GPU work at Glasgow

Benchmarking + Interactive Compute

Albert Borbely (Bruno) albert.borbely@cern.ch

04/10/2024

A step towards GPU benchmarking

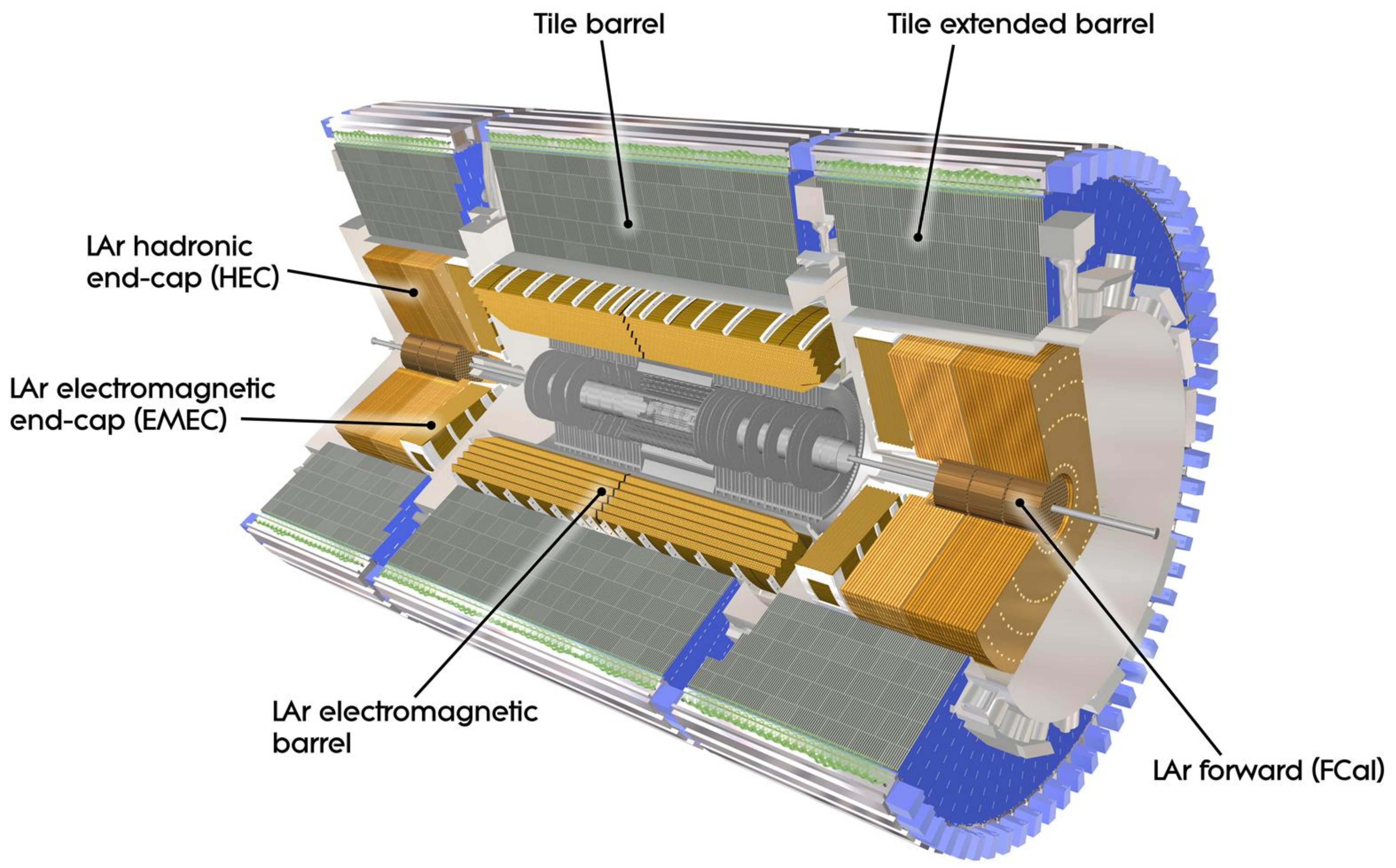
Using Celeritas ATLAS Tile calorimeter test run

- The Celeritas project is aimed at developing GPU-based Monte Carlo simulations in HEP

- Currently focused on EM physics e.g. the ATLAS Tile calorimeter

- I got in touch with the Celeritas team with the goal of setting up a benchmark

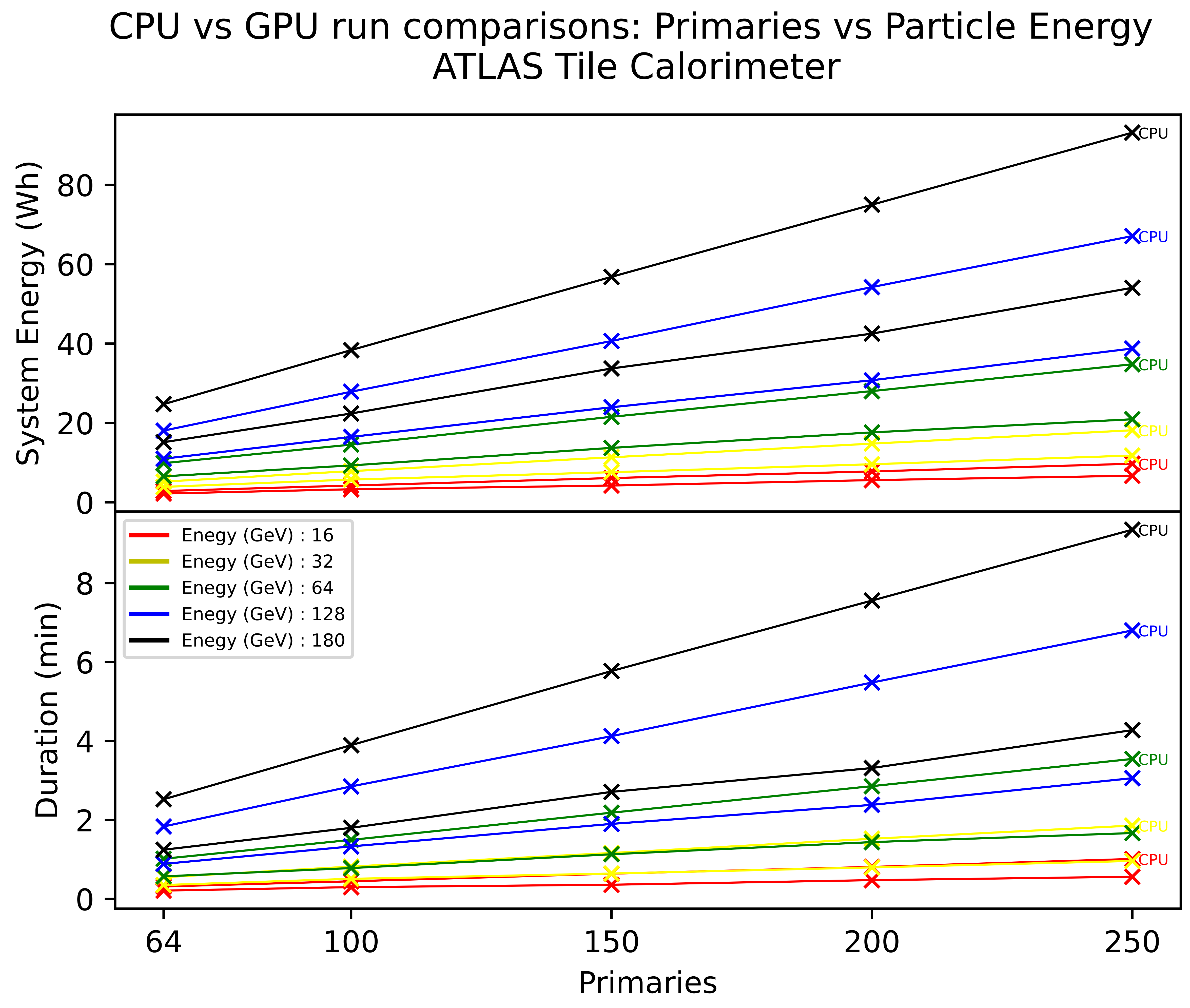
- Work is still on going and results are preliminary


Tile barrel
Tile extended barrel
LAr hadronic end-cap (HEC)
LAr electromagnetic end-cap (EMEC)
LAr electromagnetic barrel
LAr forward (FCal)

