ATLAS usage of the Czech national HPC center: HyperQueue, cvmfsexec, and other news

M. Svatoš, J. Chudoba, P. Vokáč

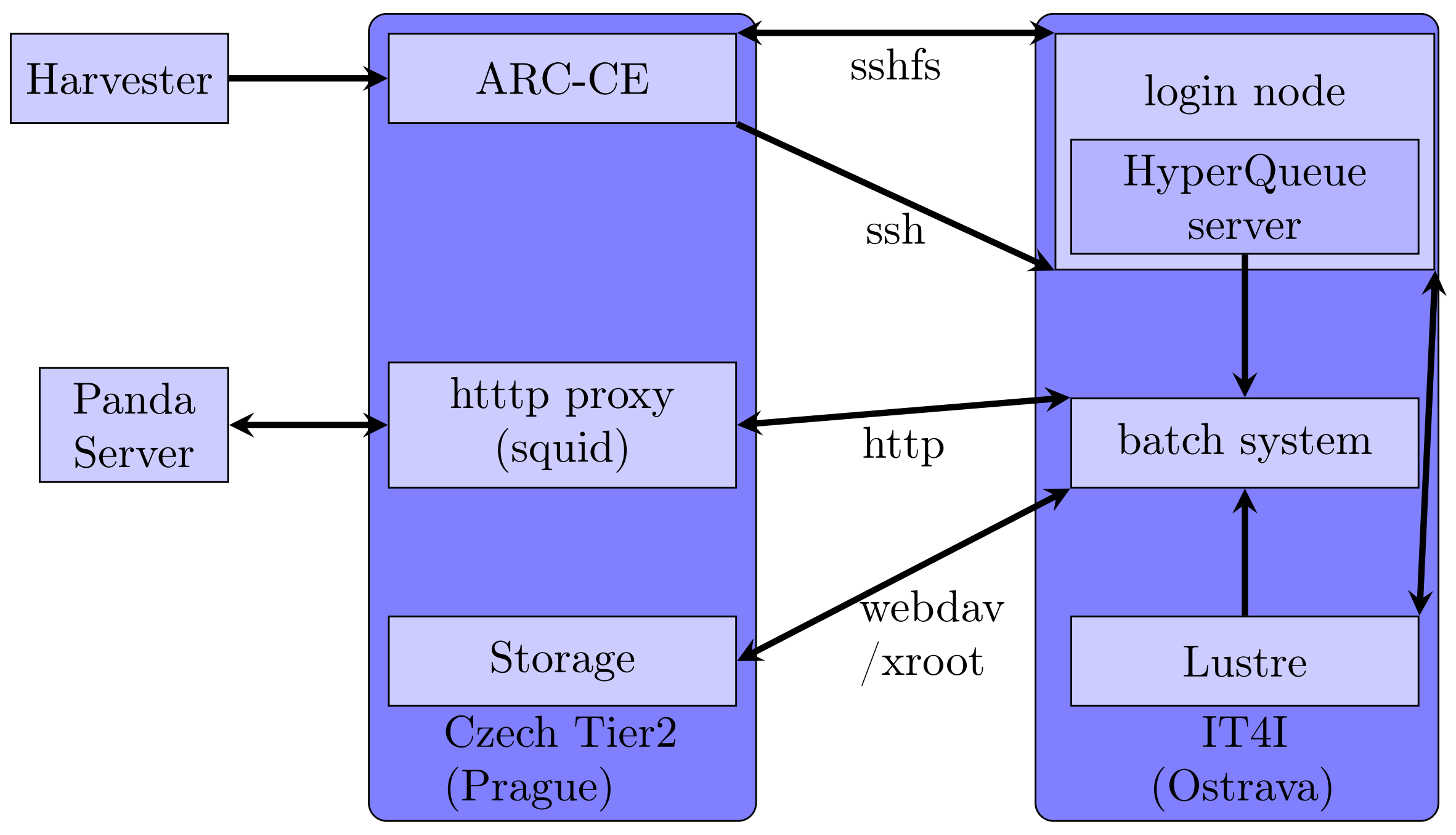
On behalf of the ATLAS Software & Computing Activity

Introduction

The ATLAS distributed computing is using resources of the Czech national HPC center IT4Innovations in Ostrava through Czech Tier2 praguelcg2 in Prague:

- **Anselm** (2013-2021)
  - CPU nodes: 180 WNs with 16 cores and 64GB of RAM
  - in production in ATLAS: 2020-2021
- **Salomon** (2017-2021)
  - CPU nodes: 576 WN with 24 cores and 128GB of RAM
  - in production in ATLAS: 2017-2021
- **Barbora** (2019-present)
  - CPU nodes: 192 WN with 36 cores and 192GB of RAM
  - in production in ATLAS: since 2020
- **Karolina** (2021-present)
  - CPU nodes: 720 WN with 128 cores and 256GB of RAM
  - in production in ATLAS: since 2021

Submission system



Harvester

ARC-CE

sshfs

login node

HyperQueue server

ssh

Panda Server

htttp proxy (squid)

http

batch system

Storage

webdav /xroot

Lustre

Czech Tier2 (Prague)

IT4I (Ostrava)

