(/)

Spark streaming makes it easy to build scalable fault-tolerant streaming applications.

Ease of use

Build applications through high-level operators.

Spark Streaming brings Apache Spark's language-integrated API (/docs/latest/streaming-programming-guide.html) to stream processing, letting you write streaming jobs the same way you write batch jobs. It supports Java, Scala and Python.

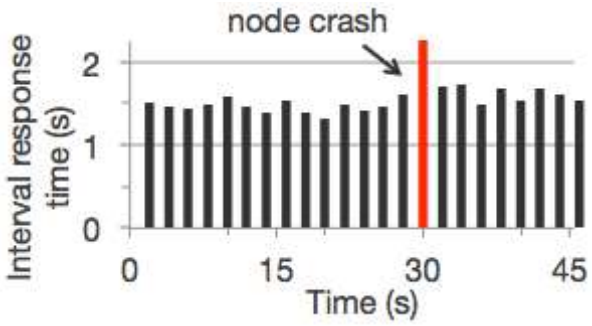
```
TwitterUtils.createStream(...)
    .filter(_.getText.contains("Spark"))
    .countByWindow(Seconds(5))
```

Counting tweets on a sliding window

Fault tolerance

Stateful exactly-once semantics out of the box.

Spark Streaming recovers both lost work and operator state (e.g. sliding windows) out of the box, without any extra code on your part.



Spark integration

Combine streaming with batch and interactive queries.

By running on Spark, Spark Streaming lets you reuse the same code for batch processing, join streams against historical data, or run ad-hoc queries on stream state. Build powerful interactive applications, not just analytics.

```
stream.join(historicCounts).filter {
  case (word, (curCount, oldCount)) =>
    curCount > oldCount
}
```

Find words with higher frequency than historic data

Deployment options

Spark Streaming can read data from HDFS (https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html), Flume (https://flume.apache.org), Kafka (https://kafka.apache.org), Twitter (https://dev.twitter.com) and ZeroMQ (http://zeromq.org). You can also define your own custom data sources.

You can run Spark Streaming on Spark's standalone cluster mode (/docs/latest/spark-standalone.html) or other supported cluster resource managers. It also includes a local run mode for development. In production, Spark Streaming uses ZooKeeper (https://zookeeper.apache.org) and HDFS (https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html) for high availability.

Community

Spark Streaming is developed as part of Apache Spark. It thus gets tested and updated with each Spark release.

If you have questions about the system, ask on the Spark mailing lists (/community.html#mailing-lists).

The Spark Streaming developers welcome contributions. If you'd like to help out, read how to contribute to Spark (/contributing.html), and send us a patch!

Getting started

To get started with Spark Streaming:

- Download Spark (/downloads.html). It includes Streaming as a module.
- Read the Spark Streaming programming guide (/docs/latest/streaming-programming-guide.html), which includes a tutorial and describes system architecture, configuration and high availability.
- Check out example programs in Scala (https://github.com/apache/spark/tree/master/examples/src/main/scala/org/apache/spark/examples/streaming) and Java (https://github.com/apache/spark/tree/master/examples/src/main/java/org/apache/spark/examples/streaming).

DOWNLOAD APACHE SPARK
INCLUDES SPARK STREAMING (/DOWNLOADS.HTML)

Latest News

Spark 3.1.3 released (/news/3-1-3-released.html) (Feb 18, 2022)

Spark 3.2.1 released (/news/spark-3-2-1-released.html) (Jan 26, 2022)

Spark 3.2.0 released (/news/spark-3-2-0-released.html) (Oct 13, 2021)

Spark 3.0.3 released (/news/spark-3-0-3-released.html) (Jun 23, 2021)

Archive (/news/index.html)

(https://www.apache.org/events/current-event.html)

DOWNLOAD SPARK
(/DOWNLOADS.HTML)

Built-in Libraries:

SQL and DataFrames (/sql/)
Spark Streaming (/streaming/)
MLlib (machine learning) (/mllib/)
GraphX (graph) (/graphx/)

Third-Party Projects (/third-party-projects.html)

