# Apache Spark and MongoDB

Turning Analytics into Real-Time Action

# Table of Contents

# Introduction

We live in a world of "big data". But it isn't only the data itself that is valuable – it is the insight it can generate. That insight can help designers better predict new products that customers will love. It can help manufacturing companies model failures in critical components, before costly recalls. It can help financial institutions detect fraud. It can help retailers better forecast supply and demand. Marketing departments can deliver more relevant recommendations to its customers. The list goes on.

How quickly an organization can unlock and act on that insight is becoming a major source of competitive advantage. Collecting data in operational systems and then relying on nightly batch ETL (Extract Transform Load) processes to update the Enterprise Data Warehouse (EDW) is no longer sufficient. Speed-to-insight is critical, and so analytics against live operational data to drive real-time action is fast becoming a necessity, enabled by technologies like MongoDB and Apache Spark.

## Why Apache Spark and MongoDB?

Apache Spark is one of the fastest-growing big data projects in the history of the Apache Software Foundation. With its memory-oriented architecture, flexible processing libraries and ease-of-use, Spark has emerged as a leading distributed computing framework for real-time analytics.

With over 10 million downloads, MongoDB is the most popular non-relational database, counting more than one third of the Fortune 100 as customers. Its flexible JSON-based document data model, dynamic schema and automatic scaling on commodity hardware make MongoDB an ideal fit for modern, always-on applications that must manage high volumes of rapidly changing, multi-structured data. Internet of Things (IoT), mobile apps, social engagement, customer data and content management systems are prime examples of MongoDB use cases.

Combining the leading analytics processing engine with the fastest-growing database enables organizations to realize real time analytics. Spark jobs can be executed directly against operational data managed by MongoDB without the time and expense of ETL processes. MongoDB

can then efficiently index and serve analytics results back into live, operational processes. This approach offers many benefits to teams tasked with delivering modern, data driven applications:

- Developers can build more functional applications faster, using a single database technology.
- Operations teams eliminate the requirement for shuttling data between separate operational and analytics infrastructure, each with its own unique configuration, maintenance and management requirements.
- CIOs deliver faster time-to-insight for the business, with lower cost and risk.

This whitepaper will discuss the analytics capabilities offered by MongoDB and Apache Spark, before providing an overview of how to configure and combine them into a real-time analytics engine. It will conclude with example use cases.

# Rich Analytics in MongoDB

Unlike NoSQL databases that offer little more than basic key-value query operations, developers and data scientists can use MongoDB's native query processing and data navigation capabilities to generate many classes of analytics, before needing to adopt dedicated frameworks such as Spark or Hadoop for more specialized tasks.

Leading organizations including Salesforce, McAfee, Buzzfeed, Intuit, City of Chicago, Amadeus, Buffer and many others rely on MongoDB's powerful query functionality, aggregations and indexing to generate analytics in real-time directly against their live, operational data.

In this section of the paper, we provide more detail on MongoDB's native analytics features, before then exploring the additional capabilities Apache Spark can bring.

## MongoDB Query Model

MongoDB users have access to a broad array of query, projection and update operators supporting analytical queries against live operational data:

- Aggregation and MapReduce queries, discussed in more detail below
- Range queries returning results based on values defined as inequalities (e.g., greater than, less than or equal to, between)
- Geospatial queries returning results based on proximity criteria, intersection and inclusion as specified by a point, line, circle or polygon
- Text search queries returning results in relevance order based on text arguments using Boolean operators (e.g., AND, OR, NOT)
- Key-value queries returning results based on any field in the document, often the primary key
- Native BI and Hadoop integration, for deep, offline analytics.

With the combination of MongoDB's dynamic document model and comprehensive query framework, users are able to store data before knowing all of the questions they will need to ask of it.

## Data Aggregation

The MongoDB Aggregation Pipeline is similar in concept to the SQL GROUP BY statement, enabling users to generate aggregations of values returned by the query (e.g., count, minimum, maximum, average, intersections) that can be used to power analytics dashboards and vizualizations.

