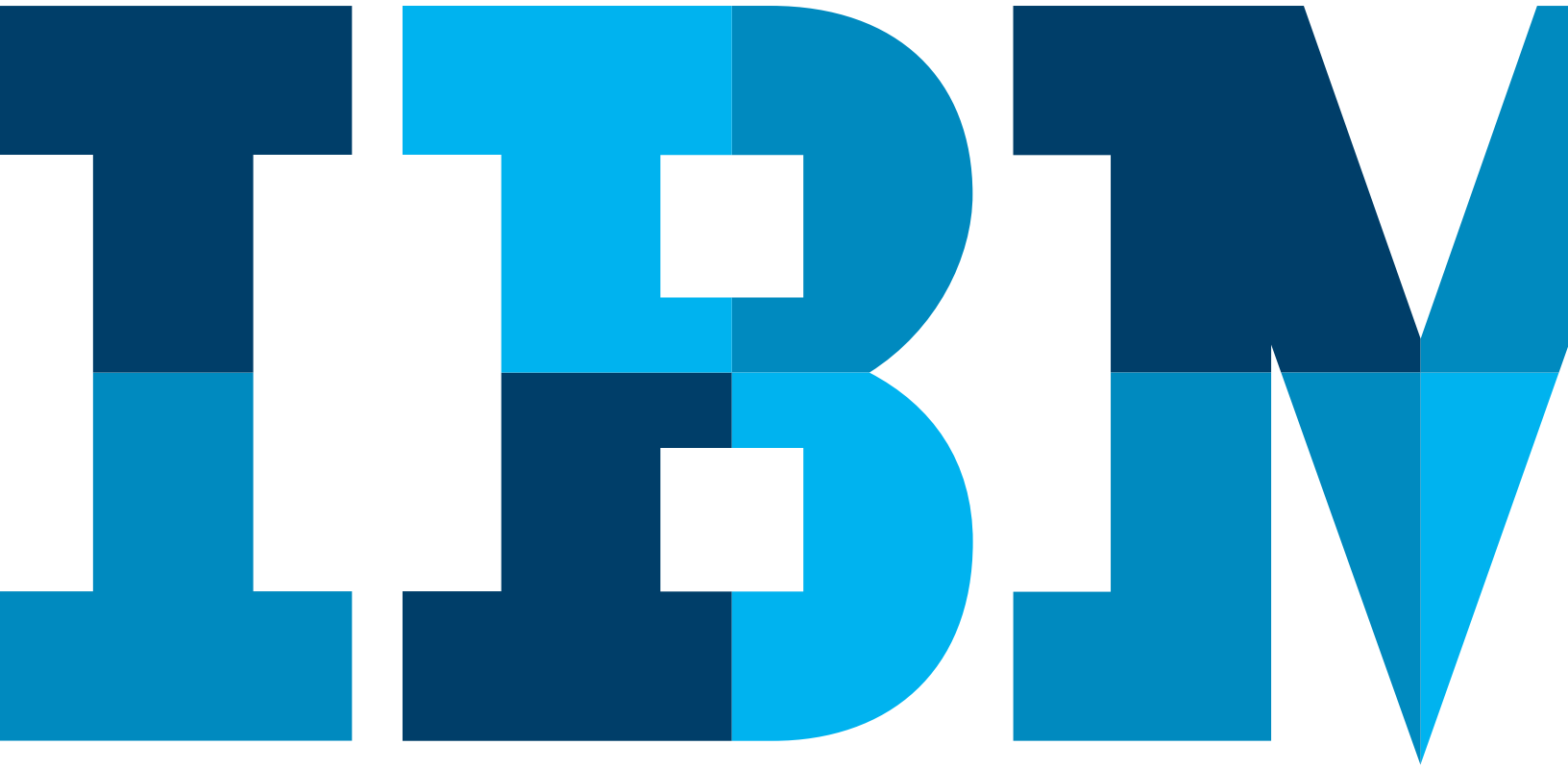
IBM Data Science Experience
White paper

PySpark

High-performance data processing without learning Scala



IBM

Executive summary

This white paper discusses the advantages of using the PySpark API, which enables the use of Python to interact with the Spark programming model. For programmers already familiar with Python, the PySpark API provides easy access to the extremely high-performance data processing enabled by Spark's Scala architecture — without the need to learn any Scala.

The PySpark API allows data scientists with experience of Python to write programming logic in the language most familiar to them, use it to perform rapid distributed transformations on large sets of data, and get the results back in Python-friendly notation.

