

Compute Cluster Energy Optimization

Better Utilizing Clusters with Grid Production or Local User Jobs

Thomas Hartmann
DESY IT

Main Cluster Flavours

HPC vs. HTC

Job Types in the HEP Community

HTC and HPC clusters

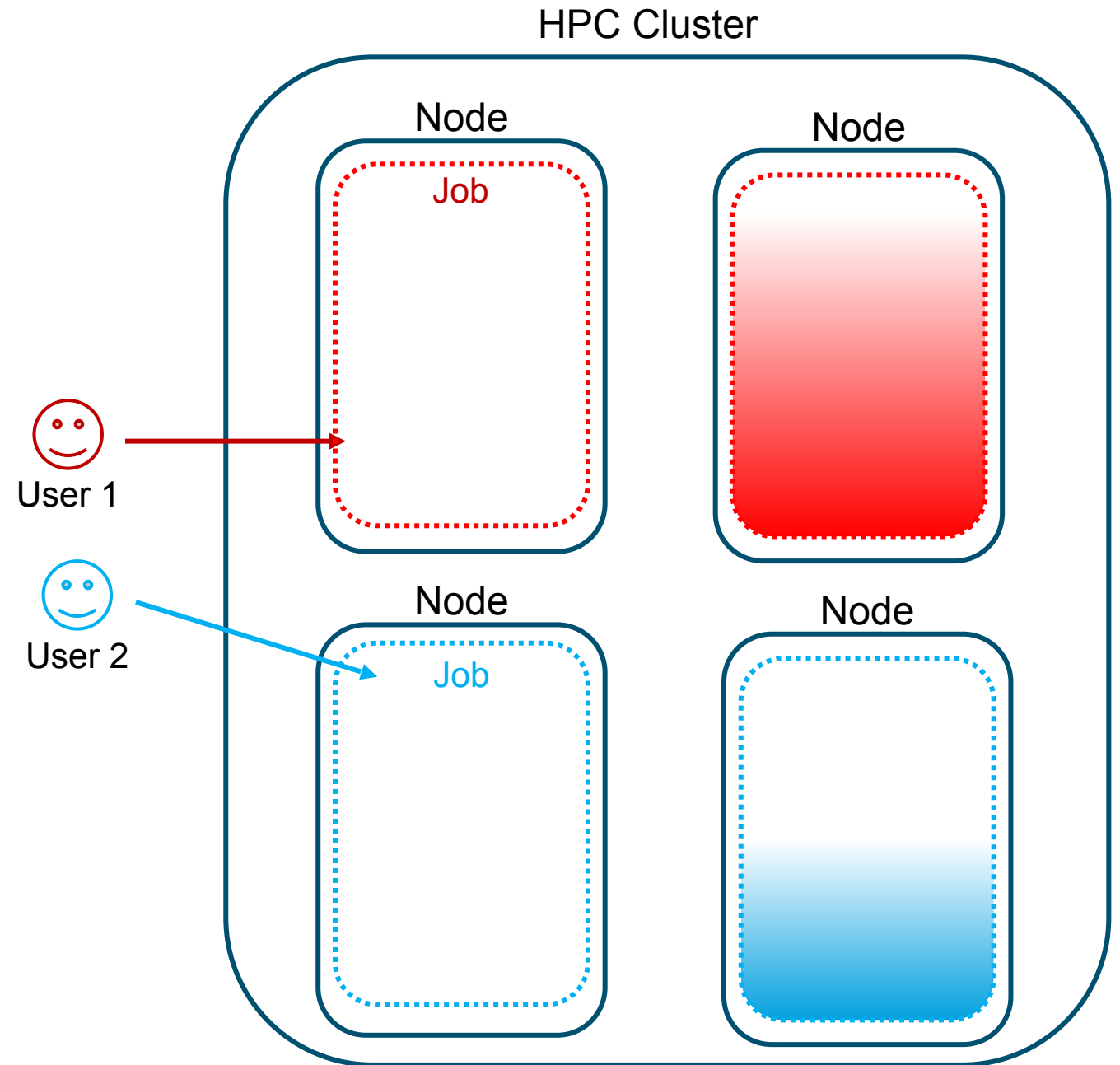
- CPU oriented scheduling
- **High-Throughput Clusters** prevalent
 - HEP events can be processed embarrassingly parallel
 - Optimal utilization of resources
- **High-Performance Clusters** opportunistically used by some groups
- “Memory Scheduling” like Apache Spark or DASK might become more complementary
 - Avoiding idling CPUs with Memory focused application might be an issue

HPC

HPC Cluster

Full Node Scheduling

- End users: direct submission to compute cluster
- Job scheduling per node granularity
- Full node available to a job
- Reasonable utilization left to the user/application
- Efficient utilization of resources not (necessarily) primary goal

