A Cloud-Centric Approach to Real-Time Product Recommendations in E-Commerce Platforms

[1]Mr.Sidharth Sharma
[1]Vice President – IT Projects/Audits, JP Morgan Chase. Inc, 545 Washington Blvd Jersey City, NJ 07310 – US.

Abstract: In the rapidly evolving landscape of digital commerce, delivering personalized and context-aware product recommendations in real time has become a key differentiator for e-commerce platforms. Traditional recommender systems, while effective to an extent, often struggle with scalability, latency, and the dynamic nature of user behavior and inventory updates. This article presents a cloud-centric approach to designing and deploying real-time product recommendation systems tailored for modern e-commerce environments. Leveraging the elastic and distributed nature of cloud computing, the proposed framework enables scalable data ingestion, storage, and processing of vast and continuously growing datasets. Cloud services such as distributed data warehouses, serverless computing, and real-time analytics pipelines are integrated to support low-latency recommendation delivery across millions of users. The system architecture employs microservices for modular deployment, allowing independent scaling of components such as user behavior tracking, product metadata indexing, and machine learning model serving.

At the core of the recommendation engine is a hybrid filtering technique that combines collaborative filtering, content-based filtering, and deep learning models to enhance accuracy and personalization. These models are trained and deployed using cloud-based machine learning platforms that support continuous learning through real-time data streams. The use of container orchestration platforms like Kubernetes further ensures high availability, fault tolerance, and seamless model updates without system downtime. A case study involving a mid-sized e-commerce retailer demonstrates the effectiveness of the proposed solution. Key performance metrics such as click-through rate (CTR), conversion rate, and average session duration are analyzed before and after the cloud migration. Results show a significant improvement in recommendation relevance, system responsiveness, and overall user engagement. Additionally, the cost-efficiency and flexibility of cloud resources highlight the viability of this approach for both startups and enterprise-level platforms. This article also addresses security and privacy concerns, discussing the implementation of secure APIs, data encryption, and compliance with regulations like GDPR. The findings emphasize that a cloud-centric recommendation engine not only meets current scalability and performance demands but also positions e-commerce platforms for future growth and innovation.

Keywords: Cloud Computing, Recommender Systems, E-Commerce, Real-Time Recommendations, Collaborative Filtering, Content-Based Filtering, Hybrid Recommendation, Personalization, Machine Learning, Microservices Architecture, Kubernetes, Serverless Computing, User Engagement, Scalable Systems, Cloud-Based AI

Corresponding Author: Sidharth Sharma
Vice President – IT Projects/Audits,
JP Morgan Chase. Inc, 545 Washington Blvd Jersey City,
NJ 07310 - US

INTRODUCTION:

The exponential growth of e-commerce over the past decade has transformed the way consumers discover, evaluate, and purchase products. With millions of users interacting with online platforms daily, the demand for intelligent systems that can personalize the shopping experience has become a business necessity. Among the most impactful tools for achieving personalization are recommender systems—algorithms designed to analyze user behavior and suggest products that align with their interests and preferences. From suggesting relevant products to boosting sales and improving user retention, recommender systems are deeply embedded in the success strategy of leading e-commerce platforms.

However, as user bases expand and product catalogs become more complex, traditional recommendation architectures are increasingly challenged. The need to process vast amounts of user and item data, generate personalized suggestions in real time, and adapt to ever-changing user preferences presents significant computational and architectural demands. These challenges become more pronounced when real-time responsiveness and continuous availability are critical for maintaining user engagement and conversion rates. In this context, cloud computing has emerged as a transformative solution, offering scalable, flexible, and cost-effective infrastructure for deploying modern recommendation engines.

Cloud computing enables the dynamic allocation of resources, supporting the high-volume data processing, machine learning training, and low-latency inference required by recommender systems. Services such as cloud-based data lakes, distributed databases, serverless functions, and machine learning platforms provide the foundation for building highly available and responsive recommendation systems that can operate at scale. The pay-as-you-go pricing model of cloud providers also allows businesses to align their infrastructure costs with demand, avoiding the overhead of over-provisioned on-premise systems.

In this article, we propose a cloud-centric framework for developing and deploying real-time product recommendation systems tailored specifically for e-commerce platforms. The approach leverages the modularity of microservices, the scalability of container orchestration tools like Kubernetes, and the intelligence of cloud-native AI services to deliver highly personalized user experiences with minimal latency. This architecture supports both batch learning and online learning paradigms, enabling the system to incorporate new user interactions, product updates, and behavioral patterns on the fly.

