Performance Modeling of Spark: An overview

YANCHENG ZHUANG
13491679 (UvA) / 2688067 (VU)

May 24, 2022

Abstract

With the emergence of big data, various distributed parallel data processing frameworks are developed to address the processing and storage challenges. Among these frameworks, Apache Spark has become the de facto open source standard for big data processing for its ease of use and performance. However, Spark itself as a complex system is highly configurable, while nowadays in the era of cloud computing, users also need to determine the instances to be used before running the actual applications. Therefore, researchers purpose plenty of performance models in aim of addressing the issue. A good performance model can help user to tune the configuration parameters of Spark and choose the appropriate execution environment. This report provided an overview of the existing state-of-art in performance modeling of Spark.

1 Introduction

Nowadays, big data is becoming part of people's daily life. Companies in all kinds of sectors like banking[13], retail[2], healthcare[10] and IoT[25] are all starting to make use of the massive amount of data generated and collected to solve complex real-world data science question. Thus, numerous big data frameworks have been introduced to address the issues with big data. The two most influential open-source data parallel frameworks for large-scale data analytics developed are MapReduce[11] and Apache Spark[31].

MapReduce is one of the earliest and widely-used distributed data-parallel frameworks [11]. MapReduce exposes a simple programming interface, which includes two stages called map and reduce. Map function takes a key-value pair to generate a set of intermediate key-value pairs. The output of Mapper is input for reducer in such a way that all key-value pairs with the same key goes to same reducer. The Reducer will then aggregate the set of key-value pairs into a smaller set of pairs as the final output. MapReduce is widely adopted by users to tackle big data challenges, but the framework has several limitations. Applications like machine learning and graph processing which are becoming more and more popular require iterative processing of the data. In MapReduce, the only way to share data across jobs is stable storge, so every job needs to read the output of the previous job through HDFS, which incurs huge overhead.

To overcome the limitation of MapReduce, Spark was purposed with numerous innovations[31]. Spark keeps in-memory states across iterations using Resilient Distributed Datasets (RDDs). Data in Spark is expressed in RDDs which are immutable, partitioned collections of records and can only be built through coarse-grained deterministic transformations. Instead of replication for fault tolerance, lineage is used which has no cost if nothing fails. Two types operations on RDDs are transformations for building new RDDs and actions for computing results. Higher level abstraction of data using Dataframe is introduced in Spark SQL which is similar to the widely used data frame concept in R [5]. After submitted to Spark cluster, an application is divided into one or more jobs, and each job is divided into a few stages according to the dependency relationship between the RDDs. Tasks in the same stage have no data dependence between each other, thus they can be handled in parallel by executors. Due to the better performance [24], rich APIs and optimizations for the complex needs of modern data analysis, Spark has become the most popular large-scale data processing framework in industry.

However, big data processing systems like Spark are complex systems with a wide range of choices open to users. On the application side, users can have different sets of Spark configurations tailored to different workloads. Each configuration option of Spark will affect the performance of the workload in a certain way. Moreover, configuration comes from different layers of Spark may intertwine with each other in a complex way in terms of performance[29], which only complicates matters. On the system side, in the era of cloud computing, users will be forced to choose suitable number of specific instance types from the cloud computing vendors. Each instance type have slightly different computing and communication capability. Given so many options to opt for, it is often quite hard for users to answer questions like "What is the cheapest instance configuration to finish this job given these configuration in one week" or "What will be the optimal Spark Configuration for this workload". Researchers tried to address these questions using performance modeling, which aims to help users to choose from all kinds of options by understanding and modeling different performance characteristics of various data-intensive applications.

The main objective of this paper is to conduct a literature review of the performance modeling of Spark, where we want to figure out challenges and difficulties in choosing suitable configurations for Spark applications, and what/how performance modeling address this problem. This report is organized into the sections described as below. First, relevant background information is introduced in *Introduction.* Secondly, research questions and motivation behind are described in *Methodology.* Then, related literature is organized to answer the raised research questions in *Literature Review.* Finally, summary of this report is included in *Conclusions.*

2 Methodology

In this section, we present the methodology used in this report. Google Scholar, IEEE Xplore, ACM and ScienceDirect are used for searching and selecting publications. Only publications on establishing performance model on Spark or solving certain Spark research questions based on performance model are included. For those works that developed the performance model for solving other research questions, and we only focus on the model itself. Some works use simulation to address the problem of choosing suitable configurations, however these kind of models often require lots of simulation time and are hard to use[17], so we will only focus on performance models. In order to understand what and how performance modeling is used in Spark, we establish the following research questions:

- Q1: What is the objective the modeling? The goal of this question is to understand that the use case of the purposed model. This could be the execution time, the resource usage, the optimal configuration set, etc.
- Q2: What approach is used for building the model? The goal of this question is to understand what kind of model is used for modeling the performance of Spark. Performance model can be grouped around three approaches: white-box, black-box or grey-box. White-box model or analytical model needs to symbolize the interactions between various system internal components which needs a strong understanding of the underlying system, while black-box model or machine learning model needs historical performance data of the workloads to establish a model which learns the relationship of performance with the interactions of the system components automatically. Grey-box model is a combination of white-box model and black-box model and is intended for taking the best of both approaches.
- Q3: What is the testbed for the evaluation? The goal of this question is to evaluate the scalability of the models. The accuracy of models developed and evaluated using a small number of nodes is probably doubtful if they are used under many nodes, which is often the case for large-scale compute-intensive applications in industry.
- Q4: What are workloads for evaluation? The goal of this question is to understand the adoptivity of the model. Different types of workloads have different performance characteristics. Some are CPU bound while some others are network I/O bound. Apache Spark now as a unified engine for large-scale data analytics is expected to execute different types of workloads instead of only iterative ones.

Besides the questions listed above, we also want to identify challenges in this field. We want to discover tradeoffs, considerations or pratical issues when building the performance models in the literature.

3 Literature review

In this section, we will review the existing state-of-art in performance modeling of Spark. In section 3.1, we presented a thorough summary of existing works in chronological order. In section 3.2, a taxonomy of performance models built is built based on the research questions to identify the research trends.

3.1 Literature summary

Wang and Khan [28] purposed a fine-grained white-box model to predict execution time and I/O cost, under the same instance configuration. The model built is a hierarchical top-down fashion, firstly job time, secondly stage time and lastly task time. In order to predict the performance, various performance metrics like run time, I/O cost and memory cost need to be collected. The number of tasks will be executed in the actual job is the main factor for predicting the performance for the actual job. The model is evaluated on WordCount, Logistic Regression, KMeans clustering and PageRank using a 13 node cluster with heterogeneous nodes.

Wang et al. [27] purposed a black-box model to predict the execution time with Spark configurations and the size of input data. For collecting training data, random search is performed over the parameters and select values uniformly random from the values. A binary classification model is built to predict whether the execution time of a job based on a set of configuration parameters values is improved. The ones classified as improved will be further used a in finer-grained multi-classification model based to the improvement. The model is tested on a 4 node cluster with heterogeneous nodes using Grep, Sort, WordCount and Naive Bayes.

Venkataraman et al. [26] purposed a grey-box model to predict the execution time based on a specified instance configuration, given the job and input data size. To be more specific, they summarised high-level computation and communication patterns in Spark and tried to predict the overall execution time using linear regression model. Only input data size and number of nodes are used as features in the model. For computation patterns, time is positively correlated to input and negatively correlated to number of nodes. For communication patterns, tree dag communication has logarithmic relation with time, all-to-one communication has linear relation with time, and one-to-one communication is a constant factor in time. Instead of collecting the training of all the combination of input data size and number of machines, they made use of optimal experiment design from Statistics which is supposed to minimize the trace of the matrix that represents all the options, while the matrix is subjected to a bounded total budget. The model was tested on various machine learning algorithms in the MLlib in Spark using AWS. The number of nodes used ranges from 16 to 64.

Islam et al. [16] purposed a grey-box model to predict the execution time based on the number of executors. They model the relationship between time and the number of executors using power function. The basic idea behind the power function is that adding too many executors can cause overheads due to serialization, de-serialization and intensive shuffle operations in the network. Three types of applications are tested only on 2 worker nodes: WordCount, Sort and PageRank.

Marco et al. [19] used black-box models to predict the memory footprint. For an incoming application, the framework first extracts the features of the program using system-wide profiling tools including vmstat, perf and PAPI. Based on the feature values, it predicts which of the off-line learned memory functions best describes the memory behavior of the application based on KNN. It then instantiates the function parameters by profiling the application on some small sets of the input data items. The framework is evaluated using HiBench, BigDataBench, Spark-Perf and Spark-Bench on a cluster with 40 nodes.

