The driving force of science and trends in its development

Rostyslav Tukalo
student
Kharkiv National University of Radioelectronics, Ukraine

Olga Gubareva
PhD, Associate Professor
Kharkiv National University of Radioelectronics, Ukraine

USING THE .NET PLATFORM FOR BIG DATA PROCESSING

With the development of computer technology, the problem of efficient processing and intelligent analysis of large data sets has an important place among others in this area. The main and most common interpretation defines this term as sets of structured or unstructured information, so large that classical methods of processing and studying data do not correspond to the volume of these arrays. There is also the following definition: the phenomenal acceleration of the rate of increase in the amount of data and its complexity (for example, it is believed that the volume of information in the world grows by 30% every year, and in the last five years humanity has accumulated more knowledge and data than in its entire previous history). It is obvious that the information technology industry faces the question of understanding and covering the entire mass of information, which grows exponentially over time, and therefore the challenges of the future require modern, powerful and effective technical solutions for obtaining knowledge in almost any area of human life.

In the big data community, there are obviously already several basic software tools that are common among data professionals and are well-established for use. In particular, there are the Python, R and Scala programming languages. What they have in common is that these languages are multi-paradigmatic (combining functional and object-oriented programming), and code is interpreted to execute programs (although Scala provides the ability to use both a compiler and an interpreter). The use of these exact languages is due to their high performance, as well as the availability of a large number of add-ons and libraries for powerful and flexible information processing.

Now let's get acquainted with the .NET platform. One of the main ideas of this software technology is the possibility of interaction of code and services between each other written in different programming languages - this aspect is provided by means of a Common Language Runtime environment, CLR, which executes CIL or MSIL bytecode on a given processor, which, in turn, all .NET-compatible languages are converted to (the most common are C#, F#, VB.NET). Other components of the platform belong to the development toolkit - among other things, these are data access technologies, class libraries, parallel execution tools. .NET is a cross-platform technology that allows applications to run on operating systems other than Windows.

The main application areas of the .NET platform include server application development, APIs, game application development (with the help of the Unity engine) and mobile application development (using .NET MAUI technology, which extends .NET with class libraries and development tools for Android and iOS). However, recently, the list of possibilities for using .NET has been expanded by another field - big data processing, which we will explore in this article.

In October 2020, the release of the first stable version of the .NET framework for Apache Spark was announced. Apache Spark is also a big data framework consisting of a core, a tool for analytical data processing using SQL queries, a module for distributed graph processing, and other subsystems. This technology essentially provides APIs for different programming languages, for which it is possible to create custom frameworks to use Apache Spark's computational and
analytical capabilities. At the time .NET support was introduced, the main languages supported were R, Python, Scala, and Java.

The prerequisite for creating a framework for interaction between .NET and Apache Spark was a request from the .NET community for an easier way to develop big data applications on the .NET platform, to avoid the need to learn Scala or Python for this purpose. Therefore, a team was created to write an open source project. The very first public version was presented in April 2019, since then there have been 12 pre-releases until the release of the first stable version. Thanks to the flexibility and modularity of the .NET platform, developers have the opportunity to supplement other parts of their projects and applications with the code of big data systems, as well as to freely use their knowledge, experience, code, as well as interact with any existing libraries; as an example, we can cite the machine learning library ML.NET, which, in combination with Spark, can become a strong competitor to other established methods and software systems for developing highly intelligent applications.

The developers claim that .NET for Apache Spark is faster than competitors for the Python and Scala languages, and in some cases the PySpark framework for Python loses in performance by a factor of two. Let's try to conduct our own assessment of the potential of frameworks for .NET and Python. For this, we will use one of the modules provided by Apache Spark, namely, a tool for analytical processing using SQL queries. We will perform the task of counting the number of words in a given text file. In our case, we will use the text of Winston Churchill's speech (https://en.wikisource.org/wiki/We_shall_fight_on_the_beaches), which was repeated 10000 times (which gets us a file with 37920000 words, 200 megabytes).

In both programs, the algorithm is as follows: start a Spark session, meaning to connect the program code through the framework to the main tools and capacities on the computer, start time measurement, load the text from the test file, count the number of words using the SQL tool, display the result (20 most frequent words) to the console, stop the time measurement and display the measurement result, terminate the Spark session. Therefore, the number and content of the steps in the programs, not taking into account the technical differences of each language, are absolutely identical.

Let's see the code of the method that performs the calculation in C#:

```csharp
// Creating Spark session
SparkSession spark = SparkSession
    .Builder()
    .AppName("word_count_sample")
    .GetOrCreate();
// Starting time measurement
Stopwatch stopWatch = new Stopwatch();
stopWatch.Start();
// Creating DataFrame object
DataFrame dataFrame = spark.Read().Text("./Churchill.txt");
// Calculating the result
DataFrame words = dataFrame
    .Select(Split(Col("value"), " ").Alias("words"))
    .Select(Explode(Col("words")).Alias("word"))
    .GroupBy("word")
    .Count()
    .OrderBy(Col("count").Desc());
// Print the result
words.Show();
// Print the time consumed
stopWatch.Stop();
TimeSpan ts = stopWatch.Elapsed;
Console.WriteLine("RunTime " + ts);
```
In the same way, we will give the program code in Python. Since the same framework is indirectly used, the signatures of the functions used are often similar so the similarity of the code can be seen.

```python
spark = SparkSession \
.builder \
.appName("word_count_sample") \
.getOrCreate()
start = time.time()
words = spark.read.text("./Churchill.txt")
words.withColumn('words', split(col('value'), ' ')) \
.withColumn('word', explode(col('words'))) \
.groupBy('word') \
.agg(count('word').alias('count')) \
.orderBy('count', ascending=False) \
.show(20)
print("RunTime", time.time() – start)
spark.stop()
```

After running both programs (obviously on the same machine for greater accuracy of comparison, but for the C# language, Release mode was used to optimize the program without using unnecessary debugging tools) the following results were obtained. The programs got the same results, but the execution time is slightly different in favor of the Python language - the .NET platform performed this task on average 12-20% slower.

![Fig. 1. The result of counting the number of words and measuring the running time of the program for the .NET platform](image)

100
Fig. 2. The result of measuring the running time of the program for the Python language

So, despite the claims of the framework developers, our benchmark revealed a slight advantage of the Python language over the .NET platform and the programming language in our case, C#. Objectivity, compared to professional systems for evaluating the performance of platforms and frameworks, is of course lower, but we can state the following: the prospect of using .NET as the main tool in the hands of a big data analyst is quite real, and even more so in connections to other application development tools such as machine learning libraries that already exist. It is necessary to pay attention to a certain difference in the time efficiency of different languages in certain usage situations, but this aspect can be justified by the fact that it is possible to solve big data problems with the participation of .NET specialists without the need for them to learn Python or Scala or hire new employees with such skills.

References: