Prognosticating Latency in Directed Acyclic Task Graphs within Distributed Execution Frameworks

Yashpreet Malhotra

yashmalhotra9323@gmail.com

***Abstract*—This paper investigates the challenge of estimating completion duration for computational workflows represented as directed acyclic graphs (DAGs) within distributed processing systems. A compound methodology, integrating analytical formulations with data-driven models trained on sub-graphs, is proposed to address this problem. The approach leverages feature engineering to capture workflow attributes and employs machine learning techniques for predictive modeling. Empirical validation is conducted on complex, real-world applications, and comparative assessments of various predictive models are presented. The results demonstrate the efficacy of the hybrid strategy in approximating workflow execution time, even for graphs of substantial complexity.**

***Index Terms*—Workflow execution time estimation, Directed Acyclic Graphs (DAGs), Distributed systems, Predictive modeling, Machine learning, Feature engineering, Hybrid methodology, Computational workflows.**

I. INTRODUCTION

The past decade has seen rapid growth in big data applications across diverse domains including business analytics [1], healthcare [2], social media [3], and natural language processing [4]. These applications are both data- and compute-intensive, often relying on distributed systems and parallel processing for scalability and efficiency. Among available platforms, Apache Spark has become a popular choice due to its performance, fault tolerance, and ease of use [4].

With increasing adoption of cloud computing platforms—such as Amazon EC2, Google Cloud, and Microsoft Azure—organizations now deploy big data applications in highly configurable environments. Users can define cluster configurations by selecting the number of nodes and hardware specifications, which significantly impacts application performance and operational cost.

Accurate performance prediction in these environments is essential for optimizing resource usage and reducing unnecessary costs. Traditional approaches to performance estimation have included analytical modeling [5]–[7] and simulation [8]. Recently, machine learning (ML) has emerged as a viable alternative, offering data-driven methods to forecast application behavior [9]–[12]. These studies vary in feature selection strategies (black-box vs. gray-box) and ML model complexity.

However, most existing ML-based approaches focus on simple, monolithic applications and do not address the intricacies of modern big data workflows—typically represented as Directed Acyclic Graphs (DAGs)—which consist of multiple interdependent tasks executed using parallel frameworks [13]–[15].

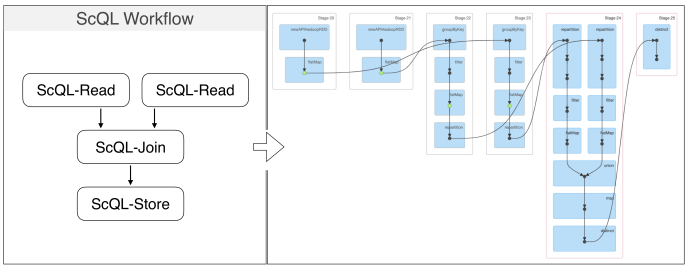

Fig. 1: A ScQL Workflow is mapped to Spark DAG Application.

.

In this work, we propose a modular performance prediction method tailored for complex Spark-based workflows. Rather than modeling the workflow as a single unit, we develop ML models for individual task types. Workflow execution time is then estimated by aggregating these per-task predictions. This method enables generalization to previously unseen workflows and supports flexible workflow composition. Since the characteristics of intermediate task outputs are unavailable beforehand, we integrate analytical modeling with ML to estimate their profiles.

Our hybrid approach addresses a significant gap in current research by focusing on task-level modeling and intermediate result estimation for Spark workflows. To the best of our knowledge, no prior work combines these strategies for performance prediction in complex data-driven workflows.

We evaluate our framework using a real-world system and compare multiple ML methods across different feature configurations. Further, we analyze model robustness under varying input sizes and computational resources.

II. DATA-DRIVEN WORKFLOW APPLICATIONS

Workflow applications can be modeled as Directed Acyclic Graphs (DAGs) $G = (V, E)$, where V denotes tasks and $E \subseteq V^2$ indicates data/control dependencies. Each task starts after all its parent tasks complete. We assume a single exit task, and each task produces one output usable by downstream tasks.

