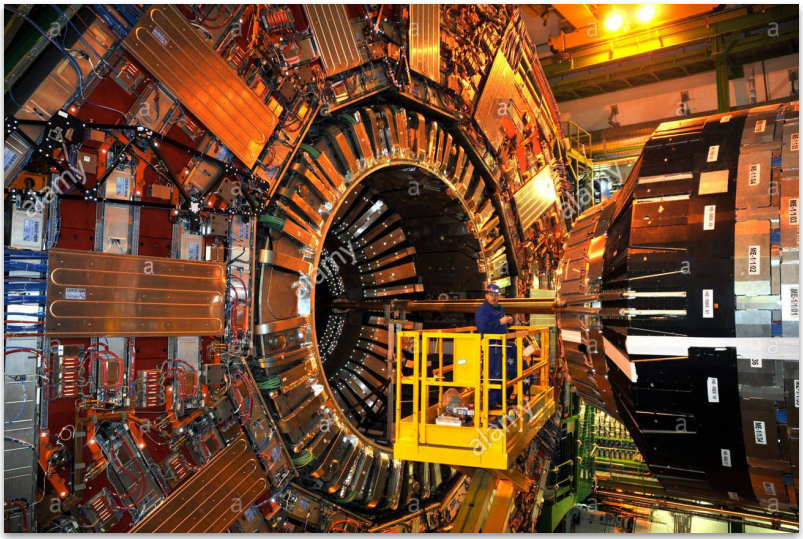
Operating a Federated HTCondor Infrastructure: Monitoring and Management for CMS Computing

Bruno Coimbra, M. Mascheroni, A. Pérez-Calero Yzquierdo,
F. Von Cube, V. Zokaite, H. Kim
for the CMS Collaboration

HTC25 Throughput Computing 2025

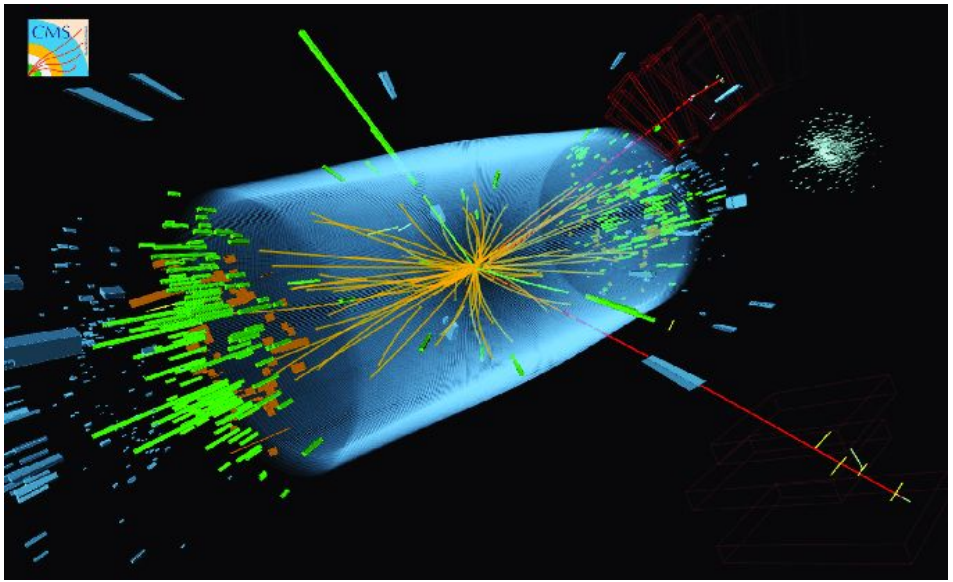
Fermilab CERN UC San Diego PIC port d'informació científica

The CMS experiment at CERN



- Experimental data is stored, distributed, reconstructed, and analyzed, comparing to simulated data (Monte-Carlo)
  - **Hundreds of PBs per year**

- **High Energy Physics general-purpose experiment** recording proton-proton collisions **at the LHC** at CERN

