Apresentação final
Go Gopher

Caroline Evangelista
Andrei Majada

Universidade Federal de Pelotas - Maio de 2023

Kernels

Das cinco versões do programa, foram desenvolvidas os kernel's EP e IS:

- **IS - Integer Sort**, random memory access
- **EP - Embarrassingly Parallel**

Avaliação de desempenho: IS

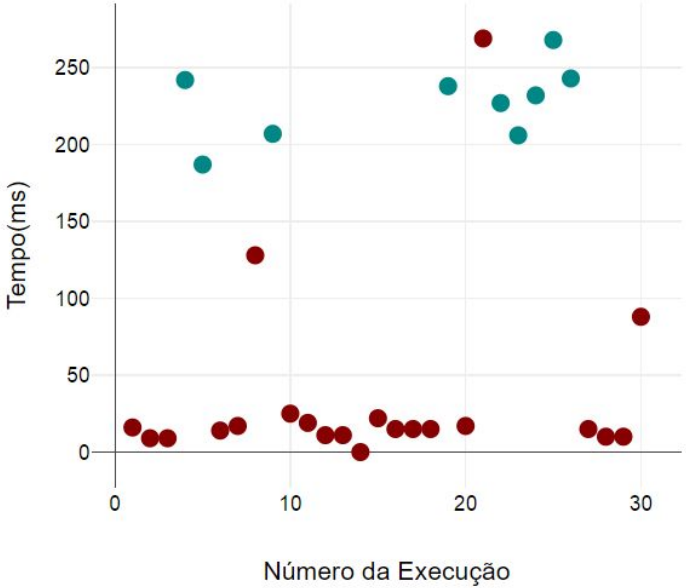
- **O kernel IS consiste na execução de uma ordenação de um conjunto de números dispersos.** Esse kernel é utilizado para simular e medir a capacidade de computação de inteiros e a comunicação de dados em sistemas de computação. O algoritmo utiliza por padrão o algoritmo Bucket-Sorting.

Configurações da máquina de execução

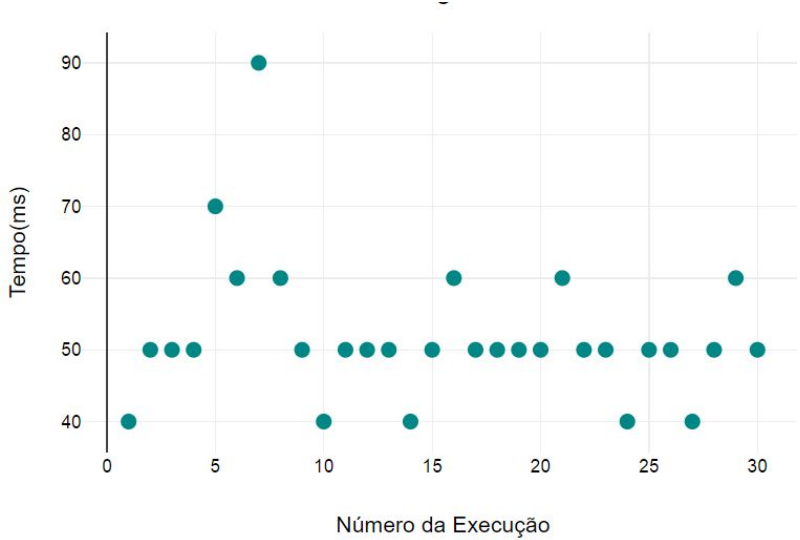
Processador: intel i7-8565U CPU / 1.80GHz / 4 cores / 8 threads

RAM: 16 GB DDR4 3200mhz

Avaliação de desempenho comparando a nossa implementação em GO com a implementação em C++ do Dalvan



Go



C++

Recursos para paralelismo

Goroutines - Utilizamos as goroutines como threads para executar o código simultaneamente.

Channels - Foi utilizado para gerenciar a sincronização das goroutines.

WaitGroup - Aguarde a conclusão de vários goroutines. Possui três funções: Add(), Done() e Wait().

Posicionamento final

Implementação

EP - A aplicação está totalmente funcional mas precisa de melhorias na utilização das go routines, utilizar uma go routine dentro de um for mostrou ser muito pesado e lento.

IS - A aplicação executa até o final mas a verificação e ordenação tem resultados voláteis. Em algumas execuções consegue ordenar todo vetor, em algumas não.

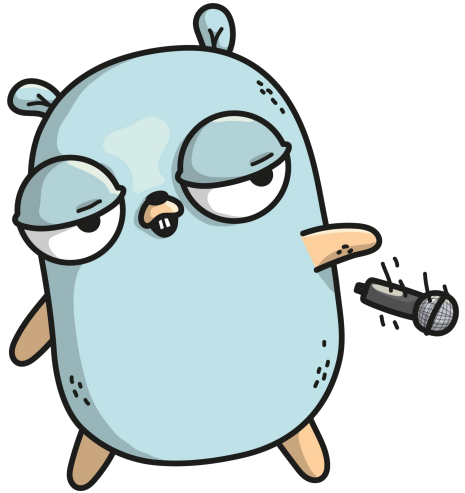
LIKE

DISLIKE

Linguagem

- **Sintaxe**: A sintaxe do go é menos verbosa e mais intuitiva se comparada a C++, por isso a curva de aprendizado pode ser considerada bem rápida.

- **Desempenho**: Quando avaliando Go com C++, em ambos as versões desenvolvidas, podemos notar que o Go tem um desempenho inferior, sendo o tempo de execução muito maior.



Obrigada!

Caroline Evangelista
Andrei Majada

https://github.com/Andrei-Majada/Go_NPB