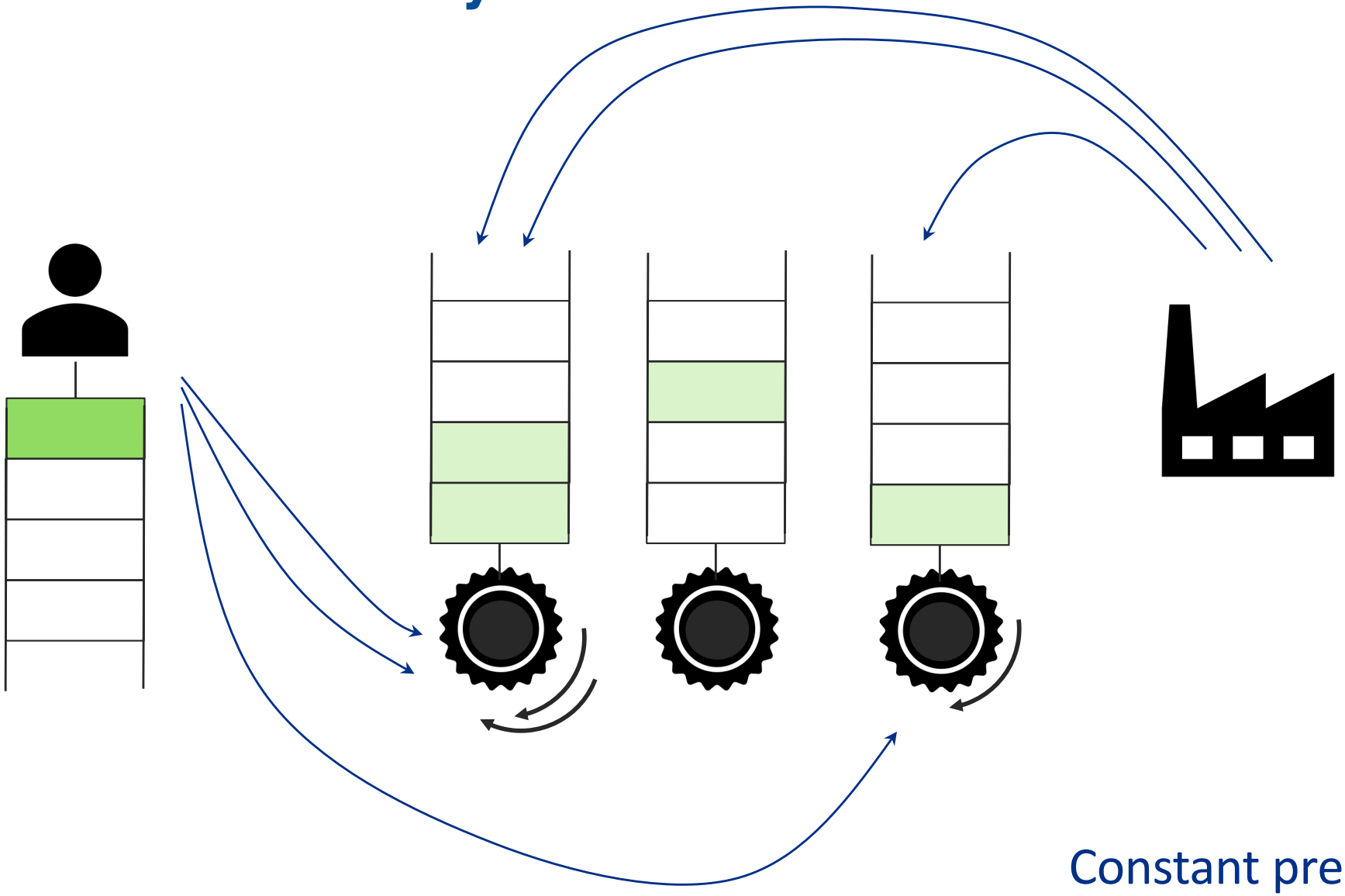
Pilots



Steady state



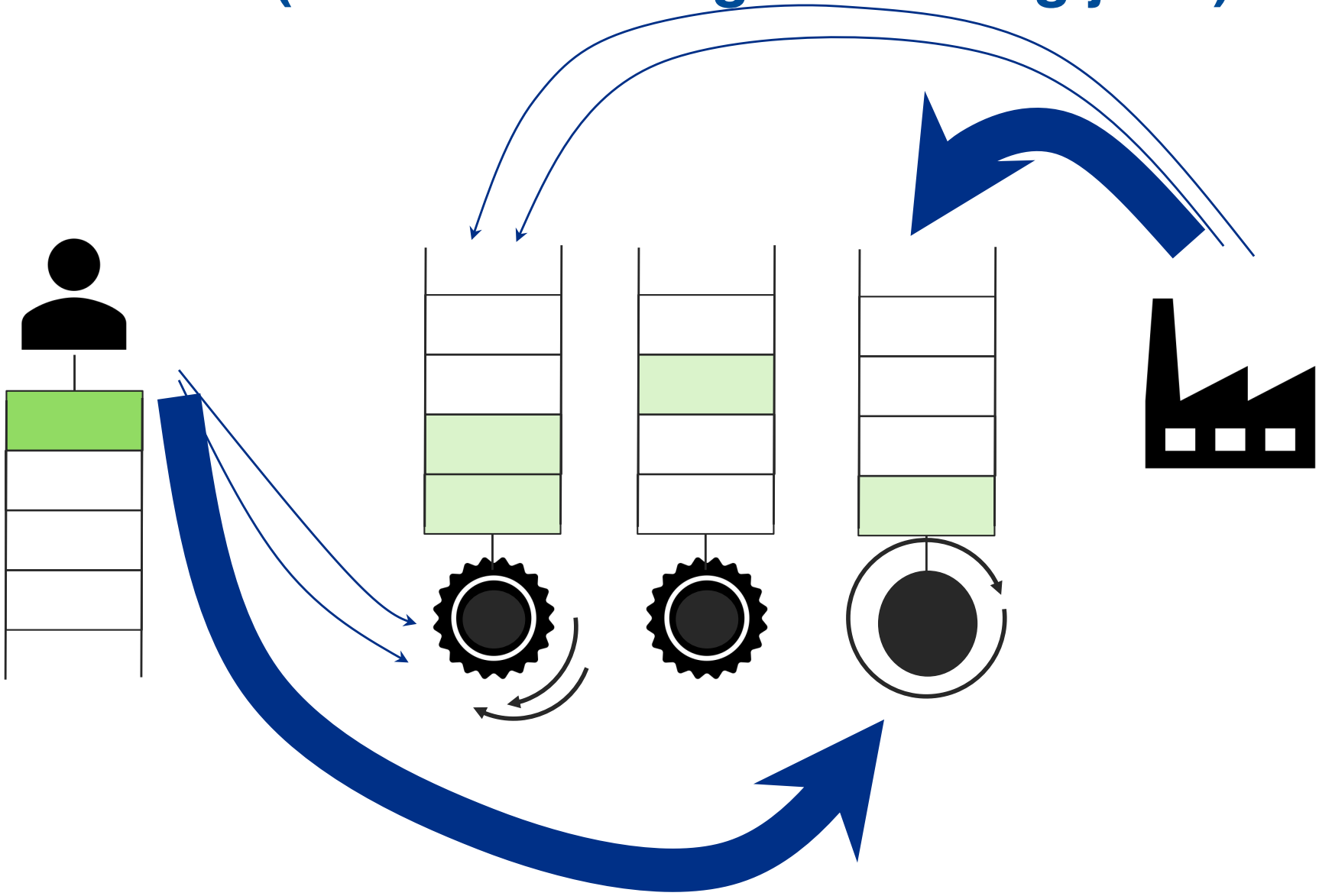
Pressure based system