PySpark is likely to be of particular interest to users of the "pandas" open-source library, which provides high-performance, easy-to-use data structures and data analysis tools. pandas enables an entire data analysis workflow to be created within Python — rather than in an analytics-specific language such as R — but it is usually limited to running locally and with relatively small data sets. If a data scientist has established a set of pandas-based operations but needs to tackle a much larger data set, PySpark makes it easy to take advantage of distributed compute resources in the cloud, and then bring back a smaller subset of data for further local analysis — all within Python. And because PySpark supports all standard Python libraries and even C extensions, existing code can take advantage of the power of Spark with only minimal modifications.

The paper concludes with an invitation for readers to try IBM Data Science Experience, which provides an all-in-one, web-based platform for data scientists, enabling the easy use of PySpark and other open-source tools running on robust cloud resources.

Spark — what it is and why it's great news for data scientists

Apache Spark is an open-source processing engine built around speed, ease of use, and analytics. Developed to utilize distributed, in-memory data structures to improve data processing speeds for most workloads, Spark performs up to 100 times faster than Hadoop MapReduce for iterative algorithms or interactive data mining. It also supports Java, Scala, and Python APIs for ease of development.

In functional terms, Spark combines SQL, streaming and complex analytics seamlessly in the same architecture to handle a wide range of data processing scenarios. This removes the cost and complexity of maintaining three different stacks, ensures that metrics are consistent across stacks, and makes it faster and easier to share data.

PySpark can work with data in a distributed storage system — for example, HDFS — and it can also take local data and parallelize it across the cluster to accelerate computations. With the ability to compute in real-time, Spark can enable faster decisions — for example, identifying why a transactional website is running slowly so that it can be fixed before financial losses are incurred. Similarly, for queries on streaming data, Spark's real-time processing can help organizations detect financial fraud or distributed denial of service (DDoS) attacks.

Spark offers rich APIs in Java, Scala and Python, making it easy to develop analytical applications, and is available from IBM as a fully managed service on the cloud.

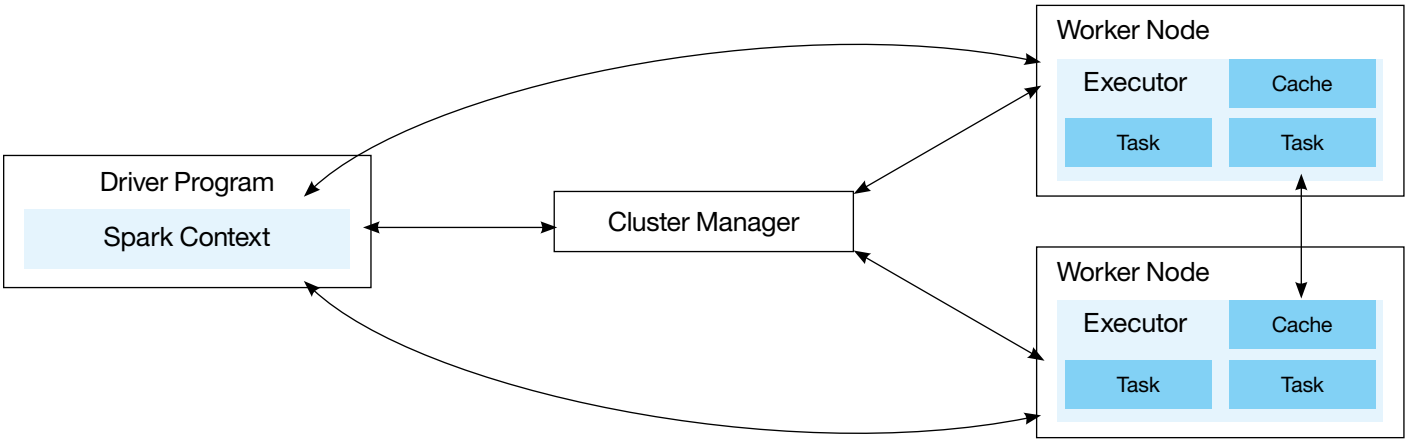


Figure 1: Spark architecture. Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in your main program (called the driver program). From https://spark.apache.org/docs/latest/cluster-overview.html.

What is PySpark, and what benefits does it offer?

One of the major strengths of Spark is its compatibility with multiple different programming languages. For data scientists and analysts familiar with Python, the PySpark API makes it easy to write code that takes advantage of Spark to deliver dramatic improvements in processing speed for large sets of data. Full support for C extensions in Python means that data scientists will typically not need to make major changes to their Python code in order to use Spark.

