Evaluating Runtime Prediction and Conservative Walltime Policies in HPC Systems

H. Romero-Ugalde, Y. Faqir-Rhazoui, M. Funk and J. Gorroñogoitia Cruz
Bull R&D

SBAC-PAD 2026

© Bull SAS

Why walltime matters

The walltime dilemma - Schedulers need accurate reservations, and users don’t want their jobs cut.

Too high

Overestimation

- Inflated reservations
- Less backfilling flexibility
- Longer queues / **Lower efficiency**

Too low

Underestimation

- **Job killed** before completion
- Lost user time
- Potential resubmission

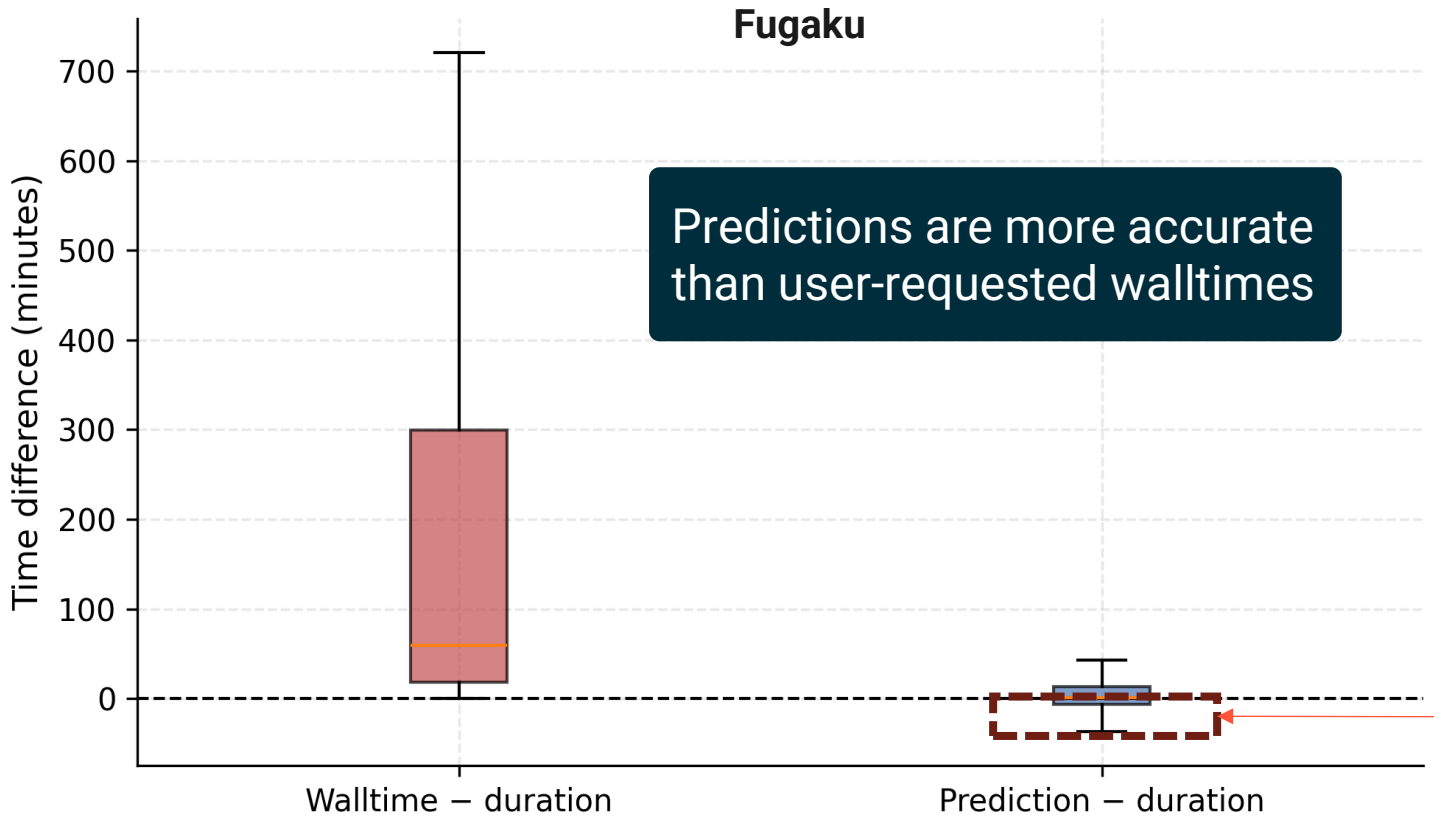
The costs are asymmetric: an overestimate is inefficient, and an underestimate can terminate the job.

Users over-request on purpose → **The whole cluster pays for it**

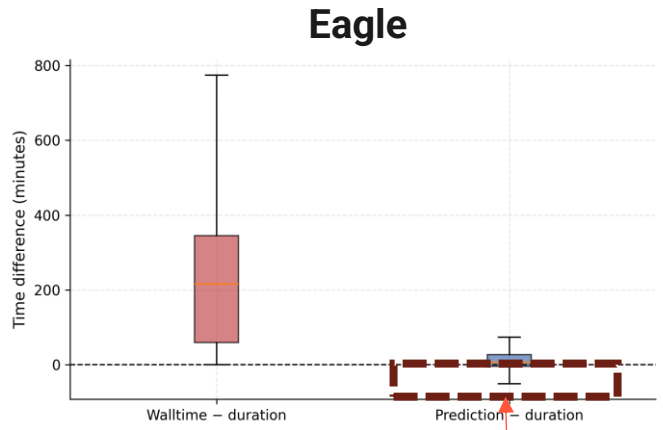
© Bull SAS

In a perfect world

Either the user or the predictor would know the exact job duration. In practice, they don’t.



