Joint Institute for Nuclear Research

Resource Management in Private Multi-Service Cloud Environments

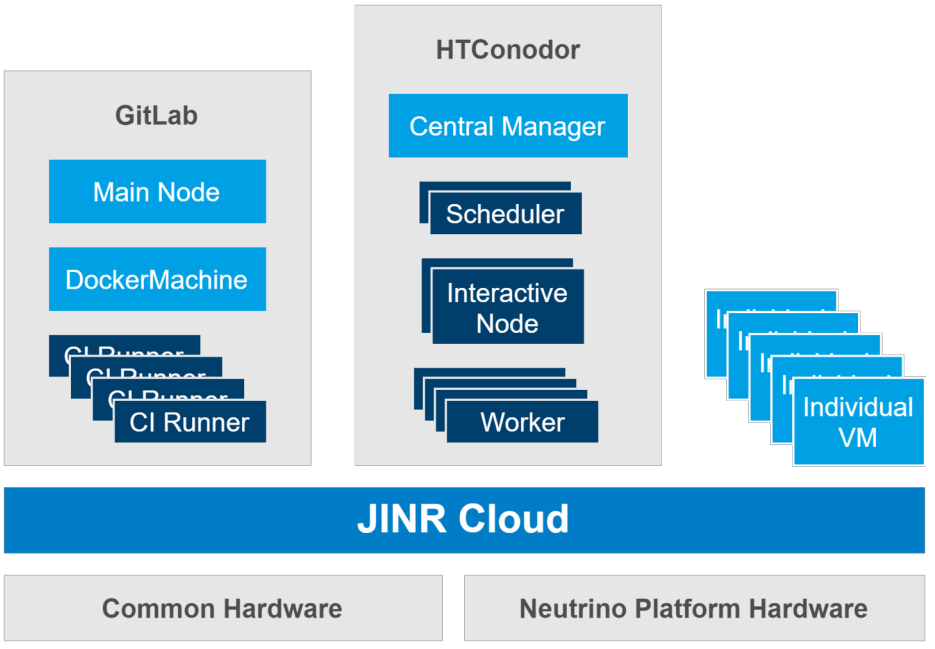
Nikita Balashov*, Nikolay Kutovskiy, Nikita Tsegelnik

The 9th International Conference "Distributed Computing and Grid Technologies in Science and Education" (GRID'2021)

9 July 2021

JINR Cloud Overview

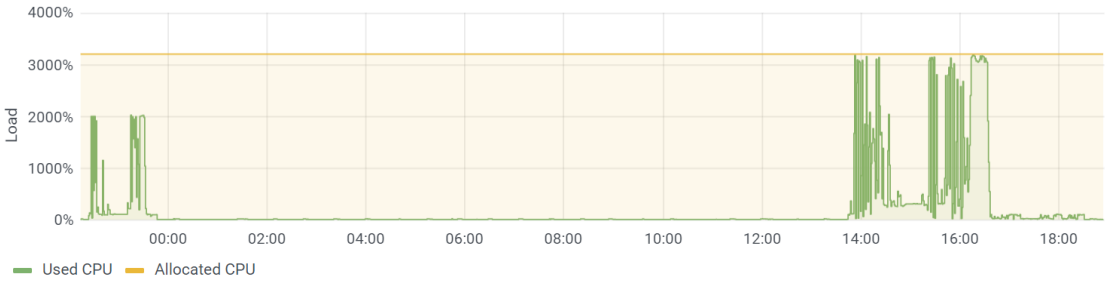
- JINR Cloud is based on **OpenNebula**:
  - Implements Infrastructure-as-a-Service model
  - Provides personal virtual machines (VM) to individual researchers
  - Hosts a number of multi-user systems and provides them as cloud services
- Cloud services:
  - GitLab/GitLab CI (in operation)
  - HTCondor cluster (in operation, restricted to neutrino experiments)
  - JupyterHub (under construction)
- Two types of cloud resources:
  - **Shared** (in common use by all JINR participants)
  - Neutrino platform resources: **owned** and partially shared by the Baikal-GVD, NOvA, DUNE and JUNO



