Automatic Performance Tuning for Distributed Data Stream Processing Systems

Herodotou, Herodotos

IEEE
2022-08

Herodotou , H , Odysseos , L , Chen , Y & Lu , J 2022 , Automatic Performance Tuning for Distributed Data Stream Processing Systems . in 38TH IEEE International Conference on Data Engineering . Data engineering , IEEE , IEEE International Conference on Data Engineering , Kuala Lumpur , Malaysia , 09/05/2022 . https://doi.org/10.1109/ICDE53745.2022.00296

http://hdl.handle.net/10138/354267
https://doi.org/10.1109/ICDE53745.2022.00296

unspecified
acceptedVersion

Automatic Performance Tuning for Distributed Data Stream Processing Systems

Herodotos Herodotou
Cyprus University of Technology
herodotos.herodotou@cut.ac.cy

Lambros Odysseos
Cyprus University of Technology
lambros.odysseos@cut.ac.cy

Yuxing Chen
Tencent Inc.
axingguchen@tencent.com

Jiaheng Lu
University of Helsinki
jiaheng.lu@helsinki.fi

***Abstract*—Distributed data stream processing systems (DSPSs) such as Storm, Flink, and Spark Streaming are now routinely used to process continuous data streams in (near) real-time. However, achieving the low latency and high throughput demanded by today's streaming applications can be a daunting task, especially since the performance of DSPSs highly depends on a large number of system parameters that control load balancing, degree of parallelism, buffer sizes, and various other aspects of system execution. This tutorial offers a comprehensive review of the state-of-the-art automatic performance tuning approaches that have been proposed in recent years. The approaches are organized into five main categories based on their methodologies and features: cost modeling, simulation-based, experiment-driven, machine learning, and adaptive tuning. The categories of approaches will be analyzed in depth and compared to each other, exposing their various strengths and weaknesses. Finally, we will identify several open research problems and challenges related to automatic performance tuning for DSPSs.**

***Index Terms*—Performance tuning, data stream processing, parameter tuning, Storm, Flink, Spark Streaming**

I. Introduction

The need for real-time analytical processing has spurred the development and evolution of data stream processing systems (DSPSs) such as Apache Storm, Flink, and Spark Streaming, which execute on distributed clusters of commodity hardware. Streaming applications on DSPSs process each data record (or each small batch of records) of a data stream as it arrives on a continuous basis, while the DSPSs transparently handle their data-parallel execution on the cluster [1]. Examples of latency-sensitive applications include identifying potential fraud in real-time, recommending personalized content, recognizing trading signals in financial markets, etc.

In many of these applications, the continuous data streams must be processed (almost) instantly as the extracted value depreciates quickly over time [2]. At the same time, the ever-increasing volume of streaming data requires systems to sustain high levels of throughput. Hence, the ability to handle and process continuous streams of data in a scalable and timely fashion is becoming crucial for the success of today's data-driven organizations [3]. However, the performance of DSPSs highly depends on a large number of system parameters that control various aspects of system execution, including load balancing, degree of parallelism, memory settings, and buffer sizes, among others [4]. Inappropriate settings of these parameters have been shown to impact application performance negatively, and hence must be tuned carefully [5]–[7].

Automatically tuning system parameters for optimizing the performance of streaming applications has several key challenges. First, the parameter search space is large and complex as there are hundreds of parameters affecting performance that can be arranged in thousands of different combinations [5]. To make matters worse, many parameters have complicated inter-dependencies among each other [8]. Improper configurations with some parameter values set too low could waste cluster resources and constrain application performance. On the other hand, if some parameter values are set too high, they could lead to resource contention and cause issues such as thrashing and page faults [9]. In addition, the input data is a real-time data stream that can experience dynamic changes in workload properties, introducing difficulties in observing and modeling workload performance [10]. Finally, good configurations depend on both the type of streaming application and the hardware characteristics. Hence, a configuration that is optimal for a particular application and workload might perform poorly in a different environment [5].

A substantial amount of research has been introduced in the last few years for addressing the problem of automatic performance tuning in distributed data stream processing systems, presented in a recent survey paper by the authors [4]. This tutorial will offer a comprehensive review of state-of-the-art performance tuning approaches, which tackle one or more of the aforementioned challenges in an attempt to achieve the dual goal of low latency and high throughput in a streaming environment. While these approaches target the same problem, their methodologies, implementations, and target DSPSs vary significantly. Hence, we introduce a taxonomy of automatic approaches into five main categories: cost modeling, simulation-based, experiment-driven, machine learning, and adaptive tuning. These approaches will be analyzed in depth and compared within the context of popular DSPSs such as Storm, Flink, and Spark Streaming.