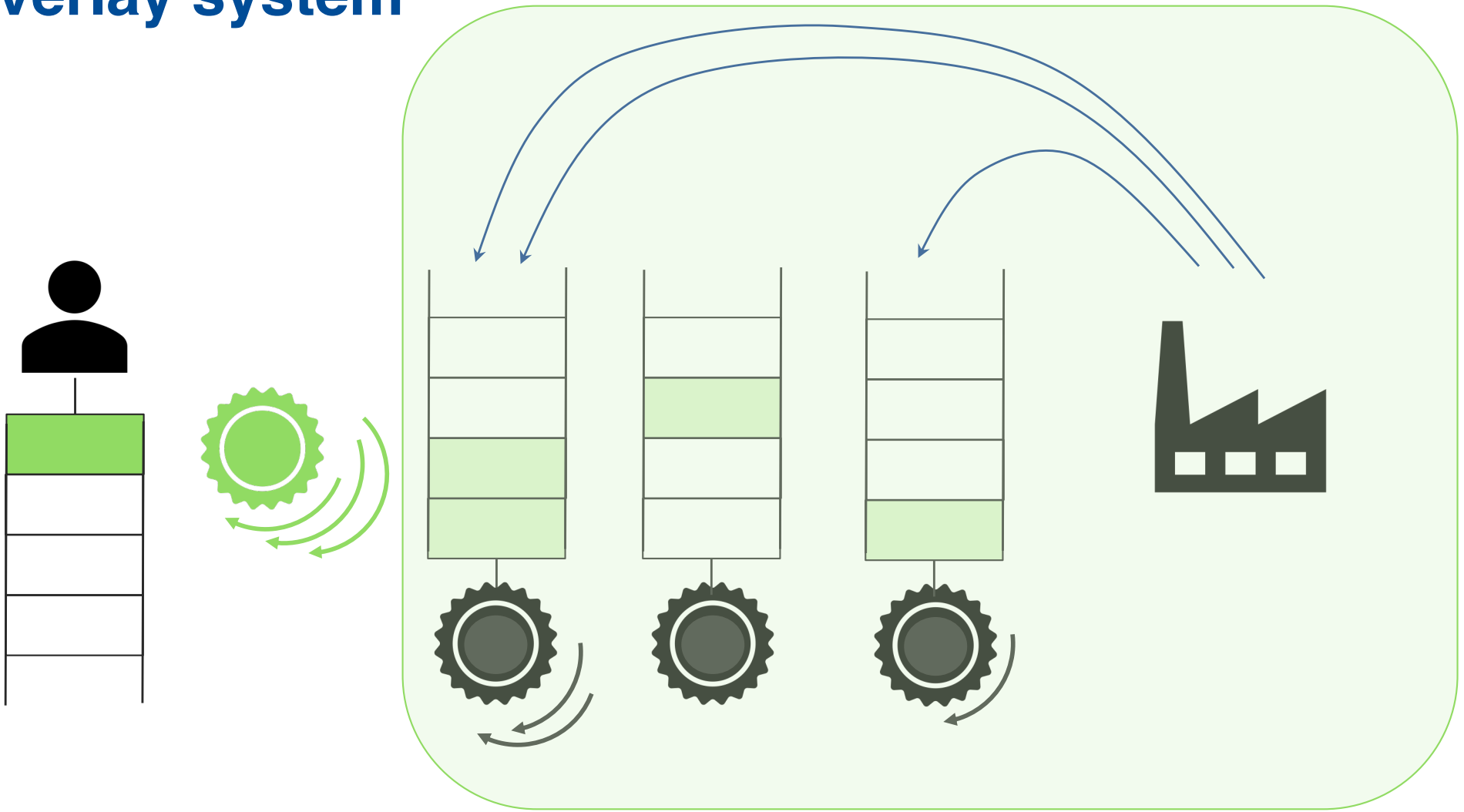
Constant pressure

Faster resources get more pilots (and jobs)

Black hole (node absorbing and failing jobs)