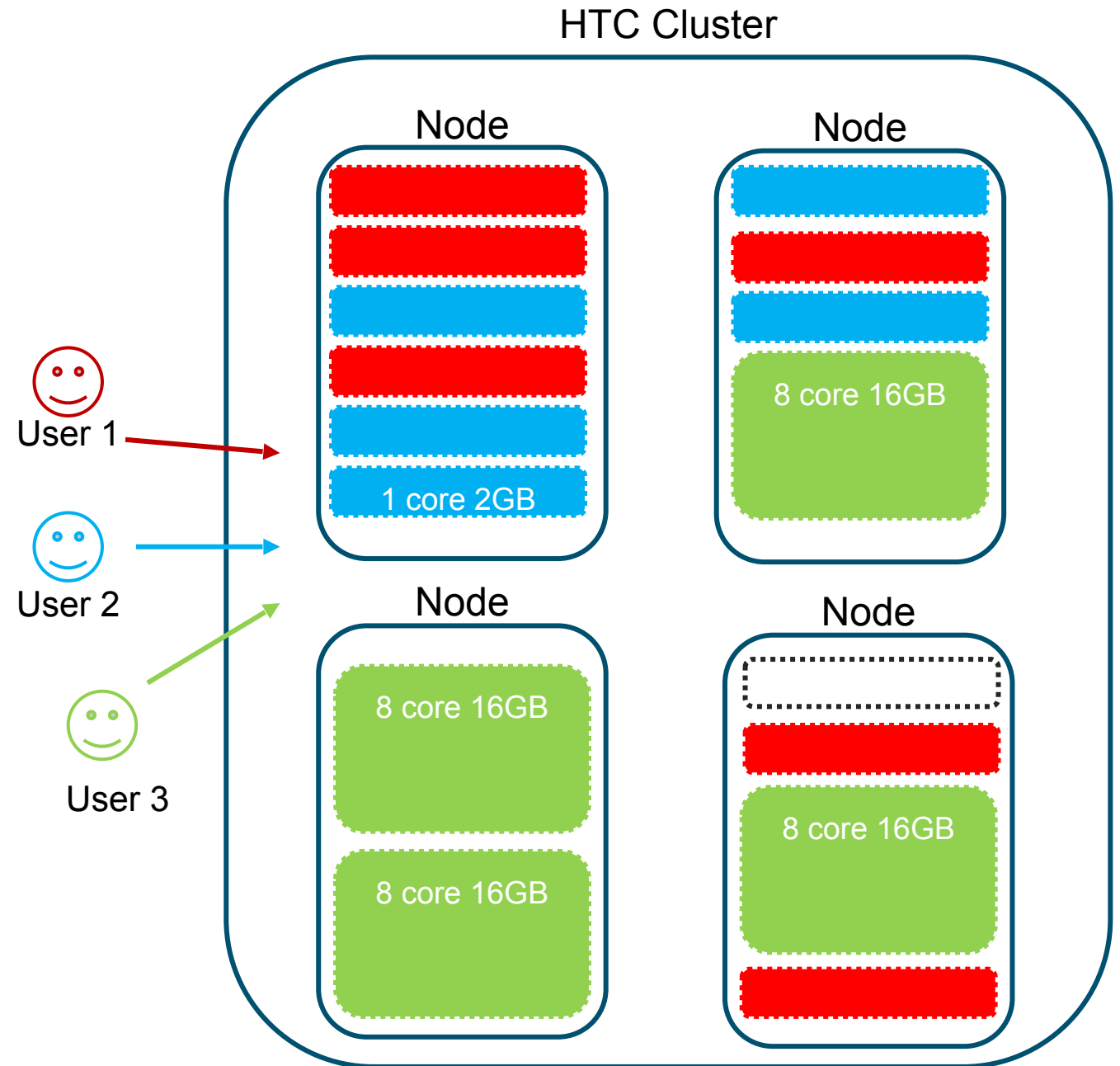


HTC

HTC Cluster

Granular Job Scheduling

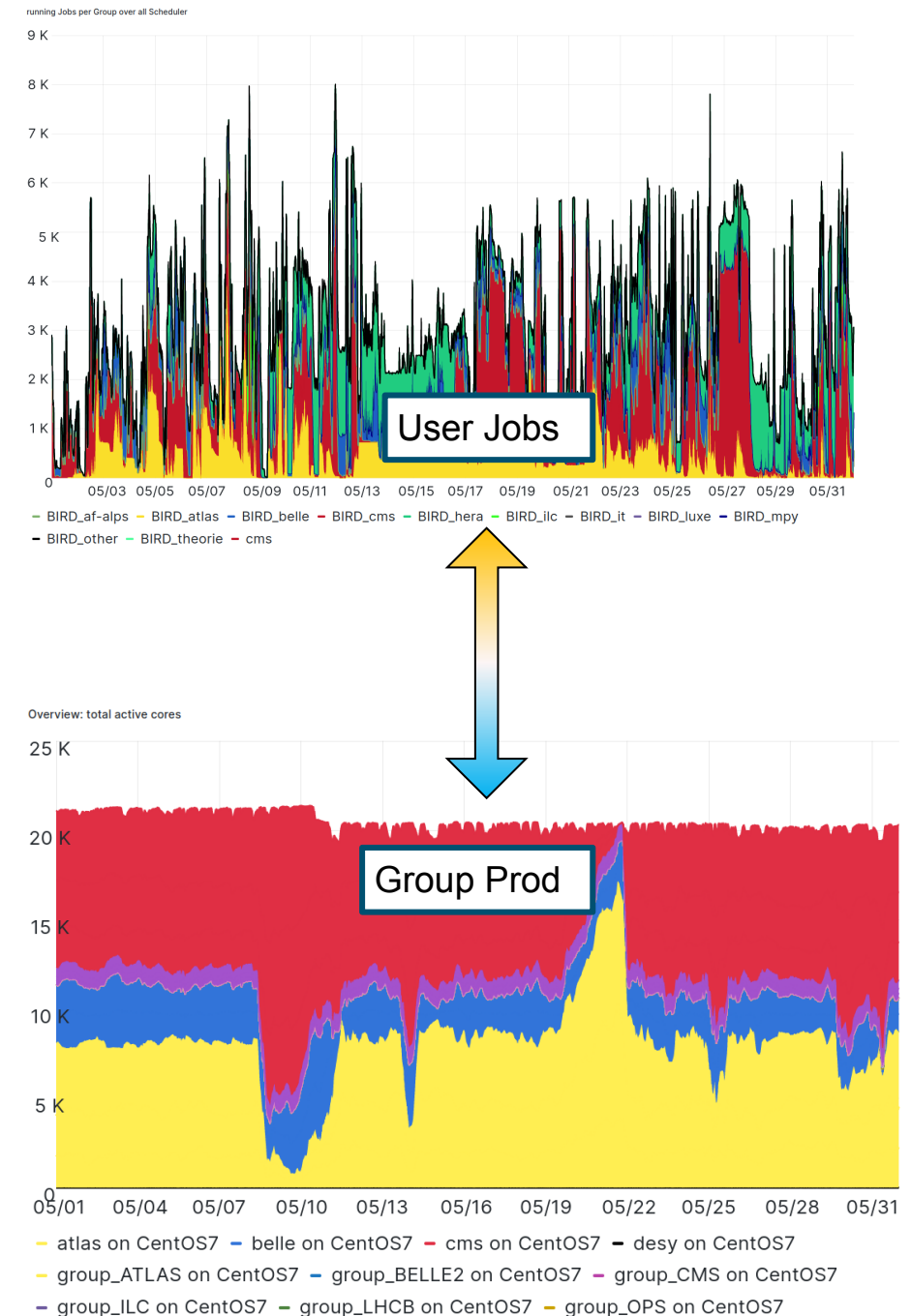
- End users: direct submission to compute cluster
- Jobs as requirements of subset of nodes, e.g., 1 core & 2GB mem
- Full utilization of clusters possible
- Job upstart latency can increase



HTC Clusters

User/Job Flavours at DESY

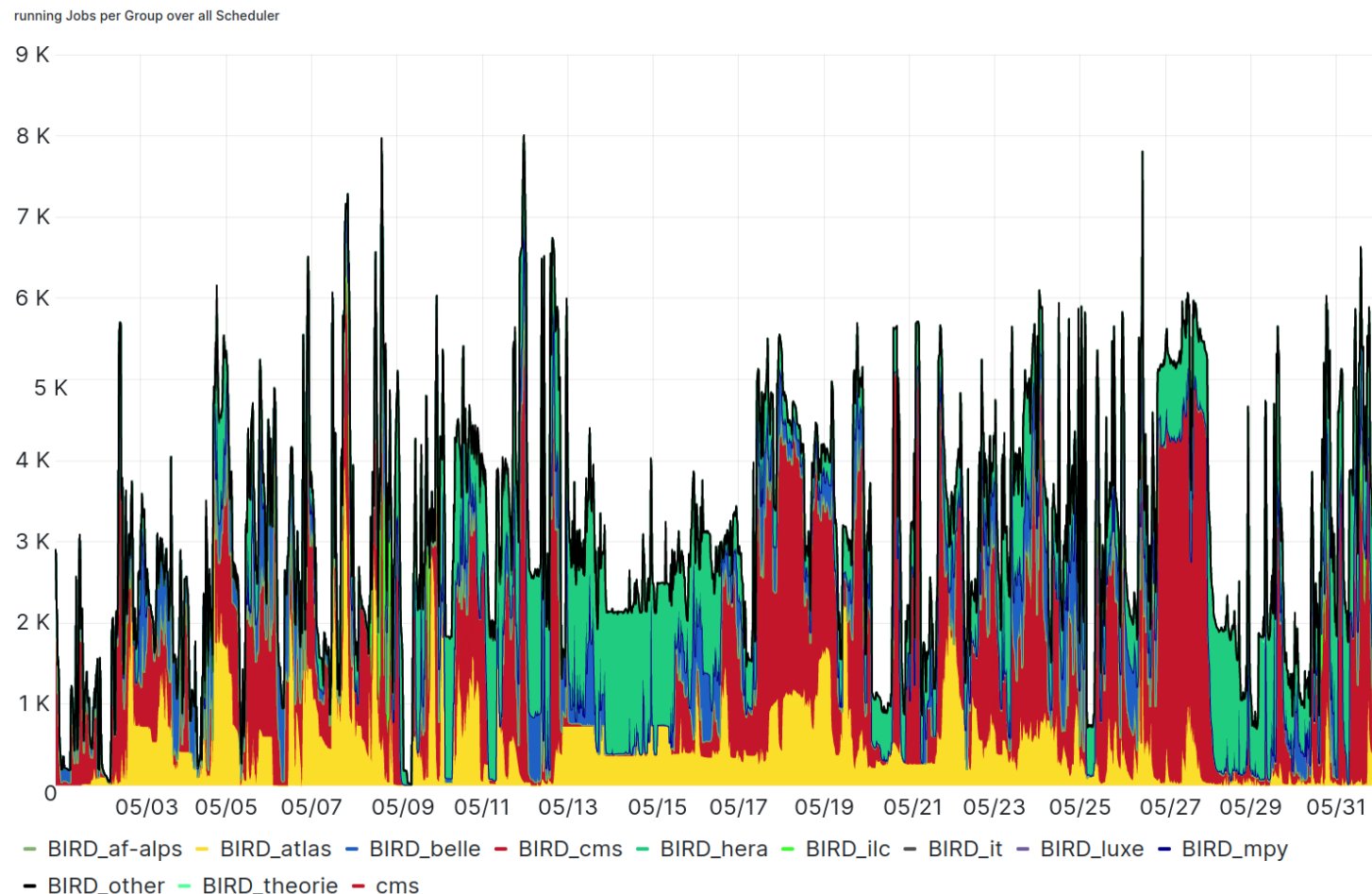
- 2 HTC clusters
 - User jobs: NAF
 - Group Production: Grid
 - Logical separated
 - Same code and admin base
- Differing workloads
 - Different energy saving options



User Cluster: National Analysis Facility

HTCondor Cluster for German HEP Users

- Individual users
- Utilization dynamic
- Overall utilization depends on work hours, holidays, ..., deadlines, conferences
- Job start up latency relevant for user satisfaction