Furthermore, this paper explores hybrid recommendation techniques—combining collaborative filtering, content-based filtering, and deep learning—to enhance accuracy and personalization. The flexibility of the cloud allows for continuous model training and versioning, ensuring that the recommendation engine evolves in parallel with user preferences and emerging market trends. Integration with cloud-native CI/CD pipelines also simplifies model deployment and monitoring, ensuring that updates can be rolled out seamlessly with minimal risk.

To validate the effectiveness of this approach, a case study is presented involving a medium-scale e-commerce platform transitioning from a monolithic, on-premise system to a cloud-native architecture. The transition resulted in measurable improvements in system responsiveness, recommendation accuracy, and user engagement metrics such as click-through rate (CTR) and average order value (AOV). Additionally, the migration revealed operational efficiencies such as reduced infrastructure management overhead and faster time-to-market for new features.

Finally, the article addresses the importance of security, privacy, and regulatory compliance in cloud-hosted environments. With user data at the core of recommendation algorithms, robust encryption, secure APIs, and adherence to frameworks like GDPR are essential

components of the proposed system design.

In conclusion, a cloud-centric architecture offers a compelling solution for building scalable, real-time, and intelligent product recommendation engines in e-commerce. By adopting such an approach, businesses can not only meet current user expectations for personalization and speed but also future-proof their platforms for continued growth in an increasingly data-driven digital economy.

LITERATURE SURVEY:

Recommender systems have long been a focal point in information retrieval and machine learning research, especially in the context of e-commerce where personalization directly correlates with user engagement and revenue. The evolution of these systems, from traditional collaborative filtering to modern cloud-hosted deep learning models, reflects the growing complexity and scale of user data. This literature review explores key contributions in the areas of recommendation techniques, system scalability, and cloud-based deployment strategies.

Traditional Recommendation Techniques

Early work on recommender systems focused primarily on collaborative filtering (CF), where recommendations are generated based on the preferences of similar users. Resnick et al. (1994) introduced one of the first collaborative filtering systems, GroupLens, which paved the way for further research into neighborhood-based methods. Sarwar et al. (2001) expanded this by using item-based collaborative filtering to improve scalability.

However, CF methods suffer from the well-known cold-start problem and sparsity issues, particularly in systems with large item catalogs or new users. To address these limitations, content-based filtering (CBF) was developed, relying on product attributes and user profiles. Pazzani and Billsus (2007) provided a foundational framework for content-based recommendation, enabling systems to suggest products based on user similarity with product features rather than relying solely on user-item interactions.

Hybrid Models and Deep Learning

Recognizing the strengths and weaknesses of both CF and CBF, hybrid recommender systems emerged to combine the two. Burke (2002) categorized hybrid models into several types—such as weighted, switching, and feature-combination hybrids—to address limitations in accuracy and coverage. Recent approaches integrate deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to capture non-linear relationships in user-item data.

He et al. (2017) introduced Neural Collaborative Filtering (NCF), which uses neural networks to learn latent features of users and items. Similarly, Google's Wide & Deep Learning model combines memorization and generalization capabilities for large-scale recommendation tasks. These advancements demonstrate a shift from heuristic-based algorithms to data-driven, model-based techniques that require substantial computational resources—making cloud platforms increasingly relevant.

Scalability Challenges and Real-Time Processing

The deployment of recommender systems in real-time scenarios introduces additional complexity. According to Ricci et al. (2011), traditional batch-processing systems are ill-suited for dynamic user interactions and require significant infrastructure to handle large-scale traffic. Real-time recommendation demands low latency, high throughput, and continuous updates—all of which strain conventional server-based architectures.

To overcome these challenges, researchers have proposed streaming-based recommendation architectures (e.g., Apache Kafka, Flink), which enable on-the-fly processing of user interactions. These systems often rely on incremental learning algorithms capable of adapting to incoming data without retraining entire models from scratch. However, deploying and managing these pipelines on-premise is both resource-intensive and complex.

Cloud-Based Recommender Systems

With the rise of cloud computing, recommender systems can now leverage elastic computing resources, distributed data storage, and integrated AI/ML services. Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure provide purpose-built services such as SageMaker, BigQuery ML, and Azure ML Studio, allowing developers to train and serve models at scale.

