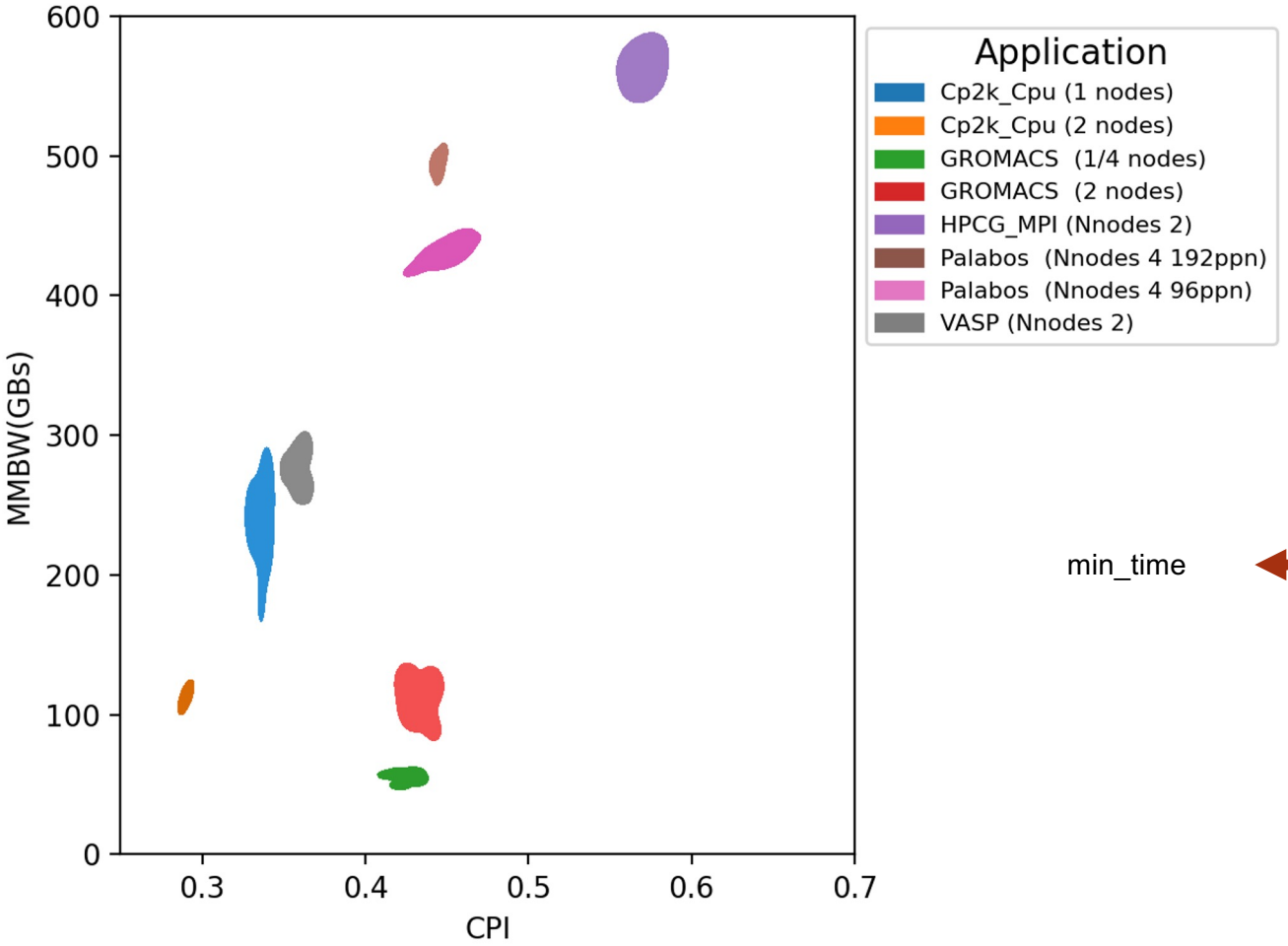
EAR ISC2024 tutorial: Advanced use cases and optimization

