Why Use Akka.NET?

Table of Contents

Why Akka.NET?

Why should you invest in Akka.NET?

Solves Hard Infrastructure Problems, Simply

In modern software development, developers are expected to take advantage of multi-core systems by designing programs that are capable of partitioning their work to be executed concurrently across each of these individual cores. This is key to achieving desirable end-user goals such as responsiveness, reliability, scalability, and high throughput.

Multi-core programming has proven to be a difficult hurtle for many software development organizations due to its innate complexity. Synchronization mechanisms, sharing state across multiple threads of execution, and correctly exploiting opportunities for parallelism are done incorrectly more often than not.

In the age of cloud development, developers are also expected to be able to leverage *multiple computers* in the course of processing end-user requests, which takes the complexities of multi-core computing and multiplies them by introducing the network into the mix.

Enter the actor model, a historically proven approach to building highly concurrent, distributed, and scalable applications. The actor model provides a conceptually easy-to-follow solution to these problems by changing the way we fundamentally organize and construct applications.

Akka.NET is an implementation of the actor model for .NET and .NET Core.

Makes Concurrency Easy

Akka.NET encapsulates all state and work into individual components called actors, which execute as independent self-contained "microprocesses" - thus providing an expressive, natural way for our applications to execute concurrently. With actors, we can avoid the messy "shared state concurrency" approach that plagues modern software development today and rest easy.

Makes Network Communication Transparent

One of the most powerful traits of actors is that they communicate over the network transparently via message-passing. You no longer have to explicitly serialize messages, address them to specific senders, and route them over the network yourself. This is all handled via Akka.NET's built-in infrastructure and greatly simplifies the experience of building networking applications.

Resilient to Failure

Akka.NET exposes lots of different components that can be used to tackle various infrastructure and computing needs, but the core actor definitions themselves all share one extremely valuable trait inside concurrent systems: a high degree of fault isolation and explicit error handling.

Fault isolation simply means that an exception that is thrown inside one actor can't propagate to other actors or parts of your .NET application *unless you explicitly allow it to* via Akka.NET's supervision strategies. By default Akka.NET handles unhandled exceptions thrown by actors by *restarting the effected actor*. This is a profound concept, coming from a traditional OOP background: that you can have a piece of code that reboots when it enters a bad state and return to a well-known, safe state.

All of this can be done transparently, without impacting other parts of your application.

Runs Everywhere

Akka.NET supports both the .NET Framework and .NET Core. It is flexible enough to build large-scale distributed, real-time applications that scale to support hundreds and thousands of individual servers or it can even be used inside mobile applications that run on iPhone and Android devices using application frameworks like Unity3D and Xamarin Forms.

Unlike other distributed computing tools, Akka.NET is nimble enough to scale down to support problems as small as propagating concurrent UI events in a Windows Presentation Foundation application or even a mobile application. It's a versatile tool that can be used to solve concurrency problems wherever they might occur, use cases big and small.

High Performance

Akka.NET actors, remoting, persistence, and streams are all designed to perform exceptionally well even under heavy loads. A `ReceiveActor` in Akka.NET can process roughly 4.5 million messages per second on average (on an older generation Windows Azure 2-core virtual machine, no less.) The entire framework is designed to help build applications that are capable of processing and reacting to data in real-time.

High Adoption and Social Proof

By the numbers, Akka.NET is the most popular and widely used implementation of the actor model in .NET. Over 20,000 developers have gone through Akka.NET Bootcamp and Akka.NET NuGet packages are installed hundreds of times per day on average.

Akka.NET is used by businesses like Bank of America, CitiGroup, Boeing, Becton Dickinson, Activision Blizzard, S&P Global, and thousands of others to solve real-world, mission-critical problems.

Beyond that though, Akka.NET is originally ported from the Akka project, a widely-adopted and heavily utilized Scala and Java implementation of the actor model developed by Lightbend, Inc.

Thus Akka.NET has the benefit of learning from the mistakes and use cases of the original project, despite different platforms.

Free and Open Source

Akka.NET is open sourced under the commercially-friendly Apache 2.0 open source license, which makes it great for adoption by businesses of any size. Akka.NET is also part of the .NET Foundation, which means the project has a strong governance model, follows open source intellectual property best practices, and is run with professional production and quality standards.

Strong Ecosystem and Commercial Support

One of the keys to any successful open source project is building a robust ecosystem around it, and Akka.NET certainly enjoys one. There are numerous third party resources, tutorials, PluralSight courses, and more built around Akka.NET and more and more are produced every year. Akka.NET is also a popular topic at .NET Conferences, meetups, and podcasts.

In addition to numerous third-party resources, there is also Petabridge - a company that provides commercial support, consulting, training, and DevOps tooling for Akka.NET. Petabridge was founded by Aaron Stannard, one of the original Akka.NET co-founders and the maintainer of the Akka.NET project.

In addition to contributing to Akka.NET itself, Petabridge develops tools such as Phobos, a turnkey DevOps suite for Akka.NET and Petabridge.Cmd, a command-line tool for querying, managing, and monitoring live Akka.NET applications.

How to Get Started with Akka.NET

Once you've decided that your organization should explore Akka.NET to solve some of your infrastructure challenges, here's how you and your team can get started quickly and easily.

Attend Akka.NET Bootcamp

Akka.NET Bootcamp is a free, 17-lesson self-paced Akka.NET course developed by Petabridge aimed at teaching developers the very basics of the actor model and Akka.NET syntax. Over 20,000 developers have attended Akka.NET Bootcamp since 2015 and it's maintained and developed still to this day.

Start with bootcamp and get a feel for the syntax and style of Akka.NET.

Read the Petabridge Blog and Examples

Petabridge produces lots of regular Akka.NET tutorials, conceptual guides, and how-to videos. After you've finished Akka.NET Bootcamp, start here with the Petabridge blog.

In addition to the blog, Petabridge also maintains a collection of Akka.NET sample projects that you and your team can tinker with here.

Take an Akka.NET Live Webinar or Onsite Training

If you're ready to take a serious look at using Akka.NET in production at your place of business, consider attending one of Petabridge's live Akka.NET webinar trainings or arrange an onsite Akka.NET training for your entire team. Both of these trainings are lead by members of the Akka.NET team.

Contact Petabridge

If you have any other questions about Akka.NET, Petabridge, or the actor model please feel free to contact Petabridge here:

1. Website: https://petabridge.com/contact/
2. Email: hi@petabridge.com
3. Twitter: https://twitter.com/petabridge
4. LinkedIn: https://www.linkedin.com/company/petabridge/