Alipourfard et al. [1] used black-box model to generate the optimal or near-optimal instance configuration that minimize cloud usage cost, guarantee application performance and limit the search overhead. To be more specific, the model is based on Bayesian Optimization, which estimates a confidence interval of the cost and running time of each candidate cloud configuration. It has two functions. Prior function is used for black box modeling, while acquisition function is used for choosing the next configuration based on the calculated expected improvement in comparison to the current best configuration. When the expected improvement is less than a threshold and at least N cloud configurations have been observed, the model stops the search. The model is evaluated using TPC-DS, TPC-H, TeraSort, SparkML Regression and SparkML Kmeans using 5 AWS instance types with 30-854 GB RAM and 12-112 cores. The framework is evaluated on applications from HiBench, BigDataBench, Spark-Perf and Spark-Bench on a cluster with 40 nodes.

Chao et al. [7] purposed a grey-box model to predict the execution time with Spark configurations and the size of input data. Firstly relatively more important configuration options are selected. The grey-box model predicts execution time of each stage firstly using different regression algorithms for different stages, and then forecast the time cost of application based on predicted runtime of each stage, using regression algorithm again. TeraSort, KMeans, PageRank and WordCount are evaluated on this model using 32 worker nodes.

Nguyen et al. [22] purposed a black-box model to predict the execution time with Spark configurations and size of input data. Firstly, 13 parameters are selected for tuning and made use of customised Latin Hypercube Sampling to generate a possible combination of values and collect training data. Best machine learning model is selected for a specific application to train the model. Finally, recursive random search algorithm is used to identify candidate combinations of configuration settings. The model is evaluated on a 6-node cluster with 72 cores using batch processing application, machine learning applications and graph processing applications.

Maros et al. [20] purposed both a black-box model and a grey-box model to predict the execution time. The first model relies only on features that capture knowledge available before the execution. The second model make use of a richer set of features which can be captured after the execution like information provided by DAG. Both model make use of LASSO, neural network, decision tree and random forest to identify the best solution for each scenario. Both models are evaluated on TPC-DS, K-means and an ad-hoc benchmark developed on SparkDL using a Microsoft Azure with 6 nodes and 8 cores per node, as well as a 4-node private cluster with 12 cores per node.

Ardagna et al. [3, 4] evaluated two different white-box parallel computation performance modeling approaches to predict the execution time of Spark application. The first model is based on a simple upper-bound on the average execution time for Fork-Join queuing networks[21] and is referred as Fork-Join. In this model, tasks are forked into identical subtasks which will be joined once they are completed by corresponding servers. It only depends only on the number of parallel tasks and the average execution time of a single task which can be estimated at based on historical data. The key factor affects the estimation is the harmonic number. The second model modifies Mean Value Analysis technique for queuing network to account for delays caused by synchronization and resource constraints in DAG[18] and is referred as Task Precedence model. The model uses DAG and the average execution time of each individual stage as input. The model estimates the overlap probability between each pair of tasks based on DAG which will be used as an inflation factor. Both model are evaluated using TPC-DS, SVM, LR and K-Means using Azure with 3-13 nodes and 4-8 cores per node.

David Buchaca et al. [23] purposed a black-box model to predict the execution time in order to find a suitable application configuration for a new workload. The model uses both information from the log of the executed application and low-level features from the workload to be optimized, and simulates a Bayesian Optimization process to make predictions. The model is evaluated on machine learning workloads from HiBench and spark-perf. The computation environment is a local 4-node cluster with 20 cores per node.

Cheng et al. [9] purposed a black-box model to predict the execution time. The model is built based on Adaboost, which is for estimating the scheduling time and execution time for each type of job using different application configurations as features. In order to lower the sampling cost, projective sampling is used to approximate the learning curve which reflects the relationship between the number of samples required to train a model and prediction accuracy. The model is evaluated on a 9-node local cluster with 24 cores per node.

Karimian-Aliabadi et al. [17] purposed an analytical white-box model to predict the execution time of Spark applications using YARN scheduler. Yarn scheduler is widely adopted for enabling Spark to run alongside with other Hadoop workloads. The model is based on Stochastic Activity Networks, which is a probabilistic generalization of Activity Networks from Petri Nets. A monolithic model was firstly purposed for the simple double-queues scenario, which is proved not scalable because of the proportional growing of the state space size with the multiplicity. To address this issue, the model is decomposed and the fixed-point theorem is used to estimate the final solution. The model is evaluated on TPC-DS using a 4-node cluster with 10 cores per node. Yu et al. [29] used a similar BO-based approach such as CherryPick [1] when trying to optimize the application configurations in their model, however, they found that such BO-based optimization approaches can not adapt to the input data size changes, so datasize-aware Gaussian process is purposed to address this issue. The model is evaluated on two different clusters. One cluster consists of four nodes with 128 cores per node; another one consists of 8 nodes with 20 cores per node.