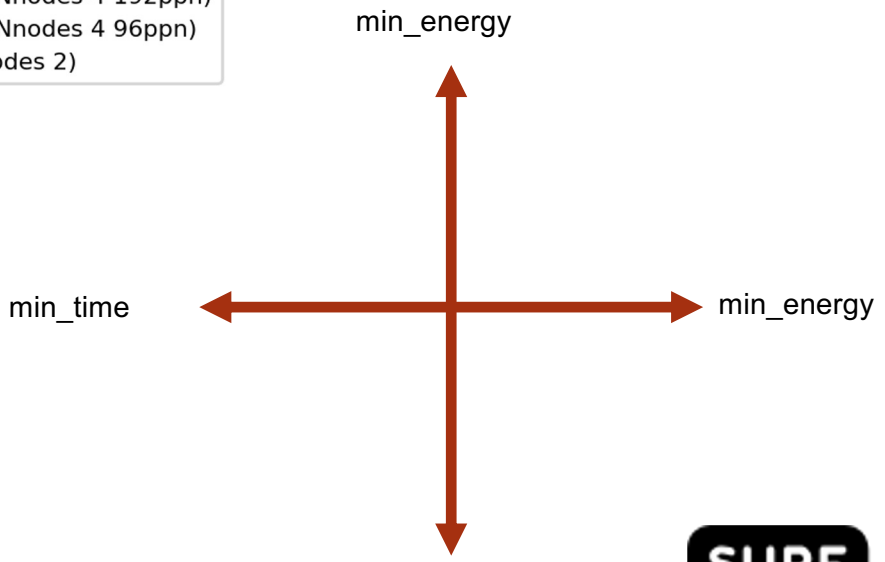
Julita Corbalan (julita.corbalan@eas4dc.com)
Benjamin Czaja (benjamin.czaja@surf.nl)

Advance use cases

- Codes in /projects/0/energy-course and GIT
- GIT: https://github.com/sara-nl/ISC-2024-EAR-tutorial/tree/main
- Get the examples and test them
  - https://github.com/sara-nl/ISC-2024-EAR-tutorial/tree/main/tutorials/monitoring_ear
  - GROMACS singularity
  - PyTorch
  - Palabos
- Data visualization
  - https://github.com/sara-nl/ISC-2024-EAR-tutorial/tree/main/tutorials/visualization
  - Grafana (using local installation)
  - ear-job-analytics tool

