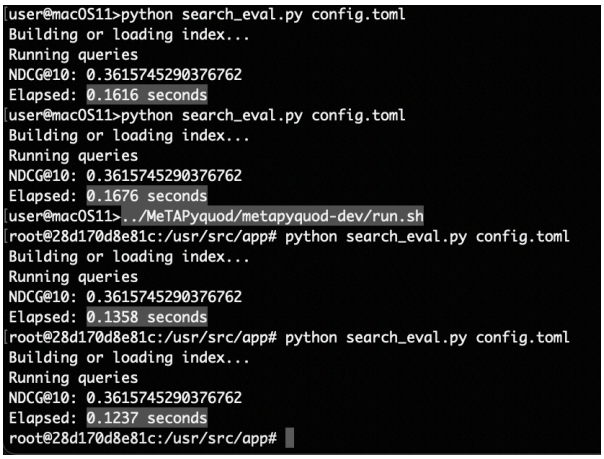
Matt Williamson — mdw8 (lead/captain)
CS410
November 27, 2020

MeTAPyquod
Progress Report

As of the end of November, the metapyquod-dev container image is essentially complete and working—at least for x86_64 platforms. Work on metapyquod-server is in its early stages.

`metapyquod-dev`

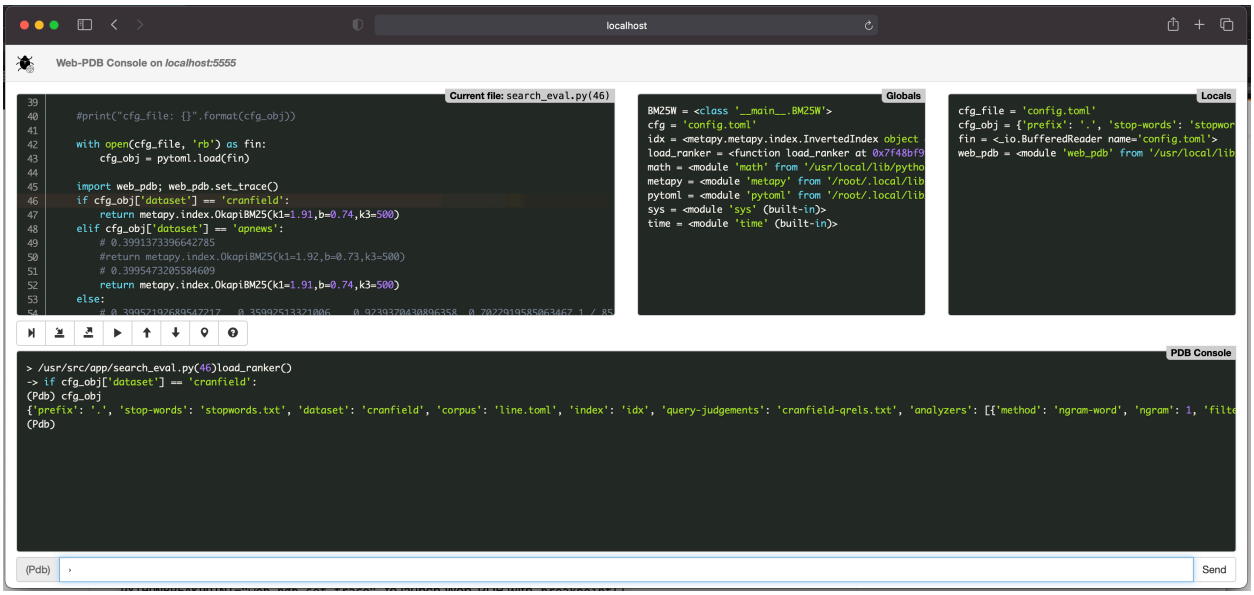
Initial creation of the container definition went smoothly and operates as expected on macOS—in fact, a surprising result was observed that running a working implementation of MP2.4 natively on a macOS 11 host and then in the metapyquod-dev container resulted in a consistently *faster* execution time in the container than natively:



Two adjustments from the initial proposal are likely necessary for the metapyquod-dev portion of the project:

1. A desired goal was to include a GUI IDE for Python development in the container—however, this was seen as high-complexity and added risk, especially on non-Linux platforms. However, in the process of researching options the web-pdb Python module was discovered,

which provides a full debugging environment within a web browser. The need for a robust debugger was the primary motivator for providing a GUI IDE, and this module provides the visual debugging capability with far lower complexity. Therefore, the final solution will map the host working directory into the container to allow any editor (PyCharm, VSCode, Spyder, etc.) to work directly with the files on the host, but provide debugging via web-pdb.



2. Another desired goal was to support armv7l/armhf and armv8/aarch64 systems, mainly because of the popularity of Raspberry Pi and similar SBC hardware. However, attempts to do so have been beset with build problems, despite the fact that the base official Python 3.6 image being used supports those architectures. First, scipy/numpy and related dependencies are not well-supported on these systems, though this may be surmountable—and if so, the benefit may be significant because the time required to build these dependencies on these platforms is a large investment, and a prebuilt container image would eliminate this cost. Second, and more difficult, the MeTA build system itself fails on aarch64 (at least) when trying to download and build icu4c—the same problem encountered by many users on many different platforms—and the solution to this problem (or even it's exact root cause) remains elusive. This may be resolvable but would be likely to consume more time than is available within the project scope.

It may be noted that—in this student's opinion—the present state of support of MeTA/metapy jeopardizes the viability of continuing to base the course content on these tools. It appears that no updates have been made to the tool in two years, and as Python and other dependencies continue to evolve, the surface area of supported configurations will continue to shrink. Additionally, it may be particularly noteworthy that the build problems described above with 64-bit ARM systems are likely to also occur with new Apple ARM processor (M1) systems, which will eventually include all new Apple systems going forward.

`metapyquod-server`

Work on metapyquod-server is just beginning, but will benefit both from work already done on metapyquod-dev and the tech review completed on AWS Lambda—which shares some implementation details including the OpenAPI specification. Given difficulties with metapyquod-dev, it is likely that some descoping may occur here as well, for instance the feedback or metrics endpoints. Remaining technical unknowns have to do with the behavior of MeTA inverted disk indexes being *updated*—e.g. it is unclear whether a total re-index will be necessary when starting the container or whether the existing index can be loaded and new documents added. Based on experience with metapyquod-dev, the metapyquod-server container will only target x86_64 platforms, though this is less of an issue for its use case.