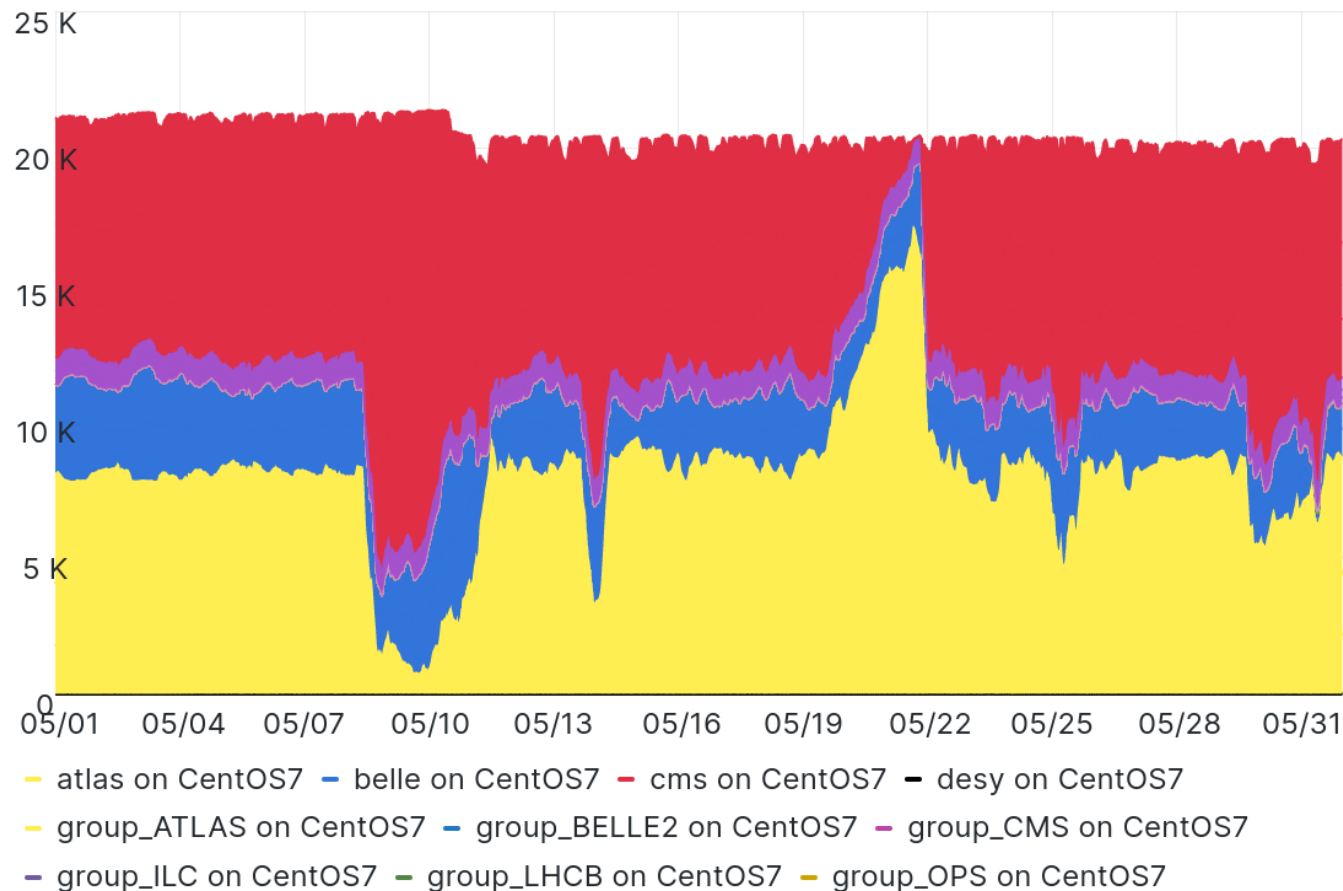


Grid Cluster at DESY-HH

HTCondor Cluster for HEP Communities

- Primarily HEP Groups
- Centralized pilot jobs
 - Group Production Payloads
- Goal: Full utilization 24/7/365
- Job start up latency not critical
 - Submission indirect via “CE”

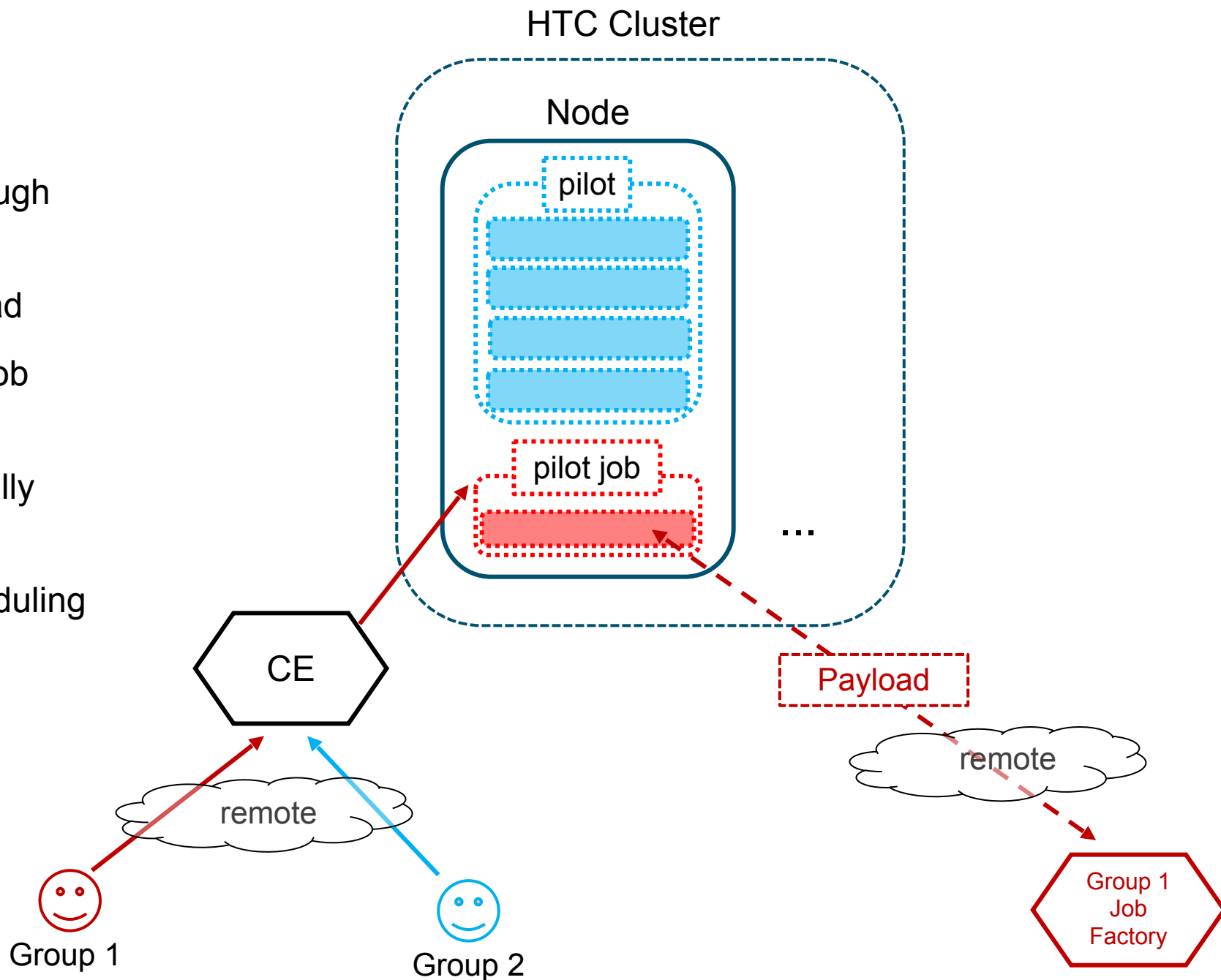
Overview: total active cores



HEP Grid Type jobs

Pilot/Payload jobs

- Remote (pilot) job submission through **Compute Element** gateways
- Pilot jobs not actual physics payload
- Actual payload pulled from group job factories & executed by pilot job
- Final payload run by pilots potentially unknown during pilot job start
- Timing dependent cluster job scheduling difficult

