IRIS-HEP AGC workshop 2023

Experience with AGC at German facilities

David Koch, Thomas Kuhr, Günter Duckeck

Objectives

This work is funded by FIDIUM (Federated Digital Infrastructures for Research on Universe and Matter)

- explore, try out, develop and test novel analysis techniques and facilities for HEP analyses
- AGC perfect match to perform reproducable benchmarks
- test distributed dask-based workflow of the AGC ttbar analysis on various sites

Changes in code

The AGC version I used for these tests is a couple of months behind the main branch now (merging jupyter notebooks is a pain)

- construct fileset that uses xrootd-urls to `LRZ-LMU_LOCALGROUPDISK` storage
- in order to be able to authenticate with xrootd, a couple of environment variables have to be set at the top of the notebook, like

```
os.environ["X509_USER_PROXY"] = "/path/to/proxy"
```

Changes in code

I want to use dask to distribute the workload across a SLURM / HTCondor cluster → create a dask client that is backed up by a cluster and pass that client to the coffea executor

```
from dask.distributed import Client
from dask_jobqueue import SLURMCluster

cluster = SLURMCluster(
    name=...,
    cores=...,
    memory=...,
)
cluster.scale(75)     # <-- easily scale the cluster up or down
client = Client(cluster)
```

Sites

LMU

institute cluster at LMU Munich consisting of one very powerful node and desktop computers

LRZ

WLCG Tier-2 site in Munich

Vispa

analysis facility operated by RWTH Aachen; provides a web-based terminal, code editor and jupyter hub: https://vispa.physik.rwth-aachen.de

Software environment

LMU

- distro: Ubuntu 20.04
- installation: all in one python virtual environment
- job-scheduler: SLURM
- reading of data via xrootd from LRZ

LRZ

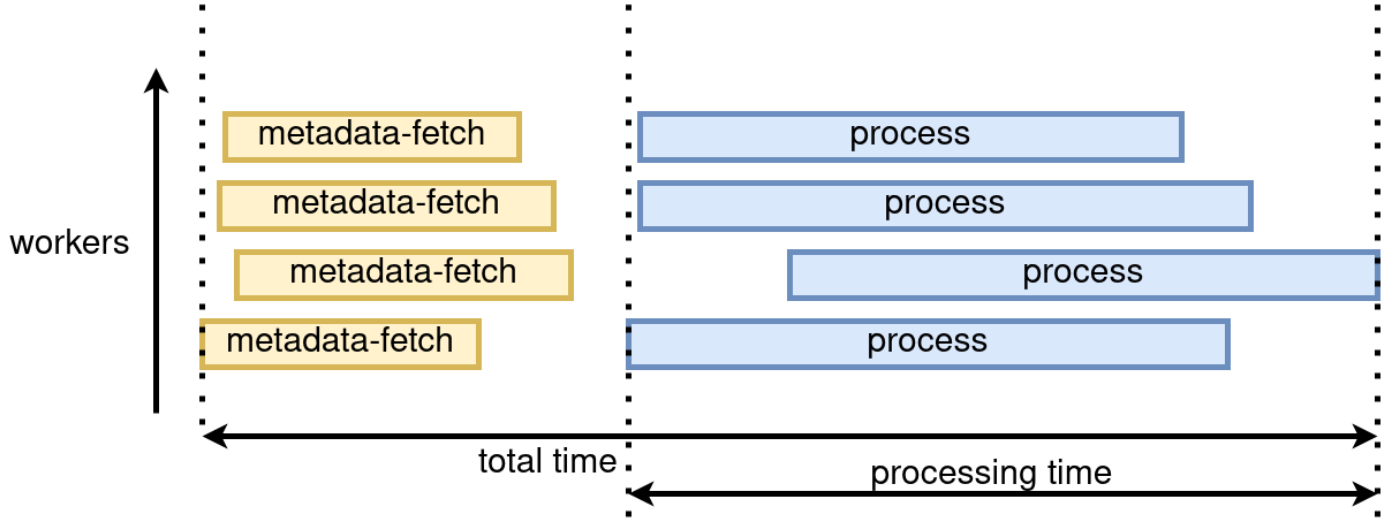
- distro: SUSE Linux
- software environment:
  - usual mode of operation for user-code: people log in, fire up a centos container (`setupATLAS -c centos8 -b`) and run their code in there
  - first I tried to run the AGC analysis from inside the container; was not able to do it technically + it's not expected to be feasable (?) $\rightarrow$ dropped this approach
  - used a conda environment in the end (issues with pip and `xrootd`)
- job scheduler: SLURM
- data is stored on regular Grid storage (HDD) as well as on a XCache server (SSD)

Vispa

- distro: Ubuntu 20.04
- software environment: conda
- jupyterhub is provided by the service, it allows to select your own conda environment as a kernel
- job scheduler: HTCondor
- data is stored on SSDs at Vispa and read via NFS; Vispa also has a very dedicated per node caching-system that I did not test with AGC yet