Using the Aggregation Framework, documents in a MongoDB collection (analogous to a table in a relational database) pass through an aggregation pipeline, where they are processed by operators. Expressions produce output documents based on calculations performed on the input documents. The accumulator expressions used in the $group operator maintain state (e.g., totals, mins, maxs, averages, standard deviations) as documents progress through the pipeline. MongoDB also includes the ability to combine data from multiple collections by implementing left outer joins through the $lookup operator, which can now be included as an aggregation stage.

The aggregation pipeline enables multi-step data enrichment and transformations to be performed directly in

the database with a simple declarative interface, supporting processes such as lightweight ETL to be performed within MongoDB.

Result sets from the aggregation pipeline can be written to a named collection with no limit to the output size (subject to the underlying storage system). Existing collections can be replaced with new results while maintaining previously defined indexes to ensure queries can always be returned efficiently over rapidly changing data.

### Incremental MapReduce within MongoDB

MongoDB provides native support for MapReduce, enabling complex data processing that is expressed in JavaScript and executed across data in the database. Multiple MapReduce jobs can be executed concurrently across both single servers and sharded collections.

The incremental MapReduce operation uses a temporary collection during processing so it can be run periodically over the same target collection without affecting intermediate states. This mode is useful when continuously updating statistical output collections used by reporting dashboards.

### Optimizing Analytics Queries with MongoDB Indexes

MongoDB provides a number of different index types to optimize performance of real-time and ad-hoc analytics queries across highly variable, fast moving data sets. These same indexes can be used by Apache spark to filter only interesting subsets of data against which analytics can be run.

Indexes can be created on any field within a document or sub-document. In addition to supporting single field and compound indexes, MongoDB also supports indexes of arrays with multiple values, short-lived data (i.e., Time To Live), partial, sparse, geospatial and text data, and can enforce constraints with unique indexes. Index intersection allows indexes to be dynamically combined at runtime to efficiently answer explorative questions of the data. Refer to the documentation for the full list of index types.

The MongoDB query optimizer selects the best index to use by running alternate query plans and selecting the index with the best response time for each query type. The results of this empirical test are stored as a cached query plan and are updated periodically.

MongoDB also supports covered queries – where returned results contain only indexed fields, without having to access and read from source documents. With the appropriate indexes, analytics workloads can be optimized to use predominantly covered queries.

Building upon MongoDB, Apache Spark can offer additional analytics capabilities to serve real time, operational processes.

## Apache Spark: Extending MongoDB's Analytics Capabilities

Apache Spark is a powerful open source processing engine designed around speed, ease of use, and sophisticated analytics. Originally developed at UC Berkeley in 2009, Spark has seen rapid adoption by enterprises across a range of industries.

Spark is a general-purpose framework used for many types of data processing. Spark comes packaged with support for machine learning, interactive queries (SQL), statistical queries with R, graph processing, ETL, and streaming. For loading and storing data, Spark integrates with a number of storage and messaging systems including Amazon S3, Kafka, HDFS, RDBMSs, and MongoDB. Additionally, Spark supports a variety of popular development languages including Java, Python and Scala.

### Apache Spark Benefits

Spark was initially designed for interactive queries and iterative algorithms, as these were two major use cases not well served by batch frameworks such as Hadoop's MapReduce. Consequently Spark excels in scenarios that require fast performance, such as iterative processing, interactive querying, large-scale batch operations, streaming, and graph computations.

Developers and data scientists typically deploy Spark for:

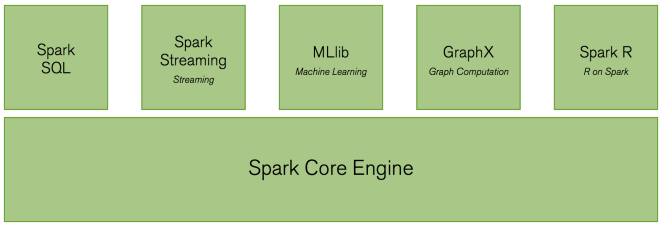


**Figure 1:** Apache Spark Ecosystem

- Simplicity. Easy-to-use APIs for operating on large datasets. This includes a collection of sophisticated operators for transforming and manipulating semi-structured data.
- Speed. By exploiting in-memory optimizations, Spark has shown up to 100x higher performance than MapReduce running on Hadoop.
- Unified Framework. Packaged with higher-level libraries, including support for SQL queries, machine learning, stream and graph processing. These standard libraries increase developer productivity and can be combined to create complex workflows.

