JEM: An AI-based engine workflow to predict simulation’s execution time on HPC cluster

JEM: An AI-based engine workflow to predict simulation's execution time on HPC cluster

1st Bartolomeo Vacchetti, 2nd Tania Cerquitelli
DAUIN
Polytechnic of Turin
Turin, Italy
{bartolomeo.vacchetti, tania.cerquitelli}@polito.it

3rd Vladi Nosenzo, 4th Enrica Capitelli
IvecoGroup
Turin, Italy

5th Luca Chiosso, 6th Manilo Trocano
Doit Systems
Turin, Italy

***Abstract*—Within the automotive industry, numerous simulations are essential for accurately modeling a vehicle's behavior and its components. However, these simulations entail an unpredictable duration for execution. Providing estimates for the required time enables users to organize their workload better and optimize resource utilization. Furthermore, the characteristics of submitted simulations can evolve, influenced by factors such as the component type, final product, car model, and other variables. In this dynamic context, staying abreast of these changes becomes imperative.**

This demo paper introduces the JEM tool, designed to accomplish two key tasks: (1) estimating the runtime of simulations and (2) allowing data scientists to monitor changes in data distribution and triggering model retraining when necessary. This tool results from a collaborative machine learning research project involving Iveco Group, Doit Systems, and Politecnico di Torino. Integrated into the Iveco group's HPC system, JEM facilitates resource allocation.

The recorded demo of the tool, accessible at https://youtu.be/iII-25FHqUo, provides a demonstration of how users can interact effectively with the JEM tool.

***Index Terms*—HPC system, Machine learning, Drift Detection**

I. Introduction

The automotive industry heavily relies on high-performance computing (HPC) systems, both on-site and in the cloud, to perform diverse simulations and analyses for vehicles and their components. Users submit daily requests through the Altair Access web portal [1]. These requests undergo processing by the queue manager, responsible for prioritizing jobs, allocating available resources, scheduling tasks, and generating log files. In the automotive context, the efficiency and effectiveness of job scheduling are critical, as they can either negatively impact time-to-market or positively influence the performance of the engineering department. An obstacle to organizing user workloads is the unknown execution time of a job, which can lead to unnecessary delays. Additionally, executing submitted jobs often faces delays due to the unavailability of immediate resources, thereby extending the time to achieve final results. After monitoring job scheduling for a few months, we identified certain inefficiencies. For instance, in cases where a user requires double the resources for a specific job compared to the demand, the job may experience prolonged waiting times for execution.

Machine learning algorithms can enhance resource allocation and utilization by suggesting the necessary resources to complete a job within a specified timeframe. Moreover, predicting runtime estimates can assist users in better organizing their activities. Additionally, machine learning could enhance the quality control of job submissions. If the estimated job execution significantly exceeds the user's expectations, it may indicate errors in job programming, warranting avoidance of job execution. Machine learning algorithms can optimize job scheduling, resulting in cost, time, and resource savings. With this objective, we present the JEMtool in this demo.

JEMprovides two data-driven services. Firstly, it offers the capability to estimate the execution time of a job. Secondly, it can monitor changes in data distribution and initiate model retraining if the degradation exceeds a predefined threshold.

For the first task, JEMrelies on the hierarchical classification model outlined in [2]. This model consists of three classifiers. The initial classifier performs a coarse-grained classification, dividing jobs into two subsets. Subsequently, the data undergoes processing by two classifiers for a fine-grained classification. Although the final model comprises four classes representing a broad spectrum of simulation execution times, a binary classification problem is addressed at each node.

To address potential changes in data distribution over time that may lead to model performance degradation, JEMimplements the methodology introduced in [3]. This approach calculates distribution distances and visually represents them to indicate significant changes in data distribution over time, prompting model retraining if necessary.

Lastly, a recorded demo illustrating user and data scientist interactions with the tool is accessible at https://youtu.be/iII-25FHqUo.