Each task is characterized by:

- **Output Profile**: Quantitative descriptors of its output.
- **Task Arguments**: Parameters (e.g., flags, options) affecting behavior.
- **Environment Parameters**: Runtime context (e.g., cores, memory).

- **Execution Time**: Actual runtime duration.

A. Target System: ScQL-to-Spark

We evaluate on a real cloud system converting ScQL (a SQL-like interval query language) into Spark workflows. A simplified query:

```
DS1 = READ() dataset_1;
DS2 = READ() dataset_2;
RES = JOIN(dist<100) DS1 DS2;
STORE RES;
```

produces a Spark DAG where each node corresponds to an operator. Key operators include:

- **Map**: Aggregates experiment intervals overlapping reference intervals.
- **Join**: Pairs overlapping intervals (possibly within a distance threshold).

Datasets are partitioned using *interval-binning*, with bin-size affecting load and replication.

III. Workflow Performance Prediction

We propose a three-phase method:

1) **Estimate intermediate profiles**: Predict output profiles for non-exit tasks, enabling input data availability (Section III-B).
2) **Predict task execution times**: Build ML models per task type using features (Section III-A).
3) **Aggregate to estimate total time**: Sum predicted times, assuming a single Spark context and full-core utilization.

A. Task Execution-Time Modeling

Models are trained per task type using features such as input profile, task arguments, and environment settings. Feature sets include:

- **Black-box**: Generic metrics (e.g., data size, cores).
- **Gray-box**: Includes domain-specific attributes.
- **Basic**: Raw measurements.
- **Composite**: Derived metrics (e.g., data/core ratio).

TABLE I: Feature set taxonomy

	Basic	Full
Black-box	Input, Params, Env	+ Derived features
Gray-box	Black-box + domain	+ Derived features

We compare Linear Regression, Decision Trees, and Random Forests. Training on small-scale runs often generalizes to larger scales in our experiments.

B. Estimating Intermediate Profiles

Intermediate profiles are predicted when actual data aren't available ahead of execution. Entry profiles are pre-computed. For others, we use:

- **Analytical models**: Exact based on known relations.
- **Heuristics**: Approximate estimations.
- **Machine Learning**: Data-driven predictions.

We favor analytical and heuristic methods due to their efficiency and accuracy, as demonstrated in Section IV.

TABLE II: Optimal Hyperparameters for Tree-based Models

Parameter	Decision Tree	Random Forest
Max Depth	5	8
Min Samples Split	4	2
Min Samples Leaf	1	2
Number of Estimators	N/A	300

TABLE III: Optimal Hyperparameters for Linear Regression

Parameter	Value
Penalty (α)	0.01
Fit Intercept	True

IV. Experimental Evaluation

We conducted comprehensive experiments to evaluate our performance prediction model on ScQL workflows, with particular focus on two complex operators. Our evaluation framework examined:

- Task-level execution time prediction accuracy
- Output profile estimation effectiveness
- End-to-end workflow performance prediction
- Model robustness through extrapolation analysis

A. Experimental Setup

All experiments were conducted on Amazon EMR clusters using `r5d.2xlarge` instances (8 vCPUs, 64GB RAM each). We evaluated cluster configurations ranging from 6 to 12 worker nodes running Spark 2.3.2 on Hadoop 2.8.5. The training dataset comprised over 4,500 execution profiles collected through our custom instrumentation framework.

B. Methodology

We employed *a-MLLibrary*, our extension of scikit-learn 0.19.1, for model training and hyperparameter optimization. Optimal parameters were determined through 5-fold cross-validation, minimizing mean absolute percentage error (MAPE).

C. Task Execution Prediction

Tables IV and V present the prediction accuracy for ScQL-Map and ScQL-Join operations across different modeling approaches.

