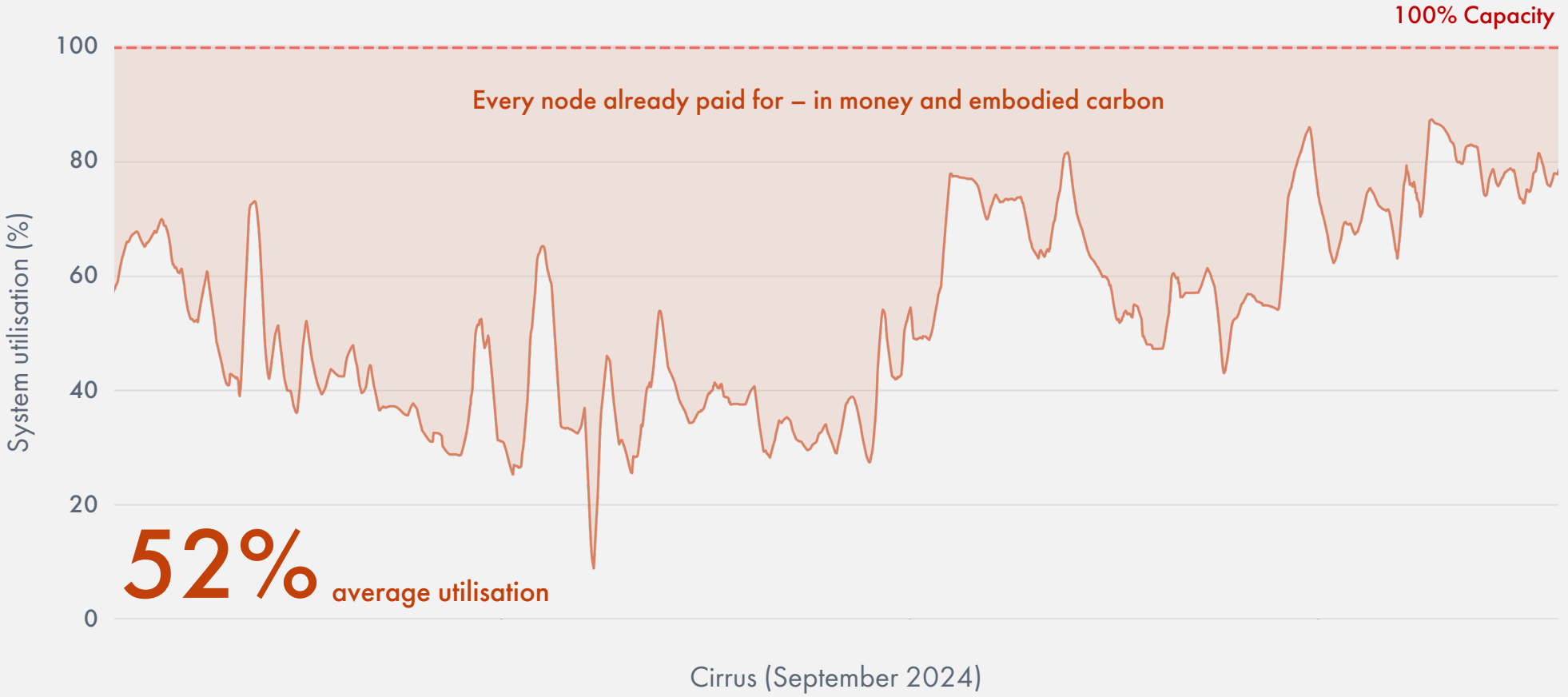
Improving HPC system efficiency through moldable and mobile job scheduling

Daniyal Arshad, Michele Weiland
EPCC, The University of Edinburgh
26 June, 2026

epcc

d.arshad@epcc.ed.ac.uk

A quiet month on a national (Tier-2) system



* These systems have variable utilisation. We deliberately picked a low-utilisation month: real headroom to reclaim!

Two interpretations of "energy efficiency"

FIRST KIND

Use less power

Reduce the average power draw, so the system consumes less total energy – real and valuable!

acts on: draw per node

SECOND KIND THIS TALK

Waste less capacity

Maximise scientific throughput per unit of energy spent. An idle node draws power for no science — and its embodied carbon is already spent.

"when the machine is quiet, the greenest core-hour is the spare one you put to work."

"In a quiet window, can the scheduler — not new hardware — turn that headroom into useful science, and what does it cost?"

Where the gaps come from...

Contributing factors: **User base**, **User behaviour** and **Workload Pattern**. **Fragmentation**: a structural cause we can act on directly!

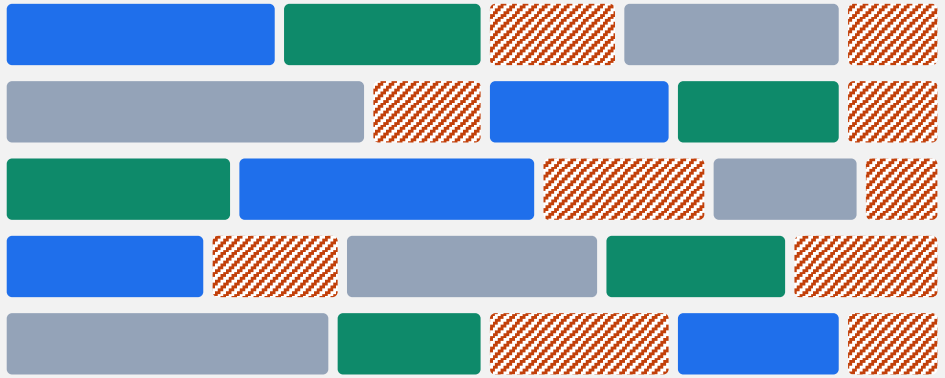
THE DEFAULT

Rigid Jobs

A fixed node count, set at submission. The scheduler packs it as-is.

