The Delta Lake Series Lakehouse

Combining the best elements of data lakes and data warehouses

databricks

What's inside?

The Delta Lake Series of eBooks is published by Databricks to help leaders and practitioners understand the full capabilities of Delta Lake as well as the landscape it resides in. This eBook, **The Delta Lake Series — Lakehouse**, focuses on lakehouse.

What's next?

After reading this eBook, you'll not only understand what Delta Lake offers, but you'll also understand how its features result in substantial performance improvements.

Here's what you'll find inside

Introduction
What is Delta Lake?

Chapter 01 What Is a Lakehouse?

Chapter 02 Diving Deep Into the Inner Workings of the Lakehouse and Delta Lake

Chapter 03 Understanding Delta Engine

What is Delta Lake?

Delta Lake is a unified data management system that brings data reliability and fast analytics to cloud data lakes. Delta Lake runs on top of existing data lakes and is fully compatible with Apache Spark™ APIs.

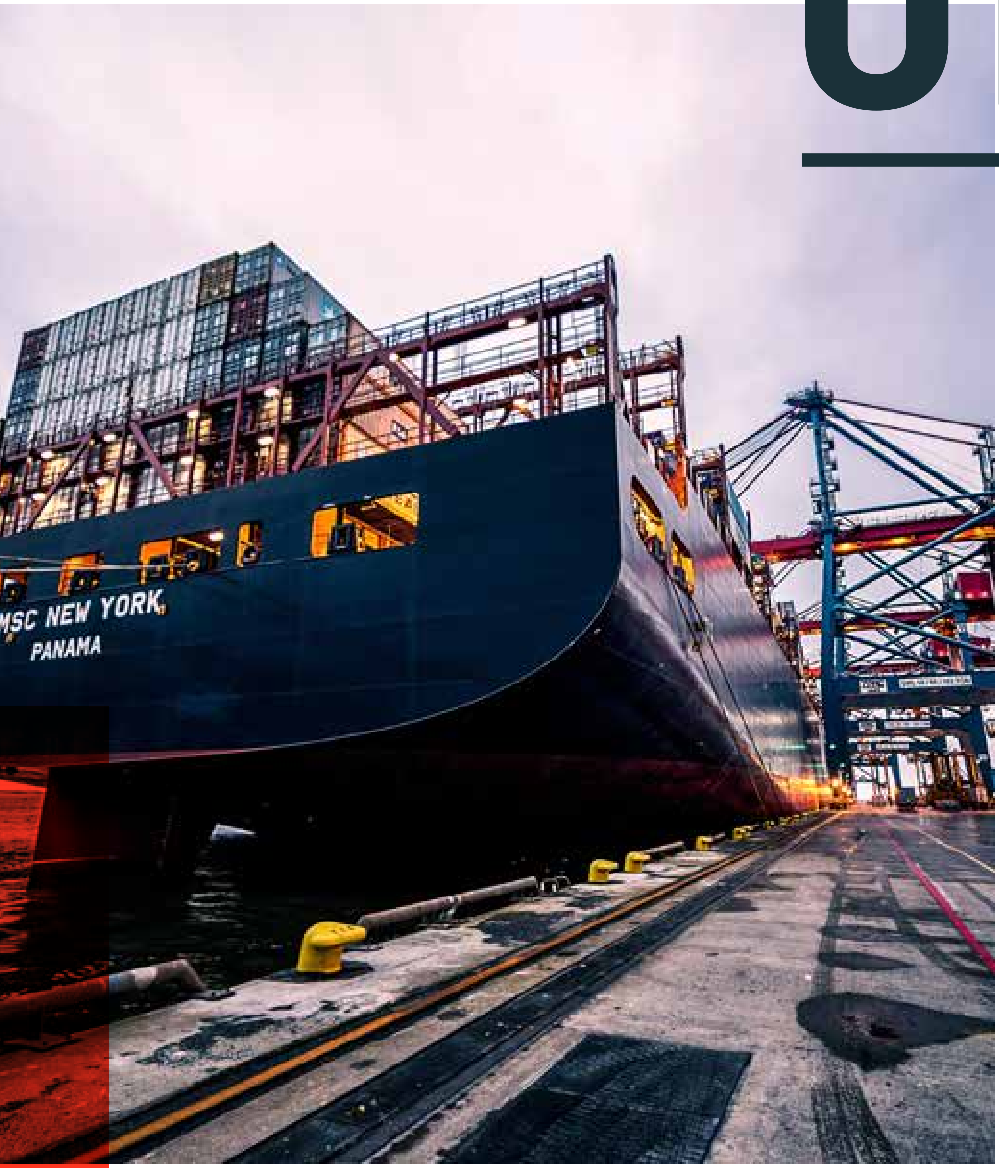
At Databricks, we've seen how Delta Lake can bring reliability, performance and lifecycle management to data lakes. Our customers have found that Delta Lake solves for challenges around malformed data ingestion, difficulties deleting data for compliance, or issues modifying data for data capture.

With Delta Lake, you can accelerate the velocity that high-quality data can get into your data lake and the rate that teams can leverage that data with a secure and scalable cloud service.

What Is a Lakehouse?

