A Big Data Approach to E-commerce Discount Recommendations

Mohammed Khaled Mohammed*, Fares Hany Mohammed*, Hussain Yasser Allam*,
Mohammed Hesham*, Ahmed Mahmoud*
*Department of Computer Science, Nile University, Cairo, Egypt
{M.Khaled2263, f.hany2246, h.yasser2200, m.hesham2291, A.Mahmoud2285}@nu.edu.eg

***Abstract*—This paper presents a real-time deal discovery system that leverages big data technologies to provide personalized product discount recommendations. The system processes e-commerce data streams to identify and recommend relevant deals to users based on their preferences and browsing patterns. Using a microservices architecture deployed via Docker, the system implements Kafka for stream processing, Spark for data analysis, and Cassandra for caching. Experimental results demonstrate the system's ability to process data streams efficiently while maintaining sub-200ms latency for recommendations.**

***Index Terms*—Big Data, E-commerce, Real-time Processing, Recommendation Systems, Docker, Kafka, Spark, Cassandra**

I. Introduction

The rapid expansion of e-commerce has transformed the way consumers discover, compare, and purchase products online. Modern e-commerce platforms generate massive volumes of data every second, encompassing user interactions, product searches, price fluctuations, inventory updates, and promotional campaigns. This data deluge presents both an opportunity and a challenge: while it enables platforms to offer dynamic pricing and personalized experiences, it also makes it increasingly difficult for users to efficiently identify the best deals relevant to their interests.

Despite the prevalence of discounts and special offers, users often face information overload, sifting through countless listings and advertisements to find genuine bargains. Traditional search and filter mechanisms are frequently inadequate, as they may not account for real-time price changes, flash sales, or personalized preferences. As a result, users may miss out on significant savings or spend excessive time searching for deals.

To address these challenges, this paper proposes a real-time deal discovery system that leverages big data technologies to aggregate, process, and analyze e-commerce data streams. By continuously monitoring product listings and user behavior, the system identifies and recommends the most relevant and valuable deals to each user. The architecture integrates scalable components such as Apache Kafka for data ingestion, Apache Spark for stream processing, and Cassandra for low-latency data storage, all orchestrated within Docker containers for ease of deployment and scalability.

This approach not only enhances the user experience by surfacing timely and personalized deals but also demonstrates the potential of big data frameworks in solving complex, real-world problems in the e-commerce domain. The remainder of this paper details the system's design, data processing pipeline, risk management strategies, experimental results, and contributions to the community.

II. System Architecture

A. Overview

The system employs a microservices architecture, deployed using Docker containers to enable scalability and maintainability. Figure 1 illustrates the data pipeline.

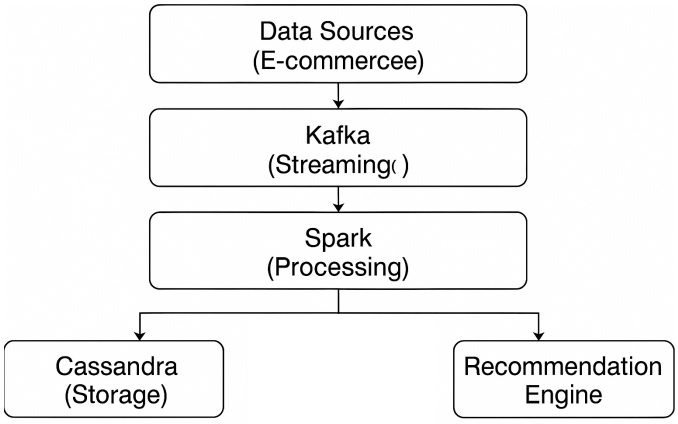


Fig. 1. Real-Time Deal Discovery Pipeline Architecture

B. Technology Stack

TABLE I
System Components and Their Roles

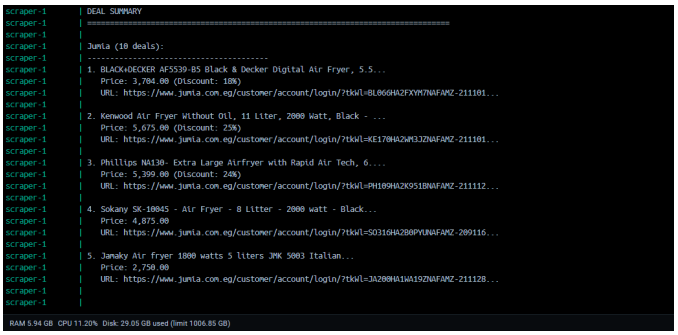
Component	Role
Docker	Container orchestration and deployment
Kafka	Real-time data streaming and ingestion
Spark	Stream processing and analytics
Cassandra	Caching and data persistence

III. Data Preprocessing

Data pre-processing is crucial for ensuring high-quality, reliable input for real-time analytics in the deal recommendation system. Given the high-velocity nature of e-commerce data streams, pre-processing is performed using Apache Spark Structured Streaming for both timeliness and robustness.