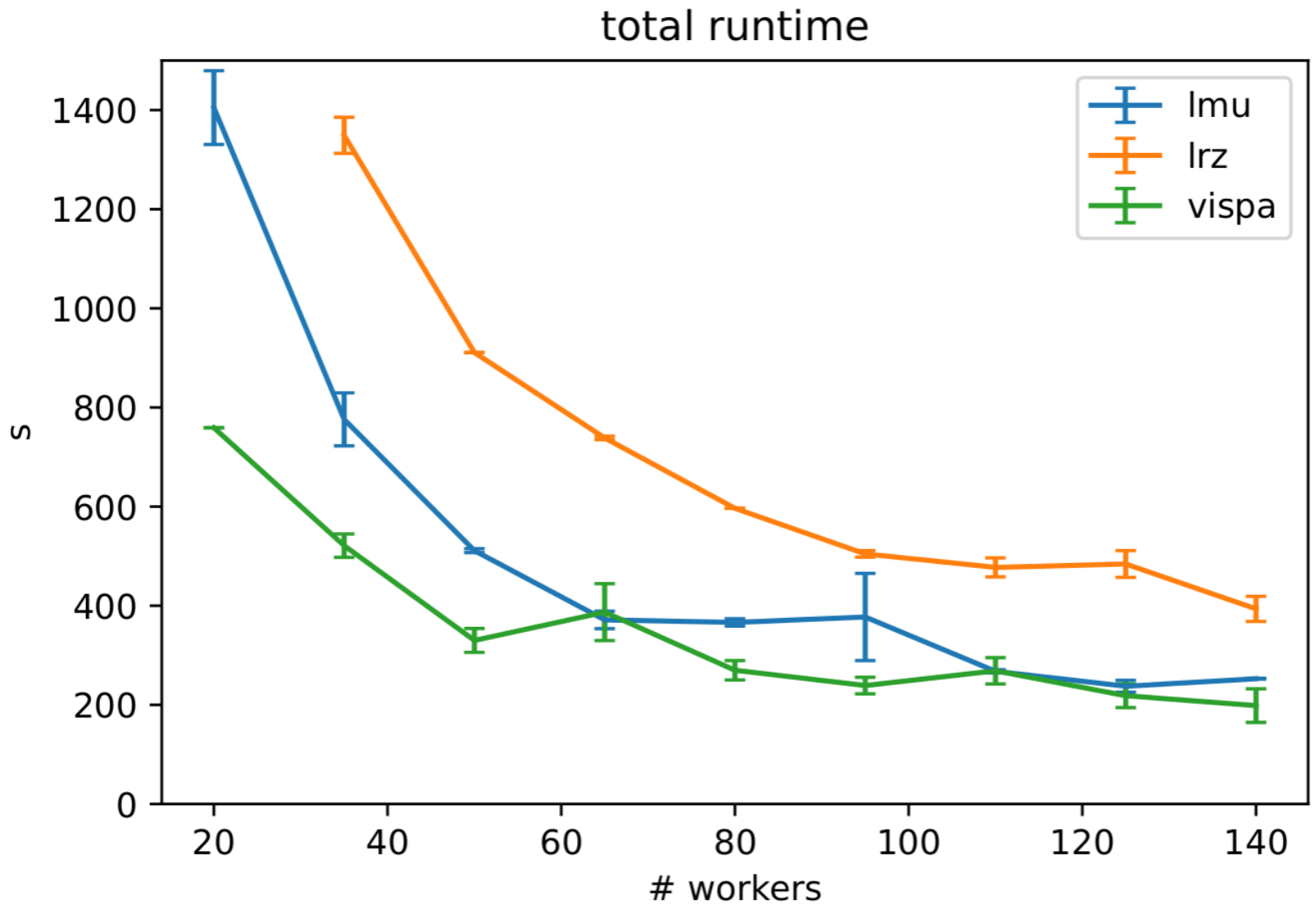
Measurements

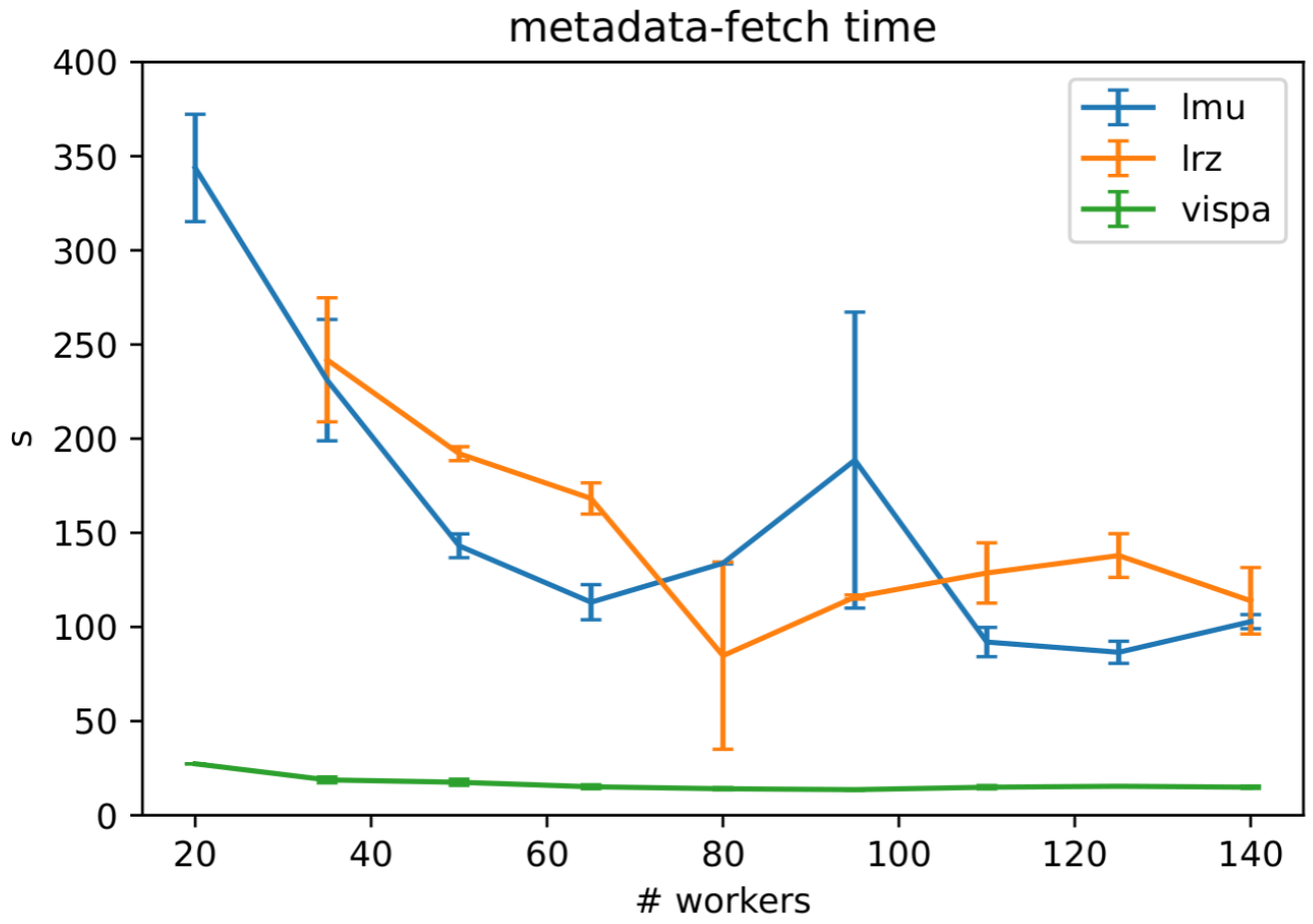
Only the part of the analysis which is run distributed is used for the benchmark → fetching some metadata and reading and processing the data



I can measure directly

- the total runtime
- the total processing time