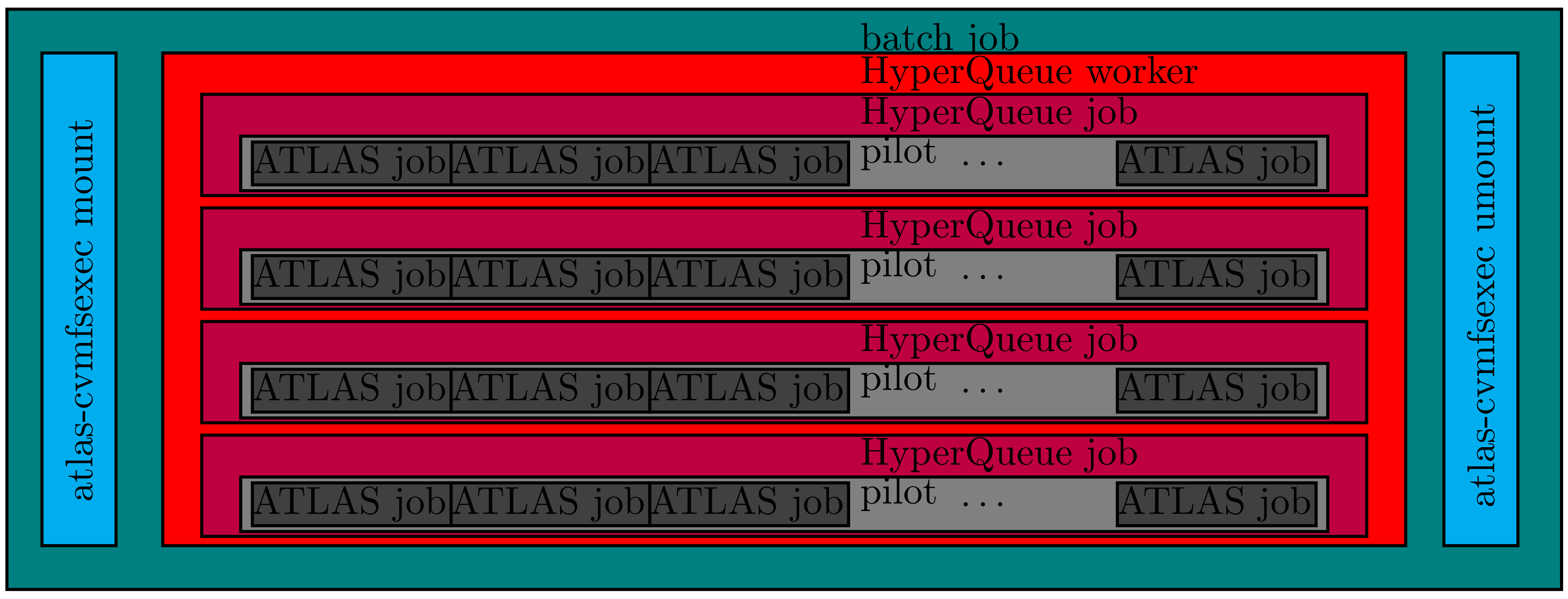
- the ARC-CE shares storage space with the Lustre via sshfs connection through a login node and communicates with the batch system via ssh connection (through other login node)
- when the ARC-CE receives a job, it translates the job description into a script that can be run in the batch system, puts necessary files into a folder within sshfs shared area and submits the job via ssh connection to the HyperQueue server running on a login node
- the HyperQueue server collects these job definitions and when there are enough of them, it submits jobs into the batch system
- when the batch job starts, atlas-cvmfsexec is mounted and then a HyperQueue worker starts and is filled with HyperQueue jobs
- in each HyperQueue job, pilot wrapper starts, launching the pilot
- pilot contacts panda server through http proxy (Czech Tier2 squid) to receive a payload (as there are only few open ports at each HPC)
- when it receives the payload, it gets input file from the Czech Tier2 storage via xroot or webdav
- then it processes the payload
- when the payload finishes, it sends outputs to the Czech Tier2 storage via xroot or webdav
- when this is finished, pilot will request another payload (if it can expect that the job can finish)
- when all HyperQueue jobs are finished, the HyperQueue worker finishes and atlas-cvmfsexec is unmounted

HyperQueue

- `https://it4innovations.github.io/hyperqueue/latest/`
- HyperQueue is a tool designed to simplify execution of large workflows on HPC clusters
- submission from ARC-CE to HyperQueue requires only minor changes in ARC-CE scripts
- HPCs moved from PBSpro to slurm and it runs on both

user perspective

- small binary with no dependencies which one can just download and run
- works as batch system under my control within a batch system outside of my control
  - batch system schedules only whole node jobs but I can split it (one 128 core job into four 32 core jobs)
  - batch system enforces re-runnable jobs but I can set them to be not re-runnable



