Workload Characterization for Resource Optimization of Big Data Analytics: Best Practices, Trends, and Opportunities

Dominik Scheinert and Alexander Guttenberger

Technische Universität Berlin, Berlin, Germany

Abstract. As distributed processing environments grow in complexity, accurate performance prediction models are essential to optimize system efficiency and resource allocation. However, modern computing workloads typically exhibit a wide variety of characteristics, which hinders optimized resource configurations. Diverse approaches have been suggested to tackle the challenge of workload characterization, employing various parameters for performance modeling in the process. To expand on this objective, this paper extensively surveys existing performance modeling methodologies and introduces a 5+1 layer classification model designed to enhance the accuracy of predictive models by classifying and reflecting on relevant modeling parameters. We conducted a systematic literature review to identify and analyze the role of six key layers: Big Data Framework, Performance, Hardware, Data, User Application, and Virtualization.
Our findings reveal that while the Big Data Framework and Performance Layers are foundational, predictive accuracy improves when combined with complementary layers, especially the Data Layer, which highlights the impact of data characteristics such as size and distribution. The Hardware Layer provides critical insights into system limitations, while the emerging Virtualization Layer reflects the increasing importance of virtualized, potentially cloud-based environments. The proposed 5+1 layer classification model offers a structured approach to capture and explain the complexity of distributed analytical workflows, providing a nuanced framework for performance modeling. This layered classification model aims to support the development of more robust, adaptable, and generalizable prediction models for use in cloud-based systems.

Keywords: Big Data Analytics, Performance Modeling, Resource Management, Cloud Computing

1 Introduction

In recent years, the volume of data stored and processed in data centers has grown rapidly, creating unprecedented challenges in data management. The increase is not limited solely to size, but encompasses a spectrum of characteristics that introduce obstacles to efficient data handling [65]. To keep up with this development, researchers are increasingly relying on sophisticated frameworks and specialized algorithms tailored to large-scale data processing. Through continuous innovation, a variety of tools and methods have emerged, providing valuable solutions to these challenges.

Among these advancements, distributed computing frameworks such as MapReduce [24], Spark [99], and Flink [18] have become central in managing large amounts of data. These frameworks leverage clusters of computing nodes to facilitate parallel and distributed data processing, enabling scalability and meeting the complex demands of contemporary data analytics. Data processing in these systems generally follows two main paradigms: streaming and batch processing [72]. Streaming processing allows for real-time data analysis, which is crucial for applications where immediacy is paramount. For example, Uber uses Flink to process extensive real-time data streams generated by its applications, enabling services such as surge pricing, fraud detection, and live ride tracking [27]. In contrast, batch processing deals with large, finite datasets, processing them in bulk for analysis, which is often more efficient for historical data analysis or intricate analytical tasks [85].

Batch processing, in particular, plays a key role in various domains, from healthcare to e-Commerce [76,89]. However, predicting the performance of analytical batch jobs remains

DOI:10.5121/ijci.2025.140204

a significant challenge, as various characteristics of the workload impact performance, complicating the identification of the most relevant factors [94], especially for completely new workloads that have not yet been executed and observed. Many existing performance models presuppose that tasks exhibit recurrence [83,81,6,69], an assumption that is often backed by cluster trace analysis [56] of major infrastructure providers, but raises the question of the exactness of recurrence. The conventional perspective on recurring jobs assumes consistency across framework configurations, algorithm parameters, and data characteristics in repeated executions, or at least similarity between datasets [15]. However, this assumption does not always hold. Even for jobs scheduled at regular intervals, datasets can fluctuate significantly in size and content. For example, a predictive algorithm may handle data from varying days or times, each with its own granularities and peculiarities [83]. This inconsistency between executions suggests that "recurrence" in data processing is more nuanced than commonly assumed, highlighting the need for more sophisticated methods to manage and predict performance in dynamic batch processing environments.

To advance research on developing robust performance models that can utilize data from varied or similar workloads, we first sought to identify the key workload characteristics commonly used in performance modeling for distributed analytic computing. For this, we conducted a comprehensive systematic review of the existing literature, using the Web of Science database to identify essential workload characteristics in this field. The review resulted in the creation of the 5+1 layer classification model, a unified framework that synthesizes and represents all relevant workload characteristics. Furthermore, we examined the use of similarity metrics in these models to improve performance prediction but found no instances where such metrics were applied. Our research results aim to offer researchers and practitioners a comprehensive overview of best practices, observed trends, and further optimization opportunities by consolidating and refining existing literature in this domain.

Outline. The paper is structured as follows. Section 2 discusses the relevant concepts and related work. Section 3 describes the methodology adopted for this research and outlines the systematic approach taken. Section 4 presents the strategy and results of the exploratory systematic literature review, as well as derived future research directions. Section 5 briefly discusses the threats to validity, whereas Section 6 concludes the paper.

2 Background and Related Work

This section presents the relevant concepts, ranging from big data analytics to performance modeling, as well as the related work.

2.1 Big Data Analytics and Frameworks

In distributed computing, a batch processing job refers to large-scale data processing tasks scheduled to run at specific times or intervals, processing historical data rather than real-time inputs [48]. Such jobs typically handle updated datasets repeatedly [77] and are automatically scheduled to fit organizational needs, such as daily or monthly runs [98]. Batch processing suits tasks that can tolerate delays and have fixed start and end times, allowing them to run during off-peak hours to optimize resource use and reduce costs or environmental impact [14,90]. However, their scheduling flexibility is influenced by constraints such as execution time and interruptibility [90]. Examples of batch processing applications include the generation of user-specific recommendations, with companies such as eBay employing this approach [89]. Research on Alibaba clusters shows that over 80.3% of batch applications are recurring, often involving multiple tasks completed within hours, highlighting batch processing's importance in managing periodic data-intensive

jobs [56]. Streaming processing, on the contrary, involves real-time data input, processing, and output, suitable for applications that require immediate insight, such as fraud detection and network monitoring [47,26,27,75]. It is widely applied in environments that continuously generate high-speed data, such as Internet of Things (IoT) devices and social media feeds [14,30,3,17].

Scalability in modern data analytics is often achieved using dedicated *distributed dataflow frameworks* like Apache Spark [99], Hadoop [24], and Flink [18], which enable parallel data processing across multiple nodes. Such frameworks handle complexities such as data distribution, scheduling, and fault tolerance, making them popular for various tasks. Within such a framework, several components are vital for scalable data processing:

- The worker nodes execute tasks and store intermediate results, working in parallel to process large datasets, while the master nodes manage the distribution of jobs and monitor the progress of tasks to ensure accurate execution and elemental fault tolerance [24,64,68,99].
- Tasks are the smallest units of work, processing parts of data in parallel to achieve efficient computation [64,68,99].
- The execution slots on each worker node, defined by available CPU and memory, dictate the number of concurrent tasks a node can handle [64,68,77].

Hadoop typically operates with two stages — Map and Reduce — whereas Spark allows multiple stages linked in a Directed Acyclic Graph (DAG) for complex operations [64,9]. The DAG structure in Spark helps define the task execution sequence, optimizing parallelism and resource allocation to improve performance [9,68].

2.2 Challenges of Resource Management

Resource management systems are critical to distributed big data analytics frameworks, managing the efficient allocation of computational resources through tasks such as scheduling, monitoring, logging, user authentication, and job status communication [33]. Systems like YARN separate resource management from the programming model and delegate scheduling duties to application-specific components [33,82].

A core aspect of resource management systems is accurately predicting resource demand to balance performance and cost, which is particularly relevant for big data analytics workloads. Both overprovisioning and underprovisioning resources can have adverse effects, leading to unnecessary costs or performance drops, respectively [79,91]. However, accurate prediction is complex due to the dynamic nature of workloads, which vary in resource needs based on factors like job size, complexity, and data type [74,39].