Spark allows programmers to develop complex, multi-step data pipelines using a directed acyclic graph (DAG) pattern. It also supports in-memory data sharing across DAGs, so that different jobs can work with the same data. Spark takes MapReduce to the next level with less expensive shuffles during data processing. Spark holds intermediate results in memory rather than writing them to disk, which is useful especially when a process needs to iteratively work on the same dataset. Spark is designed as an execution engine that works with data both in-memory and on-disk. Spark operators perform external operations when data does not fit in memory. Spark can be used for processing datasets that are larger than the aggregate memory in a cluster.

The Resilient Distributed Dataset (RDD) is the core data structure used in the Spark framework. RDD is analogous to a table in a relational database or a collection in MongoDB. RDDs are immutable – they can be modified with a transformation, but the transformation returns a new RDD while the original RDD remains unchanged.

RDDs supports two types of operations: Transformations and Actions. Transformations return a new RDD after processing it with a function such as map, filter, flatMap, groupByKey, reduceByKey, aggregateByKey, pipe, or coalesce.

The Action operation evaluates the RDD and returns a new value. When an Action function is called on a RDD object, all the processing queries are computed and the result value is returned. Some of the Action operations are reduce, collect, count, first, take, countByKey, and foreach.

## When to Use Spark with MongoDB

While MongoDB natively offers rich analytics capabilities, there are use-cases where integrating the Spark engine can extend the real-time processing of operational data managed by MongoDB, and allow users to operationalize results generated from Spark within real-time business processes supported by MongoDB.

Spark can take advantage of MongoDB's rich secondary indexes to extract and process only the range data it needs – for example, analyzing all customers located in a specific geography. This is very different from other databases that either do not offer, or do not recommend the use of secondary indexes. In these cases, Spark would need to extract all data based on a simple primary key, even if only a subset of that data is required for the Spark process. This means more processing overhead, more hardware, and longer time-to-insight for the analyst.

Examples of where it is useful to combine Spark and MongoDB include the following.

### Rich Operators & Algorithms

Spark supports over 100 different operators and algorithms for processing data. Developers can use these to perform advanced computations that would otherwise require more programmatic effort combining the MongoDB aggregation framework with application code. For example, Spark offers native support for advanced machine learning algorithms including k-means clustering and Gaussian mixture models.

Consider a web analytics platform that uses the MongoDB aggregation framework to maintain a real time dashboard displaying the number of clicks on an article by country; how often the article is shared across social media; and the number of shares by platform. With this data, analysts can

quickly gain insight on how content is performing, optimizing user's experience for posts that are trending, with the ability to deliver critical feedback to the editors and ad-tech team.

Spark's machine learning algorithms can also be applied to the log, clickstream and user data stored in MongoDB to build precisely targeted content recommendations for its readers. Multi-class classifications are run to divide articles into granular sub-categories, before applying logistic regression and decision tree methods to match readers' interests to specific articles. The recommendations are then served back to users through MongoDB, as they browse the site.

### Processing Paradigm

Many programming languages can use their own MongoDB drivers to execute queries against the database, returning results to the application where additional analytics can be run using standard machine learning and statistics libraries. For example, a developer could use the MongoDB Python or R drivers to query the database, loading the result sets into the application tier for additional processing.

However, this starts to become more complex when an analytical job in the application needs to be distributed across multiple threads and nodes. While MongoDB can service thousands of connections in parallel, the application would need to partition the data, distribute the processing across the cluster, and then merge results. Spark makes this kind of distributed processing easier and faster to develop. MongoDB exposes operational data to Spark's distributed processing layer to provide fast, real-time analysis. Combining Spark queries with MongoDB indexes allows data to be filtered, avoiding full collection scans and delivering low-latency responsiveness with minimal hardware and database overhead.

