.NET CORE TECHNOLOGY IN SCIENTIFIC TASKS

V.A. Dorokhin

Dubna State University, 19 Universitetskaya st., Dubna, 141982, Russia

E-mail: victor.doroh@gmail.com

Today we have an established stack of tools to develop applications, systems and services for scientific purposes. However, not so long ago a new technology, called .NET Core appeared. It's supervised by Microsoft, but its open-source and developed by a wide community of programmers and engineers. That technology has a lot of advantages like high performance, simple and productive parallel programming abilities, support of high-level programming languages (C# 7.0, F#) and so on.But the main advantage in comparison with its predecessors is cross-platform abilities. Once written code can be natively compiled on a large number of platforms and hardware systems. It can be used on Windows, Linux, Mac and all Unix-based operation systems. It supports different hardware components and processors, for example, it can be used on ARM processors. The technology supports containerization and can be used with Docker or Kubernetes. It gives an ability to develop applications with micro-service architecture out of the box and provides convenient deployment tools. Now, in a set of tasks .NET Core surpasses currently used tools in scientific sphere tools like Python, Go, Ruby, Java, Node.JS and others. Also, it can be used to develop native desktop applications with HTML-based GUI for administration and monitoring purposes.

Keywords: .NET, .NET Core, interfaces, scientific infrastructure, programming, framework, computations

1. Current Approaches

Today, development stack for all task groups is already formed and changes a bit. Scientific sphere is not an exception. All scientific tasks in computer area can be divided in two big groups: calculations and infrastructure. And as a rule, both groups use common stack of technologies to solve their tasks.

Almost always, calculations are performed using the following programming languages: C, C++, Fortran, Lisp. Now it's not an alternative for GPU computation. And still the old languages are used, because a lot of software are already developed and tested on it. Also, one of the most important features of programming language for those tasks are performance and parallelism.

For infrastructure goals scientists use a wide number of technologies, languages and frameworks. The most common are: Python, Go, Ruby, Java, Node.JS, JS, PHP, Rust, R, etc. Popular tasks are monitoring, communication, storage, administration, visualization, distribution, etc. The main requirements are reliability and free licensing.

2. .NET Core

.NET Core is a free, cross-platform, open source developer platform for building many different types of applications. It gives an opportunity to use multiple languages, editors, and libraries to build for web, mobile, desktop, gaming, and IoT. [1]

It has two major components. It includes a small runtime that is built from the same codebase as the .NET Framework CLR. The .NET Core runtime includes the same GC and JIT (RyuJIT), but doesn't include features like Application Domains or Code Access Security. The runtime is delivered via NuGet, as part of the package. [3]

.NET Core also includes the base class libraries. These libraries are largely the same code as the .NET Framework class libraries, but have been factored (removal of dependencies) to enable us to ship a smaller set of libraries. These libraries are shipped as System NuGet packages on NuGet.org. .NET Core technology inside .NET ecosystem is performed in the Figure 1.

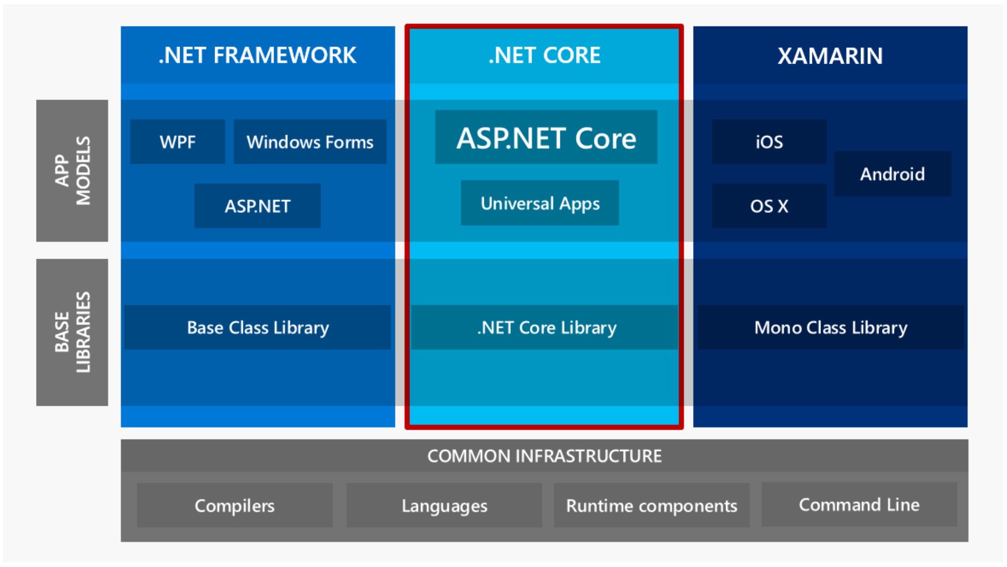

Figure 1. .NET Core in Microsoft ecosystem

In conclusion, .NET Core is a new Microsoft technology developed in open-source, which preserved a lot of ancestors' advantages and it's community; got a lot of new features and improvements in infrastructure, performance, integration; and increased its habitat to new platforms and operation systems.