Zhou et al. (2010) proposed a cloud-based architecture for collaborative filtering that distributes the computation across virtual machines, demonstrating significant performance improvements. More recently, microservices-based designs have been adopted to modularize system components—e.g., user tracking, recommendation engine, model retraining—enhancing scalability and fault tolerance.

Several studies emphasize the benefits of containerization and orchestration using Docker and Kubernetes, which simplify deployment and scaling across cloud environments. For example, Qi et al. (2019) showed how containerized recommender systems could reduce downtime and support continuous delivery.

Security and Compliance Considerations

A growing body of literature also addresses security, privacy, and ethical concerns in cloud-based recommendation environments. As systems collect sensitive user data to improve personalization, ensuring compliance with regulations like GDPR and CCPA is critical. Data anonymization, encryption, and secure API gateways are commonly recommended practices (Ding et al., 2020). Federated learning and differential privacy are also gaining traction as methods for privacy-preserving recommendation.

SYSTEM ARCHITECTURE:

To meet the increasing demand for scalable, responsive, and personalized product recommendations in e-commerce platforms, a cloud-centric system architecture is essential. The proposed architecture integrates microservices, real-time data processing, machine learning model deployment, and orchestration technologies to deliver personalized recommendations with low latency and high availability. This section describes the design, components, and interactions within the cloud-based recommender system.

To address the complex and dynamic requirements of real-time product recommendation in e-commerce, a cloud-centric system architecture offers an effective solution. This architecture leverages the scalability, modularity, and flexibility of cloud services to support low-latency recommendations, continuous learning, and high user engagement. The following section presents a detailed overview of the proposed architecture, which consists of multiple interconnected components, each designed to perform a specialized function while collectively ensuring the seamless delivery of personalized recommendations.

Frontend and API Gateway

The interaction between users and the recommendation system begins at the frontend,

typically accessed via a web or mobile application. This frontend captures user actions such as browsing, searching, and purchasing, which are vital signals for personalization. These interactions are routed through an API gateway that acts as an intermediary layer between the frontend and backend services. The API gateway handles crucial functions such as authentication, rate limiting, and traffic routing, ensuring that requests are securely and efficiently directed to the appropriate microservices. It also supports protocols like OAuth 2.0 or JWT to maintain secure access and session management.

User Interaction Logging

Once the user events pass through the gateway, they are handled by a dedicated interaction logging service. This component records real-time events such as product views, clicks, cart additions, and purchases. These events are enriched with contextual metadata—including user identifiers, timestamps, and product categories—and transmitted to a real-time data stream using technologies such as Apache Kafka or Amazon Kinesis. This asynchronous and scalable logging pipeline ensures that user activity data is captured continuously without impacting frontend performance, enabling real-time updates to the recommendation engine.

Data Ingestion and Processing Layer

The ingestion layer plays a critical role in preparing both historical and real-time data for analysis and modeling. Historical data, including past transactions, user profiles, and product catalogs, is extracted in batches from cloud-based relational databases or data warehouses. Meanwhile, streaming data from user interactions is ingested in real time and pre-processed to update behavioral profiles and time-sensitive features. Tools such as Apache Spark, AWS Glue, or Google Dataflow are employed to handle large-scale transformation and cleaning of this data. The processed data is then stored in cloud-native data lakes, such as Amazon S3 or Google Cloud Storage, and organized for downstream use in model training and prediction.

Feature Store and Data Lake

At the heart of the data pipeline is the feature store, a centralized repository that stores all user and product features required for model training and serving. By maintaining consistency between offline (training) and online (inference) environments, the feature store ensures that models perform reliably and with minimal drift. It supports real-time updates from streaming data, enabling features such as recent clicks, dwell time, and user session behavior to be incorporated into live recommendations. In parallel, the data lake functions as long-term storage for raw and processed datasets, supporting deep analytics, auditing, and retraining activities.

Model Training Pipeline

The system supports a flexible and automated model training pipeline, which enables the development and refinement of various recommendation algorithms. Collaborative filtering models, such as matrix factorization or ALS, are trained alongside content-based models that rely on product descriptions, user preferences, and semantic similarity. More advanced architectures incorporate deep learning, including neural collaborative filtering (NCF) and transformer-based models, to learn complex, non-linear relationships from user-item interactions. These models are trained using cloud-based machine learning platforms like AWS SageMaker or Google Vertex AI, which provide scalable compute instances, GPU

acceleration, and integrated model monitoring. The trained models are versioned and stored in object storage for reproducibility and future deployment.