Key observations:

- Random Forest consistently achieved the lowest prediction errors (9-17% MAPE)
- Incorporating gray-box features improved accuracy by 3-10 percentage points
- Linear models required engineered features to achieve competitive performance
- Join operations proved more challenging to predict than Map operations

D. Workflow Performance Prediction

We validated our approach on 168 production-scale workflows with complex DAG structures (averaging 23 jobs and 117 stages per workflow). Our gray-box approach achieved:

- Mean absolute prediction error: 17%

TABLE IV: ScQL-Map Prediction Accuracy (MAPE)

Model	Basic Features	Full Features
Random Forest	12%	**9%**
Decision Tree	16%	12%
Linear Regression	34%	14%

TABLE V: ScQL-Join Prediction Accuracy (MAPE)

Model	Basic Features	Full Features
Random Forest	17%	**13%**
Decision Tree	20%	16%
Linear Regression	32%	22%

- Maximum observed error: <50%
- Significant improvement over black-box methods (28% mean error, 120% max error)

E. Extrapolation Analysis

We evaluated model robustness through two scaling dimensions:

1) Data Scaling: Models trained on 20% of maximum dataset size maintained prediction accuracy when tested on:

- 2-4× larger datasets (error increase <5%)
- Full-scale production datasets

2) Resource Scaling: Models trained on small clusters (2-4 nodes) showed consistent accuracy when predicting for:

- Medium clusters (6-8 nodes)
- Large clusters (10-12 nodes)

This demonstrates our approach's practical viability for big data scenarios where full-scale training is infeasible.

V. Related Work

To the best of our knowledge, this is the first holistic study dedicated to predicting the performance of scientific workflows implemented using Apache Spark, leveraging both machine learning and analytical modeling techniques. Unlike prior research that has either concentrated on forecasting the execution time of individual cloud-based Spark applications or estimating the total makespan of workflows, our work introduces a unified and modular approach that also includes intermediate data profile estimation—a critical aspect that is typically unknown prior to execution.

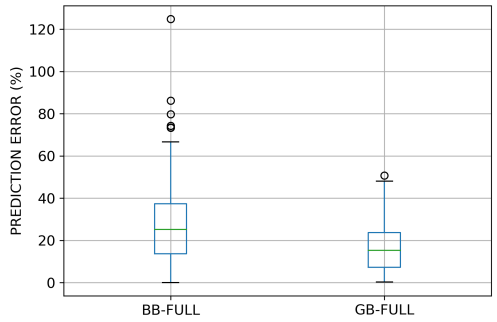

Fig. 2: Workflow execution time prediction error distribution for BB-FULL and GB-FULL

Performance Prediction for Cloud-based Applications

The problem of forecasting the execution time of cloud-native applications, particularly those relying on big data frameworks such as Spark, has been explored through various methodologies. Classical methods include the use of analytical modeling [6], [16]–[18] and simulation-based techniques [19]. While these approaches provide some level of interpretability, they often rely on restrictive assumptions and demand granular runtime information, which limits their practical applicability and accuracy in the face of complex and dynamic cloud environments.

Some architectural interventions have also been proposed to simplify performance estimation. For instance, [20] introduces a modified Spark architecture that decouples computation into monotasks to better predict job completion time. However, such interventions are not easily generalizable to existing production systems.

In more recent years, machine learning (ML) has emerged as a popular alternative for performance prediction in cloud systems [9]–[12], [21]. ML-based approaches are typically categorized as either *black-box* or *gray-box*. Black-box models rely purely on historical input-output data to predict future performance, with no knowledge of the underlying system architecture. Examples include *Ernest* [9], which employs basic input size and hardware configuration features in a non-negative least squares (NNLS) regression framework. Despite its simplicity, *Ernest* remains a widely referenced benchmark.

Gray-box models, on the other hand, incorporate partial system knowledge to enhance prediction accuracy. Studies such as [12], [22] demonstrate that gray-box models can outperform black-box models under certain conditions, particularly when the execution environment exhibits non-trivial performance behaviors. [12] conducts a comparative study on different ML models and modeling approaches, emphasizing the advantages of incorporating system-level features.

Workflow Performance Prediction

Workflow performance modeling has traditionally focused on predicting the execution time of individual tasks within a structured workflow. Such efforts aim to enable better resource allocation and workflow scheduling [23]. Many of these works use machine learning algorithms to model execution time based on a combination of task characteristics and environmental parameters [24]–[26]. However, a major limitation of these approaches is that they often treat the workflow as a static structure and use coarse-grained input descriptors—typically the overall input size—while ignoring the dynamic and interdependent nature of task-level input and output data transformations.