The paper is organized as follows. Section 2 focuses on related works, while section 3 focuses on the tasks that JEMcan address. The first part is on the job run time estimation, while the second is on the data drift estimation. The conclusions are drawn in Section 4.

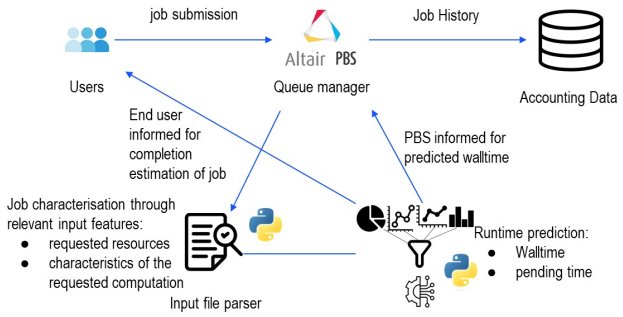

Fig. 1. JEM**'s service: Job execution time estimation**.

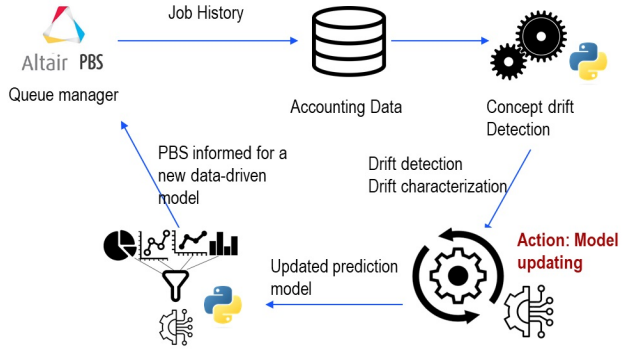

Fig. 2. JEM**'s service: Model degradation estimation**.

II. Related Works

The runtime prediction of a simulation is a task that has been addressed in different works, at different levels of complexity and using different strategies. One branch of earlier studies focuses on predicting whether or not a job will fail or complete its execution [4]–[10], while a more recent branch deploys machine and deep learning techniques to predict directly the simulation runtime. In [2], [11] machine learning algorithms, are used to make the runtime prediction, while in [12] the authors present a methodology to predict the runtime of a simulation based on transformers. In [13] in addition to predict the runtime the authors have deployed a machine learning based automatic workload manager, used by Amazon Redshift. While these works achieve impressive results, they miss a feature implemented in JEM, which is the concept drift module, and the consequent ability to be retrained to learn the new data distribution. Also in the filed of concept drift detection there have been significant studies in the last years. [3], [14]–[18]. Some are supervised and need a classifier to learn and decide what is drift and what is not, while other approaches are unsupervised as those shown in [15].Among the different available techniques we decided to rely on [3], which is able to detect drift in an unsupervised way on the overall batch of data and on every single label.

III. The JEMś services

JEM, integrated into the Altair Access Portal, offers two valuable data-driven services, shown in Figure 1 and Figure 2, respectively. The first module facilitates the estimation of job execution times by utilizing a trained hierarchical model. This aids users in effectively organizing their workloads based on the anticipated execution times of simulations.

As a secondary function, the system diligently monitors the degradation of the data-driven model over time. This proactive approach enables the timely identification of shifts in data distribution, prompting the system to re-train a new data-driven model accordingly.

To streamline the utilization of these data-driven services and their adaptation to diverse HPC architectures, we have proposed an automated workflow, detailed in [2]. This workflow encompasses an AI model evaluation process capable of selecting optimal parameters for each integrated algorithm. This ensures a user-friendly experience while maximizing the effectiveness of the proposed services across various computing environments.

JEMis designed for two types of users: end users, i.e., the engineers who submit new simulations, and data scientists, i.e., practitioners who are responsible for data-driven modeling and its performance over time and are therefore interested in monitoring the changes in the data distributions to identify when the model needs to be re-trained. End users interact with the JEM's service providing an estimate of the job's runtime, while data scientists are interested in exploiting the JEM's data drift estimation over time.