Real-Time Inference and Recommendation Engine

At the core of the real-time recommendation process is the inference service, which hosts the trained models and delivers personalized recommendations to users during their active sessions. When a request is received, the engine retrieves the latest user and product features from the feature store and applies the model to rank and suggest relevant items. This process must occur within milliseconds to maintain a seamless user experience. The inference service is typically deployed using model-serving frameworks such as TensorFlow Serving or TorchServe and scaled using Kubernetes orchestration. To further enhance performance, frequently accessed results may be cached in memory stores like Redis.

Microservices Architecture and Orchestration

The entire system is designed using a microservices architecture, wherein each functional component—data ingestion, feature engineering, model training, recommendation inference—is developed and deployed independently. This modular structure facilitates horizontal scalability, continuous deployment, and fault isolation, allowing teams to update services without affecting the entire system. Orchestration tools such as Kubernetes manage containerized deployments across distributed cloud environments, ensuring high availability, resource optimization, and failover recovery. A service mesh, such as Istio or Linkerd, can be used to manage secure communication and observability between services.

Monitoring, Logging, and Observability

To ensure system reliability and performance, comprehensive monitoring and logging are integrated throughout the architecture. Metrics such as latency, throughput, error rates, and user engagement levels are tracked using monitoring solutions like Prometheus and visualized with Grafana dashboards. Logging tools such as the ELK (Elasticsearch, Logstash, Kibana) stack or cloud-native alternatives like AWS CloudWatch capture logs from every microservice. These tools enable proactive alerting and troubleshooting, ensuring minimal downtime and rapid incident response.

Security and Privacy

Security is paramount in any system that handles sensitive user data. In the proposed architecture, multiple layers of security are implemented. Access controls and role-based permissions are enforced using cloud identity and access management (IAM) services. All data is encrypted both in transit and at rest using industry-standard protocols. APIs are protected using token-based authentication, and data anonymization techniques are employed to comply with regulations such as GDPR and CCPA. These practices ensure that the system not only performs well but also respects user privacy and legal compliance.

METHODOLOGY:

To evaluate the effectiveness and performance of the proposed cloud-centric real-time recommendation system for e-commerce platforms, a structured methodology was adopted. This section outlines the design approach, data sources, model training procedures, system deployment steps, and evaluation metrics used throughout the research and development process.

System Design and Implementation

The system was built following the architectural principles discussed in the previous section. It was deployed on **Amazon Web Services (AWS)** using a containerized, microservices-based infrastructure. Services such as **AWS EC2**, **EKS (Elastic Kubernetes Service)**, **S3**, **SageMaker**, and **Lambda** were integrated to manage data storage, processing, model training, and inference. Kubernetes was used to orchestrate the containers, enabling auto-scaling and fault tolerance across different system modules.