Spark Streaming

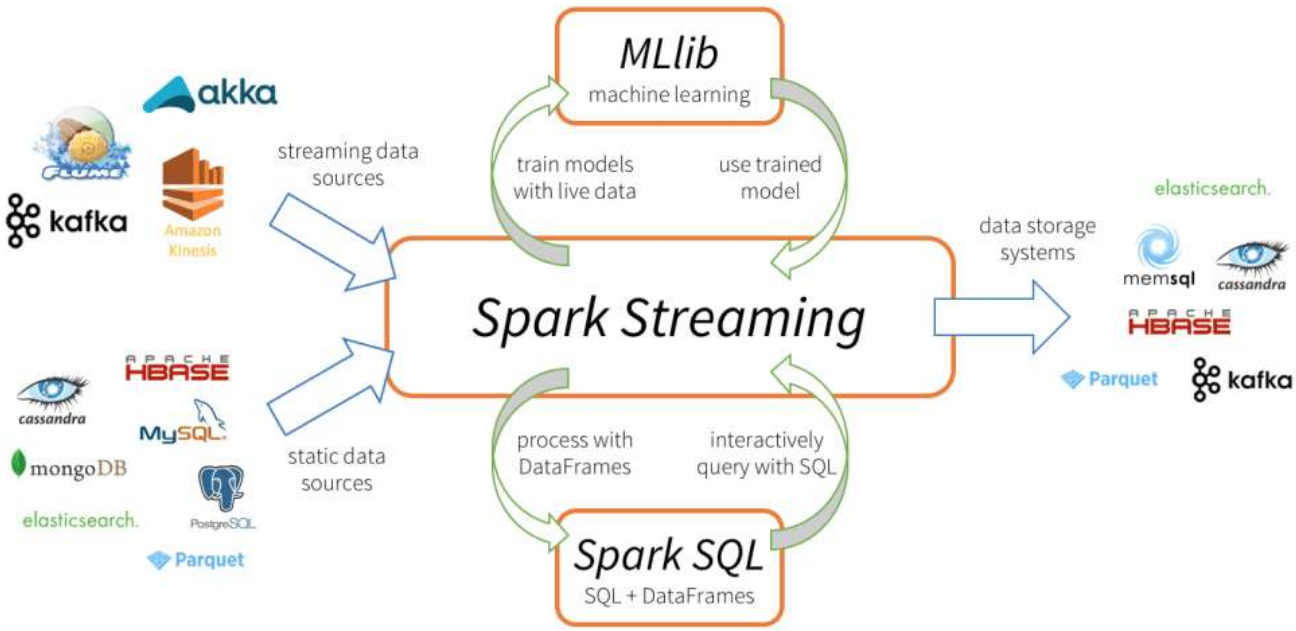
Back to glossary (/glossary/)

What is Spark Streaming?

Apache Spark Streaming is a scalable fault-tolerant streaming processing system that natively supports both batch and streaming workloads. Spark Streaming is an extension of the core Spark API that allows data engineers and data scientists to process real-time data from various sources including (but not limited to) Kafka, Flume, and Amazon Kinesis. This processed data can be pushed out to file systems, databases, and live dashboards. Its key abstraction is a Discretized Stream or, in short, a DStream, which represents a stream of data divided into small batches. DStreams are built on RDDs, Spark's core data abstraction. This allows Spark Streaming to seamlessly integrate with any other Spark components like MLlib and Spark SQL. Spark Streaming is different from other systems that either have a processing engine designed only for streaming, or have similar batch and streaming APIs but compile internally to different engines. Spark's single execution engine and unified programming model for batch and streaming lead to some unique benefits over other traditional streaming systems.

Four Major Aspects of Spark Streaming

- Fast recovery from failures and stragglers
- Better load balancing and resource usage
- Combining of streaming data with static datasets and interactive queries
- Native integration with advanced processing libraries (SQL, machine learning, graph processing)



This unification of disparate data processing capabilities is the key reason behind Spark Streaming's rapid adoption. It makes it very easy for developers to use a single framework to satisfy all their processing needs.

