Prediction of Reservoir Simulation Jobs Times Using a Real-World SLURM Log

Alan L. Nunes[2*], **Felipe A. Portella**[1*], **Paulo J. B. Estrela**[1], **Renzo Q. Malini**[1]
Bruno Lopes[2], **Arthur Bittencourt**[2], **Gabriel B. Leite**[2], **Gabriela Coutinho**[2],
Lúcia Maria de Assumpção Drummond[2]

{felipeportella,paulo.estrela,renzo}@petrobras.com.br
{alan_lira,athurbittencourt,gabriel_bittencourt,barrosgabriela}@id.uff.br
{lucia,bruno}@ic.uff.br

[1]Petróleo Brasileiro S.A. (PETROBRAS) – Rio de Janeiro – RJ – Brazil

[2]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brazil

Abstract. *Modeling petroleum field behavior provides crucial knowledge for risk quantification regarding extraction prospects. Since their processing requires significant computational power and storage capabilities, oil companies run reservoir simulation jobs on high-performance computing clusters. Efficiently using machine learning algorithms in job schedulers to predict the incoming job execution time can increase the effectiveness of cluster resources, such as improving its resource usage rate and reducing the job queue time. This paper introduces a novel and robust predictor, based on SLURM logs from Petrobras, that classifies with more than 74% accuracy the duration time interval of reservoir simulation jobs. The results reveal that our model exceeded the performance of the EASY++ algorithm-based estimator.*

1. Introduction

Petroleum Reservoir Simulation is a fundamental tool in the Oil and Gas (O&G) industry, helping to minimize risks and optimize decision-making processes during the development of petroleum reservoirs by reproducing the production history and forecasting future production, providing valuable insights into the behavior of oil, gas, and water within reservoir fields over time. These simulations, conducted using reservoir models, enable engineers to explore multiple scenarios more cost-effectively and efficiently than through real-world operations [Portella et al. 2022]. Reservoir models generally use three-dimensional grids to represent the physical reservoir in thousands or millions of cells and rely on mathematical equations derived from physical principles such as mass conservation, thermodynamic equilibrium, heat transfer, and Darcy's law for flow in porous media [Coats 1982]. Due to the large size and complexity of numerical models of real-world applications, reservoir simulations require powerful computational resources. Therefore, they are typically performed on High-Performance Computing (HPC) clusters, which explains why multiple supercomputers owned by oil companies are on the TOP500 list of the most powerful supercomputers in the world.

[*]Corresponding authors

Predicting the execution time of reservoir simulation jobs in the oil and gas sector is particularly critical to optimize computational resources and maximize productivity. Multiple realizations or scenarios must be performed within reservoir simulations to address the uncertainties associated with reservoir characterization and fluid behavior. The speed at which these simulations can be executed assumes prominent significance, as it enables a greater number of realizations, capturing inherent uncertainties and enhancing decision-making processes. In addition to optimizing HPC resource allocation, predicting the execution time of reservoir simulation jobs also contributes to better scheduling policies for those cluster queue manager systems, such as SLURM (Simple Linux Utility for Resource Management) [Yoo et al. 2003]. SLURM is an open-source job scheduler and resource management system designed to handle clusters with thousands of nodes and has been widely adopted as a workload manager at numerous HPC sites.

Cloud computing has gained popularity in the industry due to its scalability and cost-efficiency. By accurately estimating execution time, O&G companies can optimize the allocation of on-premise and cloud resources, reducing unnecessary costs associated with excessive or extended resource usage. However, it is not a simple task, as job time is influenced by several factors, such as the complexity of the reservoir modeling, the size of the simulation model, the underlying target computer architecture, and other parameters related to the software and the hardware.

Recent work [Kuchnik et al. 2019] has shown that the use of learning algorithms in job schedulers potentially increases the effectiveness of the HPC cluster. Nonetheless, deployments have, so far, been limited because three challenges have not received sufficient attention in the literature: first, the lack of data diversity can adversely affect the design of prediction systems, such as impractical feature engineering, unreliable prediction performance, and inconspicuous overfitting. Secondly, workload changes can negatively affect predictor performance, as accuracy degrades over time due to the nonstationarity (job profiles changing), which is typical across most job traces. Finally, aiming for high prediction accuracy alone does not guarantee dramatically better end-to-end performance.

This work aimed to create a novel runtime predictor for reservoir simulation jobs. To achieve this, we utilized statistical and machine learning techniques to extract as much information as possible from reservoir simulation workload records. The logs were obtained from over three hundred engineers at Petrobras, a globally recognized Brazilian energy company. With real-world data, it is possible to execute an in-depth investigation to identify patterns and trends that will improve job time prediction. Thus, the main contributions of this article are the following:

I. Employment of statistical and machine learning techniques to extract relevant features from reservoir simulation workload records;
II. Development of a novel and robust machine learning predictor that accurately classifies the duration time interval of reservoir simulation jobs based on SLURM logs;
III. Implementation of the EASY++ scheduling algorithm as a runtime estimator, considering the relevant features; and
IV. Experimental analysis of our proposed model, comparing the results to EASY++.

