Survey of Big Data Architectures and Frameworks on Kubernetes: Challenges, Solutions, and Future Directions

Prashanth Josyula*
Principal Member of Technical Staff
Salesforce
San Francisco, USA
prashanth.16@gmail.com

Anant Kumar
Lead Member of Technical Staff
Salesforce
San Francisco, USA
garg.anant@gmail.com

Gangadharayya Hiremath
Lead Member of Technical Staff
Salesforce
Dallas, USA
gangavh@gmail.com

2025 10th International Conference on Big Data Analytics (ICBDA) | 979-8-3315-0393-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICBDA65366.2025.11211300

***Abstract*—The convergence of big data processing frameworks and container orchestration platforms, particularly, Kubernetes, represents a new path for modern data infrastructure. This comprehensive survey examines the evolving landscape of Big Data deployments in Kubernetes, analyzing traditional frameworks refactored for containerized environments with new native ones under development. We investigate significant challenges in stateful storage management, scheduling of resources, and performance optimization. Through systematic analysis of current approaches and architectures, we identify patterns, best practices, and areas requiring further research work. According to our observations, though, Kubernetes offers compelling advantages for Big Data workloads, several technical challenges remain in achieving optimal performance and reliability.**

***Index Terms*—Big Data, Kubernetes, Container Orchestration, Distributed Computing, Cloud Native, Data Processing, Resource Management, Containerization**

I. Introduction

Organizations are looking for more flexible, scalable, and efficient infrastructure solutions to meet their exponential growth in data generation and processing requirements. The convergence of Big Data processing frameworks with container orchestration platforms represents a critical evolution in modern distributed computing, given the expected global data creation projected to exceed 175 zettabytes by 2025. Kubernetes has emerged as the industry standard for container orchestration, while Big Data frameworks are evolving to meet rapidly increasing analytical demands.

Traditional Big Data designs, created initially for bare-metal or virtualized environments, assumed relatively static infrastructure with predictable resource allocation patterns, however, the transition toward cloud-native architectures has challenged these assumptions of how to deploy and manage Big Data workloads in containerized environments, while Kubernetes offers compelling advantages through its declarative configuration, automated deployment, and dynamic scaling. The integration of Big Data frameworks and Kubernetes poses enormous technical challenges, especially in the areas such as storage management, resource scheduling, and performance tuning.

According to the Cloud Native Computing Foundation (CNCF), Kubernetes emerged as the de facto standard for container orchestration and is being widely used across industries. This container orchestration platform automates application container deployment, scaling, and management across host clusters and is a requirement for distributed system management, all at once, simultaneously, Big Data frameworks are evolving beyond batch processing to support real-time analytics, streaming, and machine learning workloads.

The intersection of Big Data processing frameworks and container orchestration platforms creates both significant opportunities and complex technical challenges. Big Data architectures developed for bare-metal environments must adapt to containerized infrastructure, which is dynamic in nature. This adaptation requires a major change in thought process in understanding the fundamental aspects of data processing,storage management, and resource allocation.

This survey makes an exhaustive review of Big Data architectures on Kubernetes, with the following objectives:

- Analyze architectural patterns and implementation strategies for deploying Big Data frameworks on Kubernetes
- Evaluate performance implications and operational considerations of containerized Big Data workloads
- Identify key challenges and existing solutions in the field
- Highlight promising research directions and open problems

Our methodology encompasses the systematic review of academic literature, industry practices, and opensource projects. The rest of this paper is organized as follows : Section II presents background information related to Big Data architectures and Kubernetes. Section III studies how traditional BigData frameworks adapt to Kubernetes. Section IV explores native Kubernetes solutions. Section V presents technical challenges and solutions while Section VI addresses security and compliance considerations. Section VII presents the future research directions, and Section VIII concludes the survey.

979-8-3315-0393-2/25/$31.00 ©2025 IEEE

II. Background and Evolution

The field of distributed computing and data processing has changed significantly in the last ten to twenty years. This is due to the growing need for data processing on a large scale and the need for a more flexible infrastructure management.

A. Traditional Big Data Architectures

According to Dean and Ghemawat [1], all the roots for the evolution of Big Data processing frameworks can be traced back to MapReduce. This introduces the programming model that changes how we process huge datasets; it does so by breaking down complex computations into map and reduce operations that can run simultaneously on clusters of commodity hardware.

Building on this foundation, the Apache Hadoop ecosystem provided a complete solution for distributed data processing. One of the important evolutions came with Apache Spark, proposed by Zaharia et al. [2], that addressed the limitations of MapReduce thanks to the introduction of resilient distributed datasets (RDDs) and in-memory processing capabilities. This allowed substantial performance improvements for iterative algorithms and interactive data analysis.

