RADemics

Big Data Handling and Distributed AI Workloads Using PySpark and Dask Frameworks

Vineeth V V, B. Persis Urbana Ivy, S. Senthil Kumar
UNIVERSITY OF TECHNOLOGY AND APPLIED SCIENCES, MOTHER TERESA INSTITUTE OF ENGG AND TECHNOLOGY, KARPAGAM ACADEMY OF HIGHER EDUCATION

Big Data Handling and Distributed AI Workloads Using PySpark and Dask Frameworks

[1]Vineeth V V, Lecturer, Electrical and Electronics Engineering, UNIVERSITY OF TECHNOLOGY AND APPLIED SCIENCES, IBRA. PO Box 327 Ibra 400, Al Sharqiya North, Sultanate of Oman, Mail id: vineeth32@gmail.com.

[2]B. Persis Urbana Ivy, Dean (CSE & Allied branches), Mother Teresa Institute of Engg and Technology, Melumoi (Post), Palamaner - 517408. Mail id: Mail id: urbana23@gmail.com.

[3]S. Senthil Kumar, Assistant Professor, Artificial Intelligence and Data Science, Karpagam Academy of Higher Education, Deemed to be University, Coimbatore. Mail id: skumsarsk@gmail.com.

Abstract

The rapid escalation of data volume, variety, and velocity has necessitated the adoption of scalable and distributed computing frameworks to support artificial intelligence (AI) workloads across diverse industrial and research domains. This chapter presents a comprehensive comparative study of PySpark and Dask, two leading open-source frameworks designed to handle big data and parallelized AI tasks in distributed environments. The focus lies on the architectural foundations, performance optimization techniques, and the ability of each framework to manage data shuffling, resource utilization, and fault tolerance in large-scale AI pipelines. Through the exploration of deployment strategies—ranging from standalone systems to cloud-native clusters—this work analyzes execution efficiency, scalability under heavy computational loads, and integration with popular deep learning libraries such as TensorFlow and PyTorch. The chapter further highlights practical considerations for benchmarking, diagnostics, and system monitoring, providing a technical foundation for selecting the appropriate framework based on workload characteristics and infrastructure constraints. By addressing existing research gaps in interoperability, memory efficiency, and real-world deployment practices, this contribution offers critical insights for data scientists, system architects, and AI practitioners working in distributed and data-intensive computational settings.

Keywords: Distributed Computing, PySpark, Dask, Big Data, Artificial Intelligence, Deep Learning Integration

Introduction

The proliferation of digital technologies and the accelerated expansion of data-generating sources have led to the emergence of big data as a central focus in computational science and industrial innovation [1]. From sensor networks and social media streams to biomedical imaging and financial transactions, data is being produced at scales that far exceed the capabilities of traditional computing systems [2]. The defining characteristics of big data—volume, velocity,

variety, veracity, and value—necessitate the adoption of distributed and parallelized processing architectures [3]. These architectures must not only support efficient data ingestion and storage but also enable scalable analytics, real-time querying, and AI-driven decision-making [4]. The convergence of big data with artificial intelligence (AI) has further heightened the demand for frameworks capable of executing complex models across vast, heterogeneous datasets. In this context, distributed computing tools such as PySpark and Dask have emerged as critical enablers for managing and operationalizing AI workloads in modern data ecosystems [5].

PySpark, the Python API for Apache Spark, has gained wide adoption due to its robust distributed data processing capabilities, high-level abstractions, and support for in-memory computations [6]. Its ability to integrate with big data storage platforms, batch and stream processing engines, and machine learning libraries positions it as a comprehensive framework for large-scale data analytics [7]. On the other hand, Dask is a Python-native parallel computing library that provides dynamic task scheduling and distributed execution while maintaining compatibility with familiar libraries such as Pandas, NumPy, and Scikit-learn [8]. Designed for flexibility and modularity, Dask allows users to scale their computations from single machines to multi-node clusters with minimal code modification. While both PySpark and Dask serve overlapping domains, they differ significantly in terms of architecture, performance, ease of use, and ecosystem integration [9]. An in-depth comparative study is essential to determine their suitability for various AI applications involving large and complex datasets [10].

The growing integration of AI and machine learning into big data workflows introduces new challenges related to data movement, task orchestration, computational overhead, and model deployment [11]. AI workloads, particularly those involving deep neural networks or ensemble methods, often require multiple iterations, intermediate transformations, and large memory footprints [12]. These factors place increased pressure on the underlying infrastructure to efficiently manage computation, storage, and communication across distributed nodes [13]. PySpark addresses these requirements through its DAG-based execution model, resilient distributed datasets (RDDs), and support for fault-tolerant operations. It leverages JVM-based execution and integrates seamlessly with existing Hadoop ecosystems, making it suitable for enterprise environments with pre-established data pipelines [14]. In contrast, Dask adopts a dynamic, Python-centric approach with fine-grained control over task scheduling and execution. Its lazy evaluation model enables optimized computation graphs that can be tuned for specific hardware configurations, including CPU-bound and GPU-accelerated environments [15].