

Exploring Beyond the Known and Ventured

[Home](#) ► (<https://msystechnologies.com/>) [big data](#) ► (<https://msystechnologies.com/category/big-data/>) A Beginner's Guide to Complete Analysis of Apache Spark RDDs and Java 8 Streams

A Beginner's Guide to Complete Analysis of Apache Spark RDDs and Java 8 Streams

(<https://msystechnologies.com/apache-spark-rdd-java-8-streams/>)

By: MSys Editorial

BIG DATA ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/BIG-DATA/](https://msystechnologies.com/category/big-data/)), DATA-ANALYTICS ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/DATA-ANALYTICS/](https://msystechnologies.com/category/data-analytics/)), JAVASCRIPT ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/JAVASCRIPT/](https://msystechnologies.com/category/javascript/))

0 (<https://msystechnologies.com/apache-spark-rdd-java-8-streams/#comments>)

1. What is Apache Spark RDD?

Apache Spark RDD stands for Resilient Distributed Datasets. RDD is a fault tolerant, immutable collection of elements which can be operated on, in parallel. We can perform various parallel operations on them such as map, reduce, filter, count, distinct etc. We can persist them in local memory and perform these operations on them.

RDDs can be created in two ways:

A. `parallelize()`: calling a `parallelize` method on the existing collection in our program (pass collection object to the method).

```
1 JavaRDD<Integer> javaRDD = sparkContext.parallelize(Arrays.asList(1, 2, 3, 4, 5));
```

B. **textFile()**: calling textFile method by passing the path of the file at local or shared file system (pass file URI to the method).

```
1 JavaRDD<String> lines = sparkContext.textFile("URI/to/sample/file.txt");
```

Both methods are called using the reference of **JavaSparkContext** class.

There are two types of operations that can be performed on RDDs:

1. **Transformations**: which perform some operations on RDD to return an RDD (map).
2. **Actions**: which return a value after performing the operation (reduce).

Consider the following example of map and reduce to calculate the total length of the lines in the file, using JavaRDDs:

```
1 JavaRDD<String> lines = sc.textFile("URI/to/sample/file.txt");
2 JavaRDD<Integer> lengths = lines.map(l -> l.length());
3 int totalLength = lengths.reduce((a, b) -> a + b);
```

2. What is Java 8 Streams API?

Java Stream API sounds similar to InputStream and OutputStream in Java IO, but it is completely different, so let's not get confused. Streams are specifically introduced in Java 8 to ease functional programming. Java Streams are **Monads**, a structure that represents computations as a chain of steps.

Streams are the Java APIs that let you manipulate the collections. You can chain together multiple Stream operations to achieve a complex data processing pipeline.

With Streams API, you can write the code that's

- **Declarative**: More concise as well as readable
- **Composable**: Greater flexibility
- **Parallelizable**: Better performance (using Parallel Streams)

Streams can also be created the same way as Spark RDDs

A. Collections as well as Arrays:

```
1 List<String> strings = Arrays.asList("abc", "", "bc", "efg", "abcd", "", "jkl");
2 //get count of empty string
3 int count = strings.parallelStream().filter(string -> string.isEmpty()).count();
```

B. File Systems:

```
1 Stream<String> stream = Files.lines(Paths.get("URI/to/sample/file.txt"));
```

Like RDDs Streams, operations are also of two types:

1. **Intermediate** (like Transformations): which performs some operations on Stream to return a Stream (map).
2. **Terminal**: which returns a value after performing the operation or can be void (reduce, foreach).

```
1 Stream<String> lines = Files.lines(Paths.get("URI/to/sample/file.txt"));
2 Stream<Integer> lineLengths = lines.map(s -> s.length());
3 int totalLength = lineLengths.reduce(0, (a, b) -> a + b);
```

Streams accept **Lambda Expression** as a parameter, which is a **functional interface** that specifies the exact behavior of the operation. The intermediate operations are executed **only** when the terminal operation is called over them. Once a terminal operation is called over a Stream, we cannot reuse it. If we want to use any intermediate operation of a Stream, we have to create a **Stream Supplier** which constructs a new Stream with the intermediate operations. The supplier provides **get()** method to fetch the desired intermediate Stream operation that is already saved.

3. What Can We Do with Spark RDD?

To perform very fast computations over a shared data set such as iterative distributed computing, we need to have an excellent data sharing architecture. This involves processing data using multiple ad-hoc queries and sharing and reusing of data among multiple jobs. To perform these operations, we need to have a mechanism that stores the intermediate data over a distributed data store which may lead to slower processing due to multiple IO operations.

RDDs help us do such operations by breaking the computations into small tasks which run on separate machines. We can cache these RDDs into our local discs to use them in other actions. This helps to execute the future actions much faster.

`persist()` or `cache()` methods help us keep the computed RDDs in the memory of the nodes.

Following properties make RDDs perform best for iterative distributed computing algorithms like K-means clustering, page rank, Logistic regression etc:

- **Immutable**
- **Partitioned**
- **Fault tolerant**
- **Created by coarse grained operations**
- **Lazily evaluated**
- **Can be persisted**

More importantly, all the Transformations in RDDs are lazy, which means their result is not calculated right away. The results are just remembered and are computed just when they are actually needed by the driver program. The Actions, on the other hand are eager.

4. What Can We Do with Java 8 Streams API?

Stream APIs (and of course Lambdas) were introduced in Java 8 considering parallelism as the main driving force. Streams help to write the complex code in concise way which is more readable, flexible and understandable.

Streams can be created from various data sources, like Collections, Arrays, and file resources. Streams are of two types: Sequential and Parallel Streams. We can perform distributed computing operations using multiple threads using Streams.

Parallel Streams can be used to boost the performance of Streams when there is a large amount of input data. Like RDDs, we have methods like map, reduce, collect, flatMap, sort, filter, min, max, count etc. in Streams.

Consider a list of fruits:

```
1 List fruits = Arrays.asList("apple", "orange", "pineapple", "grape", "banana", "mango", "blackberry");
```

Filter()

```
1 fruits.filter( fruit -> fruit.startsWith("b") );
```

Map()

```
1 fruits.map( fruit -> fruit.toUpperCase() )
```

Collect()

```
1 List filteredFruits = fruits.filter( fruit -> fruit.startsWith("b") )
2                        .collect(Collectors.toList());
```

Min() and Max()

```
1 String shortest = fruits.min(Comparator.comparing(fruit -> fruit.length())).get();
```

Count()

```
1 long count = fruits.filter( fruit -> fruit.startsWith("b")).count();
```

Reduce()

```
1 String reduced = fruits.filter( item -> item.startsWith("b"))
2                  .reduce("", (acc, item) -> acc + " " + item);
```

5. How Are They Same?

- RDDs and Streams can be created the same way:
 1. from Collections and
 2. File Systems.
- RDDs and Streams perform two (same) types of operations:
 1. Transformations in RDDs == Intermediate in Streams
 2. Actions in RDDs == Terminal in Streams
- Transformations (RDDs) and Intermediate (Streams) have the same important characteristic i.e. Laziness. They just remember the transformations instead of computing them unless an Action is needed. While Actions (RDDs) and Terminal (Streams) operations are eager Operations.
- RDDs and Streams help in reducing the actual number of operations performed over each element as both use Filters and Predicates.
- Developers can write much concise code using RDDs and Streams.
- RDDs and (parallel) Streams are used for parallel distributed operations where a large number of data processing is required.
- Both RDDs and Streams work on the principle of Functional Programming and use lambda Expressions as parameters.

6. How Are They Different?

1. Unlike RDDs, Java 8 Streams are of two types: Sequential and Parallel.
2. Parallelization is just a matter of calling parallel() method in Streams. It internally utilizes the Thread Pool in JVM, while Spark RDDs can be distributed and deployed over a cluster.
3. While Spark has different storage levels for different purposes, Streams are in memory data structures.

When you call parallelize method in Streams, your data is split into multiple chunks and they are processed independently. This process is CPU intensive and utilizes all the available CPUs. The Java parallel Streams use a common ForkJoinPool. The capacity of this ThreadPool to use Threads depends on the number of available CPU cores. This value can be increased or decreased using the following JVM parameter,

```
1 -Djava.util.concurrent.ForkJoinPool.common.parallelism=5
```

So for executing the parallel Stream operations the Stream utilizes all the available threads in the ThreadPool. Moreover, if you submit a long running task, this could result in blocking all the other threads in the ThreadPool. One long task could result into blocking of entire application.