Big data analytics frameworks, such as Apache Spark [99], offer extensive tuning options — over 150 configurable parameters including memory, CPU, and node count — which makes optimizing performance challenging for users and administrators alike [21,1]. In addition, resource management systems often operate in heterogeneous environments, where nodes differ in CPU, memory, and storage capabilities, further complicating resource allocation and prediction [4,28]. Concurrent execution of jobs within the same cluster can lead to resource contention, where jobs compete for shared resources, affecting individual job performance and complicating the accurate prediction of resource needs [44,74]. Resource management systems must account for these interactions to minimize bottlenecks and ensure optimal resource distribution.

2.3 Performance Modeling Methodologies

In performance modeling for distributed big data analytics, approaches are typically categorized into simulations, analytical models, and machine learning.

Simulation-Based Prediction Models Simulation-based models are a common approach for predicting performance, using computational algorithms to emulate system behavior [86,49,7,52]. These models aim to closely replicate the execution environment, allowing researchers to estimate metrics such as execution time [7,52], resource usage, and bottlenecks [92]. However, developing detailed simulators can be resource-intensive and time-consuming [44]. Often, simulation-based models use profile runs, which gather performance metrics by running representative workloads on the actual system. These metrics include I/O overhead [86,92], CPU usage [49,52], and execution time, which help calibrate the simulation model for accuracy. For instance, [86] utilizes profile runs with partial input data to gather execution traces, using mathematical formulas to predict performance at each execution stage. Although profile runs are valuable, they do have limitations; they may not capture all aspects of system behavior under varied conditions, which can affect simulation accuracy. Additionally, profile runs are time-intensive and can require substantial computational resources.

Analytical Prediction Models Analytical models use system knowledge to create mathematical frameworks linking adjustable parameters to performance outcomes, often requiring minimal or no training [25]. They incorporate specific workload traits, such as workflow structures or configurations, into performance models. For example, [33] uses a precedence graph and a queuing network model to capture task dependencies and synchronization constraints based on workload structure and configuration. Similarly, [31] applies Fluid Petri Nets to simulate Spark application behavior, integrating Spark's configuration and data flow into a comprehensive performance model. However, these models face the issue of "state space explosion", where increasing variable complexity challenges the analysis. In Spark, multiple concurrent jobs, varying task types, and user priorities managed by YARN create numerous states that a single model struggles with [44,74].

Machine Learning Prediction Models Machine learning models use statistical methods to create predictions from historical data, with accuracy largely depending on data representativeness [25]. In distributed systems, these models are classified as white-box or black-box, based on the use of workload characteristics [69]. White-box models incorporate specific internal behaviors of the system, requiring access to source code, documentation, and a deep understanding of the system components [42]. They offer high precision due to their detailed insight into application and system operations, but are complex, time consuming to build, and challenging to adapt to different systems [69]. Like analytical models, white-box models require constant updating to reflect system changes, limiting their flexibility. Conversely, black-box models approach the system as a black-box, using observed input-output data to build predictions without needing internal system knowledge. These models are simpler to develop, rely on empirical data from job executions, and are more generalizable across different frameworks with minimal adjustment [42]. Yet, the accuracy of such black-box models is highly dependent on data quality, which is why insufficient or poor data can yield less reliable predictions [69].

3 Research Methodology

The systematic review of the literature is the initial component of this research, aiming to analyze the current state of workload characterization in distributed systems. We follow a structured approach, ensuring both reproducibility and reliability. For our systematic review of the literature, this involves clearly defining and documenting the methods used to discover, select, analyze, and synthesize primary sources [29], which is why we take inspiration from the framework developed in [16]. The systematic study selection was carried out in the first quarter of 2024.

3.1 Definition of the Review Scope

As conveyed in the discussion of the related work, a plethora of workload characterization techniques used in performance prediction models for distributed analytical workflows exist. Our goal is to provide a structured overview of workload characterization in this field, which then enables a more systematic approach when designing novel performance models. This work aims to contribute to both academic research and practical applications.

3.2 Conceptualization of the Topic

We began by carefully crafting a keyword strategy, building it concept by concept, to guide our systematic literature review. Initially, we identified the core concepts central to our study, namely "performance prediction" and "distributed analytic job", and expanded them into a wider set of terms and synonyms. This expansion enabled us to capture various facets of each concept. We then combined these terms using logical operators to link related concepts and ensure comprehensive coverage. This approach was iterative; we continually refined our search terms in response to new findings. We applied this evolving strategy across multiple academic databases to ensure a broad and thorough exploration of the existing literature. The selected keywords are:

> ("dynamic adjustments" OR "dynamic scaling" OR "prediction model*" OR "performance model*" OR "estimating runtime" OR "prediction execution time") AND ("distributed dataflow" OR "distributed analytics" OR "data analytics frameworks" OR "scalable data analytics" OR "scalable data processing" OR "hadoop" OR "spark" OR "flink")

3.3 The Literature Search

To identify relevant literature, we employed a five-part search strategy: (1) identification; (2) rough screening and evaluation of eligibility; (3) filtering for new ventures; (4) forward and backward search; (5) inclusion. Our primary source for academic papers was the "All Databases"[1] collection on Web of Science. Web of Science is a respected database within the academic community, known for its advanced search capabilities, comprehensive indexing, and transparent search algorithm. This platform provides a vast collection of peer-reviewed journals, conference proceedings, and scholarly articles across diverse disciplines. Our choice of Web of Science over Google Scholar was informed by factors highlighted in the paper "Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources" [36]. While Google Scholar is known for its extensive database, it often

[1] `https://www.webofscience.com/wos/alldb/basic-search`, Accessed: October 2024

returns a large number of irrelevant results, making systematic searching both difficult and time-consuming. Additionally, Google Scholar lacks consistent reproducibility of search results, an issue that cannot simply be attributed to database expansion. In contrast, Web of Science ensures that identical searches yield consistent results over time.

We then carefully screened the titles and abstracts of each article to determine eligibility, ensuring they met the following criteria:

1. **Publication in Peer-Reviewed Journals:** Only articles from peer-reviewed journals and conferences were included to maintain academic credibility and research quality.
2. **Language:** All papers were required to be in English to ensure full comprehension and accurate analysis.
3. **Research Methodology:** We prioritized studies using empirical research methods to ensure concrete data and measurable results that can inform our investigation.
4. **Open Source Platforms:** We focused on studies involving open-source distributed analytics platforms (e.g., Hadoop, Spark) to reflect industry trends and practices.
5. **Non-Outdated Technology:** Papers needed to use Hadoop 2 or newer, as Hadoop 1 is unsupported by most cloud providers, ensuring relevance for modern computing.

Papers that passed this stage were then subjected to a more detailed review. This involved an in-depth assessment of each study's full text, rigorously evaluating its relevance and suitability for inclusion in our systematic review. Following the eligibility screening process, we conducted a comprehensive forward and backward search with the 10 most-cited papers. For this, we used an online tool[2] designed to visualize the connections between academic papers. This platform enabled us to thoroughly explore the citation network associated with our selected studies. The forward and backward searches examined both the references within the selected papers (backward search) and the papers that cited our selected studies (forward search). This approach aimed to capture a wider range of related research. By examining both citing and cited papers, we ensured we did not overlook significant studies that may have indirect relevance to our primary selections, thereby enhancing the depth and scope of our systematic review.

3.4 Literature Analysis and Synthesis

Determining the similarity of distributed workloads presents a significant challenge, requiring an in-depth investigation of the various levels and parameters influencing these workloads. In Section 4, we present the results of our exploratory analysis, where we extracted all workload-related input parameters of performance models from the systematically selected literature. We also provide a synthesis and overview of all identified workload characteristics and propose a 5+1 layer classification model of workload characteristics to enhance dataset quality and optimize performance prediction.

4 Analysis and Findings

In the following, we will analyze and synthesize all used performance prediction model input variables related to workloads into a single synthesis. Finally, we will also present other findings resulting from our research.

[2] `https://www.connectedpapers.com`, Accessed: October 2024

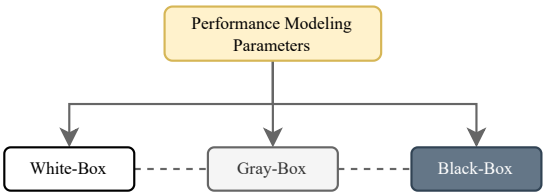

Fig. 1. Classification of parameters.