The remainder of this paper is organized as follows. Section 2 covers related works. Section 3 presents our execution time predictor for reservoir simulation jobs, while Section 4 shows the experimental results. Finally, Section 5 concludes this article and proposes future directions.

2. Related Works

Previous research has addressed the issue of user competence in determining the resources required for High-Performance Computing (HPC) jobs. This matter is far from trivial, and users are under pressure to overestimate predictions of memory, CPUs, and time to avoid their jobs being killed by the scheduler due to insufficient resources. Overestimating job resources usually leads to resources being wasted, lower throughput, and longer user response times. To solve this problem, [Tanash et al. 2019] developed a supervised machine learning model built into the SLURM resource manager simulator to predict the memory and time required to perform the computation. The adjusted model dramatically helped reduce the computational response time for larger jobs. Another study [Witt et al. 2019] focused on Predictive Performance Modeling (PPM) to estimate future metrics such as execution time, memory, and wait times, without requiring workload modifications, and achieved similar results, identifying several issues that need more in-depth research.

Scheduling policies determine the order in which the jobs are executed, affecting the system's performance. The most commonly used scheduling algorithm for HPC jobs is FCFS (First-Come, First-Served) with Backfilling, as introduced in the EASY scheduler [Lifka 1998]. The scheduler goes through the job queue in FCFS order and starts jobs until it finds a job that can not begin immediately. Then, it makes a reservation for this job at the earliest predictable time and starts backfilling the job queue in FCFS order using any other jobs that do not delay that reservation. [Tsafrir et al. 2007] presented an advancement over the EASY scheduler, named EASY++, which became a widely adopted benchmark for evaluating new algorithms in this field. The main addition in EASY++ is a predictor that considers the elapsed time of two most recently submitted jobs by a given user to predict the job execution time. Despite its relative simplicity, this history-based system proved to be very successful. [Gaussier et al. 2015] investigated whether learning techniques are worth using on job running times for improving existing scheduling algorithms. They proposed a new cost function for prediction and ran simulations based on actual workload logs for the most popular variants of backfilling. The results show an average gain of 28% compared to the EASY policy and 11% on average compared to the EASY++ policy. Later, [Gaussier et al. 2018] proposed reordering the submission queues under EASY using two methods. The first, based on a simulator, reduced the average waiting times of the baseline FCFS ordering policy by 11 to 60%, and the second one, based on a multi-armed bandit algorithm, had an improvement factor of 8 to 46%.

Other papers have tackled the real-time scheduling problem. [Cheng et al. 2022], for instance, used deep reinforcement learning to develop a real-time task scheduler in the cloud. Experimental results showed that this approach outperformed commonly used real-time scheduling algorithms, providing greater efficiency.

These related works present different approaches and techniques to tackle job scheduling challenges in high-performance systems, aiming to improve resource utilization and optimize system performance. However, to the best of our knowledge, none of them was adjusted to predict the execution time of reservoir simulator jobs in an actual energy industry scenario with more than five hundred thousand jobs executed daily.

3. Reservoir Simulation Jobs Runtime Predictor

Predictive analytics is the process responsible for extracting information from large data sets to make predictions and estimates about future outcomes [Larose and Larose 2015]. The foremost objective of predictive modeling is not to understand why something will occur (or not) but to accurately project the chances that something will happen (or not) by analyzing relevant historical data [Kuhn and Johnson 2013]. *Classification* and *Clustering* are among the most common tasks during this process. The *Classification* task approximates the value of a categorical (nominal) target variable using a set of numeric or categorical predictor variables. Understanding two- and three-dimensional relationships in the data can be done through graphs and plots. However, classification models are sometimes based on many different predictors, demanding multidimensional plotting. The *Clustering* task packs records, observations, or cases into classes of matching objects. A cluster is a collection of data similar to each other and different from those belonging to other clusters. Clustering does not attempt to classify, estimate, or predict the value of a target variable. Instead, clustering algorithms aim to segment the whole data set into relatively homogeneous subgroups or clusters, where the similarity of the records within a cluster is maximized, minimizing the similarity with data outside it.