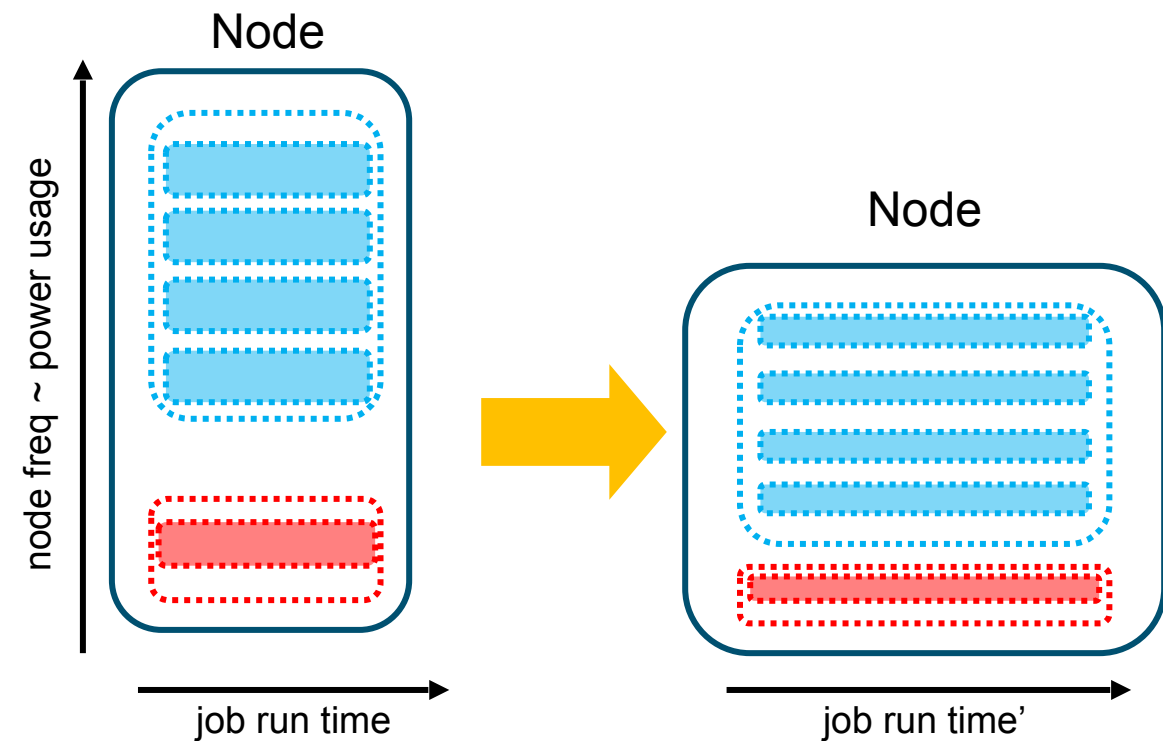


Cluster Power Shaping

Power Shaping per Node

Workload dependent Power Saving: Grid

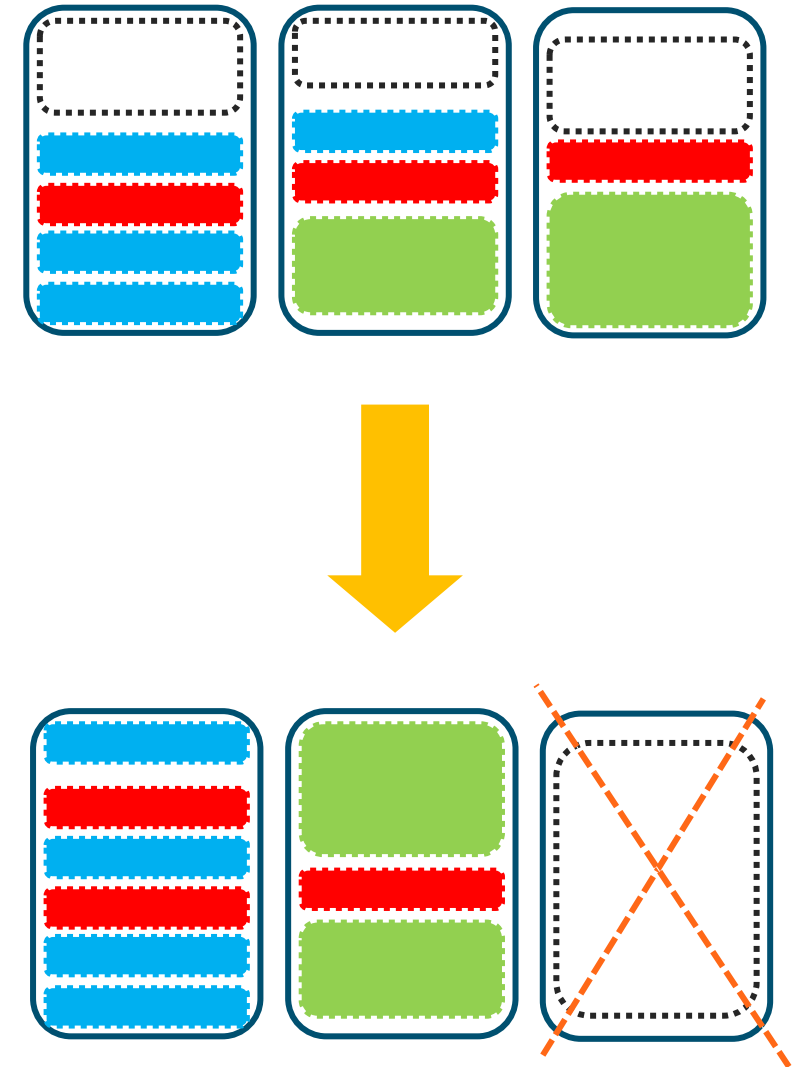
- Power consumption optimization depending on usage patterns
- Production Workload/Cluster
 - Job-Life-Time dependent scheduling difficult (payload run time potentially unknown to pilots)
 - Cluster external power shaping
 - Node/kernel power shaping transparent to payloads
- CPU Governor stepping driven by Green Energy availability



Cluster-wide Power Shaping

Workload dependent Power Saving: Users

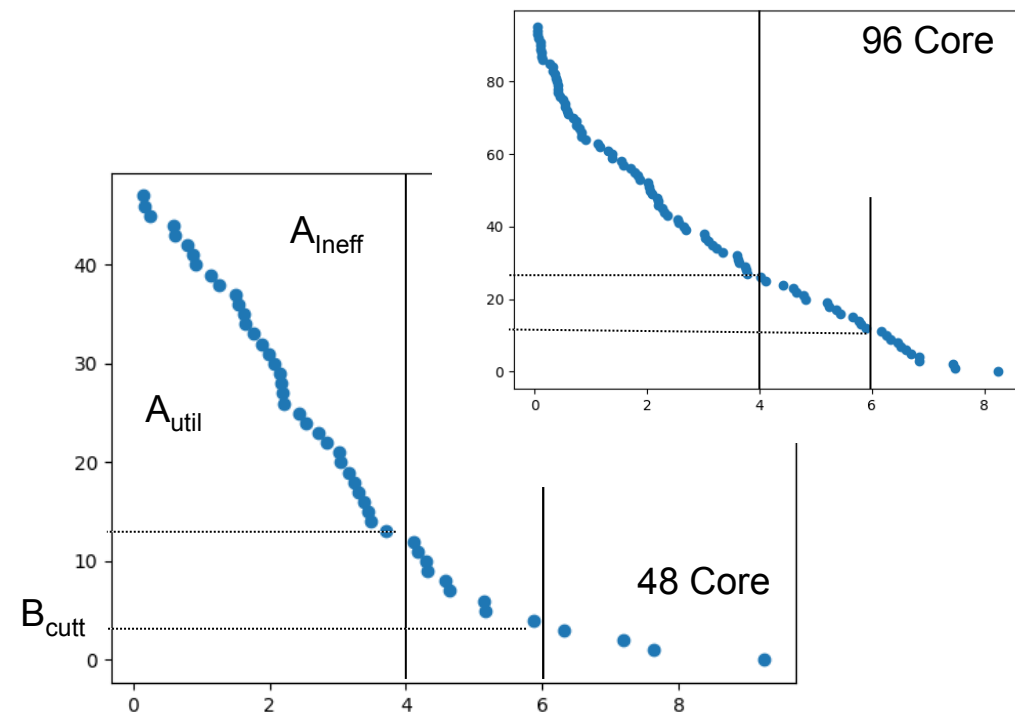
- User Clusters with more dynamic utilization
 - Potentially higher job entropy
 - Cluster intrinsic power shaping
- Horizontal vs. vertical scheduling
 - Cluster *compression*
 - price: higher job upstart latency/entropy
 - More aggressive node shedding
 - Opportunistic node ramp up with backfill workloads on standby



Preemption: Job Shedding minimizing Cycle Waste

User Side Implementation Necessary

- Draining Cluster/Nodes
 - Wasting idle CPU cycles
- Hard Node shedding
 - Wastes all CPU cycles so far of active jobs
- Ideally: Pre-emptable Jobs
 - Grace Period SIGTERM → SIGKILL
 - Snapshot/Stage results so far
- Requires: User Side Implementation...