### Skills Re-Use

With libraries for SQL, machine learning and others – combined with programming in Java, Scala and Python – developers can leverage existing skills and best practices to build sophisticated analytics workflows on top of MongoDB.

## Configuring & Running Spark with MongoDB

There are currently two packaged connectors to integrate MongoDB and Spark:

- The MongoDB Connector for Hadoop and Spark
- The Spark-MongoDB Connector, developed by Stratio

### MongoDB Connector for Hadoop & Spark

The MongoDB Connector is a plugin for both Hadoop and Spark that provides the ability to use MongoDB as an input source and/or an output destination for jobs running in both environments. Note that the connector directly integrates Spark with MongoDB and has no dependency on also having a Hadoop cluster running.

Input and output classes are provided allowing users to read and write against both live MongoDB collections and against BSON (Binary JSON) files that are used to store MongoDB snapshots. Standard MongoDB connection strings, including authentication credentials and read preferences, are used to specify collections against which Spark queries are run, and where results are written back to. JSON formatted queries and projections can be used to filter the input collection, which uses a method in the connector to create a Spark RDD from the MongoDB collection.

There are only two dependencies for installing the connector:

- Download the MongoDB Connector for Hadoop and Spark. For Spark, all you need is the "core" jar.
- Download the jar for the MongoDB Java Driver.

The documentation provides a worked example using MongoDB as both a source and sink for Spark jobs, including how to create and save Spark's Resilient Distributed Dataset (RDD) to MongoDB collections.

Databricks (founded by the creators of Spark) and MongoDB have also collaborated in creating a demonstration app that documents how to configure and

| | MongoDB Connector for Hadoop | Stratio Spark-MongoDB Connector |
|---|---|---|
| Machine Learning | Yes | Yes |
| SQL | Not currently | Yes |
| DataFrames | Not currently | Yes |
| Streaming | Not currently | Not currently |
| Python | Yes | Yes<br>Using SparkSQL syntax |
| Use MongoDB secondary indexes to filter input data | Yes | Yes |
| Compatibility with MongoDB replica sets and sharding | Yes | Yes |
| MongoDB Support | Yes<br>Read and write | Yes<br>Read and write |
| HDFS Support | Yes<br>Read and write | Partial<br>Write only |
| Support for MongoDB BSON Files | Yes | No |
| Commercial Support | Yes<br>With MongoDB Enterprise Advanced | Yes<br>Provided by Stratio |

**Table 1:** Comparing Spark Connectors for MongoDB

use Spark to execute analysis of market trading data managed by MongoDB.

## Spark-MongoDB Connector

The Spark-MongoDB Connector is a library that allows the user to read and write data to MongoDB with Spark, accessible from Python, Scala and Java API's. The Connector is developed by Stratio and distributed under the Apache Software License.

The MongoDB SparkSQL data source implements the Spark Dataframe API, and is fully implemented in Scala. The connector allows integration between multiple data sources that implement the same API for Spark, in addition to using the SQL syntax to provide higher-level abstractions for complex analytics. It also includes an easy way to integrate MongoDB with Spark Python API.

Stratio's SparkSQL MongoDB connector implements the PrunedFilteredScan API instead of the TableScan API. Therefore, in addition to MongoDB's query projections, the connector can avoid scanning the entire collection, which is not the case with other databases.

The connector supports the Spark Catalyst optimizer for both rule-based and cost-based query optimization. To operate against multi-structured data, the connector infers the schema by sampling documents from the MongoDB collection. This process is controlled by the samplingRatio parameter. If the schema is known, the developer can provide it to the connector, avoiding the need for any inference. Once data is stored in MongoDB, Stratio provides an ODBC/JDBC connector for integrating results with any BI tool, such as Tableau and QlikView.

The connector can be downloaded from the community Spark Packages repository.

Installation is simple – the connector can be included in a Spark application with a single command.