atlas-cvmfsexec mount

batch job

HyperQueue worker

HyperQueue job

ATLAS job ATLAS job ATLAS job pilot . . . ATLAS job

HyperQueue job

ATLAS job ATLAS job ATLAS job pilot . . . ATLAS job

HyperQueue job

ATLAS job ATLAS job ATLAS job pilot . . . ATLAS job

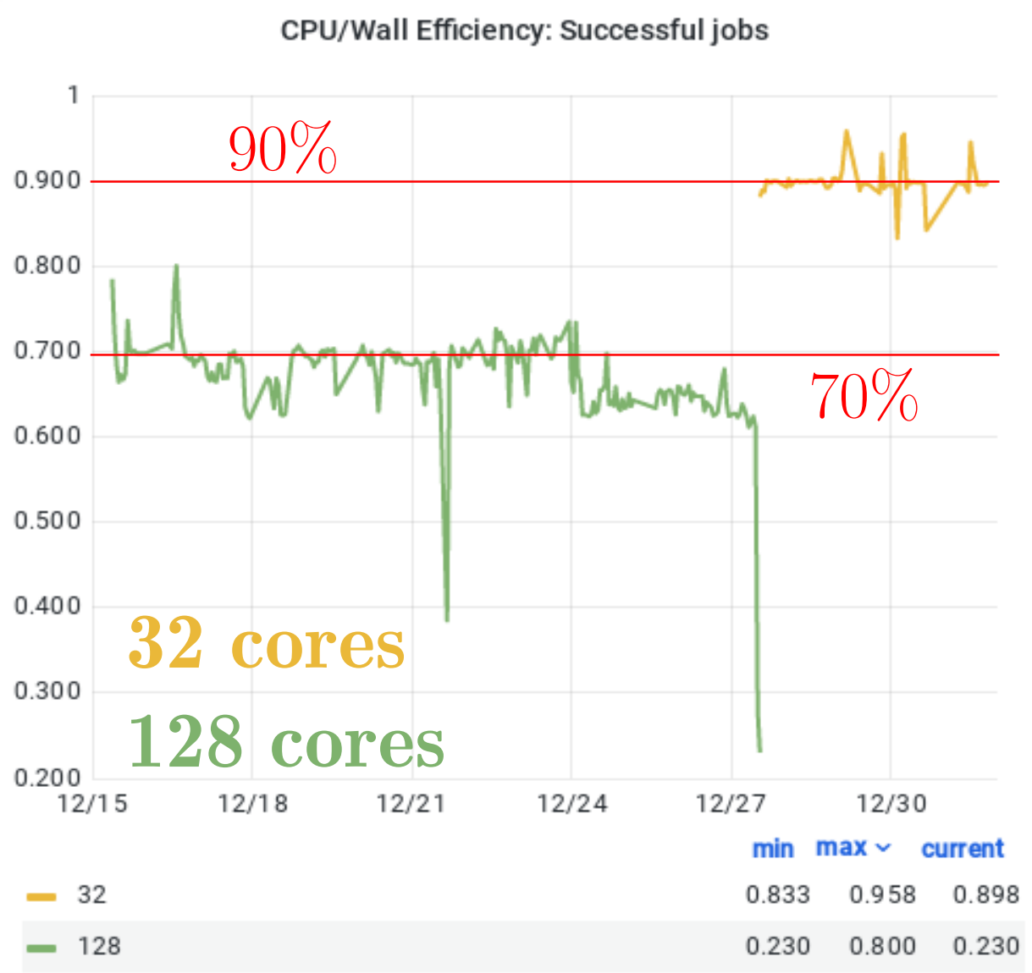
HyperQueue job

ATLAS job ATLAS job ATLAS job pilot . . . ATLAS job

atlas-cvmfsexec umount

filling efficiency

N.B.: On Karolina, the switch from one 128-core job to four 32-cores jobs (allowed by HyperQueue) was motivated by CPU efficiency increase:

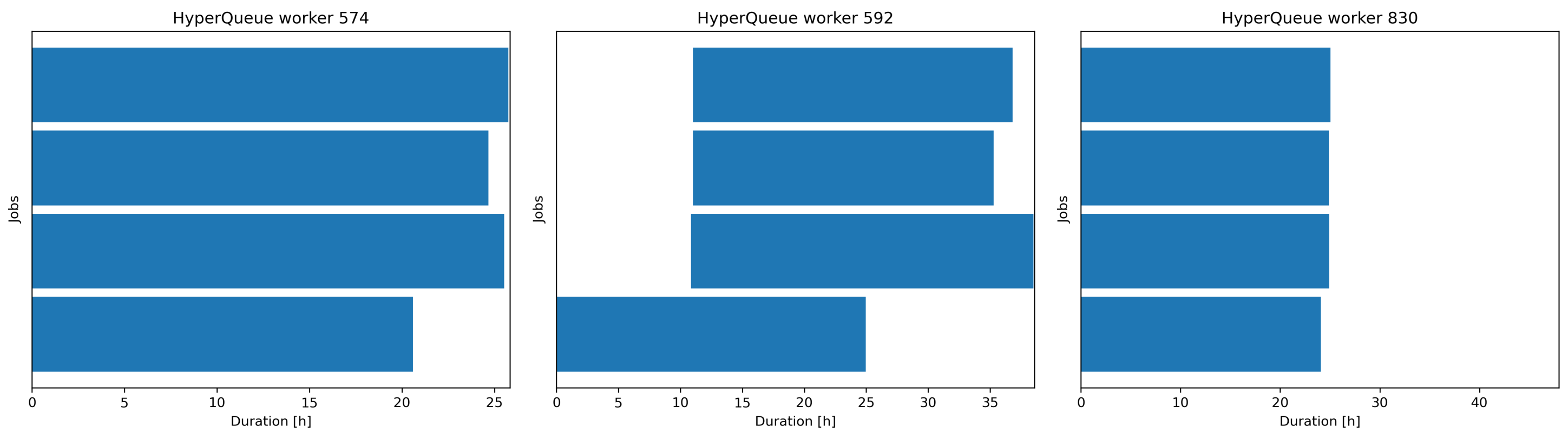


Measurement: 142 HyperQueue workers: start and end dates of jobs and workers

Results:

$$\frac{\text{sum of all time used by jobs}}{\text{sum of all time available in workers}} = 93\%$$

common case: exceptional cases:



possible causes od exceptional cases:

- killed job could cause early end (if something was stuck and keep running until max time)
- late start could be caused by lack of jobs

Acknowledgement

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254). The work was also supported by the project CERN-CZ LM2023040.