Tutorial overview. We will first provide an overview and motivating examples of performance tuning on distributed DSPSs. We differentiate between 2 main types of DSPSs, offering per-record or micro-batching stream processing. Next, we will introduce our taxonomy of existing tuning approaches, discuss each category's key characteristics, and analyze their respective advantages and disadvantages. Finally, we will identify and discuss several open research problems and challenges.

Related tutorials. One past tutorial [11] presented high-level optimizations for streaming applications, such as operator separation, fusion, fission, and placement, while another one [12] discussed fault tolerance and elastic scaling mechanisms for stream processing in a Cloud computing environment. A recent tutorial (SIGMOD 2020) [1] reviewed the evolution of streaming system aspects over the last two decades and described emerging streaming applications. However, none of the above tutorials discussed any performance tuning approaches related to parameter configurations. In terms of automatic parameter tuning, our recent tutorial (VLDB 2019) [13] surveyed the area for database systems, Hadoop MapReduce, and Spark, but did not cover data stream processing systems.

Contributions. To the best of our knowledge, this is the first tutorial to discuss the state-of-the-art research and industrial works on automatic performance tuning for popular data stream processing systems. The tuning approaches covered in this tutorial will help application developers to better understand performance trade-offs, system administrators to select appropriate tuning strategies, and researchers to develop novel automatic tuning techniques.

II. Background

A. Data Stream Processing Systems

Distributed DSPSs take an arbitrary and potentially unbounded stream of data as input and perform the processing that is needed in real (or near-real) time [2]. The processing typically takes the form of a directed acyclic graph (DAG) of operators, which are broken into parallel tasks and distributed across the cluster for execution. DSPSs are categorized into two types of processing models, namely, *record-based* (or continuous operator) streaming and *micro-batching* (or batched) streaming [10]. In the record-based model, shown in Figure 1(A), a DSPS processes each record individually as soon as it arrives in the system. This model enables real-time processing, providing lower latencies but sacrifices throughput. Batched streaming works by dividing the input data stream into mini-batches and processes each batch one at a time, as visualized in Figure 1(B). This processing model benefits from higher throughput since multiple records are processed together, but typically results in higher average latency. Apache Storm, Heron, and Flink support record-based streaming, while Spark Streaming and Storm Trident support micro-batching.

When applying performance tuning on DSPSs, it is crucial to understand and discriminate these two categories since a distinct set of configurations on a system that belongs to the first category will not provide the same performance throughput in the second, and vice versa. This concept must also be taken into consideration depending on the DSPS that will be used and on the application that will be developed because each processing model has different characteristics.

B. Performance Tuning Problem Statement

A streaming application A executing on a DSPS is of the form $A = \langle g, d, r, p \rangle$, where g denotes the DAG of operators executing as part of the application, d the input

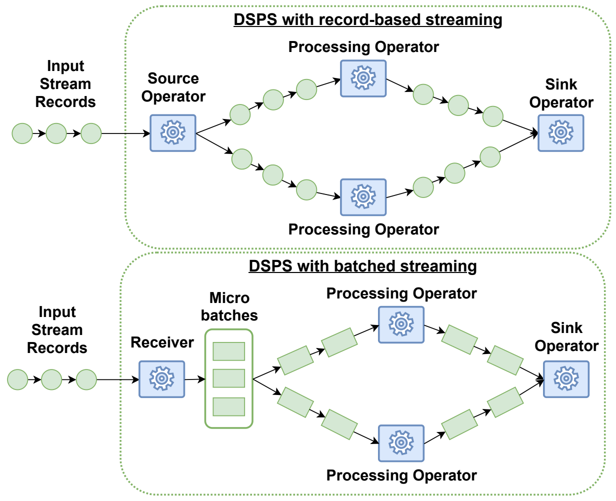

Fig. 1. Data stream processing with (A) record-based and (B) batched streaming model.

stream data properties, r the cluster resources, and p the set of parameter settings used [4]. The *parameter search space* $\mathbb{S}$ is the Cartesian product of the domains of all parameters. The performance of A (typically referring to latency or throughput) can be modeled and calculated using a function $F(g, d, r, p)$.