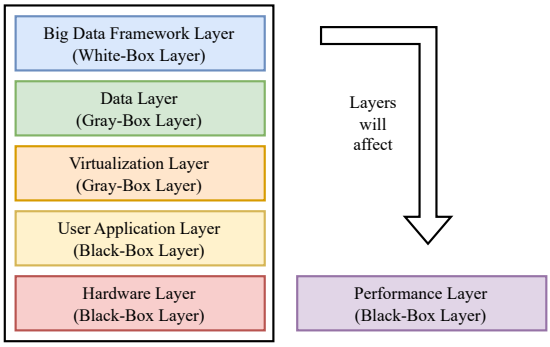

Fig. 2. 5+1 layer classification model.

4.1 5+1 Layer Classification Model

The 5+1 layer classification model (cf. Figure 2) encompasses six categories of workload characteristics used in performance modeling for distributed processing jobs. These categories include characteristics related to cluster hardware, input data, the big data framework, the submitted user application, and performance metrics. As illustrated in Figure 1, we classify input parameters as either *white-box* or *black-box* based on whether they can be observed without accessing the big data framework and application code. Some parameters may even be classified as *gray-box*, as they may or may not be framework-specific depending on the model. Consequently, layers composed solely of black-box properties are designated as black-box layers, those with only white-box properties are termed white-box layers, and layers containing both types or gray-box parameters are called gray-box layers. Figure 3 illustrates the six layers and their overall frequency in the researched articles.

Big Data Framework Layer The layer most commonly used is the Big Data Framework Layer, which appears in 45 of the 62 performance models (cf. Figure 3). This layer includes all characteristics (cf. Table 1) of the workload that reflect the internal behavior of the processing platform. Since these characteristics are platform-specific, they are classified as white-box parameters, making this a white-box layer. Modeling approaches that use this layer predict performance by incorporating system behavior, often providing deeper insights into the internal execution of a program and its resulting performance. This approach bypasses the need for high-quality training datasets and avoids the significant training overhead involved in collecting data for model construction [88,93]. However, because all parameters are framework-specific, the insights derived from this layer are challenging to apply across different frameworks. The frequent use of the Big Data Framework Layer suggests an over-representation of framework-specific performance prediction approaches in current research methodologies.

The most commonly used workload characteristic within this layer was the framework configuration, featured in 17 performance models. Choosing the appropriate framework configuration is challenging due to the vast number of parameters available. For example, the Hadoop framework includes up to 190 configuration parameters, and overall performance is highly sensitive to these settings [13]. Finding the optimal configuration for Hadoop is application-specific, meaning that applying a default or universally optimized configuration to various applications results in suboptimal performance. Manually tuning these parameters without extensive knowledge of the Hadoop system and the specific application is tedious and time-consuming, potentially leading to significant performance

Table 1. Big Data Framework Parameters.

Count	Parameter	Category	References
17	Framework Parameter Configuration	white-box	[66,23,74,22,98,33,97,20,54,34,41,57,59] [40,38,63,2]
12	Workflow Structure	white-box	[9,11,4,31,78,48,8,35,92,67,38]
9	Number of Tasks	white-box	[66,31,53,71,59,19,93,10]
5	Number of Slots	white-box	[9,31,50,73,63]
5	Running Tasks	white-box	[5,8,11,55]
4	Completed Tasks	white-box	[8,11,5]
4	Number of Job Stages	white-box	[19,9,33,93]
3	Available Resources	white-box	[80,78,5,2]
3	Number of Partitions	white-box	[73,7]
3	CPU Cores per Slot	white-box	[73,7]
2	Data Chunks Size	white-box	[66,48]
1	Parallelism Factor	white-box	[92]
1	Job Queue	white-box	[87]
1	Task Scheduler Delay	white-box	[11]

degradation [13]. Additionally, selecting a particular subset of configurations can introduce selection bias, which may result in less accurate performance predictions.

The second most commonly used characteristic, applied in 11 performance prediction models, was information about the workflow structure. Traditional performance modeling systems for big data analytics have primarily focused on MapReduce-like frameworks [33,50,87,46]. MapReduce operates through three distinct stages: map, shuffle, and reduce. During the map stage, input data is processed into intermediate key-value pairs, with each map task independently handling a subset of data, enabling high parallelization. The shuffle stage then reorganizes these intermediate data pairs, sorting and grouping values by key. In the reduce stage, the grouped data is processed to produce the final output by aggregating intermediate values. This model simplifies large-scale data processing on distributed systems, abstracting complexities in parallel and distributed computing, which enables efficient execution of extensive data operations [24]. However, the MapReduce paradigm only includes the map and reduce phases. Some researchers have leveraged Spark's Directed Acyclic Graph (DAG) to incorporate workflow structure into performance models. Spark provides numerous data manipulation operators, such as groupByKey and join, which utilize the DAG to define dependencies. Spark divides jobs into multiple stages,

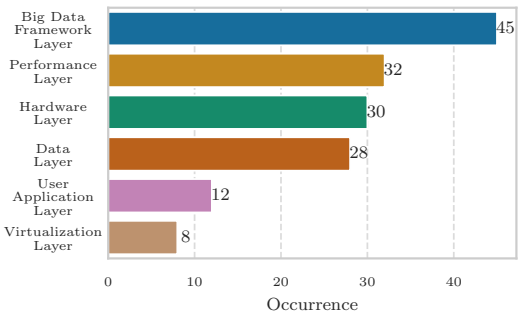

Fig. 3. Frequency analysis of parameter layers.

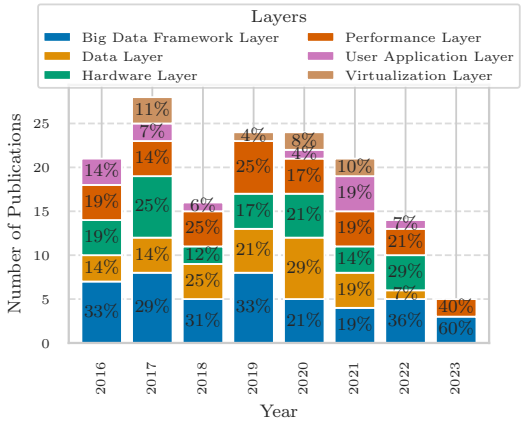

Fig. 4. Evolution and distribution of layer usage.

Table 2. Performance Parameters.

Count	Parameter	Category	References
21	(Task) (Average or Total) Execution Time (for each Stage)	gray-box	[9,11,7,74,84,97,34,92,67,59,52,60,10,55] [71,80,28,51,33,53,2]
6	(Task) CPU Utilization	gray-box	[49,78,96,70,74,100]
3	Task Memory Requirement	white-box	[86,87,1]
3	(Task) Startup and Cleanup Time	gray-box	[51,86,9]
3	(Task) I/O Write Cost	gray-box	[86,49,71]
3	(Task) I/O Read Cost	gray-box	[86,71]
2	Data Processing Ratio (Size Read Data / Size Write Data)	white-box	[92,59]
2	Processing Latency	white-box	[33,100]
2	Task Success Rate	white-box	[53,59]
1	Data Processing Rate (Size / Time)	white-box	[92]
1	Network Usage	black-box	[100]
1	Historical Number of Runs	black-box	[80]
1	Network Latency	black-box	[49]
1	Data Localization Ratio (locally stored data)	white-box	[92]
1	Number of Bytes Transferred During Shuffles	white-box	[67]

each scheduled in a distributed execution environment [101]. While each stage has distinct resource requirements, these are not represented by the DAG [101], and the multitude of stages introduces complexity to performance modeling [32].