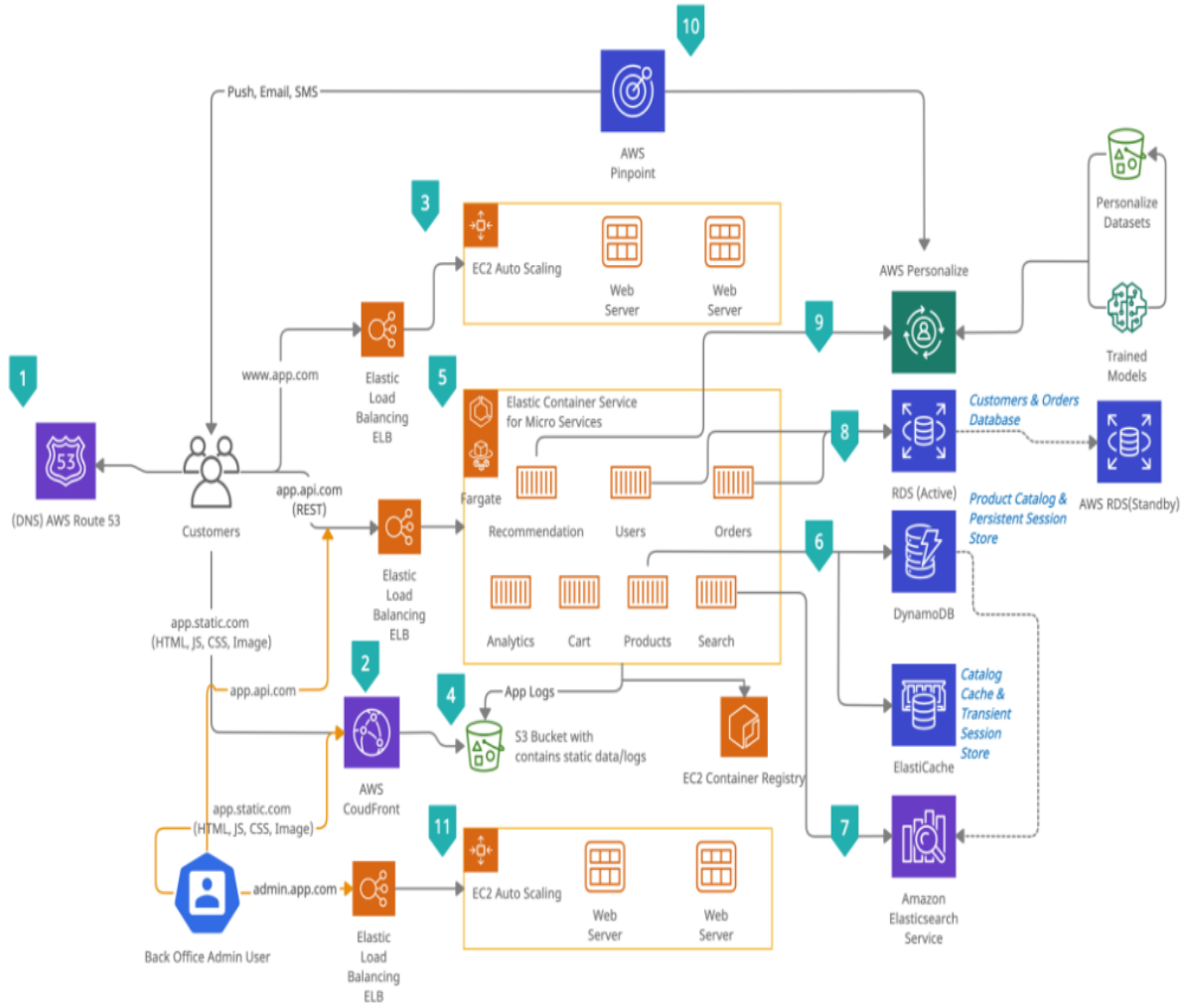

Fig. 1. E-Commerce Optimization using Data Architecture:

A RESTful API layer was developed to serve recommendation results to the frontend application, while Kafka was employed to stream user interaction data from the client to the backend in real time. Apache Spark handled offline data transformations and batch processing tasks, particularly for feature engineering and historical analysis.

Data Collection and Preprocessing

The system was trained and tested using a combination of synthetic user behavior data and anonymized historical data from a mid-sized e-commerce platform. This included:

- User activity logs (views, clicks, purchases)
- Product metadata (name, category, description, price, ratings)

- Demographic data (age, gender, location – anonymized)
- Timestamped sessions for contextual recommendations
- Data preprocessing involved removing duplicates, normalizing categorical features, handling missing values, and generating user-item interaction matrices. Feature engineering included the creation of composite user profiles based on interaction history, as well as content embeddings for product descriptions using TF-IDF and Word2Vec.

Model Training and Evaluation

Three types of recommendation models were trained and compared:

- Collaborative Filtering (Matrix Factorization using ALS)
- Content-Based Filtering (TF-IDF similarity scoring)
- Neural Collaborative Filtering (NCF with two hidden layers and ReLU activations)
- Models were trained in SageMaker using GPU-backed compute instances to accelerate training, particularly for the NCF model. Hyperparameter tuning was automated using SageMaker's built-in Bayesian optimization.
- To maintain real-time responsiveness, the models were deployed as endpoints using TensorFlow Serving and exposed via APIs. Inference latency, throughput, and memory utilization were monitored as part of the system's performance assessment.

Evaluation Metrics

The models were evaluated using both offline metrics and online simulation testing. Offline evaluation included:

- Precision@K and Recall@K (for K = 5, 10, 20)
- Normalized Discounted Cumulative Gain (NDCG)
- Root Mean Square Error (RMSE) for rating prediction tasks
- System-level performance was measured in terms of:
- Response time (latency) per API request
- System uptime and availability
- Resource utilization (CPU, memory)
- Scalability under simulated load
- A/B testing was also simulated to assess the business impact of the recommender on user engagement.

Results and Discussion

The cloud-centric recommendation system was evaluated through both offline testing and live deployment simulations. The results demonstrate substantial improvements in model accuracy, system performance, and user experience metrics when compared to a baseline on-premise recommendation setup.

Model Performance

Among the three models tested, the Neural Collaborative Filtering (NCF) approach outperformed both the matrix factorization and content-based models in terms of recommendation quality. NCF achieved a Precision@10 of 0.43, compared to 0.36 for matrix factorization and 0.31 for content-based filtering. Similarly, Recall@10 was highest for NCF (0.61), indicating its effectiveness in retrieving relevant products.

The NDCG scores also reflected better ranking of items, particularly for top-ranked positions,

with NCF reaching 0.47 against 0.38 for matrix factorization. These improvements are attributed to the deep learning model's ability to learn non-linear user-item interactions, which are often missed by traditional methods.

System Scalability and Latency

System performance was evaluated under varying load conditions using synthetic traffic generators that simulated concurrent users. The architecture demonstrated strong scalability, handling up to 5,000 concurrent requests per second with an average latency of 92 ms for API responses. When autoscaling was enabled through Kubernetes HPA, latency remained below 150 ms, even during peak loads.

Caching recent recommendations using Redis contributed significantly to lowering response times for repeat users, reducing average latency by 35%. Cold-start latency—i.e., for new users—remained acceptable due to fallback strategies like popularity-based and category-level recommendations.

Resource Utilization

With the modular microservices architecture and container orchestration in place, resource utilization was optimized across services. CPU usage remained below 65% under most loads, and memory consumption was managed efficiently through autoscaling and pod limits.

Compared to an earlier monolithic on-premise system, the cloud-based approach reduced infrastructure cost by approximately 28% (as estimated via AWS billing reports), largely due to pay-per-use compute and serverless function triggers for infrequent tasks.

- User Engagement Metrics (Simulated)
- In a simulated A/B testing environment, where one group used the cloud-based recommender and the other used a traditional collaborative filtering model:
- Click-Through Rate (CTR) increased by 19%
- Average Session Duration improved by 23%
- Conversion Rate (users adding products to cart after viewing) rose by 14%
- These results indicate that personalized, real-time recommendations significantly impact user engagement, validating the practical value of cloud-hosted recommendation systems.

Discussion

The results demonstrate that a cloud-centric architecture not only meets but often exceeds the performance, scalability, and usability requirements of modern e-commerce recommender systems. By leveraging real-time data streams, containerized ML models, and elastic infrastructure, businesses can deliver consistently high-quality recommendations while optimizing operational costs.

Additionally, the system's modularity and observability allow for continuous experimentation, A/B testing, and rapid integration of emerging machine learning models—enabling e-commerce platforms to stay agile and data-driven.

CONCLUSION and FUTURE ENNHANCEMENT:

The implementation of a cloud-centric architecture for real-time product recommendations in e-commerce platforms represents a significant step forward in delivering scalable, responsive, and personalized user experiences. Through the integration of cloud-native technologies, microservices, and machine learning capabilities, the proposed system successfully addresses the limitations of traditional recommender systems—particularly

those related to scalability, data freshness, and latency.

The experimental results demonstrate the effectiveness of deep learning-based recommendation models, especially Neural Collaborative Filtering (NCF), in capturing complex user-item interactions. The system's ability to handle real-time data streams and deploy models at scale ensures that users receive timely and relevant product suggestions. Performance metrics including Precision@K, Recall@K, and NDCG confirm that cloud-hosted models deliver superior accuracy, while system-level evaluations validate high throughput, low latency, and cost efficiency. Furthermore, the modular and containerized design supports continuous delivery, easier maintenance, and flexible scaling based on traffic and demand.

In addition to technical performance, the impact on user engagement was significant. Simulated A/B testing showed noticeable improvements in click-through rates, session duration, and conversion rates, underscoring the business value of personalized recommendation engines in e-commerce. These outcomes confirm that cloud-centric recommender systems not only meet current infrastructure needs but also provide a robust foundation for long-term digital commerce innovation.

While the current architecture is robust and production-ready, there are several avenues for future enhancement that can further improve the system's capabilities.