These limitations reduce the efficacy of such models when applied to modular and composable data workflows, especially those involving complex transformations, branching, or iterative patterns as observed in Spark pipelines. Furthermore, existing literature rarely addresses the estimation of intermediate data profiles, which are essential for accurate downstream task modeling.

Distinctiveness of Our Approach

Our work differs from previous studies in several key aspects. First, we target data-intensive workflows built on Apache Spark, which involve intricate task interdependencies and runtime dynamics. Second, we move beyond traditional black-box feature sets by incorporating intermediate data estimation strategies that enable more accurate task-level performance modeling. This makes our method applicable to a broader range of scientific and industrial workflows, where modularity, reusability, and data-dependency are crucial. Finally, by combining analytical reasoning with data-driven learning, we provide a flexible and scalable framework that adapts to diverse workflow structures and execution contexts.

VI. Conclusions

This work introduces a novel hybrid methodology for the performance prediction of complex, data-driven workflows that can be executed within Apache Spark environments. Our approach is structured in three distinct but interlinked phases and is designed to be modular, enabling scalable and reusable modeling of a wide variety of workflow configurations. A key feature of our methodology is the decomposition of the overall workflow performance into the execution times of individual constituent tasks, which are then aggregated to produce an estimate of the entire workflow's runtime. This decomposition facilitates a more granular understanding of performance bottlenecks and allows for targeted optimization strategies.

A significant advancement presented in this work is the ability to handle dynamic workflows composed of arbitrary combinations of task types. Unlike many prior studies that assume fixed workflow structures and known data characteristics, our approach explicitly addresses the challenge of estimating the input profiles of intermediate tasks—information that is typically not available in advance. This capability is critical in real-world scenarios, where workflows often evolve at runtime and exhibit non-trivial dependencies between tasks.

To validate the proposed methodology, we conducted a comprehensive experimental evaluation using a realistic and complex system as a benchmark. The experiments assessed the effectiveness of various machine learning models under different levels of application-specific knowledge. Specifically, we analyzed the impact of using black-box features (which rely solely on observable input-output behavior) versus gray-box features (which incorporate limited internal knowledge such as task semantics and data transformation logic). The results consistently demonstrate that incorporating even partial information about the internal workings of the system significantly improves prediction accuracy.

Moreover, our predictive models exhibit robust generalization capabilities. They maintain low prediction errors not only for configurations encountered during training but also when evaluated on previously unseen conditions, including new input data sizes, varying levels of cluster resources, and different workflow scales. This highlights the adaptability and practical applicability of our approach in diverse deployment contexts, such as cloud platforms and scientific computing environments.

In summary, this work contributes a flexible, extensible, and accurate framework for workflow performance prediction. By addressing key challenges such as intermediate data estimation and task heterogeneity, and by rigorously validating our models across multiple scenarios, we lay a solid foundation for further research in predictive workflow management, intelligent scheduling, and adaptive resource provisioning in big data systems. Future work will explore integration with real-time monitoring systems and dynamic feedback loops to enable live performance tuning during workflow execution.

References

[1] Z. Sun, L. Sun, and K. Strang, "Big data analytics services for enhancing business intelligence," *Journal of Computer Information Systems*, vol. 58, no. 2, pp. 162–169.

[2] A. J. Kulkarni, P. Siarry, P. K. Singh, A. Abraham, M. Zhang, A. Zomaya, and F. Baki, *Big Data Analytics in Healthcare*.

[3] N. A. Ghani, S. Hamid, I. A. T. Hashem, and E. Ahmed, "Social media big data analytics: A survey," *Computers in Human Behavior*, vol. 101, pp. 417–428.

[4] J. Hirschberg and C. D. Manning, "Advances in natural language processing," vol. 349, no. 6245, pp. 261–266.

[5] V. Mak and S. Lundstrom, "Predicting performance of parallel computations," *IEEE Trans. on Parallel & Distributed Systems*, vol. 1, no. undefined, pp. 257–270, 1990.

