
Accelerating Data-Driven Applications
*with* Couchbase and Spark
Couchbase

# Accelerating Data-Driven Applications With Couchbase and Spark

TABLE OF CONTENTS



## Introduction

In the world of "big data," data is a treasured asset. However, it's not the data itself that is valuable – it's the actionable insights that it can generate. That insight allows companies to better interact with their customers, offer enhanced individualized services, provide better products, and ultimately be more competitive and successful. Manufacturing companies can model component reliability and predict failures before they happen, avoiding costly recalls. Utility companies can compare real-time sensor data to models that identify and adjust operations before outages occur. Customer-centric services can provide information and recommendations that are individually tailored. These are just a few examples of how actionable insights can be used to reduce operating costs, increase revenue, and improve customer service quality within a company.

Crucial factors in leveraging these operational advantages is the time-to-insight (the length of time it takes to analyze the data and turn it into actionable information) and time-to-action (the length of time it takes to communicate that actionable information to users, customers, and automated applications). The traditional approach to turning raw data into actions is to collect data in multiple operational systems, use nightly batch ETL (Extract Transform Load) processes to load the data into an analytical platform, crunch the data, and then finally push the generated analytical results to yet another platform that serves to inform users, customers, and applications. This approach is no longer acceptable – it simply takes too long. Because time-to-insight and time-to-action are critical, real-time analytics against live operational data yielding immediate actionable results has become an absolute requirement. Low-latency, high volume NoSQL operational databases like Couchbase, combined with scalable in-memory processing platforms like Apache Spark provides the competitive edge that innovative companies are searching for – near real-time actionable insights based on rich analytics and sophisticated machine learning.

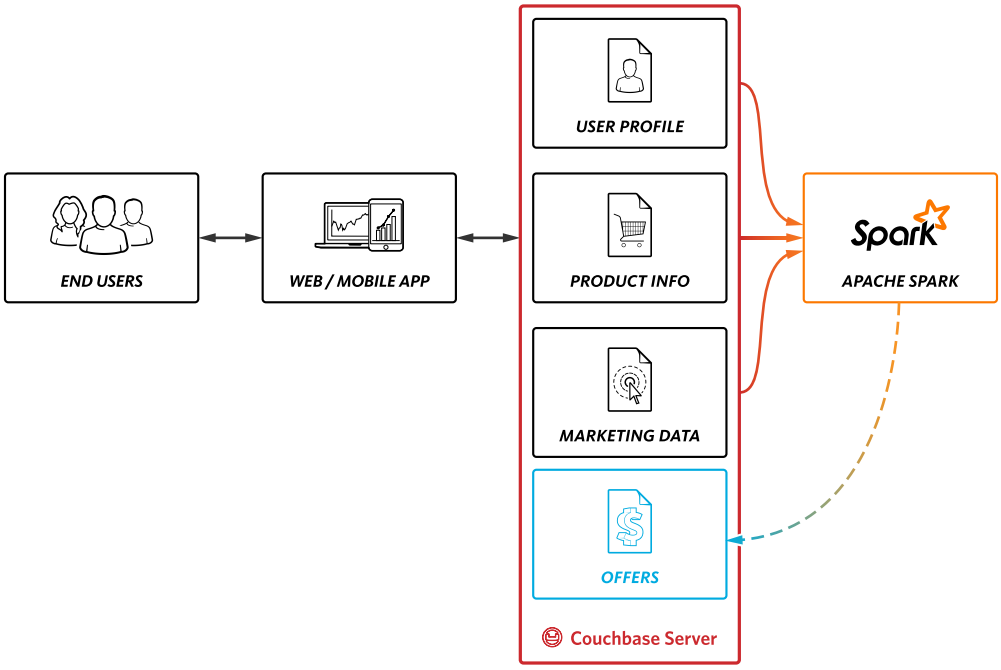
## Best-of-Breed Technologies

In the early days of big data, Hadoop MapReduce jobs were the most commonly used approach to building distributed, scalable analytics. Many of these solutions are still in use today. However, Apache Spark has clearly become the successor to Hadoop MapReduce for analytical and machine learning workloads due to its ease of use and in-memory performance.

