¡¡¡¡¡¡¡ HEAD ======= ¿¿¿¿¿¿¿ refs/remotes/origin/main
¡¡¡¡¡¡¡ HEAD

Pedro Francisco Staino Santayana pedrosantayana@gmail.com

Ideia geral do artigo

O artigo[**inproceedings**] tem como finalidade analisar as linguagens de programação Java e C/C++ com relação ao tempo de execução de alguns algoritmos. Esses algoritmos tinham como objetivo emular estresse à pilha de execução, à memória primária, à memória secundária e à unidade lógica e aritmética. Com isso, o estudo fez uma comparação entre uma linguagem compilada e uma linguagem interpretada mas que utiliza compilação dinâmica.

Problema estudado e solução

Foi dado um foco maior na metodologia dos testes aplicados. Foram realizados os mesmos testes nos compiladores GCC e JDK, com uma versão recente e uma defasada de cada um. Também usaram optimizações durante a fase de compilação, versões anteriores de cada compilador e as melhores implementações possíveis de cada algoritmo.

Também foi utilizado o Projeto Fatorial $2^k r$, que analisa os efeitos de k fatores, cada qual com dois níveis. Sendo os fatores: compilação customizada e versões do compilador. Java mostrou uma evolução com relação ao compilador com o passar do tempo, enquanto a otimização do compilador mostrou-se necessária para a linguagem C/C++.

Análise dos trabalhos relacionados

O artigo consegue passar informações prévias sobre o problema durante todas as partes do texto, mas principalmente em *Trabalhos Relacionados*. Essa sessão apresentou, de forma resumida, algumas pesquisas que trazem

resultados muito interessantes. Uma dessas, relacionada a optimização da linguagem Java, conclui que é possível que a linguagem tenha um melhor desempenho em comparação a C/C++ por causa da compilação dinâmica.

Pontos fortes e pontos fracos

O questionamento sobre a eficiência da linguagem Java em relação a linguagens compiladas como C/C++ foi o ponto de partida do artigo, e por isso acredito que a abordagem metodologica serviu para responder, de forma empírica, a questão. Mas também acredito que faltou uma abordagem mais aprofundada com relação ao resultado do algoritmo de Busca Sequencial, em que Java obteve uma pequena vantagem com relação a C/C++.

Uma das justificativas é a compilação dinâmica, que permite um ganho de performance porque parte do código é compilado e recompilado durante a execução do programa, trazendo certas optimizações que não estão disponíveis durante a compilação estática em troca de um período maior de inicialização.

Em alguns casos, parte da compilação dinâmica é feita em momentos posteriores a inicialização, causando alguns travamentos a mais. Mas, mesmo assim, o compilador Just-in-Time ainda assim é uma ferramenta essencial para a performance de programas Java. O tema realmente merece uma pesquisa à parte.

=======

Resumo

1

¿¿¿¿¿¿¿ refs/remotes/origin/main