Performance Layer The Performance Layer, the second most commonly used layer, includes historical or real-time hardware usage data for distributed workflows. Most parameters in this layer, as listed in Table 2, can be accessed without direct interaction with the big data framework. However, some performance models incorporate metrics at the framework stage [71,80] or task level [9,51,53], which requires framework access. As a result, parameters from this layer are classified as white-box parameters when performance tracking occurs at the task or stage level. Additionally, certain parameters, like data locality, are exclusively white-box, making this a gray-box layer. Utilizing the Performance Layer simplifies the modeling process by allowing to focus on accessible, unbiased, and precise low-level hardware usage details rather than higher-level abstraction layers.

According to the definition of performance prediction models, all workload characteristics from other layers directly influence performance metrics and can be reflected in this layer. However, the effectiveness of the Performance Layer depends on the completeness and quality of the training dataset, and it introduces substantial training overhead due to the data collection required for model construction [93]. While performance metrics can assist in initial resource allocation for a target runtime, factors such as data locality and failures can also impact the actual runtime [70]. Execution time for data processing jobs is further affected by the application's resource requirements. Following the classification by [34] into four types of distributed applications – CPU- and I/O-intensive, memory-intensive, and iterative-intensive applications – we identified four related categories of performance metrics relevant to distributed batch analytics. However, note that resource requirements may vary across different stages of the processing workflow [9,32].

CPU-Centric Apart from actual runtime, the most commonly used performance metric in performance models is CPU utilization. According to [49], CPU demand in a Spark system can be categorized into three types:

1. **Task Execution Time:** This represents the CPU work required by the cluster to process the task itself.
2. **Coordination Overhead:** This includes the CPU needed for coordination with the driver program, task preparation before execution, and post-processing after execution.
3. **Infrastructure Overhead:** This refers to the CPU demand for infrastructure services provided to a task, which is independent of data input and represents a static demand.

I/O-Centric In distributed analytics systems, I/O-centric resource requirements are critical because data is often distributed across multiple nodes. For many iterative data mining and machine learning algorithms, network I/O overhead is a primary factor affecting system performance [43]. I/O-centric applications depend heavily on efficient data read and write operations, and performance can be significantly impacted by these I/O activities. Nodes share data, and at times, a node may need to read data only available on another node, which can negatively affect runtime performance [49]. I/O-centric applications can be further classified into those that are read-heavy and those that are write-heavy. While black-box parameters can directly measure I/O read and write speeds, some researchers prefer white-box parameters, such as the data processing ratio and data localization ratio [92,59]. These ratios indicate the proportion of data processed and stored locally, which isn't directly captured by simple read/write speed metrics. Understanding data localization provides insights into how efficiently data is utilized and moved across the system, directly impacting performance.

Communication-Centric Even if an application does not have additional I/O demands, it may still require data computed on another node to update its own calculations [1]. Communication-centric applications depend on frequent, large data exchanges over the network to transfer information between nodes. Limiting network capabilities can result in longer execution times [96]. The network infrastructure of the cluster is the primary factor impacting communication performance. In parallel computing, common communication patterns include one-to-one, one-to-all (broadcasting), all-to-all, and all-to-one (reduction) [1]. Understanding these communication patterns and their impact on performance is valuable for optimizing distributed analytical workflows.

Memory-Centric Memory management is crucial in distributed data-parallel workloads, where efficient memory use can significantly affect performance and costs. Effective memory usage involves accurately predicting the necessary memory for processing to prevent resource overuse or underuse. For instance, when a Spark cluster runs out of memory, the efficiency of in-memory processing declines, as data must be reloaded from disk, negating Spark's in-memory advantages [45,61]. Additionally, poor memory allocation — such as frequent garbage collection — can cause job failures and longer processing times, resulting in delays and inefficiencies that impact the overall performance of data-parallel workloads [61]. Conversely, excessive memory allocation yields only marginal performance improvements while wasting significant resources. In cloud environments, this leads to higher costs without proportional gains in processing speed [61].

While the Performance Layer provides valuable metrics, it may lack the contextual insights offered by other layers. For example, without the Big Data Framework Layer, it can be challenging to determine why specific configurations result in certain performance outcomes. Similarly, without the Hardware Layer, understanding the limitations imposed by physical resources may be difficult.

Table 3. Hardware Parameters.

Count	Parameter	Category	References
16	Number of CPU Cores	black-box	[95,74,62,51,86,48,33,53,92,1,19,63]
12	Number of Nodes	black-box	[95,74,62,51,86,48,33,53,92,1,19,63]
8	Memory Size	black-box	[69,84,50,35,81,21,60,38]
6	Memory Size per Node	black-box	[95,96,62,51,53,73]
6	Number of CPU Cores per Node	black-box	[95,62,51,86,33,63]
3	Disk Write Speed	black-box	[6,96,63]
3	Disk Read Speed	black-box	[6,96,63]
3	CPU Processing Power	black-box	[6,53,63]
1	Number of Disks	black-box	[6]
1	RAM per Core	black-box	[6]
1	RAM Read Speed	black-box	[63]
1	RAM Write Speed	black-box	[63]
1	Type of Storage Medium	black-box	[53]
1	Number of Disks per Node	black-box	[33]
1	Network Bandwidth	black-box	[63]

Table 4. User Application Parameters.

Count	Parameter	Category	References
8	Workload Type	black-box	[69,97,41,70,61,40]
4	Input Parameters	black-box	[84,48,35,81,70,100]
1	Application Type (I/O Intensive, CPU Intensive, etc.)	black-box	[74]

Hardware Layer The Hardware Layer is the third most commonly used layer, featured in 30 performance models. This layer includes workload properties (cf. Table 3) that describe static details about the physical architecture and hardware configuration within the computing cluster, providing a stable foundation for understanding the system's capacity and performance limits. Examples of parameters in this layer include the number of compute nodes, CPU specifications, memory, and I/O capabilities. Unlike the Big Data Framework Layer, the Hardware Layer consists entirely of black-box parameters, classifying it as a black-box layer. Hardware information is relatively easy to obtain and does not require invasive monitoring tools. Additionally, hardware parameters are generally static and predictable, simplifying the modeling process. The architecture configuration and capabilities in this layer form the foundation upon which all other layers rely, influencing reliability and performance optimization in distributed computing systems. This layer can also reflect the heterogeneity of physical hardware within the computing cluster, as variations in hardware impact execution time [9]. The most frequently used parameters in this layer are the number of CPU cores (used in 16 models), followed by the number of nodes (used in 12 models), and memory size (used in 8 models).

However, these parameters alone do not offer insights into the effectiveness or efficiency of hardware usage for specific tasks, which is crucial for accurate performance modeling. As a result, the Hardware Layer needs to be combined with another layer that reflects actual hardware usage. All 30 performance models incorporated this layer with at least one other layer, most frequently with the Performance Layer.

User Application Layer The User Application Layer includes static characteristics of applications running within the distributed environment. This layer, used in 12 of the 62 performance models, is classified as a black-box layer due to the nature of its parameters. These parameters (cf. Table 4) typically relate to the application's computational

Table 5. Data Parameters.

Count	Parameter	Category	References
26	Data Input Size	black-box	[66,95,98,84,92,59,38,93,23,74,11,69,20,81] [73,21,63,7,62,51,60,28,70,67,19]
1	Data Output Size	white-box	[28]
1	Input Data Profiles (Size, Number of files, number of entries)	white-box	[35]

requirements, such as the type of workload (represented by the workload name [69] or a hash [81]), input parameters, and the category of the application (e.g., I/O-intensive, iterative-intensive, CPU-intensive, and memory-intensive). Additionally, the User Application Layer enhances the generalization of performance models. By capturing a diverse set of applications, models can be refined to provide accurate predictions across a wider range of scenarios, improving robustness and applicability. This layer also helps identify and mitigate application-specific hardware bottlenecks. For instance, an I/O-intensive application may require different hardware resources compared to a CPU-bound application.