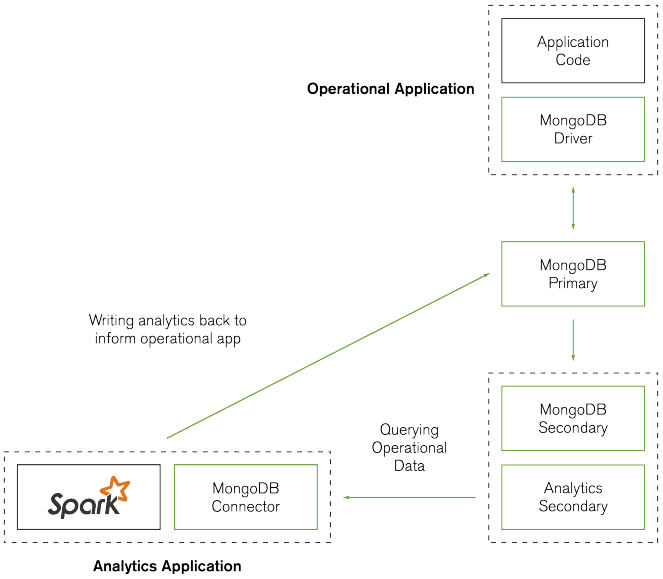


**Figure 2:** Using MongoDB Replica Sets to Isolate Analytics from Operational Workloads

You can learn more about the Spark-Mongodb connector from Stratio's GitHub page.

## Operations: Separating Operational from Analytical Workloads

Spark can be run on any physical node within a MongoDB replica set. However, executing heavyweight analytics jobs on a host that is also serving requests from an operational application can cause contention for system resources. Similarly the working set containing the indexes and most frequently accessed data in memory is overwritten by data needed to process analytics queries. Whether running MongoDB aggregations or Spark processes, it is a best practice to isolate competing traffic from the operational and analytics workloads.

Using native replication, MongoDB maintains multiple copies of data called replica sets. Replica sets are primarily designed to provide high availability and disaster recovery, and to isolate different workloads running over the same data set.

By default, all read and write operations are routed to the primary replica set member, thus ensuring strong consistency of data being read and written to the database. Applications can also be configured to read from secondary replica set members, where data is eventually consistent by default. Reads from secondaries can be useful in scenarios where it is acceptable for data to be several seconds (typically milliseconds) behind the live data, which is typical for many analytics and reporting applications. To ensure a secondary node serving analytics queries is never considered for election to the primary replica set member, it should be configured with a priority of 0.

Unlike other databases that have to offload Spark to dedicated analytics nodes that are running an entirely different configuration from the database nodes, all members of a MongoDB replica set share the same availability, scalability and security mechanisms. This approach reduces operational complexity and risk.

# Integrating MongoDB & Spark Analytics with BI Tools & Hadoop

## BI Integration

Real time analytics generated by MongoDB and Spark can serve both online operational applications and offline reporting systems, where it can be blended with historical data and analytics from other data sources. To power dashboards, reports and visualizations data stored in MongoDB can be easily explored with industry-standard SQL-based BI and analytics platforms. Using the MongoDB Connector for BI, analysts, data scientists and business users can seamlessly visualize semi-structured and unstructured data managed in MongoDB, alongside traditional data in their SQL databases, using the same BI tools deployed within millions of enterprises.

The Connector for BI translates SQL statements issued by the BI tool into equivalent MongoDB queries that are then sent to MongoDB for processing, and then returns results for visualization in the tool's UI.

## Hadoop Integration

Like MongoDB, Hadoop is seeing growing adoption across industry and government, becoming an adjunct to – and in

some cases a replacement of – the traditional Enterprise Data Warehouse.

Many organizations are harnessing the power of Hadoop and MongoDB together to create complete big data applications:

- MongoDB powers the online, real time operational application, serving business processes and end-users
- Hadoop consumes data from MongoDB, blending its with data from other operational systems to fuel sophisticated analytics and machine learning. Results are loaded back to MongoDB to serve smarter operational processes – i.e., delivering more relevant offers, faster identification of fraud, improved customer experience.

Organizations such as eBay, Orbitz, Pearson, Square Enix and SFR are using MongoDB alongside Hadoop to operationalise big data.

As well as providing integration for Spark, the MongoDB Connector for Hadoop and Spark is also the foundation for MongoDB's integration with Hadoop. A single extensible connector supporting both Spark and Hadoop makes it much simpler for developers and operations teams to build and run powerful big data pipelines.