CHAPTER 01

01 What Is a Lakehouse?



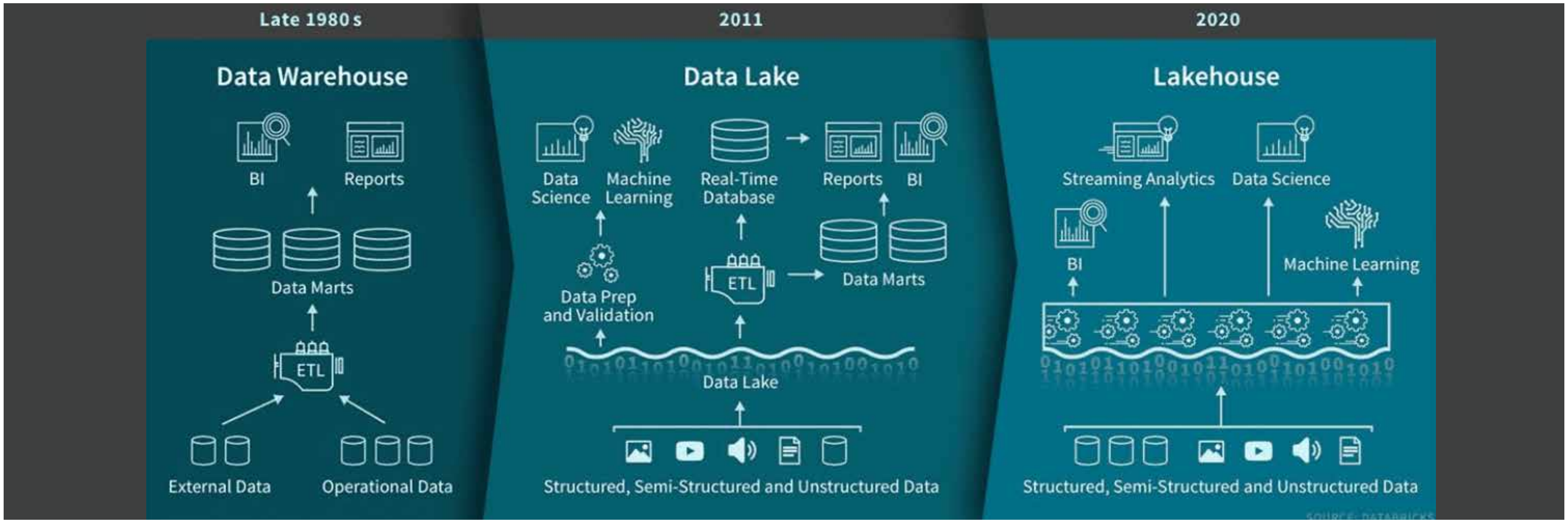
Over the past few years at Databricks, we've seen a new data management architecture that emerged independently across many customers and use cases: the **lakehouse.** In this chapter, we'll describe this new architecture and its advantages over previous approaches.

Data warehouses have a long history of decision support and business intelligence applications. Since its inception in the late 1980s, data warehouse technology continued to evolve and MPP architectures led to systems that were able to handle larger data sizes.

But while warehouses were great for structured data, a lot of modern enterprises have to deal with unstructured data, semi-structured data, and data with high variety, velocity and volume. Data warehouses are not suited for many of these use cases, and they are certainly not the most cost-efficient.

As companies began to collect large amounts of data from many different sources, architects began envisioning a single system to house data for many different analytic products and workloads.

About a decade ago, companies began building data lakes — repositories for raw data in a variety of formats. While suitable for storing data, data lakes lack some critical features: They do not support transactions, they do not enforce data quality, and their lack of consistency / isolation makes it almost impossible to mix appends and reads,



and batch and streaming jobs. For these reasons, many of the promises of data lakes have not materialized and, in many cases, lead to a loss of many of the benefits of data warehouses.

The need for a flexible, high-performance system hasn't abated. Companies require systems for diverse data applications including SQL analytics, real-time monitoring, data science and machine learning. Most of the recent advances in AI have been in better models to process unstructured data (text, images, video, audio), but these are precisely the types of data that a data warehouse is not optimized for.

A common approach is to use multiple systems — a data lake, several data warehouses, and other specialized systems such as streaming, time-series, graph and image databases. Having a multitude of systems introduces complexity and, more importantly, introduces delay as data professionals invariably need to move or copy data between different systems.

A lakehouse combines the best elements of data lakes and data warehouses

A lakehouse is a new data architecture that combines the best elements of data lakes and data warehouses.