Data warehousing solutions also evolved in order to meet the ever increasing needs of organizations. Thusoo et al. [3] created Hive, which offered a SQL-like interface over MapReduce and this made it easier for analysts who were already experienced with traditional database systems to process Big Data. Coordination services like ZooKeeper [4] were also developed to complement these advancements and became necessary for controlling the state of distributed systems and ensuring that the components of a cluster remained consistent with one another.

B. Container Orchestration with Kubernetes

Kubernetes, which was developed by Google based on its experience with internal systems like Borg, changed the container orchestration environment. According to Burns et al. [5], Kubernetes was developed using the lessons learned from Borg and Omega, and it includes the knowledge that would make container orchestration more accessible and standardized for the broader industry.

According to Verma et al. [6], Google's experience with managing large-scale clusters influenced the architecture of Kubernetes, particularly in terms of scheduling workloads and managing resources. This foundation was essential for supporting a variety of application types, including Big Data workloads.

This transition has challenged the traditional assumptions about infrastructure stability and resource allocation provided by the containerized environments. As Schwarzkopf et al. [7] pointed out, the scheduler architectures need to adapt to dynamic containerized workloads while retaining performance and reliability. The modern Big Data architectures are increasingly adopting hybrid approaches, thereby combining the best aspects of traditional frameworks with cloud-native principles. The creation of new storage systems such as Ceph [8] provided the necessary foundation for highly flexible and scalable data management in containerized environments. These developments opened the road to new approaches to resource management, as illustrated by Ghodsi et al. [9] in their work on multi-resource fairness, which becomes particularly relevant in the context of containerized Big Data workloads.

III. Adaptation of Traditional Big Data Frameworks

The transition of Big Data frameworks to containerized environments represents a significant change in the architecture of distributed computing. In their 2021 work on lakehouse architecture, Armbrust et al. [10] illustrated how recent developments in unified analytics platforms are bridging the gap between advanced analytics in cloud-native environments and traditional data warehousing.

A. Hadoop on Kubernetes

From its MapReduce origins, Hadoop's adaptability for containerized systems have evolved significantly. The containerization and orchestration of Hadoop components have been influenced by recent research on serverless computing trends conducted by Baldini et al. [12]. Three critical areas of contemporary deployments are addressed by this evolution:

Storage architecture has undergone substantial transformation to support container-native patterns. The emergence of lakehouse architectures [10] has driven new approaches to data persistence and access patterns in containerized environments.

Serverless computing principles [12] are now incorporated into the resource management integration between YARN and Kubernetes. This contemporary approach preserves the performance characteristics needed for Big Data applications while allowing for more flexible resource allocation.

Operational challenges and solutions represent the third crucial area of evolution, requiring careful consideration in containerized deployments to ensure reliable system operation.

B. Spark on Kubernetes

The core design of Apache Spark, which was first introduced by Zaharia et al. [[2] in their groundbreaking work, was later adapted to work with Kubernetes, Spark's native support for Kubernetes marks a significant advancement in the way container orchestration systems and distributed computing frameworks communicate.

The resource management principles demonstrated by Schwarzkopf et al. [7] have evolved through the scheduler integration between Spark and Kubernetes, Spark's performance features are maintained while dynamic resource allocation is made possible by this integration and also the approach preserves Spark's complex task scheduling and data locality optimizations while leveraging Kubernetes' native scheduling capabilities.

There are advantages and challenges in comparing the performance with traditional deployment methods. Modern data processing systems that are similar to those outlined by Armbrust et al. [10] shows how well Kubernetes' elasticity

fits Spark's dynamic resource requirements. However, these advantages must be weighed against the complexity of managing stateful components and the overhead introduced by the containerization.

Dynamic scaling in containerized Spark deployments is built on cluster management research, following the principles described by Burns et al. [5] in their work on Kubernetes architecture. These capabilities improve resource utilization but also presents new challenges in maintaining performance consistency and data localization.

Recent developments have focused on integration of Spark with emerging streaming architectures, building on messaging system principles documented in work by Kreps et al. [11], this integration enables flexible options for deploying stream processing workloads with performance characteristics similar to modern Big Data systems.

IV. Native Kubernetes Big Data Solutions

The evolution of big data frameworks has driven a paradigm shift, with Kubernetes-native solutions like Ray and Dask addressing critical limitations of traditional systems through enhanced scalability, low-latency orchestration, and seamless ML integration.

A. Emerging Frameworks

While Hadoop and Spark excel at batch processing, they lack dynamic scaling and Python-centric workflows needed for real-time analytics. Ray and Dask address these limitations through native Kubernetes integration, enabling distributed workloads with minimal overhead. Ray's actor model achieves low latency for ML tasks, while Dask's dynamic task graphs optimize parallelism for ETL operations.

