XXXI Seminário de Iniciação Científica da UFERSA

Núcleo de Avaliação: Núcleo II
Área temática: Ciências Exatas e da Terra
Área do Conhecimento: Linguagens de Programação

Implementação e Avaliação de Programas Paralelos usando Python e Kotlin

Breno Klywer Olegario de Moura, Paulo Henrique Lopes Silva

A computação paralela é essencial para maximizar o desempenho de processadores multicore modernos, tornando a escolha da linguagem de programação crítica. Python é limitado pelo Global Interpreter Lock (GIL), exigindo *multiprocessing* para paralelismo, enquanto Kotlin usa corrotinas na JVM e C++ emprega threads nativas. Este trabalho avaliou o desempenho dessas linguagens implementando algoritmos clássicos: Multiplicação de Matrizes (paralelismo "embaraçosamente paralelo") e Quicksort (dividir para conquistar recursivo). Os testes ocorreram em um AMD Ryzen 5 5600G (6 núcleos/12 threads), utilizando C++17, Kotlin 2.0.0 e Python 3.13.3. O benchmark incluiu aquecimento e 7 medições independentes, monitorando tempo, uso de memória e CPU para calcular o *Speedup* (tempo sequencial/paralelo).

No Quicksort com entrada de 5.000.000 elementos, C++ foi a única linguagem com ganho real (Speedup 3.21x), mantendo consumo de memória eficiente (4.73 MB). Kotlin (Speedup 0.92x) e Python (Speedup 1.21x) sofreram com *overhead* de gerenciamento, onde o custo de criação de tarefas superou o ganho computacional. Notavelmente, o Python apresentou alto consumo de tempo absoluto (20.73s vs 0.11s do C++), exacerbado pela gestão do GIL e trocas de contexto, apesar do baixo uso de CPU.

Na Multiplicação de Matrizes (N=1473), Kotlin com *speedup* de 5.93x, quase atingindo o limite teórico dos 6 núcleos, demonstrando a eficiência da JVM para cargas de trabalho intensivas e granulares. O C++ não foi avaliado para este algoritmo. Já o Python apresentou um *slowdown* severo (sequencial 0.03s vs paralelo 0.62s), indicando que suas bibliotecas nativas sequenciais (otimizadas em C) superam implementações manuais paralelas, onde o custo de serialização e comunicação entre processos (IPC) para contornar o GIL anula qualquer benefício.

Conclui-se que C++ oferece o melhor desempenho bruto com gestão manual de memória eficiente. Kotlin é uma alternativa robusta e escalável para tarefas intensivas em CPU. As ferramentas padrão de paralelismo do Python mostraram-se inadequadas para algoritmos de alta performance pura, limitadas pelos gargalos do GIL e custos de IPC.

Palavras-chave: Computação Paralela, Avaliação de Desempenho, C++, Kotlin, Python.

Agência financiadora: PIVIC.

Campus: Mossoró.