→

Rigid scheduling



Rigid jobs don't tile perfectly. *Backfill* already lets a smaller job jump into a gap but it places whole rigid jobs that happen to fit.

THE SOLUTION

How can these gaps be filled?

FLEXES IN PLACE

Moldable Jobs

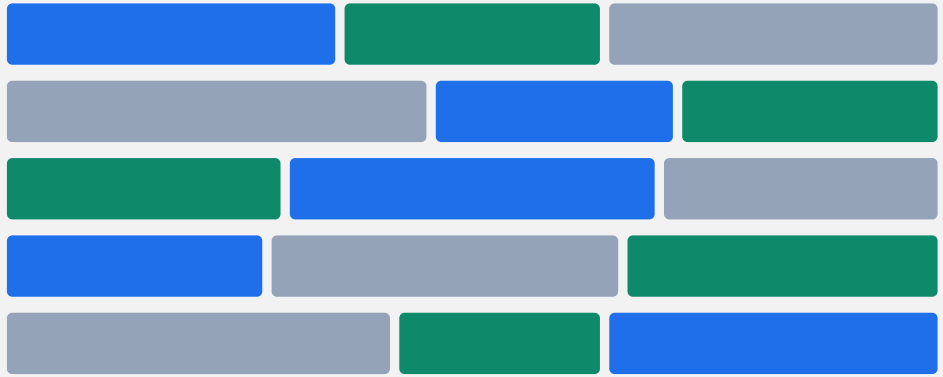
Submit a range `--nodes=<min>-<max>`; scheduler allocates as many nodes as possible, without delaying.

ARRIVES FROM OUTSIDE

Mobile Jobs

Two systems, one scheduler. Job from a busy system runs on a quiet one; freeing up space on the other.

Moldable/Mobile scheduling



Here a job changes its own shape to fit or arrives from another system entirely.

* Distinct from malleable/elastic jobs, which resize at runtime (Feitelson & Rudolph, 1998).

What we did - and what we assumed

Slurm Simulator, **replayed real job records** (Sep 2024; validated on a busier period, Jan 2025)

System	Processors (2 per node)	# Nodes (Cores)	Memory (GB)
Cirrus, Tier-2 (target)	2.1 GHz, 18-core Intel Xeon E5-2695	368 (13,248)	256
ARCHER2, Tier1	2.25 GHz, 64-core AMD 7742	5,860 (750,080)	256/512

Table. Cirrus and ARCHER2 hardware specifications

Assumptions

- **Moldable runtime**: modelled as linear speedup (½ nodes → 2× time); fixed at submission - simulator can't rescale at runtime.
- **Mobile walltime:** jobs carry runtime across different systems; a cross-architecture effect - makes absolute timing indicative.
- **Node mapping:** ARCHER2 single-node (128cores) to 4 Cirrus nodes (144cores); single-node = portable, low-risk.

The three metrics that matter...

1 **System Utilisation***

Core-hours used ÷ total core-hours available, over the steady-state period – captures job placement, runtime, and allocation size.

2 **Wait time**

Hours from job submission to job start. We report the median — robust to outliers, captures a typical user's queueing experience.

3 **Turnaround time**

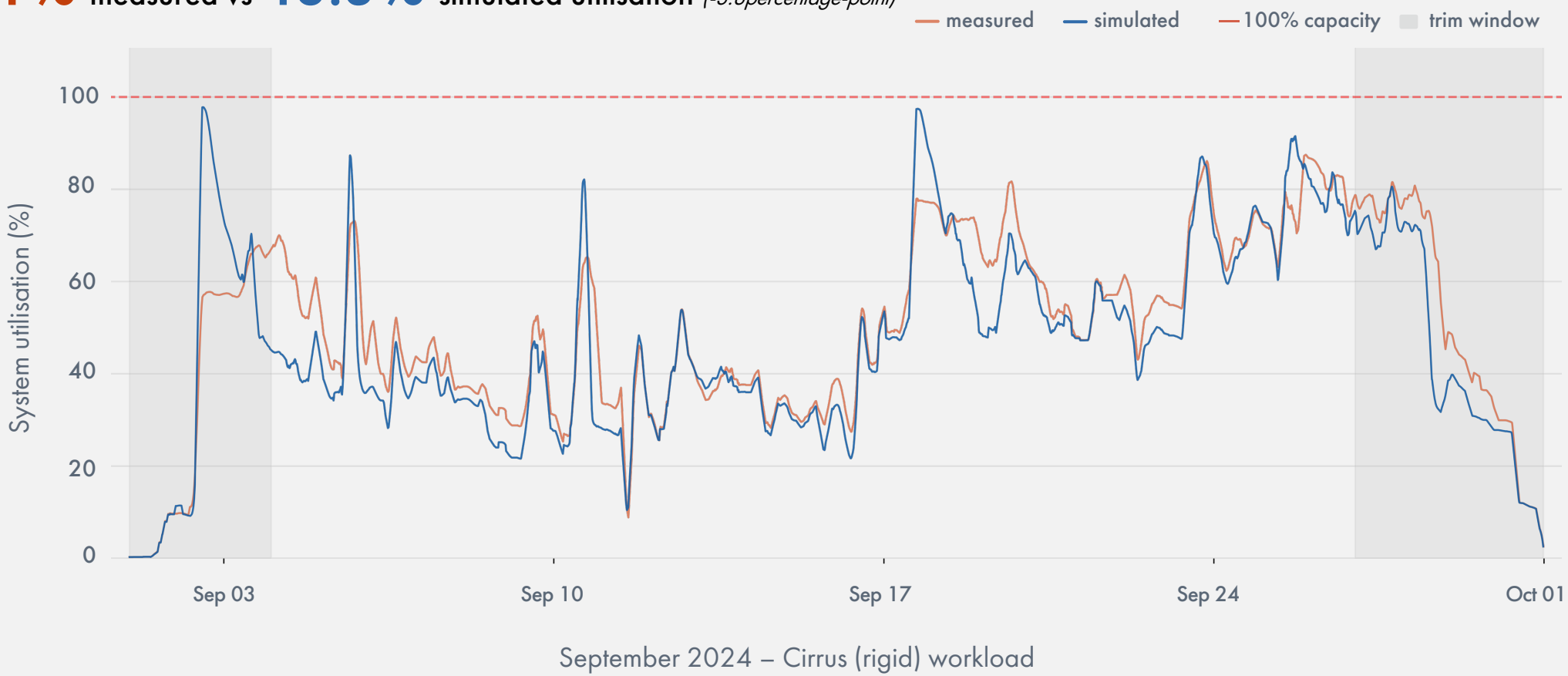
Hours from job submission to job completion. Reported as median. Unlike wait time, captures the effect of moldable scheduling on job runtime, not just queueing.