Predictions are more accurate than user-requested walltimes



But if predictions directly replace walltimes, underestimated jobs can be killed

Predictions can improve walltime estimates, but directly replacing requested walltime with a prediction is unsafe. Underestimation can terminate jobs prematurely. The question is: how can we use predictions safely?

© Bull SAS

What we propose

A predictor-agnostic decision layer between the model and the scheduler

Predictor-agnostic decision layer

Prediction

Estimate runtime from submission-time metadata using a realistic rolling temporal evaluation.

Similarity-aware

Treat previously observed workloads (IN) differently from unseen workloads (OUT).

SAFE walltime adjustment rule

Bounded reduction of the user-requested walltime: shrink only when the prediction is far below the request, **never below β×request** or the prediction.

Scheduler

Use the updated walltime

Key idea: prediction is an input signal - not the final scheduling decision.

© Bull SAS

Two large production traces

Different systems, different schedulers, same pre-run information

System	Jobs	Users	Period	Scheduler
Fugaku	24M	3,457	Mar 2021 - Apr 2024	PBS
Eagle	11M	936	Nov 2018 - Feb 2023	SLURM

Common interface: user, requested nodes, requested memory and requested walltime exist on both systems, so one pipeline serves both.

Only successful jobs. Runtimes converted to minutes. All splits strictly time-ordered.

Repetition is the norm. Heavy re-execution of the same workloads motivates the IN/OUT distinction.

35M+

production jobs

~7K

jobs per user (Fugaku)

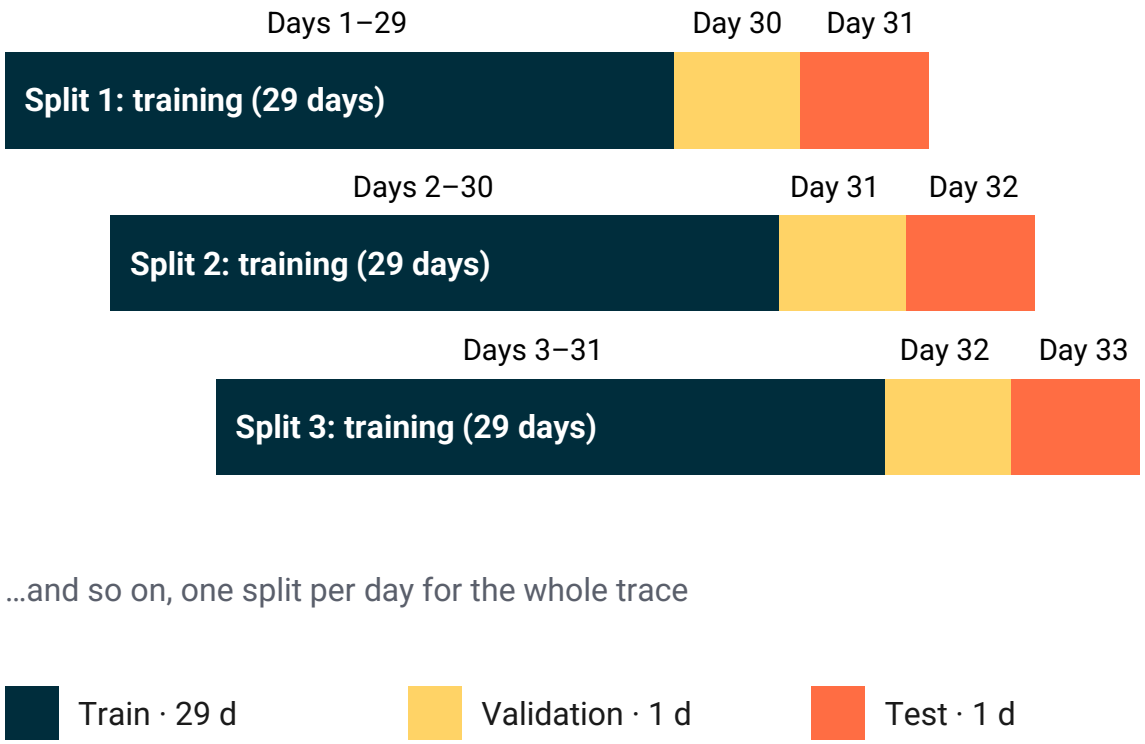
~11K

jobs per user (Eagle)

© Bull SAS

Rolling temporal evaluation

Train on the past, predict tomorrow - no retrospective bias



Every day, the pipeline:

1. Trains candidate models on 29 days
2. Selects on 1 validation day (RMSE)
3. Retrains on train + validation
4. Predicts the next unseen day, job by job
5. Adjusts each job's walltime from that prediction and the user request

All metrics are computed on days the model never saw, then aggregated over the whole period.

© Bull SAS

Features and predictors

Semantic job embeddings; representative models of increasing complexity

Feature representation [1]

Job metadata	user, job name, nodes, memory, walltime…
↓	
Text string	"jobname,jobenv,…"
↓	
SBERT encoder	all-MiniLM-L6-v2
↓	
384-d embedding	**input to every model**

Predictors

Linear regression

baseline on the same features

Neural network (MLP)

3 hidden layers, MSE or Huber loss, best of many configurations, selected by validation RMSE

NN ensemble

Top-5 validation models, weighted by inverse RMSE. **Note**: used by SAFE.

[1] F. Antici, A. Bartolini, J. Domke, Z. Kiziltan, and K. Yamamoto, “F-DATA: A Fugaku workload dataset for job-centric predictive modelling in HPC systems,” Jun. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.11467483

© Bull SAS

Contribution 1: similarity-aware regimes

Is this job similar to something we have seen in the last 30 days?