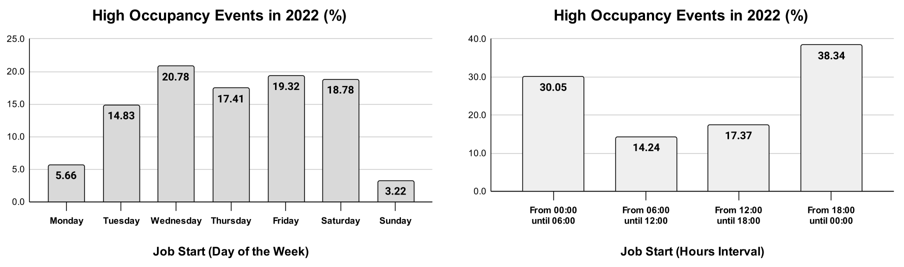
Having a prediction model to estimate the execution time of a workload under certain conditions, we could determine the most suitable computing resource configuration for a given reservoir simulation job. This section introduces the accomplished exploratory analysis of the SLURM logs and how its outcome was used to build our runtime predictor strategy. First, we investigated the peak usages of a cluster in terms of days of the week and hours of the day, seeking occupancy events spread in thirty-minute observation windows (Subsection 3.1). Next, we performed a visual *Clustering* analysis to get the most frequently submitted job profiles (Subsection 3.2). Finally, through the *Features Selection* technique, we selected relevant attributes to be used by our predictor (Subsection 3.3).

3.1. Preliminary Analysis of the SLURM Log

The main objective, as an initial step, was to identify peak usage occurrences considering the highest amount of simultaneous jobs. This analysis aimed to identify patterns over days of the week and month and their respective highest amounts of requested CPU cores. Therefore, the usage behavior of the cluster was verified by analyzing the simulation jobs submitted to the SLURM scheduler throughout the year 2022 at Petrobras, as it was the only full year from our three-year log.

We defined three cluster occupancy categories using continuous observation windows of thirty minutes: low, medium, and high. The first category represented intervals with up to 20,000 CPU cores usage. The second one described intervals with utilization between 20,000 and 40,000 CPU cores. The third one expressed intervals with utilization greater than 40,000 CPU cores. For high occupancy events in the cluster, we performed an additional analysis to group their occurrences into days of the week and hours of the day. The latter grouping was arranged into four predefined six-hour intervals: from 00:00 until 06:00, from 06:00 until 12:00, from 12:00 until 18:00, and from 18:00 until 00:00.

Considering the total number of thirty-minute observation windows, we surveyed 4,517 low occupancy events, 10,953 medium occupancy events, and 2,050 high occupancy events in 2022. Figure 1 shows that the high occupancy events in 2022 mainly occurred when jobs started on Wednesday (20.78%) and from 18:00 until 00:00 (38.34%).



(a) (b)

Figure 1: High occupancy events of the cluster in 2022 regarding (a) day of the week and (b) hours interval.

3.2. Clustering of Submitted Job Profiles

To further analyze the submitted jobs and extract relevant data attributes, we conducted a visual Clustering investigation. We generated annual *Heat maps* from the SLURM logs from May 4, 2021, to April 4, 2023, using *Kibana* [Gupta 2015], an open-source browser-based visualization tool mainly used to analyze more than three million jobs. We considered three types of submitted job profiles for each year: i) *Profile 1* – jobs that required up to 15 CPUs and ran for up to 5 hours; ii) *Profile 2* – jobs that required 20 CPUs and ran for up to 15 hours; and iii) *Profile 3* – jobs that required 40 CPUs and ran for up to 15 hours. Table 1 summarizes their year distribution regarding the number of occurrences and relative frequency.

Table 1: Submitted job profile types yearly distribution.

Job Profile	Year 2021*		Year 2022		Year 2023*	
	Jobs	**%**	**Jobs**	**%**	**Jobs**	**%**
Profile 1	164,119	26.68	453,448	24.19	119,897	20.37
Profile 2	239,935	39.01	667,319	35.59	171,417	29.12
Profile 3	169,247	27.51	597,367	31.86	170,614	28.98
Remainder	41,818	6.8	156,650	8.36	126,793	21.53
Total	**615,119**	**100.0**	**1,874,784**	**100.0**	**588,721**	**100.0**

We noticed that the evaluated job profiles represented the majority of jobs submitted each year. Jointly, they expressed 93.2%, 91.64%, and 78.47% of all jobs for the years 2021, 2022, and 2023, respectively. It is worth remembering that the 2023 year is underway at the time of writing, and the relative percentage of these three job submission profiles is expected to increase throughout the year.

3.3. Attributes Selection for the Machine Learning Models

The better the selection of attributes, the better the quality of the model predictions. The exploratory analysis steps of the SLURM logs (Subsections 3.1 and 3.2) have enabled us to understand the variations in the cluster occupancy rate and the specific characteristics of the executed jobs. Such acquired knowledge allowed to define a suitable starting subset of three nominal (categorical) attributes to be employed by the machine learning models, as follows: i) *job_start_day_of_week_class*, derived from the *@start* field; ii) *job_start_hour_of_day_class*, derived from the *@start* field; and iii) *job_duration_class*, derived from the *elapsed* field.