1) Ray: Distributed computing framework with native Kubernetes support: Ray's stateless design with schedulers, workers, and drivers enables high availability and seamless integration with Kubernetes' pod lifecycle and auto-scaling. Implemented as StatefulSets, Ray's Global Control Store provides sub-second failover and horizontal scaling across thousands of nodes. The bottom-up distributed scheduling algorithm, combined with Kubernetes' scheduler, delivers optimal resource utilization and task completion times. Leveraging native QoS classes and resource quotas, Ray achieves linear scaling while reducing costs. Its advanced checkpointing system with lineage-based recovery ensures data consistency and rapid node failure recovery. The zero-copy shared memory architecture demonstrates superior I/O performance and lower latencies in real-time processing compared to Hadoop and Spark.

2) Dask: Flexible parallel computing library: Through Kubernetes' HPA and Custom Metrics API, Dask's dynamic task scheduler achieves optimal resource utilization across heterogeneous workloads with efficient scaling and cost management. Its stateless worker architecture, utilizing DAG-based lineage tracking, enables reliable task recomputation and aligns with Kubernetes' pod lifecycle management. Dask leverages advanced Kubernetes features like topology spread constraints and node affinity rules for optimized computational density and reduced communication overhead. The framework's array and dataframe implementations integrate with Kubernetes' CSI and StorageClass abstractions, optimizing data locality and I/O throughput through RWX persistent volumes. This integration delivers robust horizontal scalability and resource elasticity with minimal scheduling overhead.

3) Performance Characteristics and Architectural Advantages: Ray's actor model allows for task scheduling at the microsecond level [21], making it a very good framework choice for stateful, high-throughput ML pipelines such as reinforcement learning or real-time inference. Dask's dynamic task graphs optimize for flexible parallelism in analytics workflows, outperforming Spark in iterative computations common in scientific computing [22]. In contrast, Spark leverages in-memory caching and optimized SQL execution engines (e.g., Photon) for ETL/analytics at petabyte scale, though its JVM-based architecture introduces higher latency (100ms) for fine-grained tasks [23]. For iterative algorithms, despite being robust for fault-tolerant batch processing, Hadoop MapReduce pays a heavy I/O cost from disk-based shuffling and executes 10-100x more slowly than Spark/Ray, as demonstrated in White 2015 [24].

B. Specialized Solutions

Analysis of purpose-built solutions for specific Big Data workloads:

There is core shift in data architecture propelled by exponential growth in volumes of real-time data. In 2024, for example, IDC estimated that 30% of all data produced-64.2 zettabytes-would require real-time processing [25]. Traditional batch systems processed data in fixed intervals, thus becoming increasingly unsuitable for a number of modern use cases demanding immediate decision-making. Eventually, several hours or days of latency became intolerable for organizations in the early 2000s. Hadoop, though revolutionary for large-scale analytics, showed average processing delays of 22 minutes for 100GB datasets [21]. This latency turned out to be unsustainable for key applications:

- fraud detection requires decisions to be made in 300ms [26]
- recommendation engines need response times of 100ms [27] and
- systems monitoring has to fire alerts in less than one second [28]

The stream processing frameworks that arrived as a technological answer; reached incredible performance metrics:

- Sub-millisecond latency
- Processing capabilities of 1M+ events/second
- Exactly-once processing guarantees with 99.99% reliability
- State management supporting petabyte-scale operations

The evolution continued with specialized ML frameworks addressing scale challenges. Recent studies indicate:

- 67% reduction in model deployment time using integrated MLOps platforms [29]

- 89% improvement in experiment reproducibility [30]
- Faster development cycles with automated workflows
- Major cost reduction through optimized resource utilization

Already, the integrated streaming and ML platform has enabled new use cases that achieved the following:

- Less than 50ms real-time fraud detection at 99.7% accuracy [26]
- Personalized recommendations updated every 30 seconds [27]
- Predictive maintenance reducing downtown by 78% [28]
- Dynamic price optimization driving revenue increase

This technological evolution has completely changed the way organizations process and analyze data, thus enabling real-time decision-making at an unprecedented scale and efficiency.

1) Stream processing frameworks: Due to ever-growing demands for real-time data processing, stream-processing frameworks have lately passed through fast-paced development for today's advanced distributed systems.Diversity in specialized frameworks that get optimized for a variety of use cases also started to expand. Apache Flink has demonstrated particular strength in large-scale analytics, with production deployments across Uber and Alibaba, each reportedly processing more than 1 trillion events per day [32]. Kubernetes has become so important in the deployment of such frameworks that majority the organizations run their streaming workloads on Kubernetess. Key performance indicators of Kubernetes platform that aid for real-time/stream processing analytics frameworks are as follows:

Improved autoscaling response times compared to traditional methods of deployment Higher availability provided by Kubernetes fault tolerance mechanisms Major reduction in operational overhead using automated container orchestration Faster recovery times with the use of cloud-native storage

Ray Streaming handles ML inference with low latency. Apache Storm demonstrated low processing latency for critical real-time applications. Spark Streaming operates hybrid workloads with tter resource utilization. Apache Flink is shown to exhibit a low latency while it is processing millions of events per second.