Train

Job	emb0	emb384
Tr1	0.1	0.2
Tr2	0.1	0.3
Tr3	0.5	0.8
Tr10	0.1	0.5

Test

Job	emb0	emb384
Te1	0.5	0.2
Te2	0.1	0.2
Te3	0.1	0.5
Te4	0.4	0.3

ALL — Every job in the test day

IN — Signature seen in the 30-day training window → **confident prediction**

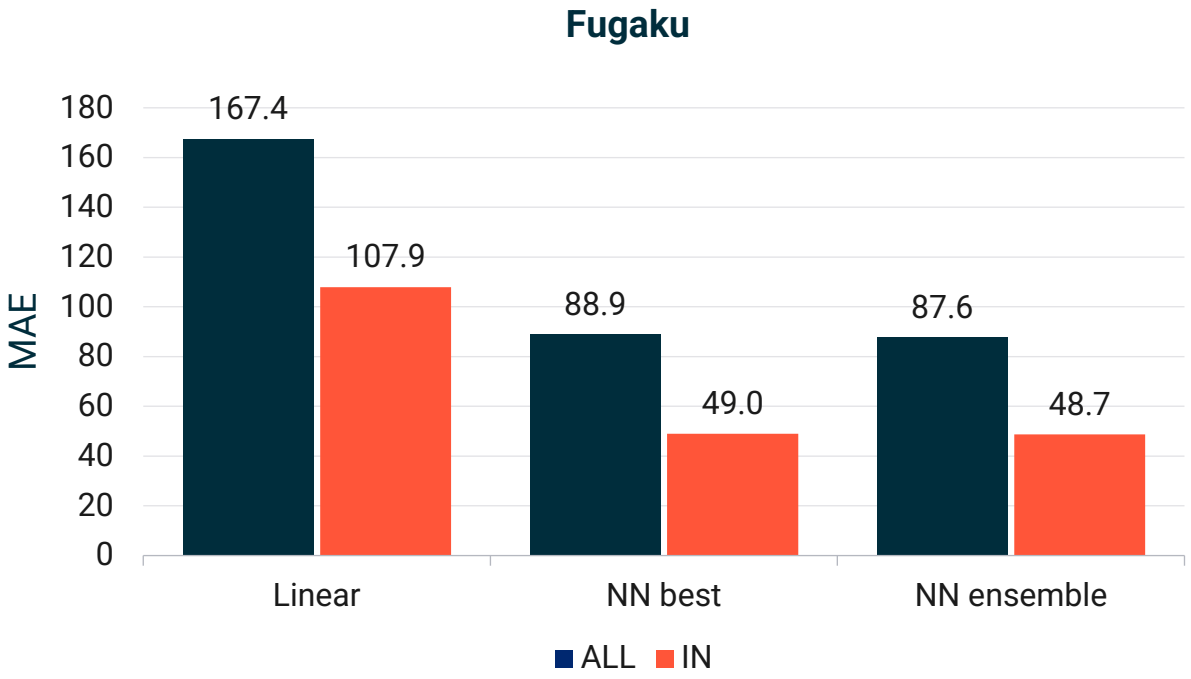
OUT — Signature never seen → **less reliable**; the model must extrapolate

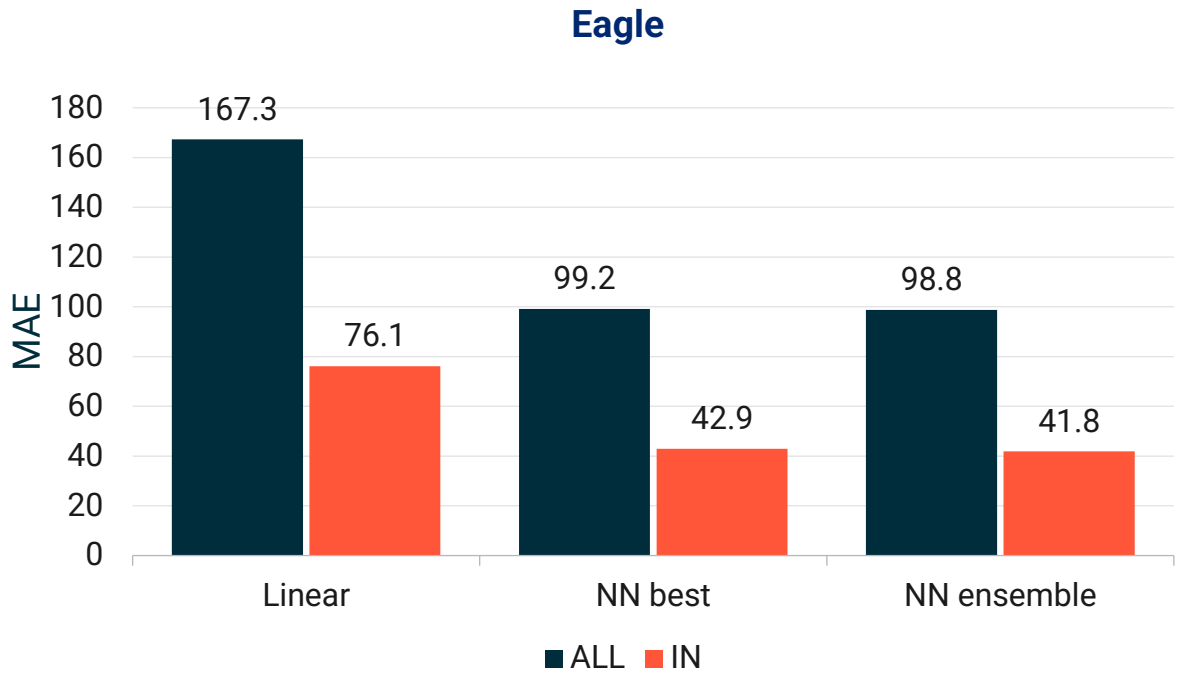
ALL = IN ∪ OUT

© Bull SAS

Result 1 - familiarity drives accuracy

Mean absolute error (minutes): repeated workloads (IN) vs ALL jobs





0.57 → 0.67

R² (ALL → IN), Fugaku, NN ensemble

~2×

lower **MAE** than linear (Fugaku, ALL)

