ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Duration-Informed Workload Scheduler

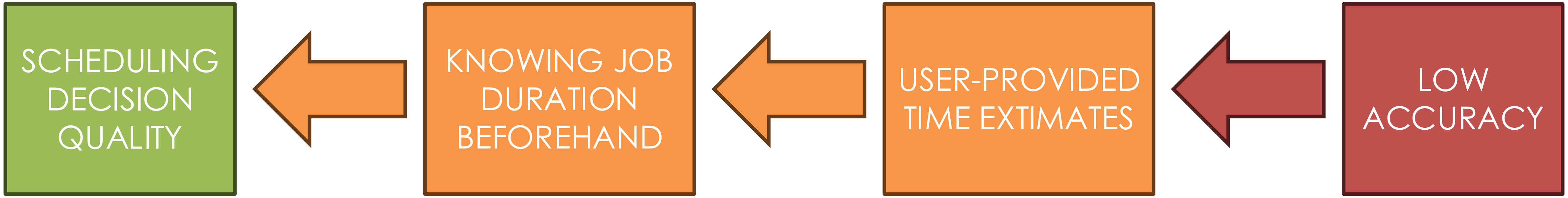
Daniela Loreti, Davide Leone and Andrea Borghesi

DISI - University of Bologna, Italy

MODA 2025

6th ISC HPC International Workshop on "Monitoring & Operational Data Analytics". June 13, 2025

HPC workload scheduling



- Scheduling decision quality is usually contingent on knowing job duration beforehand
- User-provided time extimations are known to be not very accurate, high overextimations

Goal and Contribution

We devise a ML-enhanced workload scheduler

- Time prediction module built via Machine Learning
- Devise a workload scheduler enhanced with these predictions
- Test our scheduler’s efficacy on real-life workload traces from a Tier-0 supercomputer

Runtime prediction

Duration-informed scheduler

Dataset for runtime prediction

Runtime prediction

- **PM100 [1]**, a large dataset of real-life job runs (elaboration of **M100 [2]**: a two-years-long data collection from MARCONI100 supercomputer hosted by the HPC centre CINECA)
- 628,977 elements (removed entries with missing values) with submission-time features for each job

Feature Name	Description
cpu	Number of CPU cores requested by the job.
mem (GB)	Amount of memory requested by the hob.
node	Number of nodes requested for the job.
gres/gpu	GPU resources requested by the job.
user_id	Identifier of the user submitting the job.
qos	Quality of Service level associated with the job.
time_limit	Maximum runtime allowed for the job.

[1] https://doi.org/10.5281/zenodo.8129257
[2] https://doi.org/10.5281/zenodo.7588815