NoSQL databases are the go-to technology for operational databases, due to their scalability, performance, flexibility, and lower operational cost. Couchbase is the leading NoSQL database with a scale-out, memory-first, shared nothing architecture, a powerful SQL-based query language for JSON documents, and a secure database platform that provides high availability, scalability, and performance. Companies like General Electric, Marriott, United Airlines, and many others have implemented mission-critical operational databases using Couchbase to address the demanding and changing requirements of the Internet of Things (IoT), Hospitality Management, Transportation Management, Financial Services, Retail, and eCommerce, just to name a few applications.

Combining the leading in-memory analytics processing engine (Apache Spark) with the leading memory-first architectured NoSQL database (Couchbase Server) enables organizations to execute real-time analytics that lead to real-time, actionable operational insights. Spark jobs can be executed directly against operational data managed by Couchbase without the time, expense, and overhead of traditional disk-based ETL processes.

**Combining Apache Spark with Couchbase Server enables real-time analytics and actionable insights.**



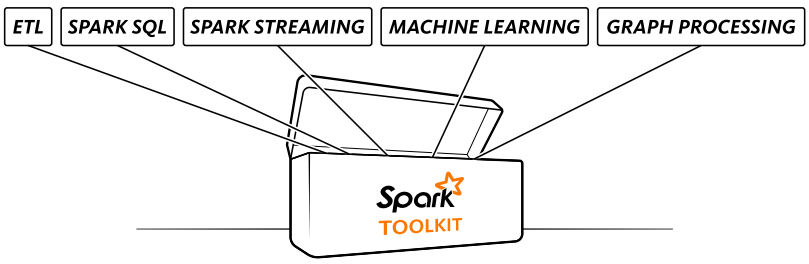

*Data + Analytics = Actionable Results*

## Apache Spark

**A high throughput, low-latency data source and target is critical in order to leverage the performance of Spark.**

Apache Spark™ is an open source, distributed, in-memory data processing platform built to provide low-latency analytics. It was created at AMPLab in University of California, Berkeley as part of the Berkeley Data Analytics Stack (BDAS) in 2009 and later released as an Apache project. Since then, it has become the fastest growing Apache project ever, with over a thousand contributors.[1] Modern web, mobile, and IoT applications leverage Spark to rapidly process huge volumes of data, performing complex analytics and machine learning algorithms in parallel with built-in fault tolerance.

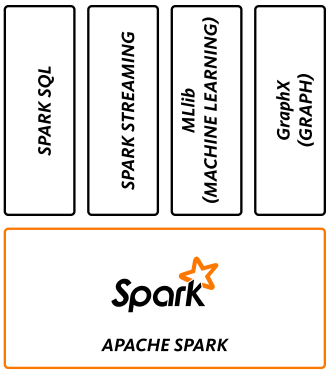
In-memory Resilient Distributed Datasets (RDDs) are the fundamental data structure within Apache Spark. Using RDDs, data analysts can easily build rich analytics, data processing, and machine learning applications in a variety of common languages including Java, Python, Scala, and R. RDDs provide a generalized data abstraction for a wide variety of data sources and targets, including any Hadoop data source, Flume, Kafka, Twitter, NoSQL, relational databases, and many more. RDDs form the basis for higher level data abstractions within Spark, such as DataFrames, Datasets, and DStreams which simplify processing the raw data. Apache Spark applications read and write RDDs to bring in the source data for processing and to write out the processed data results, interacting with a data store at both ends of the process. Providing a high throughput, low-latency data source and target is critical in order to leverage the in-memory performance advantages of Spark.




[1] Databricks: *Apache Spark 2015 Year In Review* [https://databricks.com/blog/2016/01/05/apache-spark-2015-year-in-review.html]

Learn more about Apache Spark.