Data Layer The Data Layer contains information (cf. Table 5) about the data processed by user applications. Although this layer could theoretically include various data characteristics that impact performance, such as skewness and completeness [92], almost all studies have focused primarily on data size. Some studies also consider data properties at each stage, recognizing that many iterative algorithms only need to process data that has changed since the last iteration [80]. Only one study [35] extended this to include details, such as the number of records and files at the stage level. A possible reason for the limited analysis of input data characteristics before job execution is the overhead involved in handling large datasets. This layer is classified as a gray-box layer, as the input data size can be observed at the workflow, stage, or task level.

Virtualization Layer The Virtualization Layer, though the least frequently used layer with only 8 occurrences, offers several key insights. The parameters in this layer, listed in Table 6, often relate to the isolation, efficiency and overhead of resources introduced by virtualization. This layer is essential due to the prevalent use of Java in major big data frameworks and the growing reliance on virtualization technologies and cloud computing [37]. Including this layer helps identify inefficiencies and optimize the performance of virtualized resources, promoting effective utilization of virtual machines and containers. As many big data frameworks are Java-based, some research has incorporated white-box Java Virtual Machine (JVM) characteristics into performance models. These JVM parameters include garbage collection time and Java serialization/deserialization time [52,70,92,93]. Additionally, the JVM provides performance metrics, such as CPU utilization, which are critical for understanding the performance implications of running big data applications in a virtualized Java environment [48].

Additionally, some clusters use kernel-level or system-level virtualization, employing technologies like containers or virtual machines (VMs) [6] for efficient resource utilization. Virtualized environments introduce additional configuration layers that can affect performance, adding complexity to performance modeling. In multi-tenant environments, neighboring containers or VMs can interfere with each other, leading to unpredictable performance variations due to resource contention [58,96]. Incorporating the Virtualization Layer allows models to account for factors like virtual machine configuration or resource sharing among virtual instances, resulting in more accurate predictions in virtualization-heavy en-

Table 6. Virtualization Parameters.

Count	Parameter	Category	References
3	Number of Cloud Instances	black-box	[6,22,93]
2	Price per Hour	black-box	[6,93]
2	Cloud Machine Type	black-box	[22,93]
1	Resource Interference	white-box	[96]
1	Docker Configuration	white-box	[96]
1	Data (De)Serialization Time	white-box	[92]
1	Garbage Collection Time	white-box	[92]

vironments. Cloud providers offer various computing resources with differing capabilities and pricing models. Some researchers have incorporated cloud-specific information into performance models to evaluate cost-performance trade-offs and optimize resource allocation for specific workloads [6,93]. Cloud-specific parameters can include instance pricing, resource types, and network bandwidth. An advantage of cloud resources is the ability to easily reproduce environments using the same machines, frameworks, and open-source applications, enabling the creation of shared training data repositories [79].

The relatively low frequency of the Virtualization Layer suggests it is an emerging area in performance modeling. As virtualized environments become more common in distributed systems, the inclusion of this layer is likely to increase, reflecting its growing importance in accurately capturing performance dynamics in cloud-based and virtualized settings. With the continued expansion [12] of cloud computing, incorporating the Virtualization Layer becomes increasingly relevant. This layer enables performance models to align better with modern cloud infrastructures, where virtual machines and containers are ubiquitous [37]. In summary, the Virtualization Layer is considered a gray-box layer; however, we believe it contains more non-observable than observable properties. Static cloud properties, such as hardware configuration and pricing, are easy to obtain but are more closely related to the hardware level than to virtualization itself. In contrast, VM information, JVM configurations, and detailed virtualization settings may not be directly visible in a managed cluster.

4.2 Analysis of Layer Combinations

Next, we explore the use of the six layers both individually and in combination. Table 7 presents the seven most frequently used layer combinations for performance prediction.

Table 7. Commonly Used Layer Combinations.

Combination	Count
Big Data Framework Layer, Performance Layer	10
Big Data Framework Layer	7
Big Data Framework Layer, Data Layer, Hardware Layer	5
Big Data Framework Layer, Hardware Layer, Performance Layer	4
Data Layer, Hardware Layer	3
Big Data Framework Layer, Data Layer	3
Big Data Framework Layer, Data Layer, Hardware Layer, Performance Layer	3
Hardware Layer, Performance Layer	2
Data Layer, Hardware Layer, Performance Layer	2
Big Data Framework Layer, User Application Layer	2
Data Layer, Hardware Layer, User Application Layer	2
Big Data Framework Layer, Data Layer, Hardware Layer, User Application Layer	2

The most commonly used combination is the Big Data Framework Layer and the Performance Layer, highlighting their effectiveness in capturing the complex factors influencing analytical workflow performance. Notably, these are the only two layers used as standalone layers; all others are used in combination with at least one more layer. However, comparing their standalone usage to their use in combination with other layers (cf. Table 8) suggests that understanding either the internal workings of big data frameworks or performance metrics alone is insufficient. Effective performance modeling may require contextualizing these layers with characteristics from other layers to create robust models.

The Data Layer is frequently used in pairing combinations, as different applications often vary in data characteristics and requirements, and data size is relatively easy to obtain. The increasing use of the Data Layer (cf. Figure 4) suggests that it complements all layers, including the widely used Big Data Framework Layer, to improve prediction accuracy. This indicates that data characteristics are crucial for enhancing the accuracy and reliability of performance models. The most common black-box combination in performance prediction models includes the Data Layer, the Hardware Layer, and the Performance Layer. This combination emphasizes the importance of both static and dynamic information about physical resources and data size. Its frequent use in black-box modeling implies that capturing a holistic view of the system's operating environment is essential for accurate performance predictions. Static hardware configurations provide foundational capacity, while dynamic performance metrics and data characteristics offer insights into real-time usage and processing demands. In particular, the Virtualization Layer and the User Application Layer are absent among the twelve most popular layer combinations. This absence highlights key considerations for performance modeling in distributed analytical workflows. The underrepresentation of the Virtualization Layer suggests that virtualization is still an emerging area in performance modeling for big data analytics. As virtualized environments become more common [12], future models are expected to increasingly incorporate this layer to capture the nuances of VMs and containerization. Similarly, the underrepresentation of the User Application Layer implies that the unique behaviors of different applications may not be fully represented in most performance models — a hypothesis worth further investigation.

In general, the variety of common layer combinations in performance models underscores the inherent complexity of creating effective predictive models for distributed analytical workflows. This complexity suggests that simple, single-layer models are insufficient to capture the multifaceted nature of these workflows. Achieving accurate predictions may require the combination of multiple layers, which involves advanced modeling techniques and a deep understanding of how different factors interact. This need reinforces the value of the 5+1 layer classification model in developing comprehensive and accurate performance models. Incorporating multiple layers helps mitigate potential biases in performance models. For example, relying solely on the Big Data Framework Layer could introduce biases tied to specific configurations. When combined with the Hardware and Performance Layers, models can balance these biases with broader system-level insights, leading to more reliable and generalizable predictions. This layered approach enables a nuanced understanding of the factors influencing distributed job performance, supporting

Table 8. Standalone and Combined Usage of Layers.

Layer	Standalone Usage	Combined Usage
Big Data Framework Layer	6	37
Performance Layer	1	32

the development of more accurate and adaptable models. Although adding layers can improve model accuracy, it also increases complexity. Future efforts should aim to balance complexity with simplicity to ensure that models remain practical and usable while still offering detailed insights into performance drivers. By including underrepresented layers, performance models can become more comprehensive, broadening their applicability to various environments and applications. This approach not only improves prediction accuracy, but also improves the robustness and versatility of models in diverse settings.

4.3 Evolution of Layer Usage

The evolution of layer usage in performance models reveals shifts in the focus and methodologies for predicting performance in distributed analytical workflows, as illustrated and summarized in Figure 4. Although the Big Data Framework Layer remains consistently present throughout most years, underscoring its foundational role in performance modeling, its proportional usage is decreasing. This declining reliance on the Big Data Framework Layer indicates a growing preference for black-box models, which offer greater adaptability and reduced dependency on specific frameworks. This shift underscores the need for more generalizable and less intrusive approaches to performance prediction.