- the sum of all process times across the workers (the sum of all blue rectangles)

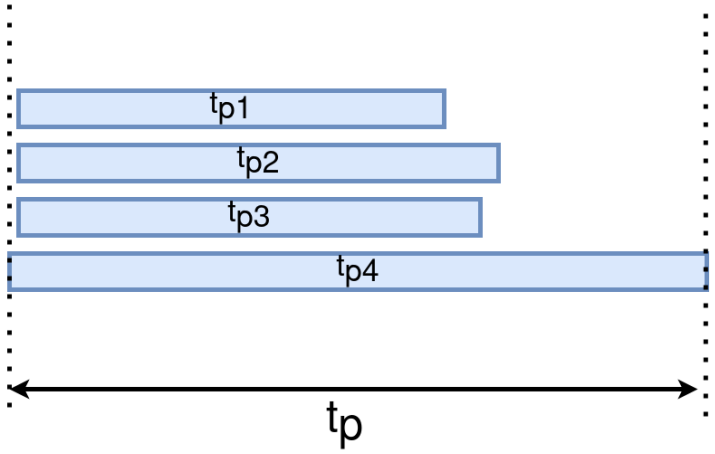

total runtime
lmu
lrz
vispa
1400
1200
1000
800
600
400
200
0
s
20
40
60
80
100
120
140
workers



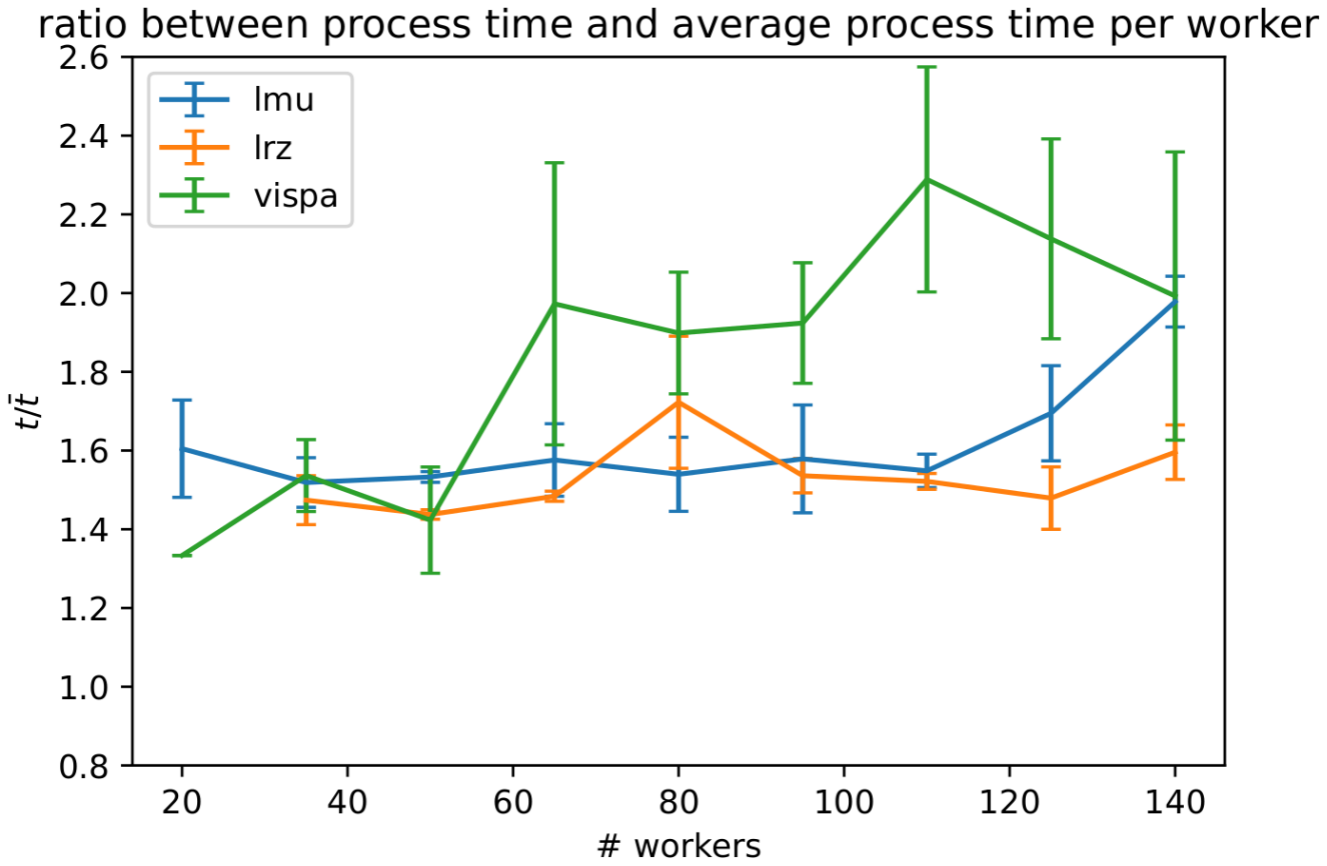
metadata-fetch time (= total - process time, also contains waiting and communication between dask workers)