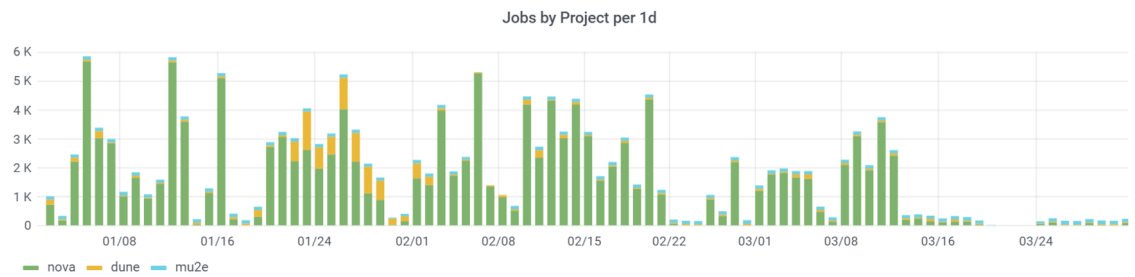
Background and Motivation

- Resources are sometimes underutilized
  - Interactive machines are rarely used at night
  - Batch job rates of certain projects may be irregular and have long periods of inactivity
- System efficiency can be defined in different ways
  - With batch systems we normally aim at maximizing hardware utilization
  - In case of interactive systems (e.g. JupyterHub) we want to keep the system responsive
- Resource owners may want to:
  - share their resources with other experiments/research groups
  - redistribute resources between cloud services