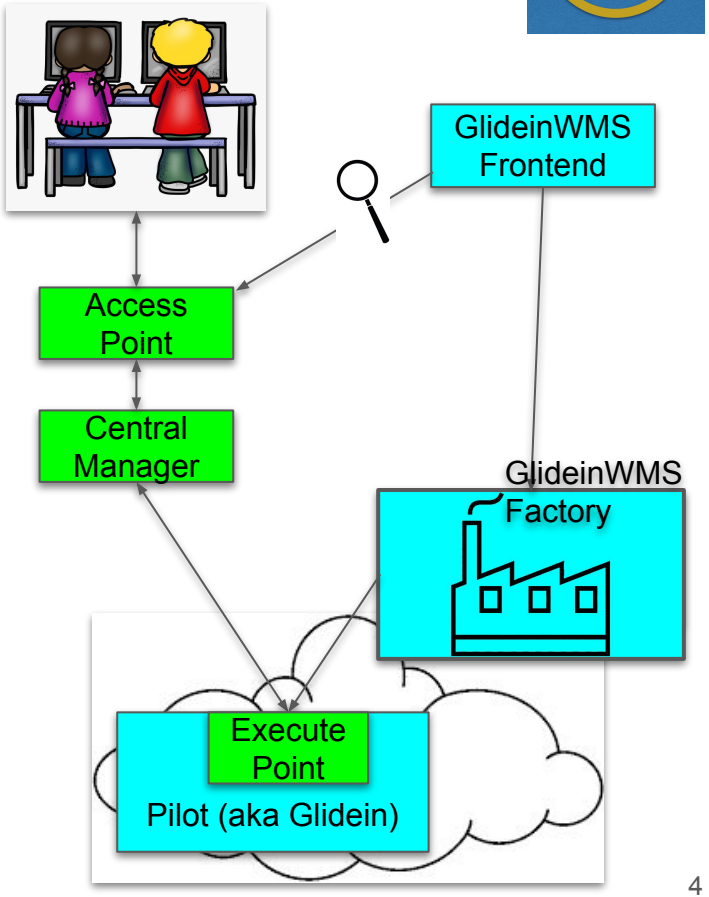


The computing landscape - the WLCG

- Data traditionally analyzed using **Worldwide LHC Computing Grid (WLCG) resources**
  - Global collaboration of around 170 computing centers
  - Access based on dedicated resources (**pledges)**
  - **Over 1M CPU cores and 2 EB in data storage**