The *@start* field has the timestamp at which a job started running, and the *elapsed* field holds the duration of this job in seconds. The possible categories of the first two attributes were categorized by the earlier defined groupings (Subsection 3.1). Our preliminary analysis of the SLURM log and the submitted job profiles types led us to establish 23 possible categories for *job_duration_class*, as shown in Table 2.

Table 2: Possible categories for the *job_duration_class* attribute.

Category	Meaning	Category	Meaning
0	The job lasted up to 1 minute.	12	The job lasted between 6 and 7 hours.
1	The job lasted between 1 and 5 minutes.	13	The job lasted between 7 and 8 hours.
2	The job lasted between 5 and 10 minutes.	14	The job lasted between 8 and 9 hours.
3	The job lasted between 10 and 15 minutes.	15	The job lasted between 9 and 10 hours.
4	The job lasted between 15 and 30 minutes.	16	The job lasted between 10 and 11 hours.
5	The job lasted between 30 and 45 minutes.	17	The job lasted between 11 and 12 hours.
6	The job lasted between 45 minutes and 1 hour.	18	The job lasted between 12 and 24 hours.
7	The job lasted between 1 and 2 hours.	19	The job lasted between 24 and 48 hours.
8	The job lasted between 2 and 3 hours.	20	The job lasted between 48 and 72 hours.
9	The job lasted between 3 and 4 hours.	21	The job lasted between 72 and 96 hours.
10	The job lasted between 4 and 5 hours.	22	The job lasted more than 96 hours.
11	The job lasted between 5 and 6 hours.	–	– – – – – – – – – – – – – – – – – –

However, raw logs contain a mixture of data, including fields that can be irrelevant and unfeasible for training prediction models. In addition to the previous exploratory analysis, the *Ranker* tool from the *Weka software package* [Hall et al. 2009] was used to evaluate the worth of each attribute by measuring the gain of information concerning the target attribute (particularly the *job_duration_class*). Weka is a collection of machine learning algorithms for data mining tasks and includes tools for data preparation, classification, regression, clustering, association rule mining, and visualization.

Moreover, the cluster managers initially assisted in the most appropriate attribute selection based on their experience, and then the final selection was the outcome of the information gain ranking tool. Table 3 summarizes the remaining selected attributes with their respective worth score.

Table 3: Remaining selected attributes for the machine learning models.

Attribute	Worth Score
work_dir_parent (derived from the *work_dir* field)	2.4289
username	1.3732
account	0.8315
employed_simulator (derived from the *script* field)	0.7169
total_cpus	0.4492
nodes_prefix (derived from the *nodes* field)	0.0964

Except for *total_cpus*, which is numerical, all attributes are nominal (categorical). The *work_dir_parent* indicates which directory a job ran from. The *username* identifies the user who submitted a job. The *account* indicates which project a job is associated with. The *employed_simulator* gives the name and version of the reservoir simulator selected to run a job. The *total_cpus* reports the number of CPU cores requested to execute a job. Finally, the *nodes_prefix* points to the subset of compute nodes on which a job was run.

3.4. Job Duration Prediction Using a Decision Tree as a Classifier

Classification problems aim to predict a class within the limited existing possibilities. In the context of our work, we were able to forecast what the duration interval, in units of time, should be for a job to run on the cluster given a set of possible duration intervals.

We employed Weka's Decision Tree implementation, known as *J48*, to perform model training and validation. J48 is an implementation of the algorithm *C4.5* [Quinlan 1993], an extension of the algorithm ID3, which produces decision trees based on information theory. We utilized a decision tree model because it operates smoothly on large data sets and can use various feature subsets and decision rules at distinct classification stages. It is worth noting that the target attribute we selected for the job duration interval predictor was *job_duration_class*.

4. Experimental Results

We present the experimental evaluation of our reservoir simulator job runtime predictor strategy, proposed in Section 3. We investigated the potential benefits of the relevant feature extraction to improve predictor quality and compared our proposed model with the EASY++ scheduling algorithm as a runtime estimator. The input data used to run the machine learning models was for years 2021, 2022, and 2023, with the first containing records as of May fourth and the last containing records up to April fourth.