* We use utilisation as the operational proxy for scientific throughput — it's what a scheduler can measure and act on!

Does the simulator tell the truth?

~60,000 Cirrus jobs, 153 unique users across 116 accounts

52.1% measured vs **48.3%** simulated utilisation *(-3.8percentage-point)*



September 2024 – Cirrus (rigid) workload

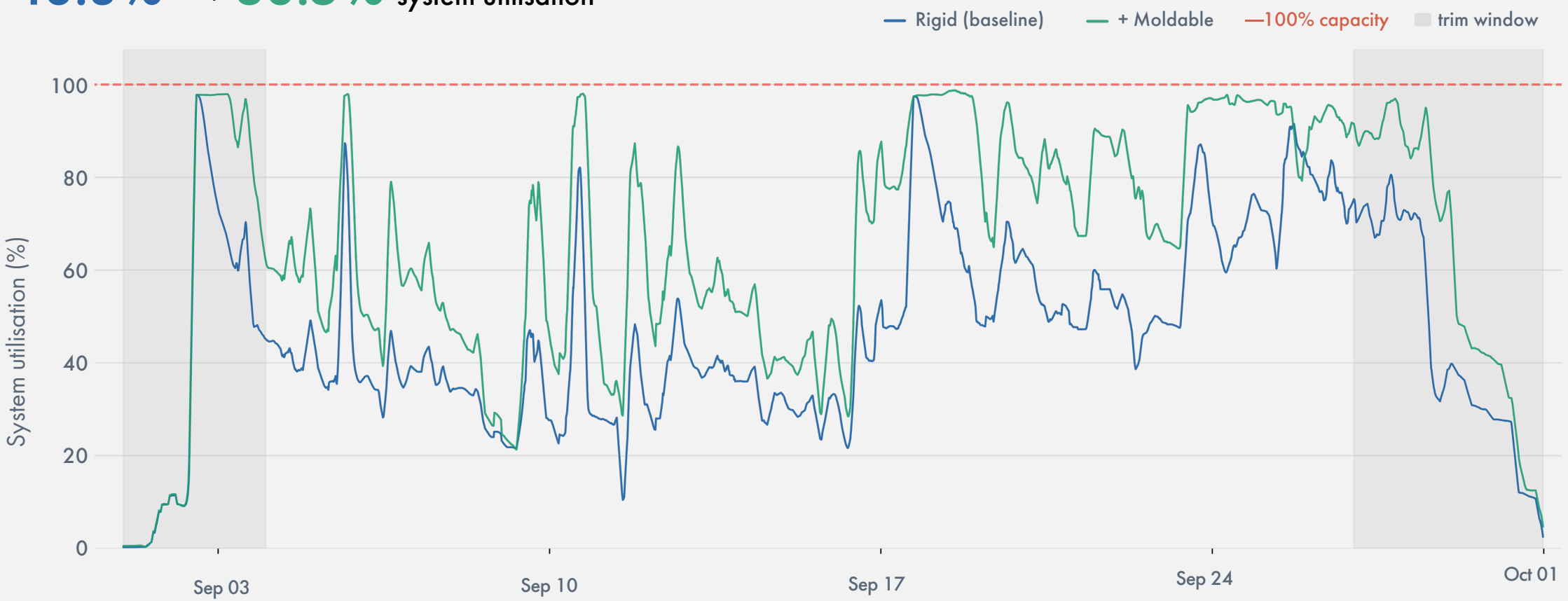
* Trim window ±3 days; Mean wait diverges - an empty-queue start artefact, not a scheduling error.

THE RESULTS

Lever 1: Moldable Jobs

Converting 5% (~3,000) Cirrus full-node jobs to moldable; *jobs slip into fragmentation gaps...*