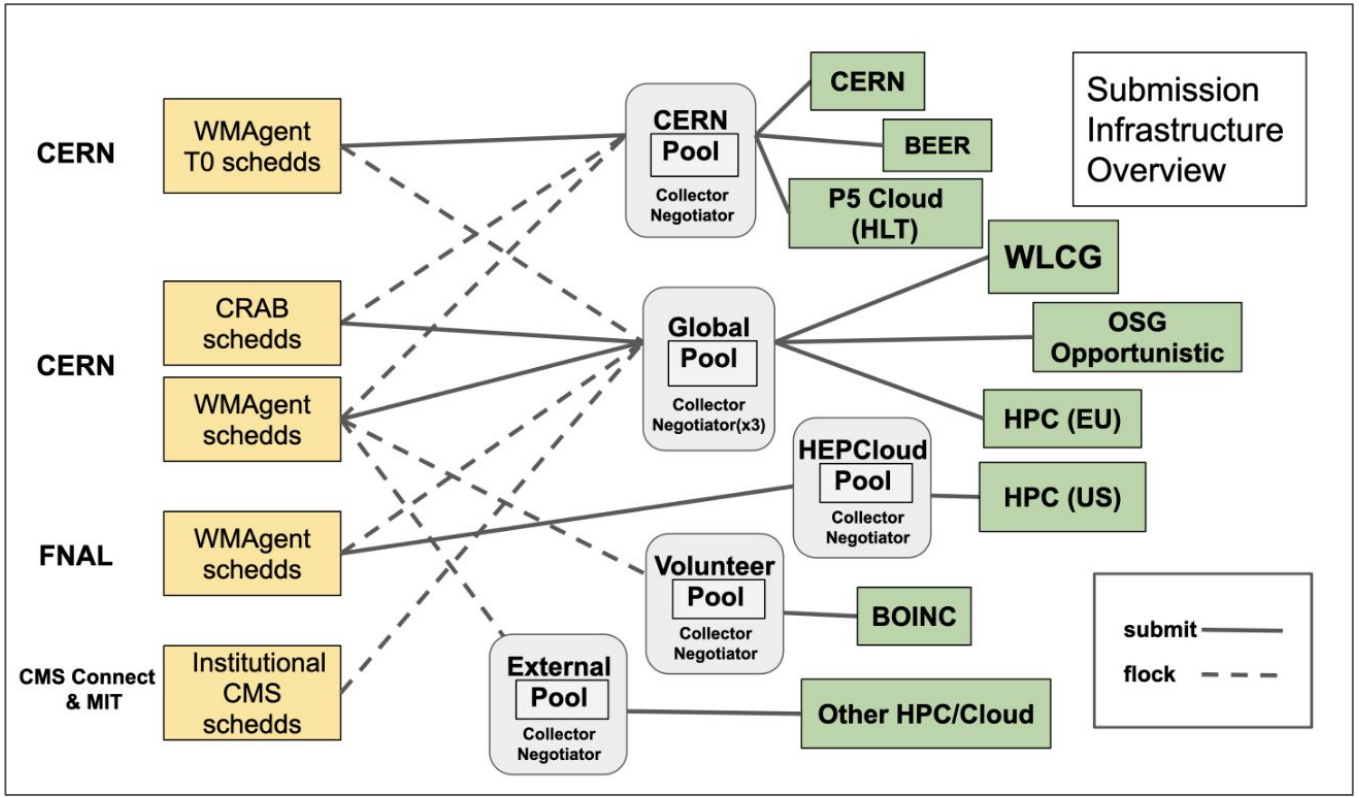
The CMS Submission Infrastructure Group

- Part of CMS Offline and Computing in **charge** of:
  - Organizing **HTCSS** and **GlideinWMS** **operations** in CMS, in particular of the **Global Pool**, an infrastructure where reconstruction, simulation, and analysis of physics data takes place
  - Communicate CMS **priorities to the development teams** of **GlideinWMS** and **Condor**
- In practice:
  - We operate a set of federated pool of resources **distributed over 70 Grid sites, plus non-Grid resources**
  - Join them into a Global Pool of resources managed by HTCondor



The CMS SI: federated HTCondor pools

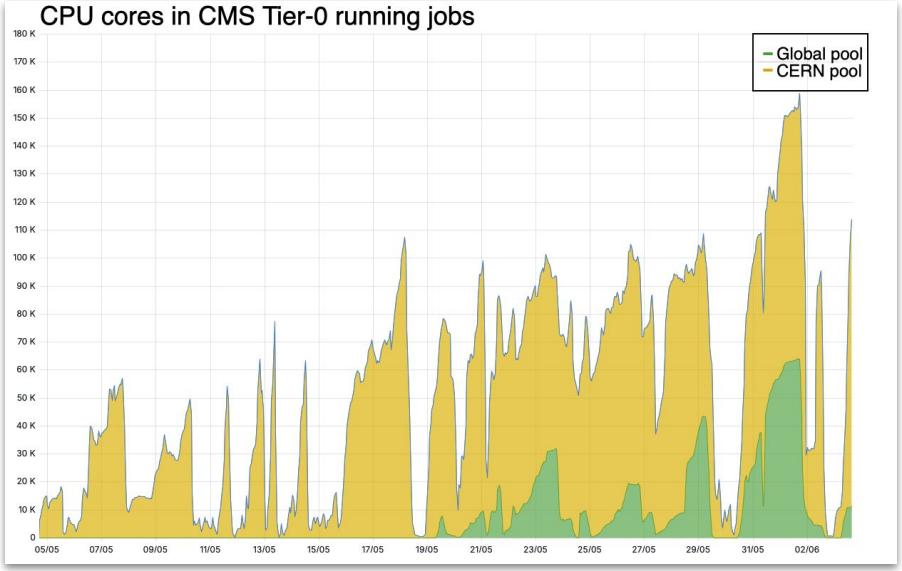
- Types of access point
- Types of execution point



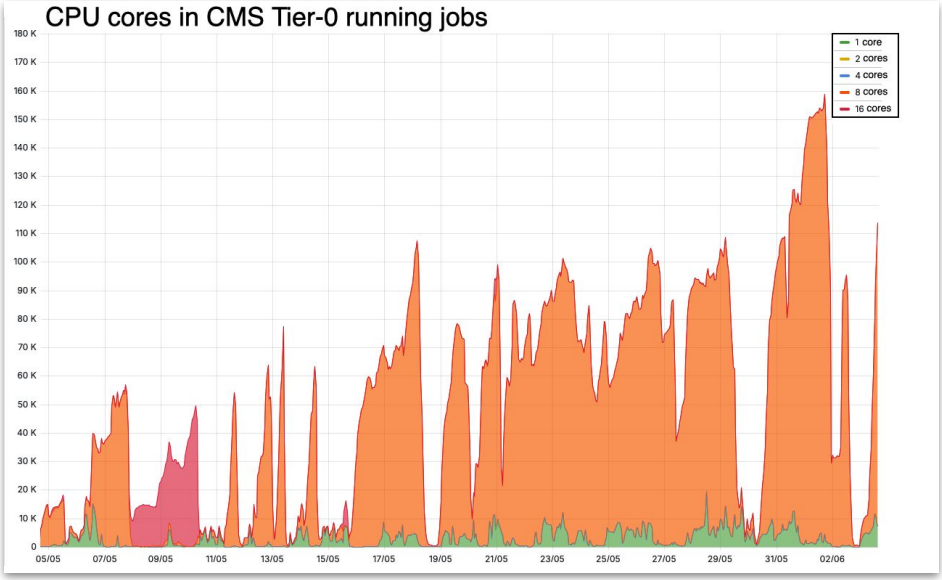
HTCondor in support of CMS data taking

As an example of the many capabilities our HTCondor setup provides CMS with, consider our submission infrastructure support in the mission-critical task of low-latency CMS detector data processing (Tier-0 tasks)

