STL/Eigen - Automatic conversion and plugins for Python based ML-backends

Khushiyant
About Me

Name - Khushiyan
Student - B.Tech in Computer Science and Engineering
Professionally - MLE @ Martian
Interest in Tech - Container Orchestration, ML Infra
cppyy is a powerful, automatic Python-C++ bindings generator that lets you seamlessly call C++ from Python and vice versa. It offers high performance, lazy loading for large projects, cross-inheritance, and interactive exploration of C++ libraries—all without the need for intermediate languages or tedious boilerplate code. Imagine dynamically mixing Python and C++ features with ease, thanks to cppyy’s use of Cling, the C++ interpreter, which matches Python’s dynamism and interactivity. cppyy is future-proof, supporting advanced C++ features and modern compilers, effortlessly handling complex tasks like working with Boost’s boost::any. cppyy can achieve near C++ performance, making it ideal for large-scale, distributed development environments.
Example

```python
>>> cppyy.cppdef(r"""
    ... void hello() {
    ...     std::cout << "Hello, World!" << std::endl;
    ... }
""")
True

>>> cppyy.gbl.hello()
Hello, World!
```
Problem Statement

Current support follows container types in STL like `std::vector`, `std::map`, and `std::tuple` and the Matrix-based classes in Eigen/Dense. Presently, Cppy’s `std::vector` is accessed by `cpppy.gbl.std.vector` doesn’t support arbitrary dimensions, and there is no support for conversion mechanisms between Python built-in types, numpy.ndarray, and STL/Eigen data structures.
What are we trying to achieve hopefully

Cppyy for CPU operations in ML backends
Arbitrary Dimension Support for STL Vector

Our major goal is to facilitate CPU operations for machine learning backends (such as JAX), however the lack of support for arbitrary dimensions is the first impediment.

Also, Provide support for basic rudimentary types that could be essential in further conversion utilities later.

For example, arbitrary dimension support will allow processing types like `<class cppyy.gbl.vector<cppyy.gbl.std.vector<double>> at 0x121053940>`
Conversion Mechanism

Since, our goal is to provide support for JAX CPU operations, it becomes kind of mandatory due to current overhead (2.2x) such as in this case of eigen comma insertions.

We are planning to implement better initialisation as well as conversion utilities to facilitate the implementation for JAX plugins.
JAX Plugins and benchmarking

After completion of above milestones, we will implement the JAX plugins where we try to replace JAX implementation with cppyy for CPU computes such as matrices, sparse based calculations. Later, it is planned to undergo the benchmarking to facilitate and quantify the comparison between JAX and our’s implementation.
Q&A