GPU benchmarking: step 1

CPU vs GPU comparison

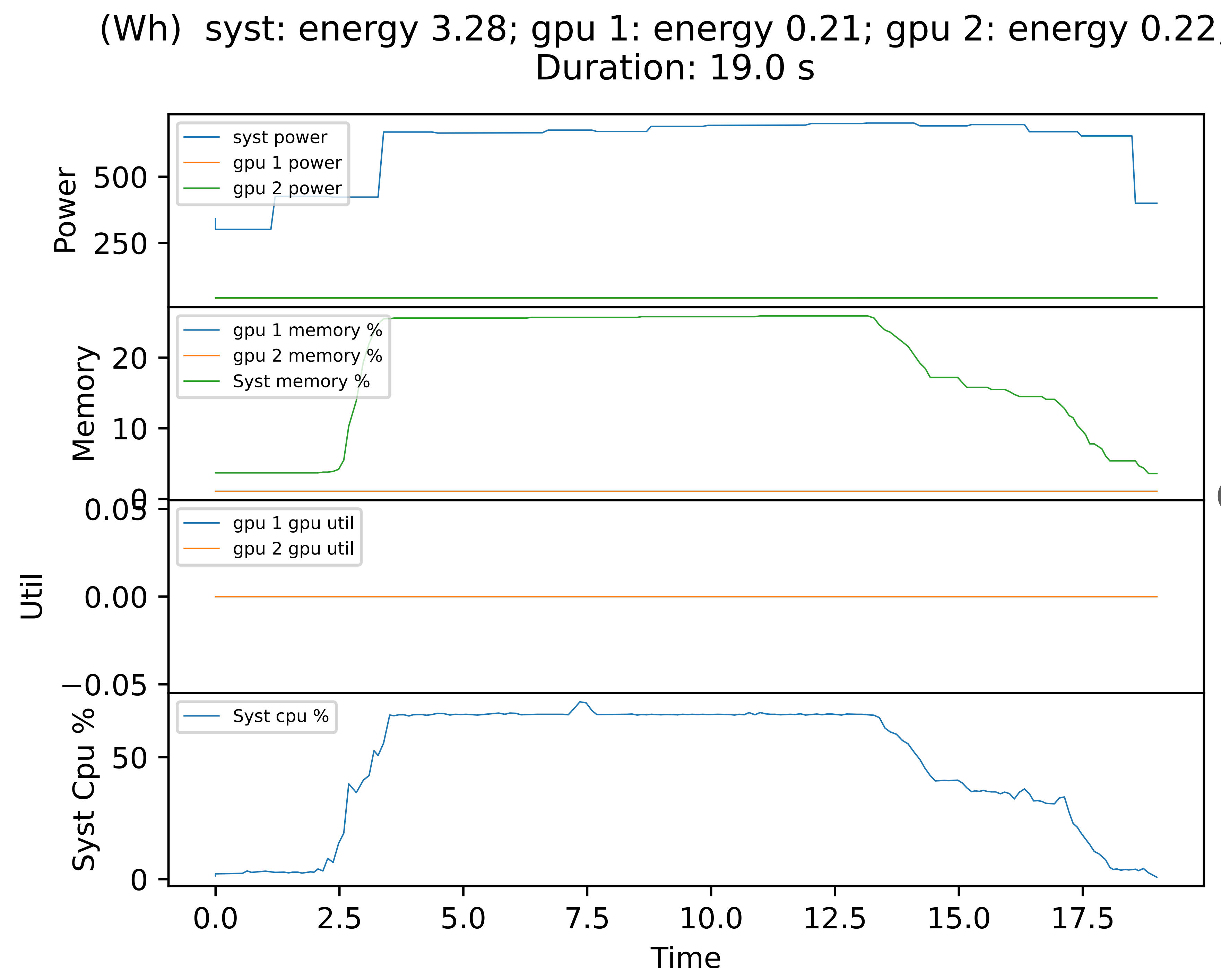
- Using the ATLAS Tile Calorimeter as a test geometry
- Using Celeritas in GPU and CPU mode
- 2 run parameters were varied:
    - Number of primaries
    - Initial particle energy
- The higher the parameters
  —> more intensive job
  —> more work offloaded to GPU
  —> greater reduction in duration/energy
- @ lowest (N64 & E16 GeV) ~ 22% & 33% decrease in job energy & duration respectively with GPU
- @ highest (N250 & E180 GeV) ~ 42% & 54% decrease in job energy & duration respectively with GPU
- Plenty of gains to be had :)