The definition of the *performance tuning problem* is as follows: Given a DAG of streaming operators g to process an input data stream d over cluster resources r, find the optimal parameter settings p^* that maximizes F over the parameter search space $\mathbb{S}$. The performance function F is hard to model mathematically but can be approximated to some degree using past measurements. Finally, finding an optimal solution to this problem is NP-hard [4].

III. Performance Tuning Categories

The performance implications of tuning are well-known in the industry, with good configurations leading to significant performance benefits, while bad ones can cause severe performance degradation. A substantial amount of research has been introduced in the last few years for automating performance tuning in DSPSs using a variety of approaches, classified into five categories: cost modeling, simulation-based, experiment-driven, machine learning, and adaptive tuning. Table I outlines and compares the various key features and functionalities of approaches falling in the five categories in terms of modeling, need for statistics or runs, prediction accuracy, and adaptability to workload and system changes [4]. These approach categories will be examined in depth and compared within the context of DSPSs during this tutorial.

Cost modeling approaches such as [14]–[16] build efficient performance prediction models that use mathematical cost functions and data statistics. These approaches are computationally very efficient and can produce predictions with reasonable accuracy. However, it is hard to represent complex system internals using mathematical models because (i) streaming systems are distributed and with numerous moving parts and

TABLE I
FEATURE COMPARISON AMONG THE FIVE PARAMETER TUNING CATEGORIES OF APPROACHES [4]

Feature	Cost modeling	Simulation	Experiment-driven	Machine learning	Adaptive
Key modeling technique	cost functions	simulation	search algorithms	ML models	mixed
Number of parameters modeled	some	some	many	many	some
System understanding	strong	strong	light	no	strong
Need for history logs	light	light	strong	strong	light
Need for data input stats	light	light	no	strong	light
Real tests to run	some	no	yes	yes	yes
Time to build model	efficient	medium	slow	slow	medium
Prediction accuracy	medium	medium	medium	high	medium
Adapt to workload	light	light	no	no	adaptive
Adapt to system changes	no	light	no	adaptive	light

pluggable components and (ii) the evolving data statistics for the input streams impact system execution. Finally, these approaches are typically specific to a particular version of a streaming platform, and are hard to be adapted after platform changes are made.

Simulation-based approaches such as [17]–[19] use partial or complete system simulation to build performance prediction models. Users can then simulate executions with different parameter settings to find the ones that optimize performance in a safe simulation environment. Thus, users can make cost-effective decisions, even when they have limited information on data, workload, or resource characteristics. Simulators offer various degrees of support with regards to network traffic, hardware properties, and resource scheduling, while most simulators only support some basic parameter configurations. Finally, creating an accurate simulator often requires a deep and comprehensive understanding of the internal system dynamics, data, and workloads.

Experiment-driven approaches such as [6], [20], [21] execute a streaming application multiple times with different parameter settings each time, until the (near) optimal settings are found. The sequence of executions is guided by a search algorithm and the feedback provided after each execution. The insights acquired from the actual runs enable a deeper understanding of how data, applications, or resources impact performance. Overall, most of the methods focus on search-based algorithms, which enable global and local search for finding (near) optimal parameter settings. Even though experiment-driven approaches often produce better settings, they often require much longer tuning times due to the repeated runs.

Machine learning approaches such as [22]–[24] train performance prediction models on historical logs and/or execution metrics using machine or deep learning methods. The models are then used for predicting application performance when using various parameter settings. These approaches consider the entire system as a black box and assume no knowledge of system internals. ML models can often achieve high prediction accuracy and find near-optimal parameter settings, especially when trained with large training data. However, acquiring training data can be time-consuming since it may require multiple runs with different settings to avoid under-fitting. The situation becomes worse as data properties can change dynamically or unseen applications appear.

Adaptive approaches such as [7], [25], [26] change system parameters adaptively while an application is running, based on online performance metrics computed on-the-fly, in order to dynamically improve performance. Most adaptive approaches employ some performance models for making predictions, along with a search or scheduling algorithm for making tuning decisions. Such approaches typically work well for ad-hoc streaming applications. However, changing parameter settings dynamically in an online system may sometimes cause various performance or stability issues (e.g., introduce stragglers).