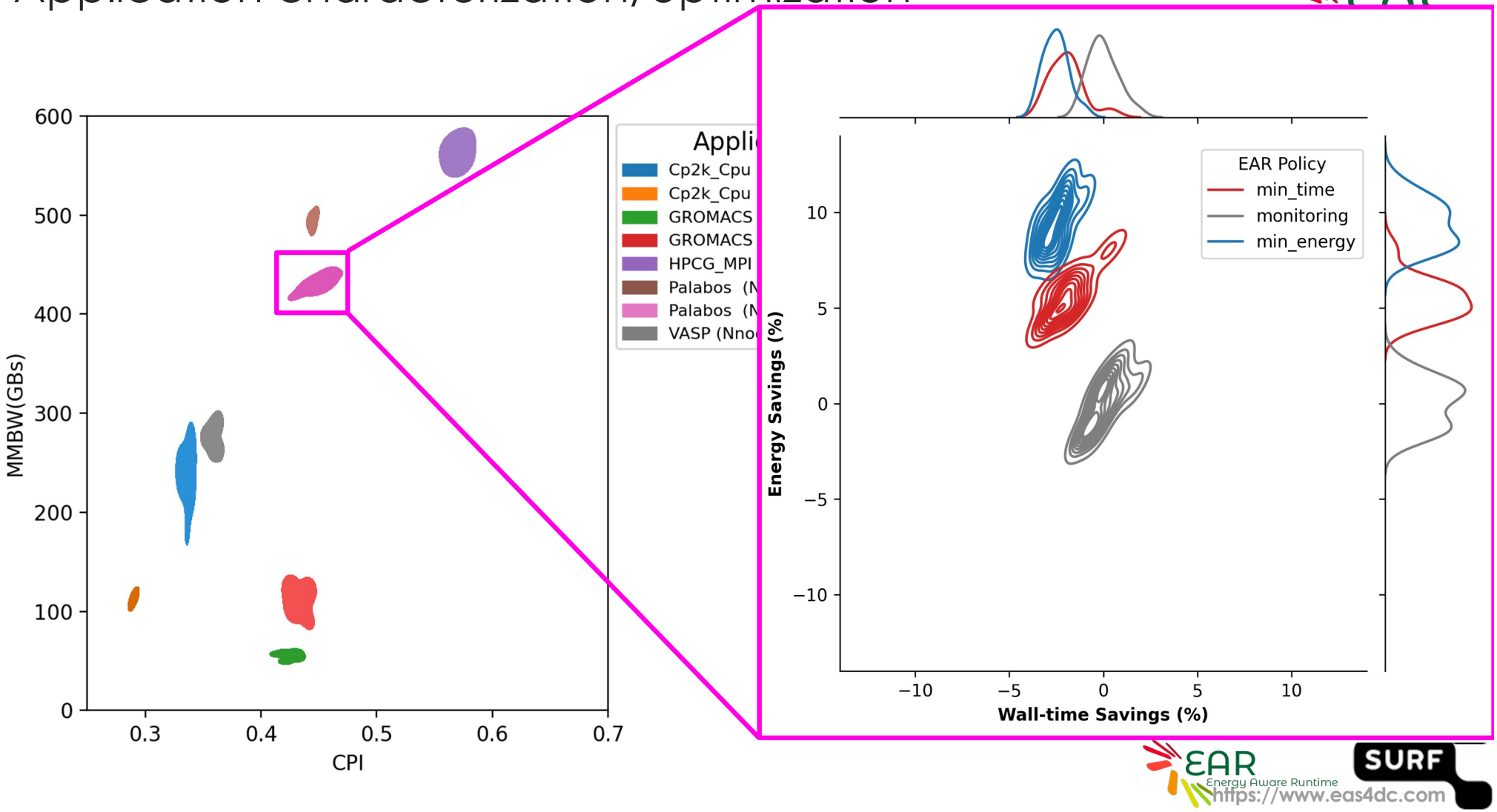
Application characterization/optimization





https://www.eas4dc.com

Application characterization/optimization



EAR Energy Aware Runtime
https://www.eas4dc.com
SURF

Use cases

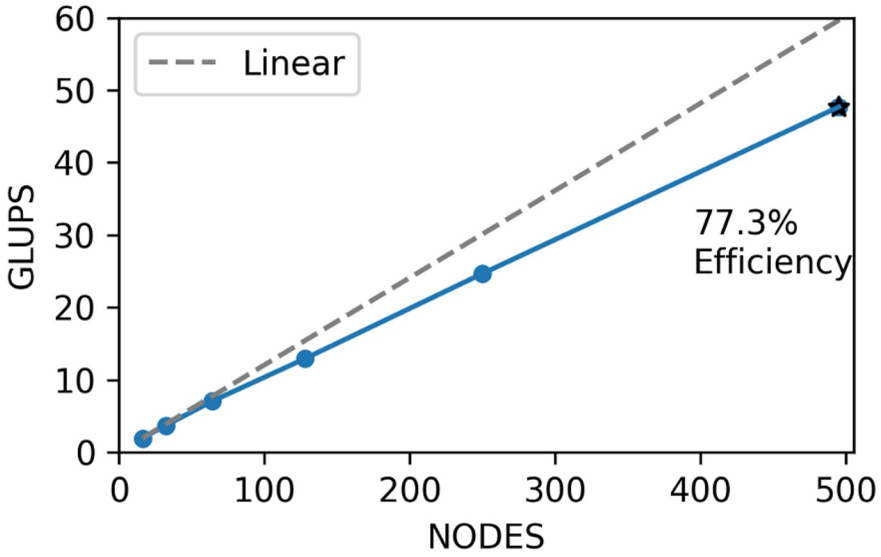
- Singularity
  - GROMACS (monitoring)
- CPU apps
  - Palabos: monitoring and dynamic optimization
  - NPB: dynamic optimization
- GPU apps
  - GROMACS singularity (monitoring)
  - PyTorch (min_energy)
- All of them, understand, visualize and compare energy efficiency when doing dynamic optimization