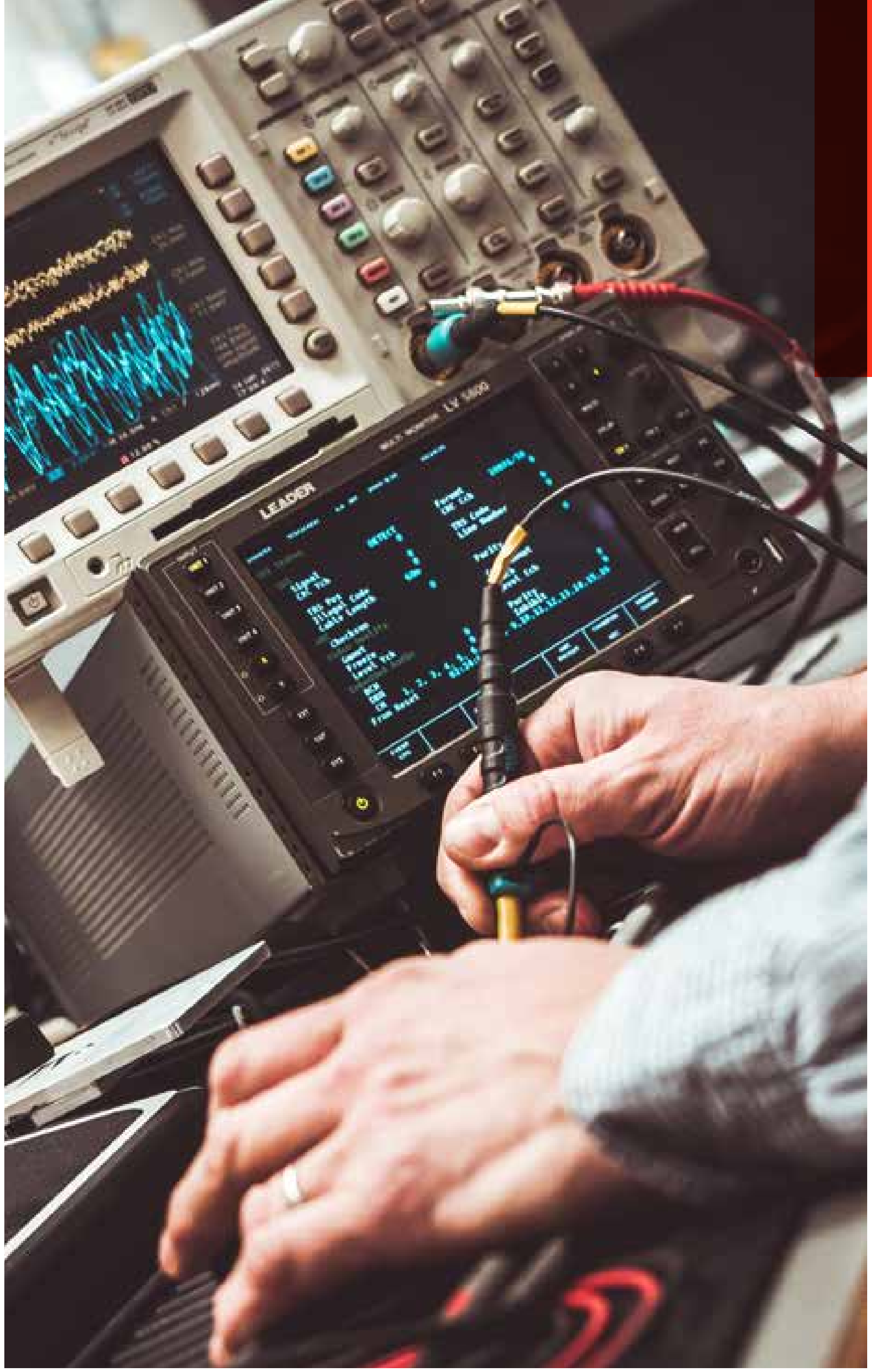
Lakehouses are enabled by a new system design: implementing similar data structures and data management features to those in a data warehouse, directly on the kind of low-cost storage used for data lakes. They are what you would get if you had to redesign data warehouses in the modern world, now that cheap and highly reliable storage (in the form of object stores) are available.

A lakehouse has the following key features:

- **Transaction support:** In an enterprise lakehouse, many data pipelines will often be reading and writing data concurrently. Support for ACID transactions ensures consistency as multiple parties concurrently read or write data, typically using SQL.

- **Schema enforcement and governance:** The lakehouse should have a way to support schema enforcement and evolution, supporting DW schema paradigms such as star/snowflake-schemas. The system should be able to reason about data integrity, and it should have robust governance and auditing mechanisms.
- **BI support:** Lakehouses enable using BI tools directly on the source data. This reduces staleness and improves recency, reduces latency and lowers the cost of having to operationalize two copies of the data in both a data lake and a warehouse.
- **Storage is decoupled from compute:** In practice, this means storage and compute use separate clusters, thus these systems are able to scale to many more concurrent users and larger data sizes. Some modern data warehouses also have this property.
- **Openness:** The storage formats they use are open and standardized, such as Parquet, and they provide an API so a variety of tools and engines, including machine learning and Python/R libraries, can efficiently access the data directly.
- **Support for diverse data types ranging from unstructured to structured data:** The lakehouse can be used to store, refine, analyze and access data types needed for many new data applications, including images, video, audio, semi-structured data, and text.
- **Support for diverse workloads:** Including data science, machine learning and SQL analytics. Multiple tools might be needed to support all these workloads, but they all rely on the same data repository.
- **End-to-end streaming:** Real-time reports are the norm in many enterprises. Support for streaming eliminates the need for separate systems dedicated to serving real-time data applications.

These are the key attributes of lakehouses. Enterprise-grade systems require additional features. Tools for security and access control are basic requirements. Data governance capabilities including auditing, retention and lineage have become essential particularly in light of recent privacy regulations. Tools that enable data discovery such as data catalogs and data usage metrics are also needed. With a lakehouse, such enterprise features only need to be implemented, tested and administered for a single system.