Overlay system

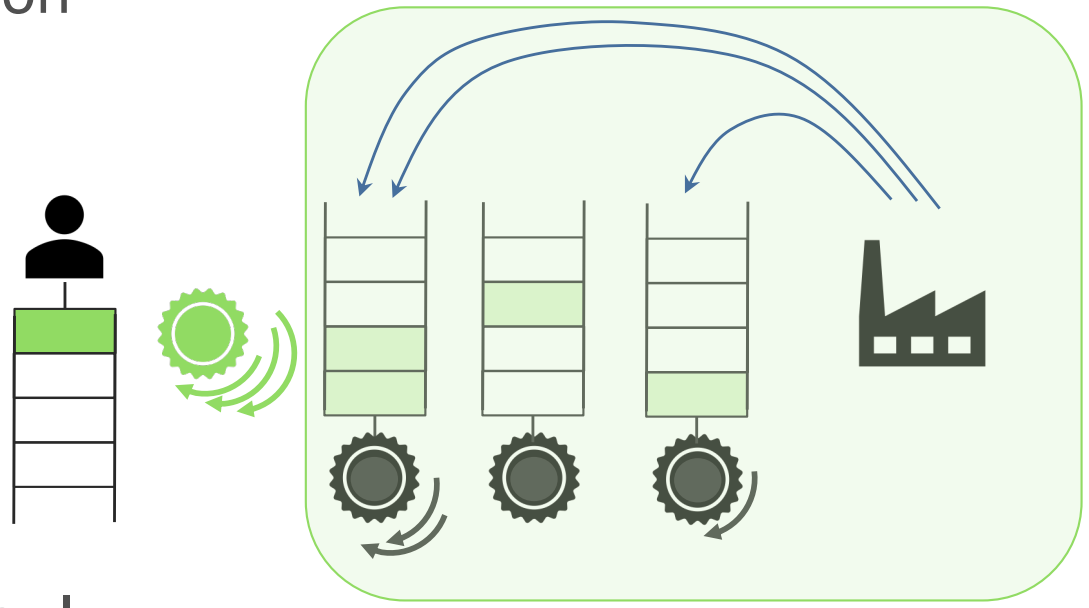


Pilots are expendable

Separation of resource and user problems

Overlay system

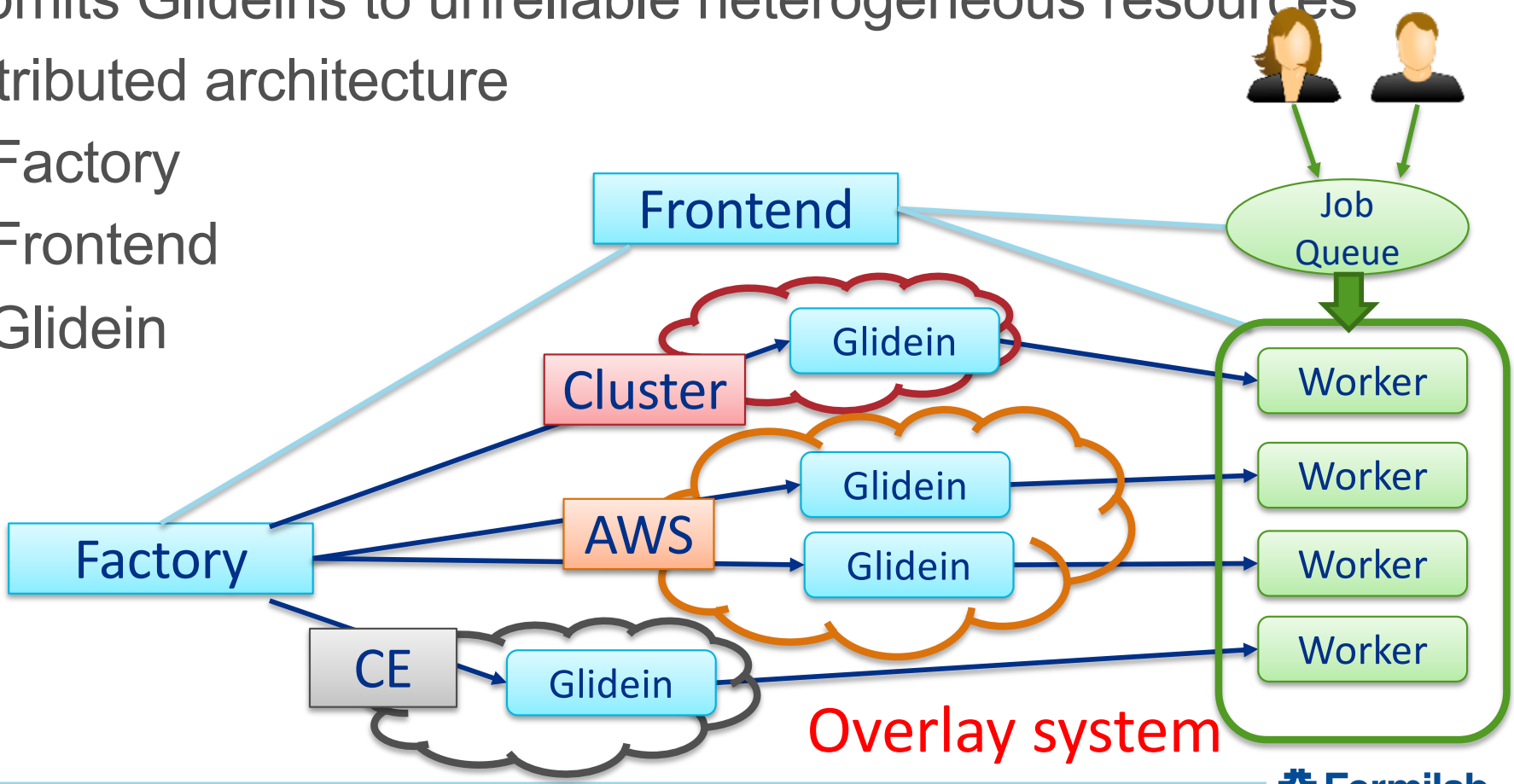
- Pilot layer
  - Distributed computing knowledge and troubleshooting
  - Reduce heterogeneity
  - Handle different speeds
  - Pressure-based submission
- Virtual cluster
  - Domain knowledge and troubleshooting
  - Elastic
- Separation for software, systems and people



GlideinWMS

GlideinWMS is a pilot based resource provisioning tool for distributed High Throughput Computing

- Provides reliable and uniform HTCondor virtual clusters
- Submits Glideins to unreliable heterogeneous resources
- Distributed architecture
  - Factory
  - Frontend
  - Glidein



Fermilab

Glidein: node testing and customization

- Scouts for resources and validates the Worker node
  - Cores, memory, disk, GPU, …
  - OS, software installed
  - CVMFS
  - VO specific tests
- Customizes the Worker node
  - Environment, GPU libraries, …
  - Starting containers (Singularity, …)
  - VO specific setup
- Provides a reliable and customized execute node to HTCondor