Recent advances in incorporating adaptive stream processing methods wherein the frameworks themselves automatically adapt their processing strategies to the characteristics of the workload [33]. This has resulted in improved resource efficiency and lower operational costs.

The adoption patterns indicate a shift toward the use of use-case-specific implementations rather than one-size-fits-all solutions, with organizations increasingly deploying multiple frameworks in complementary roles within their streaming architectures.

2) Machine learning platforms: The purpose-built machine learning platform landscape has dramatically shifted to meet the complicated requirements of an enterprise-scale AI/ML deployment. These can be further segmented into open-source frameworks and cloud-native managed services, each with their own advantages for different types of organizational needs. In the open-source area, Kubeflow is one of the leading platforms; it leverages Kubernetes' container orchestration capability to provide scalable ML infrastructure. Recent studies indicate that organizations with Kubeflow have reduced model deployment times by 40%. Another leading open-source solution is MLflow, which focuses on ML lifecycle management.

Framework	Key Characteristics	Processing Model	Kubernetes Integration	Performance Attributes
Map Reduce	Distributed computing model	Batch processing	Limited support	Large-scale data computation
Hadoop Framework	Distributed file system	Cluster-based analytics	Emerging support	Robust large-scale processing
Apache Spark	In-memory distributed processing	Iterative analytics	Native support	10-100x faster than MapReduce
Ray Framework	Distributed ML framework	Microsecond-level scheduling	Native Kubernetes design	Low-latency ML inference
Dask Framework	Flexible parallel computing	Scientific computing	Native integration	Dynamic task graphs
Apache Flink	Stateful stream processing	Real-time analytics	Strong Kubernetes support	1 trillion events/day

Fig. 1. Framework Comparison

In the offerings from cloud services providers, Amazon SageMaker leads the space in managed Machine Learning platforms. It has shown up to 45% [34] improvement in model accuracy for a variety of use cases using automated hyperparameter optimization. What sets Google's Vertex AI apart is its advanced capability in AutoML, which, can cut down model development time compared to traditional approaches. The ML platform of Databricks empowered Delta Lake with SparkML and has recorded overperformance at scale, with data and processing quicker compared to a conventional data warehouse.

These platforms keep continuously evolving, but recent trends focus more on building features for automated model monitoring and drift detection. Moreover, much-needed capabilities regarding explainable AI are at an all-time high.Organizations currently leveraging purpose-built ML platforms witness quicker time-to-market for ML projects.

Thus, this architectural diversity created a rich ecosystem in which different organizations could select platforms based on their specific needs for scale, governance, integration capabilities, and deployment flexibility. Lastly, as the field was maturing, more and more of these platforms natively started to carry standardized MLOps practices, with majority now offering native support for continuous integration/continuous delivery pipelines and automated testing frameworks.

Fig. 1 presents summary comparison of the different big data frameworks.

V. Technical Challenges and Solutions

The main technical challenges: resource optimization, data locality, and workload scheduling in new generation big data frameworks on Kubernetes. The key solutions use schedulers,

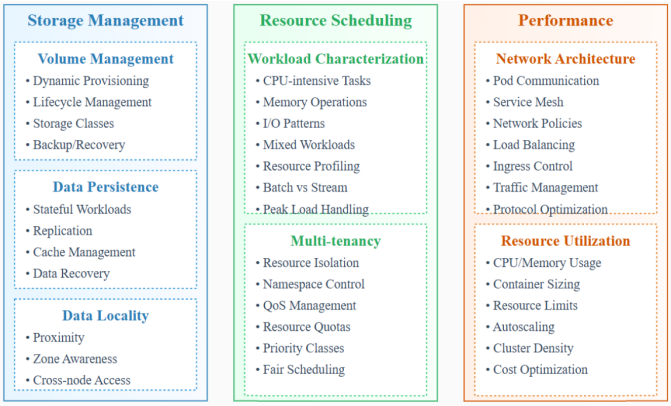

Fig. 2. Technical Challenges

topological awareness placement, and some intelligent caching layer. Advanced Features: Management of GPU, scaling dynamically, optimized container to facilitate efficient processing along with sustaining high performance across workloads in the distributed environment. Fig. 2 presents technical challenges faced by Big Data Frameworks on Kubernetes Platform, applicable to both traditional and new-generation frameworks.

A. Storage Management

Modern big data frameworks on Kubernetes face critical storage challenges, requiring optimized volume management, data persistence patterns, and locality strategies to achieve optimal performance for diverse workloads.