Apache Spark users can use command line, ad hoc, and programmatic APIs. The common Apache Spark core library supports multiple data processing libraries including Spark SQL, Spark Streaming, MLlib for machine learning, and GraphX for graph traversal. Developers can combine these libraries seamlessly within the same application. Based on a distributed, in-memory architecture Spark processing enriches, analyzes, and augments data 10-100 times faster than Hadoop MapReduce.[2] It automatically distributes tasks to fault tolerant worker processes, which can automatically recover from failures. Apache Spark applications can be developed on a simple, single-node workstation and deployed to a multi-node cluster without changing the application code.

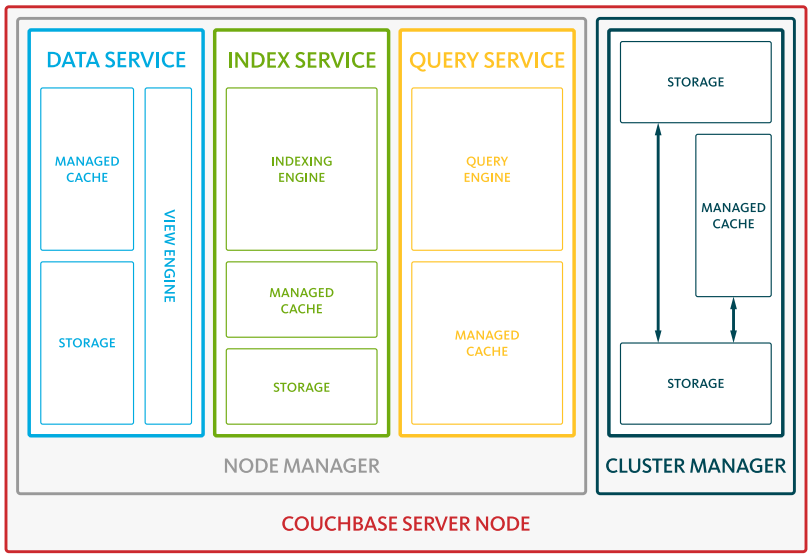



*Apache Spark Libraries*

## Couchbase Server

**Couchbase data operations are performed through a high-speed, in-memory cache for maximum throughput and minimal latency.**

Couchbase Server is the leading NoSQL database, designed with a distributed, memory-first, shared nothing architecture that ensures high performance, scalability, and availability. It manage rich, flexible JSON documents as well as simple key-value pairs. Couchbase Server includes a SQL query language, called N1QL (pronounced "nickel"), that allows application developers to use a universally understood query paradigm against JSON documents. The flexibility of JSON allows developers to change their application without having to change or migrate the underlying schema. With N1QL, developers can build applications easier and faster by leveraging the power of SQL with the flexibility of JSON. Couchbase Server includes rich indexing options to accelerate query and key-value data access. All data operations are performed through a configurable high-speed, in-memory cache, thereby providing an extremely high read/write throughput rate with millisecond latency. Couchbase Server supports a simple, flexible configuration that enables both scale-out and scale-up scenarios. It also provides built-in, high-speed in-memory replication, failover detection, and high availability capabilities.




*Couchbase Server Node Components*

Learn more about Couchbase Server.

Couchbase Server is application developer-friendly, with built-in integration to frameworks like Spring and Ottoman ODM for Node.js. It provides programmatic APIs in many languages including Java, C, Go, Node.js, PHP, Python, and .NET. It provides rich integration with Big Data tools including Hadoop, Storm, Kafka, and Elasticsearch in addition to the Couchbase Spark Connector. Like Spark, Couchbase applications can be developed on a simple, single-node workstation and deployed to a multi-node cluster without changing the application code. Application developers can leverage the flexibility and power of SQL for JSON, the flexible data model, powerful query capabilities, and scalable architecture to build highly responsive applications using less code in less time.

[2] InfoWorld: *Five things you need to know about Hadoop v. Apache Spark*
[http://www.infoworld.com/article/3014440/big-data/five-things-you-need-to-know-about-hadoop-v-apache-spark.html]

## Why Use Spark and Couchbase Together?

Why use Spark and Couchbase together? Well, it depends on the problem that you're trying to solve. For some developers, it may be about integration and adding analytics and machine learning to your Couchbase operational application. For others, it may be about adding a high performance, low-latency, scalable database to their existing Spark processing, or extending the usefulness of the analytical results by persisting them to a NoSQL operational database platform that can reach millions of customers.