The MongoDB Connector for Hadoop and Spark presents MongoDB as a Hadoop data source allowing Hadoop jobs to read data from MongoDB directly without first copying it to HDFS, thereby eliminating the need to move TB of data between systems. Hadoop jobs can pass queries as filters, thereby avoiding the need to scan entire collections and speeding up processing; they can also take advantage of MongoDB's indexing capabilities, including text and geospatial indexes.

As well as reading from MongoDB, the Connector also allows results of Hadoop jobs to be written back out to MongoDB, to support real-time operational processes and ad-hoc querying by both MongoDB and Spark query engines.

The Connector supports MapReduce, Spark on HDFS, Pig, Hadoop Streaming (with Node.js, Python or Ruby) and Flume jobs. Support is also available for SQL queries from Apache Hive to be run across MongoDB data sets. The connector is certified on the leading Hadoop distributions from Cloudera, Hortonworks and MapR.

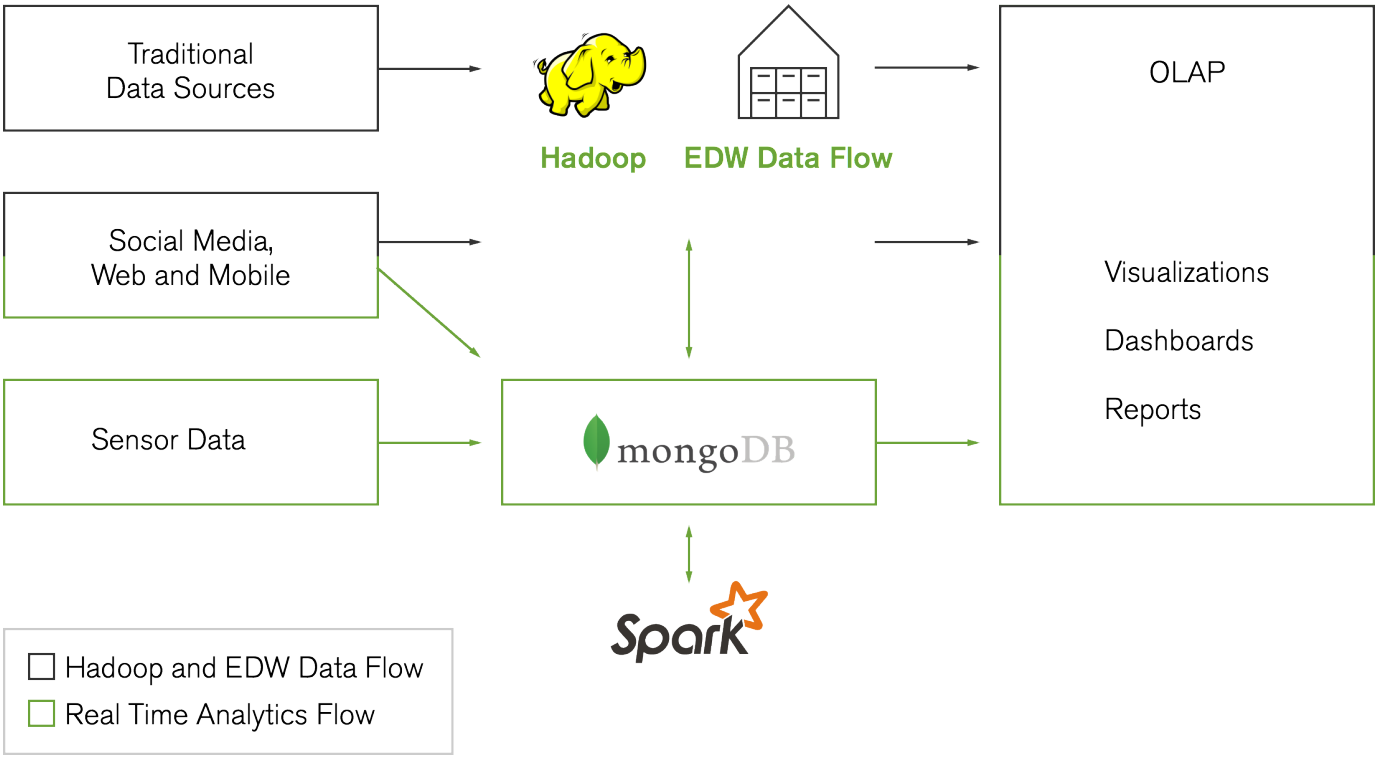


**Figure 3:** Integrating MongoDB & Spark with BI Tools

To learn more about running MongoDB with Hadoop, download the Big Data: Examples and Guidelines whitepaper