PySpark transformations (such as *map*, *flatMap*, *filter*) return resilient distributed datasets (RDDs), while actions generally return either local Python values or write the results out. Behind the scenes, PySpark's use of the Py4J library is what enables Python to make Java calls directly to Java Virtual Machine objects—in this case, the RDDs. In the usual way, short functions are passed to RDD methods using Python's lambda syntax, while longer functions are defined with the *def* keyword. PySpark automatically ships the requested functions to worker nodes (see architectural diagram above) together with any objects to which they refer. Python functions and closures are serialized using the CloudPickle module, so that they can be shipped to the cluster without any special syntax. The worker nodes then run the Python processes and push the results back to SparkContext, which stores the data as pickled (serialized) objects in the RDD. To provide resiliency, Spark automatically re-computes the results for any node that crashes, using the relevant input shards in the original input data.

PySpark also offers access via an interactive shell, providing a simple way to learn the API (which is also extensively documented). By default, the bin/PySpark shell creates a SparkContext that runs applications locally and on a single core. Users can set the MASTER environment variable to use multiple cores locally or connect to non-local clusters.

The use of "lazy evaluations"—whereby transformations (for example: *map*, *filter*, *flatMap*, *reduceByKey*) are not executed until required by an Action (for example: *count*, *reduce*, *saveAs*)—enables procedures to be pipelined and avoids materializing large intermediate results. This means that data scientists can construct complex analyses in stages and minimize the number of passes over the data.

When is it appropriate to use PySpark rather than pandas?

pandas is an open-source library that provides data structures and analysis tools for Python. pandas includes the ability to read and write data between in-memory data structures, to flexibly reshape and pivot data sets, to slice, index and subset large data sets, to perform fast merging and joining of datasets, and to perform hierarchical axis indexing. In practice, this means that data scientists familiar with Python can typically execute their entire data analysis workflow without needing to switch to a specialist language. In turn, this frees up time previously used to learn programming languages to spend on analysis instead, improving productivity and supporting faster insight.

pandas is an excellent tool for analyzing data on a local workstation or laptop, but it is naturally less well adapted to very large datasets. In some cases, an acceptable compromise is to downsample the dataset to create a representative sample small enough to deliver fast results on local resources. However, downsampling is not always appropriate. For example, when building and evaluating recommendation systems, or when training a machine-learning system, it is usually best to have as much raw data as possible. As another example, a researcher may want to augment an existing dataset with very large datasets in the cloud, such as meteorological and climatological data. Equally, a researcher may need to query a large dataset in order to produce the smaller subset of data on which they actually want to perform local analysis in pandas.

For these kinds of requirements, data scientists will benefit from the distributed processing power of Spark. And with PySpark, the workflow for accomplishing this becomes relatively simple. Data scientists can build an analytical application in Python, use PySpark to aggregate and transform the data, then bring the consolidated data back as a DataFrame in pandas. Reprising the example of the recommendation system, PySpark would be used for the creation and evaluation stages, but a task like drawing a heat map to show how well the model predicted people's preferences could be performed more economically using local resources.

The key point is that PySpark will not necessarily replace the pandas library, but rather supplement it by enabling data scientists to flex up to distributed processing when appropriate, and flex back down to local resources when the datasets are appropriately sized. This hybrid approach also keeps control of the costs, which are naturally higher for a large computational cluster versus a single local machine. On this note, users may choose to build a local Spark instance running on a single machine or small cluster, optimize their PySpark code using small amounts of test data, then flex up to a large cloud-based cluster once they are confident that it will run within the constraints of their budget. When running locally, Spark simulates distributed calculations by virtually partitioning the local machine's memory.

IBM Data Science Experience, which provides an all-in-one web-based platform for data scientists, makes it easy to use PySpark on cloud resources without needing to set up clusters manually. This avoids the potential headaches involved in getting corporate IT functions to set up the hardware with the correct software and tools—IBM Data Science Experience will automatically provide fully functional remote clusters so that data scientists can simply fire up the resources they want when they need them.

Get up and running with PySpark

By making it easy to combine local and distributed data-transformation operations, PySpark can significantly accelerate analysis while keeping control of computing costs. It also enables data scientists to avoid always having to downsample large data sets. For tasks such as training a machine-learning system or building a recommendation system, using a full set of data can make a significant difference to the final quality. Taking advantage of distributed processing can also make it easier to augment existing data sets with other types of data—for example, combining share-price data with weather data.

For more information

To get started with pySpark, please try IBM Data Science Experience, which provides a single point of access and control across multiple open-source data-transformation and analytics tools running on robust cloud resources. For more information, please visit datascience.ibm.com

IBM Corporation
Route 100
Somers, NY 10589

Produced in the United States of America
December 2016

Please Recycle

CDW12366-USEN-00