Read the research

Delta Lake: High-Performance ACID Table Storage Over Cloud Object Stores

Abstract

Cloud object stores such as Amazon S3 are some of the largest and most cost-effective storage systems on the planet, making the main attractive target to store large data warehouses and data lakes. Unfortunately, their implementation as key-value stores makes it difficult to achieve ACID transactions and high performance: Metadata operations, such as listing objects, are expensive, and consistency guarantees are limited. In this paper, we present Delta Lake, an open source ACID table storage layer over cloud object stores initially developed at Databricks. Delta Lake uses a transaction log that is compacted into Apache Parquet format to provide ACID properties, time travel, and significantly faster metadata operations for large tabular data sets (e.g., the ability to quickly search billions of table partitions for those relevant to a query). It also leverages this design to provide high-level features such as automatic data layout optimization, upserts, caching, and audit logs. Delta Lake tables can be accessed from Apache Spark, Hive, Presto, Redshift, and other systems. Delta Lake is deployed at thousands of Databricks customers that process exabytes of data per day, with the largest instances managing exabyte-scale data sets and billions of objects.

Authors: Michael Armbrust, Tathagata Das, Liwen Sun, Burak Yavuz, Shixiong Zhu, Mukul Murthy, Joseph Torres, Herman van Hövell, Adrian Ionescu, Alicja Łuszczak, Michał Szafra´nski, Xiao Li, Takuya Ueshin, Mostafa Mokhtar, Peter Boncz, Ali Ghodsi, Sameer Paranjpye, Pieter Senster, Reynold Xin, Matei Zaharia

Read the full research paper on the inner workings of the lakehouse.

Some early examples

The Databricks Unified Data Platform has the architectural features of a lakehouse. Microsoft's Azure Synapse Analytics service, which integrates with Azure Databricks, enables a similar lakehouse pattern. Other managed services such as BigQuery and Redshift Spectrum have some of the lakehouse features listed above, but they are examples that focus primarily on BI and other SQL applications.

Companies that want to build and implement their own systems have access to open source file formats (Delta Lake, Apache Iceberg, Apache Hudi) that are suitable for building a lakehouse.

Merging data lakes and data warehouses into a single system means that data teams can move faster as they are able to use data without needing to access multiple systems. The level of SQL support and integration with BI tools among these early lakehouses is generally sufficient for most enterprise data warehouses. Materialized views and stored procedures are available, but users may need to employ other mechanisms that aren't equivalent to those found in traditional data warehouses. The latter is particularly important for "lift and shift scenarios," which require systems that achieve semantics that are almost identical to those of older, commercial data warehouses.

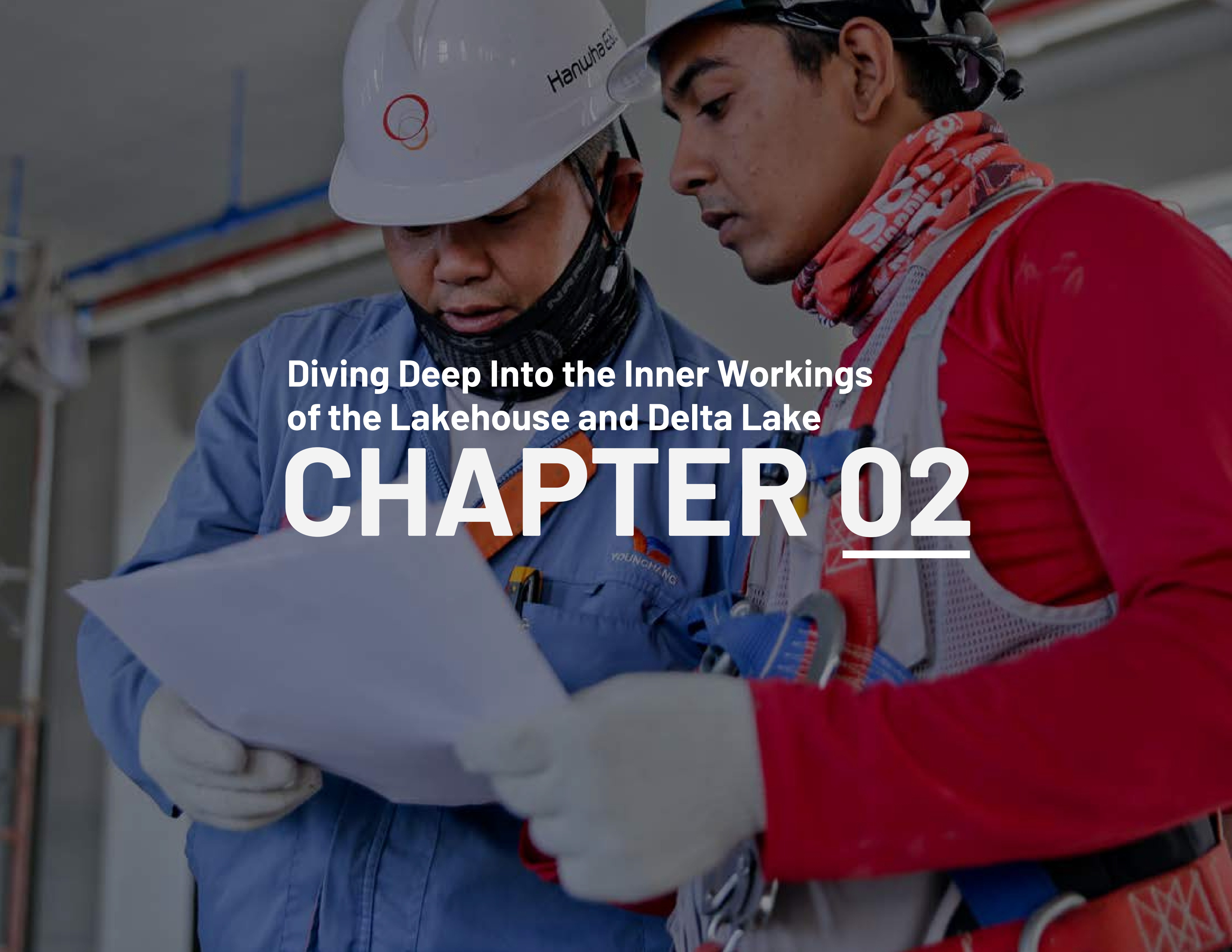
What about support for other types of data applications? Users of a lakehouse have access to a variety of standard tools (Apache Spark, Python, R, machine learning libraries) for non-BI workloads like data science and machine learning. Data exploration and refinement are standard for many analytic and data science applications. Delta Lake is designed to let users incrementally improve the quality of data in their lakehouse until it is ready for consumption.