Factory

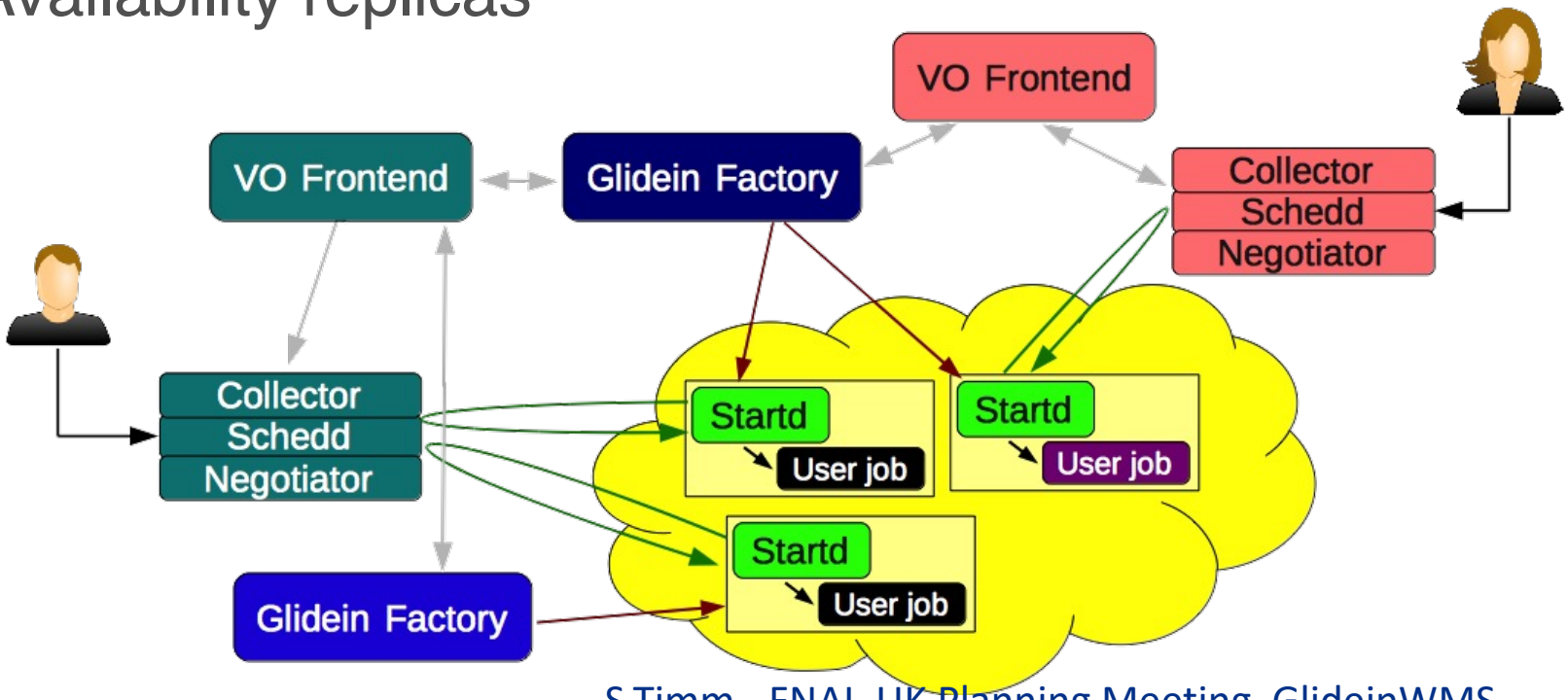
- A Glidein Factory knows how to submit to sites
  - Sites are described in a local configuration
  - Only trusted and tested sites are included
- Each site entry in the configuration contains
  - Contact info (hostname, resource type, queue name)
  - Site configuration (startup dir, OS type, …)
  - VOs authorized/supported
  - Other attributes (Site name, core count, max memory, ...)
  - Glideins can also auto-detect resources
- Configuration can be auto-generated (e.g. from CRIC), admin curated, stored in VCS (e.g. GitHub)
- Condor does the heavy lifting of submissions.

Frontend

- Monitors jobs to see how many Glideins are needed
- Compares what entries (sites) are available
- Requests Glideins from the Factory
- Requests Factory to kill Glideins if there are too many
- Pressure-based system
  - Works keeping a certain number of Glideins running or idle at the sites
  - Glideins requests are gradual to avoid spikes and overloads
- Manages credentials and delegates them to the Factory.

Distributed

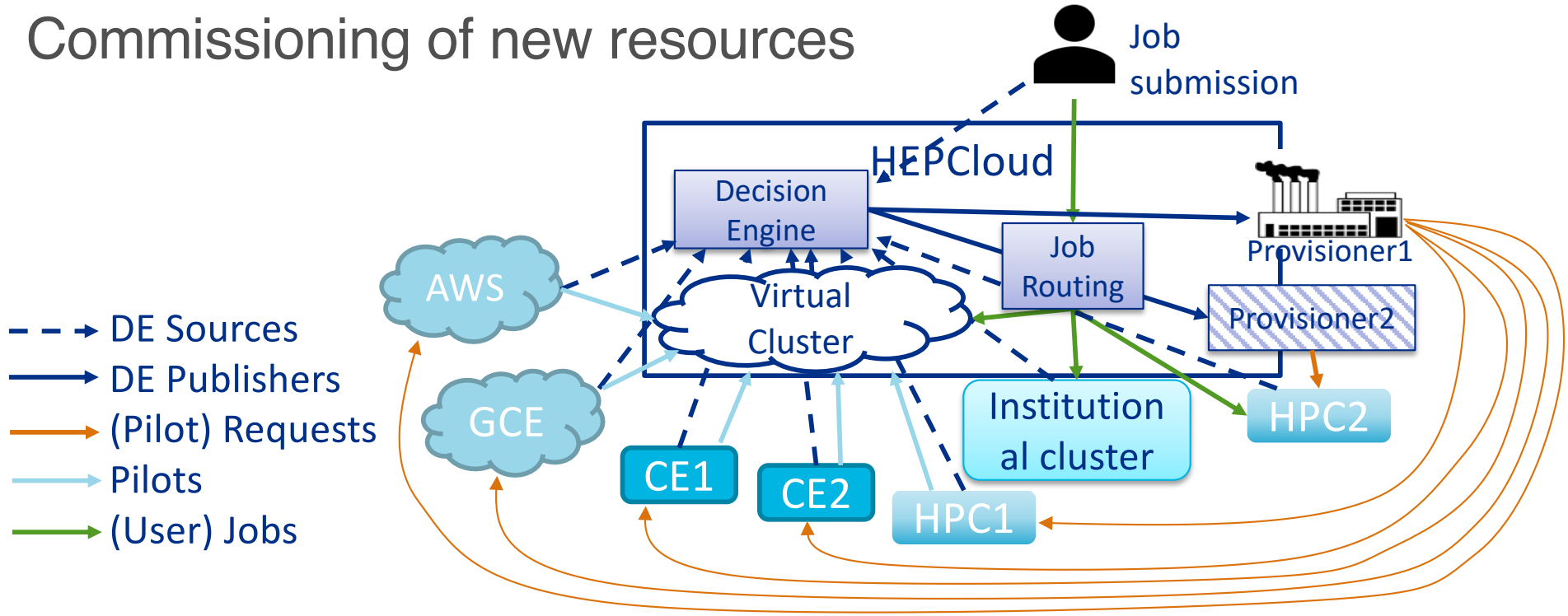
- N-to-M relationship
  - Each Frontend can talk to many Factories
  - Each Factory may serve many Frontends
- Multiple User Pools
- High Availability replicas