Flocking from CERN to Global pool resources for expanded capacity

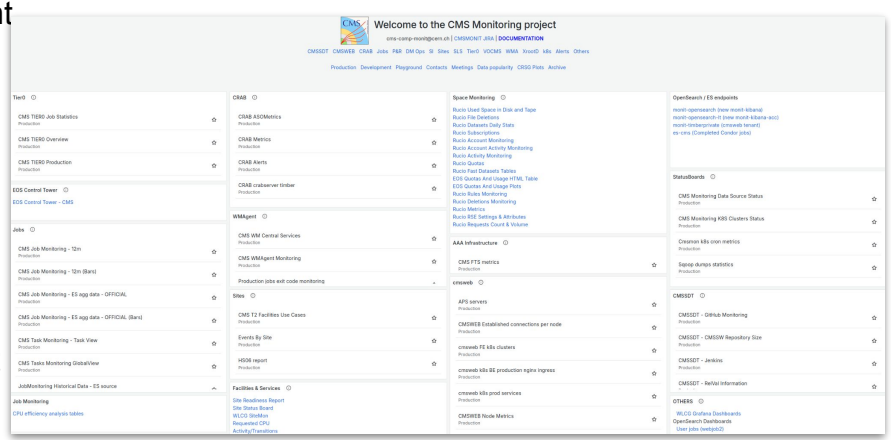


Flexible dynamic partitioning of resources for 8 and 16-core Tier-0 jobs



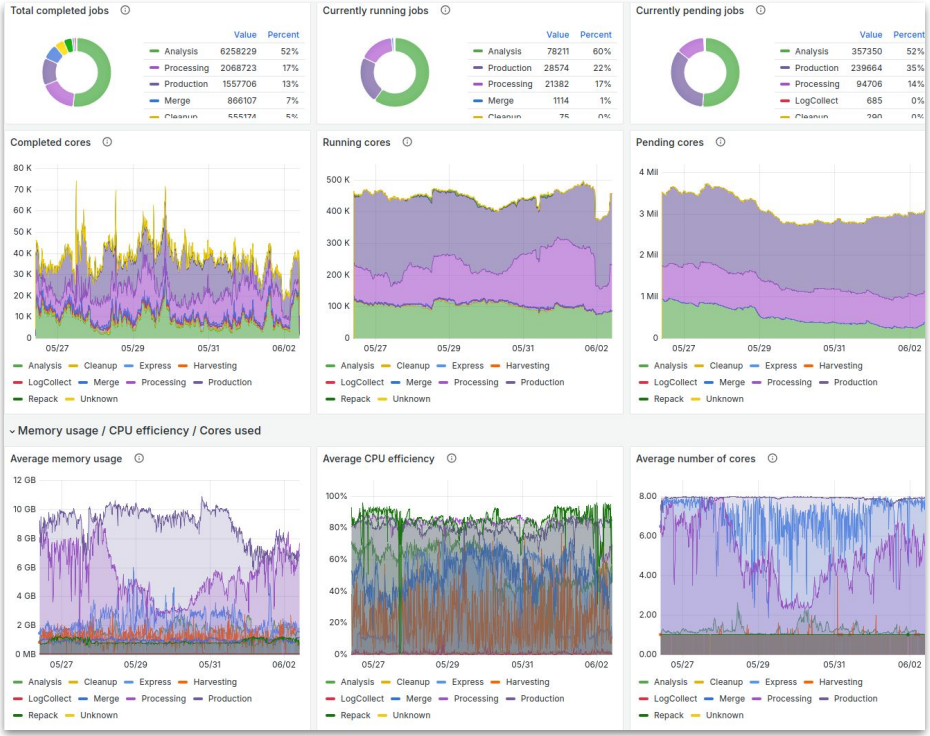
Why Monitoring Matters

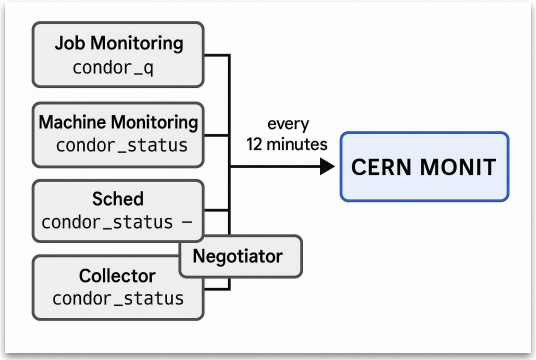
- Scale & Complexity
  - E.g.: Managing a Global Pool spanning 70+ sites and hundreds of thousands of cores requires continuous insight into system behavior
- Reliability & Performance
  - Monitoring enables early detection of issues, resource bottlenecks, and inefficiencies in job execution and resource utilization
- Operational Visibility
  - Critical for troubleshooting, planning, and maintaining trust in automated scheduling and resource provisioning
- Based on CERN MONIT Infrastructure
  - Unified monitoring framework across CMS computing
  - Metrics sent every 12 minutes from all components