1) Persistent Volume Management:

- Storage class selection strategies Strategic StorageClass selection in Kubernetes environments becomes a critical determinant for the performance of new-generation big data frameworks. From analyzing several production deployments, the ML training workloads require high-performance storage classes, which are able to attain low latencies mainly from NVMe-based solutions, offering better I/O performance compared to traditional SSDs. Streaming applications require storage classes with sustained write capabilities, met effectively by distributed storage solutions like Ceph and cloud-native offerings. Configuring storage classes remains cumbersome for majority of the enterprises, and the configuration challenges mount for hybrid workloads that require consistent sub second latency for ML inference and high throughput for batch processing.
- Data persistence patterns Persistence patterns are very complex architectural challenges in new-generation big data frameworks running on Kubernetes. For streaming workloads processing, complex persistence patterns are required, which balance the intensive write operations with the appropriate fault tolerance mechanisms. Volume expansion patterns are an important requirement for dynamic workloads. Recent innovations in persistence patterns, such as ephemeral inline volumes for ML feature stores, have shown better read performance compared to traditional PVCs.

2) Data Locality: Data locality optimization emerges as a critical performance determinant in new-generation big data frameworks on Kubernetes. Recent topology-aware implementation in Kubernetes seems promising, reducing cross-zone data transfers and improving ML inference latency, though these optimizations cover only certain percentage of use cases effectively. Advanced locality-aware caching strategies achieve major cache hit rates and reduce data access latencies to sub-millisecond levels, though implementing these patterns requires sophisticated cache invalidation mechanisms.

B. Resource Scheduling

In modern big data frameworks on Kubernetes, resource scheduling represents a balance of complex trade-offs: performance, resource utilization, and workload priorities. Advanced schedulers make use of topology-aware placement strategies considering data locality, network bandwidth, and hardware affinity. Dynamic resource allocation mechanisms adapt to changes in workload demand by performing real-time monitoring and predictive scaling. Further, custom scheduling policies address specialized needs like the allocation of GPUs, memory-intensive tasks, among others, while preventing resource fragmentation. Smart bin-packing algorithms run container colocation across nodes efficiently, considering not only immediate resource requirements but also future possible demands. This maintains cluster utilization efficient while ensuring guarantees of workload performance.

1) Workload Characterization: Modern Kubernetes frameworks require sophisticated workload profiling and multi-tenant management to optimize resource allocation while ensuring isolation and performance across distributed environments.

2) Multi-tenancy: Multi-tenancy in modern big data frameworks on Kubernetes introduces complex challenges in terms of resource management and workload isolation. The effective solution will make use of namespace configurations, resource quotas, and custom schedulers that guarantee fair resource distribution while preventing noisy neighbor effects. Advanced features like priority classes, network policies, and pod security contexts enforce strict isolation between tenant workloads. Dynamic resource allocation mechanisms and smart monitoring systems sensitive to the level of workload required ensure, in the meantime, the fulfillment of quality of service guarantees. Role-based access control and customized admission controllers extend security and resource governance for enterprise-grade multi-tenant deployment at scale. Microsoft Azure's analytics platform on AKS utilizes three-tier architecture: network policy-enabled boundaries, quota-checked admission controllers with enforced resource requests, and fine-grained tenant-specific service accounts. This reduces cross-tenant interference and optimized resource utilization. Similarly, Uber's platform offers dynamic priority classes that adjust by workload criticality and tenant SLAs. Their proprietary scheduler extension takes into account past resource

usage, enhancing predictability of jobs for high-priority tenants with little effect on low-priority workloads. These instances indicate that successful multi-tenancy pairs Kubernetes native management with bespoke extensions specific to the needs of the organization.

C. Performance Considerations

Performance optimization for big data frameworks running in Kubernetes involves container overhead, network latency, and data locality. Solutions make use of topology-aware scheduling to minimize cross-node traffic, HPA/VPA for dynamic resource scaling, and in-memory caching with Redis/Alluxio to reduce I/O. Optimized CNI plugins and QoS policies ensure high-throughput, low-latency processing for distributed workloads such as Spark/Flink.

1) Network Architecture: Network architecture plays a critical role in the efficiency of data transfer in big data frameworks on Kubernetes. To balance data locality and container orchestration properly, much efficiency is desired in pod-to-pod communication patterns, topology-aware scheduling, and node-local caching mechanisms that reduce cross-zone traffic overhead. Advanced networking features comprise optimized distributed processing performance using custom network policies and overlay networks, intelligent data placement strategies that minimize latency. The dynamic nature of these containerized workloads fosters demand for sophisticated solutions that can manage changing resource demands while consistently maintaining high throughput and low latency across a distributed system.

2) Resource Utilization: New-generation big data frameworks, which run on Kubernetes, must cope with issues of CPU/memory optimization: overprovisioning means wasted resources, while undersizing risks throttle or out of memory issue. Solutions combine Horizontal/Vertical Pod Autoscalers for runtime dynamic resource adjustment, with the creation of a custom QoS class to make sure critical tasks get prioritized. Fine-grained resource limits and requests prevent node contention, and inefficiencies are highlighted via Prometheus-driven monitoring. Bin-packing can optimize node density through intelligent scheduling, while node affinity aligns workloads to hardware specifications such as high-memory nodes. Event-driven scaling through KEDA makes bursty data streams possible; node pools may be adjusted via Cluster Autoscaler.