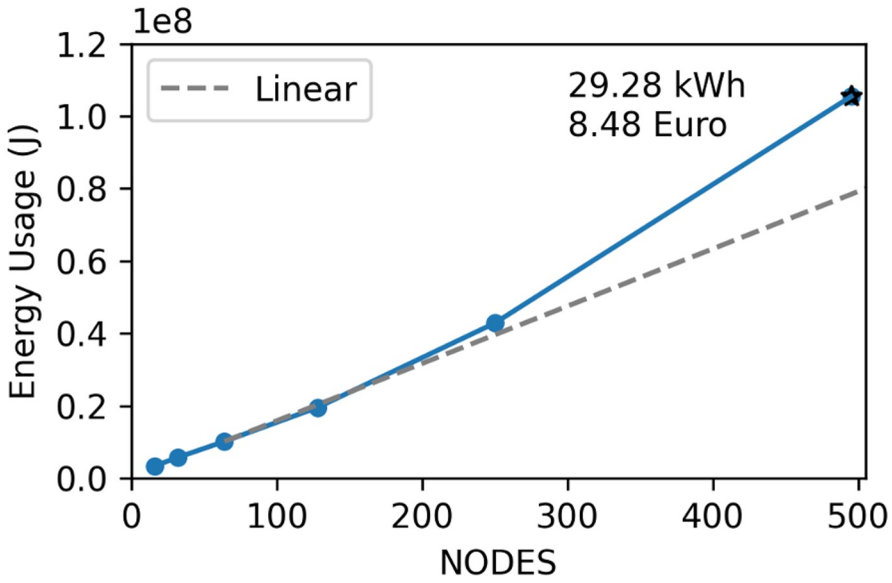
GROMACS-GPU: Singularity + EAR

- Singularity/Apptainer
  - https://apptainer.org/
  - https://docs.sylabs.io/guides/3.5/user-guide/introduction.html
  - https://catalog.ngc.nvidia.com/orgs/hpc/collections/nvidia_hpc/entities
- Singularity containers allow applications to use host services (such as ear)
  - Paths must be binded
  - Environment variables must be defined
  - https://github.com/sara-nl/ISC-2024-EAR-tutorial/blob/main/tutorials/monitoring_ear/GROMACS_SINGULARITY_GPU.sh