4.1. J48 Classifier Model

We considered only jobs submitted from a particular script call to run the J48 model as a Classifier, totaling 1,099,029 records (149,390 in 2021, 694,974 in 2022, and 254,665 in 2023). We formed six experimental scenarios. For each, 80% of the job logs were used for training and 20% for testing the trained model. Random stratified sampling was used for the splits. The J48 model took only 9.12 seconds to train and 1.23 seconds to evaluate the whole testing split in the last experimental scenario. It is worth mentioning that the model was also validated with 10-fold cross-validation, producing similar results. Thus, we present only the results obtained with the dataset split validation. Table 4 summarizes the scenarios and their respective amount of available data.

Table 4: Experimental scenarios used to run the J48 model.

Experimental Scenario	Observation Period	Training Dataset (80%)	Testing Dataset (20%)
ES_1	Year 2021*	119,512	29,878
ES_2	Year 2022	555,979	138,995
ES_3	Year 2023*	203,732	50,933
ES_4	Years 2021* and 2022	675,491	168,873
ES_5	Years 2022 and 2023*	759,711	189,928
ES_6	Years 2021*, 2022, and 2023*	879,223	219,806

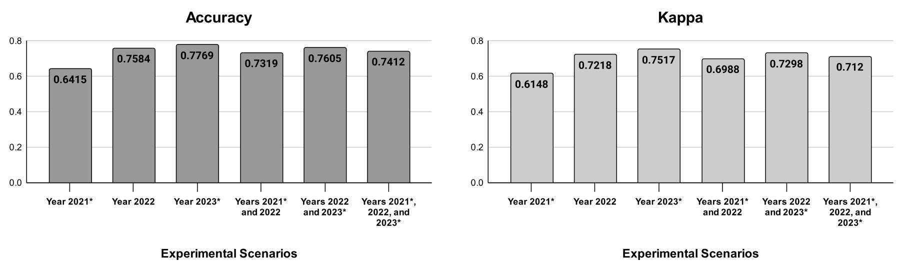
*This year's record does not include job data for all days of the year.

We used the following Classification numerical metrics for the model validation: *Accuracy*, *Cohen's Kappa Coefficient*, *True Positive Rate* (or *Recall*), *False Positive Rate*, *Precision*, *F-Measure* (or *F1 Score*), *Matthews Correlation Coefficient*, *Receiver Operating Characteristic Curve Area*, and *Precision-Recall Curve Area*. Table 5 summarizes their best possible values.

Table 5: Classification numerical metrics used to validate the trained J48 model.

Classification Numerical Metric	Best Value
Accuracy	1.0
Cohen's Kappa Coefficient (Kappa)	1.0
True Positive Rate (TP Rate)	1.0
False Positive Rate (FP Rate)	0
Precision	1.0
F-Measure	1.0
Matthews Correlation Coefficient (MCC)	1.0
Receiver Operating Characteristic Curve Area (ROC Area)	1.0
Precision-Recall Curve Area (PRC Area)	1.0

Figure 2 presents the values obtained for the Accuracy and Kappa metrics in the six experimental scenarios. Regarding the former metric, the lowest value was 0.6415, the highest was 0.7769, and the mean was 0.7351 (with a standard deviation of 4.8443). Regarding the latter metric, the lowest value was 0.6148, the highest was 0.7517, and the mean was 0.7048 (with a standard deviation of 0.0475).



(a) (b)

Figure 2: J48 model results in terms of the metrics (a) Accuracy and (b) Kappa.

The last experimental scenario (ES_6) is the most significant as it encompasses submitted jobs from three different years, showing the prediction capability of the trained model when more diversified data are available. Table 6 summarizes its training and testing samples distribution per job duration class obtained with Weka's percentage split function. Table 7 shows the values obtained for all classification numerical metrics for each job duration class and the weighted mean per metric.

Considering the last experimental scenario (ES_6), we built its *Confusion Matrix*, as illustrated in Figure 3. Classification models are often evaluated using a confusion matrix. For the multiclass classification problem, the matrix has the square form of $n \times n$, where n is the number of classes in the target attribute. In particular, $job\_duration\_class$ has 23 possibilities. For each, the matrix displays the number of *True Positives* (*i.e.*, the actual and predicted values are the same) colored gray on the main diagonal.

As shown in Figure 3, the main diagonal of the confusion matrix contains the most occurrences for each class. This result indicates that the jobs were correctly classified in general. When the classifier made an inaccurate prediction for a particular category, its suggested job duration class was usually right before or right after the actual one. Therefore, even when the classifier misestimated, its forecast for the job duration time interval was not so far from the actual duration interval.

Table 6: Training and testing samples distribution per $job\_duration\_class$ in ES_6.