0.26 → 0.55

R² (ALL → IN), Eagle, NN ensemble

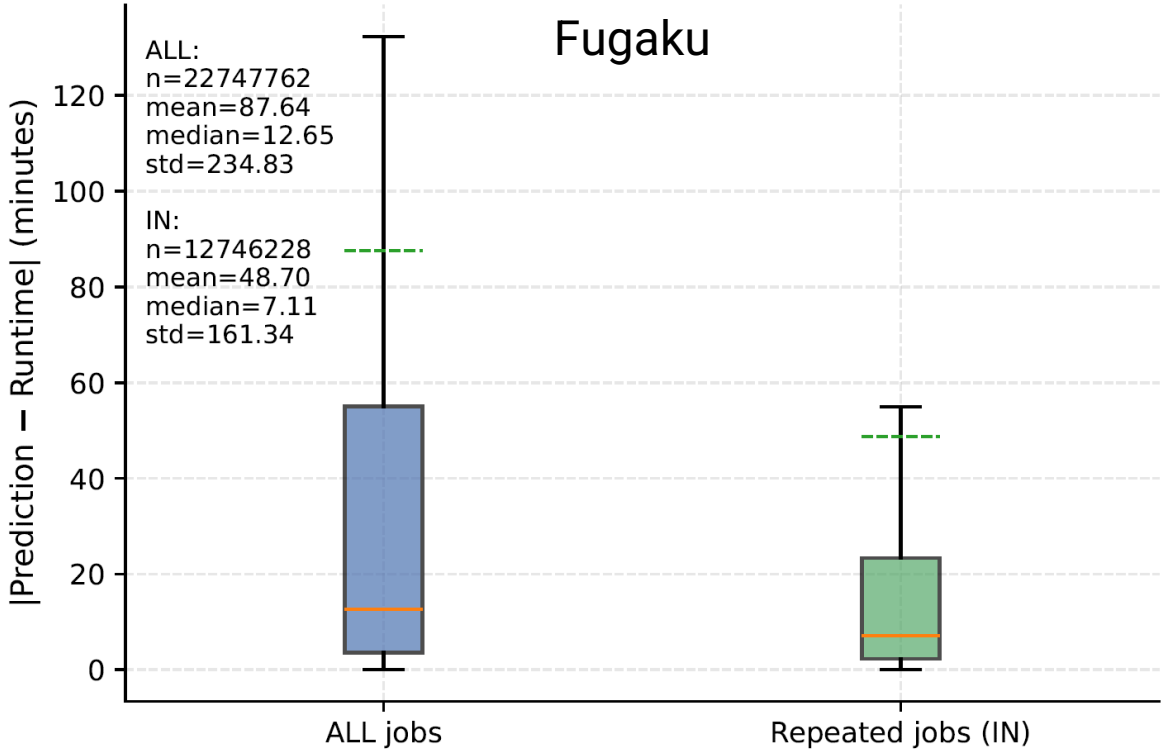
−58%

MAE on Eagle going ALL → IN

© Bull SAS

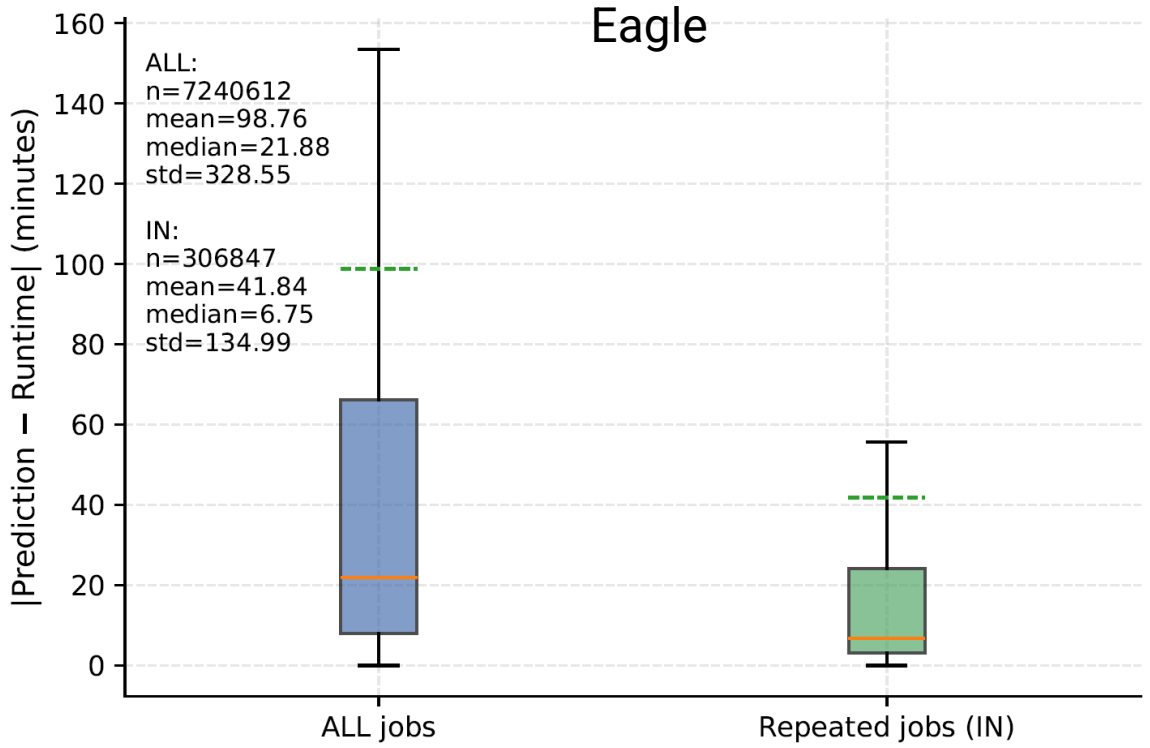
Result 1 - error distributions

Repeated jobs (IN) have far smaller errors, and the median shows it clearly



12.7 → 7.1 min

median |error|, Fugaku (ALL → IN)