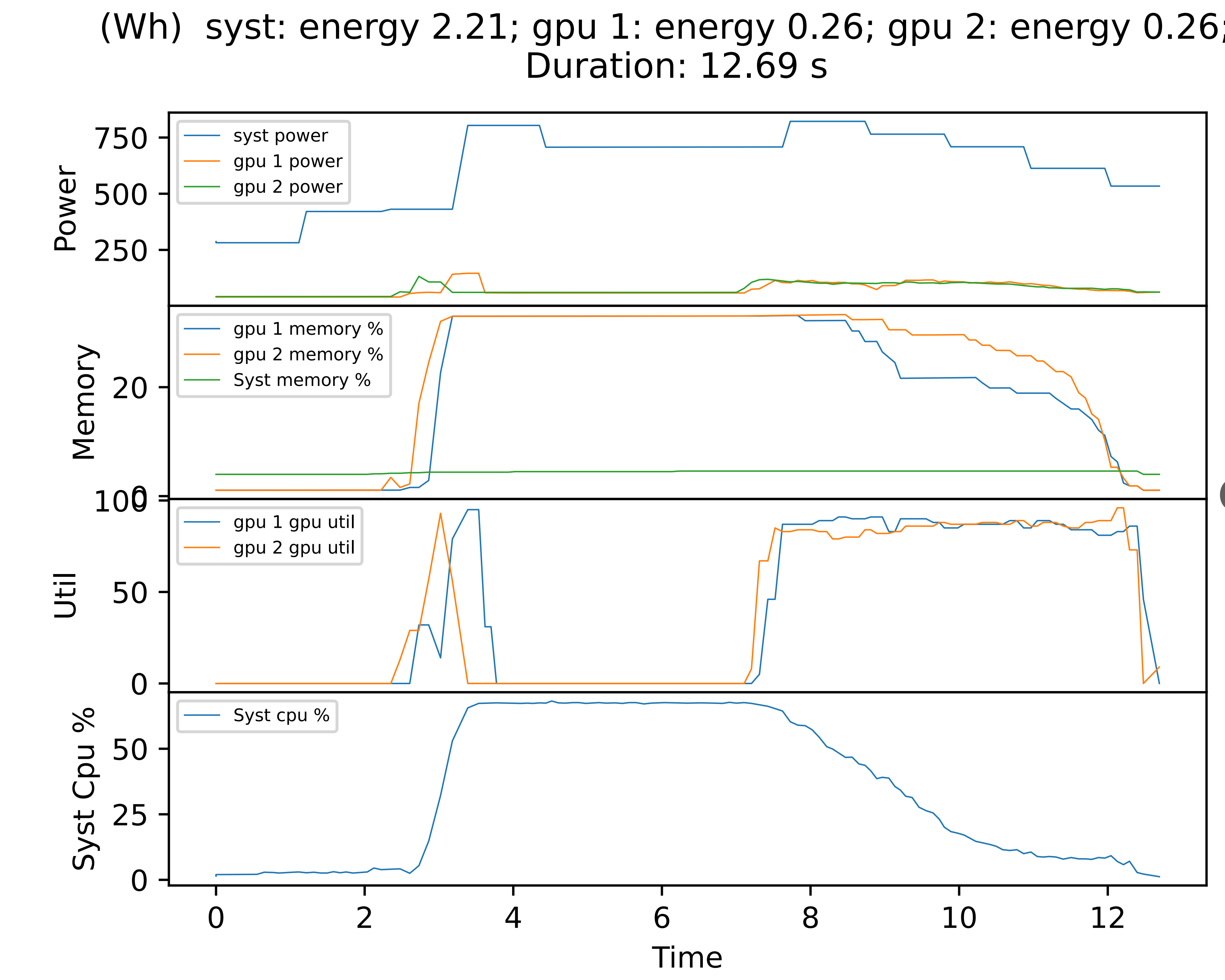
GPU benchmarking: step 2

Details

- CPU threads set to 32
- System Power gathered with IPMI
- GPU Power gathered with NVML
    - —> Some differences with sampling…
- CPU Energy ~ Syst. energy - GPU energy
    - Plan to physically pull out cards in the future
- System: GPU 2xa100 (80GB),
  CPU 2x AMD EPYC 7443 x48 cores,
  RAM 251 GB
- To keep things consistent 2 cpu jobs were launched in parallel as well as 2 gpu jobs (one targeting each card), so values shown are for two jobs in parallel in both cases
- GPU variant doesn't always maximise GPU utilisation —> need to think about CPU/GPU —> to maximise GPU utilisation
- All jobs being launched in docker containers, to deal with dependancies, environment and installation. Also makes GPU management easier.