Palabos: Lattice-Boltzmann Solver

Strong Scaling Benchmark





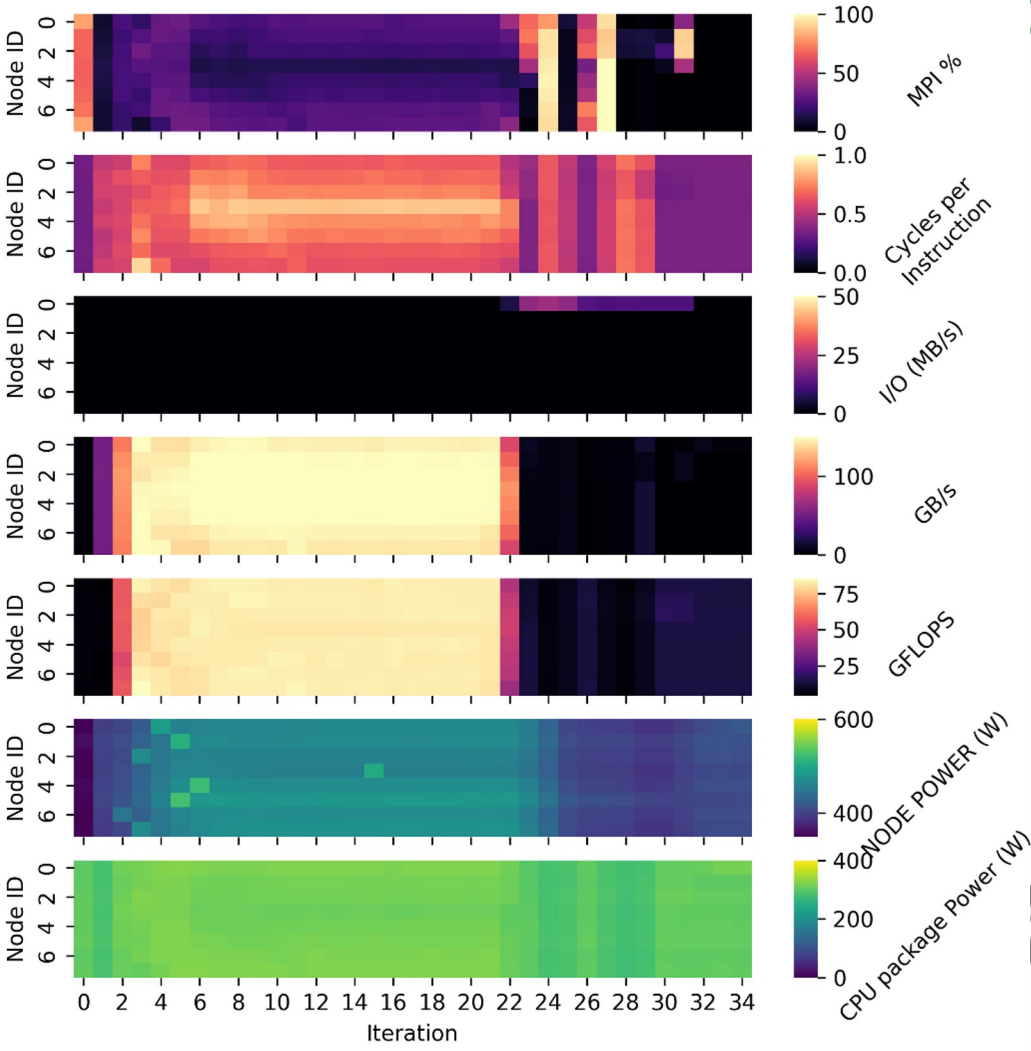
- An average electric car consumes about 0.2 kwh/km
- 495 node case (which ran for 9 minutes) just drove a car to Leeuwarden (149km from SP)!

Palabos:

Strong Scaling Benchmark

- Per-node, Per-iteration ”traces”

- Node Power
- Avg CPU Freq (node)
- Main memory BW (GB/s)
- CPI (Cycles per instruction)
- MPI% (percentage spent in MPI calls)
- I/O (network communication)