21.9 → 6.8 min

median |error|, Eagle (ALL → IN)

© Bull SAS

Contribution 2 - the SAFE rule

Shrink the request only when the prediction is far below it, and never too far

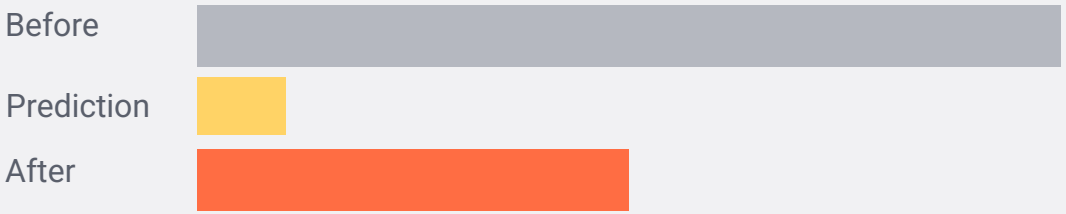
Recommended walltime

$$\hat{T}_{adj} = \begin{cases} max(\beta \times elpl, \hat{T}) & if \ \frac{\hat{T}}{elpl} < \alpha \\ elpl & otherwise \end{cases}$$

$elpl$ = user-requested walltime
$\hat{T}$ = predicted runtime
$\alpha = 0.1$, $\beta = 0.5$ (fixed, conservative)

Example 1: large slack

Request 1000 min · prediction 80 min · ratio 0.08



80 / 1000 < 0.1 → new walltime = **max(500, 80)** = 500 min (−50%)

Example 2: little slack

Request 1000 min · prediction 700 min · ratio 0.70

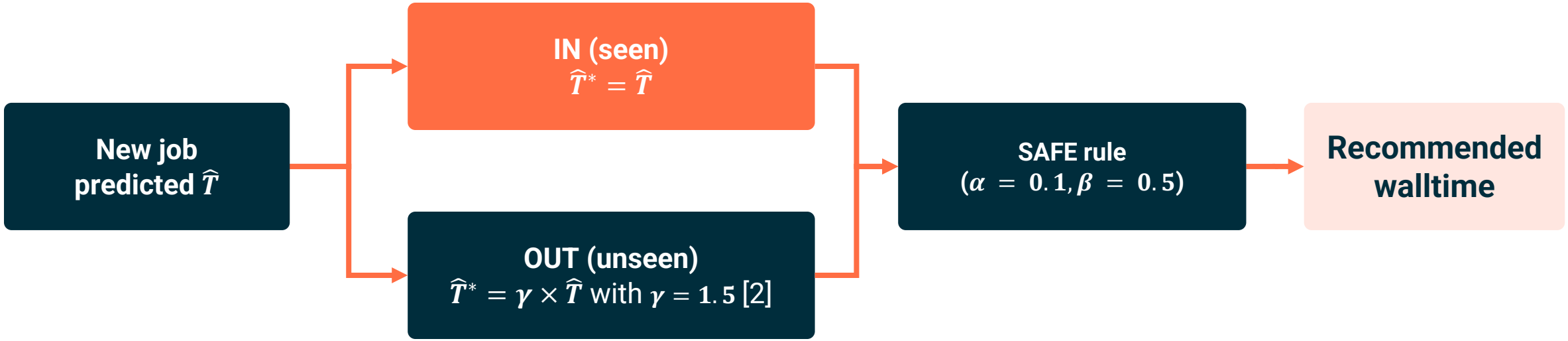


700 / 1000 ≥ 0.1 → **keep the user's 1000 min**

© Bull SAS

Contribution 3 - similarity-aware SAFE

Unseen workloads get an **extra safety margin before the SAFE rule** is applied



Same job, different treatment (request 1000 min, prediction 80 min)

If IN: 80/1000 = 0.08 < 0.1 → recommend 500 min

If OUT: 120/1000 = 0.12 ≥ 0.1 → keep 1000 min

[2] M. S. Vallabhajosyula and R. Ramnath, "Towards practical, generalizable machine-learning training pipelines to build regression models for predicting application resource needs on HPC systems," in *Practice and Experience in Advanced Research Computing 2022: Revolutionary: Computing, Connections, You*, 2022, pp. 1–5.

© Bull SAS

Benchmarked methods

Three baseline ways to set the walltime, compared with SAFE on one illustrative job

1 Direct allocation

$$walltime = \hat{T}$$

Replace the request by the prediction. Largest reclaim, no safety margin. Run with linear, NN best and NN ensemble.

2 Walltime scaling

$$walltime = c \times elpl$$

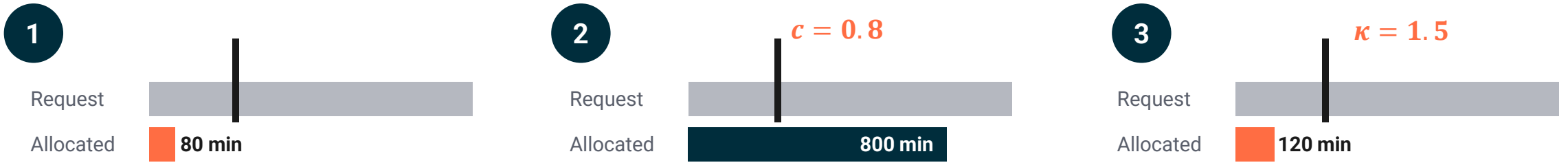
Multiply the request by a fixed factor c. Prediction-free reference; the black curve of the trade-off plot sweeps c from 0.01 to 1.