S.Timm - FNAL-UK Planning Meeting GlideinWMS

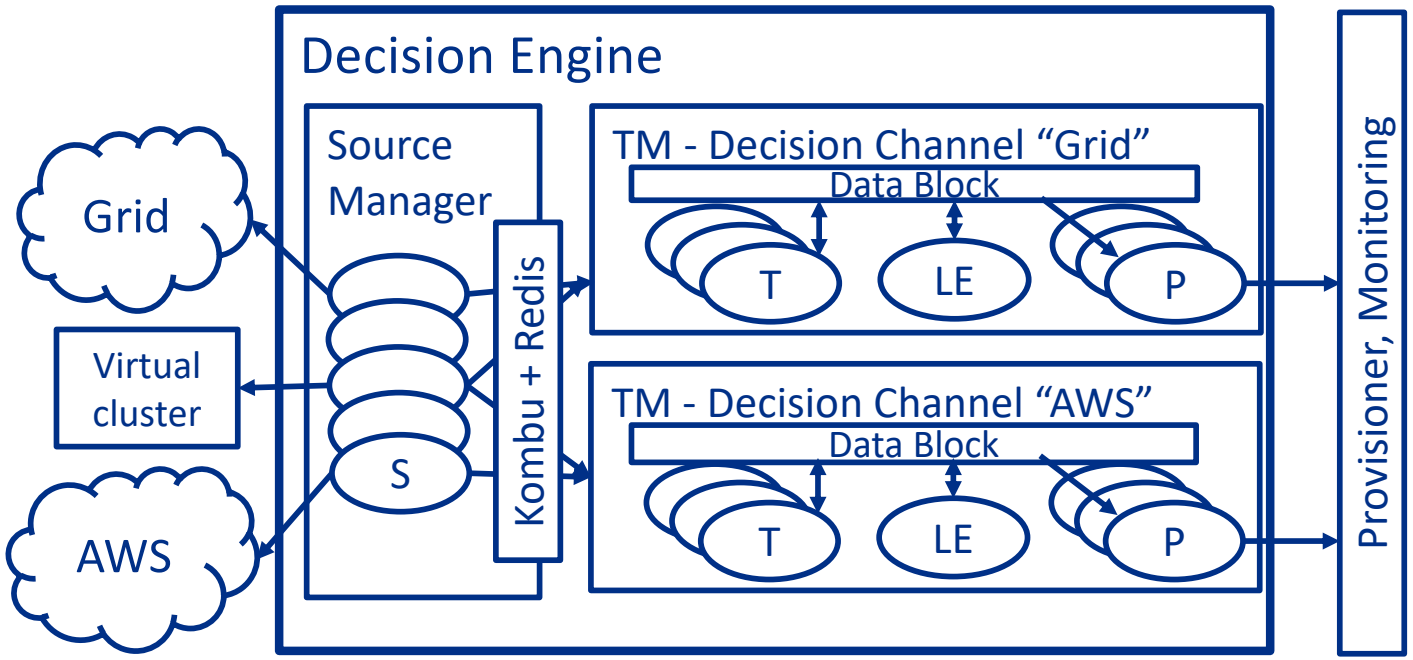
HEPCloud Facility

- Built on top of dHTC (GlideinWMS and HTCondor)
- Portal, job routing, resource provisioning
- Decision Engine
  - Business rules
  - Figure of Merit: multidimensional optimization
- Commissioning of new resources



Decision Engine architecture

- Sources (S) provide information to the channels via publish-subscribe message queues
- In each channel, the Task Manager (TM) coordinates Transforms (T), Logic Engine (LE) and Publishers (P) to take provisioning decisions and publish status information



GlideinWMS and HEPCloud References

- GlideinWMS

  https://github.com/glideinWMS/glideinwms
  https://glideinwms.fnal.gov/doc.prd/index.html
  https://cdcvs.fnal.gov/redmine/projects/glideinwms
  https://zenodo.org/record/4242727

- Decision Engine

  https://github.com/HEPCloud/decisionengine
  https://github.com/HEPCloud/decisionengine_modules
  https://hepcloud.github.io/decisionengine/
  https://hepcloud.github.io/decisionengine_modules/
  https://hepcloud.fnal.gov/

Storage types summary

- System Volumes
  - Read only
- Locally Mounter Volumes (Local or RAM disk)
  - CWD (Current Work Directory)
  - TMP
- Interactive Storage Volumes (NAS - NFS, GPFS, Luster, …)
  - Shared file systems
  - Shared home directories
- Remotly-accessible storage volumes
  - Distributed file system (HDFS, dCache, Xrootd, S3, ceph)
  - Storage Element
- CernVM FS (CVMFS)
  - Write once read everywhere HTTP based distributed FS

Authentication and authorization

- Authentication: verifies the identity of a user or service before granting them access.
- Authorization: determines what they can do once they have access/permission, verified identity to control access.