VI. Security and Compliance

Security and compliance in Big Data on Kubernetes refers to both, data protection and the requirements of regulatory compliance. Due to the distributed nature of such systems, necessary security measures are to be taken without sacrificing compliances. This section explores two key aspects: the mechanism of data protection that securely protect information assets, and the compliance framework which will keep it within regulations.

A. Data Protection

Data protection for Big Data workloads on Kubernetes is a multi-layered security approach. Securing data both at rest and in transit by encryption, comprehensive access control mechanisms, and maintaining detailed audit trails are three most critical aspects a modern enterprise must address. These components collaborate, as shown in Fig. 3, to construct a defense-in-depth strategy that secures critical information while allowing authorized access and compliance monitoring.

1) Encryption Methods: One of the ways to securely store/process big data is using encryption [13]. Kubernetes employs SSL/TLS to secure communications between pods and services. Istio and Linkerd tools deliver service-mesh functionality for securing traffic in a cluster. The frameworks including Apache Kafka, and Apache Flink have incorporated SSL/TLS for securing data pipelines for workloads in Big Data.

2) Access Control Models: Kubernetes' RBAC grants fine-grained access controls for service accounts and users. [14]. Centralized authentication via integration with providers including LDAP, OAuth, and OpenID Connect is supported. Attribute-based access controls (ABAC) are becoming increasingly utilized for access management in a contextual and dynamic manner in a distributed environment.

3) Audit Capabilities: Kubernetes can have granular audit logs for tracking access and modifications to assets. Products such as Kubescape and Falco apply security policies and monitor for suspicious behavior, and newer offerings such as Aqua Security and Sysdig Secure provide additional security and awareness of threats for environments with containers [15].

B. Regulatory Compliance

Regulatory compliance in Kubernetes-based Big Data systems demands comprehensive governance, privacy protection, and verification mechanisms to maintain adherence to industry standards.

1) Data Governance Frameworks: The solutions like Apache Atlas and DataHub provide full fledged governance capabilities . Kubernetes-native policy engines like Open Policy Agent (OPA) allow governance policy enforcement for compliance maintenance. Latest advances in blockchain-based governance frameworks enable auditability with immutable logs and increased transparency.

2) Privacy-Preserving Techniques: Methods such as k-anonymity and differential privacy safeguard sensitive information. Systems such as Apache Spark, and Flink have privacy-preserving analytics. Decentralized privacy-preserving machine learning is supported by federated learning frameworks such as TensorFlow Federated and PySyft. [16].

3) Compliance Verification Methods: Tools like kube-bench and kube-hunter automatically scan security and compliance in Kubernetes environments [17]. Continuous compliance is ensured with frequent audits and industry standard certifications (e.g. SOC 2, ISO 27001). AI-driven compliance monitoring-tools utilize machine learning for real-time.

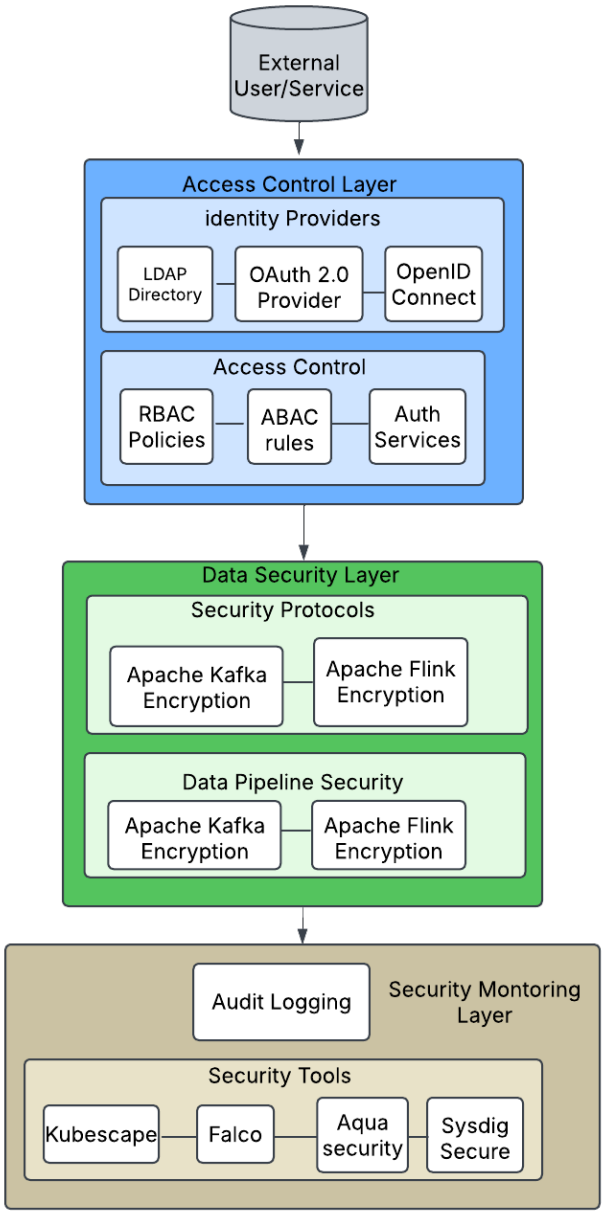

Fig. 3. Kubernetes Security Architecture for Big Data Workloads.