Additional Resources

- **The Data Scientist's Guide to Apache Spark™ eBook (https://databricks.com/p/ebook/the-data-scientists-guide-to-apache-spark)**
- **Structured Streaming Documentation (https://docs.databricks.com/spark/latest/structured-streaming/index.html)**
- **Simplify CDC Pipeline with Spark Streaming SQL and Delta Lake (https://databricks.com/session_na20/simplify-cdc-pipeline-with-spark-streaming-sql-and-delta-lake)**
- **Introducing Apache Spark 3.0: Now available in Databricks Runtime 7.0 (https://databricks.com/blog/2020/06/18/introducing-apache-spark-3-0-now-available-in-databricks-runtime-7-0.html)**

Back to glossary (/glossary/)

TRY DATABRICKS FOR FREE (/TRY-DATABRICKS?ITM_DATA=GLOSSARY-FOOTERCTA-TRIAL)

Product (/product/data-lakehouse)

Learn & Support (/spark/about)

Solutions (/solutions)

Company (https://databricks.com/company/about-us)

(https://databricks.com)

Worldwide

(https://www.linkedin.com/company/databricks)

(https://www.facebook.com/pages/Databricks/560203607379694)

(https://twitter.com/databricks)

(https://databricks.com/feed)

(https://www.glassdoor.com/Overview/Working-at-Databricks-EI_IE954734.11,21.htm)

(https://www.youtube.com/c/Databricks)

Databricks Inc.
160 Spear Street, 15th Floor
San Francisco, CA 94105
1-866-330-0121

Top Apache Spark Use Cases

When it comes to big data tools, Apache Spark is quickly gaining steam both in the headlines and real-world adoption. UC Berkeley's AMPLab developed Spark in 2009 and open-sourced it in 2010. Since then, it has grown to become one of the largest open source communities in big data with over 200 contributors from more than 50 organizations. This open-source analytics engine stands out for its ability to process large volumes of data significantly faster than MapReduce because data is persisted in memory on Spark's own processing framework.

When considering the various engines within the *Hadoop ecosystem*, it's important to understand that each engine works best for certain use cases, and a business will likely need to use a combination of tools to meet every desired use case. That being said, here's a review of some of the top use cases for **Apache Spark**.

1. Streaming Data

Apache Spark's key use case is its ability to process streaming data. With so much data being processed on a daily basis, it has become essential for companies to be able to stream and **analyze it all in real-time**. And Spark Streaming has the capability to handle this extra workload. Some experts even theorize that Spark could become the go-to platform for stream-computing applications, no matter the type. The reason for this claim is that Spark Streaming unifies disparate data processing capabilities, allowing developers to use a single framework to accommodate all their processing needs. Among the general ways that **Spark Streaming** is being used by businesses today are:

- *Streaming ETL* – Traditional ETL (Extract, Transform, Load) tools used for batch processing in data warehouse environments must read data, convert it to a database compatible format, and then write it to the target database. With Streaming ETL, data is continually cleaned and aggregated before it is pushed into data stores.
- *Data Enrichment* – This *Spark Streaming* capability enriches live data by combining it with static data, thus allowing organizations to conduct more complete real-time data analysis. Online advertisers

use data enrichment to combine historical customer data with live customer behavior data and deliver more personalized and targeted ads in real-time and in context with what customers are doing.

- *Trigger Event Detection* – Spark Streaming allows organizations to detect and respond quickly to rare or unusual behaviors ("trigger events") that could indicate a potentially serious problem within the system. Financial institutions use triggers to detect fraudulent transactions and stop fraud in their tracks. Hospitals also use triggers to detect potentially dangerous health changes while monitoring patient vital signs—sending automatic alerts to the right caregivers who can then take immediate and appropriate action.
- *Complex Session Analysis* – Using Spark Streaming, events relating to live sessions—such as user activity after logging into a website or application—can be grouped together and quickly analyzed. Session information can also be used to continuously update machine learning models. Companies such as Netflix use this functionality to gain immediate insights as to how users are engaging on their site and provide more real-time movie recommendations.

2. Machine Learning

Another of the many **Apache Spark** use cases is its machine learning capabilities.

Spark comes with an integrated framework for performing advanced analytics that helps users run repeated queries on sets of data—which essentially amounts to processing machine learning algorithms. Among the components found in this framework is Spark's scalable Machine Learning Library (MLlib). The MLlib can work in areas such as clustering, classification, and dimensionality reduction, among many others. All this enables Spark to be used for some very common big data functions, like predictive intelligence, customer segmentation for marketing purposes, and **sentiment analysis**. Companies that use a recommendation engine will find that Spark gets the job done fast.

Network security is a good business case for **Spark's machine learning** capabilities. Utilizing various components of the Spark stack, security providers can conduct real-time inspections of data packets for traces of malicious activity. At the front end, Spark Streaming allows security analysts to check against known threats prior to passing the packets on to the storage platform. Upon arrival in storage, the packets

undergo further analysis via other stack components such as MLlib. Thus security providers can learn about new threats as they evolve—staying ahead of hackers while protecting their clients in real-time.