atlas-cvmfsexec

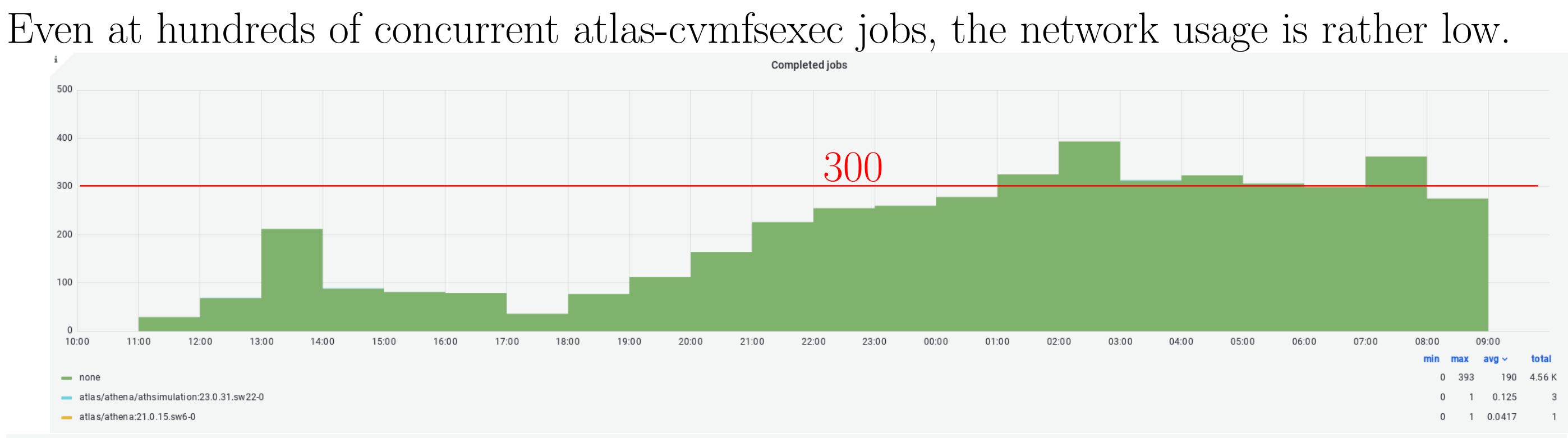
When the CVMFS is not installed, jobs can run in so-called fat containers (FC), which contain everything the job needs, or use software from atlas-cvmfsexec:

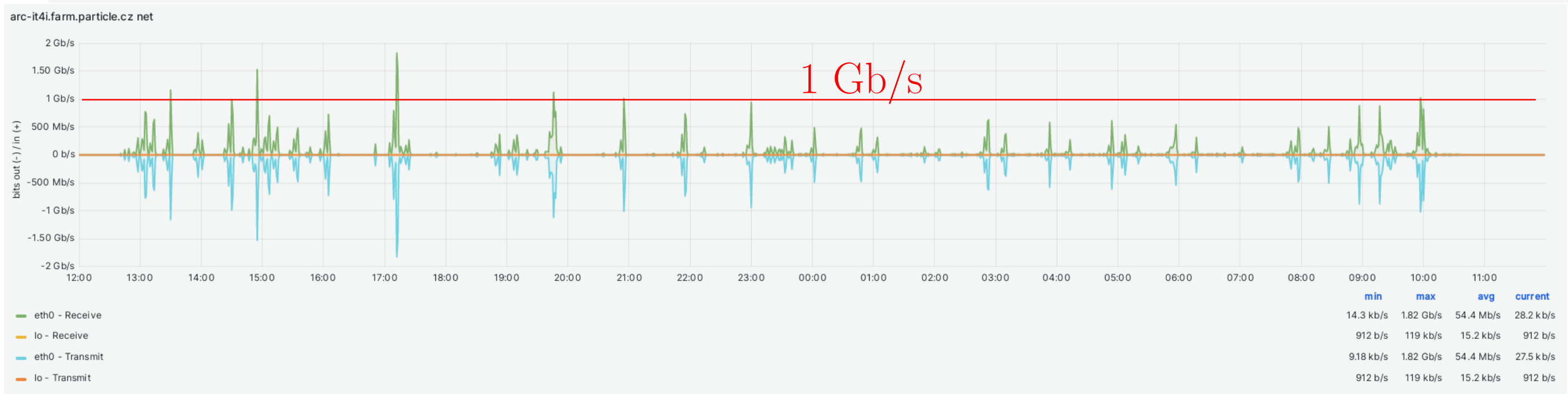
- `https://gitlab.cern.ch/atlas-tier3sw/atlas-cvmfsexec`
- WNs need access to a squid
- atlas-cvmfsexec is highly modular, i.e. there are parts of CVMFS (large or often used) which can be used from a local (rsynced) copy
  - local:
    - base system (CC7, alma9) container
    - fat container (FC) releases
    - condition ROOT files
    - DBRelease
  - from cvmfsexec
    - ALRB (for every job)
    - release software (for non fat container releases)