A note about technical building blocks. While distributed file systems can be used for the storage layer, object stores are more commonly used in lakehouses. Object stores provide low-cost, highly available storage that excels at massively parallel reads — an essential requirement for modern data warehouses.

From BI to AI

The lakehouse is a new data management architecture that radically simplifies enterprise data infrastructure and accelerates innovation in an age when machine learning is poised to disrupt every industry. In the past, most of the data that went into a company's products or decision-making was structured data from operational systems, whereas today, many products incorporate AI in the form of computer vision and speech models, text mining and others. Why use a lakehouse instead of a data lake for AI? A lakehouse gives you data versioning, governance, security and ACID properties that are needed even for unstructured data.

Current lakehouses reduce cost, but their performance can still lag specialized systems (such as data warehouses) that have years of investments and real-world deployments behind them. Users may favor certain tools (BI tools, IDEs, notebooks) over others so lakehouses will also need to improve their UX and their connectors to popular tools so they can appeal to a variety of personas. These and other issues will be addressed as the technology continues to mature and develop. Over time, lakehouses will close these gaps while retaining the core properties of being simpler, more cost-efficient and more capable of serving diverse data applications.



Diving Deep Into the Inner Workings of the Lakehouse and Delta Lake

CHAPTER 02

02 Diving Deep Into the Inner Workings of the Lakehouse and Delta Lake

Databricks wrote a blog article that outlined how more and more enterprises are adopting the lakehouse pattern. The blog created a massive amount of interest from technology enthusiasts. While lots of people praised it as the next-generation data architecture, some people thought the lakehouse is the same thing as the data lake. Recently, several of our engineers and founders wrote a research paper that describes some of the core technological challenges and solutions that set the lakehouse architecture apart from the data lake, and it was accepted and published at the International Conference on Very Large Databases (VLDB) 2020. You can read the paper, "Delta Lake: High-Performance ACID Table Storage Over Cloud Object Stores," here.

Henry Ford is often credited with having said, "If I had asked people what they wanted, they would have said faster horses." The crux of this statement is that people often envision a better solution to a problem as an evolution of what they already know rather than rethinking the approach to the problem altogether. In the world of data storage, this pattern has been playing out for years. Vendors continue to try to reinvent the old horses of data warehouses and data lakes rather than seek a new solution.

More than a decade ago, the cloud opened a new frontier for data storage. Cloud object stores like Amazon S3 have become some of the largest and most cost-effective storage systems in the world, which makes them an attractive platform to store data warehouses and data lakes. However, their nature as key-value stores makes it difficult to achieve ACID transactions that many organizations require. Also, performance is hampered by expensive metadata operations (e.g., listing objects) and limited consistency guarantees.

Based on the characteristics of cloud object stores, three approaches have emerged.

1. Data lakes

The first is directories of files (i.e., data lakes) that store the table as a collection of objects, typically in columnar format such as Apache Parquet. It's an attractive approach because the table is just a group of objects that can be accessed from a wide variety of tools without a lot of additional data stores or systems. However, both performance and consistency problems are common. Hidden data corruption is common due to failed transactions, eventual consistency leads to inconsistent queries, latency is high, and basic management capabilities like table versioning and audit logs are unavailable.

2. Custom storage engines

The second approach is custom storage engines, such as proprietary systems built for the cloud like the Snowflake data warehouse. These systems can bypass the consistency challenges of data lakes by managing the metadata in a separate, strongly consistent service that's able to provide a single source of truth. However, all I/O operations need to connect to this metadata service, which can increase cloud resource costs and reduce performance and availability. Additionally, it takes a lot of engineering work to implement connectors to existing computing engines like Apache Spark, TensorFlow and PyTorch, which can be challenging for data teams that use a variety of computing engines on their data. Engineering challenges can be exacerbated by unstructured data because these systems are generally optimized for traditional structured

data types. Finally, and most egregiously, the proprietary metadata service locks customers into a specific service provider, leaving customers to contend with consistently high prices and expensive, time-consuming migrations if they decide to adopt a new approach later.

3. Lakehouse

With Delta Lake, an open source ACID table storage layer atop cloud object stores, we sought to build a car instead of a faster horse with not just a better data store, but a fundamental change in how data is stored and used via the lakehouse. A lakehouse is a new architecture that combines the best elements of data lakes and data warehouses. Lakehouses are enabled by a new system design: implementing similar data structures and data management features to those in a data warehouse, directly on the kind of low-cost storage used for data lakes. They are what you would get if you had to redesign storage engines in the modern world, now that cheap and highly reliable storage (in the form of object stores) are available.