3. Advantages

.NET Core has a lot of advantages in comparison with previous versions and competitors. The main of them are described below.

Cross-platform

New technology can be deployed to different platforms and operation systems. Now Windows, Mac and Linux-like systems are supported. Also, it can be used in IoT tasks. For example, technology can be used on ARM processors, for instance, on Raspberry PI computer.

Native\self-hosted distribution

It is a couple of ways to distribute your application. It can be published as a library, which is emulated with usage of runtime on the specific machine. The second way is self-hosted application. It means that your application is built with runtime integrated in it and you can run it without any preparations and additional software. And in the future, as developers say, it will be an availability to compile applications native. Now, that feature can be used only for windows systems.

Containerization

System is optimized to work inside Docker containers and provides a wide opportunities for developing applications using micro-service architecture.

Multilanguage solution

One of the features is inherited from parent technology. There are three main programming languages in .NET: C#, F# and VB.NET. Also they are used in different spheres and on different platforms. And common .NET solution gives an ability to use them all at once. So, developer can write some specific code in one language and use it from another project with different language. And all of that is available in one solution.

Frameworks and libraries

There are a lot of useful and powerful libraries and frameworks for that technology. The main are Entity Framework ORM, ASP.NET web applications engine, ML.NET framework for Artificial Intelligence, Q# for quantum computing, SignalR for real-time communication, etc.

Performance

.NET Core's optimization and performance of concrete program depends on the number of packages developer used. Now, in comprehension with other cross-platform technologies .NET Core has one of the best performance scores, except C++. But now in the areas where it can't be used it has one of the leading positions. For example, in requests per second. Statistics is shown in the Figure 2.

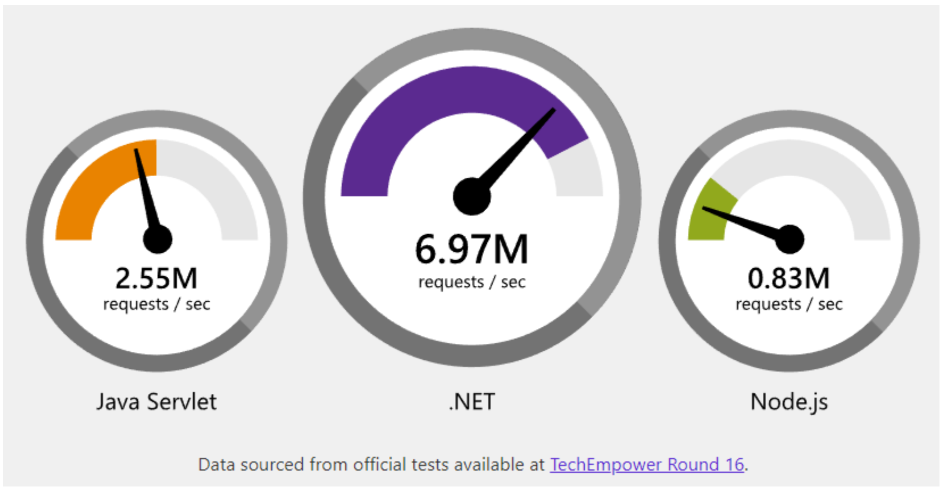

Figure 2. Requests per second benchmark [2]

Architecture

One of the main advantages of .NET stack is a high level of the language and really impressive tools for building complicated architecture and high scalable solutions. In .NET Core there is a built-in dependency injection tool and many other pre-built patterns.

Parallelism

Languages used in technology give a great functionality in parallel cases. Developer can work with additional threads in different ways. It can be performed directly or via using Tasks abstraction of TPL library. There are parallel language constructions, like *Parallel.For* and concurrent implementations of data structures.

Development tools

At the present day, described development stack has one of the best IDEs. Also there are a lot of useful tools and libraries, cloud services, built-in version control systems, modern debugging and profiling tools. Unix users can use Visual Studio Code and .NET Core CLI.

4. Conclusion

Tools, used by scientific community to solve their infrastructure tasks and sometimes even calculation are getting out of date. In some areas there are better and more performant tools.

Now, a stable version of the new, promising technology from Microsoft – .NET Core appeared. This system solves most problems of predecessors, it is better optimized, supports a huge ecosystem of ready-made solutions, and most importantly – it is cross-platform.

It is much more performant, scalable and perspective. That technology is open-source and it is developed and contributed by Microsoft. Today, there are a couple of scientific tasks, where it can be used better, than current solutions. For example, monitoring, storage and collection of data, communication and distribution tasks, GPU calculations, interfaces and web-interfaces, machine learning and quantum computing tasks.

References

[1] What is dotnet? Available at: https://www.microsoft.com/net/learn/dotnet/what-is-dotnet. (accessed 31.10.2018)

[2] Web Framework Benchmarks. Round 16. Available at: https://www.techempower.com/benchmarks/#section=data-r16&hw=ph&test=plaintext. (accessed 31.10.2018)

[3] Announcing .NET 2015 Preview: A New Era for .NET Available at: https://blogs.msdn.microsoft.com/dotnet/2014/11/12/announcing-net-2015-preview-a-new-era-for-net/#_.NET_Core_5. (accessed 31.10.2018)