Conversational Function-Calling with OpenAI in Parsl Workflows

Alejandro Duque
2023 IRIP fellow
Undergraduate Student
Computer Science, 4th year
Kastan Day, Matthew Berry, Daniel S. Katz, Volodymyr Kindratenko

Research Report

In scientific research, it is common to process large amounts of data. Currently, there are software solutions to create workflows that allow to define and concatenate data processing steps in code. However, these solutions are not always the friendliest for scientists, who are the end users. This research explores ways to facilitate the development as well as the use of workflows. We focused specifically on Phyloflow, which is a workflow tool for phylogenetic tree computations.

We researched Parsl's capabilities for workflow description. Parsl is a Python-based tool that allows us to simplify workflow steps and enable parallelization of phylogenetic workflows. The results obtained are that Parsl is more flexible for experienced developers, but this does not always translate well into simpler development for the general public. Our second objective was investigating the use of AI-driven tools to enhance the creation, translation, and execution of these workflows. OpenAI function calling has been successfully used to execute workflow tasks from natural language commands. It was found that integrating NLP for the use of workflows is possible; however it requires additional development considerations.

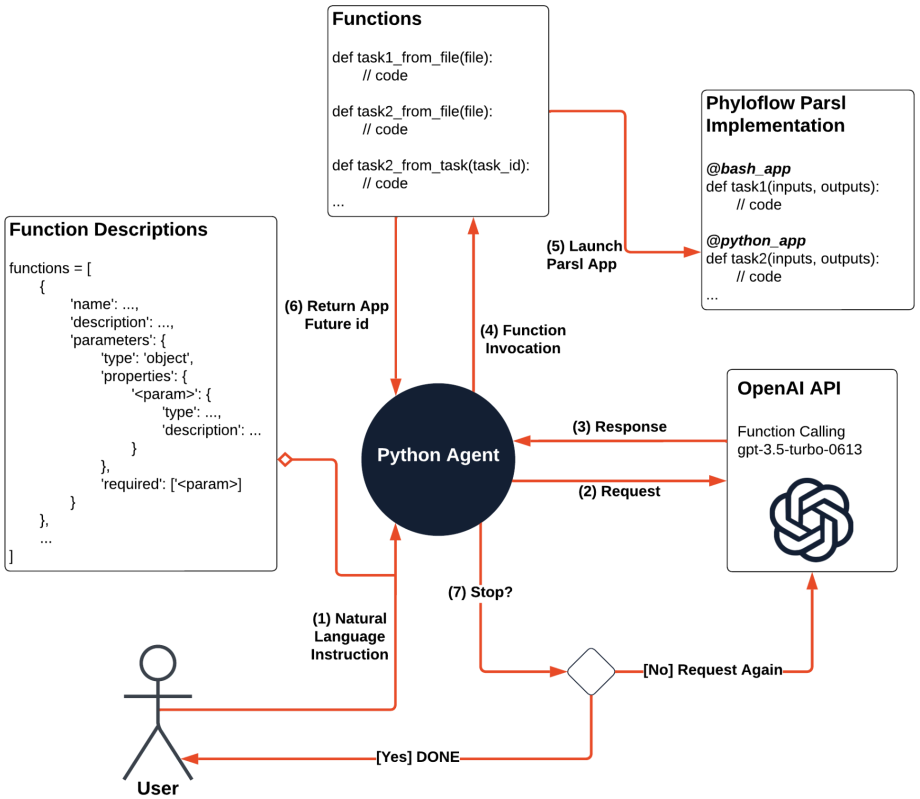
Workflow Translation

We translated the entire Phyloflow workflow that was originally written in WDL and bash to Parsl/Python. We implemented a Parsl App for each task in the workflow. All bash scripts previously used to run tasks have been replaced by Python functions. Test functions with test files were created for each task. The workflow has been extended to provide parallelization for processing multiple files. The runtime was changed from multiple Docker containers, each with their own Conda environment, to a single Docker container with multiple Conda environments. This was done because, unlike WDL in Parsl, there is no way to define a different execution environment (Docker image) for each task.

AI Integration

Initially, we attempted using Langchain to link OpenAI and Phyloflow to generate Parsl code that extends the workflow. The approach was to contextualize queries with relevant information from the files of the project. It did effectively parse through the files and could respond to user natural language queries. However, this approach had notable limitations, as it could only generate high-level workflow step descriptions and couldn't effectively enhance or execute any of the workflow steps.

The focus then shifted to workflow usage instead of code generation. We used the OpenAI Function Calling API to execute individual tasks in the workflow. To make this work, we created a new set of functions that work as an interface between Parsl applications and the OpenAI API. By indexing the Parsl App Futures with identifiers, we were able to chain function calls to execute multiple tasks in a single natural language instruction. Below is a diagram describing the AI integration.



Results

We created a working Parsl implementation of the Phyloflow tool. The project is packaged inside a Docker image with all its dependencies. The container was tested on Windows and Linux systems, as well as on the Delta supercomputer. Translating the workflow from WDL to Parsl is shown in https://github.com/grimloc-aduque/Phyloflow-Parsl-Implementation.

We implemented a proof of concept for AI integrations within workflows developed in Parsl using the OpenAI Function Call API. We tested it on two Phyloflow workflow tasks and on various user prompts. The agent was able to execute the appropriate tasks based on the user's instruction, chain tasks, and determine when to stop. The concept can be easily extended for all tasks in the workflow, as well as for other Parsl workflows.

Conclusions

In terms of developer experience we found that Parsl gives finer control over the file dependencies, it is easy to extend with native python functionality, and it offers straightforward parallelization. However, Parsl also requires more experience with the filesystem, it is harder to run tasks with conflicting environment dependencies, and it is not intuitive for retrieving inputs and outputs by indexing an array.

Regarding the AI integration, we found that it is possible to use NLP to execute multi-step workflows written in Parsl. However, it requires additional considerations during the workflow development stage, such as the ability to communicate between functions via string information (e.g. ids of scheduled tasks).

References

Ncsa. (2022, July 13). NCSA/phyloflow: Phylogenic tree computation and packaging with Docker, WDL. GitHub. https://github.com/ncsa/phyloflow

OpenAI platform. OpenAI Platform. (2023). https://platform.openai.com/docs/api-reference

Parallel scripting library. Parsl. (2020). https://parsl.readthedocs.io/en/stable/