Monitoring of the CMS Pools (I)

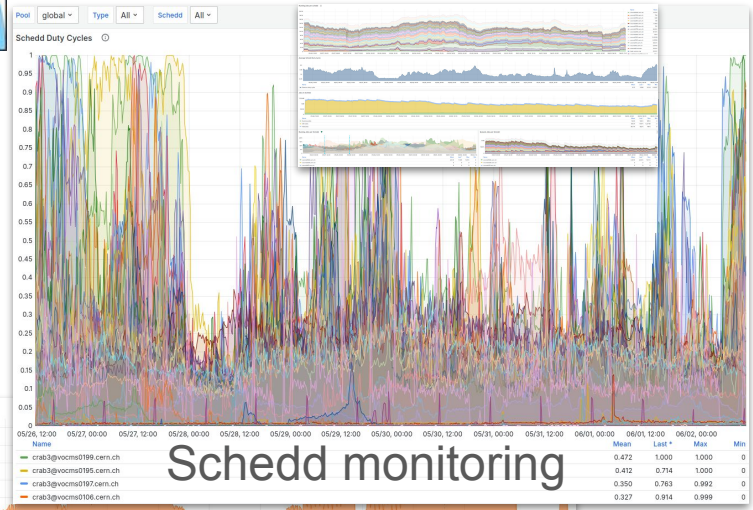
- HTCondor jobs and daemons
  - **Job Monitoring** –
    from condor_q: state, runtime, efficiency
  - **Machine Monitoring** –
    from condor_status: slot availability, partitionable resources, site status
  - **Scheduler & Negotiator & Collector**–
    from condor_status -sched (-negotiator) (-coll): match-making and negotiation cycles, status of demons, duty cycle



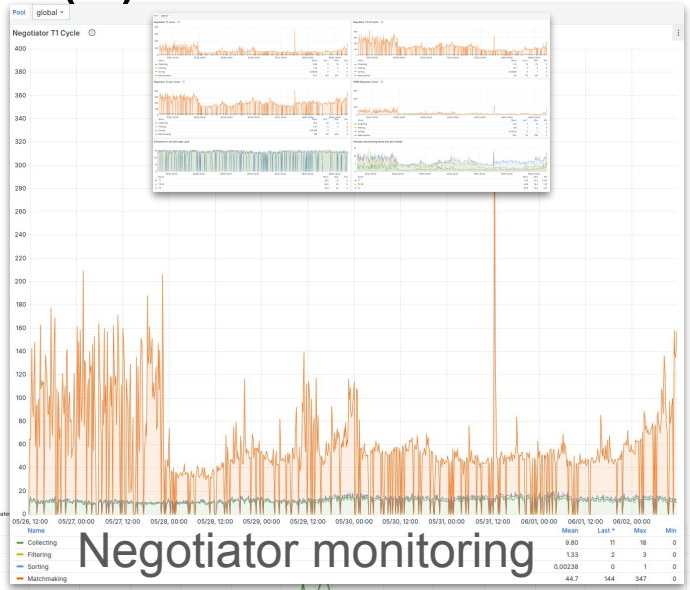


Job Monitoring …

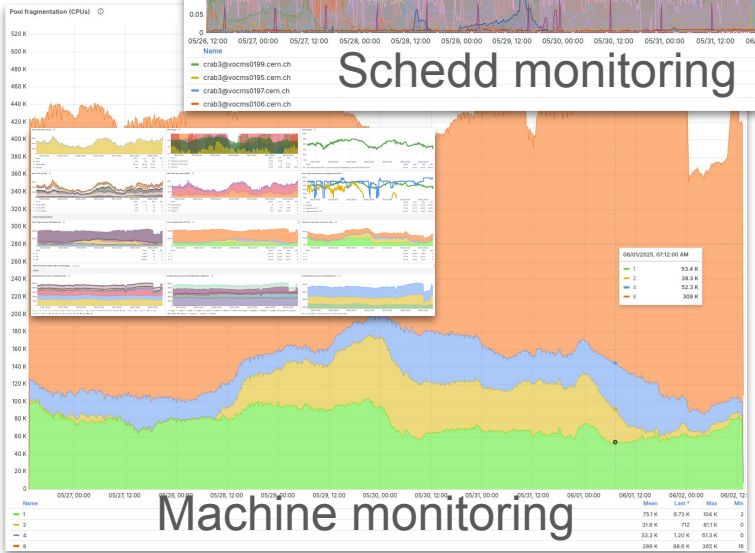
Monitoring of the CMS Pools (II)