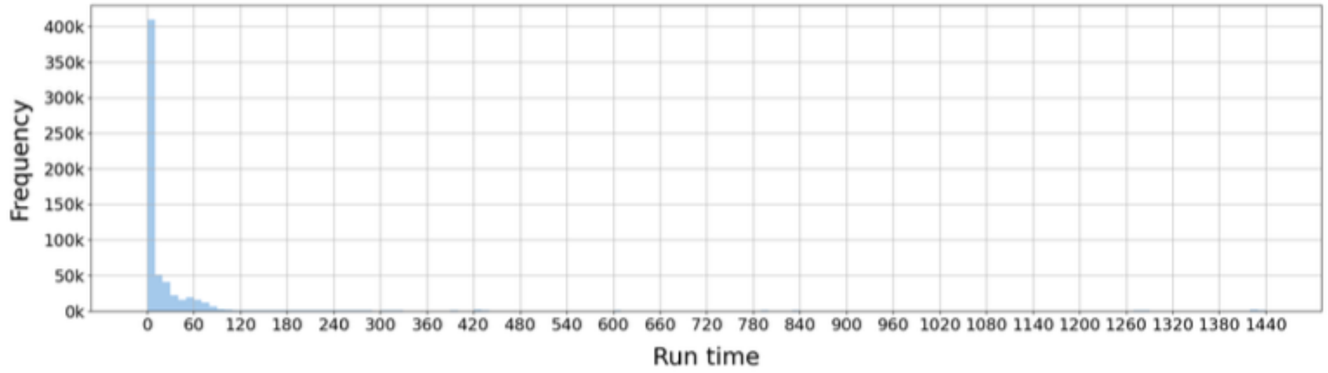
Brief statistical analysis

Runtime prediction

Dataset from a real Tier-0, production supercomputer → non-trivial workload to handle

	CPU	mem(GB)	nodes	GRES/GPU	user_id	QoS	time_limit	run_time
mean	121.379	236.068	1.693	5.630	110.895	0.051	1038.069	43.433
std	246.657	1008.594	6.961	27.927	118.594	0.368	506.318	168.719
min	1.000	0.098	1.000	1.000	0.000	0.000	1.000	0.017
25%	4.000	7.813	1.000	1.000	2.000	0.000	720.000	0.017
50%	80.000	230.000	1.000	4.000	93.000	0.000	1440.000	0.83
75%	128.000	237.5000	1.000	4.000	191.000	0.000	1440.000	22.700
max	32768.000	61500.000	256.000	1024.000	387.000	3.000	1440.000	1439.912

- high variability of cpu and memory metrics (large standard deviations and substantial range between the minimum and maximum values)
- pronounced skewness across most variables (few extreme **outliers** inflate the averages, creating a substantial gap between the mean and the more representative median values)



Prediction module

Runtime prediction

- Decision Tree Regressor (DT)
- Random Forest (RF)
- Gradient Boosting (GB)
- Fully Connected Neural Network (FCNN)
    - three hidden layers and dropout to prevent overfitting
    - Huber loss, (less sensitive to outliers)
    - number of layers and the number of neurons in each layer are the result of a non-exhaustive naïve grid-like search: 15 networks trained varying only these two parameters to find the best combination.

Prediction task: further setup details

Runtime prediction

Dataset split ratio of 70%/30%.

We evaluated DT, RF, GB and FCNN based on:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Coefficient of determination (R2)
- 95% confidence interval for prediction errors

Investigation of error characteristics categorized as:

- Overestimations
- Underestimations (most problematic)
- Exact estimations (down to half a second)

Prediction task: results for random split

Runtime prediction

	Decision Tree	Random Forest	Gradient Boosting	Neural Network
MAE	23.51	23.53	40.11	21.95
MSE	8001.99	7968.58	13060.00	9202.41
RMSE	89.45	89.27	114.28	95.93
R^2	0.72	0.72	0.54	0.68
Confidence interval (95%)	[0.00, 326.70]	[0.00, 325.60]	[0.00, 227.48]	[0.00, 344.92]
OVERESTIMATION				
Total cases	79.49%	79.59%	82.07%	80.79%
min error	0.01	0.01	0.01	0.01
max error	1431.00	1303.47	806.34	1624.70
avg error	14.76	14.72	24.35	12.39
error < 60 minutes	96.30%	96.26%	92.38%	97.98%
UNDERESTIMATION				
Total cases	20.02%	19.95%	17.93%	19.18%
min error	0.01	0.01	0.01	0.01
max error	1425.53	1425.54	1418.82	1427.30
avg error	58.85	59.23	112.26	62.23
error < 60 minutes	86.67%	86.34%	73.95%	83.33%
EXACT ESTIMATION				
Total cases	0.50%	0.47%	0.02%	0.02%
EFFECTIVENESS				
General	78.09%	78.23%	74.16%	78.72%
Valid prediction	97.94%	97.96%	92.79%	97.63%

Best models : RF and DT

Also, best R^2 value (better explanatory power)

Exact predictions are rare

Underestimations rather high

Prediction task: results with data augmentation

Runtime prediction

Adding the <u>average resource requested by each user</u>, i.e.,the mean values for the requested:

- number of CPUs
- Memory
- physical nodes
- GPUs
- time limit.