48.3% → 68.5% system utilisation

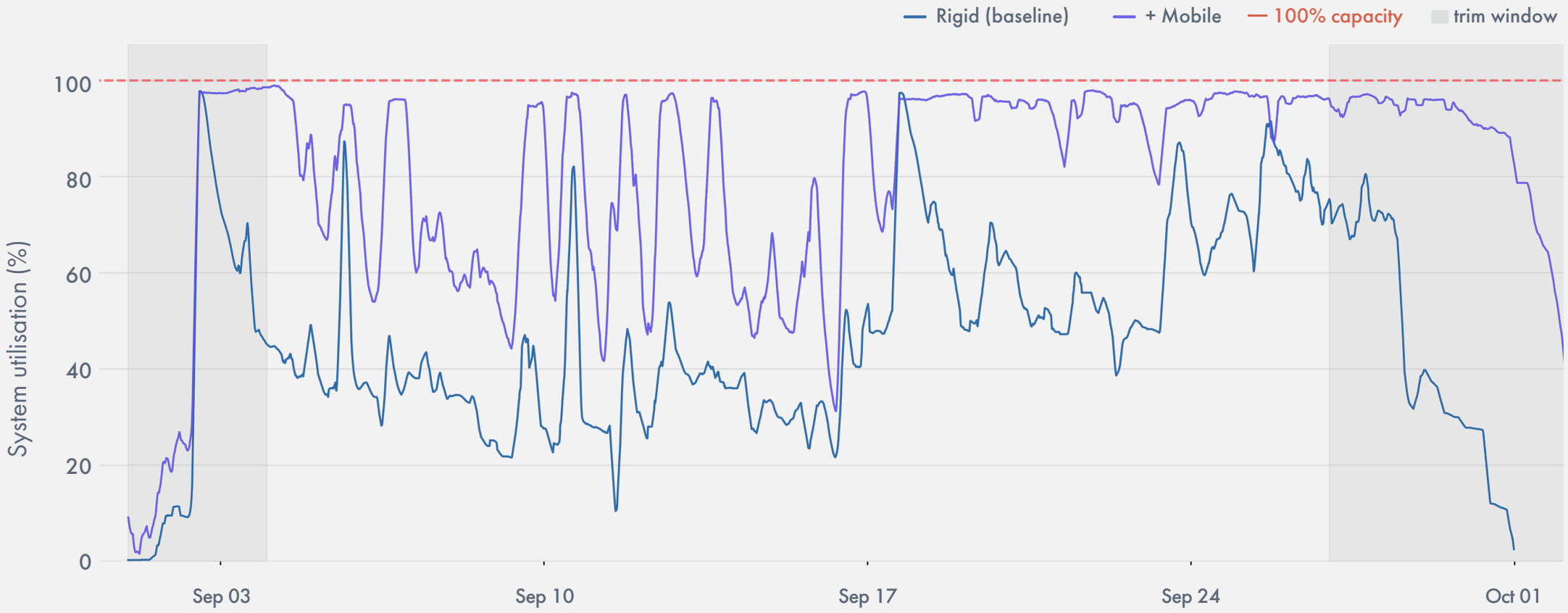


September 2024 – Cirrus (rigid) baseline vs moldable workload

Lever 2: Mobile Jobs

Combining workload with 25% (~15,000) single-node ARCHER2 jobs

48.3% → 81.7% system utilisation

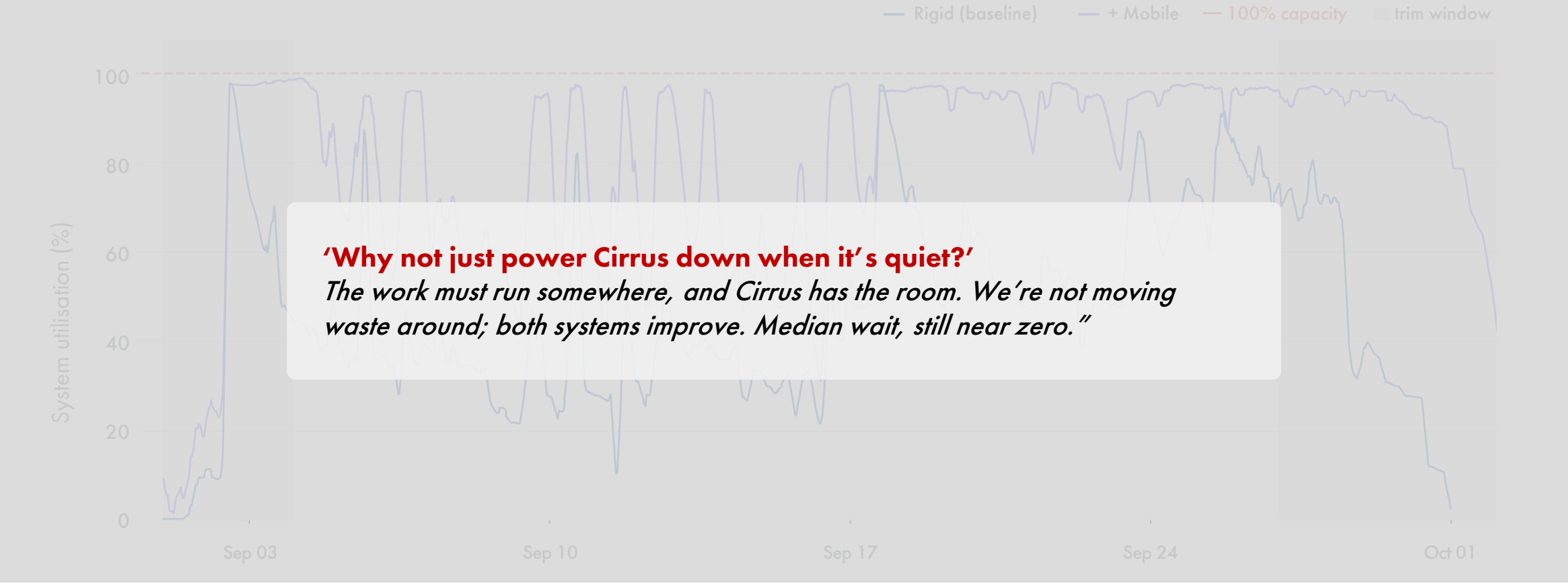


September 2024 – Cirrus (rigid) baseline vs mobile workload

Lever 2: Mobile Jobs

Combining workload with 25% (~15,000) single-node ARCHER2 jobs; *fills the quiet system and drains the busy one!*