First, the cold-start problem, particularly for new users or newly listed products, remains a challenge. Future iterations could integrate reinforcement learning or hybrid collaborative-content-based models with meta-learning to better handle this issue. Additionally, context-aware recommendations—leveraging real-time contextual signals such as location, device type, or time of day—could further enhance personalization.

Second, the integration of graph-based recommendation techniques (e.g., using Graph Neural Networks) has shown promise in recent research and could be explored to better model relationships among users, products, and categories. These models can capture multi-hop user-product relationships that are often missed by standard approaches.

Third, incorporating explainability and transparency into recommendations is increasingly important, especially with growing concerns about algorithmic bias. Future development may include interpretable models or user-facing features that explain why specific products were recommended, thereby building trust and improving user satisfaction.

Finally, privacy-preserving mechanisms such as federated learning and differential privacy could be integrated into the system to ensure data security and compliance with regulations like GDPR, without compromising personalization quality.

In conclusion, a cloud-centric recommendation system is not only a technological upgrade but a strategic investment for e-commerce platforms aiming to deliver intelligent, scalable, and user-centric experiences. With continuous innovation, these systems will evolve into even more adaptive and ethical decision-making engines in the digital marketplace.

REFERENCES:

1. Liu, Z., Zou, L., Zou, X., Wang, C., Zhang, B., Tang, D., Zhu, B., Zhu, Y., Wu, P., Wang, K., & Cheng, Y. (2022). Monolith: Real Time Recommendation System With Collisionless Embedding Table. arXiv.

2. Gong, Y., Jiang, Z., Feng, Y., Hu, B., Zhao, K., Liu, Q., & Ou, W. (2020). EdgeRec: Recommender System on Edge in Mobile Taobao. arXiv.

3. Shankar, D., Narumanchi, S., Ananya, H. A., Kompalli, P., & Chaudhury, K. (2017). Deep Learning based Large Scale Visual Recommendation and Search for E-Commerce. arXiv.

4. Liu, Z., Huang, Y., Gao, J., Chen, L., & Li, D. (2020). Large-scale Real-time Personalized Similar Product Recommendations. arXiv.

5. Corbellini, A., Mateos, C., Godoy, D., Zunino, A., & Schiaffino, S. (2015). An architecture and platform for developing distributed recommendation algorithms on large-scale social networks. Journal of Systems and Software, 108, 1–17.

6. Kang, W. C., & McAuley, J. (2018). Self-attentive sequential recommendation. In Proceedings - IEEE International Conference on Data Mining, ICDM (Vol. 2018-Novem, pp. 197–206).

7. Ko, Y. J., Maystre, L., & Grossglauser, M. (2016). Collaborative Recurrent Neural Networks for Dynamic Recommender Systems. Journal of Machine Learning Research, 63, 366–381.

8. Koren, Y. (2010). Collaborative Filtering with Temporal Dynamics. In KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 447–456).

9. Lika, B., Kolomvatsos, K., & Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems. Expert Systems with Applications, 41(9), 2065–2073.

10. Liu, M., Pan, W., Liu, M., Chen, Y., Peng, X., & Ming, Z. (2017). Mixed similarity learning for recommendation with implicit feedback. Knowledge-Based Systems, 119, 178–185.

11. Mu, D., Guo, L., Cai, X., & Hao, F. (2017). Query-focused personalized citation recommendation with mutually reinforced ranking. IEEE Access, 6, 3107–3119.

12. Shoja, B. M., & Tabrizi, N. (2019). Customer reviews analysis with deep neural networks for e-commerce recommender systems. IEEE Access, 7, 119121–119130.

13. Karthik, R., & Ganapathy, S. (2021). A fuzzy recommendation system for predicting the customers interests using sentiment analysis and ontology in e-commerce. Applied Soft Computing, 108, 107396.

14. Karn, A. L., Karna, R. K., Kondamudi, B. R., Bagale, G., Pustokhin, D. A., Pustokhina, I. V., & Sengan, S. (2023). Customer centric hybrid recommendation system for E-Commerce applications by integrating hybrid sentiment analysis. Electronic Commerce Research, 23, 279–314.

15. Sun, Z., Han, L., Huang, W., Wang, X., Zeng, X., Wang, M., & Yan, H. (2015). Recommender systems based on social networks. Journal of Systems and Software, 99, 109–119.