- Identity and Access Management (IAM)

- Distributed AA
  - Grid certificates and Virtual Organizations (VO) trust model
  - Authentication tokens

AA in distributed systems:
https://deimos.io/post/authentication-and-authorization-in-a-distributed-system

Credential types

- X509 Certificate and Proxy
  - VOMS Extension
  - Identity based (you and your affiliations)
- JASON Web Token
  - SciToken
  - IDTOKEN
  - WLCG (IAM) token
  - Bearer token (capability based)

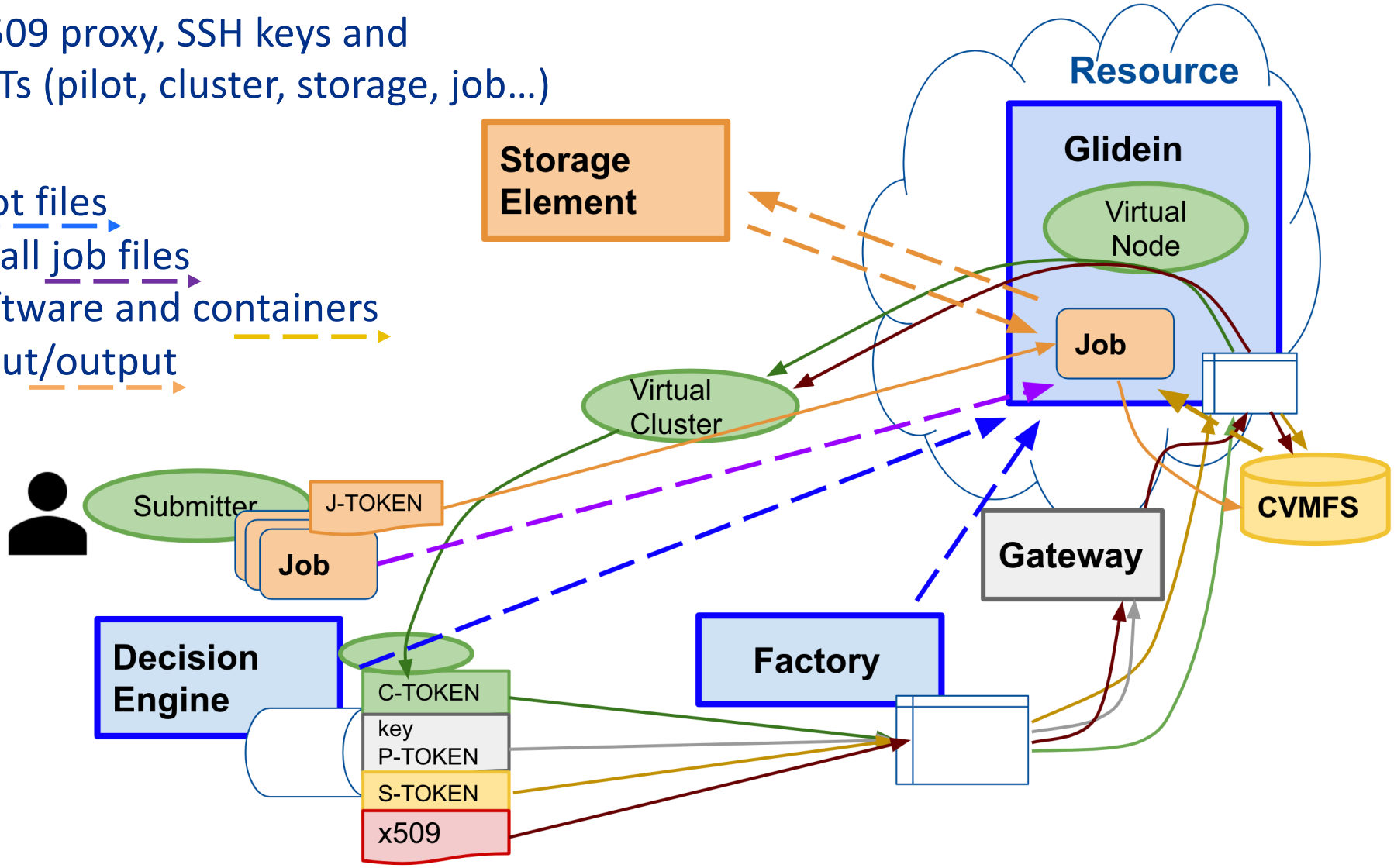
Credentials and data movement in a Glidein

Credentials

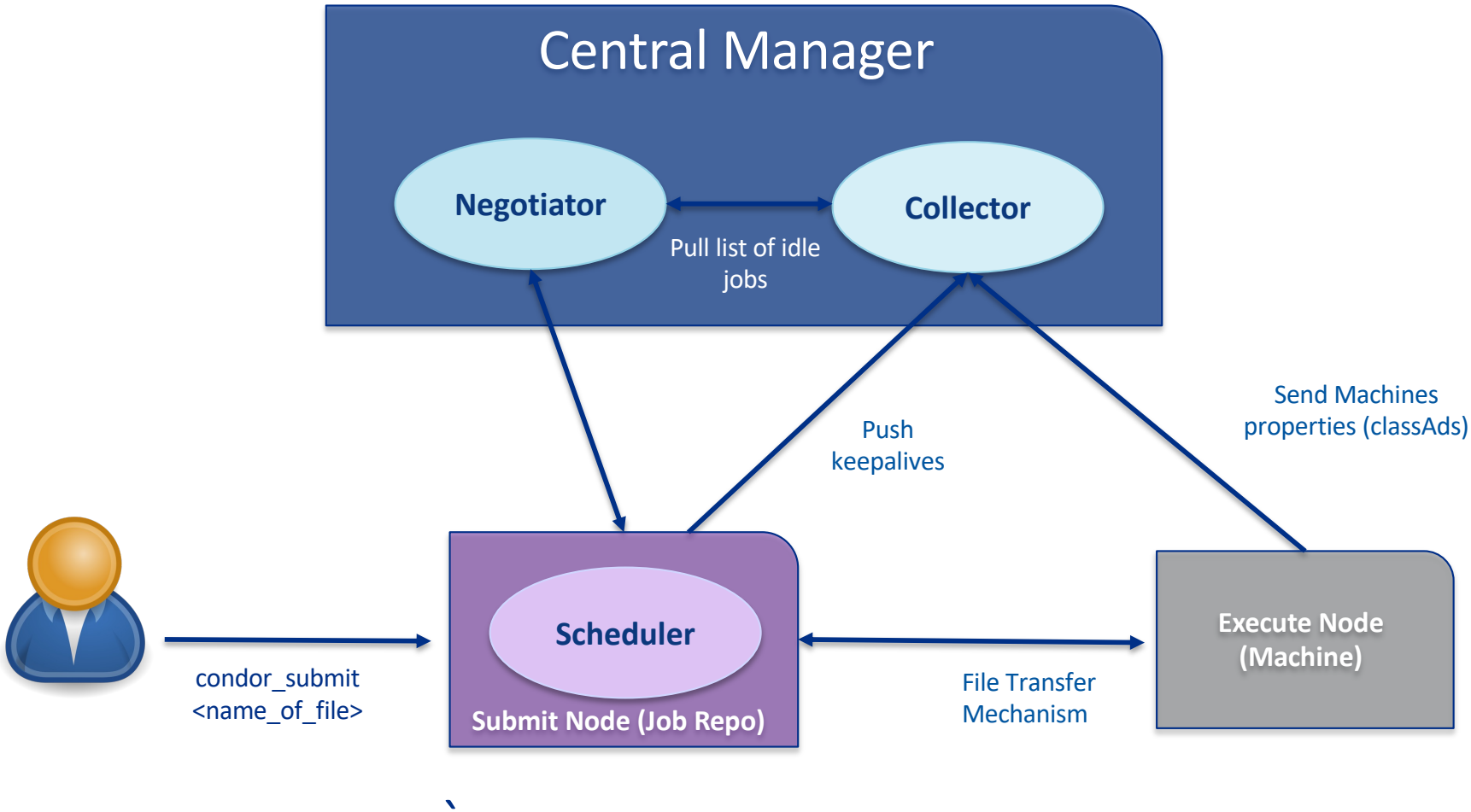
- x509 proxy, SSH keys and JWTs (pilot, cluster, storage, job…)