48.3% → 81.7% system utilisation



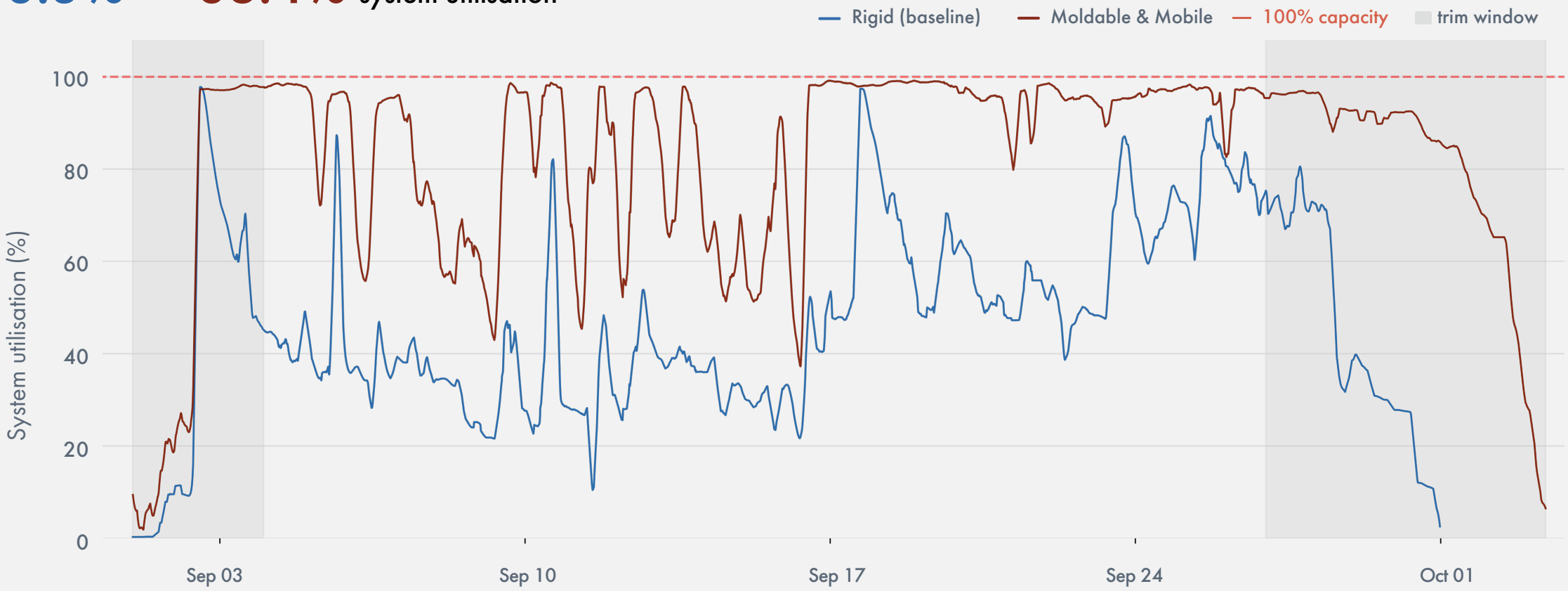
'Why not just power Cirrus down when it's quiet?'
The work must run somewhere, and Cirrus has the room. We're not moving waste around; both systems improve. Median wait, still near zero."

September 2024 – Cirrus (rigid) baseline vs mobile workload

Lever 3: Moldable plus Mobile Jobs

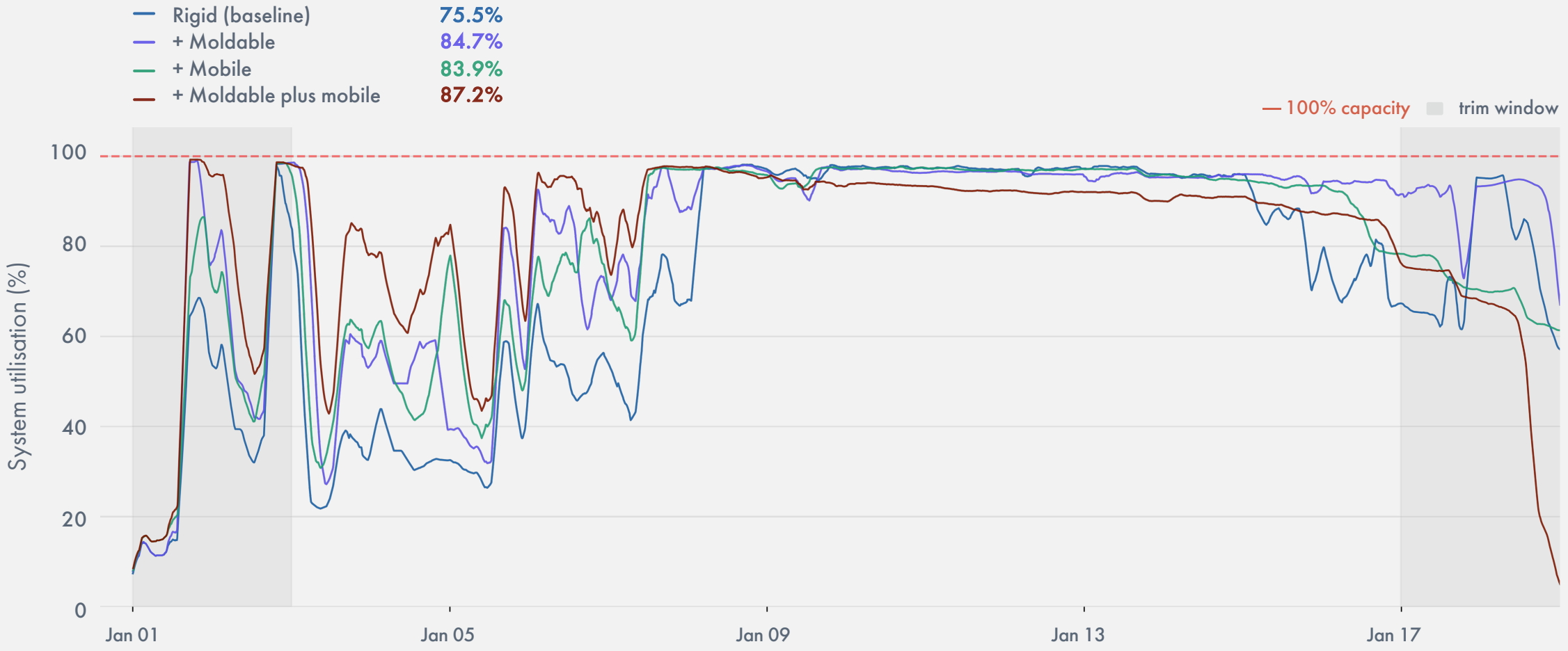
Keeping 5% Cirrus moldable and 25% ARCHER2 migrated jobs; *running close to full!*