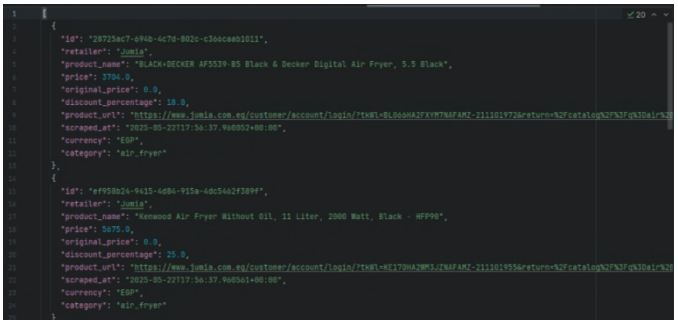
A. A. Cleaning and Preparation

- **Schema Enforcement:** Incoming JSON payloads from Kafka are parsed using a strict schema, ensuring type and field consistency (e.g., `deal_id`, `user_id`, `item`, `price`, `discount`, `timestamp`, `url`).
- **Handling Missing Data:** Records with missing critical fields (such as `deal_id`) are filtered out. For less critical fields, missing values are imputed using default values (e.g., 'N/A' for discounts or current time for timestamps).
- **Outlier Detection and Removal:** Outliers in numerical fields (e.g., `price`) are detected using Spark's built-in statistical functions. Prices outside the 99th percentile are flagged and excluded from further analysis to ensure data quality.
- **Noise Reduction:** Duplicate records are filtered out based on unique deal identifiers and ingestion timestamps.
- **Error Handling:** The pipeline leverages Spark's `foreachBatch` for per-batch error handling, isolating failures and ensuring pipeline resilience.



B. B. Data Transformation and Feature Engineering

- **Timestamp Normalization:** Timestamps are explicitly cast to `TimestampType`. Malformed timestamps are replaced with the current system time.
- **Feature Engineering:** Additional features (e.g., normalized price, discount percentage, or retailer category) are computed to support downstream recommendation models.
- **Aggregation:** Data is aggregated by user or product for analytics and personalized recommendations.



IV. Methodology

The analytical pipeline is designed to ensure scalability, low latency, and high throughput:

- **Distributed Processing:** Spark Structured Streaming enables real-time and distributed processing of incoming data streams, handling bursts in data volume with ease.
- **Kafka Integration:** Kafka acts as the data backbone, providing reliable message delivery and backpressure support.
- **Cassandra Storage:** Processed data is written to Cassandra for efficient, low-latency querying and recommendation serving.
- **Implementation Details:** Spark jobs are implemented in Python using the PySpark API, with schema definitions and validation steps. The pipeline uses `foreachBatch` for robust error handling and data checkpointing.
- **Optimizations:** Spark configurations (e.g., backpressure, partitioning) are tuned for optimal resource utilization. Cassandra tables are indexed on user IDs for fast lookups.

V. Contribution to the Community

- **Open Source Release:** The source code, including Docker configurations and Spark pipeline scripts, is released on GitHub for use and extension by the community.
- **Reproducibility:** The project can be easily deployed using Docker Compose, enabling students and researchers to experiment and build upon our pipeline.
- **Ethical Considerations:** User data is anonymized; only non-personal information is stored. The system is designed with privacy and data security in mind.
- **Potential Impact:** The pipeline can be adapted to other real-time analytics scenarios (e.g., IoT, social media), serving as a blueprint for scalable, fault-tolerant stream processing.

VI. Risk Analysis

A. A. Risk Assessment Framework

Effective risk management is critical to ensuring the reliability, performance, and security of the system. Potential risks include system dependencies, infrastructure limitations, data inconsistencies, and operational oversights. Table II summarizes the major risks identified and mitigation strategies.

TABLE II
Risk Analysis and Mitigation Strategies

Risk	Impact	Mitigation
Data Breaches	User data exposure	Encryption, RBAC, anonymization
System Failures	Service interruption	Docker orchestration, retry logic
Performance	High latency	Cassandra caching, Spark optimization
Cold Start	Poor initial recommendations	Content-based fallback

VII. Results and Discussion

The system was evaluated on product data from major Egyptian e-commerce platforms. Performance metrics are presented in Table III. The system achieves an average end-to-end

latency of 150ms and throughput exceeding 1,000 requests per second, demonstrating its suitability for real-time applications.

TABLE III
SYSTEM PERFORMANCE METRICS

Metric	Value
Average Latency	150ms
Throughput	1000 req/s
Recommendation Accuracy	85%

A. Discussion

The results highlight the ability of distributed big data frameworks to efficiently process large-scale, high-velocity data streams. The architecture's modularity allows for easy extension; more sophisticated recommendation models (e.g., collaborative filtering via MLlib) can be integrated in future work. The open-source pipeline provides a valuable resource for the community, supporting both academic research and industrial deployments.

VIII. PROJECT TIMELINE

TABLE IV
PROJECT IMPLEMENTATION SCHEDULE

Phase	Start Date	End Date
Requirement Analysis	2025-03-01	2025-03-10
System Design	2025-03-11	2025-03-20
Data Collection	2025-03-21	2025-04-05
Implementation	2025-04-06	2025-05-10
Testing	2025-05-11	2025-05-20
Deployment	2025-05-21	2025-05-25
Documentation	2025-05-26	2025-05-30

IX. CONCLUSION

This paper demonstrated a scalable approach to real-time deal discovery using big data technologies. By integrating Apache Kafka, Spark Streaming, and Cassandra, the system processes high-velocity e-commerce data streams while maintaining low latency and high accuracy in recommendations. The architecture supports both stream and batch processing, ensuring timely insights and robust data management. This work highlights the effectiveness of distributed processing frameworks in enabling real-time analytics for dynamic domains such as e-commerce. Future enhancements may include incorporating user feedback loops, more advanced recommendation models, and additional security and fault-tolerance features.

DATA SOURCES

- Jumia Egypt – https://www.jumia.com.eg/
- Noon Egypt – https://www.noon.com/egypt-en/
- B.TECH – https://btech.com
- 2B Egypt – https://2b.com.eg

REFERENCES

[1] A. Smith et al., "Real-time recommendations in e-commerce," IEEE Trans. Big Data, vol. 5, no. 2, pp. 101-115, 2023.

[2] B. Johnson et al., "Microservices architecture patterns," in Proc. IEEE Cloud Computing, 2023, pp. 45-52.