7. Which One Is Better? When?

Though RDDs and Streams provide quite similar type of implementations, APIs and functionalities, **Apache Spark is much more powerful than Java 8 Streams**. While it's completely our choice what to use when, we should always try to analyze our requirements and proceed for the implementations.

As **parallel Streams use a Common Thread Pool**, we must ensure that there **won't be any long running task** which will cause other threads to stuck. **Apache Spark RDDs will help you distribute the data over cluster**. When there are such complex situations which involve a real huge amount of data and computations, we should avoid using Java 8 Streams.

So for non-distributed parallel processing my choice would be to go with Streams, while Apache Spark RDDs would be preferable for real time analytics or continuously streaming data over distributed environments.

Tags In

APACHE ([HTTPS://MSYTECHNOLOGIES.COM/TAG/APACHE/](https://msystechnologies.com/tag/apache/))

APACHE SPARK RDD ([HTTPS://MSYTECHNOLOGIES.COM/TAG/APACHE-SPARK-RDD/](https://msystechnologies.com/tag/apache-spark-rdd/))

APACHE8 ([HTTPS://MSYTECHNOLOGIES.COM/TAG/APACHE8/](https://msystechnologies.com/tag/apache8/))

JAVA 8 STREAMS ([HTTPS://MSYTECHNOLOGIES.COM/TAG/JAVA-8-STREAMS/](https://msystechnologies.com/tag/java-8-streams/))



(https:	(https:
//ms	//ms
ystec	ystec
hnolo	hnolo
gies.c	gies.c
om/e	om/e
mbed	mbed
ded-	ded-
linux-	linux-
part-	part-
1-	2-
bootl	linux-
oader	kerne
/)	/)


Leave a Reply

Message 

Name 

Email 

Website 

Required fields are marked 

Send A Comment

Search here...



Subscribe for latest update

☐ Newsletter ☐ Blogs

EMAIL ADDRESS *

Example: john@gmail.com

Subscribe

Archive

+ 2018

+ 2017

+ 2016

+ 2015

+ 2014

+ 2013

+ 2012

+ 2011

Categories

AGILE ([HTTPS://MSYTECHNOLOGIES.COM/CATEGORY/AGILE/](https://msystechnologies.com/category/agile/))

AMAZON EC2 ([HTTPS://MSYTECHNOLOGIES.COM/CATEGORY/AMAZON-EC2/](https://msystechnologies.com/category/amazon-ec2/))

APPLICATIONS ([HTTPS://MSYTECHNOLOGIES.COM/CATEGORY/APPLICATIONS/](https://msystechnologies.com/category/applications/))

AUTOMATION ([HTTPS://MSYTECHNOLOGIES.COM/CATEGORY/AUTOMATION/](https://msystechnologies.com/category/automation/))

BIG DATA ([HTTPS://MSYTECHNOLOGIES.COM/CATEGORY/BIG-DATA/](https://msystechnologies.com/category/big-data/))

CHEF ([HTTPS://MSYTECHNOLOGIES.COM/CATEGORY/CHEF/](https://msystechnologies.com/category/chef/))

CLOUD ([HTTPS://MSYTECHNOLOGIES.COM/CATEGORY/CLOUD/](https://msystechnologies.com/category/cloud/))

CLOUD COMPUTING ([HTTPS://MSYTECHNOLOGIES.COM/CATEGORY/CLOUD-COMPUTING/](https://msystechnologies.com/category/cloud-computing/))

CLOUDSTACK ([HTTPS://MSYTECHNOLOGIES.COM/CATEGORY/CLOUDSTACK/](https://msystechnologies.com/category/cloudstack/))



CUSTOM-COMPONENTS ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/CUSTOM-COMPONENTS/](https://msystechnologies.com/category/custom-components/))

DATA-ANALYTICS ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/DATA-ANALYTICS/](https://msystechnologies.com/category/data-analytics/))

DATA-SECURITY ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/DATA-SECURITY/](https://msystechnologies.com/category/data-security/))

DATA-STORAGE ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/DATA-STORAGE/](https://msystechnologies.com/category/data-storage/))