48.3% → 86.4% system utilisation



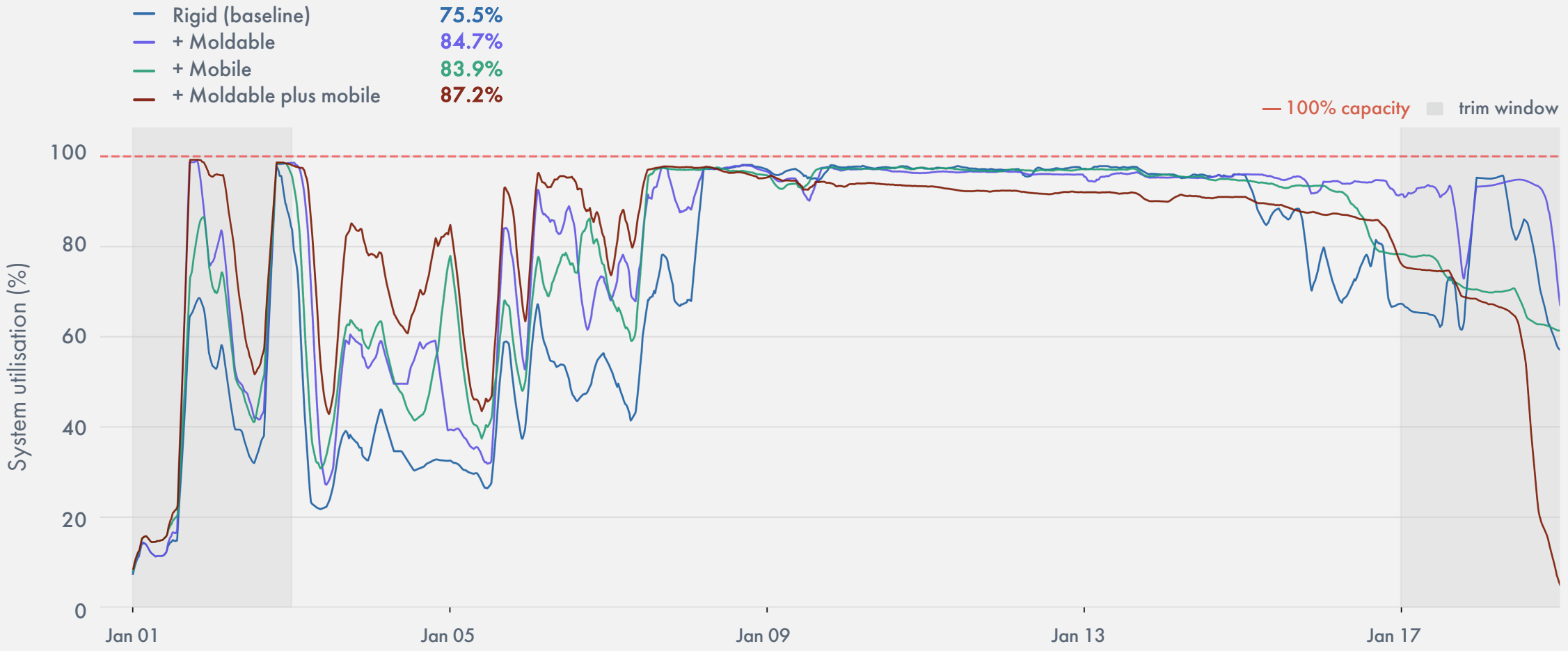
September 2024 – Cirrus (rigid) baseline vs moldable plus mobile workload

A busier period: the pattern holds;



January 2025 – Cirrus (all 4 workloads, ~85,000 jobs)

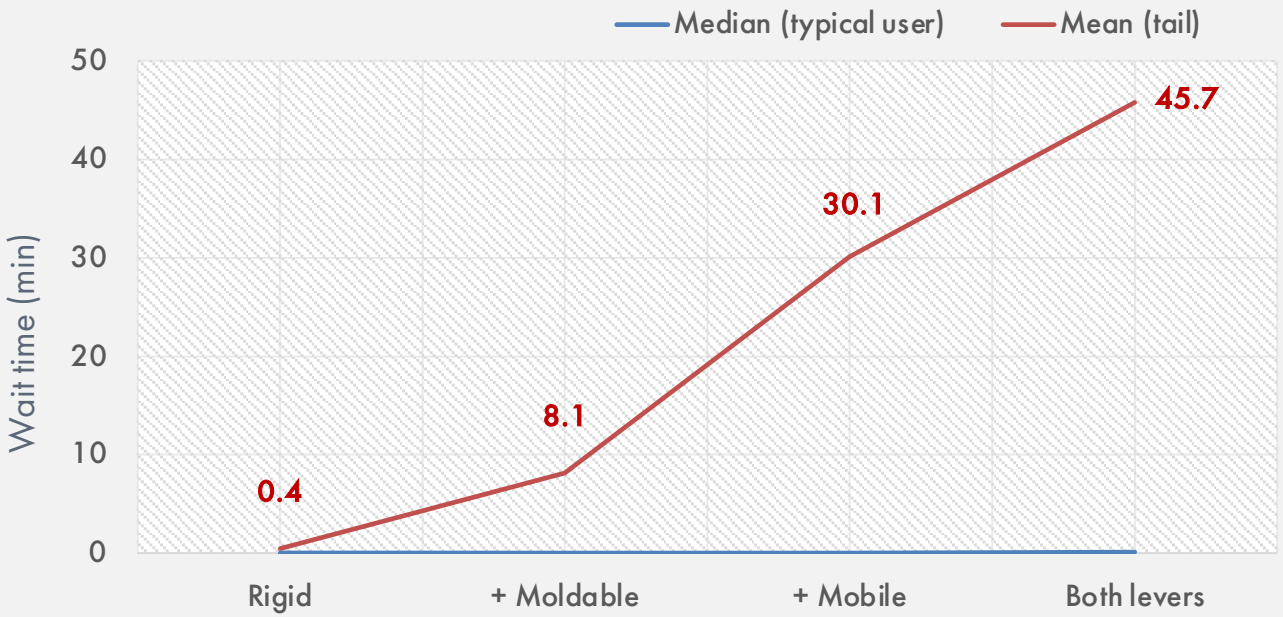
A busier period: the pattern holds; *the gain is smaller, but it's there!*