Typical interactive machine usage

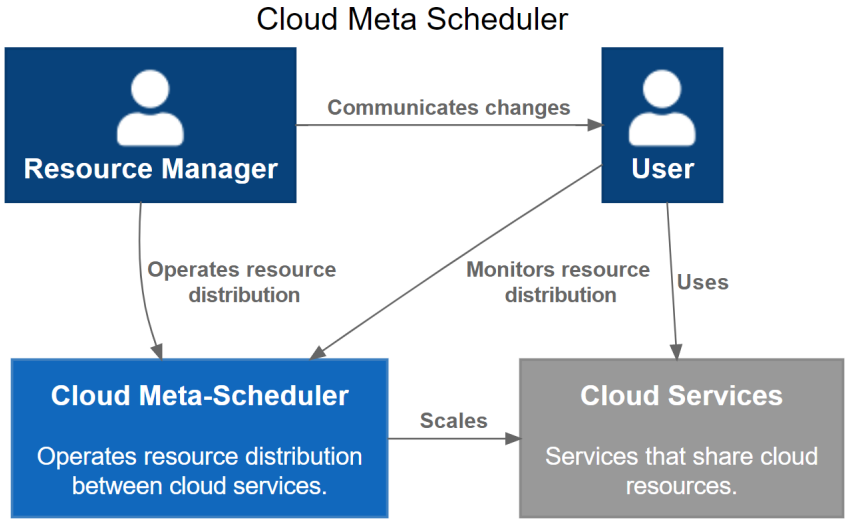


Batch jobs rate



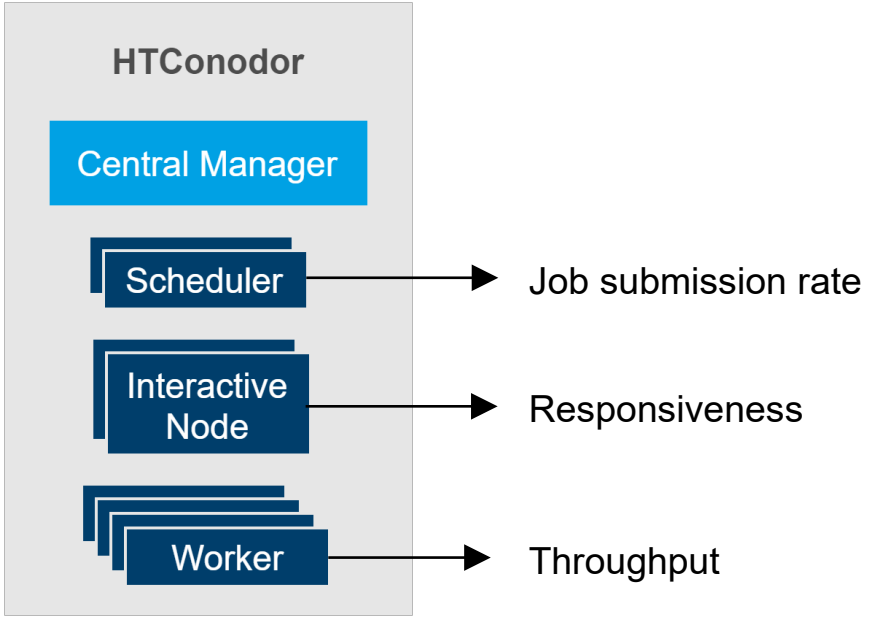
Cooperative Resource Sharing

- Facilitate cooperation in resource sharing
- Resource manager operates resource distribution on behalf of resource owners:
  - Scales available cloud services
  - Creates lease requests to other resource owners
  - Approves lease requests from other resource owners
- Users can monitor current resource distribution and usage
- Meta-scheduler handles cloud services scaling



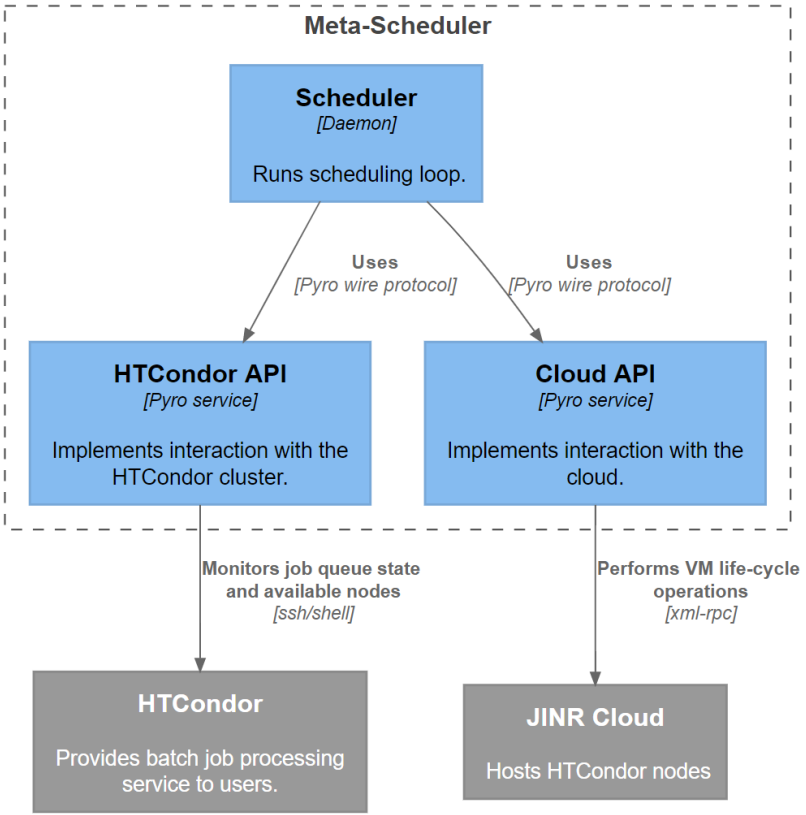
Scaling Cloud Services

- Many services can be scaled horizontally, but it may be tricky
- Different services (or their different parts) can be scaled for **different reasons**
- We need a tool to perform **centralized** scaling in two modes:
  - Automatic – based on service usage metrics
  - Manual – based on resource manager requests
- **Simple visual interface** for resource managers



Meta-Scheduler Prototype

- Evaluate possible technical solutions
- Try to discover potential pitfalls
- Refine system requirements
- Automatic scaling of HTCondor worker-nodes based on job queue size
- Test out microservices approach



Current Results and Plans

- We developed a scheduler with a simple strategy for scaling HTCondor worker nodes automatically
- The prototype was tested in a sandbox environment
- After testing it in the production environment, we'll start working towards a more generic solution for automatic scaling of other services
- So far, we plan at supporting at least 3 services: HTCondor, GitLab/CI and JupyterHub
- Develop a system for resource lease requests/approvals
- Develop a web-interface for end-users (most likely based on Django)

Thanks!