Delta Lake maintains information about which objects are part of a Delta table in an ACID manner, using a write-ahead log, compacted into Parquet, that is also stored in the cloud object store. This design allows clients to update multiple objects at once, replace a subset of the objects with another, etc., in a serializable manner that still achieves high parallel read/write performance from the objects. The log also provides significantly faster metadata operations for large tabular data sets. Additionally, Delta Lake offers advanced capabilities like time travel (i.e., the ability to query point-in-time snapshots or roll back erroneous updates), automatic data layout optimization, upserts, caching, and audit logs. Together, these features improve both the manageability and performance of working with data in cloud object stores, ultimately opening the door to the lakehouse architecture that combines the key features of data warehouses and data lakes to create a better, simpler data architecture.

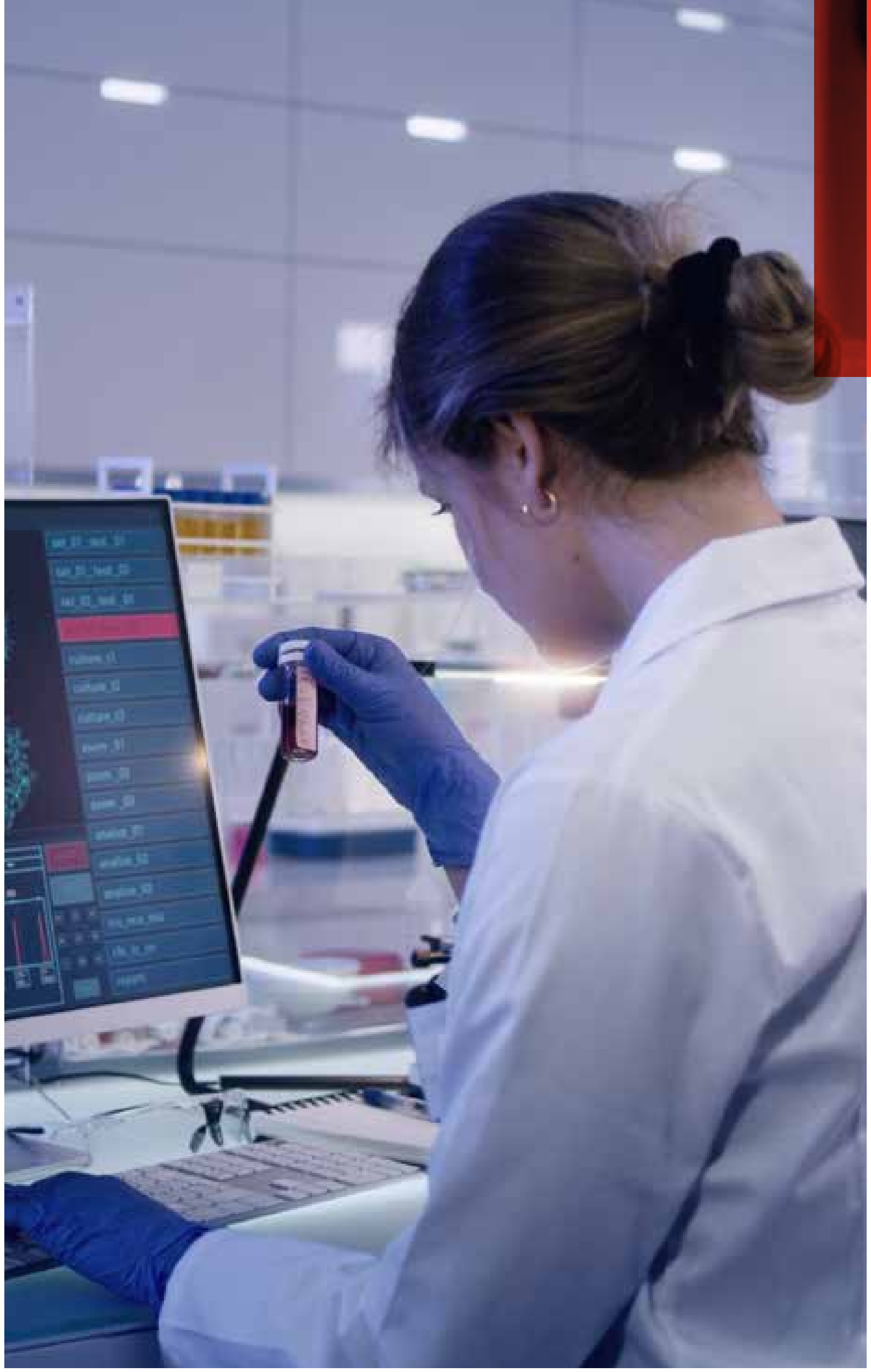
Today, Delta Lake is used across thousands of Databricks customers, processing exabytes of structured and unstructured data each day, as well as many organizations in the open source community. These use cases span a variety of data sources and applications. The data types stored include Change Data Capture (CDC) logs from enterprise OLTP systems, application logs, time-series data, graphs, aggregate tables for reporting, and image or feature data for machine learning. The applications include SQL workloads (most commonly), business intelligence, streaming, data science, machine learning and graph analytics. Overall, Delta Lake has proven itself to be a good fit for most data lake applications that would have used structured storage formats like Parquet or ORC, and many traditional data warehousing workloads.