Chen st al. [8] developed a white-box model to predict the execution time of Spark applications. By taking inspiration from the field of Computational Geometry, they constructed a d-dimensional mesh using Delaunay Triangulation over a selected set of features. Delaunay Triangulation partitions the d feature space into a set of interconnected d-simplexes, which helps to avoid overfitting. The prediction of the execution time is done by calculating a hyperplane in d+1 dimensional space by bringing in the runtime dimension. Also adaptive sampling was integrated to minimize the samples needed. The model is evaluated on a 10-node server with 16 vcores per node using WordCound, PageRank, Kmeans, Bayesian and SQL Join from HiBench, and 4 other synthetic, more complex benchmarks.

3.2 Literature classification

3.2.1 Objective

Predicting running time Prediction of the time Spark needs to finish executing the application is the objective of most of the works [28, 27, 26, 16, 7, 22, 20, 3, 4, 23, 9, 17, 29, 8]. It is easier to understand because the running time is the most important performance metrics of the application which reflects the overall performance of the application. The estimation of running time can be further directly used in other specific tasks, like application configuration tuning [23, 9, 30, 29, 8] and instance selection [26, **?**].

Choosing application configuration This objective is closely related to prediction of running time, but the model purposed in [27] generates the recommendation of application configuration directly instead of the running time.

Prediction of resource utilisation Model purposed in [28] is able to predict the I/O cost of the application, while model purposed in [19] is able to predict the memory footprint. Both models are valuable for some specific tasks like resource scheduling, task co-location, etc.

Choosing instance configuration CherryPick [1] addressed resource utilisation in a different way, which used black-box model to generate the recommendation directly for meeting a certain deadline.

3.2.2 Approach

White-box White-box needs deep understanding of the interactions of different components of the underlying computing framework as well as suitable mathematical modeling approaches, which results fewer publications using these approach [28, 3, 4, 17, 8]. These models make use of different modeling approaches including queuing networks, Petri nets and computational geometry.

Black-box Black-box approach mainly relies on machine learning techniques. This kind of approach is popular due to its simplicity and it could produce good results without deep knowledge of the underlying system. Most models are based on this approach [27, 19, 1, 22, 20, 23, 9, 29] using all kinds of different linear or non-linear models to capture the pattern of the data. The data is instrumental in the black-box approach. Some works just utilize some basic knowledge known before the execution [27, 26, 16, 20], while other works use more fine-grained data collected from logs and hardware counters. Except the works based machine learning model, some works make use of Bayesian optimization to approximate the target cost function [1, 29].

Grey-box A hybrid approach is adopted in some works at the aim of combining the merits of black-box approach and white-box approach. [7], [16] and [26] tried this approach.

3.2.3 Testbed

The testbed used for evaluating the model is important for the scalability of the result. Some works are evaluated on very limited hardware resources (less than 10 nodes) [27, 16, 22, 20, 23, 29]. These works might need further evaluations before being put into use in industry. Testbed with about 50 nodes are rare probably because of the cost involved.

3.2.4 Workload

Apache Spark is now becoming a unified big data processing engine, so different types of representative workloads are expected in the evaluations. Most works use widely-used benchmark suite HiBench [15], which includes micro, machine learning, sql, graph, websearch and streaming applications. Some works also use BigDataBench[12], which is also a popular benchmark suite providing real-world data sets and workloads. There are also some other benchmarks which focused on some specific applications used by some works, like spark-perf benchmark suite, which consists of a set of core RDD benchmarks and a set of machine learning algorithm benchmarks using MLLib, TPC-DS / TPC-H which models complex decision support functions and is widely used in evaluating the performance of Spark SQL systems[6]. Some early works [28, 27, 26, 16] focuses only on batch processing and machine learning applications when Spark SQL is not mature. Most of the recent works include SQL-like workloads from benchmarks during the evaluations.