3 Prediction inflation

$$walltime = \kappa \times \hat{T}$$

Inflate the prediction by κ = 1.5 before using it as walltime, a safety factor from prior ML runtime-prediction work [2].

Example: true runtime = 200 min , request = 1000 min, prediction $\hat{T}$ = 80 min.



SAFE on the same job: 80/1000 < 0.1 → max(500, 80) = 500 min. The job is not cut.

© Bull SAS

How we measure safety and efficiency

Two metrics: reclaim unused walltime while keeping premature terminations low

Walltime gap reduction (%)

How much of the user's original over-allocation is removed?

$$\text{Original gap}_i = \text{requested walltime}_i - \text{observed runtime}_i$$
$$\text{New gap}_i = \text{recommended walltime}_i - \text{observed runtime}_i$$

$$Saved\ time_i = \begin{cases} original\ gap_i - new\ gap_i & if\ job\ is\ not\ cut \\ -\text{recommended walltime}_i & if\ job\ is\ cut \end{cases}$$

$$Gap\ reduction(\%) = \frac{\sum Saved\ time_i}{\sum original\ gap_i} \times 100$$

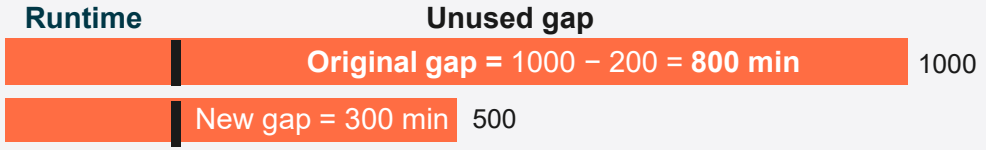
Jobs cut (%)

Safety metric: how often is the recommended walltime shorter than the true runtime?

$$Job\ cut = \frac{\#\ jobs\ with\ recommended\ walltime < runtime}{total\ \#\ jobs} \times 100$$

Example: Gap reduction.

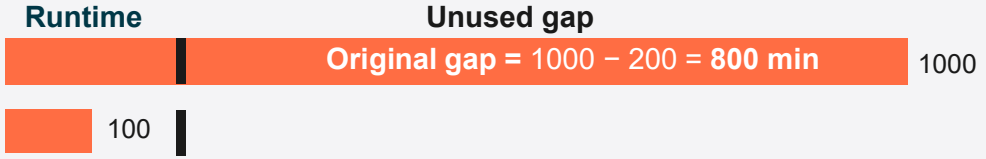
- runtime = 200 min
- request = 1000 min
- recommendation = 500 min



Saved time = 800 - 300 = 500 → Gap reduction = 100×500/800 = 62.5 %
Jobs cut = 0 %

Example: Jobs cut.

- runtime = 200 min
- request = 1000 min
- recommendation = 100 min



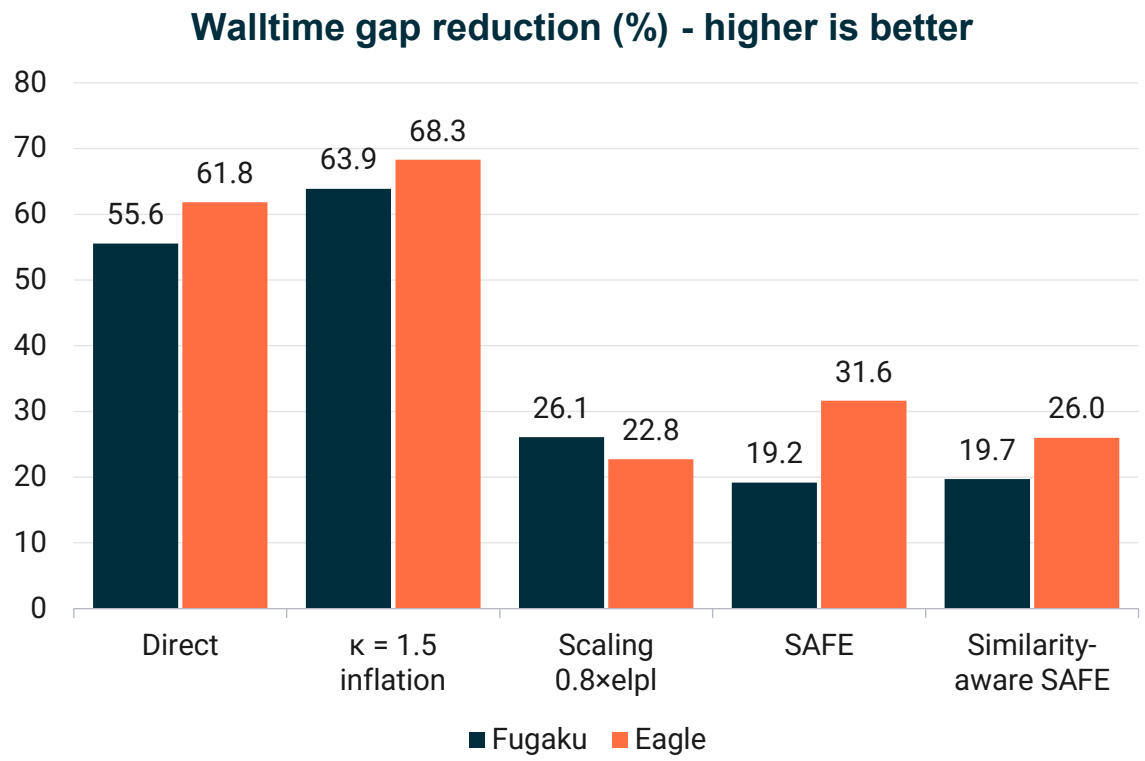
Jobs cut = 100 %
Saved time = -100 min → reduction = -12.5 %

