The RyuJIT transition is complete!

Bruce Forstall

June 19th, 2018

RyuJIT is the code name for the .NET just-in-time compiler, one of the foundational components of the .NET runtime. In contrast, the Roslyn C# compiler compiles C# code to IL byte code. The RyuJIT compiler compiles IL byte code to machine code for multiple processors.

With the recent merge of the dotnet/coreclr #18064 pull request, the remaining legacy code generation components (from older JIT implementations) have been removed from the RyuJIT source code. This PR removed almost 50k lines of code! All four of the processor architectures targeted by RyuJIT (x86, x64, ARM32, ARM64) are now built purely on the RyuJIT architecture.

It is interesting to consider a short history of RyuJIT, and how we've gotten here.

The RyuJIT architecture has designs originating almost nine years ago, with implementation work starting over seven years ago. RyuJIT was implemented as an evolution of the existing JIT32 compiler (which supported x86 and ARM32), and gradually replaced most of the "back end" of that compiler with a new register allocator and code generation code, as well as introducing many new and improved "front end" optimization components. Throughout this transition to a new code generation architecture, we maintained the old code side-by-side with RyuJIT. Doing so provided benefits, but was also very costly, both in terms of testing and maintenance cost, as well as developer confusion needing to deal with so much legacy code scattered around. Now, with RyuJIT functionally superior to the legacy code, it was finally time to remove the legacy code, simplify the JIT code overall, and focus on the future.

RyuJIT for x64 was introduced to the world as a Community Technology Preview for .NET Framework almost 5 years ago. This was unusual for us at the time. We provided pre-release updates of RyuJIT for multiple years for people to try, and to give us feedback before we released it as a product. RyuJIT for x64 was released with .NET Framework 4.6, replacing JIT64, a now-legacy code base, almost 3 years ago.

For x86, RyuJIT replaced JIT32 in .NET Core 2.0 last year. We announced the JIT32 replacement as part of the public development process and then shared the completion of the work in the .NET Core 2.0 release announcement.

.NET Core 2.0 included support for the ARM32 architecture as a preview, using the legacy backend code path. As of last December, with dotnet/coreclr #15134, RyuJIT became the ARM32 JIT for .NET Core 2.1. This change included many contributions from Samsung.

ARM64 architecture support in RyuJIT is close to preview quality in .NET Core 2.1, and was built from the start of its implementation on the RyuJIT architecture. We've actually been working on ARM64 support in RyuJIT for over four years, and this work was pushed forward more recently with significant work by Qualcomm contributors.

Overall, our RyuJIT investments have been focused on evolving the code base towards enabling better support for:

- Multiple code generation targets (instruction sets and operating systems),
- Improved optimizations,
- Better and more flexible code generation, and
- Open, flexible, and robust design and implementation.

We believe that the new RyuJIT compiler architecture is a major improvement over the (now-removed) legacy code generator for achieving these goals.

We have recently invested in new code generation technology solely in the RyuJIT code generator, for example, SIMD support, architecture-specific hardware intrinisics, and support for the Linux software conventions.

It’s very satisfying to get to this point, and we can already see how removing all this old code will free us to be more agile going forward.

Thanks to everyone who contributed to this long effort!

Bruce Forstall Senior Software Engineer, .NET JIT Compiler Team

Follow