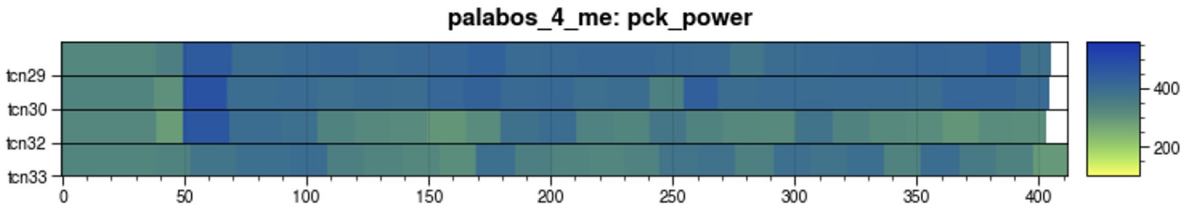


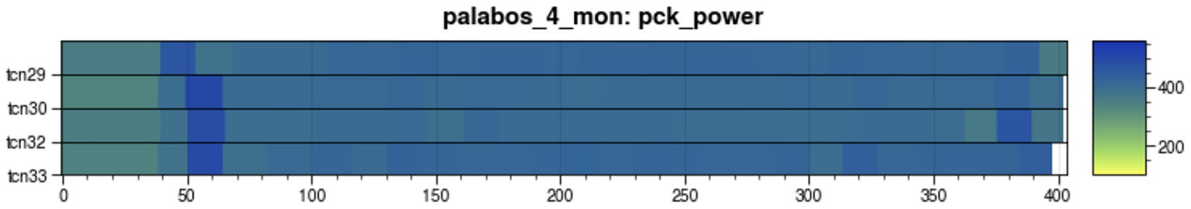
optimization

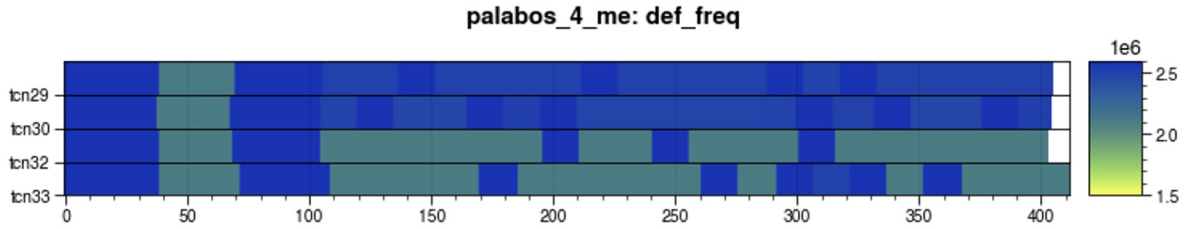
- Codes in /projects/0/energy-course
- GIT: https://github.com/sara-nl/ISC-2024-EAR-tutorial/tree/main
- Get the examples, add EAR **monitoring and min_energy** policy and compare
  - –ear=on → monitoring
  - –ear-policy=min_energy → selects min_energy policy
  - Run 2 steps in the same job to guarantee both runs are executed in the same node (s)
  - https://github.com/sara-nl/ISC-2024-EAR-tutorial/tree/main/tutorials/monitoring_ear
  - NPB : Rome vs Genoa
  - Palabos (use 1 and 4 nodes, node input_1_node_XL.xml input file)
    - Rome vs Genoa
  - PyTorch
    - Get some extra savings when running in exclusive mode
    - export EAR_JOB_EXCLUSIVE_MODE=1

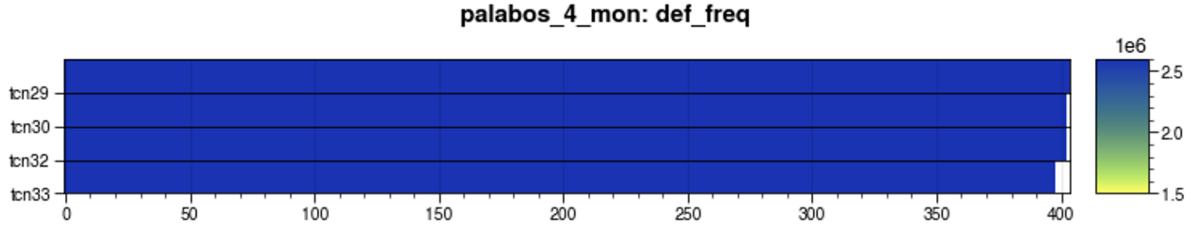
Data visualization

- https://github.com/sara-nl/ISC-2024-EAR-tutorial/blob/main/tutorials/visualization/README.md
- ear-job-analytics
  - Requires loops in DB: export EARL_REPORT_LOOPS=1
  - Use ear-job-anaytics directly or use create_traces.sh script
- Grafana
  - Running al DC and executing SQL queries (more powerful, but depends on DC)
  - Local installation:
    - Grafana server installed and running locally
    - Data gathered in CSV format with eacct and using CSV plugin
    - Not mandatory but more information if loops are in DB
    - Can be use also without EAR DB: –ear-user-db=filename


palabos_4_me: pck_power
tcn29
tcn30
tcn32
tcn33
0
50
100
150
200
250
300
350
400
400
200