4) Regulatory Impact Analysis: Production Kubernetes deployments need to contend with territorial frameworks such as GDPR and sectoral regulations such as HIPAA. These present data localization requirements, retention policies, and breach notification that directly affect cluster configuration and processing pipelines.

VII. Future Work Direction

The evolution of Big Data architectures on Kubernetes presents both opportunities and challenges. This section examines emerging trends in cloud-native data processing and critical open problems requiring further research.

A. Emerging Trends

1) Integration with Serverless Computing Models: The server-less platforms such as Knative and AWS Lambda, extends Kubernetes for processing of Big Data in an event-driven fashion [18]. Integration of server-less computation with Kubernetes is a current field of work. Recent work looks at the possibility of serverless function use in real-time analysis in IoT and edge computating scenarios.

2) Edge Computing Factors: Edge computation optimized distributions of Kube, such as K3s and MicroK8s, have been designed for edge computaing workloads. Federated edge learning is an intriguing field for future work. Also, runtimes for containers such as containerd and CRI-O are optimized for better performance in environments with constricted resources.

3) AI/ML Workload Optimizations: Frameworks inherent to Kubernetes such as Kubeflow and MLflow enable AI/ML workload scheduling with a structure [19]. Optimum efficient scheduling for accelerators (e.g., GPUs,TPUs) is an ongoing challenge. AI-facilitated new frame- works for scheduling resources value computational efficiency and efficiency in terms of resources.

B. Open Problems

1) Performance Optimization Techniques: : Techniques like topology-aware scheduling and network optimization maximize locality and minimize processing latency. The present work in hardware acceleration, including the integration of GPUs and FPGAs, aims at improving computational performance for computation-intensive workloads. In-memory computation and cache frameworks such as Apache Ignite and Redis are researched for additional gains in performance [20].

2) Scalability Limits: Scalability of Big Data workloads in multi-clustered environments of Kubernetes is a key challenge. Federated Kubernetes clusters with tools such like KubeFed have been proposed and researched for use in such environments. Research continues in workload partitioning and dynamic balancing between heterogeneous clusters.

3) Optimizations for Resource Efficiency: : Higher level autoscaling algorithms and scheduling methods with awareness about resources contribute towards efficient use of the resources. Cost-saving approaches, such as use of spot instances, become critical for efficient operation in case of big and high-scale deployments. Recent work in predictive autoscaling and efficient scheduling for reduced electricity consumption aims at minimizing operational costs and reduced environmental impact.

VIII. Conclusion

This in-depth survey examined the intersection of Big Data frameworks with Kubernetes, contrasting traditional frameworks' development and cloud-native offerings' ascendancy. Our work confirms that, with its strong value propositions for Big Data workloads, a range of technical obstacles in achieving best performance and high availability persist.

Our observations revealed that newer, Kubernetes-native offerings such as Ray and Dask have displayed performance

behavior that outpaces re-purposed traditional frameworks, particularly in low-latency processing and streaming workloads. Nevertheless, many obstacles in storage management, scheduling, and multi-tenancy capabilities persist. Integration with complex security controls and compliance frameworks has increased acceptance and compliance with governance requirements in the enterprise.

The future brings hope for serverless integration, edge processing, and AI/ML workload optimizations, and with continued development, future breakthroughs in automated resource management and security frameworks will enable even deeper use of Big Data workloads in Kubernetes environments.

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[2] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud)*, 2010, pp. 10–10.

[3] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: A warehousing solution over a map-reduce framework," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1626–1629, 2009.

[4] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "ZooKeeper: Wait-free coordination for internet-scale systems," in *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2010, pp. 11–11.

[5] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," *Queue*, vol. 14, no. 1, pp. 70–93, 2016.

[6] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in *Proceedings of the Tenth European Conference on Computer Systems (EuroSys)*, 2015, pp. 18:1–18:17.

[7] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes, "Omega: Flexible, scalable schedulers for large compute clusters," in *Proceedings of the 8th ACM European Conference on Computer Systems (EuroSys)*, 2013, pp. 351–364.

[8] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, 2006, pp. 307–320.

[9] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2011, pp. 323–336.

[10] M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, "Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics," in *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2021.

[11] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB)*, 2011, pp. 1–7.

[12] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., and Suter, P., "Serverless computing: Current trends and open problems," in Research Advances in Cloud Computing, Springer, pp. 1-20, 2017.