Data

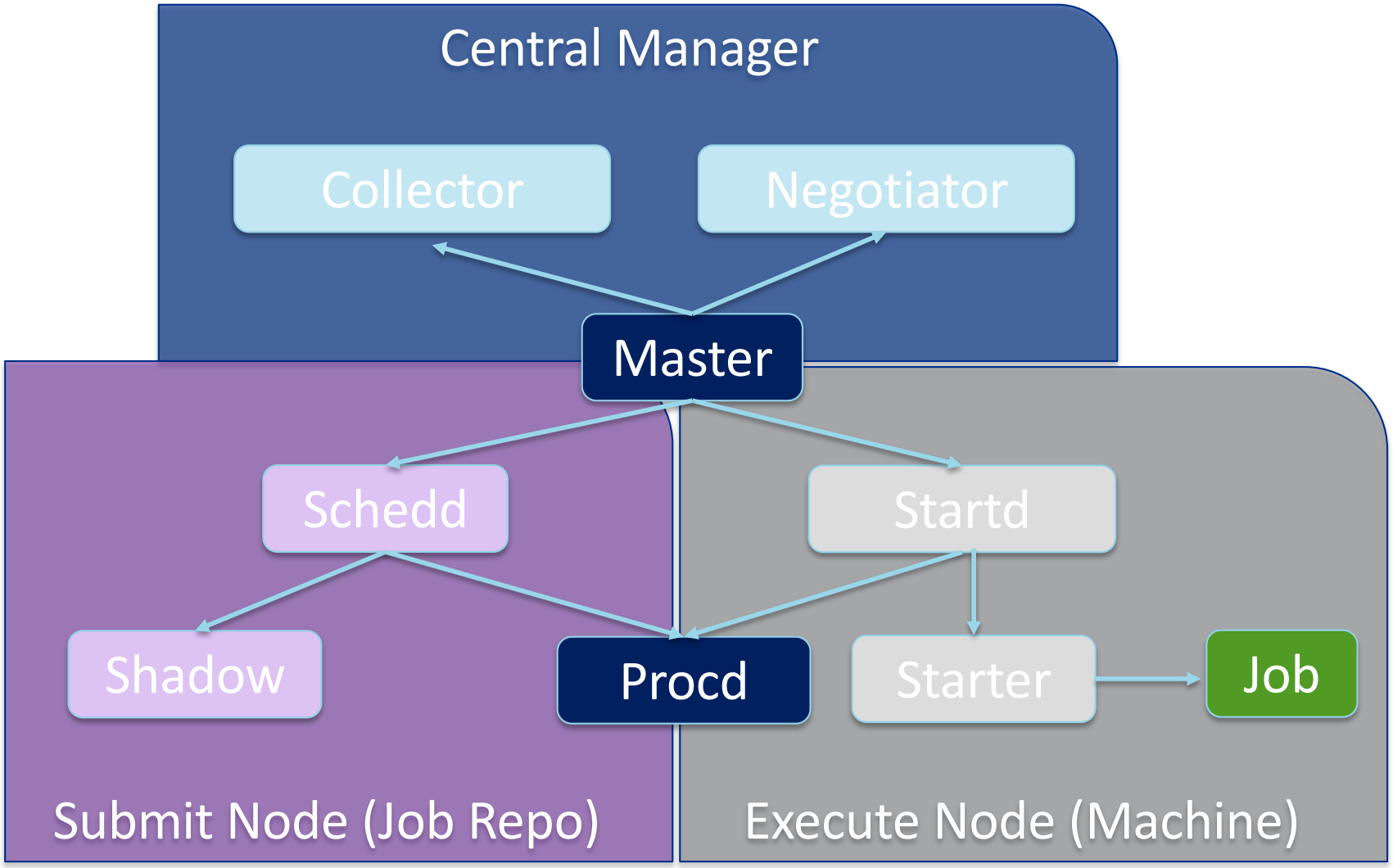
- Pilot files
- Small job files
- Software and containers
- Input/output