January 2025 – Cirrus (all 4 workloads, ~85,000 jobs)

What does it cost...

- The typical user is unaffected: the median wait stays near zero throughout.
- Turnaround — wait plus runtime — barely moves either (6-8mins); it stays flat!
- The mean (wait) climbs, because the system now fills up a minority of large/low-priority jobs which queue longer...



* Feitelson & Rudolph (1998): utilisation gains mustn't come at users' expense. Median and mean are exactly that check – and a dial an operator can set.

Where does the gain come from?

Mobility does the heavy lifting;

Scenario	Flexible jobs	Utilisation	Δ vs rigid	Median wait	Median Turnaround
Rigid (baseline)	---	48.3%	---	0.02min	6.05min
+ Moldable	5%*	68.5%	+20.2 pp	0.03min	6.12min
+ Mobile	25% (ARCHER2)	81.7%	**+33.4 pp**	0.03min	6.50min
Both levers	↑	86.4%	+38.1 pp	0.10min	7.63min

Table. Simulation results reported against 48.3% baseline (Cirrus, Sept 2024)

Where does the gain come from?

Mobility does the heavy lifting; *moldability is a smaller, workload-limited top-up!*

Scenario	Flexible jobs	Utilisation	Δ vs rigid	Median wait	Median Turnaround
Rigid (baseline)	---	48.3%	---	0.02min	6.05min
+ Moldable	5%*	68.5%	+20.2 pp	0.03min	6.12min
+ Mobile	25% (ARCHER2)	81.7%	**+33.4 pp**	0.03min	6.50min
Both levers	↑	86.4%	+38.1 pp	0.10min	7.63min

Table. Simulation results reported against 48.3% baseline (Cirrus, Sept 2024)

* Workload is dominated by single-node, tightly-coupled jobs. The 5% isn't a knob we chose; it's what the workload allows.

Why utilisation is an energy win?

More jobs mean more power? Haven't we just traded idle waste for active consumption...

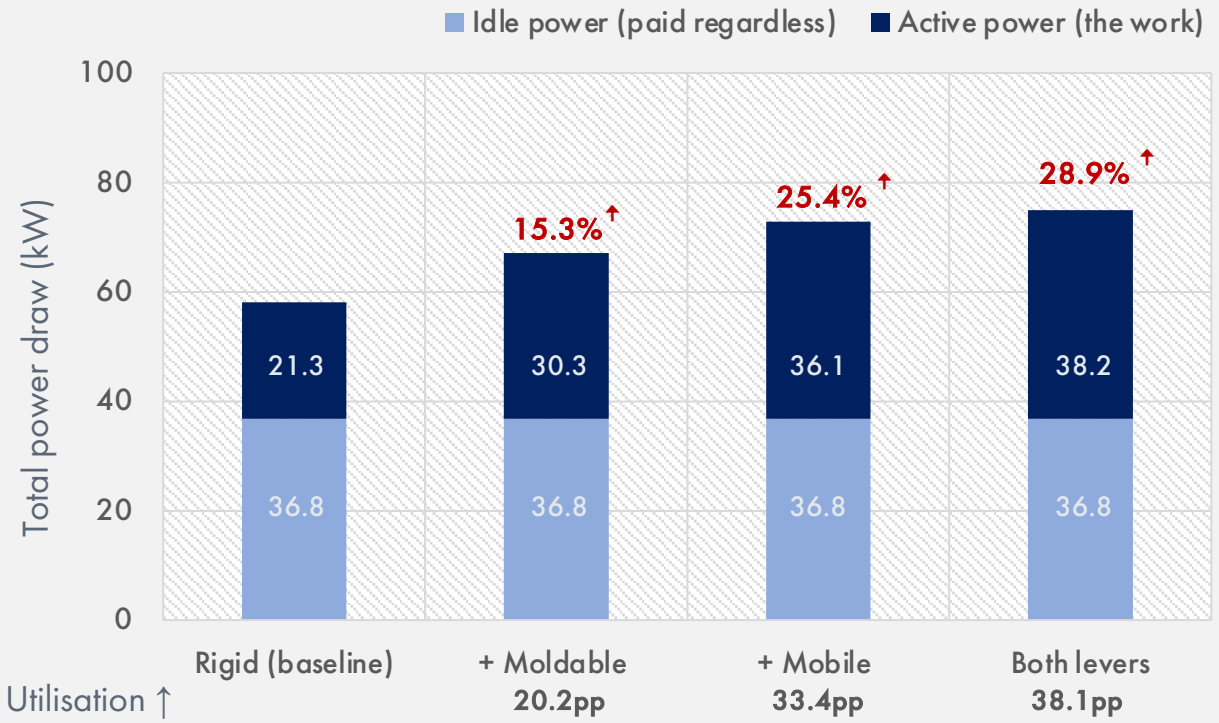


Fig. Power grows slower than work; against 48.3% baseline (Cirrus, Sept 24)

Assumption:
(i) Idle power/node=100W; 368 nodes×100W=36.8kW
(ii) Power draw (fully loaded node)=220W; workload adds 120W
(ii) Sept 2024 utilisation; 0.483×368×120W=21.3kW

Why utilisation is an energy win?

