Beyond Infrastructure:

Conductor's Implementation of Onehouse for Scalable Data Lakehouse Solutions

ONEHOUSE × conductor

Contents

conductor

Conductor is the world's leading organic marketing platform, helping businesses accelerate search traffic, digital growth, and revenue. Their technology empowers marketers to create powerful content that drives high-quality traffic while providing comprehensive organic performance tracking. Conductor serves forward-thinking global and emerging enterprise brands including Citibank, Visa, and Casper.

When you're operating at this scale, data infrastructure becomes crucial to success. This case study delves into Conductor's journey in handling vast amounts of data efficiently and cost-effectively through the implementation of Onehouse's Universal Data Lakehouse platform.

Challenges: Closed System Limitations and the Technical Debt Reality

Before implementing Onehouse, Conductor faced a complex web of data management challenges that were hampering their ability to scale effectively. At the core of their problems was a fragmented data architecture, with terabytes of data scattered across multiple services including Amazon Aurora, MySQL, and Snowflake. This dispersion led to significant data duplication and made it increasingly difficult to maintain a single source of truth for their operations.

The **infrastructure management** burden was particularly heavy on their team. They were attempting to manage their own Apache Spark and Amazon EMR clusters while implementing Apache Hudi, but their efforts to get Hudi working contributed to frequent out-of-memory errors in Spark. The team also struggled with poor system observability throughout these challenges. When problems arose, such as Spark failures, the team had to navigate complex organizational procedures just to access logs, requiring time-consuming coordination with infrastructure teams. A significant amount of time was spent trying to fine-tune their Hudi implementation, experimenting with various configuration settings to find the optimal combination.

Performance issues plagued their system as they struggled to implement an effective data modeling strategy with Apache Hudi. Their Redshift queries showed wildly inconsistent performance—cold queries took 8 seconds while cached queries completed in 200 milliseconds. The unpredictable nature of cache invalidation, combined with their evolving data model, made it impossible to maintain consistent performance levels. This was especially problematic for their user-facing analytics, where queries could take 20-30 seconds—far from their target of sub-3 second response times. Their challenges with Hudi implementation and data modeling strategy led to inefficient query patterns that significantly impacted performance.

Scalability presented another set of critical challenges. Conductor needed to handle concurrent queries from hundreds, potentially thousands of users while processing tens of terabytes of data. Their weekly data collection process, combined with their nascent big data modeling approach, created significant performance inconsistencies across their system. This made it particularly difficult to support both batch and real-time data processing needs while maintaining reliable query performance.

The cumulative effect of these challenges was a significant drain on development time and resources. The team found themselves spending excessive time on infrastructure management rather than focusing on core business features. They lacked the specialized expertise needed to optimize their data lake technologies effectively, and the constant need to coordinate across multiple teams for infrastructure access slowed their development velocity considerably.

Solution: The Onehouse Integration & Partnership

Conductor implemented Onehouse as their data lakehouse solution, which provided them with a Spark Kubernetes cluster running in their own VPC. This managed solution handled their Hudi implementation on S3, providing automatic table services, AWS Glue Data Catalog sync capabilities, and comprehensive reporting functionality.

Their query engine evaluation process was extensive. While Amazon Athena proved suitable for exploratory work and AI feature extraction, they found limitations with other solutions. With Redshift, they encountered inconsistent performance—cold queries would take 8 seconds while cached queries ran in 200 milliseconds, but they couldn't predict when data would be ejected from cache. While Conductor uses Snowflake for certain workloads, they ruled it out for these specific reports after a POC revealed it didn't offer better performance than alternatives. Additionally, Snowflake's cost scaling at larger warehouse sizes was a significant concern.

Guided by Onehouse, Conductor systematically evaluated StarRocks configurations to optimize their data architecture. Through careful testing, they determined that StarRocks' shared-nothing architecture on Amazon EBS was the best solution for their specific requirements. While alternative approaches like direct Hudi integration and shared-data architecture were explored, Conductor's commitment to delivering exceptional performance led them to choose the EBS-based solution. This configuration successfully achieved their goal of sub-second query performance for complex analytics, making it worth the additional investment of approximately $100 per terabyte.

Their data modeling strategy evolved significantly based on each tool's capabilities. For one particular dataset, when using Hudi, they had to implement a complex three-level partitioning approach to handle data distribution:

- First level: Year
- Second level: Week number (1-52)
- Third level: A derived column created by hashing string values into a 16-byte number and using modulo 50 to distribute data evenly across buckets

However, they later discovered that StarRocks supports this type of bucketing natively, eliminating the need for such complex partitioning schemes. This effort proved that their data modeling approach could be simplified by leveraging StarRocks' built-in capabilities rather than implementing custom solutions in Hudi.