measure for the amount of overhead
relative to the runtime

$$\frac{t_p}{\sum t_{p_i}/n}$$

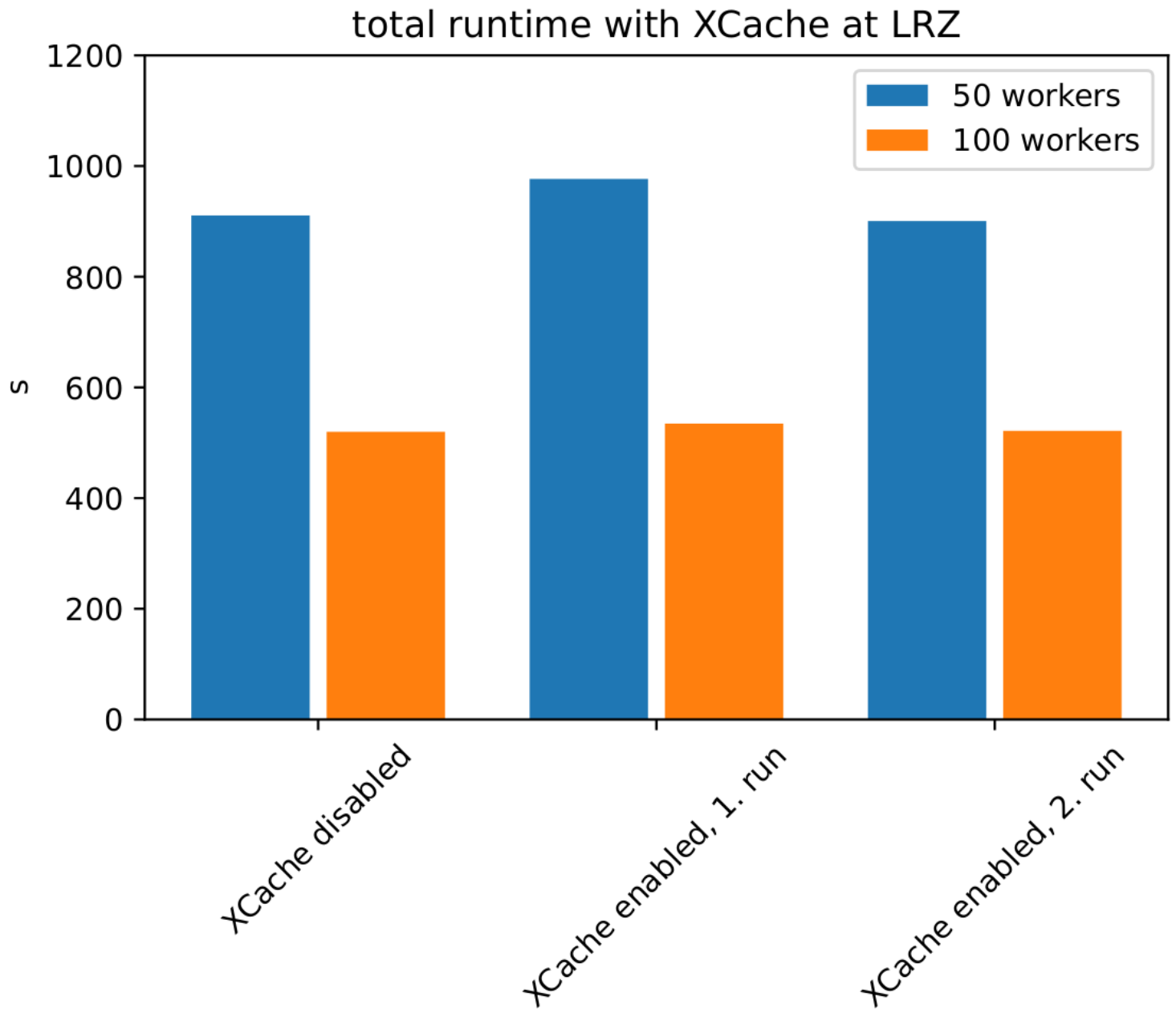


$\sum t_{p_i}/n =: \bar{t}$ is the average process time per worker without overhead