More jobs mean more power? Haven't we just traded idle waste for active consumption...

No, a 38-point gain, (~79% relative rise) in utilisation costs ~29% more power.

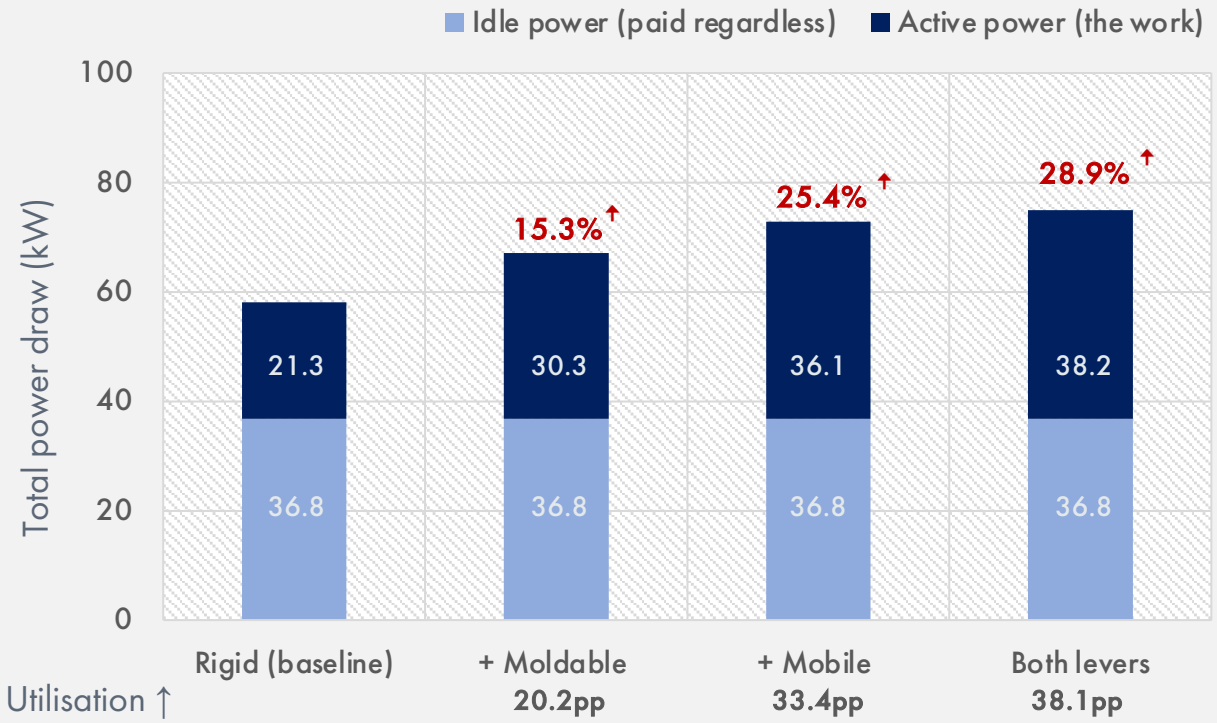


Fig. Power grows slower than work; against 48.3% baseline (Cirrus, Sept 24)

Assumption:
(i) Idle power/node=100W; 368 nodes×100W=36.8kW
(ii) Power draw (fully loaded node)=220W; workload adds 120W
(ii) Sept 2024 utilisation; 0.483×368×120W=21.3kW

From simulation to real-world deployment

Nothing here is a scheduler patch we switch on tomorrow - A tiered, **opt-in model**: the user decides...

Modelling

How faithfully jobs are represented

- Linear speedup (½ nodes → 2× time) is a ceiling real MPI rarely meets.
- *Gains are strongest for codes that can run on arbitrary number of threads or processors.*

Engineering

What mobility needs to deploy

- Binary portability, data co-located with compute, matching module environments – are prerequisites.
- *Mobility gains concentrate where applications exist on both systems and storage is shared/federated.*

Administrative

The agreements across systems

- Cross-system allocation, licensing restricted codes, security/access/ data-handling policy.
- *Manageable — but they are agreements, not code.*

Where do we go from here...

1 **From utilisation to carbon**
Model embodied + operational emissions per scenario

2 **Custom walltime-scaling**
Generate more realistic moldable schedules using the simulator

3 **A dynamic scheduler**
Toggle between moldable/mobile jobs based on system load

Thank you!

Engineering and Physical Sciences Research Council
The work presented was supported by EPSRC, grant numbers *EP/Y53061X/1* and *EP/Z531170/1.*

Appendix: Moldable walltime model

The scaling model assumes walltime is inversely proportional to node count (linear speedup) where ***n*** is the number of allocated nodes:

$$walltime(n) = original\_walltime \times original\_nodes / n$$

For node range [min, max]:

$$walltime\_at\_min = original\_walltime \times original\_nodes / min\_nodes$$
$$walltime\_at\_max = original\_walltime \times original\_nodes / max\_nodes$$

$$avg\_walltime = (walltime\_at\_min + walltime\_at\_max) / 2$$

Pre-computed and written to the ***.events*** file at submission; the SLURM Simulator can't rescale walltime at runtime.

Appendix: Energy-efficiency (Analysis)

Scenario	Utilisation Increase	Active Power (W)	Total Power (active + idle, W)	Power Increase
Rigid (baseline)	---	21,329	58,129	---
+ Moldable	+ 20.2 pp	30,250	67,050	15.3%
+ Mobile	+ 33.4 pp	36,079	72,879	25.4%
Both levers	+ 38.1 pp	38,154	74,954	28.9%

Table. Simulation results reported against 48.3% baseline (Cirrus, Sept 2024)