Slight error reduction for DT and RF

	Decision Tree	Random Forest	Gradient Boosting	Neural Network
MAE	22.24	22.26	26.01	20.53
MSE	7312.82	7275.61	8406.57	8623.19
RMSE	85.52	85.30	91.69	92.86
R^2	0.71	0.72	0.67	0.66
Confidence interval (95%)	[0.00, 307.77]	[0.00, 306.92]	[0.00, 291.46]	[0.00, 319.62]
OVERESTIMATION				
Total cases	79.90%	79.98%	80.57%	79.34%
min error	0.01	0.01	0.01	0.01
max error	1425.65	1425.66	1118.25	1470.79
avg error	13.97	13.97	16.20	12.26
error < 60 minutes	96.20%	96.15%	95.64%	97.93%
UNDERESTIMATION				
Total cases	19.69%	19.63%	19.41%	20.63%
min error	0.01	0.01	0.01	0.01
max error	1425.42	1425.41	1424.76	1427.48
avg error	56.25	56.48	66.72	55.36
error < 60 minutes	87.46%	87.29%	83.63%	86.07%
EXACT ESTIMATION				
Total cases	0.41%	0.39%	0.02%	0.03%
EFFECTIVENESS				
General	78.81%	78.81%	78.81%	78.81%
Valid prediction	98.51%	98.51%	98.51%	98.51%

Prediction task: results for time-consecutive split

Runtime prediction

Schedulers are requested to **estimate the runtime of future jobs given the jobs arrived in the past** →Random split may not represent a real-life case

- all the error values are better (average better predictions)
- R2 is worse → worse model performance
- Significantly less underestimations

	Decision Tree	Random Forest	Gradient Boosting	Neural Network
MAE	8.33	8.35	22.90	8.22
MSE	3438.32	3432.47	5086.70	3674.63
RMSE	58.64	58.59	71.32	60.62
R^2	0.62	0.62	0.44	0.60
Confidence interval (95%)	[0.00, 155.35]	[152.11]	[0.00, 164.60]	[0.00, 165.33]
OVERESTIMATION				
Total cases	94.40%	94.86%	95.99%	94.49%
min error	0.01	0.01	0.02	0.02
max error	1196.12	1184.39	722.10	1311.46
avg error	4.18	4.08	16.96	4.18
error < 60 minutes	99.44%	99.13%	98.90%	99.36%
UNDERESTIMATION				
Total cases	5.25%	5.13%	4.00%	5.50%
min error	0.01	0.01	0.02	0.02
max error	1425.71	1425.71	1399.33	1424.30
avg error	83.43	87.44	165.44	77.58
error < 60 minutes	81.69%	80.87%	67.75%	82.63%
EXACT ESTIMATION				
Total cases	0.35%	0.01%	0.00%	0.01%
EFFECTIVENESS				
General	94.22%	94.27%	93.40%	93.42%
Valid prediction	99.45%	99.37%	97.79%	98.90%

Prediction module

Runtime prediction

In general:

- the values predicted by the models are better at approximating the runtime than the user-provided *time limit* value:
  - When the models **overestimate** the runtime (on average around 95% of total cases), this results in almost a **98% improvement (on average)**
  - ML models **underestimate** the runtime on average around **5% of total cases**, while the *time_limit* value does so in just 1.4% of cases

Duration-informed workload scheduler (DIWS)

Duration-informed scheduling

Offline phase:

- Train a DT with historical job data (only once at the beginning of the algorithm execution)

Online phase:

- At submission time, the runtime of each job is predicted
- time requested by each job is set to the predicted value.
- submitted jobs with smaller predicted runtimes are given higher priority

In practice:

- Online SJF algorithm enhanced with runtime estimations derived through ML

DIWS Evaluation

Duration-informed scheduling

Implemented DIWS in Batsim[3] simulator

SETUP:

- Split the original dataset (~630,000 elements) into:
  - *df_sched*: last 24 hours (4,407 jobs)
  - *df_train*: the rest of the data (to train the DT)
- Compared **DIWS** with **EASY backfilling** in two different setups:
  - *Setup A*: 15,680 computing resources (=MARCONI100)
  - *Setup B*: 512 computing resources (to test the schedulers in stressing conditions)

[3] Dutot, P.F., Mercier, M., et al.: Batsim: a Realistic Language-Independent Resources and Jobs Management Systems Simulator. In: 20th Workshop on Job Scheduling Strategies for Parallel Processing. Chicago, US (2016)

DIWS Evaluation – further details

Duration-informed scheduling

Comparison based on:

- makespan (completion time of the last job)
- scheduling time (seconds spent in the scheduler)
- *mean and max waiting time* (time between job submission and its actual start time)
- mean and max turnaround time (time between job submission and its end)
- *mean and max slowdown* (turnaround/execution time.)

DIWS Evaluation – *Setup A (large infrastructure)*

Duration-informed scheduling

	DIWS	EASYBF	Improvement
makespan	86272.0068	86272.0024	+0.00%
scheduling time	37.8449	240.7396	-84.28%
mean waiting time	846.5391	953.3813	-11.21%
mean turnaround time	2351.1828	2458.0250	-4.35%
mean slowdown	2.3089	45.8519	-94.96%
max waiting time	17003.0928	12608.0384	+34.88%
max turnaround time	64657.0068	00657.0024	+0.00%
max slowdown	261.0818	12156.04.06	-97.85%

mean waiting time of a job is more than 11% lower

mean and max slowdown are significantly improved (-94.96% and -97.85%)

maximum waiting time is higher (+34.88%)

→ DIWS is better at estimating the jobs' duration beforehand, it is also able to identify how a few jobs are extremely more time-consuming than others and, accordingly, it changes their position further down the queue

DIWS Evaluation – *Setup B (constrained infrastructure)*

Duration-informed scheduling

	DIWS	EasyBF	Improvement
makespan	1029198.2116	1090869.2938	-5.99%
scheduling time	210.3460	194.5042	+7.53%
mean waiting time	127474.5570	163846.3107	-28.54%
mean turnaround time	128979.2008	165350.9545	-28.21%
mean slowdown	22785.2399	20097.1711	+11.80%
max waiting time	994211.2116	1026349.2598	-3.21%
max turnaround time	1024094.2116	1027933.2894	-0.37%
max slowdown	450511.6491	1026350.2601	-128.09%

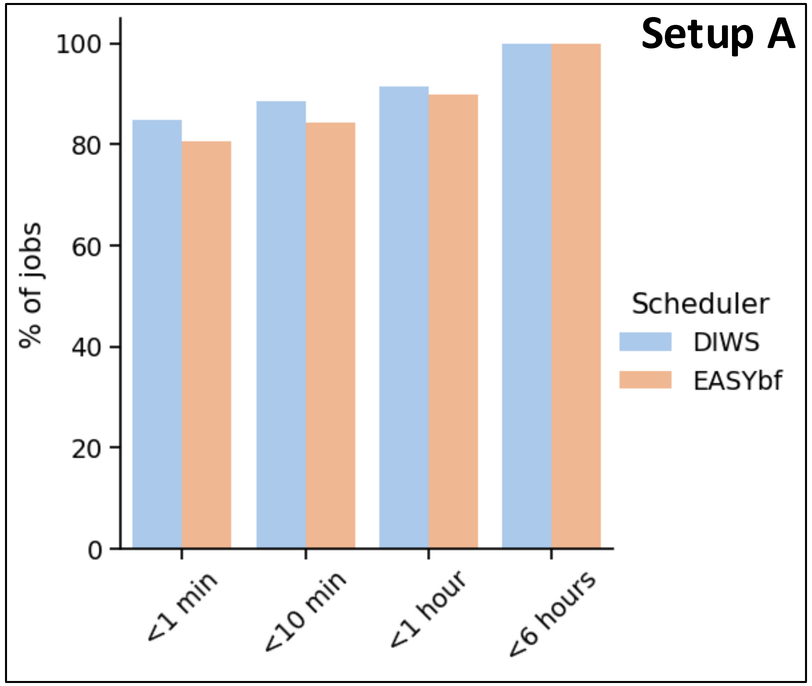
- mean waiting & turnaround time of a job are more than 28% lower
- mean slowdown shows a +11% increase