Additionally, there is an increasing trend to include the Data Layer, reinforcing the idea that data size is a critical factor in improving performance prediction. The increasing prominence of the Data Layer highlights the essential role data characteristics play in influencing performance. As the volume and variety of data in distributed systems grow, incorporating detailed, data-specific parameters has become vital for enhancing model accuracy. Furthermore, researchers are beginning to incorporate virtualization-related properties, reflecting the growing importance of virtualized environments in modern distributed systems. This emerging focus on virtualization indicates the need to capture the nuances introduced by these environments. As virtualized systems become more prevalent, accounting for resource allocation efficiency and isolation challenges is crucial for accurate performance modeling. Including the Virtualization Layer may be essential for developing comprehensive models that reflect the realities of modern distributed systems. The increasing attention to the User Application Layer also suggests a trend toward more application-specific performance models. Understanding the unique behaviors and requirements of various applications is key to creating precise and robust predictions, enabling performance models to adapt to a wider range of scenarios. In summary, the evolving use of layers in performance models reveals a continuous effort to balance complexity with accuracy and generalization. By integrating diverse layers and addressing emerging trends, future performance models can become more comprehensive, robust, and versatile, enhancing the utility of distributed analytical workflows.

4.4 Future Research Directions

The exploratory analysis and synthesis of workload characteristics in distributed processing environments have revealed significant insights and introduced a 5+1 layer classification model to inform performance prediction. However, there are several promising directions for future research to build upon these findings.

Firstly, deeper utilization of the Data Layer presents a potential research avenue. Although the current focus is mainly on data size, exploring other data characteristics — such as skewness, distribution, and quality — could provide more nuanced performance predictions. Developing methods to capture and utilize these characteristics efficiently,

without excessive overhead, would further improve model accuracy. As virtualization continues to grow in prevalence [12], future research should also focus on modeling virtualized environments in more detail, including VM configurations, container orchestration, and their impacts on performance. Understanding the interaction between physical and virtual resources will be essential for accurate performance predictions. In addition, incorporating cloud-specific characteristics — such as instance types, pricing models, and resource allocation policies — into performance models could optimize cost-performance trade-offs in cloud-based distributed systems. Application-specific performance modeling is another area ripe for exploration. Expanding the models to cover a broader range of applications and workload types, especially those not commonly represented, will improve the generalization and robustness of the predictions. Developing adaptive models that can dynamically adjust to changing workload patterns and application requirements would enhance efficiency and accuracy of performance prediction. Finally, examining the interactions and dependencies between layers in the 5+1 layer classification model could provide a deeper understanding of how various factors collectively influence performance. This includes studying synergies between layers such as hardware, data, and application. Creating composite metrics that capture the combined effects of multiple layers could offer a more holistic view of system performance and help identify optimization opportunities.

5 Threats to Validity

In this study, several potential threats to validity should be considered, particularly in relation to the literature search process and the scope of the review. Despite a comprehensive and systematic search strategy designed to cover key sources and fields relevant to performance modeling for distributed analytic computing, it is possible that some relevant studies were inadvertently overlooked. The literature search was conducted primarily using the Web of Science database, which, while extensive, may not include all studies in this rapidly evolving area or may not contain certain studies at the time of use. Furthermore, limitations in keyword selection or scope definition may have further influenced the coverage of the relevant literature.

The reading and classification of workload characteristics were undertaken with careful attention to detail and consistency, guided by established methodologies and our best understanding of the field. Each paper was thoroughly reviewed and the identified parameters were cross-checked and categorized according to the newly developed 5+1 layer classification model. However, there remains an inherent risk of misinterpretation or misclassification of parameters due to subjective judgments and potential ambiguities in the articles themselves. Although every effort was made to minimize these risks, it is important to acknowledge that these classifications, despite being systematic and well-considered, may still contain inaccuracies or biases.

6 Conclusion

This study investigated workload characterization in distributed processing environments, proposing a 5+1 layer classification model to inform performance prediction models. Through a systematic review of the literature, we identified six critical layers used in performance models: Big Data Framework, Performance, Hardware, Data, User Application, and Virtualization Layers. The findings highlight the foundational roles of the Big Data Framework and Performance Layers, though predictive accuracy benefits from combining them with other layers, especially the Data Layer, which underscores the impact

of data characteristics such as size and distribution. The Hardware Layer provides essential insights into physical limitations, while the emerging Virtualization Layer reflects the need to model virtualized environments, which are increasingly prevalent in distributed systems, especially in light of cloud computing.

Based on our findings, future work should expand the characteristics of the data and virtualization properties to further refine the accuracy of the model and address the complexity of cloud-based environments. Another consideration is the increased focus on a black-box perspective when designing performance modeling techniques to foster reusability. This layered approach underscores the need for adaptable and accurate models that robustly capture the multifaceted nature of modern distributed systems.

References

1. Ahmed, N., Barczak, A.L.C., Rashid, M.A., Susnjak, T.: An enhanced parallelisation model for performance prediction of apache spark on a multinode hadoop cluster. Big Data Cogn. Comput. **5**(4), 65 (2021)
2. Ahmed, N., Barczak, A.L.C., Rashid, M.A., Susnjak, T.: A parallelization model for performance characterization of spark big data jobs on hadoop clusters. J. Big Data **8**(1), 107 (2021)
3. Ahmet, A., Abdullah, T.: Real-time social media analytics with deep transformer language models: A big data approach. In: BigDataSE. IEEE (2020)
4. Aliabadi, S.K., Ardagna, D., Entezari-Maleki, R., Gianniti, E., Movaghar, A.: Analytical composite performance models for big data applications. J. Netw. Comput. Appl. **142**, 63–75 (2019)
5. Aliabadi, S.K., Aseman-Manzar, M., Entezari-Maleki, R., Ardagna, D., Egger, B., Movaghar, A.: Fixed-point iteration approach to spark scalable performance modeling and evaluation. IEEE Trans. Cloud Comput. **11**(1), 897–910 (2023)
6. Alipourfard, O., Liu, H.H., Chen, J., Venkataraman, S., Yu, M., Zhang, M.: Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics. In: NSDI. USENIX (2017)
7. Amannejad, Y., Shah, S., Krishnamurthy, D., Wang, M.: Fast and lightweight execution time predictions for spark applications. In: CLOUD. IEEE (2019)
8. Ardagna, D., Barbierato, E., Evangelinou, A., Gianniti, E., Gribaudo, M., Pinto, T.B.M., Guimarães, A., da Silva, A.P.C., Almeida, J.M.: Performance prediction of cloud-based big data applications. In: ICPE. ACM (2018)
9. Aseman-Manzar, M., Aliabadi, S.K., Entezari-Maleki, R., Egger, B., Movaghar, A.: Cost-aware resource recommendation for dag-based big data workflows: An apache spark case study. IEEE Trans. Serv. Comput. **16**(3), 1726–1737 (2023)
10. Ataie, E., Evangelinou, A., Gianniti, E., Ardagna, D.: A hybrid machine learning approach for performance modeling of cloud-based big data applications. Comput. J. **65**(12), 3123–3140 (2022)
11. Ataie, E., Gianniti, E., Ardagna, D., Movaghar, A.: A combined analytical modeling machine learning approach for performance prediction of mapreduce jobs in cloud environment. In: SYNASC. IEEE (2016)
12. B, K.S., R, S.K.: Load balancing in cloud computing environment. 2022 IEEE International Conference for Women in Innovation, Technology & Entrepreneurship (ICWITE) pp. 1–9 (2022)
13. Bei, Z., Yu, Z., Zhang, H., Xiong, W., Xu, C., Eeckhout, L., Feng, S.: RFHOC: A random-forest approach to auto-tuning hadoop's configuration. IEEE Trans. Parallel Distributed Syst. **27**(5), 1470–1483 (2016)
14. Benjelloun, S., Aissi, M.E.M.E., Loukili, Y., Lakhrissi, Y., Ali, S.E.B., Chougrad, H., Boushaki, A.E.: Big data processing: Batch-based processing and stream-based processing. 2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS) pp. 1–6 (2020)
15. Bilal, M., Canini, M., Rodrigues, R.: Finding the right cloud configuration for analytics clusters. In: SoCC. ACM (2020)
16. vom Brocke, J., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R., Cleven, A.: Reconstructing the giant: On the importance of rigour in documenting the literature search process. In: European Conference on Information Systems (2009)
17. Calavaro, C., Russo, G.R., Cardellini, V.: Real-time analysis of market data leveraging apache flink. In: DEBS. ACM (2022)
18. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.: Apache flink™: Stream and batch processing in a single engine. IEEE Data Eng. Bull. **38**(4) (2015)
19. Chalvantzis, N., Konstantinou, I., Koziris, N.: BBQ: elastic mapreduce over cloud platforms. In: CCGRID. IEEE/ACM (2017)