A. JEM*'s service: estimation of the job's runtime*

In Figure 1, the primary functions of the job execution time estimation service are depicted, illustrating enhancements to streamline job submissions. The process begins with the queue manager handling user-submitted jobs. Historical accounting data and job details are stored in a database, where data-driven algorithms analyze this information to construct a predictive model.

As described in [2], the adopted hierarchical approach employs a tree structure, shown in Figure 3. Each node in the structure features an XGBoost classifier [19], with the leaves representing final classes. The hierarchical model employs a two-layered classification process. The initial layer categorizes input data into coarse subsets, and the subsequent layer refines the classification using new classifiers for each subset. Although each classifier addresses a binary classification problem, the overall model considers four classes. This hierarchical model inputs job features and produces one of the four classes as output. Each label is a time range in which the estimation of the job's runtime falls.

The hierarchical model was trained using data provided by Iveco Group. As various solvers representing distinct software and tools were employed for diverse simulations, a dedicated classifier was trained for each solver. Each job is characterized by multiple input parameters, with the solver name being one such feature. While certain features are standard across all

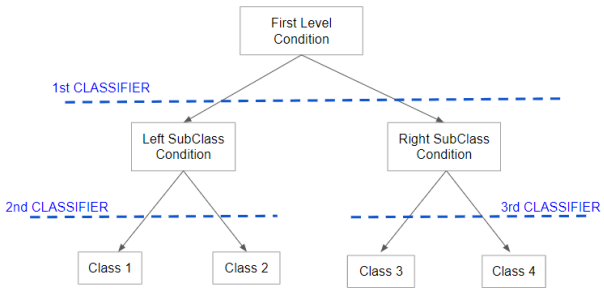

Fig. 3. **Hierarchical model architecture**.

solvers, others are unique to specific solvers or shared by multiple ones.

To ensure balanced classes before feeding the data into the classifier, we employed the K-means-constrained clustering algorithm [20]. This clustering approach assigned labels to each job based on its duration. Although the ideal output for the model would be an execution time, making regression a more fitting choice, the limited data availability prevented us from identifying a regression model that achieved satisfactory performance. Consequently, we treated the task as a classification problem.

When a new job is submitted, the input file characterizes the request regarding requested resources and simulation specifics. The prediction model then estimates the runtime, informing PBS of the predicted runtime and notifying end users of the expected job completion time.

Figure 4 illustrates the graphical interface integrated into the Altair Access Portal to obtain job simulation runtimes predictions. When users submit a new job, they are required to upload a file containing simulation parameters and specify the desired number of cores for simulation execution. Users who want to estimate a job's running time can utilize the "Predict Walltime" checkbox on the Altair submission portal.

The job parameters are then extracted, and depending on the solver employed, they are forwarded to a hierarchical model trained on all past simulations utilizing the exact solver. The hierarchical model processes the job parameters and predicts a class. The end-user receives the predicted class's upper and lower bounds as output. Figure 5 provides a snippet of the submission interface, offering details on the "Predict Walltime" option. The user can select different simulations to feed the prediction model during the demo and analyze the output provided by JEM.

The interface is an extension of the one used by the user to submit a new job. It contains the option "Prediction Walltime," which estimates the simulation run time. The prediction can help users to organize their workload according to the job's execution time. More importantly, users can decide to change some parameters, such as the number of cores, and see if the job will require less time to be executed before submitting it. All the different options can be tested during the demo.

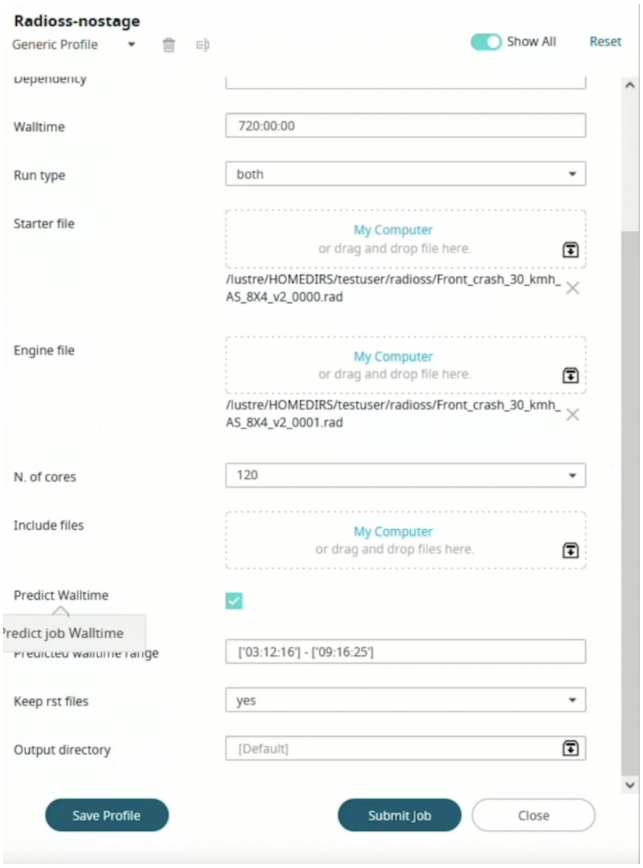

Fig. 4. **Graphical interface for job submission and run-time prediction**.

B. JEM*'s service: data drift estimation*

A concept drift management module was integrated into JEM to mitigate potential performance degradation of the predictive model caused by temporal shifts in the input data. This service identifies and monitors the drift and initiates a new training phase, as shown in Figure 2. In addition, it ensures the ongoing adaptability and precision of the predictive model.

The methodology relies on an adaptation of DriftLens [3], an unsupervised data drift detection approach that leverages the FID score [21] for calculating the distance between embedding distributions (training data versus real-time data). In our specific scenario, where embeddings are absent, the FID score was used to measure the distance between data distributions. To achieve this, each data feature underwent a transformation into a normal distribution to simulate a multivariate Gaussian distribution.

Then JEM proceeds as follows: i) it characterizes the data distribution of the training data and calculates a threshold, and ii) it evaluates the data distribution of the new incoming data with respect to the threshold. The user can set the threshold experimentally. JEM suggests whether the real-time data has changed consistently over time by comparing the distribution distance of the new incoming jobs with the threshold. The proposed interface provides a graph over time that allows

Fig. 5. **Snippet of the Predicting request**.

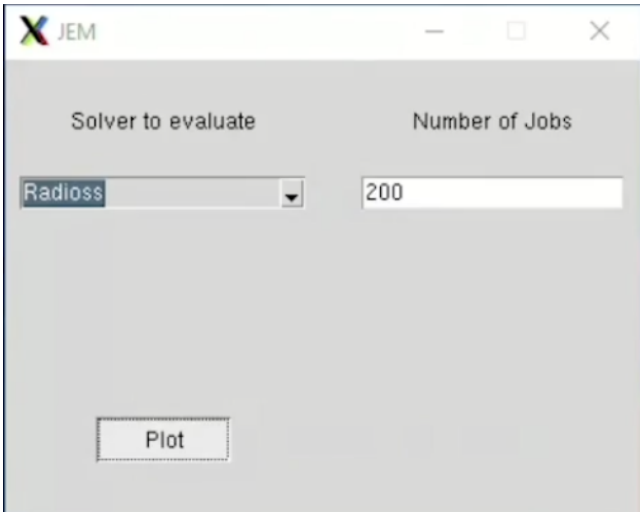

Fig. 6. JEM**'s service: data drift estimation: Configuration settings**.

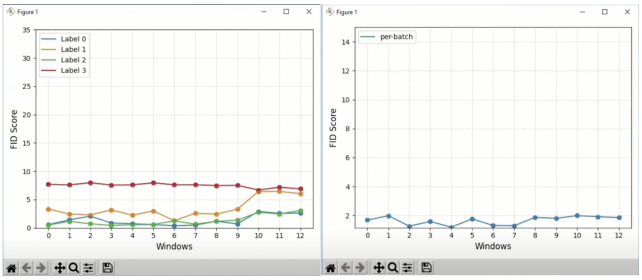

Fig. 7. JEM**'s service: data drift estimation - Solver: Radioss, #jobs: 200** .

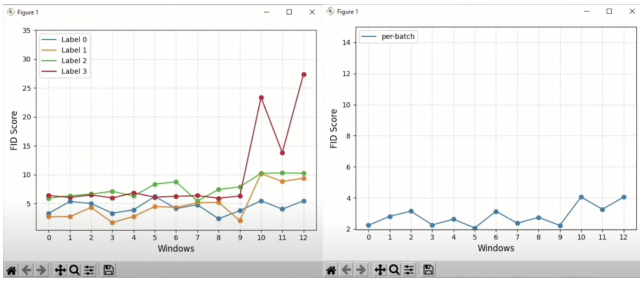

Fig. 8. JEM**'s service: data drift estimation - Solver: Abaqus, #jobs:150**.

the user to quantify how much the data distribution has changed compared to the training data. If the distribution distance between the new data and the training data exceeds the threshold, this indicates the presence of a concept deviation in the upcoming data and thus JEMneeds to retrain the model.

If data scientists want to analyze to what extent the data distribution of the newly submitted simulations differs from the historical simulations, the graphical interface shown in Figure 6 can be used. During the demo, the user can try out the software by setting a specific solver from the available solvers and a particular number of jobs to be analyzed. For each configuration setting, two diagrams are provided as output. The diagrams in Figure 7 and Figure 8 show the output provided by JEMin two different settings. The FID per class of the new incoming data is shown in the last three timestamps.

Figure 7 shows the most recent 200 jobs executed using the *Radioss* solver. In this scenario, the jobs exhibit subtle variations in their data properties without significant changes. On the left side of the figure, there is a slight increase in the data associated with class labels 0, 1, and 2, whereas the plot for label 3 remains unchanged. On the right, the graph illustrates the distribution distance plotted for the entire batch of data without segregating the labels.

The second scenario is characterized by 150 jobs executed using the *Abaqus* solver. The data distribution undergoes significant alterations compared to previous instances. Notably, label 3 experiences the most pronounced change in distribution, followed by label 1, while labels 0 and 2 are only mildly impacted or remain unchanged from their usual trend. Moreover, the overall data distribution has shifted slightly compared to historical patterns, as illustrated in the right plot in Figure 8.

IV. Conclusions

This paper introduces JEM, an integrated tool within the Altair Access Portal, designed to accomplish two primary objectives: (i) estimating the runtime of a simulation at the time of job submission and (ii) monitoring changes in data distribution, with the capability to retrain hierarchical models when necessary. Moreover, users have the option to monitor data distribution changes through JEMindependently.

As future work, we aim to (i) enhance and broaden the applicability of JEMby incorporating it into heterogeneous High-Performance Computing (HPC) environments, be it in the cloud or a blended setup involving both on-premises and cloud resources; (ii) diversify the array of solvers (software employed for simulations) to evaluate the potential of JEMin evolving into a more versatile and general-purpose model-based framework.

Furthermore, we are also working to expand the dataset utilized for analysis, thereby refining the accuracy of estimating simulation execution time. Achieving these objectives has the prospect of bolstering the capabilities of JEM, subsequently enhancing the support it offers to its designated users (i.e., engineers and data-scientists) in their daily and weekly professional activities.

References

[1] Altair access web portal. https://altair.com/access.

[2] Paolo Bethaz, Bartolomeo Vacchetti, Enrica Capitelli, Vladi Nosenzo, Luca Chiosso, and Tania) Cerquitelli. Predicting job execution time on a high-performance computing cluster using a hierarchical data-driven methodology. In *Proceedings of the Workshops of the EDBT/ICDT 2022 Joint Conference*, Edinburg, UK, 2022. Proceedings of the Workshops of the EDBT/ICDT 2022 Joint Conference.