Across these use cases, we found that customers often use Delta Lake to significantly simplify their data architecture by running more workloads directly against cloud object stores, and increasingly, by creating a lakehouse with both data lake and transactional features to replace some or all of the functionality provided by message queues (e.g., Apache Kafka), data lakes or cloud data warehouses (e.g., Snowflake, Amazon Redshift).

In the research paper, the authors explain:

- The characteristics and challenges of object stores
- The Delta Lake storage format and access protocols
- The current features, benefits and limitations of Delta Lake
- Both the core and specialized use cases commonly employed today
- Performance experiments, including TPC-DS performance

Through the paper, you'll gain a better understanding of Delta Lake and how it enables a wide range of DBMS-like performance and management features for data held in low-cost cloud storage. As well as how the Delta Lake storage format and access protocols make it simple to operate, highly available, and able to deliver high-bandwidth access to the object store.



Understanding Delta Engine

CHAPTER 03

03 Understanding Delta Engine

The Delta Engine ties together a 100% Apache Spark-compatible vectorized query engine to take advantage of modern CPU architecture with optimizations to Spark 3.0's query optimizer and caching capabilities that were launched as part of Databricks Runtime 7.0. Together, these features significantly accelerate query performance on data lakes, especially those enabled by Delta Lake, to make it easier for customers to adopt and scale a lakehouse architecture.

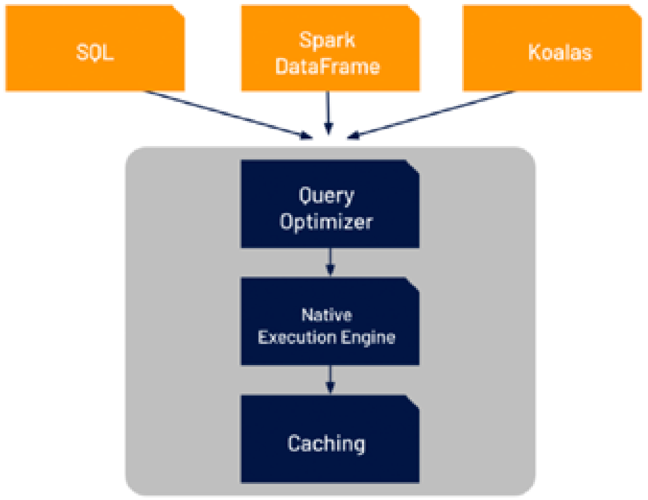
Scaling execution performance

One of the big hardware trends over the last several years is that CPU clock speeds have plateaued. The reasons are outside the scope of this chapter, but the takeaway is that we have to find new ways to process data faster beyond raw compute power. One of the most impactful methods has been to improve the amount of data that can be processed in parallel. However, data processing engines need to be specifically architected to take advantage of this parallelism.

In addition, data teams are being given less and less time to properly model data as the pace of business increases. Poorer modeling in the interest of better business agility drives poorer query performance. Naturally, this is not a desired state, and organizations want to find ways to maximize both agility and performance.

Announcing Delta Engine for high-performance query execution

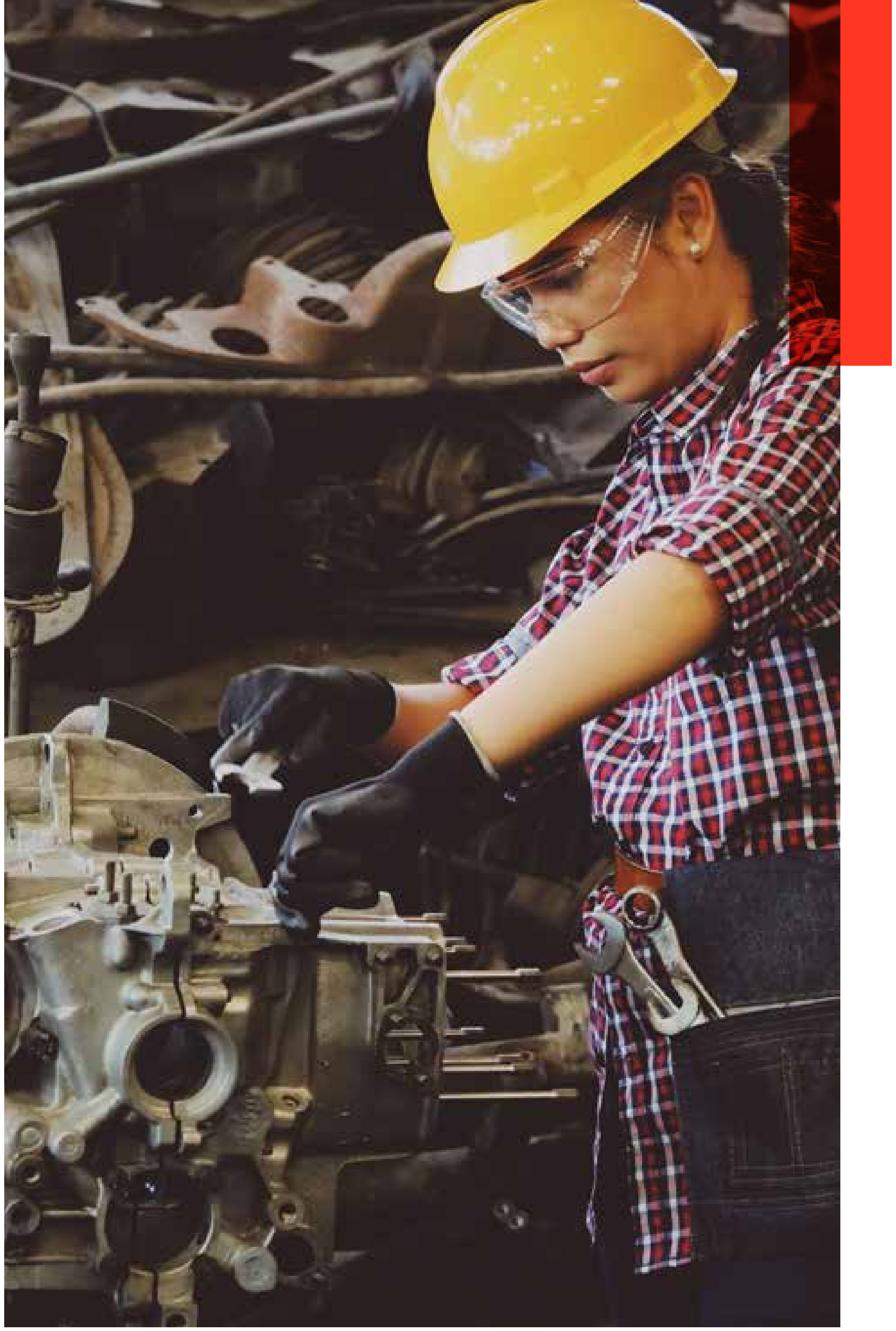
Delta Engine accelerates the performance of Delta Lake for SQL and DataFrame workloads through three components: an improved query optimizer, a caching layer that sits between the execution layer and the cloud object storage, and a native vectorized execution engine that's written in C++.



