Codon – Python Compiler

ABSTRACT

Codon is a high-performance Python compiler that converts Python code to native ma chine code with no runtime overhead. It provides huge speedups over Python, generally in the range of 10-100x, and enables native multithreading. While Codon supports nearly all of Pyth on's syntax, it is not a drop-in replacement and big codebases may require adjustments. Codon differs from CPython in terms of performance, and it frequently creates the same code as an e quivalent C or C++ program. It provides seamless Python interoperability and supports per-fu nction compilation via the @codon.jit decorator. Codon is free for non-commercial use and m akes use of the Boehm trash collector. It is released under the Business Source License, and th e source code is open to the public. The performance of a codon varies according on the appli cation, and missing Python functions can still be accessed via Python via from python import.

AIM AND OBJECTIVES

Codon's aim is to create a high-performance Python compiler that can compile Python code to native machine code without any runtime penalty, resulting in 10-100x or greater spee dups. It seeks to eliminate dynamic runtime and virtual machines to improve efficiency and e nsure smooth Python compatibility, while also embracing Python's syntax and semantics to m ake it easier to use for Python developers.

The main objectives of the Codon compiler are:

- Codon is a Python compiler that produces native machine code without any runtime o verhead.
- It offers significant speedups over traditional Python, typically on the order of 10-100 x or more.
- Codon supports multithreading, which can lead to even higher performance gains.
- It is extensible via a plugin infrastructure, allowing users to add new libraries and opti mizations.
- To provide a user-friendly interface for researchers and students to explore genetic cod e and generate custom DNA sequences with desired properties.

INTRODUCTION

Codon is a Python-to-C++ compiler designed to provide high-performance code for P ython programs. Marek yliski produced it, and it is licenced under the Apache License 2.0. Th e compiler analyses Python code and converts it to optimised C++ code, which is subsequentl y compiled into machine code. This enables Codon to produce code that is significantly faster than Python's default interpreter. Codon also supports a number of optimisation techniques, in cluding as loop unrolling, function inlining, and vectorization, which increase the efficiency o f the resulting code even further. Furthermore, Codon has a number of user-friendly features, such as support for Python's standard library and the ability to communicate with C++ code. O verall, Codon is a strong tool for developers who want to construct high-performance Python code while retaining Python's simplicity of use and flexibility.

BACKGROUND & PREVIOUS WORK

Python is an interpreted language, which implies that rather than a compiler, it is exec uted by an interpreter. However, there are a number of Python compilers and tools available f or optimising Python code or converting it to other languages.

PyPy, which was launched in 2007, was one of the first Python compilers. PyPy is an alternative Python implementation that uses a Just-In-Time (JIT) compiler to accelerate Pytho n code execution. PyPy is built in Python and can be used as a stand-in for the regular Python interpreter.

Another popular Python compiler is Cython, which is a superset of Python that enable s the easy construction of C extensions for Python. Cython code is compiled to C, which is su bsequently compiled to machine code with the help of a C compiler. Cython is a popular choi ce for high-performance scientific computing applications because of this. Nuitka, Shedskin, a nd RPython are some other Python compilers and tools.

Nuitka is a Python compiler capable of producing independent executables from Pytho n code. Shedskin is a Python-to-C++ compiler for generating C++ code from Python code.