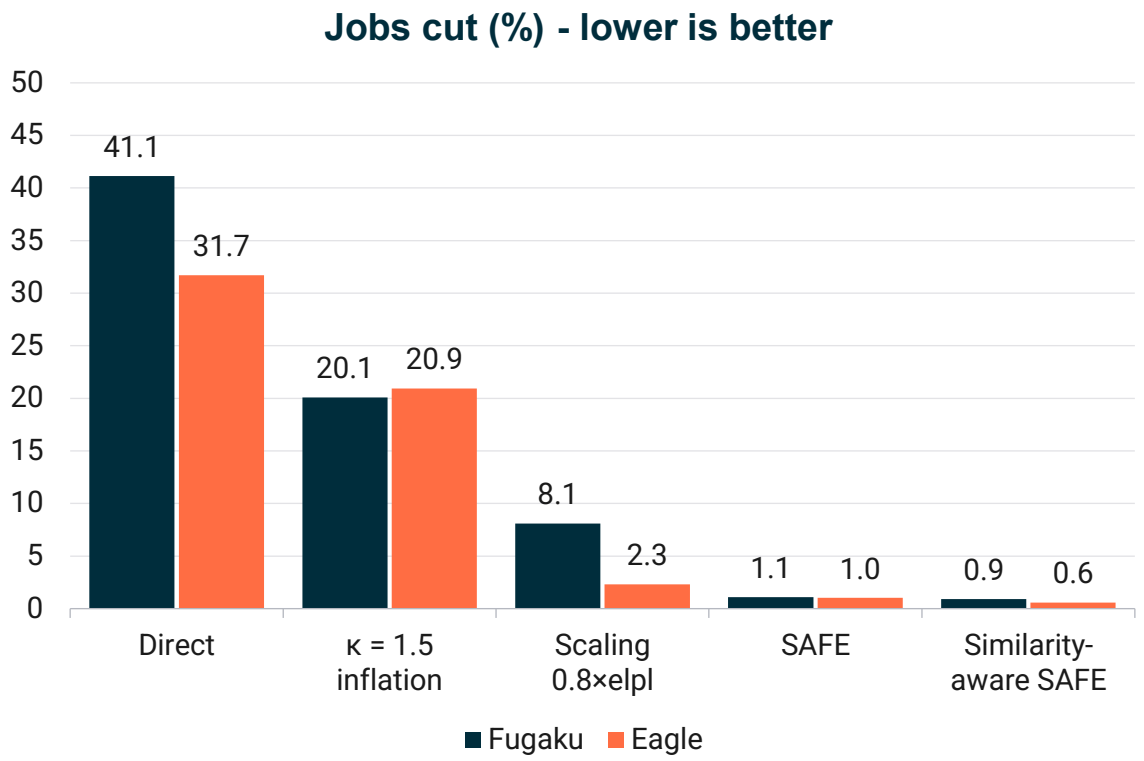
Goal: maximize gap reduction while minimizing job cuts.