The improved query optimizer extends the functionality already in Spark 3.0 (cost-based optimizer, adaptive query execution, and dynamic runtime filters) with more advanced statistics to deliver up to 18x increased performance in star schema workloads.

Delta Engine's caching layer automatically chooses which input data to cache for the user, transcoding it along the way in a more CPU-efficient format to better leverage the increased storage speeds of NVMe SSDs. This delivers up to 5x faster scan performance for virtually all workloads.

However, the biggest innovation in Delta Engine to tackle the challenges facing data teams today is the native execution engine, which we call Photon. (We know.



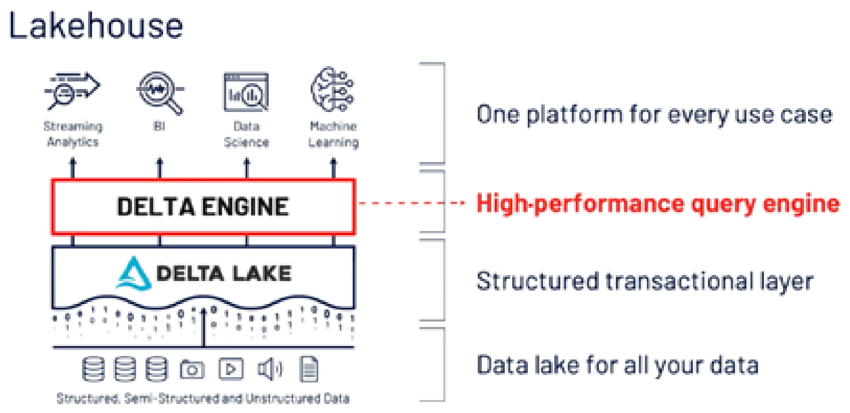
It's in an engine within the engine...). This completely rewritten execution engine for Databricks has been built to maximize the performance from the new changes in modern cloud hardware. It brings performance improvements to all workload types while remaining fully compatible with open Spark APIs.

Getting started with Delta Engine

By linking these three components together, we think it will be easier for customers to understand how improvements in multiple places within the Databricks code aggregate into significantly faster performance for analytics workloads on data lakes.

We're excited about the value that Delta Engine delivers to our customers. While the time and cost savings are already valuable, its role in the lakehouse pattern supports new advances in how data teams design their data architectures for increased unification and simplicity.

For more information on the Delta Engine, watch this keynote address from Spark + AI Summit 2020: Delta Engine: High-Performance Query Engine for Delta Lake.


Lakehouse
Streaming Analytics
BI
Data Science
Machine Learning
DELTA ENGINE
DELTA LAKE
Structured, Semi-Structured and Unstructured Data
One platform for every use case
High-performance query engine
Structured transactional layer
Data lake for all your data

What's next?

Now that you understand Delta Lake and how its features can improve performance, it may be time to take a look at some additional resources.

Data + AI Summit Europe 2020 >

- Photon Technical Deep Dive: How to Think Vectorized
- MLflow, Delta Lake and Lakehouse Use Cases Meetup and AMA
- Common Strategies for Improving Performance on Your Delta Lakehouse
- Achieving Lakehouse Models With Spark 3.0
- Radical Speed for Your SQL Queries With Delta Engine

Vodcasts and podcasts >

- Welcome to Lakehouse. Data Brew | Episode 2
- Data Brew by Databricks | Season 1: Lakehouses
- Data Alone Is Not Enough: The Evolution of Data Architectures

Explore subsequent eBooks in the collection >

- The Delta Lake Series — Fundamentals and Performance
- The Delta Lake Series — Features
- The Delta Lake Series — Streaming
- The Delta Lake Series — Customer Use Cases

Do a deep dive into Delta Lake >

- Analytics on the Data Lake With Tableau and the Lakehouse Architecture
- Visit the site for additional resources

Try Databricks for free >

Learn more >