Julia Interface Integration for PODIO: Enhancing Functionality and Performance

Ananya Gupta
Indira Gandhi Delhi Technical University for Women

June 2023

Mentors: Benedikt Hegner (CERN), Graeme A Stewart (CERN)

PODIO:

PODIO [1], also known as plain-old-data I/O, is a C++ library developed specifically for facilitating the creation and management of data models within the field of particle physics. Its core principle revolves around the utilization of plain-old-data (POD) structures, prioritizing them over deep-object hierarchies and virtual inheritance. This approach not only enhances the runtime performance but also simplifies the implementation of persistency services.
PODIO goes beyond performance optimizations by providing essential high-level functionality tailored for physicists. It includes features for supporting inter-object relations and automating memory management, ensuring convenient data handling. Additionally, it offers a Python interface supported by the widely-used ROOT framework, enabling seamless integration into existing workflows.
To streamline the process of creating efficient data models, PODIO employs code generation techniques utilizing a user-friendly yaml-based markup syntax. This approach simplifies the definition and construction of data models, promoting efficiency and maintainability.
In order to align with modern software technologies, PODIO has been purposefully designed with concurrency in mind. It provides foundational support for concurrency and also offers basic capabilities for vectorization technologies, enabling optimal performance in multi-threaded and vectorized computing environments.

Project Motivation and Description:

Julia [2], a dynamically typed programming language, combines the user-friendly syntax of Python with the computational speed typically associated with languages like C/C++. It has gained significant traction in the field of data science due to its unique combination of user-friendliness and high-performance capabilities.
The motivation behind this project is to develop a Julia backend interface for the PODIO library. This interface aims to provide the same functionality as the existing implementation, while preserving the performance optimizations that are fundamental to PODIO.
Building upon the early prototype developed during a previous Google Summer of Code project [3], which demonstrated its feasibility, this project seeks to leverage the advanced capabilities of Julia to enhance the PODIO library and facilitate its integration within the High-Energy Physics (HEP) ecosystem. The envisioned outcome is a seamless incorporation that empowers researchers and practitioners within the HEP community to harness the combined power of PODIO's comprehensive data modeling capabilities and Julia's high-performance computing environment [4]. By achieving this objective, the project endeavors to advance efficient data processing in the HEP domain and contribute to the ongoing evolution of scientific analysis techniques.

Project Goals:

The main objective of this project is to add a comprehensive Julia backend to the PODIO library. The initial prototype developed during a previous Google Summer of Code project [3] has already established the feasibility of this integration. The primary goal now is to complete the feature set required for the Julia backend. Thorough testing, with a specific focus on performance, will follow the implementation phase. By accomplishing these goals, the project aims to deliver a robust and efficient Julia backend for PODIO. This will

enable seamless integration into the High-Energy Physics (HEP) ecosystem, allowing researchers and data scientists to leverage the combined benefits of PODIO's data modeling capabilities and Julia's high-performance computing environment [4].

Proposed Timeline:

Week 1-2:

- Gain a solid understanding of the Julia programming language, focusing on its key concepts such as Just-in-Time (JIT) compilation and multiple dispatch paradigm.
- Thoroughly test the existing PODIO library and evaluate the functionality of the previous Google Summer of Code project.
- Identify any potential issues or areas for improvement in the current implementation.
- Develop a comprehensive testing plan for future iterations.

Week 3-4:

- Dive deeper into the design considerations of PODIO, emphasizing plain-old-data (POD) structures and avoiding inheritance.
- Explore methodologies to ensure that the Julia implementation exhibits a natural and idiomatic coding style, distinct from both C++ and Python interfaces, while still preserving the desired functionality and compatibility with the PODIO library.
- Explore memory management in Julia and ensure automatic memory management for users.
- Investigate code generation techniques in Julia to create classes based on higher-level abstractions.

Week 5-6:

- Begin implementing the necessary data types, custom components and member functions in Julia to replicate the functionality of PODIO.
- Ensure proper composition and struct generators in the design of data models.
- Conduct thorough testing after each feature implementation, addressing any issues or bugs.
- Document progress and challenges encountered during the implementation phase.

Week 7-8:

- Continue implementing additional features and functionalities in the Julia backend for PODIO.
- Perform regular testing after each feature implementation, ensuring correctness and evaluating performance.
- Seek feedback from mentors to validate the compatibility and effectiveness of the Julia backend.

Week 9-10:

- Finalize the feature set of the Julia backend for PODIO based on project goals and requirements.
- Conduct comprehensive performance testing to evaluate efficiency and effectiveness.
- Refine and optimize the Julia backend code as necessary to enhance performance.
- Prepare final documentation summarizing the project, including design decisions, implementation details, and testing results.

Week 11-12:

- Conduct a thorough review of the entire project, addressing any remaining issues or bugs.
- Refactor code, optimize performance, and improve overall code quality.
- Perform final testing to ensure the stability and reliability of the Julia backend.
- Prepare the project for submission, including the finalized documentation and presentation.

References:

1. https://github.com/AIDASoft/podio
2. https://julialang.org/
3. https://hepsoftwarefoundation.org/gsoc/blogs/2022/blog_PODIO_SoumilBaldota.html
4. Eschle, J., Gal, T., Giordano, M., Gras, P., Hegner, B., Heinrich, L., ... & Vassilev, V. (2023). Potential of the Julia programming language for high energy physics computing. *arXiv preprint arXiv:2306.03675*.