© Bull SAS

Result 2 - safety vs efficiency

Direct prediction reclaims the most time but terminates 31–41% of jobs



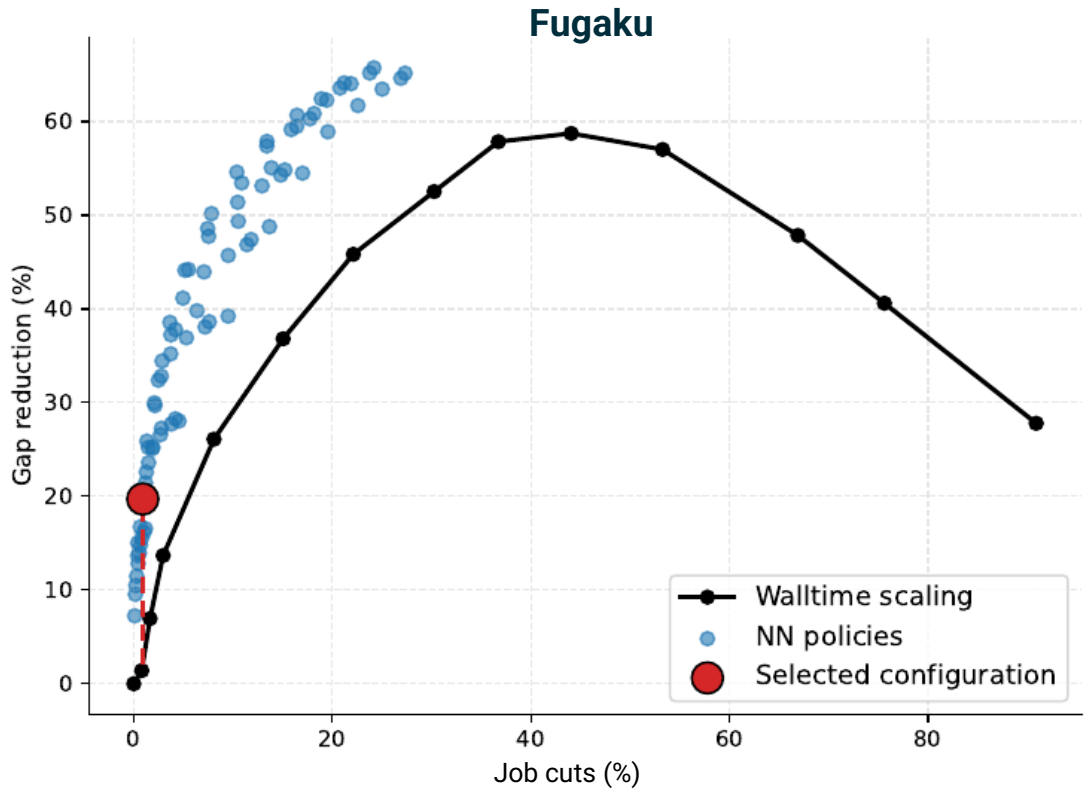


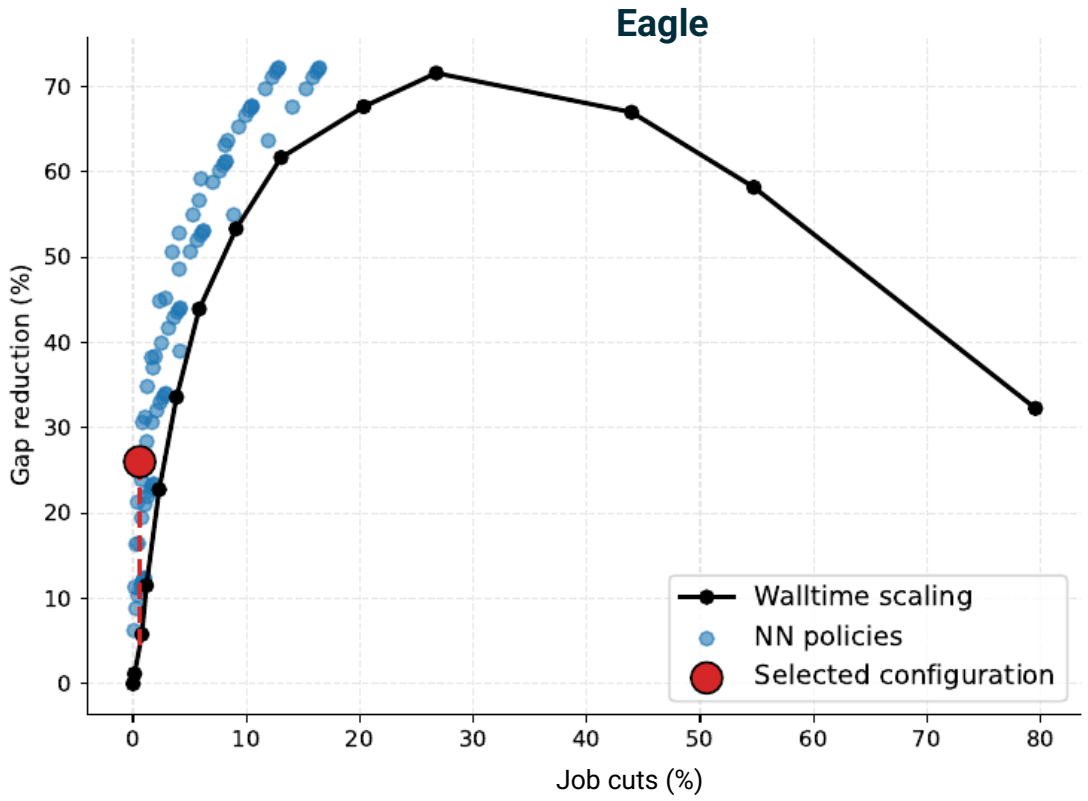
Only the SAFE variants keep job cuts at or below ~1% job cuts while still reclaiming 19–32% of the walltime gap.

© Bull SAS

Result 2 - trade-off curve

At comparable job-cut rates, prediction-informed policies beat blind scaling





Black: uniform scaling (0.01–1.00). **Blue:** similarity-aware SAFE (α, β ∈ 0.1…0.9). **Red:** selected configuration (α = 0.1, β = 0.5) → 0.91% cuts and 19.7% gap reduction (Fugaku), 0.58% and 26.0% (Eagle).

© Bull SAS

Limitations

What this study does and does not show

1

Hand-picked parameters

α, β and γ are fixed conservatively; no automatic calibration yet.

2

No scheduler simulation

We measure walltime gap and job cuts, not queue waiting time or backfilling efficiency.

3

Binary, exact-match similarity

A distance threshold in embedding space would give graded confidence.

4

Only two systems

Fugaku (PBS) and Eagle (SLURM); more systems would test generality.

© Bull SAS

Ongoing work - constraint-aware SAFE

Let the operator set a job-cut target; select the parameters automatically

Automatic parameters selection (α, β, γ)

Optimize SAFE parameters to satisfy a target maximum job-cut rate while maximizing walltime reduction.

Scheduler-level evaluation

Walltime gap is a proxy. We want to evaluate recommendations through scheduler replay, **measuring queueing** time, **utilization** and system-level impact.

Simpler non-linear predictors

Evaluate Random Forest, Extra Trees and HistGradientBoosting as alternatives to neural networks.

Feature engineering and encoding

Evaluate alternatives to SBERT, including structured/categorical features and lightweight encodings of submission-time metadata.

© Bull SAS

Takeaways

Runtime predictions can help set better walltimes safely

Direct prediction is unsafe

Using predicted runtimes as walltimes reclaims 55–62% of the gap but cuts 31–41% of jobs.

SAFE makes prediction deployable

Job cuts stay ≤ 1%: similarity-aware SAFE reaches 0.91% (Fugaku) and 0.58% (Eagle) while reducing the gap by 19.7% and 26.0% gap reduction.

Familiar jobs are predictable

MAE falls from 87.6 to 48.7 min (Fugaku) and 98.8 to 41.8 min (Eagle) for repeated workloads.

Acknowledgement: The **SEANERGYS** project receives funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No. 101177590. The JU receives support from the European Union's Horizon Europe research and innovation programme and Czech Republic, France, Germany, Greece, Italy, and Spain.

© Bull SAS



Thank you!

Hector.romerougalde@bull.com
youssef.el@bull.com
marlon.funk@bull.com
jesus.gorronogoitia@bull.com

Confidential information owned by Bull SAS, to be used by the recipient only. This document, or any part of it, may not be reproduced, copied, circulated and/or distributed nor quoted without prior written approval from Bull SAS.

© Bull SAS