## MongoDB & Spark in the Wild

Many organizations are already combining MongoDB and Spark to unlock real-time analytics from their operational data:

- A global manufacturing company has built a pilot project to estimate warranty returns by analyzing material samples from production lines. The collected data enables them to build predictive failure models using Spark Machine Learning and MongoDB.
- A video sharing website is using Spark with MongoDB to place relevant advertisements in front of users as they browse, view and share videos.

A multinational banking group operating in 31 countries with 51 million clients implemented a unified real-time monitoring application with the Stratio Big Data (BD) platform, running Apache Spark and MongoDB. The bank wanted to ensure a high quality of service across its online channels, and so needed to continuously monitor client activity to check service response times and identify potential issues. The technology stack included:

- Apache Flume to aggregate log data
- Apache Spark to process log events in real time
- MongoDB to persist log data, processed events and Key Performance Indicators (KPIs).

The aggregated KPIs enable the bank to meet its objective of analyzing client and systems behavior in order to improve customer experience. Collecting raw log data allows the bank to immediately rebuild user sessions if a service fails, with the analysis providing complete traceability to identify the root cause of the issue.

MongoDB was selected as the database layer of the application because of it's always-on availability, high performance and linear scalability. The application needed a database with a fully dynamic schema to support high volumes of rapidly changing semi-structured and unstructured data being ingested from the variety of logs, clickstreams and social networks. Operational efficiency was also important. The bank wanted to automate orchestrating the application flow and have deep visibility into the database with proactive monitoring. MongoDB Cloud Manager provided the required management platform.

## Conclusion

MongoDB natively provides a rich analytics framework within the database. Multiple connectors are also available to integrate Spark with MongoDB to enrich analytics capabilities by enabling analysts to apply libraries for machine learning, streaming and SQL to MongoDB data. Together MongoDB and Apache Spark are already enabling developers and data scientists to turn analytics into real-time action.

## We Can Help

We are the MongoDB experts. Over 2,000 organizations rely on our commercial products, including startups and more than a third of the Fortune 100. We offer software and services to make your life easier:

MongoDB Enterprise Advanced is the best way to run MongoDB in your data center. It's a finely-tuned package of advanced software, support, certifications, and other services designed for the way you do business.

MongoDB Cloud Manager is the easiest way to run MongoDB in the cloud. It makes MongoDB the system you worry about the least and like managing the most.

MongoDB Professional helps you manage your deployment and keep it running smoothly. It includes support from MongoDB engineers, as well as access to MongoDB Cloud Manager.

Development Support helps you get up and running quickly. It gives you a complete package of software and services for the early stages of your project.

MongoDB Consulting packages get you to production faster, help you tune performance in production, help you scale, and free you up to focus on your next release.

MongoDB Training helps you become a MongoDB expert, from design to operating mission-critical systems at scale. Whether you're a developer, DBA, or architect, we can make you better at MongoDB.

Contact us to learn more, or visit www.mongodb.com.

# Resources

For more information, please visit mongodb.com or contact us at sales@mongodb.com.

Case Studies (mongodb.com/customers)
Presentations (mongodb.com/presentations)
Free Online Training (university.mongodb.com)
Webinars and Events (mongodb.com/events)
Documentation (docs.mongodb.org)
MongoDB Enterprise Download (mongodb.com/download)

mongoDB

New York • Palo Alto • Washington, D.C. • London • Dublin • Barcelona • Sydney • Tel Aviv
US 866-237-8815 • INTL +1-650-440-4474 • info@mongodb.com