For data ingestion into Hudi via Onehouse, Conductor discovered that using S3 with large Parquet files could be more cost-effective than their current Kafka and Flink setup (potentially reducing monthly costs from $1,000-2,000 to approximately $5) and potentially faster than using a 50-partition Kafka topic. While they currently maintain their Kafka-based system with optimized costs, this Parquet-based approach is being considered as a recommendation for new workloads.

Throughout the implementation, Conductor benefited significantly from Onehouse's support team, who provided not just technical assistance but also education about best practices for data modeling and partitioning. The ability to implement custom transformations became straightforward—they could simply package a Java file and insert it into the Onehouse interface without managing the underlying Spark infrastructure.

The improved system provided better observability through the user interface, allowing them to view logs directly instead of coordinating across multiple teams for access. This streamlined infrastructure management allowed Conductor to focus more on their core business of website monitoring, content creation, and SEO services, rather than spending time configuring and maintaining data infrastructure.

Results: From Maintenance to Innovation

The implementation of Onehouse marked a transformative moment for Conductor's data infrastructure. As Emil Emilov, Conductor's Principal Engineer, described it, Onehouse was "a godsend" that provided them with a functioning data lake "overnight," immediately addressing most of their needs. **The most immediate impact came from no longer having to manage their own Spark and EMR infrastructure—instead of spending time configuring clusters and troubleshooting memory issues, they could rely on a managed infrastructure that handled the complexity of Spark operations while maintaining full visibility through comprehensive logging and monitoring.**

Conductor saw **significant improvements in their query performance and data accessibility.** Through Onehouse's automated table services and optimizations, they could begin querying their data as soon as tables were synced. The platform's support for Apache Hudi, combined with guidance from Onehouse on proper partitioning strategies, helped them address their previous issues with partition skew and query latency. This was particularly crucial for their user-facing analytics, where their original 20-30 second query times were reduced to 5-7 seconds through optimizations, though they continue to work towards their target of sub-3 second response times for optimal user experience.

Cost management and operational efficiency also improved significantly. By running in Conductor's own cloud infrastructure with configurable usage limits, they could better control their resource utilization and costs. A particularly significant cost reduction could come from migrating from Kafka/Flink to S3-based ingestion, as discovered in the team's POCs, dramatically reducing monthly costs from $1,000-$2,000 to approximately $5. Onehouse's auto-scaling capabilities provided dynamic resource adjustment based on workload demands. Furthermore, the implementation of custom transformations became straightforward—the team could simply package a Java file and insert it into the Onehouse interface without having to manage the underlying Spark infrastructure. The improved observability through Onehouse's user interface meant they no longer had to coordinate across multiple teams just to access basic system logs.

Perhaps most importantly, Onehouse enabled Conductor to shift their focus from infrastructure maintenance to actual business value. Instead of spending time tweaking Spark configurations and managing clusters, they could concentrate on improving their website monitoring, content creation, and SEO services—their core business offerings. The platform's ability to handle their complex data requirements while remaining cost-effective and manageable proved crucial in supporting Conductor's continued growth and evolution.

Looking forward, Conductor plans to keep evolving their data platform with several technical initiatives: query performance optimization focusing on join operations; additional cost reduction through improved data scanning efficiency; exploration of StarRocks 3.3's inverted full-text index capabilities; integration with Hudi 1.0's column stats index for enhanced query performance; and retrying StarRocks native S3.

The Engineering Takeaway

Conductor's experience highlights a crucial insight for data engineers: sometimes the most sophisticated technical solution involves knowing when to leverage specialized expertise. By maintaining architectural control while delegating infrastructure complexity to domain experts, they are achieving better performance and reliability than their previous self-managed approach.

As Emil notes, "With Onehouse, there's a lot of things we don't have to figure out anymore. If it's managed for you and configured and it's cost effective, you're better off using it. There is never enough time and never enough money." For data teams facing similar infrastructure modernization challenges, Conductor's journey offers a compelling blueprint for achieving high-performance, scalable data architecture without getting lost in infrastructure complexity.

"With Onehouse, there's a lot of things we don't have to figure out anymore. If it's managed for you and configured and it's cost effective, you're better off using it."

Emil Emilov,
Conductor's Principal Engineer

Experience the Universal Data Lakehouse Advantage:

onehouse.ai/schedule-a-test-drive