Schedd monitoring



Negotiator monitoring



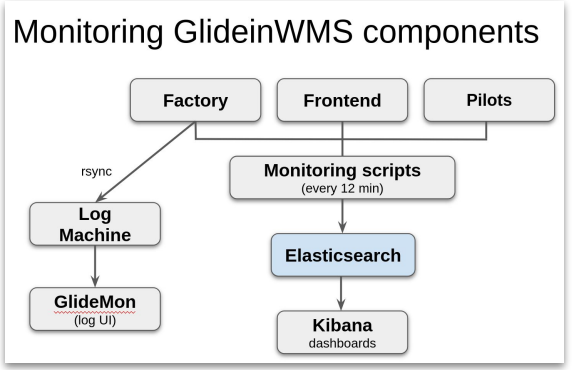
Machine monitoring

Collector monitoring

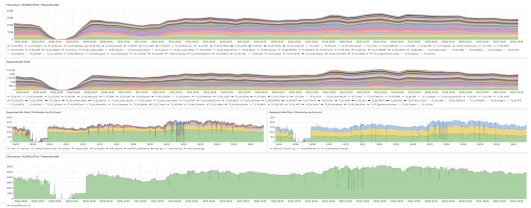
Monitoring glideinWMS Components

Purpose
To complement HTCondor monitoring by capturing the full lifecycle of pilot-based workload submission, from factory to site.

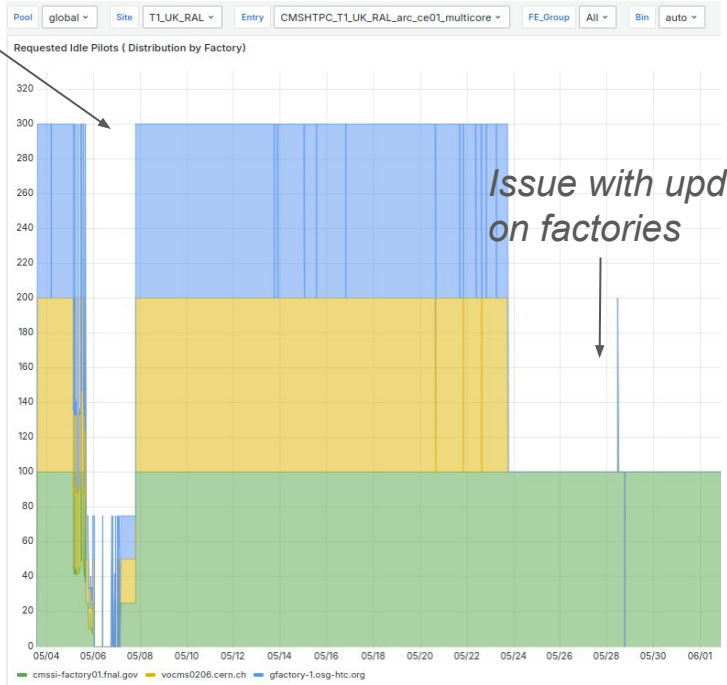
- **Data Collection**
  - Custom scripts extract metrics from glideinWMS components every **12 minutes**, and publish to **Elasticsearch**.
- **Covered Components**
  - **Frontend** – submission requests, resource pressure, and constraints
  - **Factory** – glidein submission success/failure, per-entry status, site responsiveness
  - **Pilots** – status, lifespan, errors, and site-level anomalies
- **Pilot Log Aggregation**
  - A dedicated **log machine** collects logs from all factories using **rsync**
  - Logs are indexed and exposed via the **GlideMon** web tool for inspection and troubleshooting



Frontend monitor



Lack of pressure



Issue with update on factories

- Track pilot requests sent from the frontend to each factory over time.
- Operational insights
  - Helps identify **loss of site pressure**, e.g., when no jobs are queued for a given site
    - E.g.: The **site drained** due to lack of demand (idle jobs not matching site requirements)
  - Useful for catching **factory-side failures**, such as misconfigurations or problems after updates
    - Also, wrong description of workload resource requirements

Factories monitor

Purpose
Monitor pilot startup and validation issues across sites to ensure reliable job execution.

