numba MCJIT to ORCv2 Migration

Context PowerPC / Z

Required Actions for PowerPC
V0.4

Feb. 2021 - abebeos@lazaridis.com
(parts are based on emails with lhames@gmail.com)
(context: https://github.com/abebeos/PowerPC/issues/4, Stage 2)

Summary

The MCJIT-to-OrcV2 migration should be (and will be, most possibly) executed by the numba core-developers themselves, potentially after llvm13 is supported by llvmlite (usually many months after llvm release).

There is no need for action now. The worst-case scenario is that one or two IBM PowerPC LLVM developers would need to invest one or two days to respond to requests from the numba-developers, to fix PowerPC related migration barriers.

Architectural Overview

[LLVM Compiler]

[RuntimeDyld]	1st generation JIT Linker, limited functionality/maintenance
[JITLink]	next generation JIT Linker, under development
[MCJIT]	first generation JIT API (very stable)
[ORCv1]	next gen JIT API, v1 (stable)
[ORCv2]	next gen JIT API, v2 (quite new, changes a bit)

OrcV2 can work with both, RuntimeDyld and the (under development) JITLink.

=> OrcV2.RuntimeDyld - has PerfJITEventListener
=> OrcV2.JITLink => needs port of RuntimeDyld.PerfJITEventListener

PowerPC

Should work out-of-the-box

[OrcV2.LLJIT] ---uses---> [RTDyldObjecLinkingLayer] ---wraps---> [RuntimeDyld]

Who should do the Migration?

The numba core-developers, which are familiar with the llvmlite bindings and JitEngine classes.

A capable external developer would potentially have the usual problems to do the implementation to the liking of the core-devs.

Estimated Effort

Numba MCJIT->OrcV2 migration General

One core-dev-month, assuming availability of all involved parties for upcoming problem-analysis / bug-fixing

PowerPC Related

One dev week (if done in parallel with the numba-migration executed by the numba-devs)

Suggested Immediate Tasks

- None directly related
- Indirect: provide CI for PowerPC/Z to ensure public visible test-results

Some Resources

- https://github.com/numba/numba/issues/2290 - older issue
- https://github.com/numba/llvmlite/pull/245 - older migration PR
- https://github.com/dotnet/llilc/pull/614 - migration sample, other project

Details

=> OrcV2.RuntimeDyld - has PerfJITEventListener
=> OrcV2.JITLink => needs port of RuntimeDyld.PerfJITEventListener

LH: OrcV2 is a set of components for building JITs. It does *not* hide the underlying JIT linker: You build your JIT class on top of RTDyldObjectLinkingLayer to use RuntimeDyld, or on top of ObjectLinkingLayer to use JITLink. This all contrasts with MCJIT which was a black box and always used RuntimeDyld.

LH: LLJIT is an off-the-shelf OrcV2 based JIT class. The LLJITBuilder class (used to create instances of LLJIT) will choose an underlying JIT linking layer (RTDyldObjectLinkingLayer or ObjectLinkingLayer) for you by default based on your platform, but you can call LLJITBuilder::setObjectLinkingLayerCreator to override this and control the choice of linking layer yourself.

Numba

LH: For numba the correct long-term solution is almost certainly to accept the default.

=> Numba MCJIT->OrcV2 migration: one dev-month, assuming availability of any involved parties for upcoming problem-analysis / bug-fixing

PowerPC

[OrcV2.LLJIT] ---uses---> [RTDyldObjectLinkingLayer] ---wraps---> [RuntimeDyld]

=> Today I would expect MCJIT -> OrcV2 to more-or-less "just work". It will use RuntimeDyld automatically since that still the default on Linux/PowerPC.

=> Maybe tweaking code / relocation model default settings. Hopefully not more than a day or two.