Conversational Function-Calling with OpenAl in Parsl Workflows

Alejandro Duque* and Abdullah Syed**, Kastan Day, Matthew Berry, Daniel S. Katz, Volodymyr Kindratenko

National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign

INTRODUCTION

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 (parsl-project.org) is a Python-based tool that allowed 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.

CONTEXT

What is Phyloflow?

Cancer phylogenies are graphs that represent evolutionary relationships of tumors. Phyloflow is a pipeline used to build phylogenetic graphs by processing genomic and mutagenic data in a multistep process (github.com/ncsa/phyloflow). Phyloflow made heavy usage of WDL, a workflow description language heavily used by the bioinformatics community.

Workflow steps

- Load a VCF file generated by 'mutect' and its annotated version from VEP (Variant Effect Pipeline).
- Convert the mutations from the VCF file into the required input format for 'pyclone-vi'.
- Execute 'pyclone-vi' to cluster the mutations.
- Adapt the output of the pyclone clustering to be compatible with 'spruce' tree inference.
- Gather the output files and merge them into a JSON that works with the PhyloDiver visualization tool.

METHOD

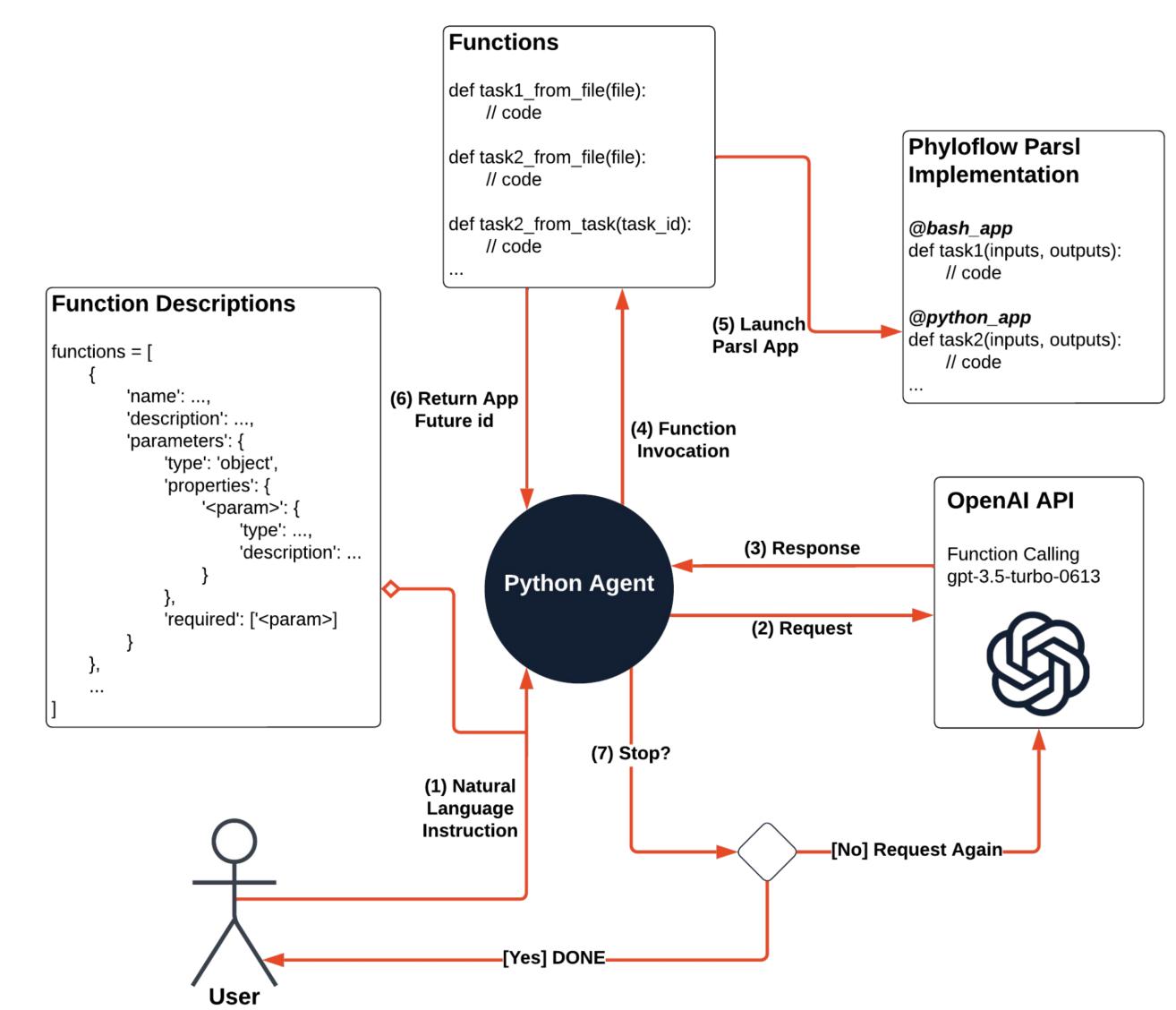
Workflow Translation

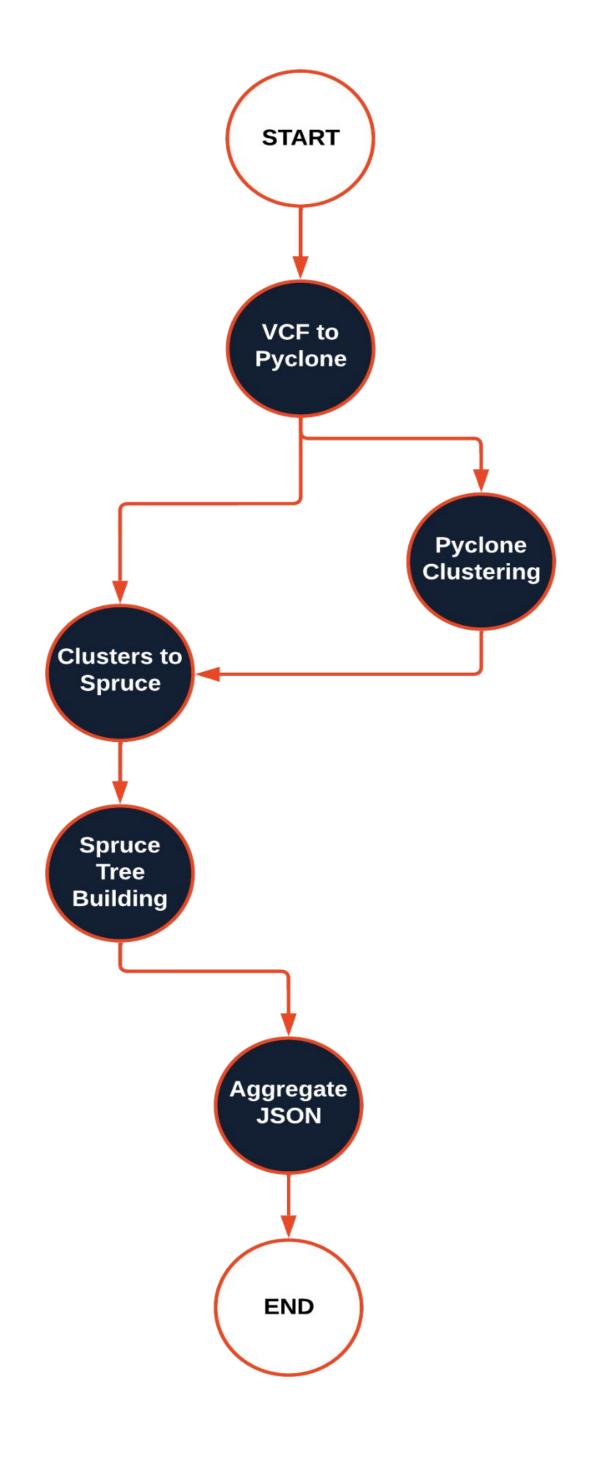
We translated the entire Phyloflow workflow that was originally written in WDL and bash to Parsl/Python. We implemented a Parsl App for each task in the workflow. All bash scripts previously used to run tasks have been replaced by Python functions. Test functions with test files were created for each 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.