IV. OPEN PROBLEMS AND CHALLENGES

In the final part of the tutorial, we will focus on open problems and challenges that remain in the field of automatic performance tuning. We identify three general areas of challenges that are related to *cluster heterogeneity*, *cloud computing*, and *edge computing*. In each of these areas, we will briefly overview partial/preliminary solutions and discuss the various challenges involved.

As organizations often own multiple generations of hardware, heterogeneous clusters are becoming common in practice, with nodes having different types of CPUs or memory sizes, which make tuning even more challenging. The adoption of modern hardware such as NVRAM, GPUs, and FPGAs also calls for investigation, including its impact on the performance of streaming applications.

The proliferation of the Cloud led to new cloud-based data streaming engines such as Amazon Kinesis and Confluent Cloud, which have some unique requirements as well as new challenges, including how to (i) manage performance interactions among multiple tenants, (ii) ensure high scalability and elasticity, and (iii) navigate the tradeoffs between high performance and fault tolerance.

A recent trend in stream-based computing, especially in the Internet-of-Things (IoT) domain, involves decentralized processing at the source of the data (i.e., at the edge), to alleviate the pressure of computation at the cluster or Cloud. This process, however, creates new problems regarding interoperability, managing devices with limited capabilities, application reconfiguration at the edge, etc.

V. Tutorial Organization

The tutorial is planned for 1.5 hours (90 minutes) and will have the following structure:

Introduction and motivation (10'). We introduce the problem of performance tuning in large-scale data stream processing systems and motivate the need for automatic tuning approaches with several applications/scenarios.

Method taxonomy (10'). We present the taxonomy of automatic performance tuning approaches in DSPSs, including cost modeling, simulation-based, experiment-driven, machine learning, and adaptive tuning.

Parameter tuning approaches (50'). We introduce representative approaches from each category for tuning the performance of streaming applications.

Comparison of approaches (10'). We compare the solutions in the various tuning categories and across the two streaming models, i.e., record-based and batched streaming.

Open problems and challenges (10'). We discuss open problems and challenges for performance tuning.

VI. Goals of the Tutorial

Intended Audience. This tutorial is intended for a wide scope of audience ranging from academic researchers and students to industrial developers and practitioners that want to understand the impact of performance tuning on streaming applications executing on large-scale DSPSs. Basic knowledge in the execution and configuration of streaming applications in DSPSs is sufficient to follow the tutorial. Some background in distributed computing, performance tuning, and basic machine learning techniques would be useful but not necessary.

Learning Outcomes. The main learning outcomes of this tutorial are: (1) understanding the impact of tuning on the performance of DSPSs and streaming applications; (2) learning the taxonomy of performance tuning approaches used for current streaming applications; (3) comparison of features, functionalities, advantages, and disadvantages of tuning approaches; (4) identification of open problems and research challenges of automatic performance tuning. Practitioners and students will be able to quickly build an extensive understanding as well as grasp the latest trends and state-of-the-art techniques in automatic performance tuning. In addition, this tutorial can guide both researchers and developers in contributing their expertise and advancing this important and challenging field.

VII. Short Biographies

Herodotos Herodotou is an Assistant Professor at the Cyprus University of Technology. His research work focuses on automatic performance tuning of both centralized and distributed data-intensive computing systems. His Ph.D. dissertation on MapReduce performance tuning received the ACM SIGMOD Jim Gray Doctoral Dissertation Award Honorable Mention.

Lambros Odysseos is a PhD student at the Cyprus University of Technology. He has been working as a research associate for the past few years and his research interests include stream processing, data analytics and visualizations, smart data processing, Internet of Things (IoT), and machine learning.

Yuxing Chen is a senior engineer at the Tencent Inc. His research topics are parameter tuning on big data systems and transaction processing.

Jiaheng Lu is a Professor at the University of Helsinki, Finland. He has written four books on Hadoop and NoSQL databases, and more than 100 journal and conference papers published in SIGMOD, VLDB, TODS, TKDE, etc.

References

[1] P. Carbone, M. Fragkoulis *et al.*, "Beyond Analytics: The Evolution of Stream Processing Systems," in *SIGMOD*. ACM, 2020, pp. 2651–2658.

[2] H. Isah, T. Abughofa, S. Mahfuz *et al.*, "A Survey of Distributed Data Stream Processing Frameworks," *IEEE Access*, vol. 7, 2019.

[3] H. Isah and F. Zulkernine, "A Scalable and Robust Framework for Data Stream Ingestion," in *Big Data*. IEEE, 2018, pp. 2900–2905.

