Fully **Pythonic** RDataFrame Applications

Pawan Johnson, V. E. Padulano, E. Tejedor

ROOT

Data Analysis Framework

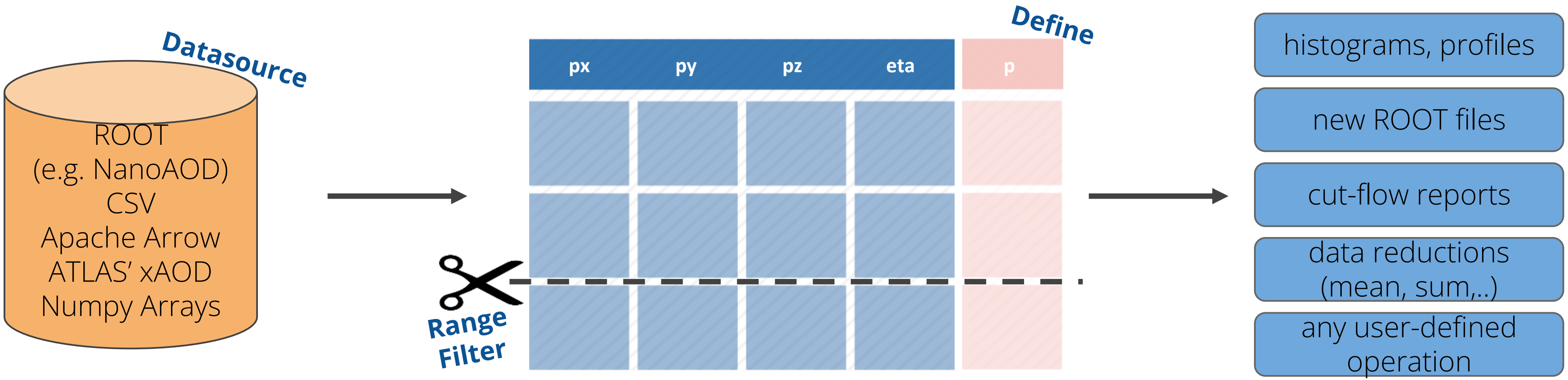
https://root.cern

Pythonized ROOT RDF

- RDataFrame in a Nutshell
- Current Interface v/s Fully Pythonic Interface
- Implementation
- Performance Comparison
- Future Direction

RDataFrame in a Nutshell

- High Level Interface for Data Analysis.
- Data from TTrees, TChains, CSV… all accessible from one place.
- A modern interface for operations like Filter and Defines.



Old Interface

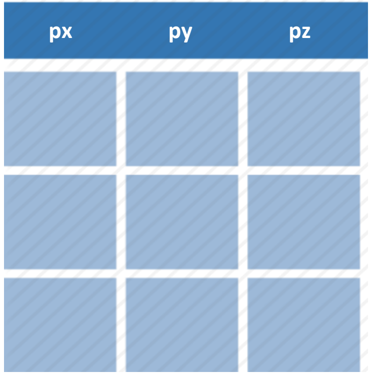
```
ROOT.gInterpreter.Declare("""
std::array<double, 3> p(double px, double py, double pz){
   std::array<double, 3> p{px, py, pz};
   return p;
}
bool momentum_cut(std::array<double, 3> p){
   double p2 = 0.0;
   for (auto&& x : p)
       p2 += x*x;
   return sqrt(p2) < 10.0;
""")
rdf.Define("p", "p(px,py,pz)").Filter("momentum_cut(p)")
```

C++ code inside python string

New Interface

```python
rdf.Define("p", lambda px, py, pz: np.array([px, py, pz]))\
    .Filter(lambda p: np.linalg.norm(p)<10)
```