**For Spark users, Couchbase provides the lowest latency, highest throughput, most reliable NoSQL data source and target.**

For Spark users, Couchbase provides the lowest latency, highest throughput, most reliable NoSQL data source and target available, thus accelerating your analytical and machine learning applications and reducing the overall time it takes to go from raw data to actionable insights. Additionally, the Couchbase Spark Connector provides the most complete Spark/NoSQL integration available, unlike some other Spark connectors on the market that have uneven integration and built-in limitations. The Couchbase Spark Connector supports all of the common Spark data structures and libraries, including Spark Core (RDDs), Spark SQL (Datasets, DataFrames), and Spark Streaming (DStreams) for both read and write operations. Additionally, through the N1QL query language Spark users can leverage a broad set of useful extensions for working with flexible JSON schemas, including INFER, NEST, UNNEST, IS MISSING, and functions over arrays and nested objects, which are not available natively in Spark SQL.

**For Couchbase users, Spark enables rich, scalable analytics to their operational applications.**

For Couchbase users looking to add rich, scalable analytics to their operational database applications, Apache Spark is a great fit. It provides the same "Develop With Agility, Deploy at Any Scale" approach that they are already familiar with in Couchbase, as well as a rich set of analytical functions and machine learning capabilities. The Couchbase Spark Connector allows Couchbase data to be read by or automatically streamed to Spark applications, and to quickly write the resulting insights back into Couchbase in order to make them actionable at scale. Applications can utilize Spark to analyze and enrich the Couchbase data. Some specific examples include:

- **Advanced Analytics.** Perform data analysis and machine learning, persisting the results back into Couchbase.
- **Data Aggregation and Integration.** Combine data from multiple data sources into Couchbase to provide a comprehensive view, a common requirement for Customer and Product 360 applications.
- **Data Enrichment.** Add data, from other sources or as the result of an analytical process, to existing data already in Couchbase. This is a common element of product recommendation, product promotion, and fraud detection applications, which add data based on analytics and predictive models to an existing customer profile for big performance gains.
- **ETL.** Transform data inside or outside of Couchbase and store back into Couchbase, a common task in data aggregation or data cleansing applications.
- **Streaming Analytics.** Database updates are streamed and analyzed in real time in Spark, resulting in near real-time insights and immediate action. Common in cases where Couchbase manages real-time operational data like security logs, click streams, shopping carts, financial transactions, IoT, and mobile information.

## Using Spark and Couchbase Together

**Use Couchbase and Spark together to enrich operational data and generate actionable insights.**

The Couchbase Spark Connector provides APIs for Java and Scala with support for Maven and sbt respectively, making it easy to integrate them and get started. As a developer, you can build Java or Scala applications using the development environment of your choice, or simply start by using the Spark Shell.

There are several ways to bring Couchbase data into a Spark context:

**Learn more about working with RDDs.**

- **RDDs.** A common scenario is to create resilient distributed datasets (RDDs) out of documents stored in Couchbase. When you need to extract data out of Couchbase, the Couchbase Spark Connector creates RDDs for you. You can create and persist RDDs by using either Couchbase key-value pairs, views, spatial views, or N1QL (SQL).

**Learn more about DataFrames and Spark.**

- **DataFrames.** In order to use Spark SQL queries, you need to create and persist DataFrames via the Spark SQL DataFrame API. A Spark DataFrame is like an RDD, but it includes a schema. The Couchbase Spark Connector provides support for automatic schema inference, as well as manual schema specification when creating the DataFrame. The Couchbase Spark Connector supports both implicit imports using Scala, as well as direct access using other languages of DataFrames to bring data in from Couchbase. The Connector also allows Spark applications to persist DataFrames to Couchbase.

- **Datasets.** Introduced in Apache Spark 1.6, Datasets are a type-safe version of DataFrames with compile time checks, built-in performance optimizations, and a convenient object-oriented interface to a wide array of programming languages.