ANAYSIS

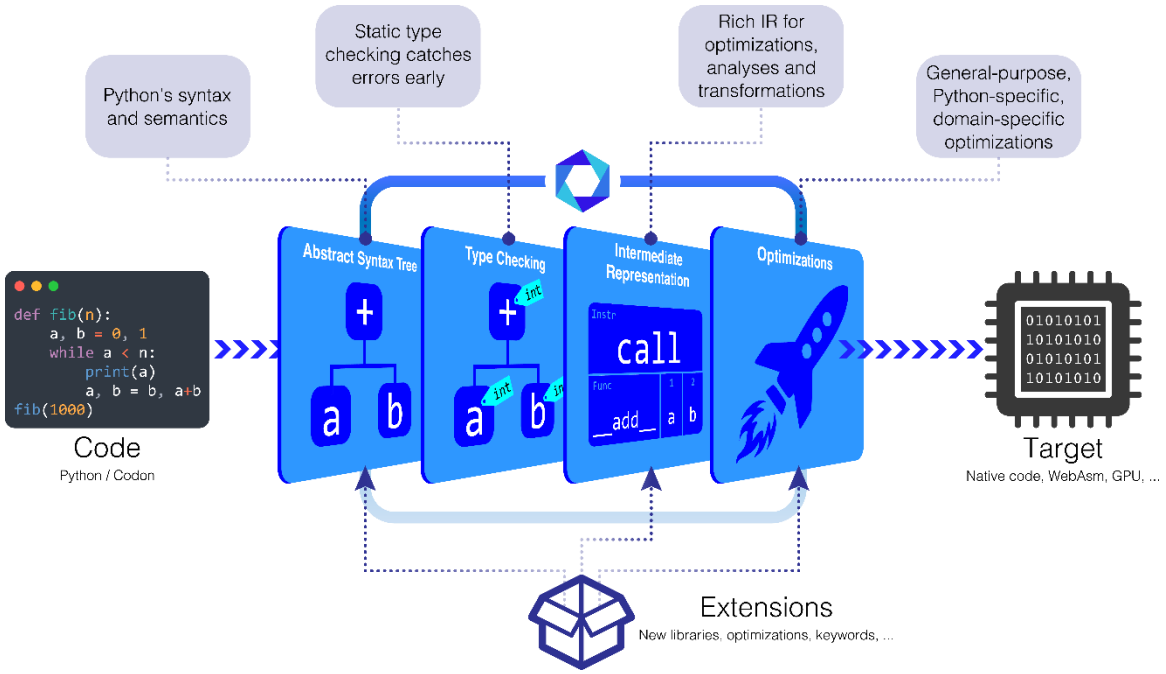

Figure 5.1: Working of Codon Compiler

Codon is a Python compiler that uses LLVM to generate optimized machine code from Pytho n code. Here's a high-level overview of how Codon works:

Parsing: The Codon compiler parses the input Python code to create an abstract syntax tree (A ST) that represents the structure of the code. Type Inference: Codon uses type inference to det ermine the type of each variable in the code. This information is used later in the optimization process to generate more efficient code.

Optimization: Codon applies a series of optimizations to the AST, such as loop unrolling, con stant propagation, and dead code elimination. These optimizations improve the performance o f the generated code.

LLVM IR Generation: Codon generates LLVM IR code from the optimized AST. LLVM IR i s a low-level representation of the code that can be easily optimized and translated into machi ne code for a wide range of architectures.

LLVM Optimization: Codon uses LLVM's optimization passes to further optimize the LLVM

IR code. This includes instruction combining, constant folding, and loop optimizations.

Machine Code Generation: Finally, Codon generates machine code from the optimized LLVM IR code using LLVM's code generator. This machine code can be executed directly on the tar get architecture, providing efficient and optimized performance.

CONCLUSION

Codon is a Python compiler that uses LLVM to generate efficient machine code from Python code. It applies optimizations to the code, generates LLVM IR code, applies further o ptimizations, and generates machine code. This provides high-performance execution of Pyth on applications.

REFERENCE

1. Python-based compiler achieves orders-of-magnitude speedups: **https://news.mit.edu/2023/codon-python-based-compiler-achieve-orders-magnitude-speedups-0314**
2. MIT's Codon compiler allows Python to 'speak' natively with computers **https://interestingengineering.com/innovation/mits-codon-compiles-python**
3. Codon project official website: **https://exaloop.io/**
4. Codon GitHub repository: **https://github.com/exaloop/codon**
5. Codon project documentation: **https://docs.exaloop.io/codon**