3.2.5 Challenges

During the literature study, several challenges in this field are discovered and described below:

- Cost is considerable for collecting training samples for black-box models. Numerous configuration parameters could be used in the model and it takes plenty of time for each run of the Spark application if the input data set is large. So if the application is not periodic long job, then build accurate black-box performance model will incur more cost than running the application using sub-optimal configurations, which makes models useless.
- White-box models are built on mathematical models that inevitably need to make assumptions for controlling the complexity and improving the scalability, which sacrifices the accuracy of the model.
- More and more Spark users run the their applications on cloud, which introduces substantial noises due to cloud dynamics like network congestion. Moreover, running Spark on cloud probably means some information that can be easily collected in local clusters will become unavailable. Especially nowadays PaaS platforms provided by companies like Databricks have gained popularity on the market, which actually includes lots of proprietary optimization on the open-sourced Apache Spark. White-box models will certainly become not suitable to be used under such case.

Some of the models purposed take some of this challenges into consideration and try to address them from different perspective using different approaches. For example, in order to lower the cost for collecting samples, some authors introduce techniques from statistics like projective sampling[9], adaptive sampling [8], optimal experiment design[26], etc. However, further research is needed to address these challenges better.

4 Conclusions

In this report, literatures on performance modeling for Spark have been surveyed and analysed. So far, there are plenty of works trying to find a better way to model the performance of Spark. Three conclusions can be drawn: (1) black-modeling approach is favored due to the simplicity and good results, (2) evaluation on the model using large-scale industry-level infrastructure is rare, which reflects that research on performance modeling on Spark is majorly carried out in academia, (3) most models can produce good results on different applications, but the cost behind building such a model is non-trivial. Overall, research on performance modeling of Spark has made good progress so far, but several practical challenges still need more research. Besides addressing the challenges, there is little study on some topics like performance modeling based on other schedulers like Mesos[14], how to sample data for building good performance model, etc.

Limitation of this literature study is that the publications are only related to performance modeling of Spark. There are plenty of literature on developing performance model on MapReduce. These models cannot be directly used on Spark though, they probably might be able to shed some lights on how to build performance model on Spark.

References

[1] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. CherryPick: Adaptively unearthing the best cloud configurations for big data analytics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 469–482, Boston, MA, March 2017. USENIX Association.

[2] John A. Aloysius, Hartmut Hoehle, Soheil Goodarzi, and V. Venkatesh. Big data initiatives in retail environments: Linking service process perceptions to shopping outcomes. *Annals of Operations Research*, 270:25–51, 2018.

[3] Danilo Ardagna, Enrico Barbierato, Athanasia Evangelinou, Eugenio Gianniti, Marco Gribaudo, Túlio B. M. Pinto, Anna Guimarães, Ana Paula Couto da Silva, and Jussara M. Almeida. Performance prediction of cloud-based big data applications. In *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering*, ICPE '18, page 192â199, New York, NY, USA, 2018. Association for Computing Machinery.

[4] Danilo Ardagna, Enrico Barbierato, Eugenio Gianniti, Marco Gribaudo, Túlio B. M. Pinto, A. P. C. da Silva, and Jussara M. Almeida. Predicting the performance of big data applications on the cloud. *The Journal of Supercomputing*, pages 1–33, 2020.

[5] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, and Matei Zaharia. Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference*

on Management of Data, SIGMOD '15, page 1383â1394, New York, NY, USA, 2015. Association for Computing Machinery.

[6] Melyssa Barata, Jorge Bernardino, and Pedro Furtado. An overview of decision support benchmarks: Tpc-ds, tpc-h and ssb. *New Contributions in Information Systems and Technologies*, pages 619–628, 2015.

[7] Zemin Chao, Shengfei Shi, Hong Gao, Jizhou Luo, and Hongzhi Wang. A gray-box performance model for apache spark. *Future Gener. Comput. Syst.*, 89(C):58â67, dec 2018.

[8] Yuxing Chen, Peter Goetsch, Mohammad A. Hoque, Jiaheng Lu, and Sasu Tarkoma. d-simplexed: Adaptive delaunay triangulation for performance modeling and prediction on big data analytics. *IEEE Transactions on Big Data*, 8(2):458–469, 2022.

[9] Guoli Cheng, Shi Ying, Bingming Wang, and Yuhang Li. Efficient performance prediction for apache spark. *Journal of Parallel and Distributed Computing*, 149:40–51, 2021.

[10] Sabyasachi Dash, Sushil Kumar Shakyawar, Mohit Sharma, and Sandeep Kaushik. Big data in healthcare: management, analysis and future prospects. *Journal of Big Data*, 6(1):1–25, 2019.

[11] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107â113, jan 2008.

[12] Wanling Gao, Jianfeng Zhan, Lei Wang, Chunjie Luo, Daoyi Zheng, Xu Wen, Rui Ren, Chen Zheng, Xiwen He, Hainan Ye, Haoning Tang, Zheng Cao, Shujie Zhang, and Jiahui Dai. Bigdatabench: A scalable and unified big data and ai benchmark suite, 2018.

