WRENCH Python and REST API Design and Development

Liliana Royer, Erick Orozco-Ciprian, Sukaryo Heilscher

Project Sponsor: Dr. Casanova, Sponsor Organization: UHM

ICS 496

Spring 2024

Introduction

Background

- WRENCH is a C++ API for easing the development of fast and accurate simulators of distributed computing systems
- To use WRENCH from any programming language and/or from the browser, an incomplete REST API was developed for managing simulations via HTTP requests
- A key component of this API is the WRENCH *daemon*, which runs on the local machine and includes both a Web server and a simulation thread
- An incomplete Python API had been developed on top of the REST API

Project Objectives

Main Objective: Make significant extensions to the WRENCH Python and REST APIs so that they serve a broader set of users and use cases

Specific Goals:

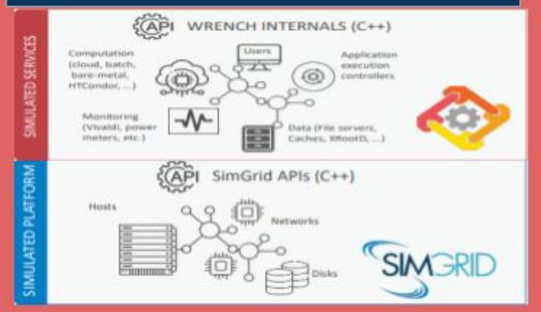
- Add minor functionality to the Python and REST as a learning experience
- Make the entire functionality of the powerful and generic WRENCH "Action" C++ API available via the REST and Python API
- Implement example simulators and testing

Methodology

- Design new classes and methods to be added to the Python API
- Define and implement relevant REST API endpoints in the WRENCH daemon
  - Write endpoint specifications to an OpenAPI JSON description file
  - Implement server-side C++ methods in the WRENCH daemon
- Implement and test Python methods and classes
- Project Management via Discord:
  - Weekly meetings with the sponsor to discuss progress & project direction
  - Direct communication throughout for Q&A and troubleshooting

Flow of Execution:

1. Simulation objects are created band used by the simulator using the Python API
2. Objects interact with the Simulation class which implements the REST calls
3. Post/Put/Get REST API calls placed based on the OpenAPI JSON specification
4. The Web server thread in the WRENCH daemon creates work items to perform to change simulation state and places them in a thread-safe work queue
5. The simulation thread in the WRENCH daemon retrieves work items from the work queue
6. The simulation thread modifies simulation state using the WRENCH C++ API
7. The simulation thread notidies the Web server thread of work unit completions
8. The Web server thread sends a REST reply to the client
9. The Simulation class updates the Simulation objects
10. The simulator can react to simulation object changes



WRENCH

Solution

Tasks Accomplished

- As a first learning experience, augment the REST and Python API with a working implementation of the "wrench::FileRegistryService" abstraction available in the C++ WRENCH API
- Port the entire WRENCH "Action" API to the REST and Python API
  - Implementation of classes and methods equivalent to those provided in the C++ wrench::Action and wrench::CompoundJob classes
- Created illustrative examples in Python to drive development and perform initial testing
- Made design choices for consistency and clarity in both Python and C++ implementations
- Put in place a testing framework

Challenges

- Difficult design choices and decisions had to be made:
  - What application states should be kept only server-side and which should be replicated client-side?
  - What amount of code duplication between the client and the server (e.g., specification of Enums) is tolerable? In some cases, code duplication, while inelegant, leads to increased readability
  - What C++ classes should be conserved as classes for the Python API, and which should instead be replaced by native Python data structures?
- Steep learning curve:
  - A pre-existing code base meant a steep learning curve
  - Spanned two languages: C++ and Python
  - Unfamiliar and somewhat cumbersome OpenAPI JSON standard to specify REST endpoints
  - It took some time to get to the point where development progressed smoothly and rapidly

Takeaways

- Expanded proficiency in C++, Python, and RESTful API development, including JSON
- Gained hands-on experience in integrating and extending existing APIs
- Enhanced understanding of software design principles and of the difficulties involved for making sound design decisions
- Gained experience is software development when different programming languages need to interoperate