many thanks to Asoka De Silva for making atlas-cvmfsexec possible

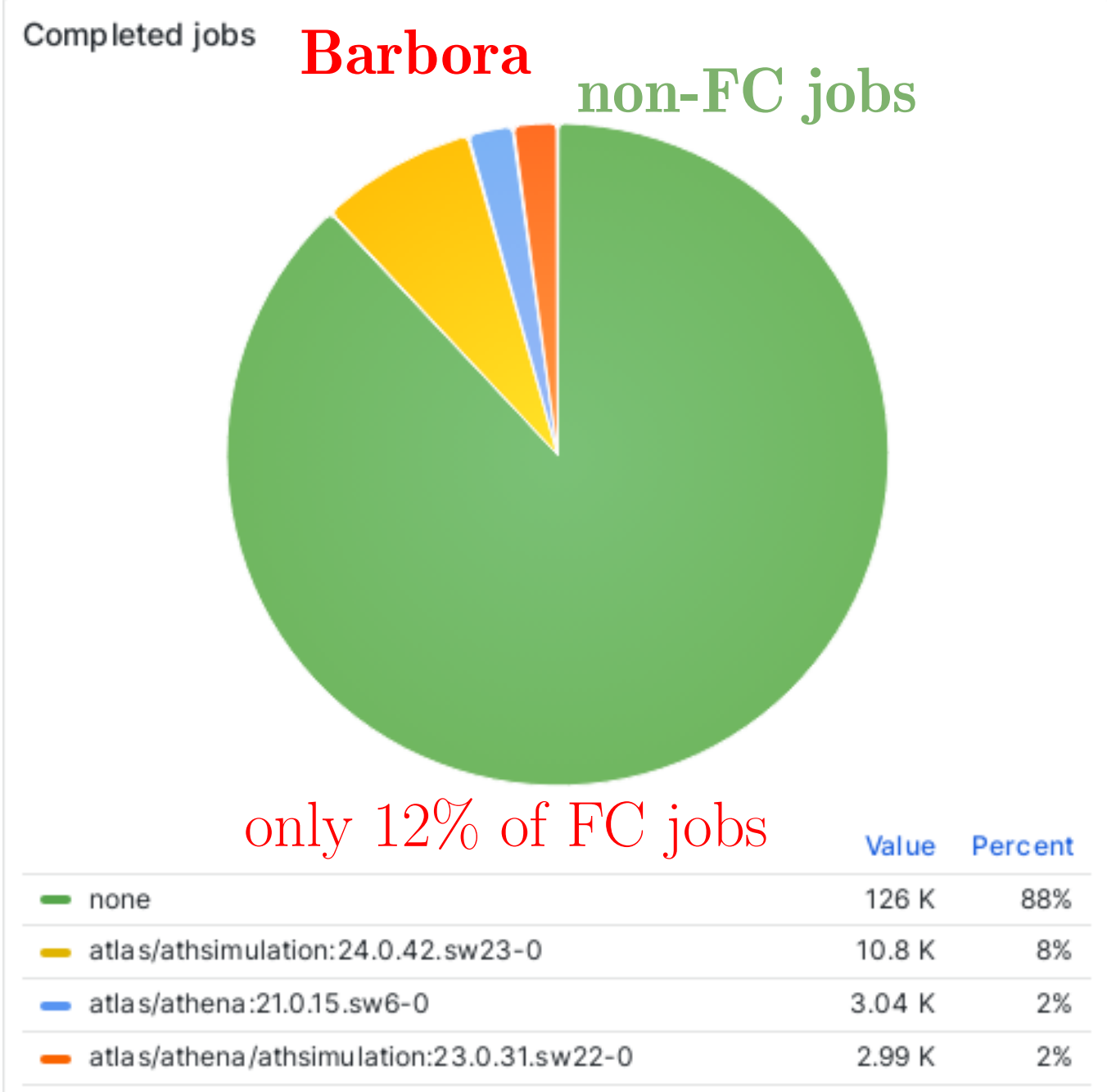
network usage

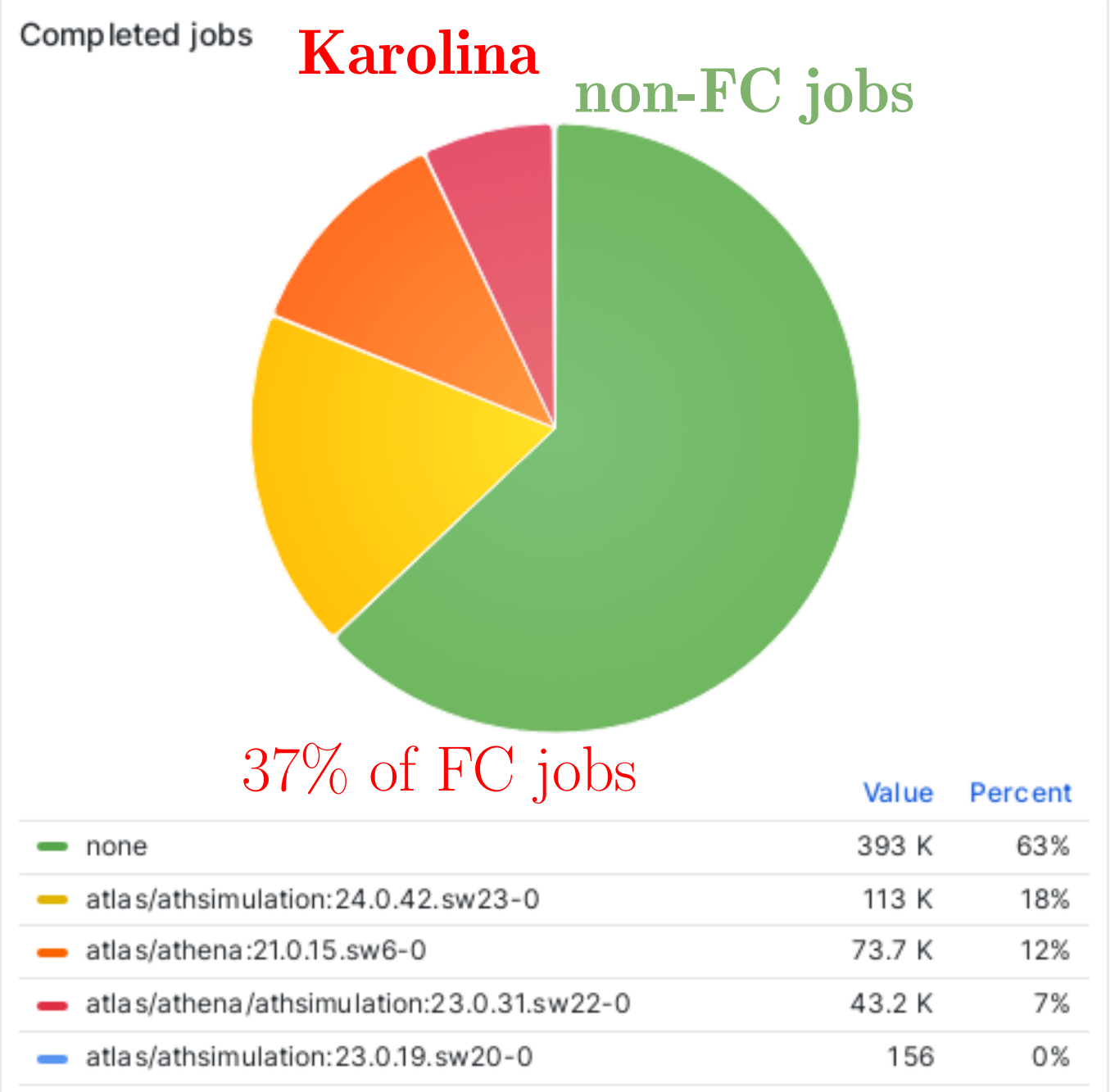
Even at hundreds of concurrent atlas-cvmfsexec jobs, the network usage is rather low.