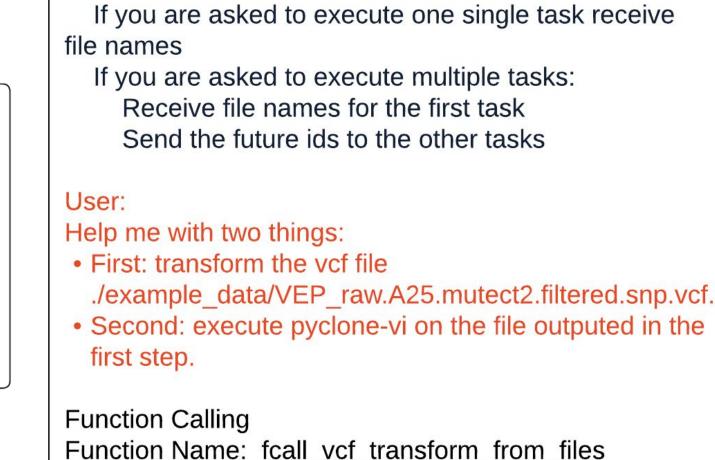
AI Integration

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.







Function Name: fcall_vcf_transform_from_files
Function Args: {'vep_vcf':
'./example_data/VEP_raw.A25.mutect2.filtered.snp.vcf'}
<AppFuture at 0x7f90af178b90 state=pending>

User: Task scheduled with AppFuture id: future_5_run_vcf_transform '
Now what?

Function Calling
Function Name: fcall_pyclone_vi_from_futures
Function Args: {'vcf_future_id':
'future_5_run_vcf_transform'}
<AppFuture at 0x7f9072014490 state=pending>

User: Task scheduled with AppFuture id: future_6_run_pyclone_vi '
Now what?

DONE

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

Developer Experience with Parsl vs. WDL

Parsl

- Gives finer control over data file dependencies.
- Is easy to extend with native Python functionality.
- Offers straightforward parallelization.

However, Parsl

- Requires more experience with the filesystem.
- Makes it harder to run tasks with conflicting environment dependencies.
- Is not intuitive for retrieving inputs and outputs by indexing an array.

AI Integration

It is possible to use NLP to execute multi-step workflows. 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).

ACKNOWLEDGEMENTS

This work was sponsored by the FoDOMMaT REU.







CONTACT INFORMATION

*aduquead13@gmail.com

**aasgdc@mail.missouri.edu