$job\_duration\_class$	Training Samples		Testing Samples		$job\_duration\_class$	Training Samples		Testing Samples	
	Count	%	Count	%		Count	%	Count	%
0	192,519	78.90	51,491	21.10	12	17,749	83.66	3,467	16.34
1	96,892	79.65	24,750	20.35	13	15,012	75.86	4,778	24.14
2	58,978	81.62	13,284	18.38	14	12,578	83.24	2,533	16.76
3	51,907	81.06	12,125	18.94	15	10,805	85.47	1,837	14.53
4	77,768	76.86	23,412	23.14	16	9,874	86.18	1,583	13.82
5	38,810	84.08	7,348	15.92	17	9,496	90.60	985	9.40
6	24,335	83.97	4,644	16.03	18	53,157	77.21	15,690	22.79
7	61,718	77.70	17,713	22.30	19	9,149	82.83	1,897	17.17
8	48,906	80.08	12,166	19.92	20	2,094	82.60	441	17.40
9	45,458	79.87	11,457	20.13	21	273	83.23	55	16.77
10	25,075	83.25	5,045	16.75	22	305	85.67	51	14.33
11	16,365	84.27	3,054	15.73	–	–	–	–	–

Table 7: Metrics values per $job\_duration\_class$ in ES_6 using the J48 model.

$job\_duration\_class$	TP Rate	FP Rate	Precision	F-Measure	MCC	ROC Area	PRC Area
0	0.935	0.035	0.884	0.909	0.882	0.991	0.965
1	0.817	0.025	0.800	0.809	0.785	0.979	0.887
2	0.716	0.013	0.792	0.752	0.736	0.979	0.834
3	0.673	0.017	0.713	0.692	0.674	0.975	0.750
4	0.827	0.034	0.712	0.765	0.742	0.979	0.829
5	0.565	0.010	0.708	0.629	0.618	0.977	0.724
6	0.579	0.006	0.721	0.642	0.638	0.976	0.672
7	0.794	0.025	0.717	0.754	0.734	0.981	0.829
8	0.651	0.021	0.645	0.648	0.628	0.976	0.733
9	0.666	0.019	0.662	0.664	0.645	0.977	0.744
10	0.476	0.010	0.560	0.515	0.504	0.971	0.552
11	0.485	0.006	0.610	0.540	0.537	0.972	0.589
12	0.426	0.008	0.520	0.468	0.461	0.971	0.527
13	0.517	0.012	0.439	0.474	0.465	0.975	0.470
14	0.321	0.007	0.378	0.347	0.340	0.970	0.354
15	0.305	0.005	0.424	0.355	0.353	0.974	0.375
16	0.305	0.004	0.435	0.358	0.359	0.970	0.367
17	0.214	0.003	0.445	0.289	0.304	0.961	0.308
18	0.855	0.018	0.758	0.804	0.791	0.988	0.873
19	0.613	0.003	0.707	0.657	0.656	0.981	0.665
20	0.570	0.001	0.687	0.623	0.625	0.968	0.662
21	0.302	0	0.345	0.322	0.323	0.975	0.314
22	0.449	0	0.608	0.517	0.522	0.934	0.475
Weighted Mean	**0.741**	**0.022**	**0.734**	**0.735**	**0.717**	**0.981**	**0.798**

4.2. Comparison with the EASY++ Algorithm-based Estimator

Since *EASY++* continues to be a reference for scheduling algorithm, we used it as a predictor benchmark. Its implementation is simple: the algorithm considers the execution times of the last two jobs to dynamically adjust task prioritization and scheduling decisions, optimizing resource allocation and enhancing overall job performance. First, we evaluated EASY++ in its proposed form, which relies solely on user-submitted job information. To facilitate a more equitable comparison with J48, we extended this vanilla EASY++ functionality to include the attributes we selected whenever possible.