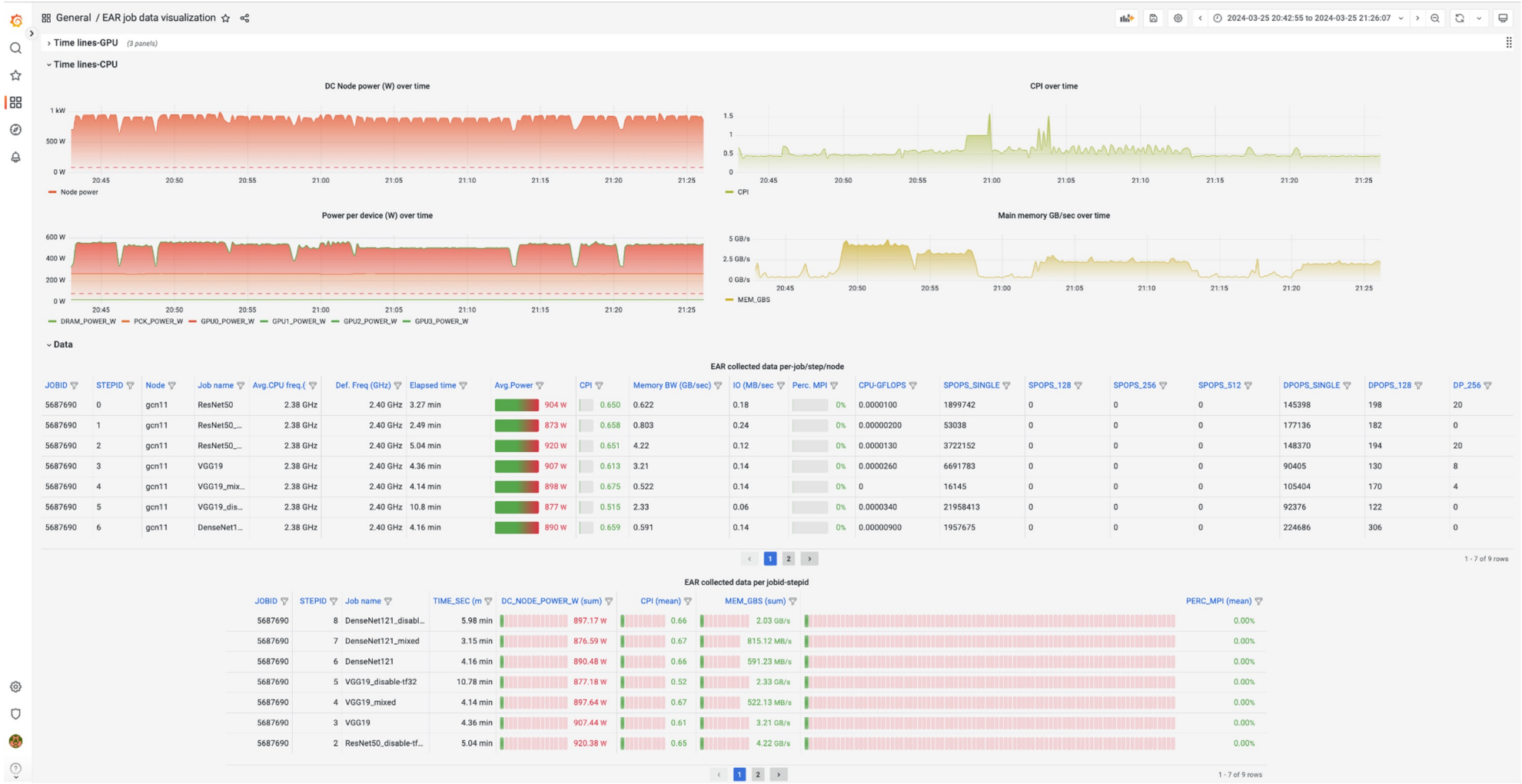

palabos_4_mon: pck_power
tcn29
tcn30
tcn32
tcn33
0
50
100
150
200
250
300
350
400
400
200


palabos_4_me: def_freq
1e6
tcn29
tcn30
tcn32
tcn33
0
50
100
150
200
250
300
350
400
2.5
2.0
1.5


palabos_4_mon: def_freq
1e6
tcn29
tcn30
tcn32
tcn33
0
50
100
150
200
250
300
350
400
2.5
2.0
1.5