runtimes at LRZ with and without XCache enabled: makes no significant difference

$\Rightarrow$ with this setup, the analysis is hardly I/O limited



WIP / Further ideas

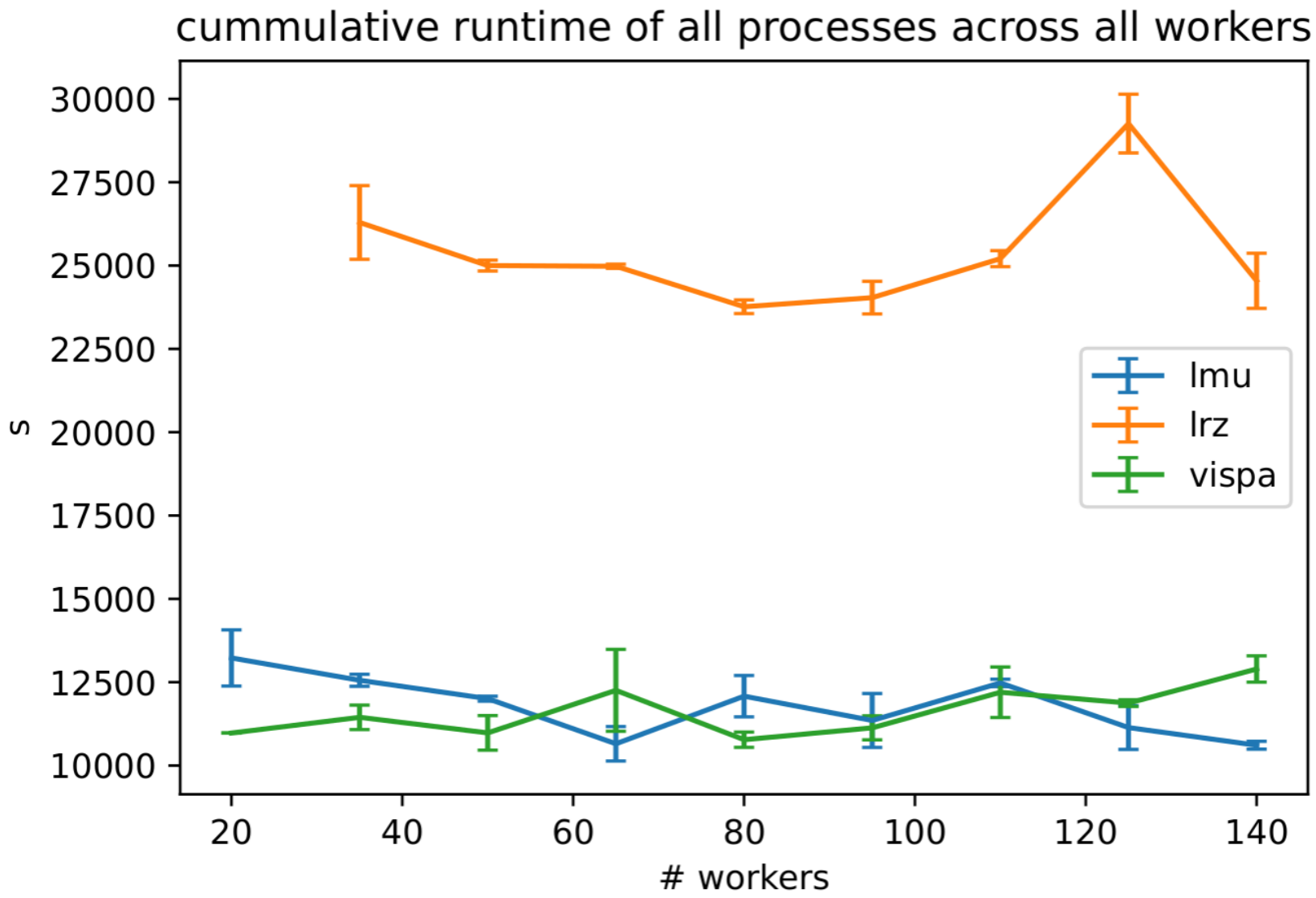
- find out what causes the overhead
- measure scaling of runtime with amount of data
- possible future project: have a benchmark analysis for Belle2