The predictor algorithm works as follows: the SLURM log is fully iterated concerning the actual job submission timestamps (*submit* field) to mimic the submission events. For each simulated job submission, the last two jobs that finished before the current job submission timestamp and have coincident parameters are selected and used to predict the execution time of the simulated job underway. Notice that the predictor hypothesizes that the execution behavior of the current job should be the same or very similar to its most recent related jobs.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Classified as 0	45533	1911	169	41	312	112	32	176	158	44	21	3	34	2	27	8	6	0	95	22	7	0	4
Classified as 1	3088	19805	425	105	155	102	89	244	41	33	19	11	9	6	14	6	2	4	53	15	1	0	4
Classified as 2	740	1224	10526	1420	293	50	40	261	14	21	9	10	21	22	0	2	0	0	40	4	11	1	2
Classified as 3	159	265	1469	8647	1873	144	51	185	7	4	2	2	0	3	0	0	0	0	27	5	6	0	1
Classified as 4	186	306	256	1450	16680	720	38	325	59	12	15	2	3	6	7	7	0	2	94	6	3	0	0
Classified as 5	127	119	59	286	2897	5205	265	153	16	14	8	4	5	5	4	0	3	2	39	1	1	0	1
Classified as 6	61	132	26	78	556	680	3348	797	23	15	5	0	3	14	6	1	3	0	24	3	5	0	0
Classified as 7	118	155	142	84	377	207	731	12702	1165	79	29	12	25	6	8	6	1	3	117	23	10	0	0
Classified as 8	231	145	33	7	77	55	10	2153	7853	1266	79	16	18	2	18	3	4	3	75	11	0	0	1
Classified as 9	275	68	46	3	33	21	4	379	1807	7579	923	40	47	12	20	5	8	4	105	4	3	0	0
Classified as 10	126	56	35	1	59	9	16	96	386	1681	2824	360	104	37	13	3	6	4	108	4	0	0	0
Classified as 11	65	71	17	0	14	1	5	47	141	230	694	1864	441	119	29	3	10	1	89	5	0	0	0
Classified as 12	117	50	11	0	2	2	5	42	182	208	238	457	1804	768	145	50	18	2	135	1	0	0	0
Classified as 13	84	67	5	0	2	2	2	15	54	68	76	131	501	2096	512	124	50	11	250	6	1	0	0
Classified as 14	59	39	2	0	5	7	0	8	51	19	27	68	233	889	958	290	53	15	251	8	0	0	0
Classified as 15	30	18	27	0	10	9	1	16	27	14	14	23	81	384	463	778	312	45	290	5	1	0	1
Classified as 16	48	16	0	0	4	12	0	12	58	26	24	11	51	175	154	332	688	197	446	3	0	0	0
Classified as 17	33	35	2	0	5	7	1	4	17	9	8	12	30	91	73	115	230	438	931	8	1	0	0
Classified as 18	267	226	17	0	49	3	4	56	101	134	30	23	43	134	76	95	180	249	11900	320	13	2	0
Classified as 19	89	36	9	3	5	0	1	28	5	1	0	1	8	4	5	6	9	5	584	1342	44	3	0
Classified as 20	43	2	7	0	2	0	1	14	1	0	0	3	5	2	1	0	0	0	32	92	303	19	5
Classified as 21	5	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	3	7	25	19	1
Classified as 22	7	4	0	0	2	0	0	0	0	0	0	0	1	1	0	2	0	0	2	2	6	11	31

Figure 3: Confusion matrix obtained in ES_6 using the J48 model.

We evaluated this algorithm as an *Estimator*, such that we wanted to predict the numerical value of the target attribute, particularly the job execution time, *i.e.*, the *elapsed* field of the SLURM log. Therefore, the following subset of attributes from Table 3 was used as coincident attributes of the two most recent related jobs: *work_dir_parent*, *username*, and *account*. Regarding the dataset, we considered the same used in ES_6 (Table 4) but without splitting it into training and testing parts. Furthermore, we use the following regression numerical metrics for the model validation: *Mean Average Error* (MAE), *Mean Average Percentage Error* (MAPE), and *Mean Squared Error* (MSE). It is worth noting that the best possible value for them all is zero. We obtained MAE $= 2.06$ hours, MAPE $= 67.84\%$, and MSE $= 14.75$ hours, meaning the estimator can grossly misestimate runtimes for shorter-duration jobs.

Next, we compared the estimator results with our proposed runtime predictor strategy, *i.e.*, the J48 classifier with relevant attribute selection. We mapped the estimated and actual execution times of the simulated jobs into one of the 23 possible categories for the *job_duration_class* attribute (Table 2). Moreover, we use the following classification numerical metrics for model validation: *Accuracy* and *Cohen's Kappa Coefficient*. Recall that the best possible value of both metrics is one. We obtained Accuracy $= 0.3041$ and Kappa $= 0.3692$, meaning that the adapted version of EASY++ is not an effective classifier in accurately determining the duration interval of an upcoming job.

Finally, Table 8 shows the comparison result for both classification strategies concerning the Accuracy and Kappa metrics. It can be seen that the proposed model exceeded the performance of the EASY++ implementation, with J48 achieving Accuracy $= 0.7412$ and Kappa $= 0.712$. Furthermore, the obtained results corroborate the observations in recent works on the challenge of using predictors to accurately forecast the duration of a job and indicate that our proposed classifier strategy has a reasonable predictive capability.

Table 8: Comparison result with J48 and EASY++ as classifiers.

Classification Metric	EASY++ as a Classifier	J48 as a Classifier
Accuracy	0.3041	0.7412
Kappa	0.3692	0.712

5. Concluding Remarks

Oil companies execute reservoir simulation jobs on high-performance computing clusters to discover petroleum field behavior, helping to minimize risks and optimize decision-making processes regarding extraction opportunities. Resource manager systems, such as SLURM, are typically employed to handle HPC clusters, which usually comprise thousands of nodes that run massive amounts of jobs daily. The extraction of relevant information from the workload logs allows one to understand the job submission profiles, which can be used by learning algorithms to increase the efficacy of the HPC cluster resources.