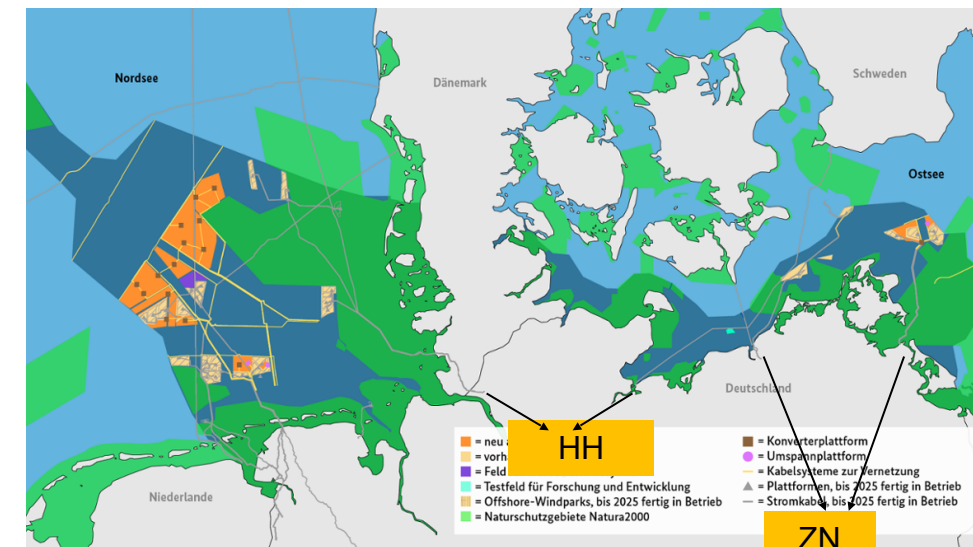
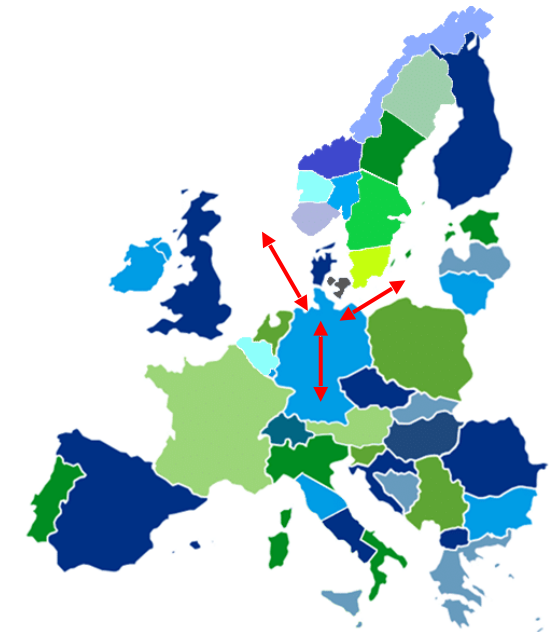
Simulation: Node Utilization while Draining

Green Energy Driven Computing

Cluster Power Shaping

Positive/Negative external energy incentives

- **Positive Incentive:** Surplus of green energy, i.e., depressed prices
 - Job fan out to opportunistic resources
 - Opportunistic green energy sink with opportunistic computing
 - Offshore wind farms (general weather patterns)
 - Photovoltaics (night/day cycle + general weather patterns)
 - limited transport capacity to southern Germany
- **Negative Incentive:** Lulls of green energy resources
 - CPU frequency throttling
 - High responsiveness, smaller savings (baseline power consumption)
 - Node shedding
 - Low responsiveness, larger savings (depending on draining efficiencies)

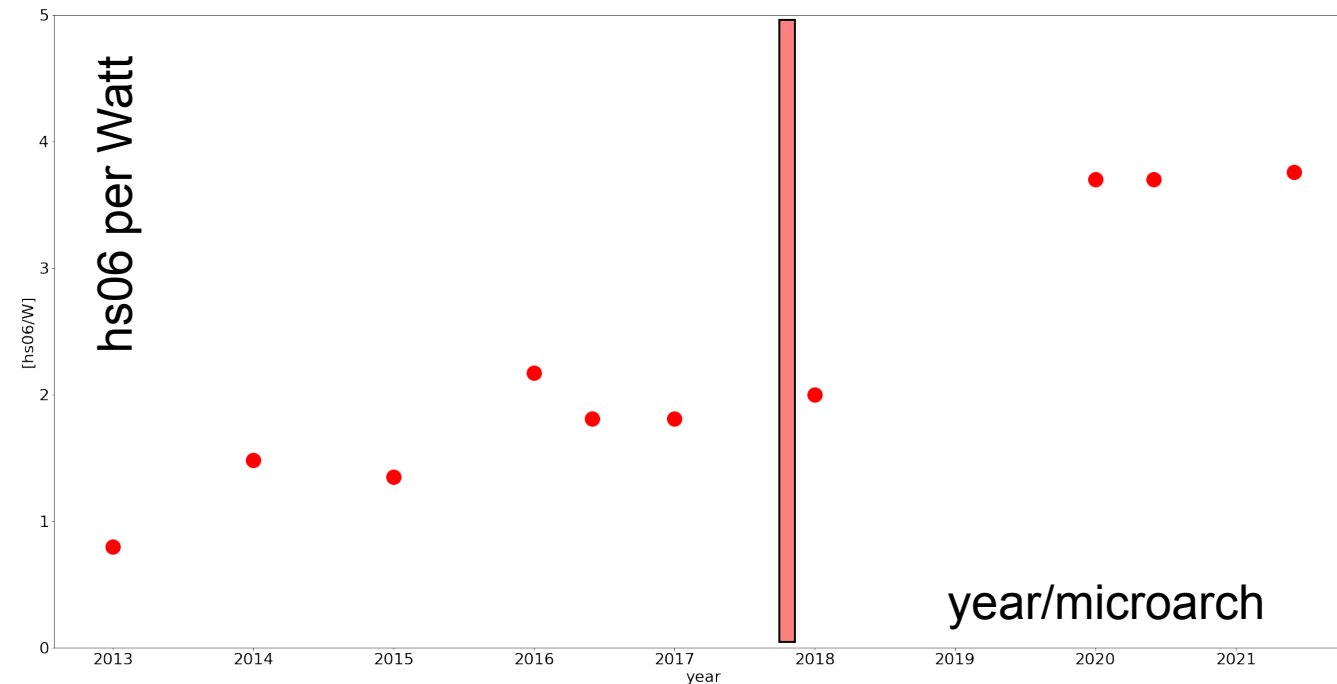


<https://www.offshore-windindustrie.de/windparks/deutschland>

Architecture/Generation Energy Efficiency

CPU Efficiency per Electric Power Consumption

- Significant efficiency gains with recent microarchs (aka Zen)
- CPU compute power per Watt gain ~4x from oldest workers still in production
- Old, energy inefficient nodes as dynamic moderators for shedding/fan out
- Shaping Frequency depending on production job run time/drainage rate

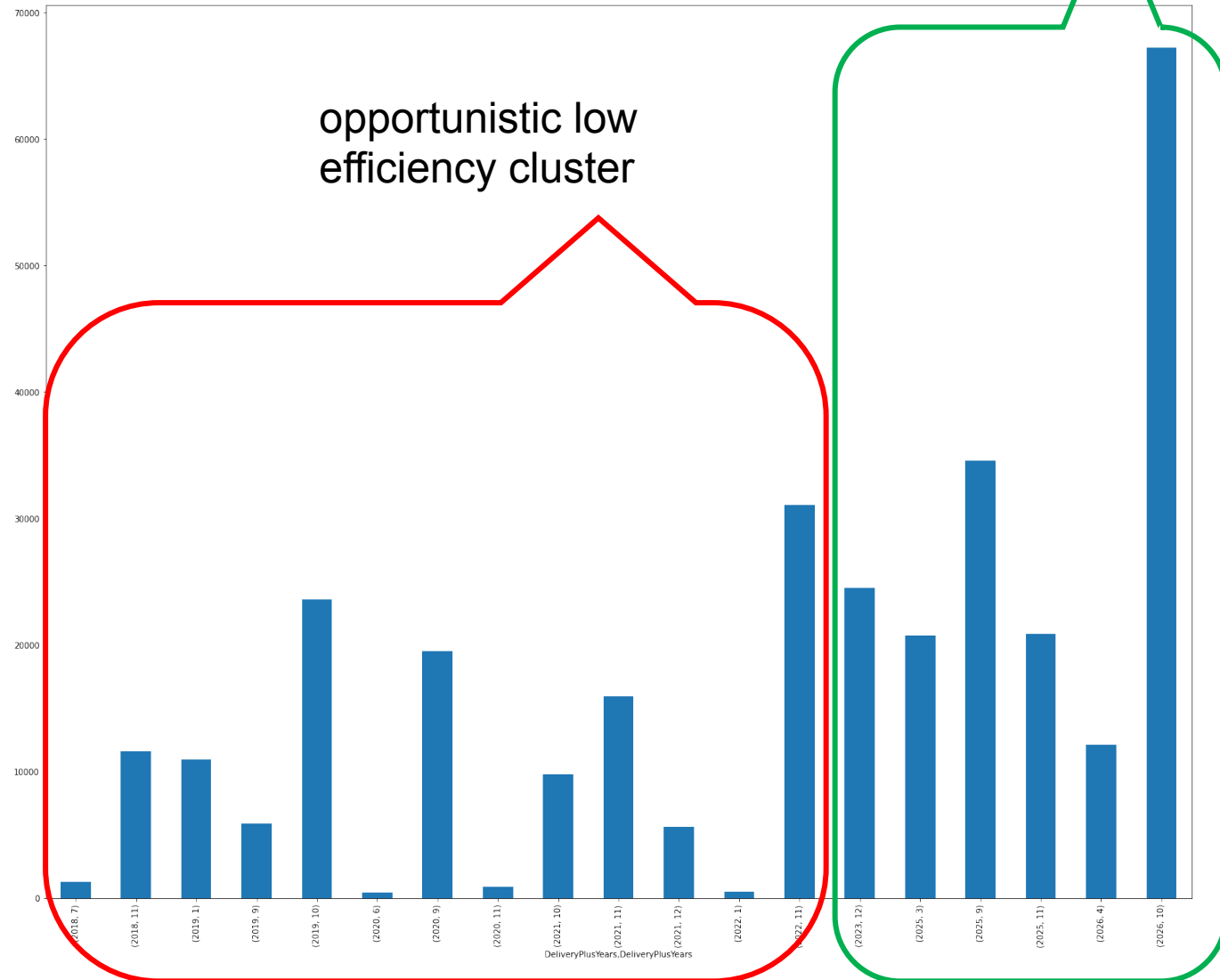


Production Cluster Energy Efficiency

WattHours consumed for HS06 delivered

E.g.