[3] Salvatore Greco and Tania Cerquitelli. Drift lens: Real-time unsupervised concept drift detection by evaluating per-label embedding distributions. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 341–349, 2021.

[4] Chunhong Liu, Liping Dai, Yi Lai, Guinbing Lai, and Wentao Mao. Failure prediction of tasks in the cloud at an earlier stage: a solution based on domain information mining. *Computing*, 102:2001–2023, 2020.

[5] M. Jassas and Q. H. Mahmoud. Failure analysis and characterization of scheduling jobs in google cluster trace. In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, pages 3102–3107, Omni Shoreham, United States, 2018. IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society.

[6] C. Liu, J. Han, Y. Shang, C. Liu, B. Cheng, and J. Chen. Predicting of job failure in compute cloud based on online extreme learning machine: A comparative study. *IEEE Access*, 5:9359–9368, 2017.

[7] A. Rosà, L. Y. Chen, and W. Binder. Failure analysis and prediction for big-data systems. *IEEE Transactions on Services Computing*, 10(6):984–998, 2017.

[8] P. Li, B. Zhang, Y. Weng, and R. Rajagopal. A sparse linear model and significance test for individual consumption prediction. *IEEE Transactions on Power Systems*, 32(6):4489–4500, 2017.

[9] S. Ganguly, A. Consul, A. Khan, B. Bussone, J. Richards, and A. Miguel. A practical approach to hard disk failure prediction in cloud platforms: Big data model for failure management in datacenters. In *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 105–116, Oxford, United Kingdom, 2016. 2016 IEEE Second International Conference on Big Data Computing Service and Applications.

[10] J. M. Navarro, G. H. A. Parada, and J. C. Dueñas. System failure prediction through rare-events elastic-net logistic regression. In *2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation*, pages 120–125, Madrid, Spain, 2014. IEEE.

[11] Kenneth Lamar, Alexander Goponenko, Omar Aaziz, Benjamin A Allan, James M Brandt, and Damian Dechev. Evaluating hpc job run time predictions using application input parameters. In *Proceedings of the 17th ACM International Conference on Distributed and Event-Based Systems*, DEBS '23, page 127–138, New York, NY, USA, 2023. Association for Computing Machinery.

[12] Fengxian Chen. Job runtime prediction of hpc cluster based on pc-transformer. *The Journal of Supercomputing*, 79:1–27, 06 2023.

[13] Gaurav Saxena, Mohammad Rahman, Naresh Chainani, Chunbin Lin, George Caragea, Fahim Chowdhury, Ryan Marcus, Tim Kraska, Ippokratis Pandis, and Balakrishnan (Murali) Narayanaswamy. Auto-wlm: Machine learning enhanced workload management in amazon redshift. In *Companion of the 2023 International Conference on Management of Data*, SIGMOD '23, page 225–237, New York, NY, USA, 2023. Association for Computing Machinery.

[14] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2019.

[15] Rosana Noronha Gemaque, Albert França Josuá Costa, Rafael Giusti, and Eulanda Miranda dos Santos. An overview of unsupervised drift detection methods. *WIREs Data Mining and Knowledge Discovery*, 10(6):e1381, 2020.

[16] Hanqing Hu, Mehmed Kantardzic, and Tegjyot S. Sethi. No free lunch theorem for concept drift detection in streaming data classification: A review. *WIREs Data Mining and Knowledge Discovery*, 10(2):e1327, 2020.

[17] Firas Bayram, Bestoun S. Ahmed, and Andreas Kassler. From concept drift to model degradation: An overview on performance-aware drift detectors. *Know.-Based Syst.*, 245(C), jun 2022.

[18] Pingfan Wang, Hang Yu, Nanlin Jin, Duncan Davies, and Wai Lok Woo. QuadCDD: A quadruple-based approach for understanding concept drift in data streams. *Expert Systems with Applications*, 238:122114, 2024.

[19] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[20] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[21] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017.