[6] D. Ardagna, E. Barbierato, A. Evangelinou, E. Gianniti, M. Gribaudo, T. B. Pinto, A. Guimarães, A. P. Couto da Silva, and J. M. Almeida, "Performance prediction of cloud-based big data applications," in *ICPE 2018*.

[7] E. Vianna, G. Comarela, T. Pontes, J. Almeida, V. Almeida, K. Wilkinson, H. Kuno, and U. Dayal, "Analytical performance models for mapreduce workloads," *International Journal of Parallel Programming*, vol. 41, no. 4, pp. 495–525, 2013.

[8] A. Verma, L. Cherkasova, and R. H. Campbell, "ARIA: Automatic Resource Inference and Allocation for MapReduce Environments," in *ICAC 2011*.

[9] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient performance prediction for large-scale advanced analytics," in *13th NSDI*, 2016.

[10] X. Pan, S. Venkataraman, Z. Tai, and J. Gonzalez, "Hemingway: modeling distributed optimization algorithms," *arXiv preprint arXiv:1702.05865*.

[11] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, "Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics," in *14th NSDI*, 2017.

[12] A. Maros, F. Murai, A. P. Couto da Silva, J. M. Almeida, M. Lattuada, E. Gianniti, M. Hosseini, and D. Ardagna, "Machine learning for performance prediction of spark cloud applications," in *2019 IEEE CLOUD*.

[13] M. Atkinson, S. Gesing, J. Montagnat, and I. Taylor, "Scientific workflows: Past, present and future," 2017.

[14] W. A. Warr, "Scientific workflow systems: Pipeline pilot and knime," *Journal of computer-aided molecular design*, vol. 26, no. 7, pp. 801–804.

[15] F. Lordan, E. Tejedor, J. Ejarque, R. Rafanell, J. Álvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, and R. M. Badia, "Servicess: An interoperable programming framework for the cloud," *Journal of grid computing*, vol. 12, no. 1, pp. 67–91, 2014.

[16] R. Nelson and A. N. Tantawi, "Approximate analysis of fork/join synchronization in parallel queues," *IEEE transactions on computers*, vol. 37, no. 6, pp. 739–743.

[17] V. W. Mak and S. F. Lundstrom, "Predicting performance of parallel computations," *IEEE Transactions on Parallel and Distributed Systems*.

[18] D.-R. Liang and S. K. Tripathi, "On performance prediction of parallel computations with precedent constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 5, pp. 491–508.

[19] M. Bertoli, G. Casale, and G. Serazzi, "Jmt: performance engineering tools for system modeling," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 4, pp. 10–15.
[20] K. Ousterhout, C. Canel, S. Ratnasamy, and S. Shenker, "Monotasks: Architecting for performance clarity in data analytics frameworks," in *SOSP 2017*.
[21] "A machine learning approach for predicting execution time of spark jobs," *Alexandria Engineering Journal*, vol. 57, no. 4, pp. 3767 – 3778.
[22] J. Shon, H. Ohkawa, and J. Hammer, "Scientific workflows as productivity tools for drug discovery." *Current opinion in drug discovery & development*, vol. 11, no. 3, pp. 381–388.
[23] G. Kousalya, P. Balakrishnan, and C. P. Raj, "Workflow scheduling algorithms and approaches," in *Automated Workflow Scheduling in Self-Adaptive Clouds*, 2017, pp. 65–83.
[24] T. P. Pham, J. J. Durillo, and T. Fahringer, "Predicting workflow task execution time in the cloud using a two-stage machine learning approach," *IEEE Transactions on Cloud Computing*, 2017.
[25] R. F. Da Silva, G. Juve, M. Rynge, E. Deelman, and M. Livny, "Online task resource consumption prediction for scientific workflows," *Parallel Processing Letters*, vol. 25, no. 03, p. 1541003, 2015.
[26] M. H. Hilman, M. A. Rodriguez, and R. Buyya, "Task runtime prediction in scientific workflows using an online incremental learning approach," in *IEEE/ACM UCC*, 2018.