- target deliverable: 1000 kHS06
- “combo” cluster: ~410 kWh
- “high efficiency” cluster: ~298 kWh
- “low efficiency” cluster: ~587 kWh
- Low efficiency cluster as opportunistic resource
 - Load shedding when necessary
 - Scheduling has to be adapted



Opportunistic Resource Utilization

Utilizing surplus green energy

- Complementary to load shedding
- Node ramp up $O(\sim \text{minutes})$
- $O(\text{shedding})$? Depends on payload runtimes and overall cluster job entropy
- Need interface to weather/green energy pricing forecasts
 - helper HTCondor Daemon with external input for cluster shaping?
- Damping wavelengths by payloads
- How to avoid significant draining idle waste cycles
 - Backfilling short jobs?
 - Assist users implementing preemption?

Final Thoughts

Final Thoughts

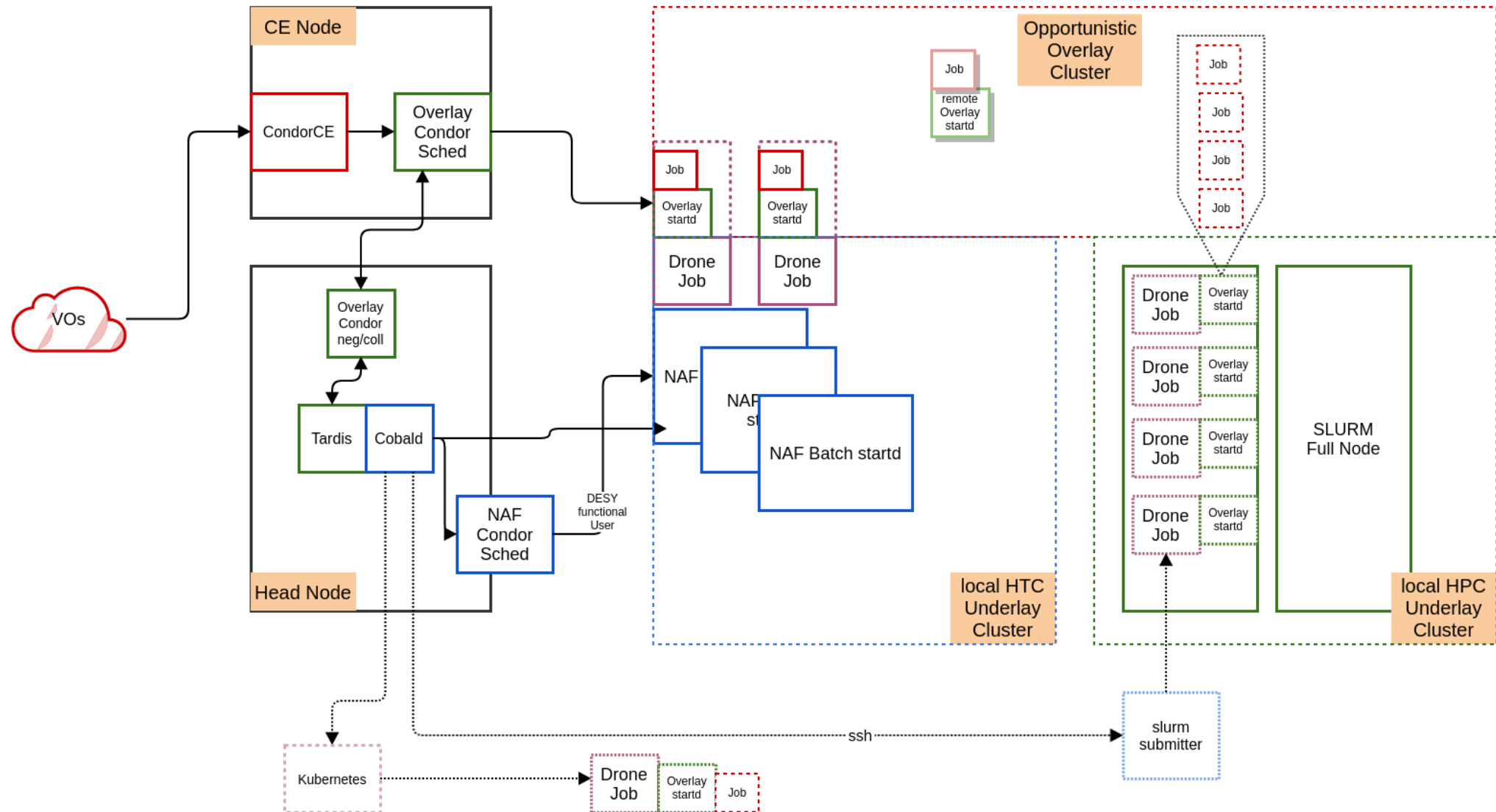
Utilizing surplus green energy

- Green Energy will bring down compute costs
 - When done right!
 - Compute as flexible sponge for Green Energy
- Have to become more flexible & dynamic
 - Dynamic cluster requires the input and input processing
 - Better monitor the clusters themselves
 - Monitor external factors
 - Green Energy Feedback Cluster Control

Appendix

Opportunistic Resource Utilization

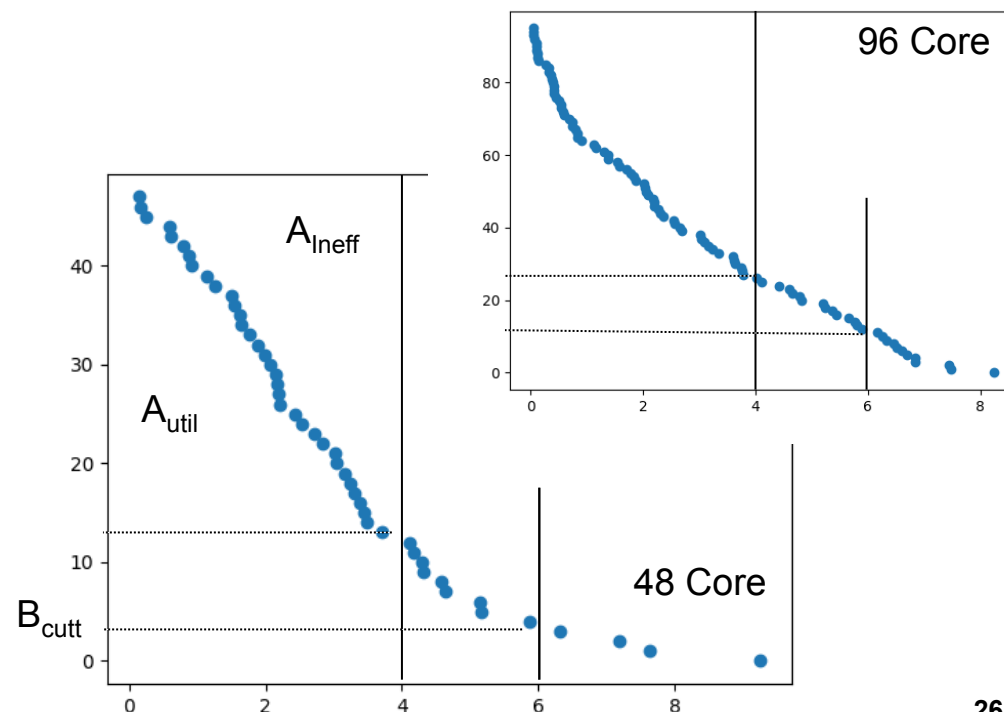
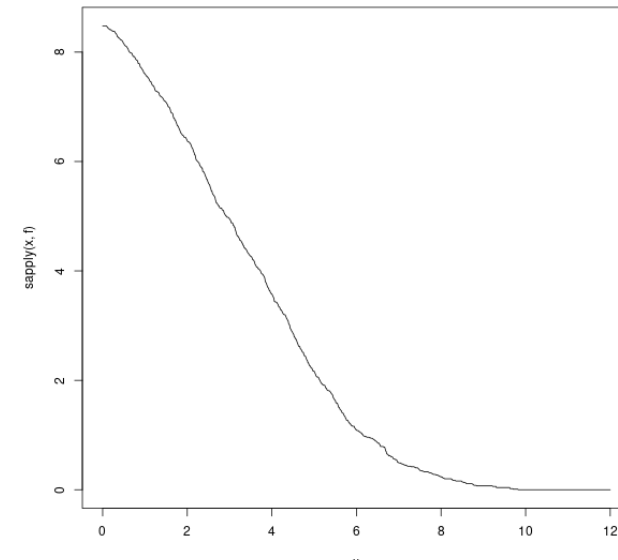
Dynamic Overlay Cluster ~ Breathing Scale up/down