[13] Hossein Hassani, Xu Huang, and Emmanuel Silva. Digitalisation and big data mining in banking. *Big Data and Cognitive Computing*, 2(3), 2018.

[14] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Mesos: A platform for Fine-Grained resource sharing in the data center. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, Boston, MA, March 2011. USENIX Association.

[15] Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang. The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*, pages 41–51, 2010.

[16] Muhammed Tawfiqul Islam, Shanika Karunasekera, and Rajkumar Buyya. dspark: Deadline-based resource allocation for big data applications in apache spark. In *2017 IEEE 13th International Conference on e-Science (e-Science)*, pages 89–98, 2017.

[17] Soroush Karimian-Aliabadi, Mohammad-Mohsen Aseman-Manzar, Reza Entezari-Maleki, Danilo Ardagna, Bernhard Egger, and Ali Movaghar. Fixed-point iteration approach to spark scalable performance modeling and evaluation. *IEEE Transactions on Cloud Computing*, pages 1–1, 2021.

[18] A. R. Lebeck, M. Thottethodi, and S. S. Mukherjee. Exploiting global knowledge to achieve self-tuned congestion control for k-ary n-cube networks. *IEEE Transactions on Parallel Distributed Systems*, 1(03):257–272, mar 2004.

[19] Vicent Sanz Marco, Ben Taylor, Barry Porter, and Zheng Wang. Improving spark application throughput via memory aware task co-location: A mixture of experts approach. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*, Middleware '17, page 95â108, New York, NY, USA, 2017. Association for Computing Machinery.

[20] Alexandre Maros, Fabricio Murai, Ana Paula Couto da Silva, Jussara M. Almeida, Marco Lattuada, Eugenio Gianniti, Marjan Hosseini, and Danilo Ardagna. Machine learning for performance prediction of spark cloud applications. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 99–106, 2019.

[21] R. Nelson and A.N. Tantawi. Approximate analysis of fork/join synchronization in parallel queues. *IEEE Transactions on Computers*, 37(6):739–743, 1988.

[22] Nhan Nguyen, Mohammad Maifi Hasan Khan, and Kewen Wang. Towards automatic tuning of apache spark configuration. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 417–425, 2018.

[23] David Buchaca Prats, Felipe Albuquerque Portella, Carlos H. A. Costa, and Josep Lluis Berral. You only run once: Spark auto-tuning from a single run. *IEEE Transactions on Network and Service Management*, 17(4):2039–2051, 2020.

[24] Juwei Shi, Yunjie Qiu, Umar Farooq Minhas, Limei Jiao, Chen Wang, Berthold Reinwald, and Fatma Özcan. Clash of the titans: Mapreduce vs. spark for large scale data analytics. *Proc. VLDB Endow.*, 8(13):2110â2121, sep 2015.

[25] Abhishek Singh, Ashish Payal, and Sourabh Bharti. A walkthrough of the emerging iot paradigm: Visualizing inside functionalities, key features, and open issues. *Journal of Network and Computer Applications*, 143:111–151, 2019.

[26] Shivaram Venkataraman, Zongheng Yang, Michael Franklin, Benjamin Recht, and Ion Stoica. Ernest: Efficient performance prediction for large-scale advanced analytics. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, NSDI'16, page 363â378, USA, 2016. USENIX Association.

[27] Guolu Wang, Jungang Xu, and Ben He. A novel method for tuning configuration parameters of spark based on machine learning. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 586–593, 2016.

[28] Kewen Wang and Mohammad Maifi Hasan Khan. Performance prediction for apache spark platform. In *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, pages 166–173, 2015.

[29] Jinhan Xin, Kai Hwang, and Zhibino Yu. Locat: Low-overhead online configuration auto-tuning of spark sql applications. In *Proceedings of the 2022 ACM SIGMOD International Conference on Management of Data*, SIGMOD '22, page 457â468, New York, NY, USA, 2022. Association for Computing Machinery.

[30] Zhibin Yu, Zhendong Bei, and Xuehai Qian. Datasize-aware high dimensional configurations auto-tuning of in-memory cluster computing. *SIGPLAN Not.*, 53(2):564â577, mar 2018.

[31] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A Fault-Tolerant abstraction for In-Memory cluster computing. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 15–28, San Jose, CA, April 2012. USENIX Association.