**Learn more about Spark Streaming with Couchbase.**

- **DStreams.** The Couchbase Spark Connector works with Spark Streaming by using the Couchbase Server replication protocol (called DCP) to receive mutations from the server as they happen and provide them to you in the form of a Spark DStream. You can create and persist DStreams.

The Couchbase Spark Connector is aware of the Couchbase cluster topology (where the nodes and the services within the Couchbase cluster are located). This provides a couple of advantages:

**1. Operations on Spark RDDs, DataFrames, Datasets, and DStreams do not need to know where the Couchbase data is stored or organized.** The Couchbase cluster can be changed without having to change the Spark applications. The Couchbase Spark Connector will ensure that operations are automatically sent to the appropriate Couchbase Server nodes.

**2. The Couchbase Spark Connector can pass node location hints back to the Spark processor**, allowing the Spark scheduler/dispatcher to use co-located workers when appropriate. This reduces the amount of data transfer around the network and improves performance of the Spark applications.

Here is a quick example using the Spark Shell:

```
>$SPARK_HOME/bin/spark-shell --packages com.couchbase.client:spark-connector_2.10:1.2.0
```

Once in the Shell, you can choose to perform key-value or query-based operations and queries. An example of a simple key-value operation using an RDDs would be as follows:

```
scala> import com.couchbase.client.java.document.JsonDocument
import com.couchbase.client.java.document.JsonDocument

scala> import com.couchbase.client.java.document.json.JsonObject
import com.couchbase.client.java.document.json.JsonObject

scala> sc.parallelize(Seq(JsonDocument.create("mydoc", JsonObject.create().put("hello",
"spark")))).saveToCouchbase()
...
scala> sc.parallelize(Seq("mydoc")).couchbaseGet[JsonDocument]().foreach(println)
...
JsonDocument{id='mydoc', cas=39773408329728, expiry=0, content={"hello":"spark"}, muta-
tionToken=null}
...
```

An example of a simple query operation using DataFrames that takes advantage of the Spark SQL and Couchbase N1QL integration:

```
scala> val airlines = sqlContext.read.couchbase(schemaFilter = org.apache.spark.sql.
sources.EqualTo("type", "airline"))

15/10/20 15:02:51 INFO N1QLRelation: Inferring schema from bucket travel-sample with
query 'SELECT META(`travel-sample`).id as `META_ID`, `travel-sample`.* FROM `trav-
el-sample` WHERE  `type` = 'airline' LIMIT 1000'

...

15/10/20 15:02:52 INFO N1QLRelation: Inferred schema is StructType(StructField(META_
ID,StringType,true), StructField(callsign,StringType,true), StructField(country,String-
Type,true), StructField(iata,StringType,true), StructField(icao,StringType,true),
StructField(id,LongType,true), StructField(name,StringType,true), StructField(-
type,StringType,true))

airlines: org.apache.spark.sql.DataFrame = [META_ID: string, callsign: string, country:
string, iata: string, icao: string, id: bigint, name: string, type: string]
scala> airlines.printSchema

root

 |-- META_ID: string (nullable = true)
 |-- callsign: string (nullable = true)
 |-- country: string (nullable = true)
 |-- iata: string (nullable = true)
 |-- icao: string (nullable = true)
 |-- id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- type: string (nullable = true)

scala> airlines.select("name", "callsign").sort(airlines("callsign").desc).show(5)

...

+----------------+------------+
|      name      |  callsign  |
+----------------+------------+
|   Aws express  |     aws    |
|      Atifly    |    atifly  |
|    XAIR USA    |    XAIR    |
|  World Airways |    WORLD   |
|Western Airlines|   WESTERN  |
+----------------+------------+
```

For detailed information on the Couchbase Spark Connector as well as full examples, please see the Spark Connector Developer Guide.

## Use Cases

Modern applications are using Apache Spark for scalable, high performance analytics and machine learning. Customers are using NoSQL and Couchbase for scalable, low-latency operational database applications. But what kinds of solutions are being deployed today that combine both technologies? Here are just a few examples:

- **Real-time recommendations:** Many eCommerce applications recommend products to shoppers based on various data such as demographic profile, purchase history, web history, geographic location, web session context, etc. Using the connector, Spark applications can analyze this diverse data and pass the results to Couchbase, where the application uses the results to make specific recommendations, promotions, or discounts directly to customers.