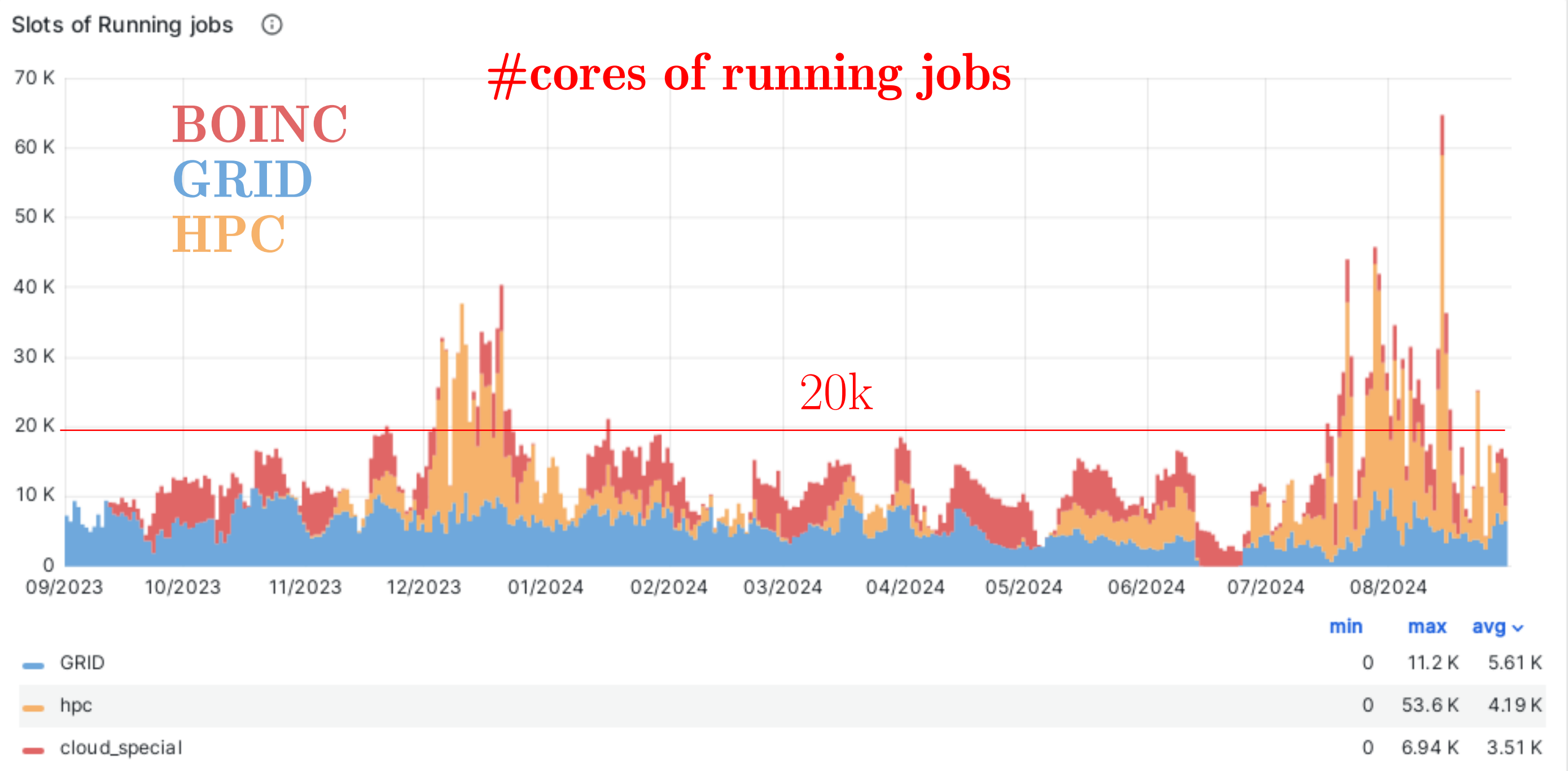
FC vs non-FC releases usage (9.2023 - 8.2024)