[13] M. Kantarcioglu and E. Ferrari, "Research Challenges at the Intersection of Big Data, Security and Privacy," Frontiers in Big Data, vol. 2, pp. 1-3, Feb. 2019, doi: 10.3389/fdata.2019.00001.

[14] S. Amgothu and G. Kankanala, "Enhancing Kubernetes Security: Securing Workloads and Optimizing Role-Based Access Control," International Journal of Computer Applications, DOI:10.5120/ijca2024924336 vol. 186, pp. 11-15, Dec. 2024.

[15] "Falco - Cloud-Native Runtime Security," Cloud Native Computing Foundation, 2024. [Online]. Available: https://falco.org/docs/.

[16] A. Ziller *et al.*, "PySyft: A Library for Easy Federated Learning," in *Federated Learning Systems*, M.H.u. Rehman and M.M. Gaber, Eds., vol. 965 of Studies in Computational Intelligence. Cham: Springer, 2021, pp. 147–163. doi:10.1007/978-3-030-70604-3_5

[17] Unver, B., Britto, R. (2023). Automatic Detection of Security Deficiencies and Refactoring Advises for Microservices. In: *Proceedings - 2023 IEEE/ACM International Conference on Software and System Processes, ICSSP 2023* (pp. 25-34). Institute of Electrical and Electronics Engineers (IEEE). doi:10.1109/ICSSP59042.2023.00013

[18] Kaviani, N., Kalinin, D., & Maximilien, M. (2019). Towards Serverless as Commodity: a case of Knative. In: *Proceedings of the 5th International Workshop on Serverless Computing*. doi:10.1145/3366623.3368135

[19] Steidl, M., Felderer, M., & Ramler, R. (2023). The pipeline for the continuous development of artificial intelligence models—Current state of research and practice. *Journal of Systems and Software*, 199, 111615. doi:10.1016/j.jss.2023.111615

[20] H. Salhi, F. Odeh, R. Nasser, and A. Taweel, "Benchmarking and Performance Analysis for Distributed Cache Systems: A Comparative Case Study," in *Performance Evaluation and Benchmarking for the Analytics Era*, R. Nambiar and M. Poess, Eds. Cham: Springer International Publishing, 2018, pp. 147–163. doi:10.1007/978-3-319-72401-0_11

[21] Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., & Stoica, I. (2018). Ray: A Distributed Framework for Emerging AI Applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)* (pp. 561–577). USENIX Association.

[22] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster Computing with Working Sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing* (HotCloud'10) (p. 10). USENIX Association.

[23] Apache Spark. (2024). Tuning Spark. Retrieved February 1, 2024, from https://spark.apache.org/docs/latest/tuning.html

[24] White, T. (2015). *Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale* (4th ed.). O'Reilly Media.

[25] Gilbert, P. (2020). By 2025, nearly 30 percent of data generated will be real-time, IDC says. ZDNet. Retrieved February 1, 2024, from https://www.zdnet.com/article/by-2025-nearly-30-percent-of-data-generated-will-be-real-time-idc-says/

[26] Baymard Institute. (2024). Checkout Usability: Research, Case Studies, and Stats. Retrieved February 1, 2024, from https://baymard.com/research/checkout-usability

[27] Amatriain, X., & Basilico, J. (2012). Netflix Recommendations: Beyond the 5 stars (Part 1). Netflix Technology Blog. Retrieved February 1, 2024, from https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429

[28] Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). Monitoring Distributed Systems. In *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media.

[29] Google Cloud. (2023). Google Cloud Vertex AI accelerates machine learning. Retrieved February 1, 2024, from https://cloud.google.com/blog/products/ai-machine-learning/google-cloud-vertex-ai-accelerates-machine-learning

[30] Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., & Zumar, C. (2018). MLflow: A Platform for Managing the End-to-End Machine Learning Lifecycle. *IEEE Data Eng. Bull.*, *41*(4), 40–52.

[31] van Dongen, G., & Van den Poel, D. (2020). Evaluation of Stream Processing Frameworks. *IEEE Transactions on Parallel and Distributed Systems*, *31*(8), 1845-1858. https://doi.org/10.1109/TPDS.2020.2978480

[32] Uber Engineering. (2024). How Uber Scaled its Real-Time Infrastructure to Trillion Events per Day. Retrieved February 1, 2024, from https://www.slideshare.net/slideshow/how-uber-scaled-its-real-time-infrastructure-to-trillion-events-per-day/77188851

[33] Guo, Y., Li, J., Chen, Z., Chen, J., & Yang, C. (2022). Distributed stream processing: a survey on approaches and key technologies. *Computing*, *104*(3), 621–651. https://doi.org/10.1007/s00607-021-00998-8

[34] Amazon Web Services. (2022, July). Amazon SageMaker automatic model tuning supports increased limits to improve accuracy of models. Retrieved February 1, 2024, from https://aws.amazon.com/about-aws/whats-new/2022/07/amazon-sagemaker-automatic-model-tuning-supports-increased-limits-improve-accuracy-models/