Visualization with Grafana

- Steps to use EAR data are in grafana with CSV are
  - Have a local grafana installation with CSV plugins supported
  - Export EAR application data in csv format using eacct (-l -c option) or –ear-user-db flag
  - Export EAR application runtime data in csv format using eacct (-r -c option) or –ear-user-db flag
  - Add a source data based on a local file (Public folder)
    - julita.corbalan$ cp tensorflow.csv ear_data_apps.csv
    - julita.corbalan$ cp tensorflow_loops.csv ear_data_loops.csv
  - Import the EAR json file with the dashboards for data visualization
    - "EAR job data visualization.json"
  - Reload the dashboards


General / EAR job data visualization
2024-03-25 20:42:55 to 2024-03-25 21:26:07
Time lines-GPU (3 panels)
Time lines-CPU
DC Node power (W) over time
1 kW
500 W
0 W
20:45 20:50 20:55 21:00 21:05 21:10 21:15 21:20 21:25
Node power
CPI over time
1.5
1
0.5
0
CPI
Power per device (W) over time
600 W
400 W
200 W
0 W
DRAM_POWER_W PCK_POWER_W GPU0_POWER_W GPU1_POWER_W GPU2_POWER_W GPU3_POWER_W
Main memory GB/sec over time
5 GB/s
2.5 GB/s
0 GB/s
MEM_GBS
Data
EAR collected data per-job/step/node
JOBID STEPID Node Job name Avg.CPU freq.(Def. Freq (GHz) Elapsed time Avg.Power CPI Memory BW (GB/sec) IO (MB/sec Perc. MPI CPU-GFLOPS SPOPS_SINGLE SPOPS_128 SPOPS_256 SPOPS_512 DPOPS_SINGLE DPOPS_128 DP_256
5687690 0 gcn11 ResNet50 2.38 GHz 2.40 GHz 3.27 min 904 W 0.650 0.622 0.18 0% 0.0000100 1899742 0 0 0 145398 198 20
5687690 1 gcn11 ResNet50_... 2.38 GHz 2.40 GHz 2.49 min 873 W 0.658 0.803 0.24 0% 0.00000200 53038 0 0 0 177136 182 0
5687690 2 gcn11 ResNet50_... 2.38 GHz 2.40 GHz 5.04 min 920 W 0.651 4.22 0.12 0% 0.0000130 3722152 0 0 0 148370 194 20
5687690 3 gcn11 VGG19 2.38 GHz 2.40 GHz 4.36 min 907 W 0.613 3.21 0.14 0% 0.0000260 6691783 0 0 0 90405 130 8
5687690 4 gcn11 VGG19_mix... 2.38 GHz 2.40 GHz 4.14 min 898 W 0.675 0.522 0.14 0% 0 16145 0 0 0 105404 170 4
5687690 5 gcn11 VGG19_dis... 2.38 GHz 2.40 GHz 10.8 min 877 W 0.515 2.33 0.06 0% 0.0000340 21958413 0 0 0 92376 122 0
5687690 6 gcn11 DenseNet1... 2.38 GHz 2.40 GHz 4.16 min 890 W 0.659 0.591 0.14 0% 0.00000900 1957675 0 0 0 224686 306 0
1 2
1 - 7 of 9 rows
EAR collected data per jobid-stepid
JOBID STEPID Job name TIME_SEC (m DC_NODE_POWER_W (sum) CPI (mean) MEM_GBS (sum) PERC_MPI (mean)
5687690 8 DenseNet121_disabl... 5.98 min 897.17 W 0.66 2.03 GB/s 0.00%
5687690 7 DenseNet121_mixed 3.15 min 876.59 W 0.67 815.12 MB/s 0.00%
5687690 6 DenseNet121 4.16 min 890.48 W 0.66 591.23 MB/s 0.00%
5687690 5 VGG19_disable-tf32 10.78 min 877.18 W 0.52 2.33 GB/s 0.00%
5687690 4 VGG19_mixed 4.14 min 897.64 W 0.67 522.13 MB/s 0.00%
5687690 3 VGG19 4.36 min 907.44 W 0.61 3.21 GB/s 0.00%
5687690 2 ResNet50_disable-tf... 5.04 min 920.38 W 0.65 4.22 GB/s 0.00%
1 2
1 - 7 of 9 rows