20. Chao, Z., Shi, S., Gao, H., Luo, J., Wang, H.: A gray-box performance model for apache spark. Future Gener. Comput. Syst. **89**, 58–67 (2018)
21. Chen, Y., Goetsch, P., Hoque, M.A., Lu, J., Tarkoma, S.: dd-simplexed: Adaptive delaunay triangulation for performance modeling and prediction on big data analytics. IEEE Trans. Big Data **8**(2), 458–469 (2022)
22. Cheng, G., Ying, S., Wang, B.: Tuning configuration of apache spark on public clouds by combining multi-objective optimization and performance prediction model. Journal of Systems and Software **180**, 111028 (2021)
23. Cheng, G., Ying, S., Wang, B., Li, Y.: Efficient performance prediction for apache spark. J. Parallel Distributed Comput. **149**, 40–51 (2021)
24. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51**(1) (2008)
25. Didona, D., Quaglia, F., Romano, P., Torre, E.: Enhancing performance prediction robustness by combining analytical modeling and machine learning. In: ICPE. ACM (2015)
26. Dong, Y., Wang, R., He, J.: Real-time network intrusion detection system based on deep learning. 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS) pp. 1–4 (2019)
27. Fu, Y., Soman, C.: Real-time data infrastructure at uber. In: SIGMOD. ACM (2021)
28. Gandomi, A., Movaghar, A., Reshadi, M., Khademzadeh, A.: Designing a mapreduce performance model in distributed heterogeneous platforms based on benchmarking approach. J. Supercomput. **76**(9), 7177–7203 (2020)
29. García-Peñalvo, F.J.: Developing robust state-of-the-art reports: Systematic literature reviews (2022)
30. Geldenhuys, M.K., Will, J., Pfister, B.J.J., Haug, M., Scharmann, A., Thamsen, L.: Dependable iot data stream processing for monitoring and control of urban infrastructures. In: IC2E. IEEE (2021)
31. Gianniti, E., Rizzi, A.M., Barbierato, E., Gribaudo, M., Ardagna, D.: Fluid petri nets for the performance evaluation of mapreduce and spark applications. SIGMETRICS Perform. Evaluation Rev. **44**(4) (2017)
32. Gibilisco, G.P., Li, M., Zhang, L., Ardagna, D.: Stage aware performance modeling of DAG based in memory analytic platforms. In: CLOUD. IEEE Computer Society (2016)
33. Glushkova, D., Jovanovic, P., Abelló, A.: Mapreduce performance model for hadoop 2.x. Inf. Syst. **79** (2019)
34. Gu, J., Li, Y., Tang, H., Wu, Z.: Auto-tuning spark configurations based on neural network. In: ICC. IEEE (2018)
35. Gulino, A., Canakoglu, A., Ceri, S., Ardagna, D.: Performance prediction for data-driven workflows on apache spark. In: MASCOTS. IEEE (2020)
36. Gusenbauer, M., Haddaway, N.R.: Which academic search systems are suitable for systematic reviews or meta-analyses? evaluating retrieval qualities of google scholar, pubmed, and 26 other resources. Research Synthesis Methods **11**, 181 – 217 (2020)
37. Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., Khan, S.U.: The rise of "big data" on cloud computing: Review and open research issues. Inf. Syst. **47**, 98–115 (2015)
38. Hu, Z., Li, D., Guo, D.: Balance resource allocation for spark jobs based on prediction of the optimal resource. Tsinghua Science and Technology **25**(4), 487–497 (2020)
39. Huang, S., Huang, J., Dai, J., Xie, T., Huang, B.: The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In: ICDE. IEEE (2010)
40. Islam, M.T., Karunasekera, S., Buyya, R.: dspark: Deadline-based resource allocation for big data applications in apache spark. In: eScience. IEEE (2017)
41. Jie, H.: A performance modeling-based hadoop configuration tuning strategy. Nanotechnology for Environmental Engineering **7**, 725 – 736 (2022)
42. Kadirvel, S., Fortes, J.A.B.: Grey-box approach for performance prediction in map-reduce based platforms. In: ICCCN. IEEE (2012)
43. Kang, M., Lee, J.: An experimental analysis of limitations of mapreduce for iterative algorithms on spark. Clust. Comput. **20**(4), 3593–3604 (2017)
44. Karimian-Aliabadi, S., Ardagna, D., Entezari-Maleki, R., Movaghar, A.: Scalable performance modeling and evaluation of mapreduce applications. Communications in Computer and Information Science (2019)
45. Khan, M.M., Alam, M.A.U., Nath, A.K., Yu, W.: Exploration of memory hybridization for RDD caching in spark. In: ISMM. ACM (2019)
46. Khan, M., Jin, Y., Li, M., Xiang, Y., Jiang, C.: Hadoop performance modeling for job estimation and resource provisioning. IEEE Trans. Parallel Distributed Syst. **27**(2), 441–454 (2016)
47. Kolajo, T., Daramola, O.J., Adebiyi, A.A.: Big data stream analysis: a systematic literature review. J. Big Data **6**, 47 (2019)

