Bringing Workflows to Life Using a Graph Database With BEE

Steven Anaya

New Mexico Tech

steven.anaya@student.nmt.edu

Mentors: Tim Randles & Patricia Grubel

Abstract

Scientific workflows can be complex and require much planning, setup, and maintenance to execute on HPC machines. BEE seeks to simplify this process by modeling workflows using a workflow language specification, storing and visualizing the workflows using a graph database, and executing them using the BEE workflow engine. To easily map workflows into the graph database, I implemented a simple, high-level programming interface for use with BEE development. In addition to this interface, I have also worked to set up our development environment—including code auditing, dependency management, unit testing, and package deployment tools—to improve the efficiency of our development and the quality of our software.

BEE Overview

- BEE: Build and Execution Environment
- Workflow management/visualization/analysis
  - Uses a graph database (Neo4j)
- Executes Common Workflow Language workflows:
  - Locally
  - On a cluster (in-development)
  - In the cloud (in-development)
- Supports Charliecloud containers natively

Development Environment Setup

- Entirely Python 3 for portability
- **Pyenv** to manage Python versions
- **Poetry** to manage project dependencies
- **Pylama** to enforce consistent style, good practices
- **Unittest** framework to implement unit testing
- **To Done: GitHub + Travis CI** to implement CI/CD

Why a Graph Database?

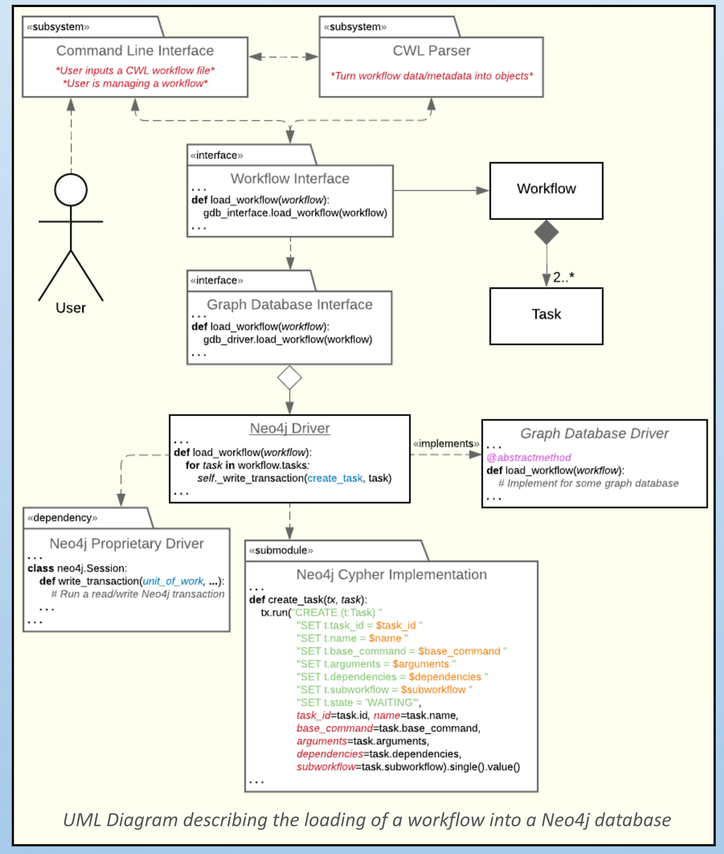
- Natural mapping from workflows to graphs
- Storage of workflow data/metadata in one database

About Neo4j

- Single database per instance
- Uses Cypher Query Language exclusively
- Visualize and manipulate database:
  - In a browser
  - In a Cypher shell
- Python 3 driver for running query transactions

Our Graph DB Abstraction Layer

- Abstracts away the graph database implementation
- Easily load workflows into the graph database
  - Store metadata as node/relationship properties
  - Visualize the workflow in a browser
- Design allows alternate Graph DB implementations



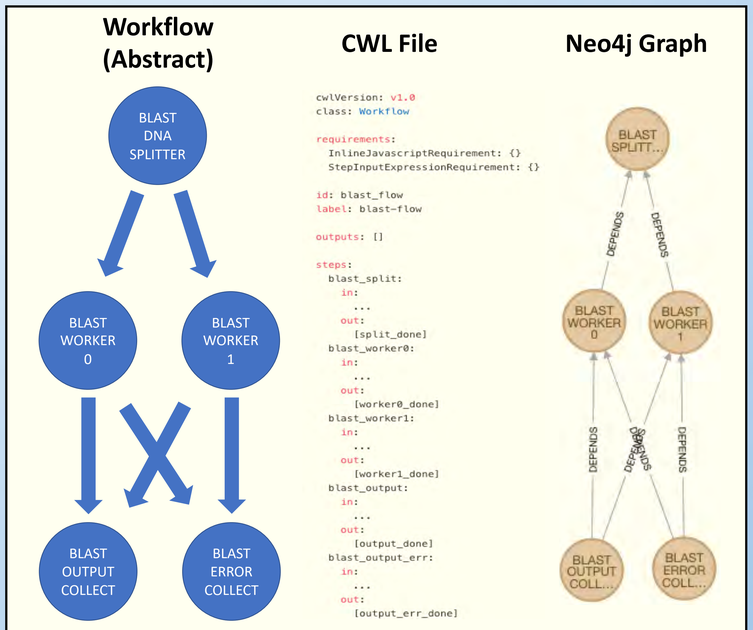
UML Diagram describing the loading of a workflow into a Neo4j database

Why CWL?

- Unified language for workflow modeling
- Models metadata (requirements, resources, etc.)
- Community-driven project, abundance of support
- Follows OpenStand principles of open development

Our CWL Parser

- Convert CWL into Python objects
- Load these into the graph database
- BEE extends CWL to add unsupported features
  - Use Python expressions instead of JavaScript
  - Use HPC containers instead of Docker containers



A simple BLAST DNA sequencing workflow is described in CWL and loaded into Neo4j

Future Work

- Export/import workflows to other databases
- Back-up/cache/save workflows as CWL files
- Modify workflow loaded into database
- Build workflows entirely within Neo4j
- Implement other graph databases

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH UNIVERSITY

Los Alamos NATIONAL LABORATORY EST.1943

LA-UR-19-27098