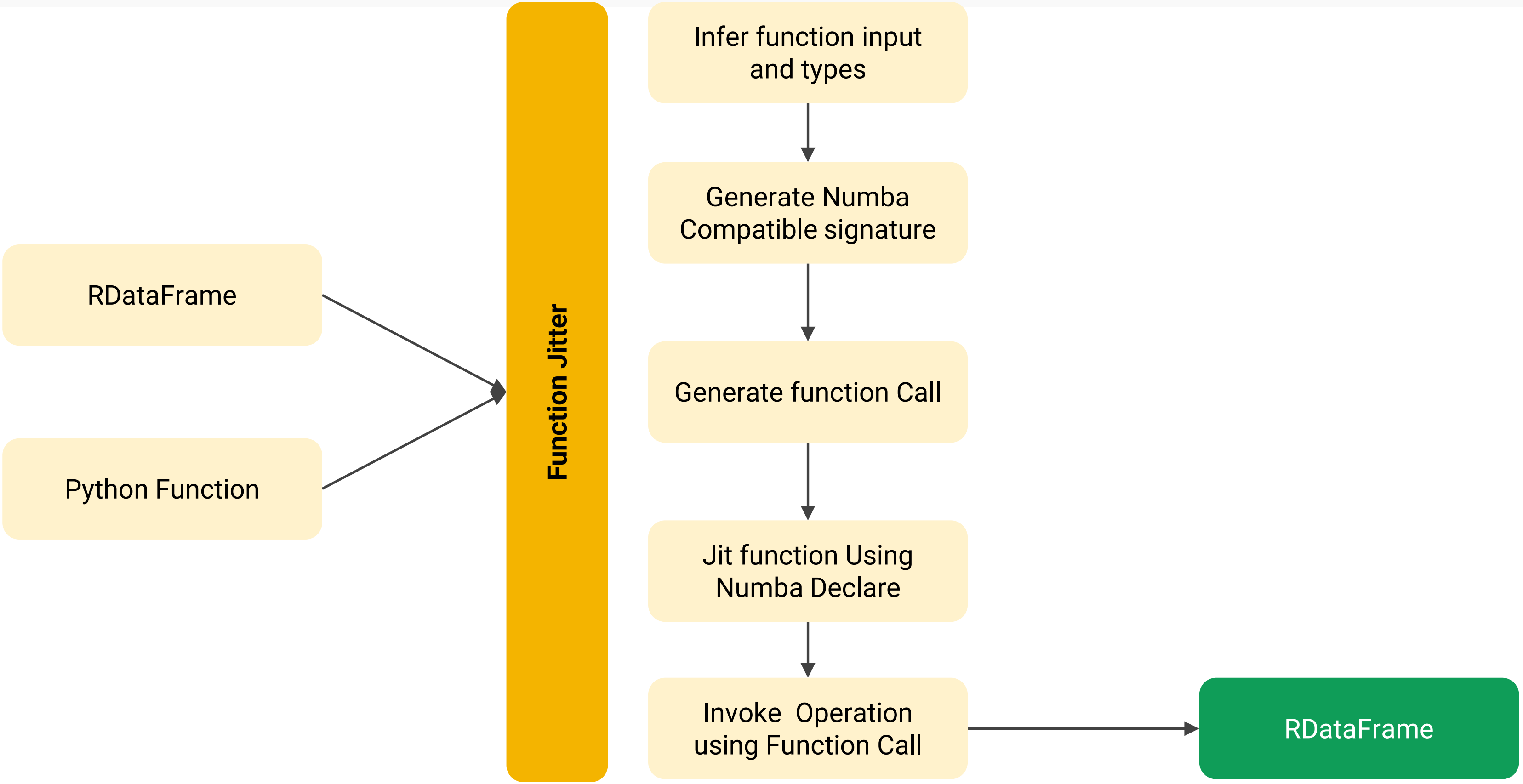
```python
lambda px, py, pz: np.array([px, py, pz])
```