Key Error Categories

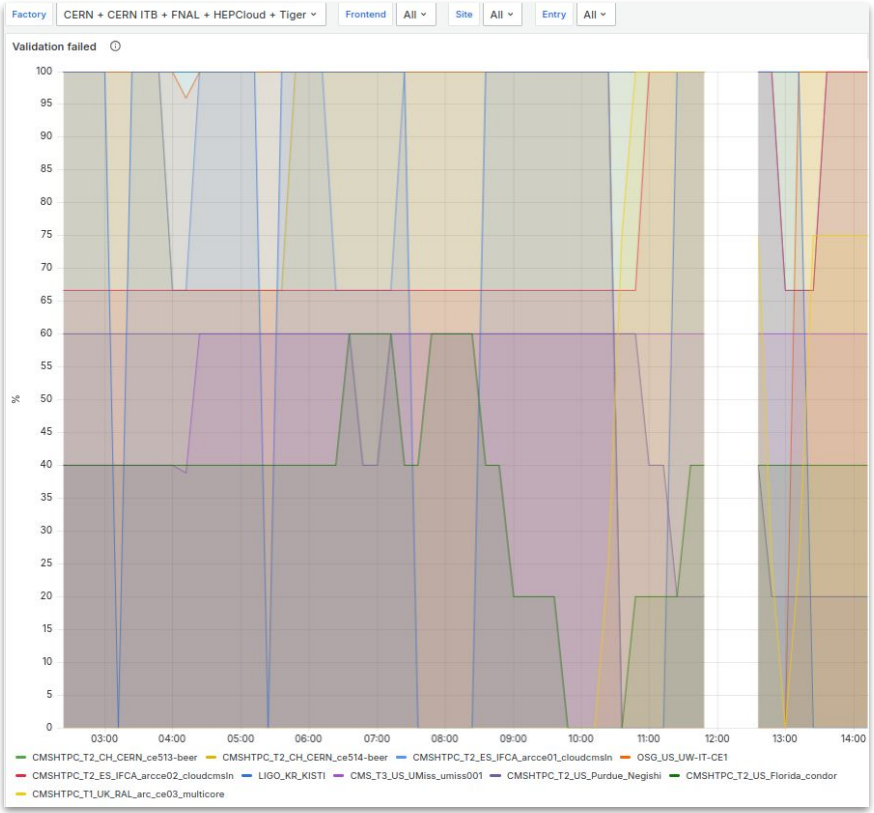
- **Startup Failures**: Pilots fail to launch on the CE or batch system (e.g., `condor_startd` not initializing)
- **Validation Failures**: Pilot launches, but fails the validation script (e.g., missing dependencies, misconfiguration)
- **Idle Pilots**: Pilot runs and connects to the collector, but no jobs are matched or start

Operational Use

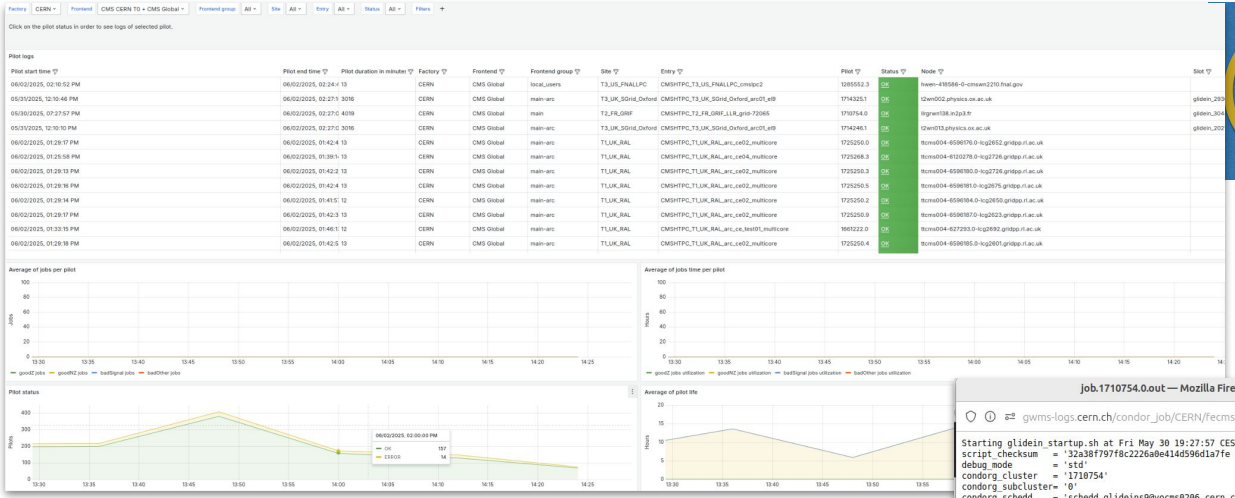
- This plot highlights **problematic sites** with recurring validation issues
- Enables prompt action: **open GGUS tickets** or contact site admins directly
- Essential for **keeping the pool healthy** and minimizing wasted resources

Example
A persistent spike in validation failures at a specific site typically indicates:

- Misconfigured worker nodes
- Broken CVMFS, missing CMS environment
- Changes in site middleware or OS upgrades not compatible with pilot setup