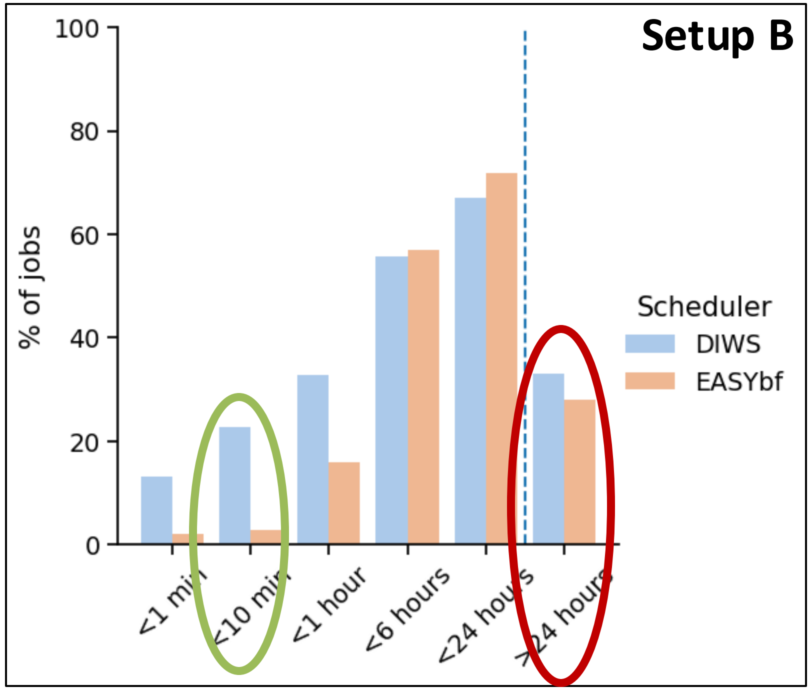
    → probably, SJF not best in this setting (does not consider the amount of resource requested)

DIWS Evaluation

Duration-informed scheduling

Percentage of jobs that wait less than arbitrarily chosen time intervals





- In large infrastructure setting, waiting time significantly improved for most jobs

- waiting time is less than 10 minutes for almost 8 times more jobs
- using the DIWS, the waiting time is very high (more than 1 day) for almost 5% more jobs than when using the EasyBF

Conclusion

Appling ML techniques to runtime prediction seems promising

- Prediction performance test on a real-life dataset of job runs, show the enhancement that ML can bring w.r.t. *time_limit metric* provided by users
- DIWS
  - Tests on Batsim show clear superiority w.r.t. EasyBF in reducing the average waiting time
  - However, better runtime predictions can negatively affect the waiting time of a non-negligible number of jobs that require much more computing time than others

Future/current work:

- Test different ML strategies (e.g. classification instead of regression)
- Improve DIWS turning SJF+Prediction into *"Smaller Energy First (SEF)"+Prediction*: consider the resource request together with predicted runtime
- SEF+Prediction can still be combined with backfilling…

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

QUESTIONS?

Duration-Informed Workload Scheduler

Daniela Loreti, Davide Leone and Andrea Borghesi

daniela.loreti@unibo.it