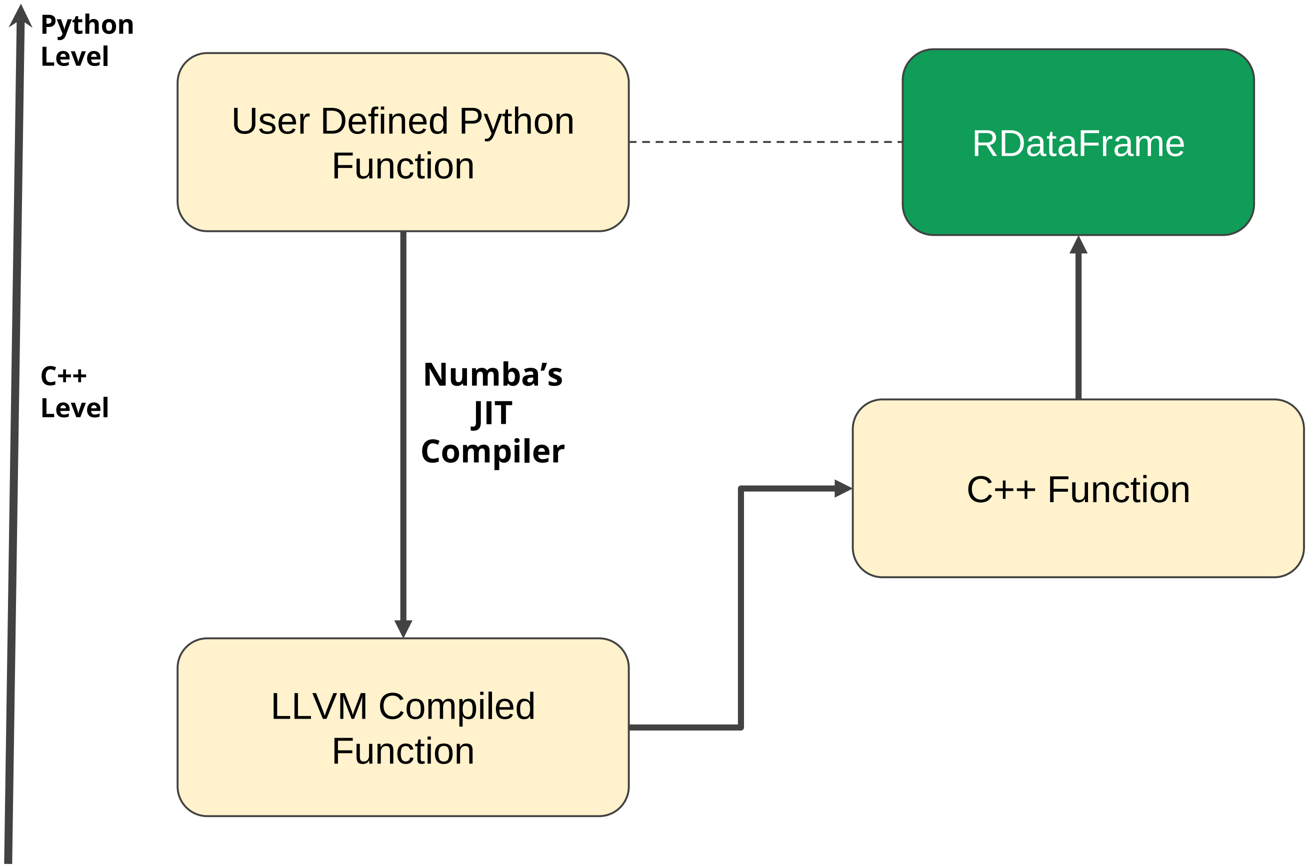
New Pythonization

1. Infers the data type from the user input or makes use RDataFrame's GetColumnType method
2. The function is now JIT-ed using Numba's compiler
3. The JIT-ed function's memory address is extracted and recast into a C++ function
4. The C++ Function is declared
5. After the function is declared, we now invoke the operations using the new "C++" Function