Pilot logs



Why Pilot Logs Matter

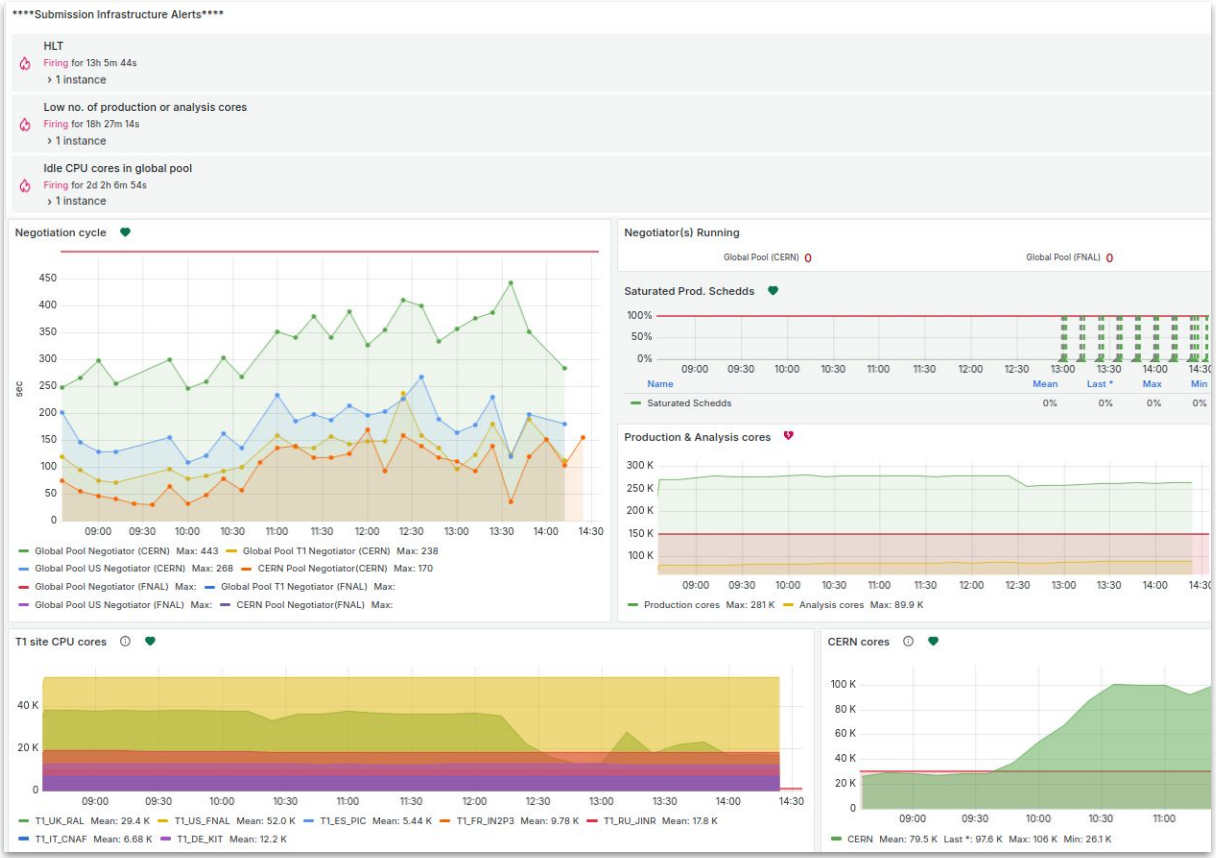
- Complement monitoring metrics with **low-level detail** from actual pilot execution
- Critical for diagnosing **site-specific issues** that are not visible through high-level metrics
- Helps understand **pilot lifecycle failures**, environment problems, misbehaving nodes

Architecture Overview

- Each factory **collects logs** from running pilots
- Logs are transferred via **rsync** to a **central log machine**
- Logs are **indexed and served** via the **GlideMon** interface

Proactive Alerting

- Enables **rapid response** to failures or anomalies
- Reduces downtime by catching issues **before they impact users**
- Complements dashboards and log analysis with **real-time notifications**
- Built on top of **CERN MONIT + Elastic/Kibana** stack



Conclusions

- **CMS relies on a large-scale, federated HTCondor infrastructure** to execute data processing, simulation, and analysis across 70+ grid sites.
- **Monitoring is essential** to ensure reliability, efficiency, and scalability at all layers — from HTCondor daemons to glideinWMS components and pilot logs.
- We have built a **comprehensive, modular monitoring ecosystem** leveraging **CERN MONIT**, custom scripts, log aggregation, and real-time alerts.
- This system enables **proactive operations**, fast diagnosis, and effective collaboration with site admins to resolve issues and maintain a healthy global pool.

Acknowledgements

- **Congratulations from the CMS experiment on the 40 Years of the Condor Project Commemoration!**
- **… And our deepest gratitude for the many years of successful collaboration!**
- **Looking forward to many more years of the HTCondor technology suite in support of CMS scientific programme!**

Backup slides

Evolution of total CPU capacity managed by CMS with HTCondor over the years