Load Shedding: Worker Draining

Worker Draining Projections

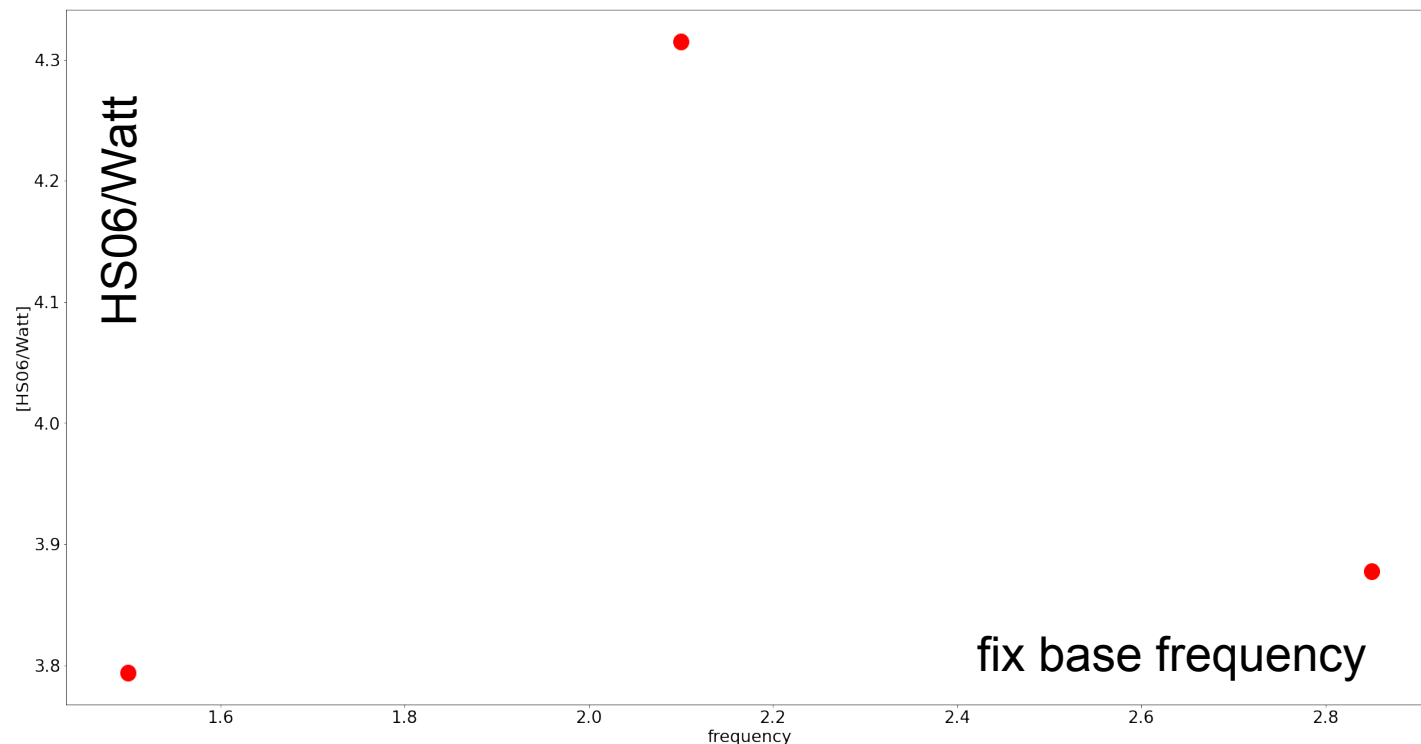
- Without scheduling information only statistic estimates
- Efficiency stochastic draining vs. scheduled draining
- Load shedding efficiency
 - Utilization drain start and shut/cutt off
- Simulation + analytic projections (K. Severin, L. Mansur, L. Janssen)
- ATLAS jobs $\sim$ 6h lifetime Gaussian + 2h width
- Old 48 core workers & new 96 core workers
- What draining inefficiency acceptable?
- Going more for vertical scheduling?



Worker Frequency Scaling

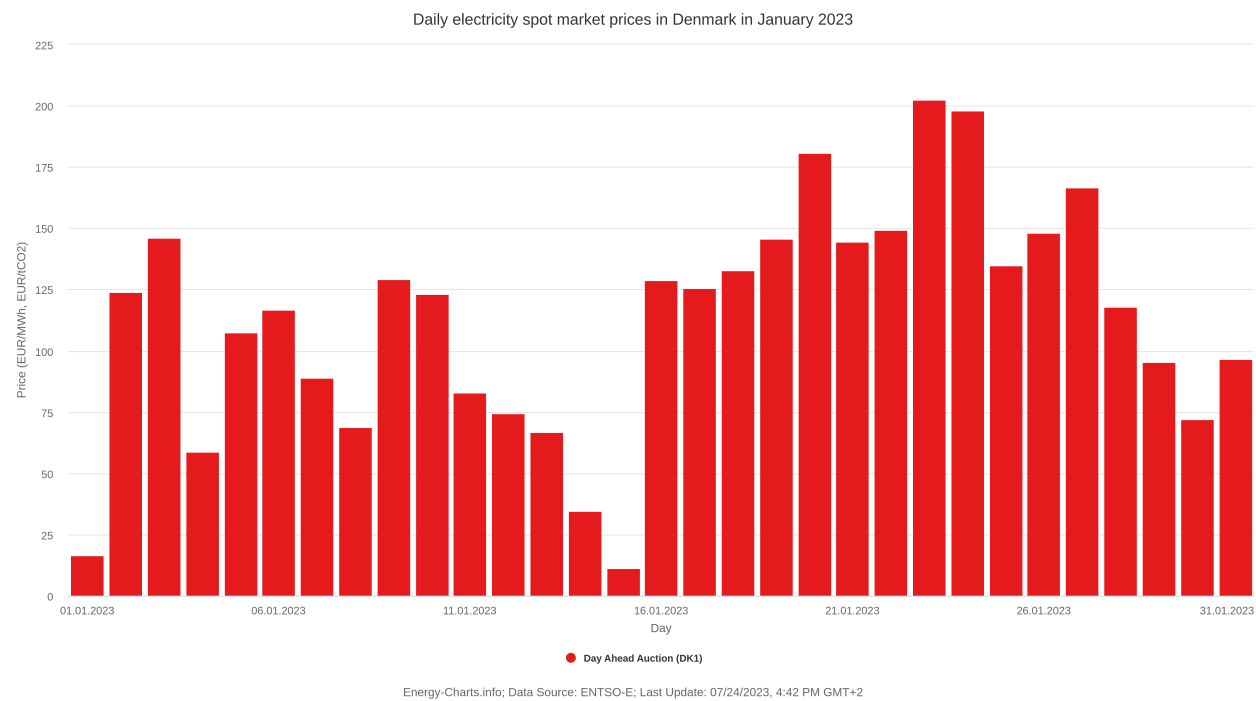
CPU Governor Scaling vs/ Sub-Clusters

- Zen only three freqs with 3.10
 - Idle offset ~150W
 - Normalized to HS06 benchmark runs
 - Efficiency sweet spot at mid freq
-
- recap: 1000 kHS06 delivered
 - “uncapped combo” cluster: ~410 kWh
 - “min freq combo” cluster: ~419 kWh
 - “high efficiency” cluster: ~298 kWh



Energy Prices

Germany/Denmark

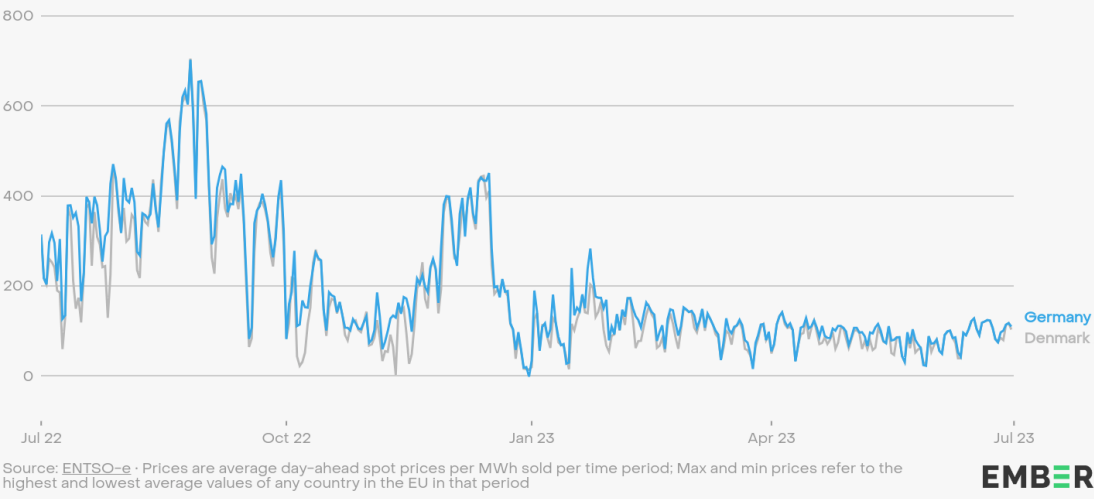


Wholesale electricity prices in Europe

€ per megawatt hour

Monthly Daily (1 year) Daily (3 months)