- **Component failure detection:** IoT sensor-driven applications are often used to evaluate and predict product or component failures. Using the connector, Spark can apply existing failure data models and complex machine learning algorithms to incoming sensor data (possibly being streamed from Couchbase) and then pass the results to Couchbase, where automated applications can take action based on the results of the analysis.

- **Network intrusion detection:** Network monitoring systems typically store network topology and intrusion history in a low-latency NoSQL database like Couchbase for fast access. Network access logs are often accumulated in a Hadoop Distributed File System (HDFS). Using the connector, Spark processes can read the access logs looking for signs of network intrusion. The Spark processes can execute joins (using DataFrames) or fast data lookups (using RDDs) of network access-point metadata, configuration information, and past network attack models stored in Couchbase and then record the results of that analysis in Couchbase for immediate action.

- **Fraud detection:** Similar to component failure detection and network intrusion detection, fraud detection is about using analytics to identify one or more related events that match an existing model. Using the connector, Spark processes can examine transactions in a real-time data stream or from a historical repository, looking for signs of fraud. These analytical processes can execute joins or fast lookups of customer profile data, vendor information, and known fraud models stored in Couchbase and record the results of that analysis in Couchbase for immediate action.

- **Product and Customer 360:** In order to effectively interact with a customer or provide product recommendations it is necessary to know as much about the product and the customer as possible. A common challenge is that the relevant customer and/or product information is often stored across multiple applications and database repositories of record. Using the connector, Spark processes can aggregate data from multiple sources, thereby building a rich customer and product profile in a centralized "360 degree" repository for use by highly scalable, customer-facing applications and services.

## Conclusion

Modern applications generate massive amounts of operational data. Turning that data into sustainable competitive advantage requires big data analytical processing systems like Spark working together with flexible, scalable NoSQL databases like Couchbase. Deriving the competitive advantage from the raw data is the result of complex analytical and machine learning processes, which are implemented by massively scalable, high performance in-memory processing engines like Apache Spark. Apache Spark, in turn, relies on the performance and scalability of its data sources and targets. Couchbase provides the industry leading high performance, low-latency, scalable NoSQL database. By using the Couchbase Spark Connector, Spark users can accelerate their analytical and machine learning processes, thus reducing the time it takes to produce actionable insights from the raw data. By using Spark, Couchbase users can extend their application capabilities, enhancing their customer experience and increasing their operational efficiency and competitiveness. The Couchbase Spark Connector allows customers to combine these two best-of-breed technologies designed for ease of use and flexible scalability.

## Additional Resources

Couchbase Spark Connector Data Sheet
Couchbase Server Overview
Couchbase Big Data Connectors
Getting Started With the Couchbase Spark Connector
Getting Started With Couchbase Server

## About Couchbase

Couchbase delivers the database for the Digital Economy. Developers around the world choose Couchbase for its advantages in data model flexibility, elastic scalability, performance, and 24x365 availability to build enterprise web, mobile, and IoT applications. The Couchbase platform includes Couchbase, Couchbase Lite - the first mobile NoSQL database, and Couchbase Sync Gateway. Couchbase is designed for global deployments, with configurable cross data center replication to increase data locality and availability. All Couchbase products are open source projects. Couchbase customers include industry leaders like AOL, AT&T, Cisco, Comcast, Concur, Disney, Dixons, eBay, General Electric, Marriott, Nordstrom, Neiman Marcus, PayPal, Ryanair, Rakuten / Viber, Tesco, Verizon, Wells Fargo, as well as hundreds of other household names. Couchbase investors include Accel Partners, Adams Street Partners, Ignition Partners, Mayfield Fund, North Bridge Venture Partners, Sorenson Capital and WestSummit Capital.

2440 West El Camino Real | Ste 600
Mountain View, California 94040

1-650-417-7500

www.couchbase.com