[4] H. Herodotou, Y. Chen, and J. Lu, "A Survey on Automatic Parameter Tuning for Big Data Processing Systems," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–37, 2020.

[5] A. Dethise, "A Reinforcement Learning Approach to Optimize Performance of Stream Processors," in *EuroDW*. EuroSys, 2018.

[6] X. Liu *et al.*, "A Stepwise Auto-Profiling Method for Performance Optimization of Streaming Applications," *ACM TAAS*, vol. 12, 2018.

[7] M. Petrov, N. Butakov, D. Nasonov, and M. Melnik, "Adaptive Performance Model for Dynamic Scaling Apache Spark Streaming," *Procedia Computer Science*, vol. 136, pp. 109–117, 2018.

[8] M. Trotter, T. Wood, and J. Hwang, "Forecasting a Storm: Divining Optimal Configurations using Genetic Algorithms and Supervised Learning," in *ICAC*. IEEE, 2019, pp. 136–146.

[9] S. Kulkarni, N. Bhagat, M. Fu, V. Kedigehalli *et al.*, "Twitter Heron: Stream Processing at Scale," in *SIGMOD*. ACM, 2015, pp. 239–250.

[10] M. Dayarathna and S. Perera, "Recent Advancements in Event Processing," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, p. 33, 2018.

[11] S. Schneider, M. Hirzel, and B. Gedik, "Tutorial: Stream Processing Optimizations," in *DEBS*. ACM, 2013, pp. 249–258.

[12] T. Heinze, L. Aniello, L. Querzoni, and Z. Jerzak, "Cloud-based Data Stream Processing," in *DEBS*. ACM, 2014.

[13] J. Lu, Y. Chen, H. Herodotou, and S. Babu, "Speedup Your Analytics: Automatic Parameter Tuning for Databases and Big Data Systems," *PVLDB*, vol. 12, no. 12, pp. 1970–1973, 2019.

[14] I. Bedini *et al.*, "Modeling Performance of a Parallel Streaming Engine: Bridging Theory and Costs," in *ICPE*. ACM/SPEC, 2013, pp. 173–184.

[15] M. Bansal *et al.*, "Trevor: Automatic Configuration and Scaling of Stream Processing Pipelines," *CoRR*, vol. abs/1812.09442, 2018.

[16] F. Kalim, T. Cooper *et al.*, "Caladrius: A Performance Modelling Service for Distributed Stream Processing Systems," in *IEEE ICDE*, 2019.

[17] J. I. Requeno *et al.*, "Performance Analysis of Apache Storm Applications Using Stochastic Petri Nets," in *IRI*. IEEE, 2017, pp. 411–418.

[18] J. Kroß and H. Krcmar, "Model-based Performance Evaluation of Batch and Stream Applications for Big Data," in *MASCOTS*. IEEE, 2017.

[19] J. Lin, M. Lee, I. C. Yu, and E. B. Johnsen, "Modeling and Simulation of Spark Streaming," in *AINA*. IEEE, 2018, pp. 407–413.

[20] P. Jamshidi and G. Casale, "An Uncertainty-aware Approach to Optimal Configuration of Stream Processing Systems," in *MASCOTS*. IEEE, 2016, pp. 39–48.

[21] M. Bilal and M. Canini, "Towards Automatic Parameter Tuning of Stream Processing Systems," in *SoCC*. ACM, 2017.

[22] T. Li *et al.*, "Performance Modeling and Predictive Scheduling for Distributed Stream Data Processing," *IEEE TBD*, vol. 2, no. 4, 2016.

[23] C. Wang *et al.*, "Automating Characterization Deployment in Distributed Data Stream Management Systems," *IEEE TKDE*, vol. 29, no. 12, 2017.

[24] L. M. Vaquero and F. Cuadrado, "Auto-tuning Distributed Stream Processing Systems using Reinforcement Learning," *CoRR*, vol. abs/1809.05495, 2018.

[25] T. Z. J. Fu, J. Ding, R. T. B. Ma *et al.*, "DRS: Dynamic Resource Scheduling for Real-Time Analytics over Fast Streams," in *ICDCS*. IEEE, 2015, pp. 411–420.

[26] S. Venkataraman *et al.*, "Drizzle: Fast and Adaptable Stream Processing at Scale," in *SOSP*. ACM, 2017, pp. 374–389.