Germany x Denmark x Enter countries to show



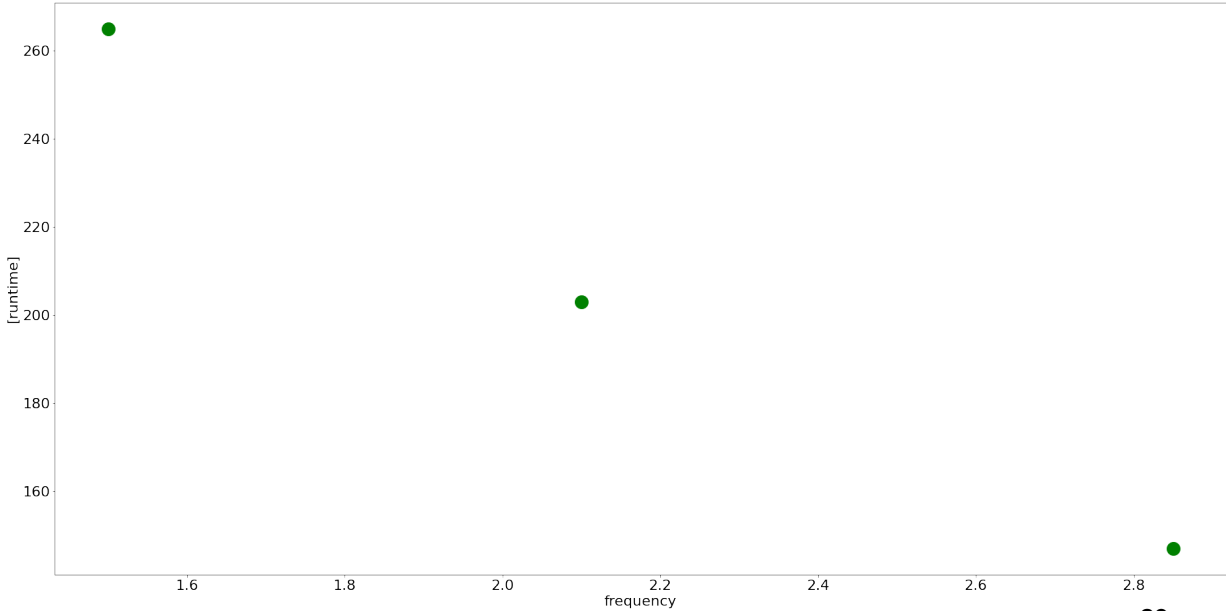
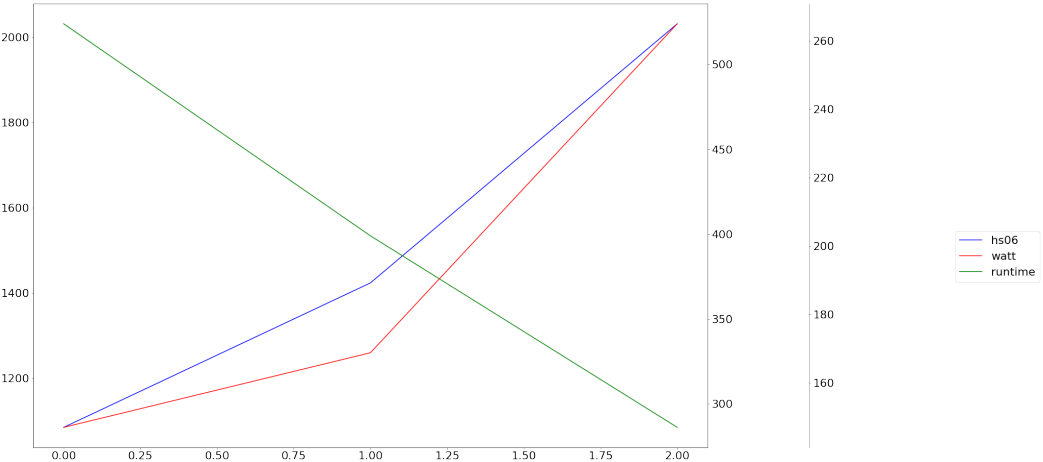
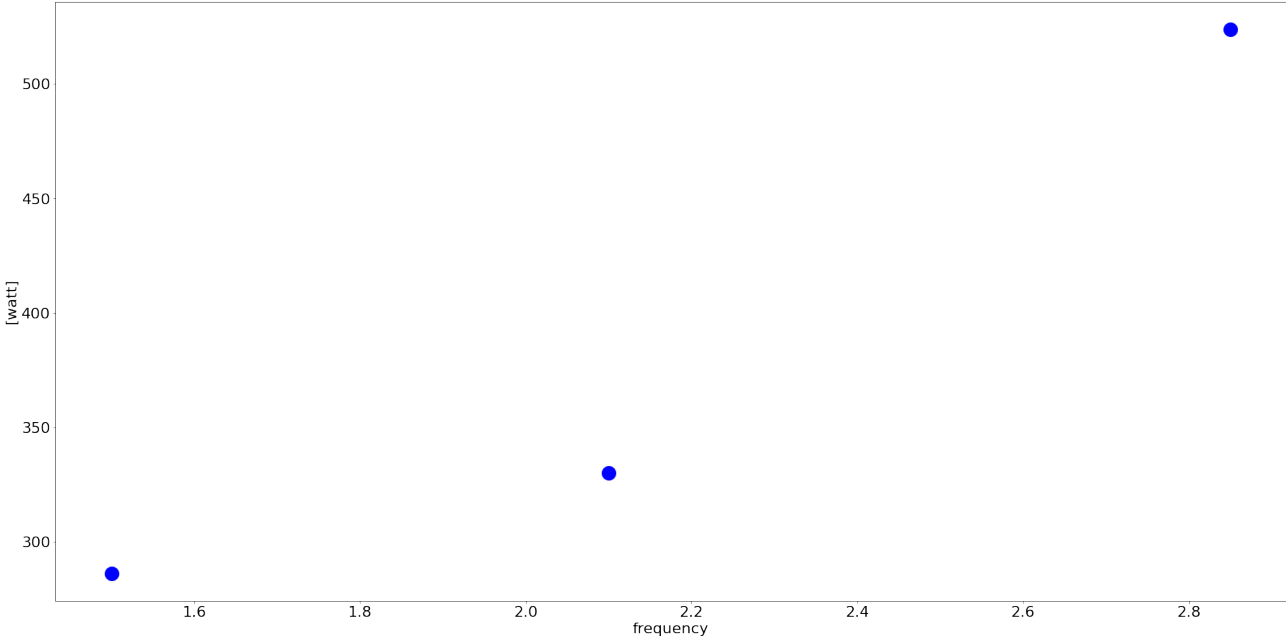
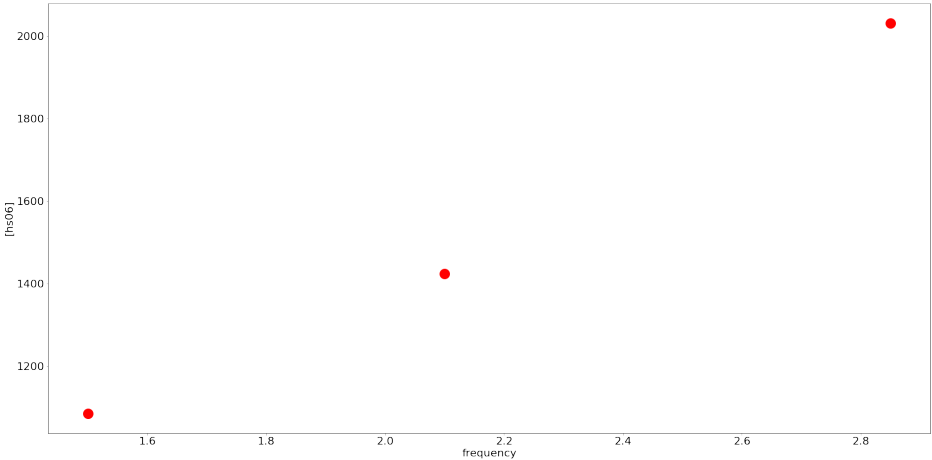
https://energy-charts.info/charts/price_average/chart.htm?l=en&c=DK&chartColumnSorting=default&interval=day&day=m01&legendItems=100

<https://ember-climate.org/data/data-tools/europe-power-prices/>

https://energy-charts.info/charts/price_average/chart.htm?l=en&c=DE&chartColumnSorting=default&interval=day&day=m01

Worker Frequency Scaling

Appendix



Worker Frequency Scaling