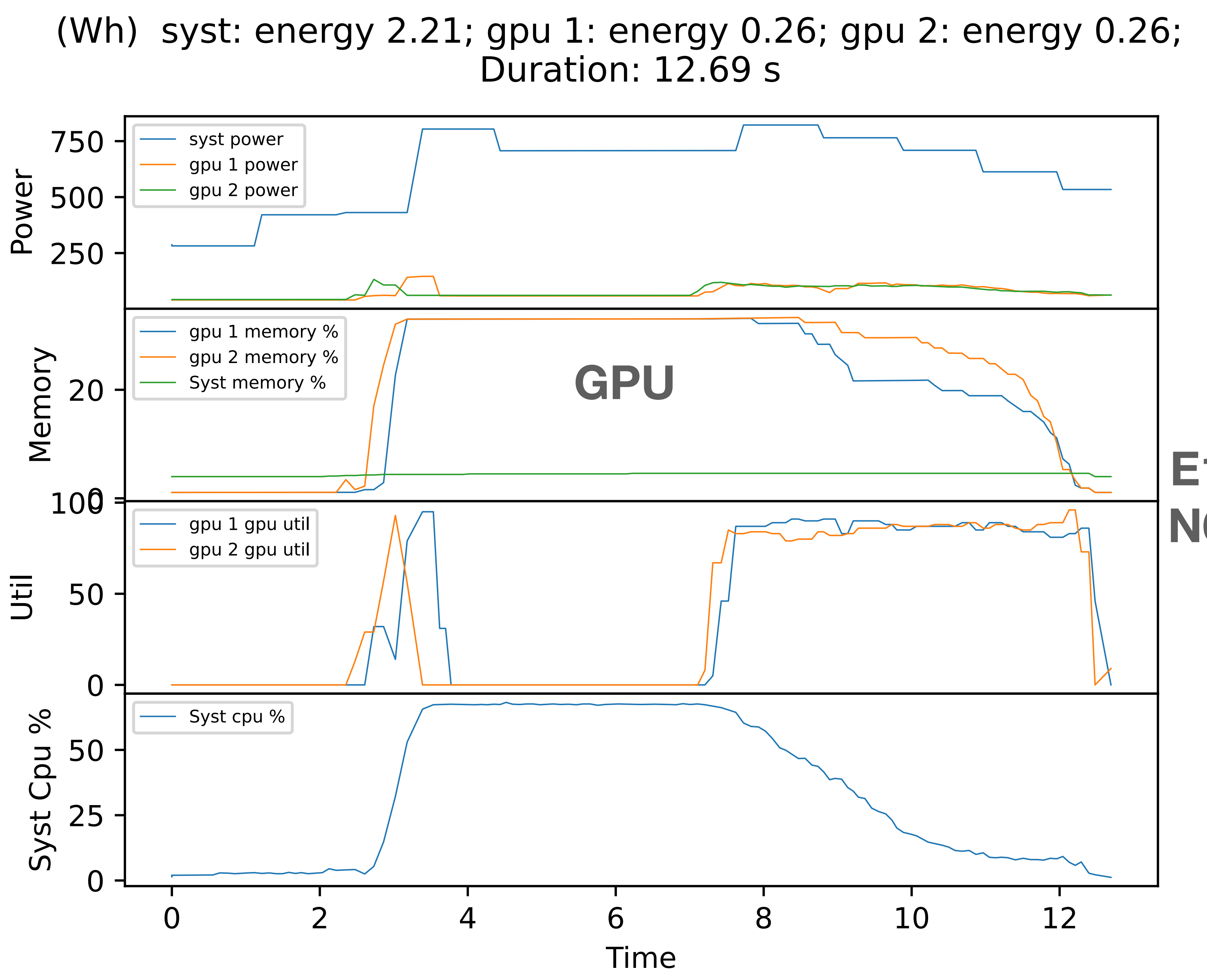


CPU

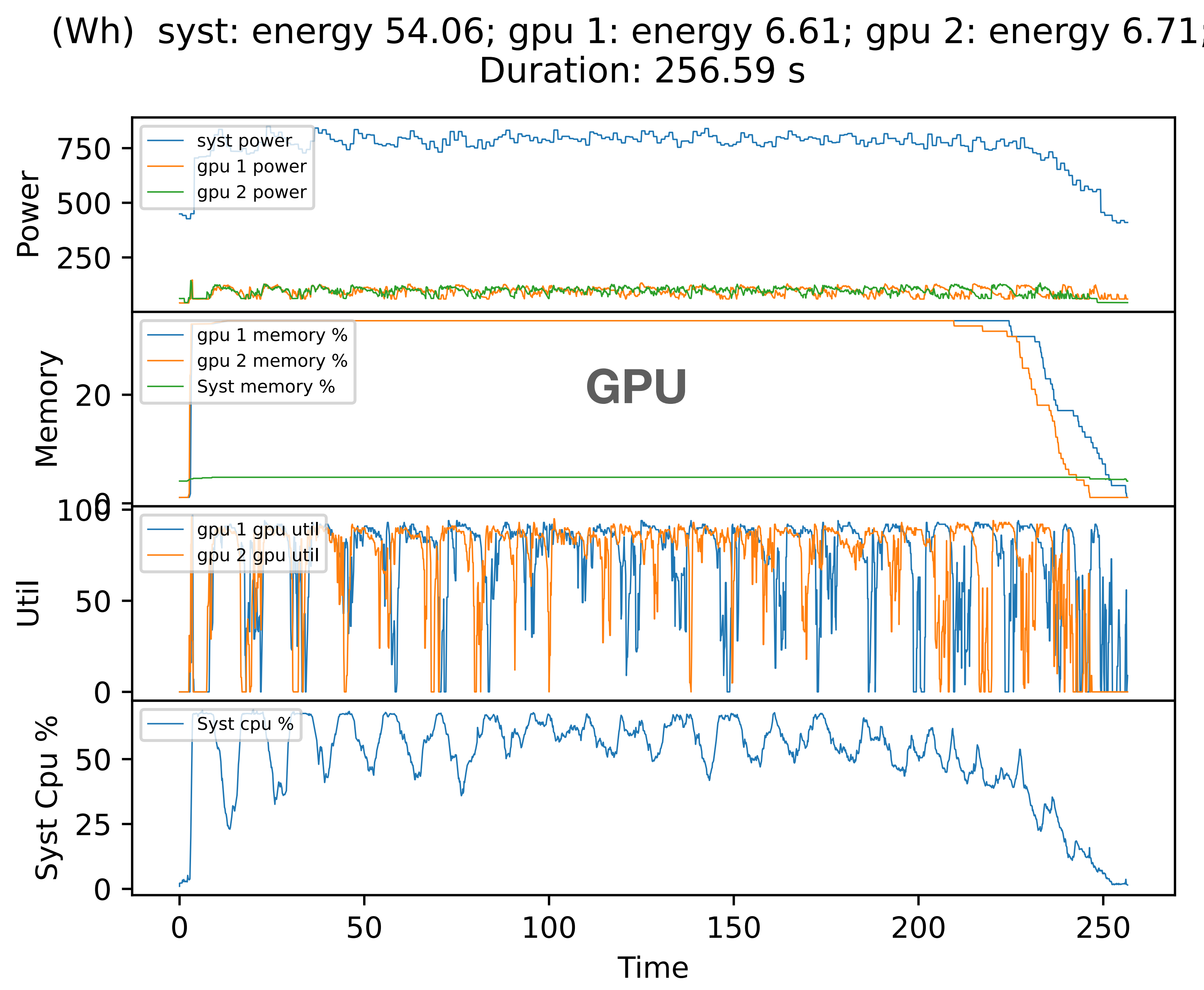


GPU

GPU benchmarking: step 3

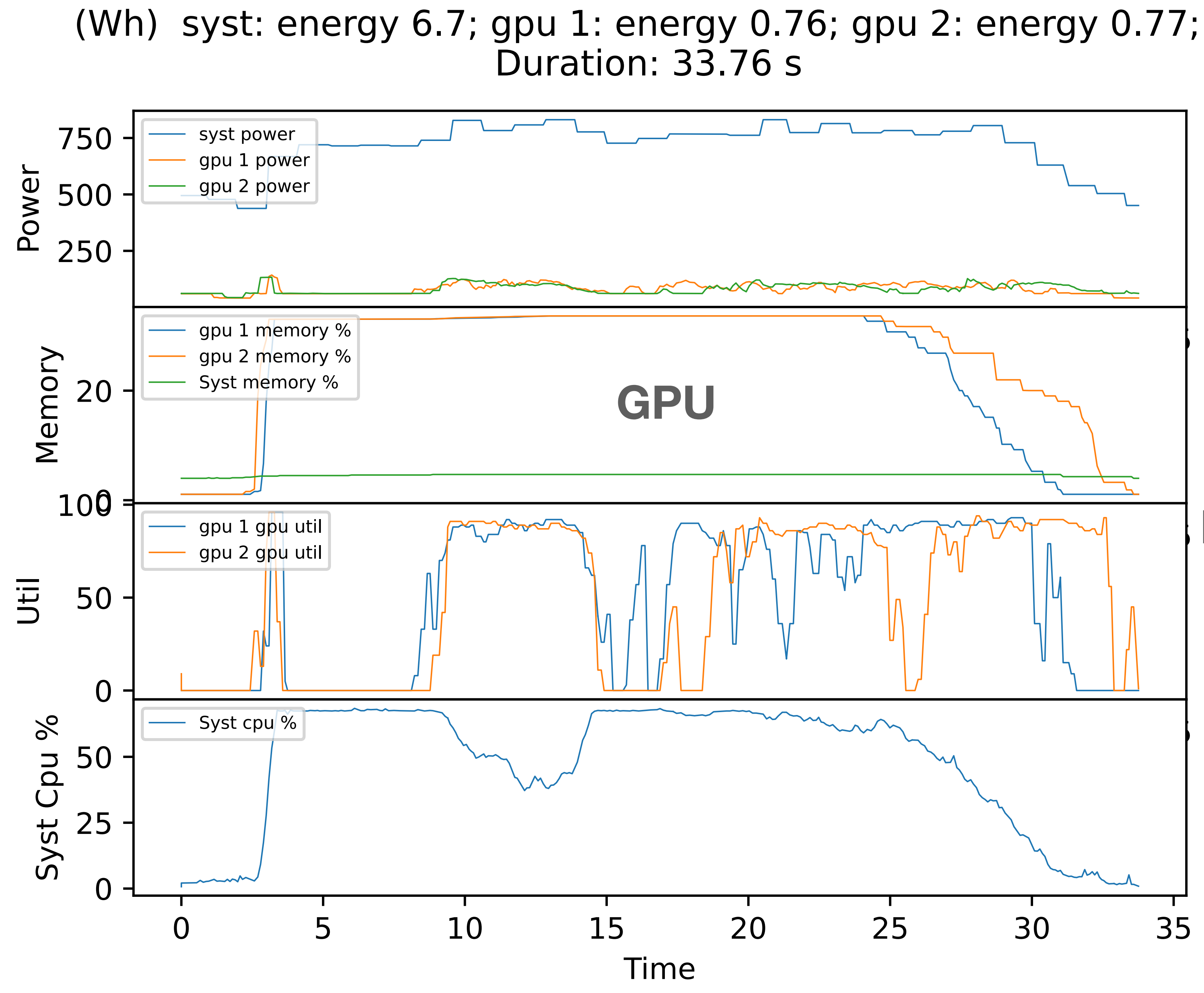


E16
N64

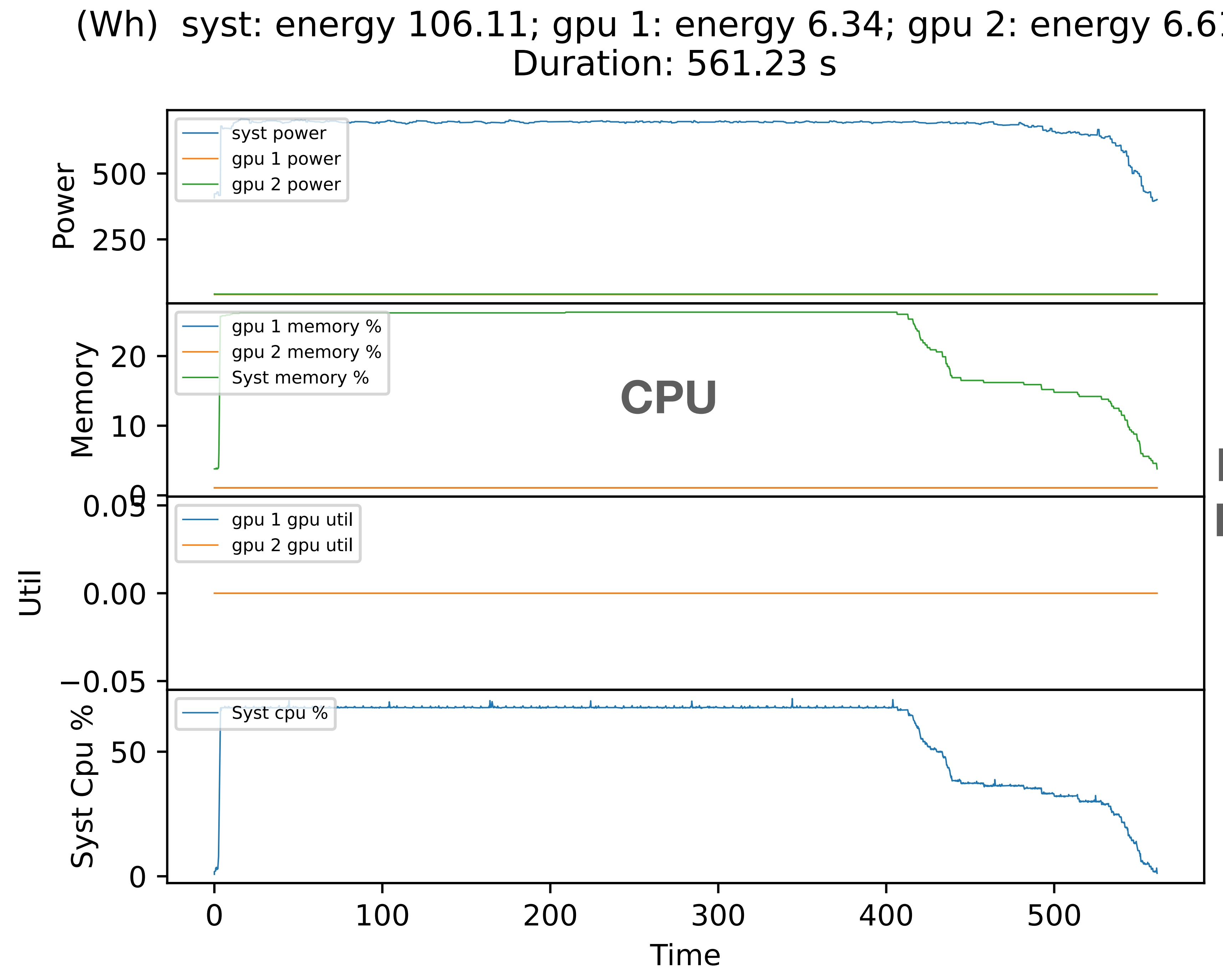


E180
N250

- Job “work” heavily depends on parameters used
- CPU variants effectively use a constant fraction of CPU resources (32 threads per job)
- GPU variant is constantly offloading parts of the job to the GPU
    - —> This causes fluctuations in GPU utilisation (see left/top-right plots)
- Need to think about how to maximise GPU utilisation, often most expensive part should not be sitting idle
- Current setup allows a job to hog 1 GPU
    - —> Potential solution to allow multiple job slots to share a GPU resource



E16
N250



E180
N250

GPU benchmarking

Acknowledgments and Future steps

Acknowledgments:

- Many thanks to the Celeritas team for getting me started and dealing with my emails/slack messages
- In particular to:
  - Ben Morgan, email
  - Seth R. Johnson, email
  - Julien Esseiva, email

Future steps:

- Physically pull out GPU cards for CPU run
- Attempt to run multiple jobs in parallel to maximise GPU/CPU utilisation
- Test GPU MIG / Compute Instances for multiple process use of GPU
- Attempt to define an event "throughput" per node
- Compile and run on Grace + A100
- Run the CMS variation of the job
  - —> more complex test geometry
  - —> test new geometry definition format

On-Grid Interactive GPU development

Many problems to solve

Problems vaguely fall into 2 categories:

- GPU problems
- Variety of cards (what to target)
- Variety of software tools
- Large dependancy issues
- Because of their nature user jobs tend to be small enough to not need to scale out to the Grid
- The idea is to facilitate on-Grid development and reduce the overhead in submitting Grid jobs via a submission engine

- Interactive job develop problems (Analysis Facilities?)
- User authentication
- Flexible development environment
  - —> allow users to install packages
  - —> maintain site security
- Data storage / integration
- Scalability, i.e. easily scale out to the rest of the Grid

Authentication

User connection Strategy

- Currently authenticated with x509 certificates (for now)
- A user can request an interactive job via our test ce: ce-test.gla.scotgrid.ac.uk
- Aimed at our GPU queue: (queue="condor_gpu")
- The user supplies an email, ssh key, and initiates an interactive job via the int_condor executable on the node.
- This then spins up a docker container and emails ssh instructions to the user
  - The node is not directly exposed to the internet
  - You have to login via an ssh proxy and target a specific port range on the node
  - This then lands you directly in the container with only basic user privileges
  - To add a layer of security only connections from institutes are currently accepted: Glasgow, CERN, DESY, Nikhev
  - This final step causes some issues as I can't configure another institutes machines / requires a certain amount of user competence to properly configure the user's ssh config
  - Looking to bullet proof this step by taking this out of the user's hands

Container Solution

Development Environment

- Docker vs Apptainer (Singularity) were tested for this
- Docker wins, especially for interactive GPU development use
- Apptainer essentially doesn’t allow for interactive containers and GPU interaction simultaneously
  - No matter what SUID variations you use
- Docker —> Apptainer conversion is straightforward
- Once development is over and a user wants to submit their job to the Grid it can be converted to Apptainer (the standard Grid tool)
- This allows for environment transportability

Transparent Data access

Connect with the Ceph cluster

- Ceph-FS partitions can directly be mounted with XrootD into a directory in the container, currently set up for Ligo VO (as we have no pre-existing data at Glasgow)
- This works for any type of file i.e. hdf5 ect. not just root files.
- Allows users to natively access files in a POSIX file system directly on the Ceph cluster, allowing to easily read/write. Care should be taken when doing I/O heavy operations.
- Draw backs: heavily relies on XrootD and Ceph-FS. Mounted instances have crashed before causing bizarre behaviour in the chain of mounted folders.
    - In the past the XrootD was running on the host system, this has now been moved in to the container
- ATLAS data is not stored in CEPH-FS so users have to rely on root files and their ability to stream over XrootD
    - Looking into alternatives to access other types of files ect. hdf5 without the need to effectively download them

Development Environment 2

Initial base packages

- Current plan for user defined docker images are implemented via DockerFiles on GitHub
- Each job builds a fresh docker image (not necessary) potential plans to implement a registry of sorts once user testing is further along
- Plans to have groups of users (not every individual) be able to submit custom docker images to the repo which would be merged in AFTER manual review —> limit attack surface
- CVMFS is available in each container
- Spack has proved a versatile package manager to install packages without privileged root permissions
- Python / pip will also be available

Current Issues

Ongoing

- Condor loses control of the docker process when the container is launched
- This results in the job not being killable via arc / condor
- If the container is stopped then the job is also shown as finished by arc
- Currently container stays alive for eternity (or until I restart the node)
- A way to stop the container and commit the changes and store / manage them planning to potentially use rucio for this
- A way to trim the interactive bits off the container (i.e. ssh daemon) for batch submission
- Currently investigating ways to share GPUs between users i.e. MIG slices and Compute Instances
- Currently all GPU instances are attached to a GPU condor slot

Next steps

Grid submission framework

- Once the development environment problems have been solved and tested
- Effort will be focused on the submission framework to make use of the wider GPUs available on the Grid i.e. the CMS trigger :)
- It will only use tools readily available in the Grid community i.e. apptainer
- It will involve moving data around to the relevant sites i.e. Rucio
- Trimming and converting the container into apptainer format
- Checking the sites various queues have available GPUs ect.
- Submitting and then gathering the required data.
- Ideally re-using the existing infrastructure already in-place with the addition of the custom container images.