48. Kroß, J., Krcmar, H.: Model-based performance evaluation of batch and stream applications for big data. In: MASCOTS. IEEE (2017)
49. Kroß, J., Krcmar, H.: Pertract: Model extraction and specification of big data systems for performance prediction by the example of apache spark and hadoop. Big Data Cogn. Comput. **3**(3) (2019)
50. Lee, G.J., Fortes, J.A.B.: Hadoop performance self-tuning using a fuzzy-prediction approach. In: ICAC. IEEE (2016)
51. Li, P., Dong, L., Xu, H., Lau, T.F.: Spark's operation time predictive in cloud computing environment based on SRC-WSVR. J. High Speed Networks **24**(1), 49–62 (2018)
52. Li, Y., Ma, J., Cao, D.: Cross-domain workloads performance prediction via runtime metrics transferring. JCC (2020)
53. Li, Y., Liu, F., Chen, Q., Sheng, Y., Zhao, M., Wang, J.: Marvelscaler: A multi-view learning-based auto-scaling system for mapreduce. IEEE Trans. Cloud Comput. **10**(1), 506–520 (2022)
54. Li, Y., Lee, B.C.: Phronesis: Efficient performance modeling for high-dimensional configuration tuning. ACM Trans. Archit. Code Optim. **19**(4), 56:1–56:26 (2022)
55. Liu, Q., Cai, W., Jin, D., Shen, J., Fu, Z., Liu, X., Linge, N.: Estimation accuracy on execution time of run-time tasks in a heterogeneous distributed environment. Sensors **16**(9), 1386 (2016)
56. Liu, Y., Xu, H., Lau, W.C.: Cloud configuration optimization for recurring batch-processing applications. IEEE Trans. Parallel Distributed Syst. **34**(5), 1495–1507 (2023)
57. Luo, N., Yu, Z., Bei, Z., Xu, C., Jiang, C., Lin, L.: Performance modeling for spark using SVM. In: CCBD. IEEE (2016)
58. Makroo, A., Dahiya, D.: A systematic approach to deal with noisy neighbour in cloud infrastructure. Indian journal of science and technology **9** (2016)
59. Maros, A., Murai, F., da Silva, A.P.C., Almeida, J.M., Lattuada, M., Gianniti, E., Hosseini, M., Ardagna, D.: Machine learning for performance prediction of spark cloud applications. In: CLOUD. IEEE (2019)
60. Maroulis, S., Zacheilas, N., Theocharis, T., Kalogeraki, V.: Fast. efficient performance predictions for big data applications. In: ISORC. IEEE (2019)
61. Myung, R., Choi, S.: Machine-learning based memory prediction model for data parallel workloads in apache spark. Symmetry **13**(4), 697 (2021)
62. Myung, R., Yu, H.: Performance prediction for convolutional neural network on spark cluster. Electronics (2020)
63. Peyravi, N., Moeini, A.: Estimating runtime of a job in hadoop mapreduce. J. Big Data **7**(1), 44 (2020)
64. Polato, I., Ré, R., Goldman, A., Kon, F.: A comprehensive view of hadoop research - A systematic literature review. J. Netw. Comput. Appl. **46**, 1–25 (2014)
65. Radhika, D., Kumari, D.A.: Adding big value to big businesses: A present state of the art of big data, frameworks and algorithms (2018)
66. Ramanathan, R., Latha, B.: Towards optimal resource provisioning for hadoop-mapreduce jobs using scale-out strategy and its performance analysis in private cloud environment. Clust. Comput. **22**(6), 14061–14071 (2019)
67. Rizzi, A.M.: Support vector regression model for bigdata systems. CoRR **abs/1612.01458** (2016)
68. Salloum, S., Dautov, R., Chen, X., Peng, P.X., Huang, J.Z.: Big data analytics on apache spark. Int. J. Data Sci. Anal. **1**(3-4), 145–164 (2016)
69. Scheinert, D., Thamsen, L., Zhu, H., Will, J., Acker, A., Wittkopp, T., Kao, O.: Bellamy: Reusing performance models for distributed dataflow jobs across contexts. In: CLUSTER. IEEE (2021)
70. Scheinert, D., Zhu, H., Thamsen, L., Geldenhuys, M.K., Will, J., Acker, A., Kao, O.: Enel: Context-aware dynamic scaling of distributed dataflow jobs using graph propagation. In: IPCCC. IEEE (2021)
71. Sewal, P., Singh, H.: A machine learning approach for predicting execution statistics of spark application. 2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC) pp. 331–336 (2022)
72. Shahrivari, S.: Beyond batch processing: Towards real-time and streaming big data. Comput. **3**(4), 117–129 (2014)
73. Shao, Q., Pan, L., Liu, S., Liu, X.: A collaborative filtering based approach to performance prediction for parallel applications. In: CSCWD. IEEE (2017)
74. Shen, C., Chen, C., Rao, G.: A novel multi-task performance prediction model for spark. Applied Sciences (2023)
75. Shoetan, P.O., Oyewole, A.T., Okoye, C.C., Ofodile, O.C.: Reviewing the role of big data analytics in financial fraud detection. Finance & Accounting Research Journal (2024)
76. Taher, N.C., Mallat, I., Agoulmine, N., El-Mawass, N.: An iot-cloud based solution for real-time and batch processing of big data: Application in healthcare. 2019 3rd International Conference on Bio-engineering for Smart Technologies (BioSMART) pp. 1–8 (2019)

77. Thamsen, L.: Dynamic resource allocation for distributed dataflows. Ph.D. thesis, Technical University of Berlin, Germany (2018)
78. Thamsen, L., Renner, T., Kao, O.: Continuously improving the resource utilization of iterative parallel dataflows. In: ICDCS. IEEE (2016)
79. Thamsen, L., Scheinert, D., Will, J., Bader, J., Kao, O.: Collaborative cluster configuration for distributed data-parallel processing: A research overview. Datenbank-Spektrum **22**(2), 143–151 (2022)
80. Thamsen, L., Verbitskiy, I., Beilharz, J., Renner, T., Polze, A., Kao, O.: Ellis: Dynamically scaling distributed dataflows to meet runtime targets. In: CloudCom. pp. 146–153. IEEE (2017)
81. Thamsen, L., Verbitskiy, I., Schmidt, F., Renner, T., Kao, O.: Selecting resources for distributed dataflow systems according to runtime targets. In: IPCCC. IEEE (2016)
82. Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., Saha, B., Curino, C., O'Malley, O., Radia, S., Reed, B.C., Baldeschwieler, E.: Apache hadoop YARN: yet another resource negotiator. In: SoCC. ACM (2013)
83. Venkataraman, S., Yang, Z., Franklin, M.J., Recht, B., Stoica, I.: Ernest: Efficient performance prediction for large-scale advanced analytics. In: NSDI. USENIX (2016)
84. Verbitskiy, I., Thamsen, L., Renner, T., Kao, O.: Cobell: Runtime prediction for distributed dataflow jobs in shared clusters. In: CloudCom. IEEE (2018)
85. Wang, J., Xu, C., Zhang, J., Zhong, R.Y.: Big data analytics for intelligent manufacturing systems: A review. Journal of Manufacturing Systems (2021)
86. Wang, K., Khan, M.M.H.: Performance prediction for apache spark platform. In: HPCC/CSS/ICESS. IEEE (2015)
87. Wang, M., Wu, C.Q., Cao, H., Liu, Y., Wang, Y., Hou, A.: On mapreduce scheduling in hadoop yarn on heterogeneous clusters. In: TrustCom. IEEE (2018)
88. Wang, N., Yang, J., Lu, Z., Li, X., Wu, J.: Comparison and improvement of hadoop mapreduce performance prediction models in the private cloud. In: APSCC. Lecture Notes in Computer Science, vol. 10065 (2016)
89. Wang, T., Brovman, Y.M., Madhvanath, S.: Personalized embedding-based e-commerce recommendations at ebay. CoRR **abs/2102.06156** (2021)
90. Wiesner, P., Behnke, I., Scheinert, D., Gontarska, K.K., Thamsen, L.: Let's wait awhile: how temporal workload shifting can reduce carbon emissions in the cloud. In: Middleware. ACM (2021)
91. Will, J., Thamsen, L., Bader, J., Scheinert, D., Kao, O.: Get your memory right: The crispy resource allocation assistant for large-scale data processing. In: IC2E. IEEE (2022)
92. Xu, F., Jiang, H., Zheng, H., Shao, W.: ispot: Achieving predictable performance for big data analytics with cloud transient servers. In: ISPA/IUCC. IEEE (2017)
93. Xu, F., Zheng, H., Jiang, H., Shao, W., Liu, H., Zhou, Z.: Cost-effective cloud server provisioning for predictable performance of big data analytics. IEEE Trans. Parallel Distributed Syst. **30**(5), 1036–1051 (2019)
94. Yadwadkar, N.J., Hariharan, B., Gonzalez, J.E., Smith, B., Katz, R.H.: Selecting the *best* VM across multiple public clouds: a data-driven performance modeling approach. In: SoCC. ACM (2017)
95. Ye, G., Liu, W., Wu, C.Q., Shen, W., Lyu, X.: On machine learning-based stage-aware performance prediction of spark applications. In: IPCCC. IEEE (2020)
96. Ye, K., Ji, Y.: Performance tuning and modeling for big data applications in docker containers. In: NAS. IEEE (2017)
97. Yeh, C., Zhou, J., Chang, S., Lin, X., Sun, Y., Huang, S.: Bigexplorer: A configuration recommendation system for big data platform. In: TAAI. IEEE (2016)
98. Yu, Z., Bei, Z., Qian, X.: Datasize-aware high dimensional configurations auto-tuning of in-memory cluster computing. In: ASPLOS. ACM (2018)
99. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I.: Apache spark: a unified engine for big data processing. Commun. ACM **59**(11) (2016)
100. Zaouk, K., Song, F., Lyu, C., Diao, Y.: Neural-based modeling for performance tuning of spark data analytics. CoRR **abs/2101.08167** (2021)
101. Zhao, H., Rao, Y., Li, D., Tang, J., Liu, S.: A DAG refactor based automatic execution optimization mechanism for spark. In: NPC. Lecture Notes in Computer Science, vol. 11783, pp. 338–344. Springer (2019)