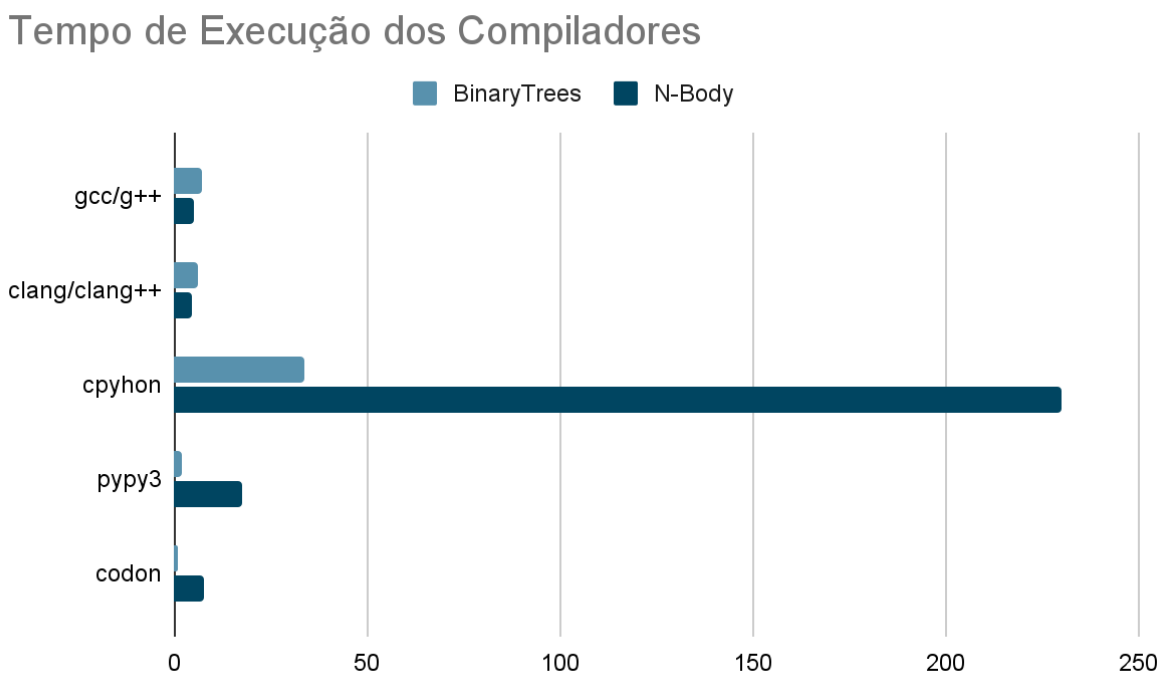
# Comparação entre compiladores/interpretadores Python & C/C++

Aluno: Felipe Archanjo da Cunha Mendes

**Tabela com os valores de tempo de execução para cada compilador:**

| | gcc/g++ | clang/clang++ | cpython | pypy3 | codon |
|---|---|---|---|---|---|
| **Binary Trees** | 0m7.082s | 0m6.423s | 0m33.971s | 0m1.977s | 0m0.860s |
| **N-Body** | 0m5.289s | 0m4.645s | 3m49.961s | 0m17.461s | 0m7.868s |

**Gráfico de comparação de tempo de execução:**
Para visualizar melhor as diferenças entre os compiladores em relação ao tempo de execução, podemos criar um gráfico de barras que compara os tempos de execução de cada compilador para as duas tarefas propostas, binary_trees (com entrada 16) e nbody (com entrada 10000000). O gráfico a seguir mostra esses resultados:




Eixo X: Tempo de execução (em segundos)
Eixo Y: Compiladores utilizados

Como podemos observar, para ambas as tarefas, o compilador Codon se mostrou significativamente mais rápido que os outros compiladores. O PyPy3 também obteve tempos de execução bastante inferiores aos dos outros compiladores, enquanto os

compiladores C/C++ da GNU e do LLVM tiveram desempenho semelhante entre si. O CPython, por outro lado, foi o mais lento em ambas as tarefas.