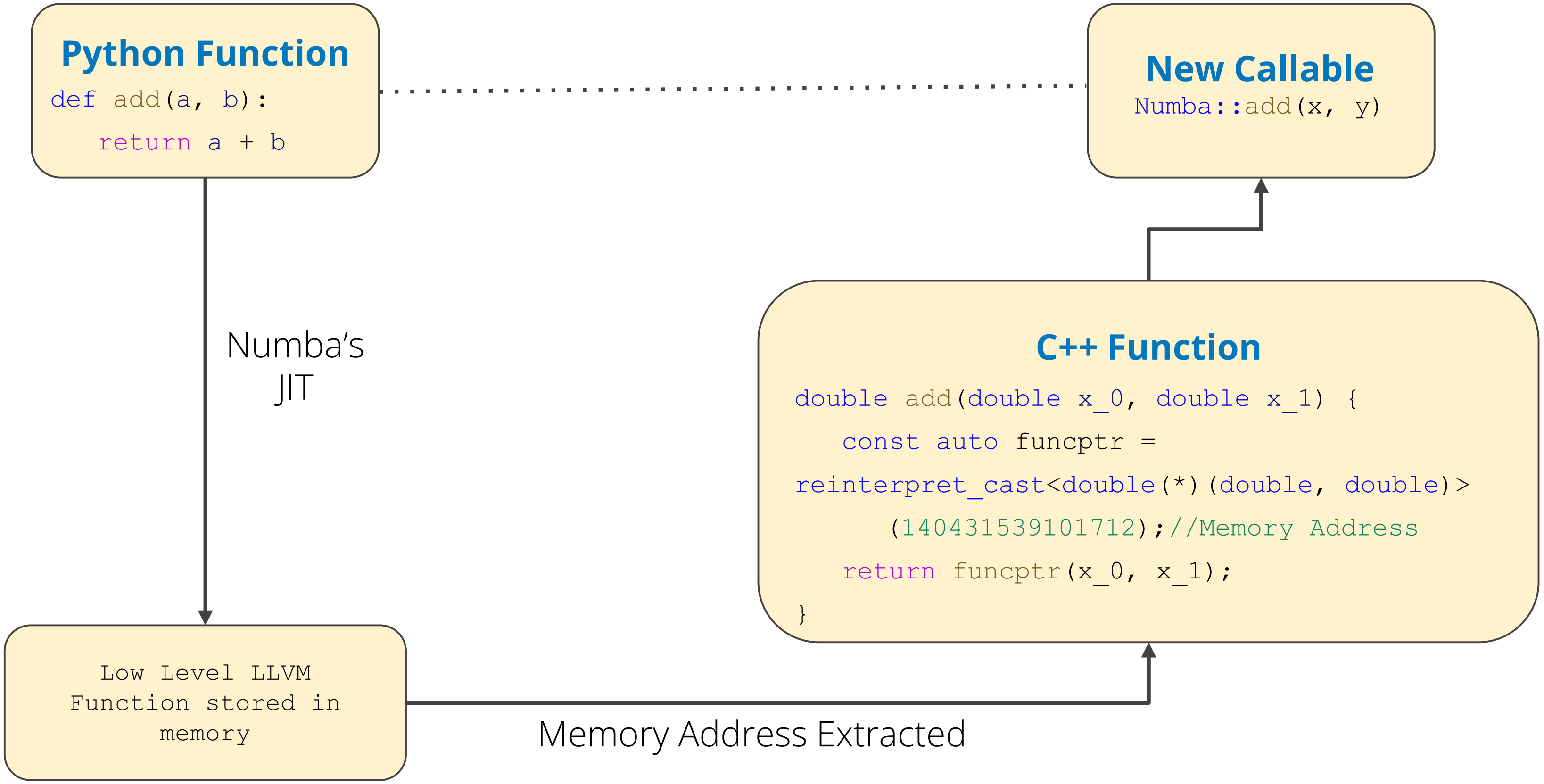
Function Jitter WorkFlow



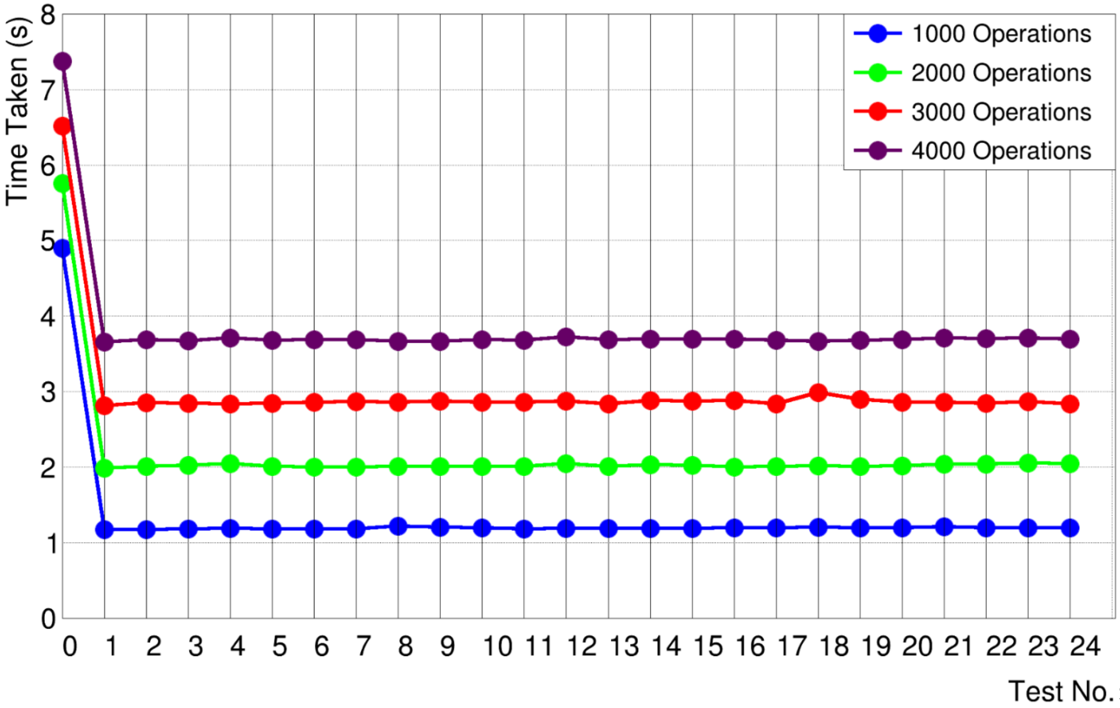
Implementation



What it looks like in code



How it Performs



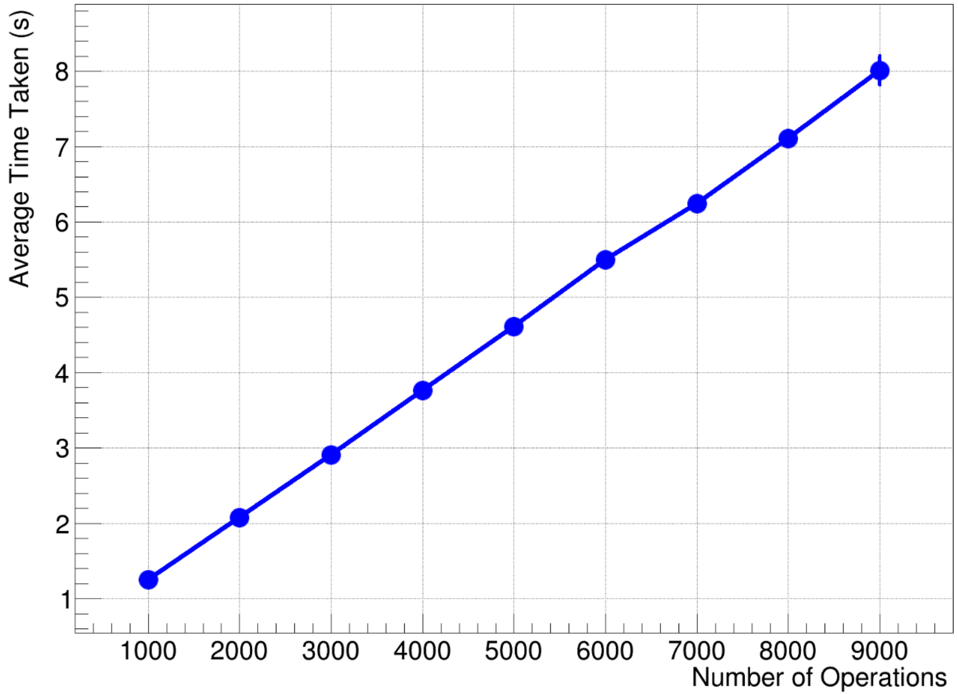
Size of RDF: 2^{20}

Key Takeaways

1. JIT-ing occurs only once.
   The function is now in cache.

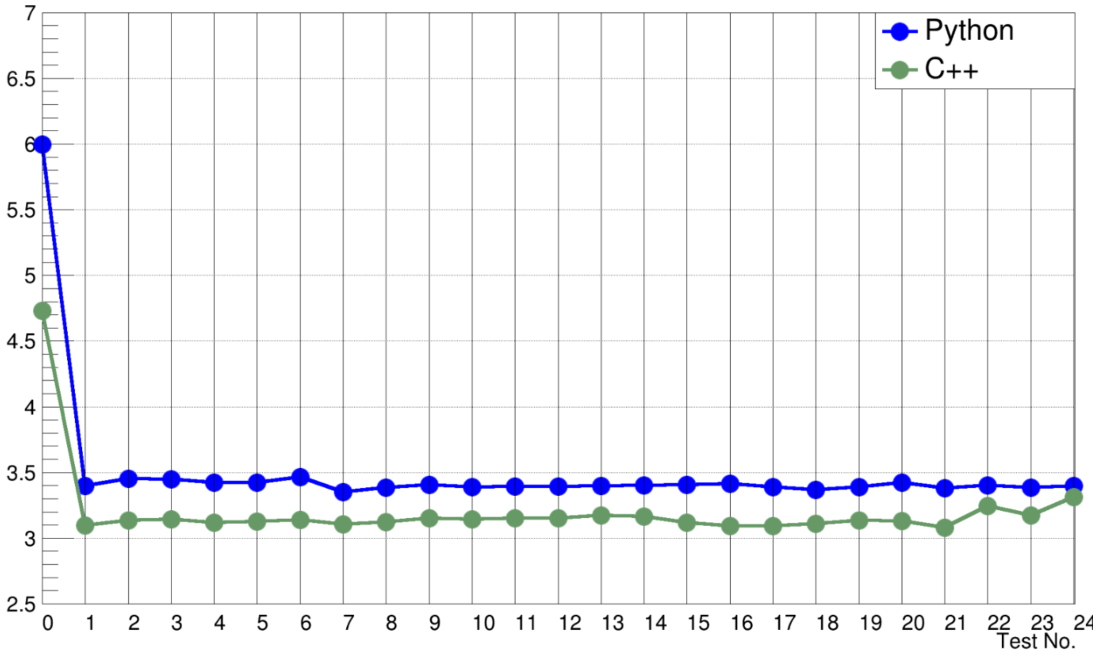
2. Time taken increases linearly with number of operations.
   (0.84s for 1000 Operations)

Variation of Time with No. of Operations



Size of RDF: 2^{20}

How it Compares



Key Takeaways

1. The Python version is slower than the C++ version.
2. The slowness can be attributed to the Function signature being checked against the cache and recreating the function call on each iteration.

Code for Benchmarks: https://github.com/Pawan-Johnson/RDF_Pythonization_Defines_and_Filters_Benchmarks/

Future Directions

- Test performance in Distributed RDataFrames
- Bring up the speed closer to C++ Levels
- Improve error handling for nopython mode failures
- Support for multidimensional RVec
- Bring down to single JIT by interfacing with NumPy directly

Thank you for your attention