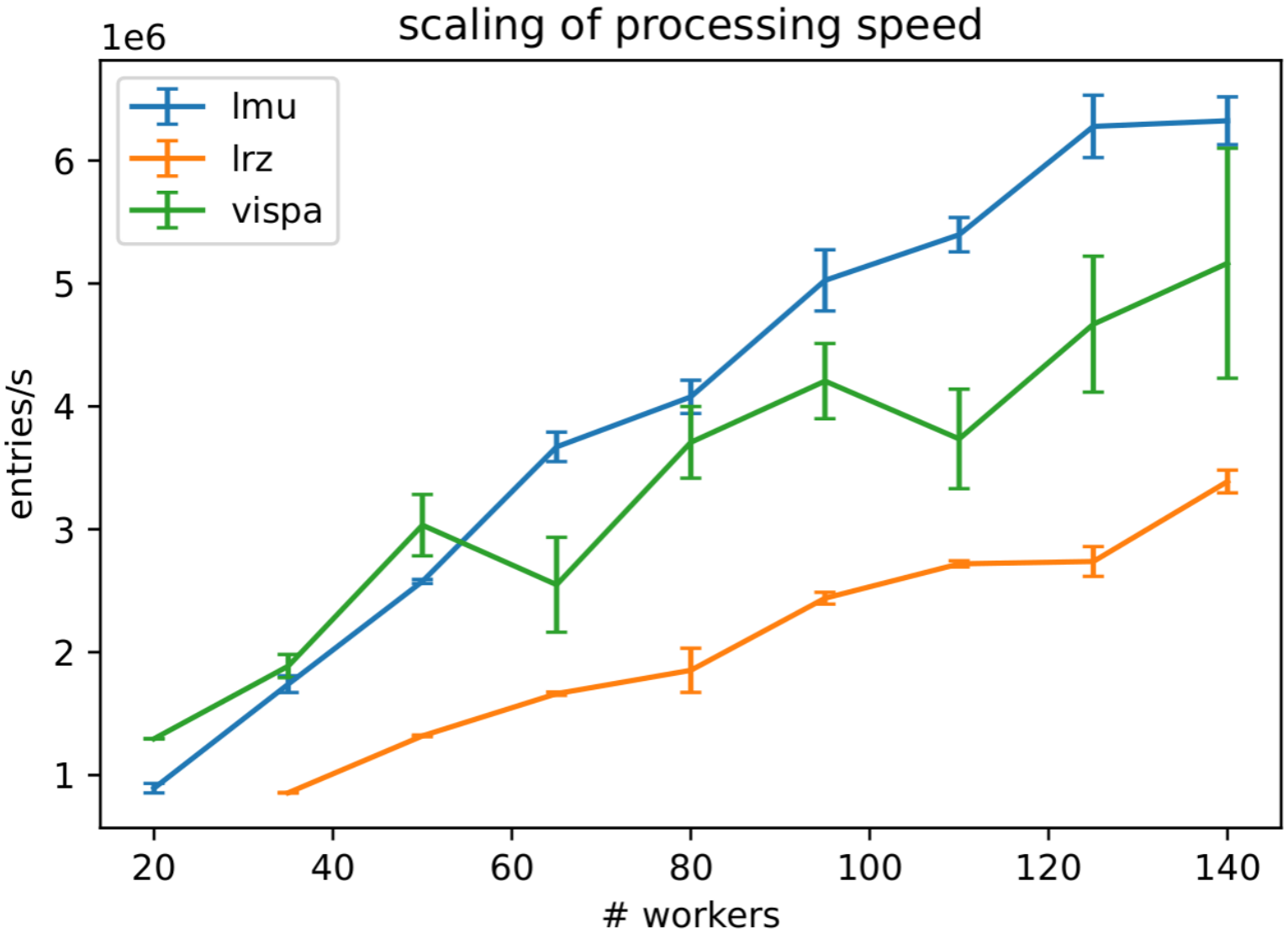
Questions?