HTCondor components

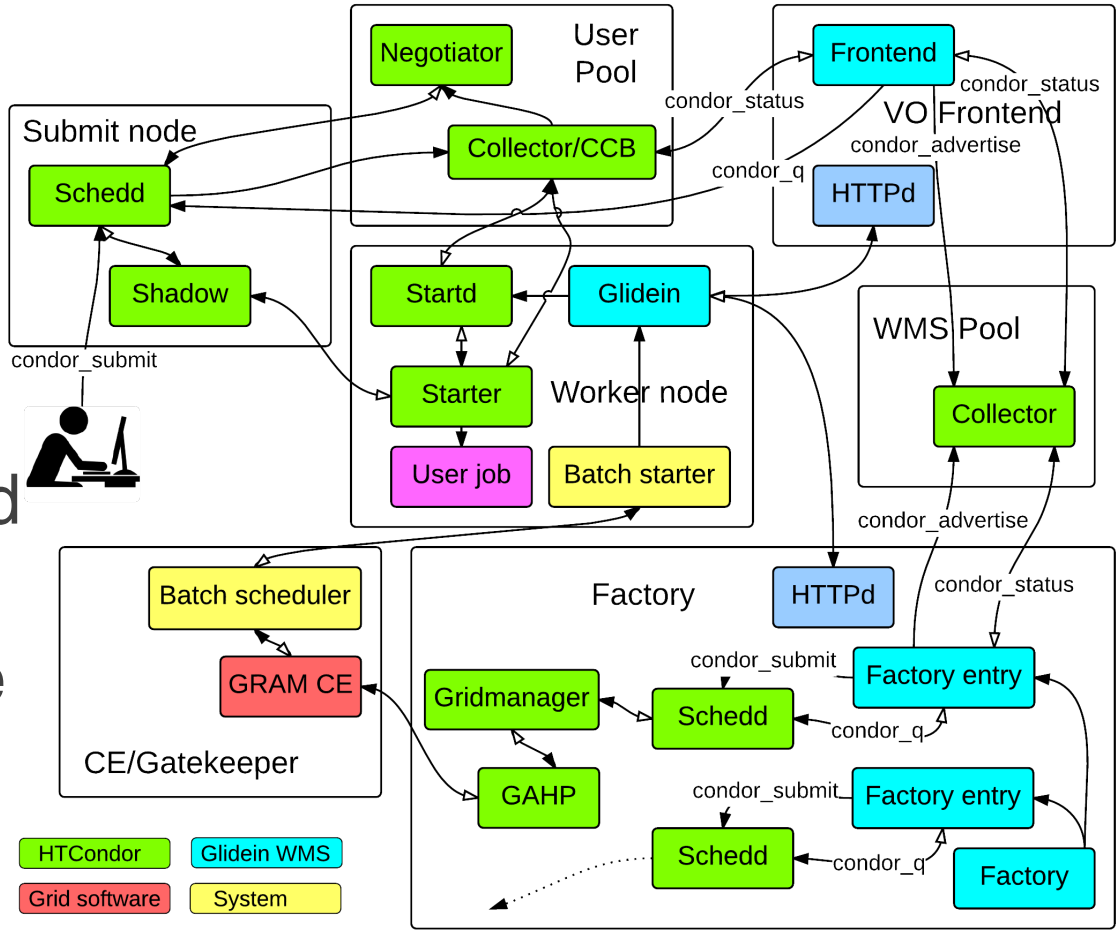


HTCondor components (daemons)



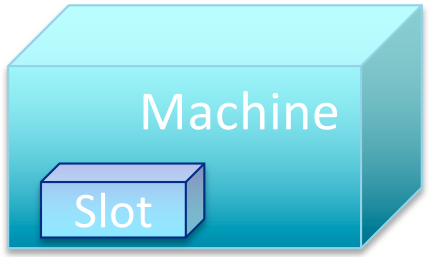
HTCondor building blocks in Glidein WMS

- The Factory works with an HTCondor pool, WMS pool, to submit Glideins to different resources
- The HTCondor Glideins are pilots that launch a startd that registers on a second HTCondor pool, User pool
- User jobs are matched and execute on the resources
- The Frontend monitors the user schedds and notifies the Factory about the need for more Glideins

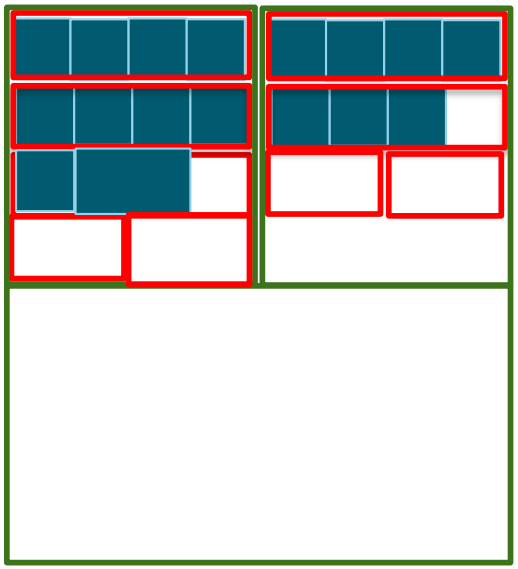


Glideins run on Machines

- Machines (worker node, host, node, resource) are managed by a (Local) Resource Manager
- More frequently virtual than not
- Characterized by its resources (dimensions):
  - CPUs (or total number of cores)
  - RAM (memory)
  - Disk
  - Lifetime (length of the lease)
- There can be other special resources that the node provides: GPUs, QPU, access to devices, software, …
- The Glidein will receive all the node or part of it (slot)
- Sometime is not easy to identify everything used by a job



Partitioning in an overlay system



- Along multiple dimensions
  - Cores, Memory, Disk, Lifetime
- At multiple levels
  - An Execute node is a VM (partial hw)
  - The resource manager partitions its Execute nodes
  - GlideinWMS further partitions the resources it receives
- E.g. 64 Cores machine split in 16 or 32 cores cluster slots; 16 or 12 cores Glideins in 4 or 2 cores partitionable slots; 2 or 1 core jobs
- Aspects to consider
  - Fragmentation (unused small spaces)
  - Flexibility (vs Complexity)
  - Under or over provisioning (overbooking or be prudent)
  - Scaling (big slots, fewer slices)

Job and Machine ‘dimensions’

- Job request
  - request_cpus: number of cores, integer, default 1.
  - request_disk: amount of disk space in Kbytes, default to sum of sizes of the job's executable and all input files (or image size)
  - request_memory: amount of memory space in Mbytes, default to executable size
- Machine
  - Cpus: number of cores, integer, by default the available cores
  - Disk: amount of disk space on this machine available for the job in KiB, by default the available space
  - Memory: amount of RAM in MiB in this slot
- Over and Under provision are possible

Summary

- Experiments' Jobs can run on many different resource types
  - Many have specific advantages/limitations
- Think about systems at different levels
- Start from the overall problem
- Look at all details, angles, and tradeoffs