3. Interactive Analysis

Among Spark's, most notable features are its capability for interactive analytics. MapReduce was built to handle batch processing, and SQL-on-Hadoop engines such as Hive or Pig are frequently too slow for interactive analysis. However, Apache Spark is fast enough to perform exploratory queries without sampling. Spark also interfaces with a number of development languages including SQL, R, and Python. By combining Spark with visualization tools, complex data sets can be processed and visualized interactively.

Debuting in April or May of this year, the next version of Apache Spark (Spark 2.0) will have a new feature—*Structured Streaming*—that will give users the ability to perform interactive queries against live data. Combining live streaming with other types of data analysis, Structured Streaming is predicted to provide a boost to **Web analytics** by allowing users to run interactive queries against a Web visitor's current session. It could also be used to apply machine learning algorithms to live data. In this scenario, the algorithms would be trained on old data and then redirected to incorporate new—and potentially learn from it—as it enters the memory.

4. Fog Computing

While big data analytics may be getting a lot of attention, the concept that really sparks the tech community's imagination is the **Internet of Things** (IoT). The IoT embeds objects and devices with tiny sensors that communicate with each other and the user, creating a fully interconnected world. This world collects massive amounts of data, processes it, and delivers revolutionary new features and applications for people to use in their everyday lives. However, as the IoT expands so too does the need for distributed massively parallel processing of vast amounts and varieties of a machine and sensor data. All that processing, however, is tough to manage with the current analytics capabilities in the cloud.

That's where **fog computing** and Apache Spark come in.

Fog computing decentralizes data processing and storage, instead of performing those functions on the edge of the network. However, Fog computing brings new complexities to processing decentralized data, because it increasingly requires low latency, massively parallel processing of machine learning, and extremely complex graph analytics algorithms. Fortunately, with key stack components such as Spark Streaming, an interactive real-time query tool (Shark), a machine learning library (MLib), and a graph analysis engine (GraphX), Spark more than qualifies as a fog computing solution. In fact, as the IoT industry gradually and inevitably converges, many industry experts predict that— compared to other open-source platforms— Spark has the potential to emerge as the de facto fog infrastructure.

Spark in the Real World

As mentioned earlier, online advertisers and companies such as Netflix are leveraging Spark for insights and competitive advantage. Other notable businesses also benefiting from Spark are:

- *Uber* – Every day this multinational online taxi dispatch company gathers terabytes of event data from its mobile users. By using Kafka, Spark Streaming, and HDFS, to build a continuous ETL pipeline, Uber can convert raw unstructured event data into structured data as it is collected, and then use it for further and more complex analytics.
- *Pinterest* – Through a similar ETL pipeline, Pinterest can leverage Spark Streaming to gain immediate insight into how users all over the world are engaging with Pins—in real-time. As a result, Pinterest can make more relevant recommendations as people navigate the site and see related Pins to help them select recipes, determine which products to buy or plan trips to various destinations.
- *Conviva* – Averaging about 4 million video feeds per month, this streaming video company is second only to YouTube. Conviva uses Spark to reduce customer churn by optimizing video streams and managing live video traffic—thus maintaining a consistently smooth, high-quality viewing experience.

When NOT to Use Spark

Even though it is versatile, that doesn't necessarily mean Apache Spark's in-memory capabilities are the best fit for all use cases. More specifically, Spark was not designed as a **multi-user environment**. Spark users are required to know whether the memory they have access to is sufficient for a dataset. Adding more users further complicates this since the users will have to coordinate memory usage to run projects concurrently. Due to this inability to handle this type of concurrency, users will want to consider an alternate engine, such as Apache Hive, for large, batch projects.

Over time, Apache Spark will continue to develop its own ecosystem, becoming even more versatile than before. In a world where big data has become the norm, organizations will need to find the best way to utilize it. As seen from these Apache Spark use cases, there will be many opportunities in the coming years to see how powerful Spark truly is.

As more and more organizations recognize the benefits of moving from batch processing to real-time data analysis, Apache Spark is positioned to experience wide and rapid adoption across a vast array of industries

Interested in learning more about Apache Spark, collaboration tools offered with **QDS for Spark**, or giving it a test drive? Click the button to learn more about Apache Spark-as-a-Service.

Learn About Spark