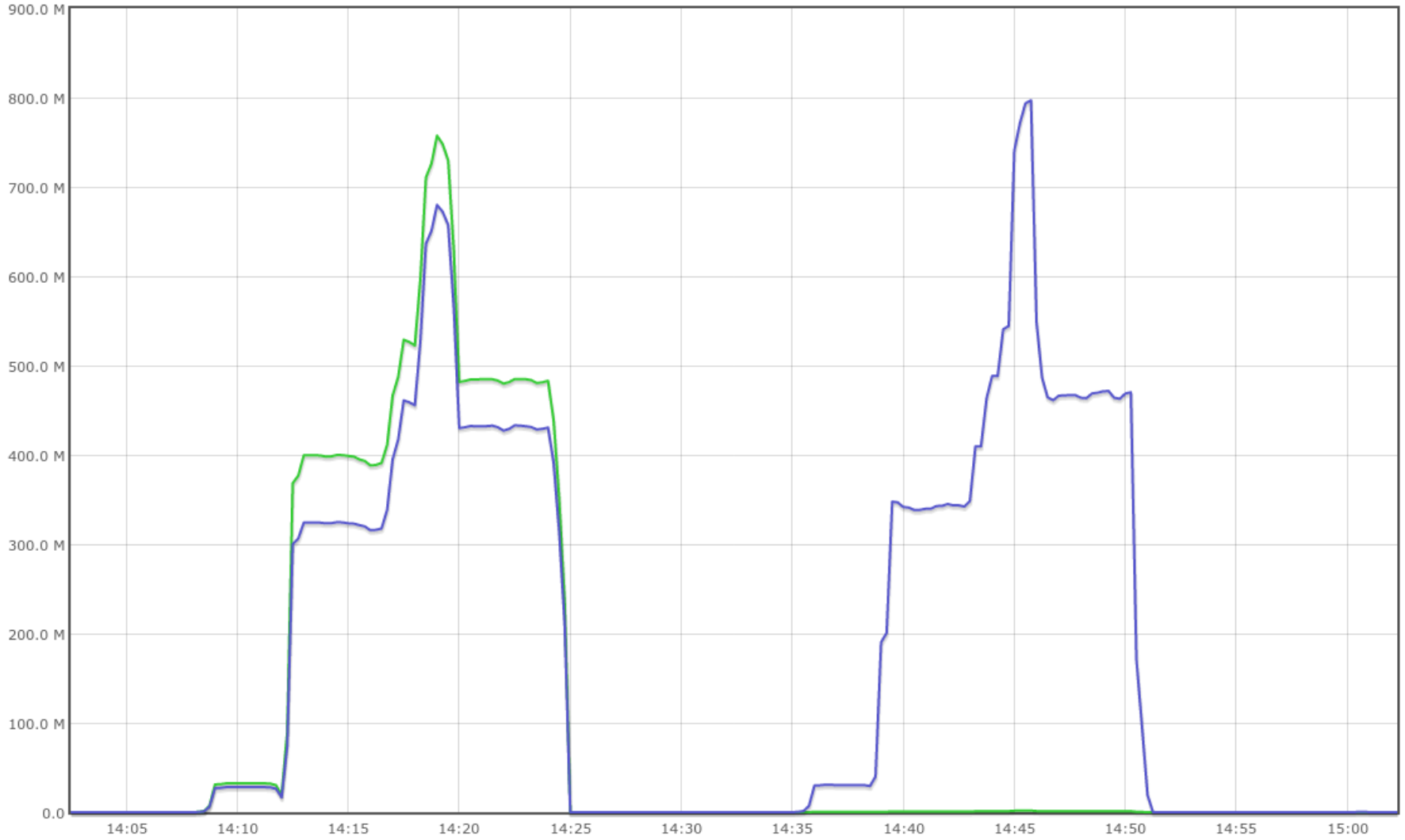
Backup

installation with conda + pip

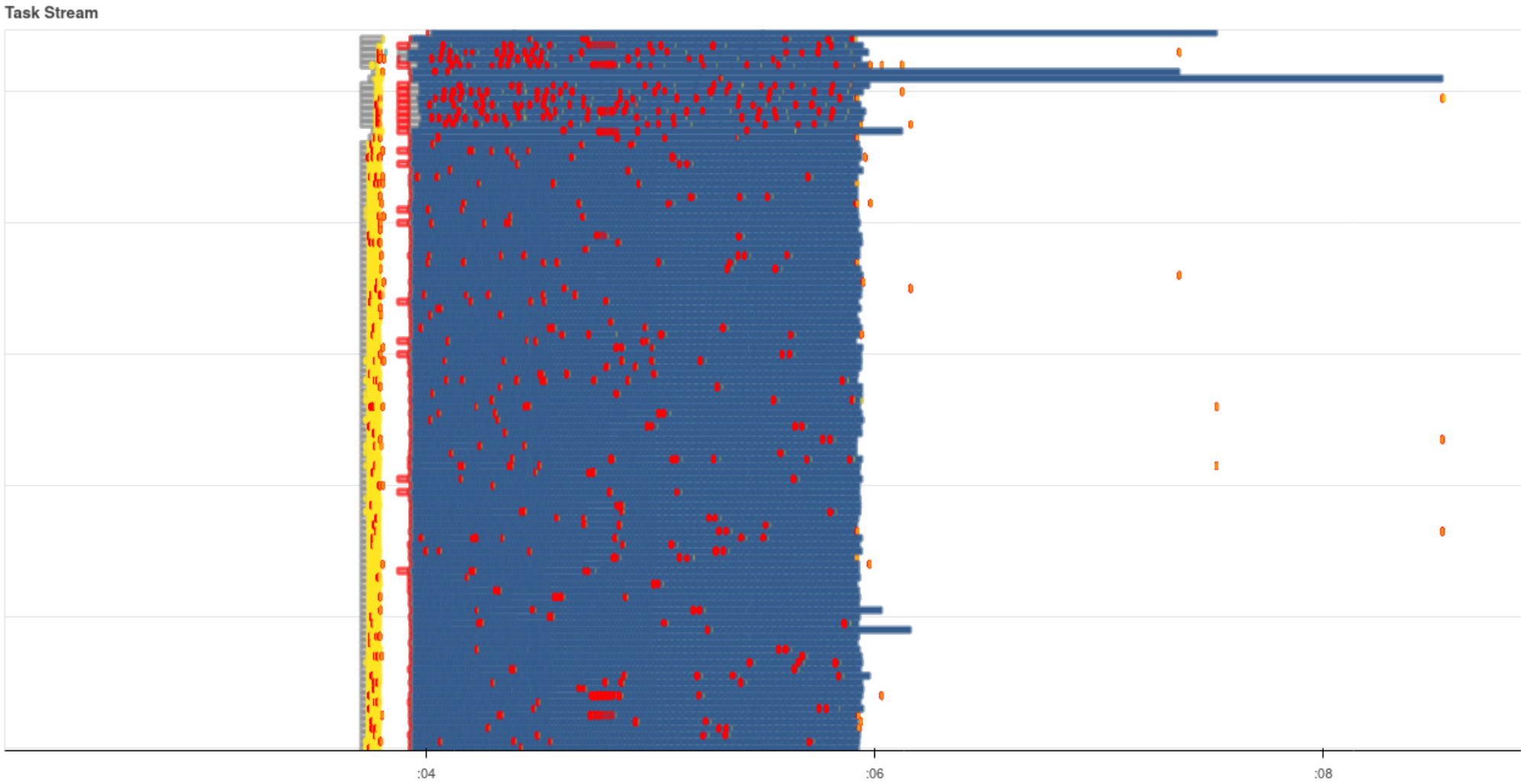
```
conda create -n agc python=3.8.10
conda activate agc
conda install -c conda-forge cabinetry
conda install -c conda-forge coffea vector awkward uproot
pip install servicex func-adl-servicex func-adl aiostream
conda install -c conda-forge dask dask-jobqueue
conda install -c conda-forge xrootd
conda install -c conda-forge jupyter
```


cummulative runtime of all processes across all workers
30000
27500
25000
22500
20000
17500
15000
12500
10000
s
20
40
60
80
100
120
140
workers
lmu
lrz
vispa