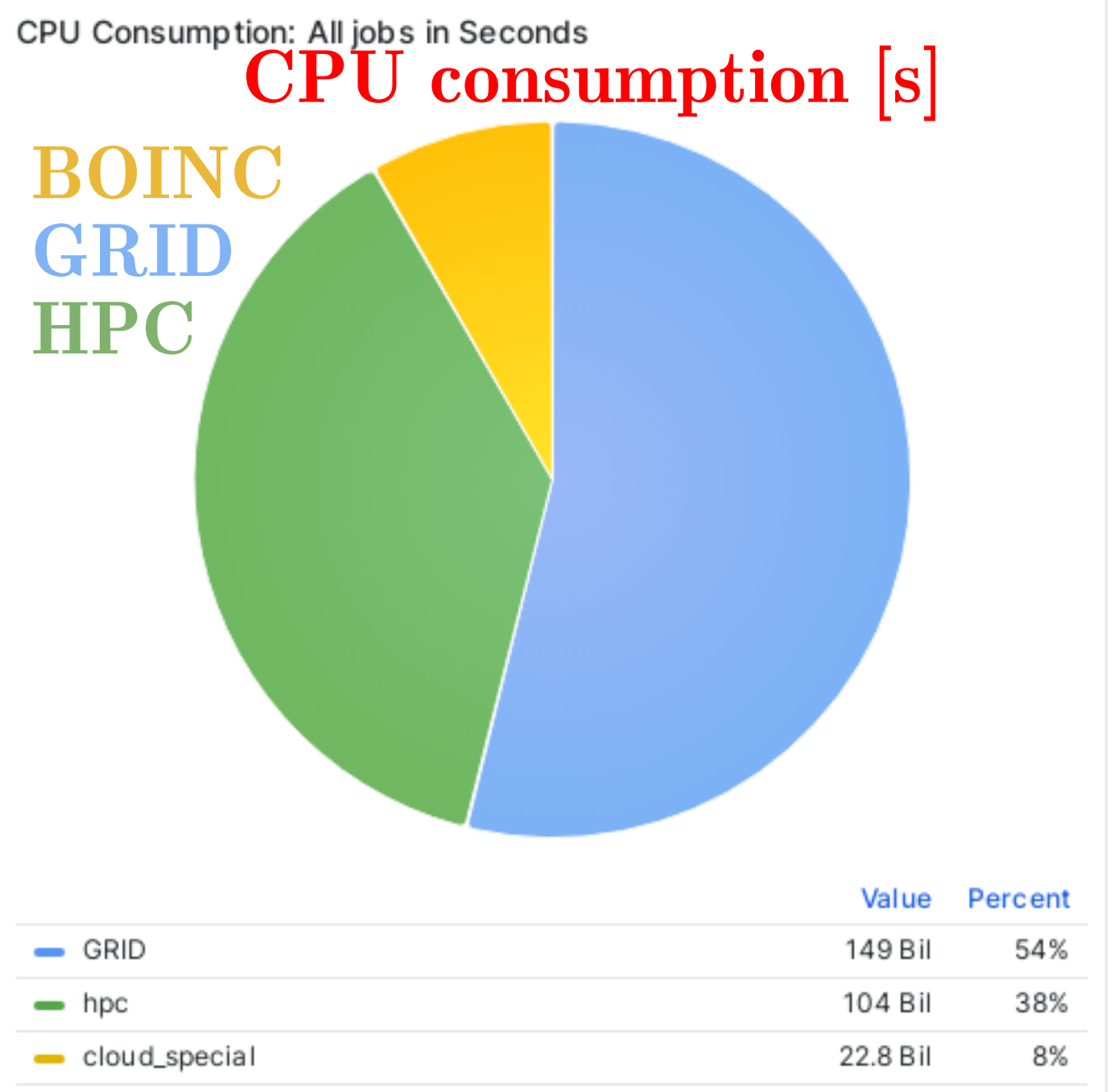


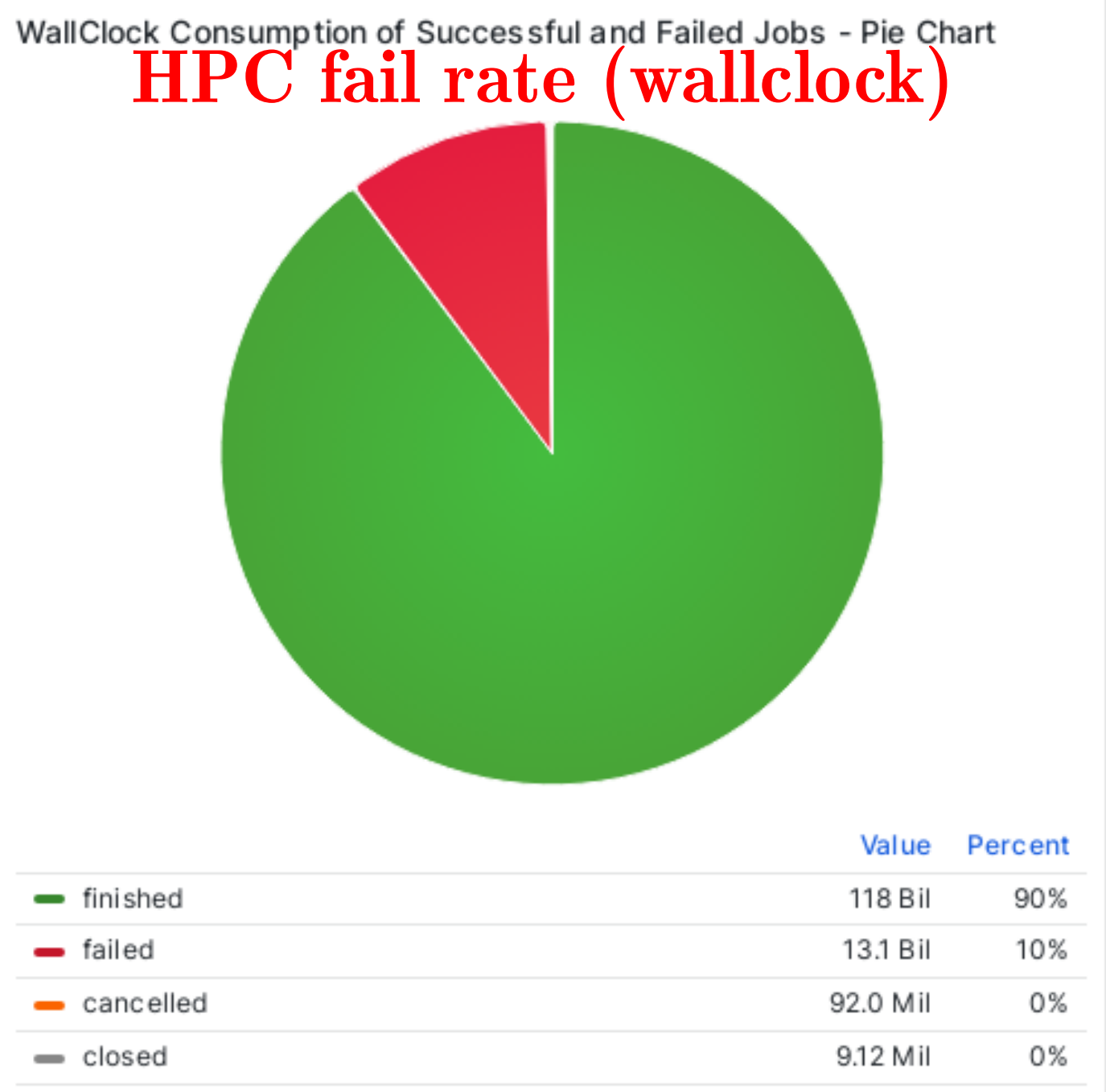


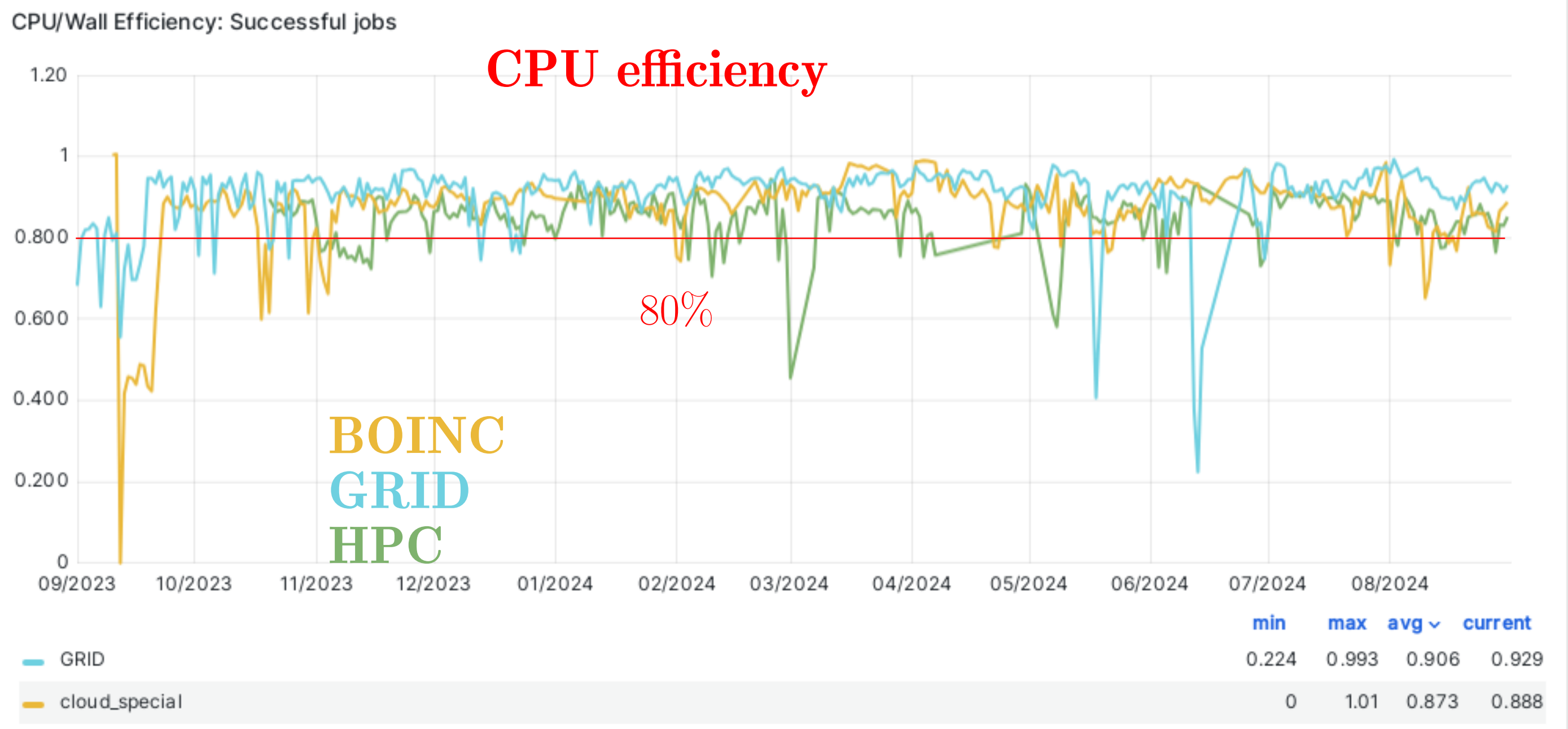
Performance

The grid jobs are running at Czech Tier2 computing center. The BOINC jobs are back-filling a cluster dedicated to local users.


Slots of Running jobs
#cores of running jobs
BOINC
GRID
HPC
70 K
60 K
50 K
40 K
30 K
20 K
10 K
20k
09/2023
10/2023
11/2023
12/2023
01/2024
02/2024
03/2024
04/2024
05/2024
06/2024
07/2024
08/2024
min max avg
GRID 0 11.2 K 5.61 K
hpc 0 53.6 K 4.19 K
cloud_special 0 6.94 K 3.51 K


CPU Consumption: All jobs in Seconds
CPU consumption [s]
BOINC
GRID
HPC
Value Percent
GRID 149 Bil 54%
hpc 104 Bil 38%
cloud_special 22.8 Bil 8%


WallClock Consumption of Successful and Failed Jobs - Pie Chart
HPC fail rate (wallclock)
Value Percent
finished 118 Bil 90%
failed 13.1 Bil 10%
cancelled 92.0 Mil 0%
closed 9.12 Mil 0%


CPU/Wall Efficiency: Successful jobs
CPU efficiency
BOINC
GRID
HPC
09/2023
10/2023
11/2023
12/2023
01/2024
02/2024
03/2024
04/2024
05/2024
06/2024
07/2024
08/2024
min max avg current
GRID 0.224 0.993 0.906 0.929
cloud_special 0 1.01 0.873 0.888
hpc 0.456 0.944 0.850 0.851