**Motivos que influenciam a diferença de tempo de execução entre os compiladores:**

Existem diversos fatores que podem influenciar o desempenho dos compiladores em relação ao tempo de execução. Alguns desses fatores podem ser encontrados nas páginas oficiais dos compiladores e outros podem ser identificados com base na comparação dos resultados obtidos para cada um deles. A seguir, apresentamos alguns dos principais motivos que podem influenciar a diferença de tempo de execução entre os compiladores utilizados:

Presença de JIT (Just-in-time compilation): O JIT é uma técnica de compilação que permite que o código seja compilado sob demanda, ou seja, durante a execução do programa. Isso pode levar a um desempenho mais rápido em alguns casos, uma vez que o compilador pode ajustar a otimização de acordo com a execução real do programa. No entanto, essa técnica também pode adicionar overhead na execução do programa, já que a compilação JIT pode levar algum tempo. O PyPy3, por exemplo, utiliza JIT para otimizar o código Python, o que pode explicar seu desempenho superior em relação ao CPython.

LLVM: O LLVM é um conjunto de bibliotecas e ferramentas que permitem a criação de compiladores otimizados para várias linguagens de programação. O clang é um compilador que utiliza o LLVM como backend e pode oferecer um desempenho superior em relação ao gcc, por exemplo, devido a suas técnicas avançadas de otimização de código.

Compilação direta para linguagem de máquina: Os compiladores que geram código nativo podem ter desempenho superior em relação a compiladores que geram bytecode ou que dependem de uma máquina virtual para executar o código. O Codon, por exemplo, é um compilador que gera código nativo, o que pode explicar seu desempenho superior em relação aos compiladores Python padrão.

Grau de otimização: O grau de otimização aplicado pelo compilador pode influenciar significativamente o tempo de execução do programa. Compiladores que aplicam técnicas avançadas de otimização, como análise estática de código, inlining de funções e vetorização, podem gerar código mais eficiente e, portanto, mais rápido em relação a compiladores que aplicam menos otimizações. O Codon, por exemplo, utiliza várias técnicas avançadas de otimização para gerar código Python nativo altamente eficiente, resultando em um tempo de execução mais rápido em comparação com os compiladores de Python padrão.

**Resultados:**

Os resultados obtidos nos testes mostram claramente que os compiladores C/C++ são os mais eficientes na execução dos códigos de Binary Trees e N-Body, com tempos muito inferiores aos obtidos pelos interpretadores Python. Em relação aos compiladores C/C++, o Clang foi ligeiramente mais rápido que o GCC em ambos os casos.

Já em relação aos interpretadores Python, o PyPy3 se mostrou bastante mais rápido que o CPython, com tempos de execução cerca de 10 vezes menores em ambos os casos. Porém, o Codon foi ainda mais rápido que o PyPy3, chegando a ser mais de 40 vezes mais rápido que o CPython em Binary Trees e quase 30 vezes mais rápido em N-Body.

Essas diferenças nos tempos de execução podem ser explicadas pelas características de cada compilador/interpretador. Os compiladores C/C++ geralmente são mais eficientes em termos de tempo de execução porque geram código de máquina diretamente, sem a necessidade de interpretar o código fonte. Além disso, esses compiladores geralmente possuem alto grau de otimizações, como a remoção de código redundante e a reorganização do código para minimizar o número de acessos à memória.

Já os interpretadores Python, por outro lado, precisam interpretar o código fonte a cada execução, o que adiciona uma camada extra de sobrecarga de tempo de execução. Além disso, os interpretadores Python geralmente possuem menos otimizações que os compiladores C/C++, pois precisam lidar com a complexidade da linguagem dinâmica.

O PyPy3, por sua vez, é um interpretador alternativo para Python que utiliza uma técnica chamada Just-In-Time (JIT) Compilation, que compila o código Python para código de máquina em tempo de execução. Isso pode levar a um grande aumento de desempenho em relação ao CPython, que não utiliza essa técnica. Já o Codon, como mencionado anteriormente, é um compilador de Python que gera código de máquina diretamente, sem a necessidade de interpretar o código fonte, o que o torna ainda mais rápido que o PyPy3 e outros interpretadores Python.