DATACENTER ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/DATACENTER/](https://msystechnologies.com/category/datacenter/))

DEVELOPMENT ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/DEVELOPMENT/](https://msystechnologies.com/category/development/))

DEVOPS ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/DEVOPS/](https://msystechnologies.com/category/devops/))

DOCKER ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/DOCKER/](https://msystechnologies.com/category/docker/))

EMBEDDED SYSTEMS ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/EMBEDDED-SYSTEMS/](https://msystechnologies.com/category/embedded-systems/))

END-TO-END-TEST-AUTOMATION ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/END-TO-END-TEST-AUTOMATION/](https://msystechnologies.com/category/end-to-end-test-automation/))

HADOOP ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/HADOOP/](https://msystechnologies.com/category/hadoop/))

INFOGRAPHIC ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/INFOGRAPHIC/](https://msystechnologies.com/category/infographic/))

INNOVATION ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/INNOVATION/](https://msystechnologies.com/category/innovation/))

INTERNET-OF-THINGS ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/INTERNET-OF-THINGS/](https://msystechnologies.com/category/internet-of-things/))

IOT ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/IOT/](https://msystechnologies.com/category/iot/))

JAVASCRIPT ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/JAVASCRIPT/](https://msystechnologies.com/category/javascript/))

LINUX ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/LINUX/](https://msystechnologies.com/category/linux/))

MACHINE-LEARNING ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/MACHINE-LEARNING/](https://msystechnologies.com/category/machine-learning/))

MOBILE-APPLICATION-TESTING ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/MOBILE-APPLICATION-TESTING/](https://msystechnologies.com/category/mobile-application-testing/))

MOBILE-AUTOMATION-TESTING-TOOL ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/MOBILE-AUTOMATION-TESTING-TOOL/](https://msystechnologies.com/category/mobile-automation-testing-tool/))

NEWS ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/NEWS/](https://msystechnologies.com/category/news/))

OPSCODE-2 ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/OPSCODE-2/](https://msystechnologies.com/category/opscod e-2/))

OTHERS ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/OTHERS/](https://msystechnologies.com/category/others/))

QA ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/QA/](https://msystechnologies.com/category/qa/))

QUALITY-ASSURANCE ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/QUALITY-ASSURANCE/](https://msystechnologies.com/category/quality-assurance/))

SECURITY ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/SECURITY/](https://msystechnologies.com/category/security/))

SOFTWARE ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/SOFTWARE/](https://msystechnologies.com/category/software/))

SOFTWARE-TESTING ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/SOFTWARE-TESTING/](https://msystechnologies.com/category/software-testing/))

STORAGE ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/STORAGE/](https://msystechnologies.com/category/storage/))

TECHNOLOGY ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/TECHNOLOGY/](https://msystechnologies.com/category/technology/))

TEST-AUTOMATION ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/TEST-AUTOMATION/](https://msystechnologies.com/category/test-automation/))

TRENDS ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/TRENDS/](https://msystechnologies.com/category/trends/))

TUTORIALS ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/TUTORIALS/](https://msystechnologies.com/category/tutorials/))

UNCATEGORIZED ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/UNCATEGORIZED/](https://msystechnologies.com/category/uncategorized/))

VIRTUALIZATION ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/VIRTUALIZATION/](https://msystechnologies.com/category/virtualization/))

WIRELESS ([HTTPS://MSYS TECHNOLOGIES.COM/CATEGORY/WIRELESS/](https://msystechnologies.com/category/wireless/))



MSys technologies is an innovator in offering technology services and domain-specific test automation software (for both desktop and mobile apps).

[Read More >> \(/about-us/\)](#)

Follow Us:   

Quick Links

News & Events (<https://msys technologies.com/news-events/>)

© 2017 MSys Technologies. All rights reserved.

Blogs (<https://msys technologies.com/blogs/>)

[Home \(/\)](#) [About Us \(/About-Us\)](#) [Contact Us \(/Contact-Us\)](#) [Privacy Policy](#) [Sitemap \(/Site-Map\)](#)

[Newsletters \(https://msys technologies.com/newsletters/\)](https://msys technologies.com/newsletters/)

[Careers \(https://msys technologies.com/careers/\)](https://msys technologies.com/careers/)