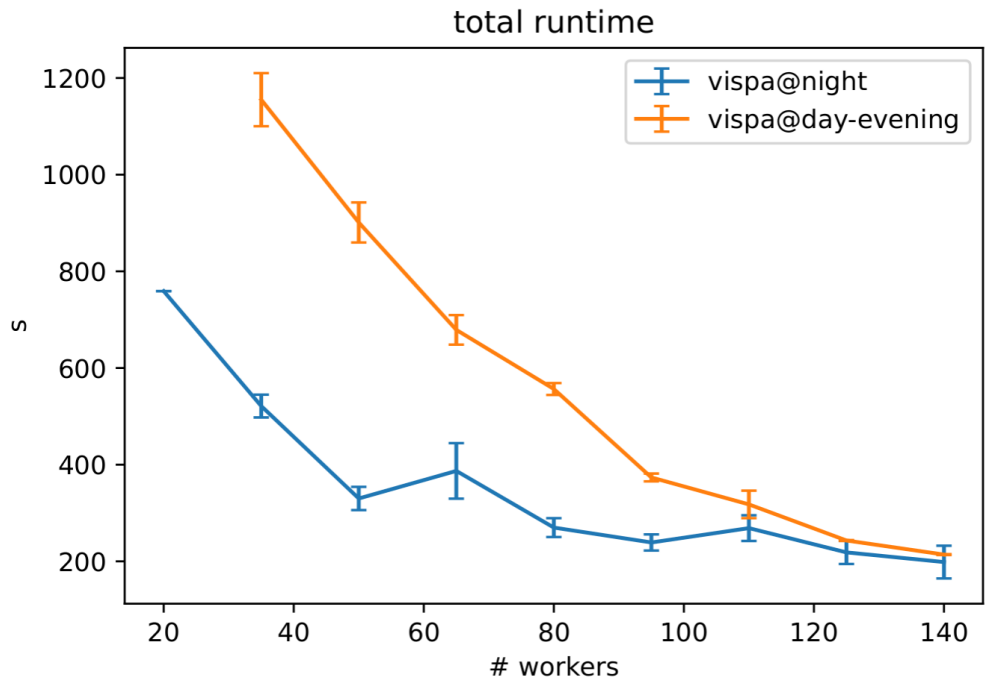

scaling of processing speed
1e6
lmu
lrz
vispa
entries/s
1
2
3
4
5
6
20
40
60
80
100
120
140
workers

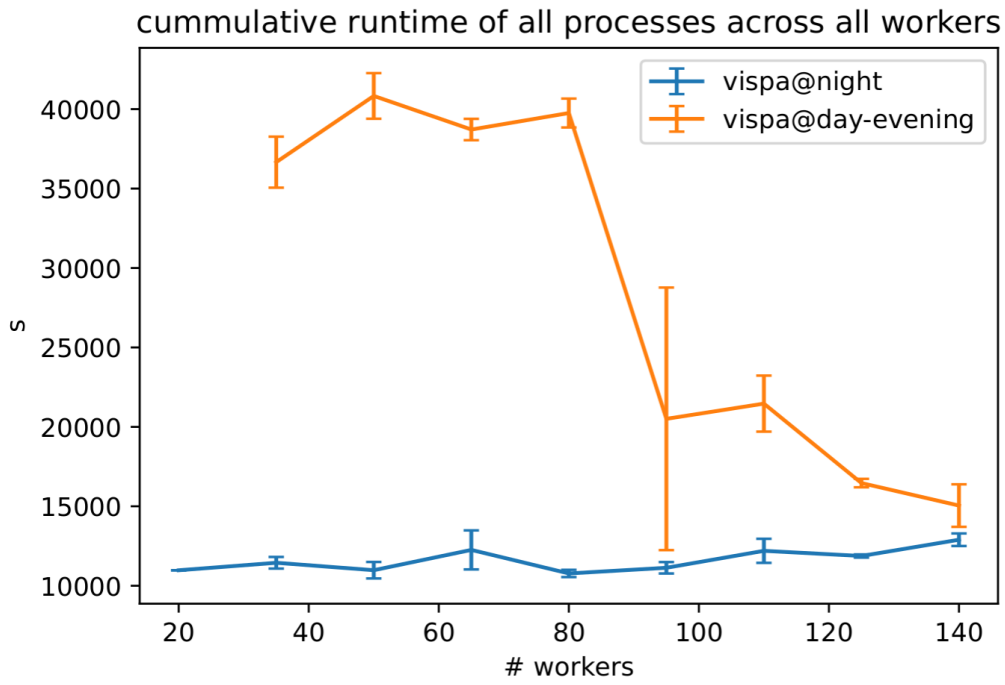


XCache load during two consecutive runs with 50 workers



Dask dashboard @ Vispa





Runtimes @ Vispa during the day and night