In this paper, we used a real-world SLURM log from Petrobras, a globally recognized Brazilian energy company, to build a reservoir simulation jobs runtime predictor incorporating strategically selected attributes that reasonably distinguish submitted jobs from each other. Furthermore, we implemented the *EASY++* scheduling algorithm for comparison purposes, since it is a reference algorithm for job scheduling. The results demonstrate that our classification model exceeded the performance of the EASY++ algorithm-based estimator on the Accuracy and Kappa metrics. Among other machine learning models, a regressor was also investigated to avoid the elapsed time discretization into classes, but further investigation is necessary to improve the predictions.

Our future work plans include exploring other learning models, expanding our runtime prediction strategy and validating it for the entire use case. Our goal is to provide real-time decision-making capability for reservoir simulation job execution to increase on-premises and cloud resources effectiveness. This system-level optimization will include resource usage rate improvement and job queue time reduction. Within this complete system we will be able to evaluate the benefits of our method over EASY++ and fine tune our strategy for the job duration initial guess. Potential prediction errors will be corrected through dynamic prediction. Nevertheless, we have a non-stationary problem, and proper strategies should be addressed to deal with workload changes over time. Besides, the minority classes' recall is lower than the mean, suggesting the need for suitable techniques to balance the dataset, such as oversampling and undersampling.

Acknowledgment

This work was funded by Petróleo Brasileiro S.A. (Petrobras), by Project Universal/CNPq n^o 404087/2021-3, and CNE/FAPERJ n^o E-26/201.012/2022(271103).

References

[Cheng et al. 2022] Cheng, F., Huang, Y., Tanpure, B., Sawalani, P., Cheng, L., and Liu, C. (2022). Cost-aware job scheduling for cloud instances using deep reinforcement learning. *Cluster Computing*, pages 1–13.

[Coats 1982] Coats, K. H. (1982). Reservoir Simulation: State of the Art. *Journal of Petroleum Technology*, 34(8):1633–1642.

[Gaussier et al. 2015] Gaussier, E., Glesser, D., Reis, V., and Trystram, D. (2015). Improving backfilling by using machine learning to predict running times. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–10.

[Gaussier et al. 2018] Gaussier, E., Lelong, J., Reis, V., and Trystram, D. (2018). Online Tuning of EASY-Backfilling using Queue Reordering Policies. *IEEE Transactions on Parallel and Distributed Systems*, 29(10):2304–2316.

[Gupta 2015] Gupta, Y. (2015). *Kibana Essentials*. Packt Publishing Ltd.

[Hall et al. 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

[Kuchnik et al. 2019] Kuchnik, M., Park, J. W., Cranor, C., Moore, E., DeBardeleben, N., and Amvrosiadis, G. (2019). This is why ML-driven cluster scheduling remains widely impractical. Technical report, Carnegie Mellon University.

[Kuhn and Johnson 2013] Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*, volume 26. Springer.

[Larose and Larose 2015] Larose, D. T. and Larose, C. D. (2015). *Data Mining and Predictive Analytics*. John Wiley & Sons.

[Lifka 1998] Lifka, D. A. (1998). *An extensible job scheduling system for massively parallel processor architectures*. Illinois Institute of Technology.

[Portella et al. 2022] Portella, F., Buchaca, D., Rodrigues, J. R., and Berral, J. L. (2022). TunaOil: A tuning algorithm strategy for reservoir simulation workloads. *Journal of Computational Science*, 63:101811.

[Quinlan 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

[Tanash et al. 2019] Tanash, M., Dunn, B., Andresen, D., Hsu, W., Yang, H., and Okanlawon, A. (2019). Improving HPC System Performance by Predicting Job Resources via Supervised Machine Learning. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*, pages 1–8. Association for Computing Machinery.

[Tsafrir et al. 2007] Tsafrir, D., Etsion, Y., and Feitelson, D. G. (2007). Backfilling Using System-Generated Predictions Rather than User Runtime Estimates. *IEEE Transactions on Parallel and Distributed Systems*, 18(6):789–803.

[Witt et al. 2019] Witt, C., Bux, M., Gusew, W., and Leser, U. (2019). Predictive performance modeling for distributed batch processing using black box monitoring and machine learning. *Information Systems*, 82:33–52.

[Yoo et al. 2003] Yoo, A. B., Jette, M. A., and Grondona, M. (2003). SLURM: Simple Linux Utility for Resource Management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer.