Comparação do Desempenho de Diferentes Compiladores em Ambientes Virtualizados através de Aplicações Paralelas

Jonatha P. Silveira[1], **Arthur C. Silveira**[1], **Arthur F. Lorenzon**[2]

[1]Faculdade São Francisco de Assis – Porto Alegre – RS – Brasil

[2]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre – RS – Brasil

{jonatha.psilveira, arth.csil}@gmail.com, aflorenzon@inf.ufrgs.br

Resumo. *A subutilização de recursos computacionais pode ser evitada com a virtualização, onde diferentes versões de sistemas operacionais e aplicações podem ser executadas em paralelo. Assim, um número maior de aplicações paralelas serão executadas em ambientes virtualizados, as quais podem ser compiladas com diferentes compiladores. Através da execução de aplicações paralelas compiladas com diferentes compiladores em dois monitores de máquinas virtuais, este trabalho mostra a influência da escolha do compilador e do monitor de máquina virtual no desempenho da aplicação paralela.*

1. Introdução

Com o aumento do poder computacional e a possível subutilização dos recursos disponíveis, a virtualização está se tornando cada vez mais comum [Lorenzon et al. 2009]. Através dela, é possível aproveitar o hardware disponível de um único computador para a execução de sistemas operacionais com objetivos diferentes, como por exemplo, aplicações paralelas [Barham et al. 2003]. No entanto, cada monitor de máquina virtual (MMV) possui características distintas, e que impactam de maneira diferente no comportamento do desempenho de aplicações paralelas. Adicionalmente, as aplicações podem ter seu código de máquina gerado por diferentes compiladores. Historicamente, o sistema operacional Linux usa como compilador padrão o *GNU Compiler Collection* (GCC), devido a sua capacidade em gerar código eficiente. Porém recentemente, o compilador *Low Level Virtual Machine* (LLVM) vem ganhando momento e tornando-se mais eficaz em gerar código eficiente.

Desta forma, este trabalho apresenta um estudo experimental com relação ao desempenho dos principais compiladores utilizados hoje em dia (GCC e LLVM), em ambientes virtualizados. Através da execução de quatro aplicações do NAS Parallel Benchmark em dois diferentes monitores de máquinas virtuais (Oracle VirtualBox e VMWare), este artigo mostra que o GCC apresenta desempenho superior na maioria dos casos, enquanto que LLVM possui melhor escalabilidade.

O restante do artigo está organizado da seguinte maneira. A Seção 2 discute os trabalhos relacionados. Os compiladores utilizados e o ambiente de execução são brevemente apresentados na Seção 3. A Seção 4 discute os principais resultados obtidos, enquanto as conclusões e oportunidades de trabalhos futuros são discutidos na Seção 5.

2. Trabalhos Relacionados

O impacto de diferentes compiladores na execução de aplicações paralelas em ambientes virtualizados é pouco estudado hoje em dia. Os autores em [Younge et al. 2011] comparam o desempenho entre monitores de máquinas virtuais para sistemas de alto desempenho. Os resultados mostram que, para a execução do benchmark SPEComp, VirtualBox

possui desempenho similar a execução em uma máquina com hardware real. Por outro lado, [Li 2010] comparam VirtualBox e VMWare de maneira qualitativa, sem a execução de aplicações paralelas.

Considerando o uso de processadores reais, o trabalho desenvolvido em [Park et al. 2014] mostra que o LLVM possui bons resultados em aplicações que são CPU-intensiva, enquanto que o GCC é até 18% mais rápido em aplicações que usam bastante a memória (alocação de estruturas de dados), devido a otimizações de saltos e alocação de registradores. Por outro lado, [Kim et al. 2010] destaca a capacidade do LLVM obter melhor desempenho na execução de chamadas de funções frequentes. Recentemente, [Krause et al. 2017] avaliam o comportamento de aplicações paralelas compiladas com diferentes compiladores, entre eles, LLVM e GCC. Os resultados mostram que o compilador GCC em sua versão 6.2.0 obteve melhor desempenho que o LLVM (Clang 3.9.0) e o ICC 16.0.4. No entanto, o trabalho não considera o uso de ambientes virtualizados.

Considerando os trabalhos destacados, observa-se um espaço de exploração de projeto com relação ao uso de diferentes compiladores para compilar aplicações paralelas que serão executadas em ambientes virtualizados. Desta maneira, este artigo colabora com o estado da arte, comparando o desempenho de aplicações paralelas compiladas com compiladores amplamente utilizados em ambientes virtualizados.

3. Metodologia

3.1. Compiladores

Um compilador pode ser definido como um programa de sistema que traduz um programa descrito em uma linguagem de alto nível para um programa equivalente em código de máquina para um processador [Aho et al. 2006]. Em geral, um compilador não produz diretamente o código de máquina, mas sim um programa em linguagem simbólica (*assembly*), que é então traduzido para o programa em linguagem de máquina através de montadores. Para desempenhar suas tarefas, um compilador deve executar dois tipos de atividade. A primeira atividade é a análise do código fonte, onde a estrutura e o significado do programa de alto nível são reconhecidos. A segunda atividade é a síntese do programa equivalente em *assembly*.

O GCC é um conjunto de compiladores de linguagens de programação produzido pelo projeto GNU para construir um sistema operacional. Ele tem sido adotado como compilador preferencial para o desenvolvimento de softwares que necessitam ser executados em vários tipos de hardware. Ao utilizar os compiladores do projeto GCC, o mesmo analisador gramatical é usado em todas as plataformas, fazendo com que se o código compila numa, muito provavelmente compilará em todas [Stallman 1988].

LLVM é um compilador amplamente modular que favorece a implementação e a experimentação com diversas análises e transformações de programas. Ele é uma infraestrutura de compilação escrita em C++, criada na Universidade de Illinóis, no início de 2000. Quando se usa LLVM, aplicações são compiladas para programas na linguagem intermediária de LLVM, chamada LLVM-IR. A infraestrutura de compilação LLVM possui também back-ends para diversos processadores, sendo capaz de gerar código para diversas arquiteturas distintas. Com LLVM pode-se, por exemplo, traduzir um programa C para LLVM-IR e após gerar código objeto para executar em um processador MIPS, X86, X86-64, entre outros [Lattner and Adve 2004].

3.2. Ambiente de Execução

Quatro aplicações do conjunto de *benchmarks* paralelos NAS foram utilizados: *conjugated gradient* (CG), *discrete 3D fast Fourier Transform* (FT), *integer sort* (IS), e *Unstructured Adaptative mesh* (UA); todos eles na classe de entrada C. Eles foram executados com diferentes números de *threads* (1, 2, 4 e 8) em cada configuração (MMV e compilador). A versão do GCC utilizada foi a 7.2.0 enquanto que a versão do LLVM (clang) foi a 6.0.0.

Cada MMV (Virtual Box 5.2.0 e VMWare 14.0.0) foi configurado com as seguintes características: processador de 8 núcleos, 8 GB RAM, e sistema operacional Linux Ubuntu 16.04, *kernel* v. 4.4.0. Adicionalmente, o sistema hospedeiro consiste de um processador Intel Core i7 com suporte à execução simultânea de 8 *threads* e 16 GB de memória RAM. Os resultados apresentados na próxima seção consideram a média aritmética da execução de 10 vezes de cada configuração (aplicação, número de *threads*, compilador e MMV). Em todos os casos, o desvio padrão foi inferior a 1% do tempo total de execução.

4. Resultados Experimentais

A Figura 1 apresenta os resultados obtidos para cada aplicação. O eixo x de cada gráfico representa a execução com diferentes números de *threads*, enquanto que o eixo y, o tempo de execução, em segundos para cada configuração (compilador + MMV).

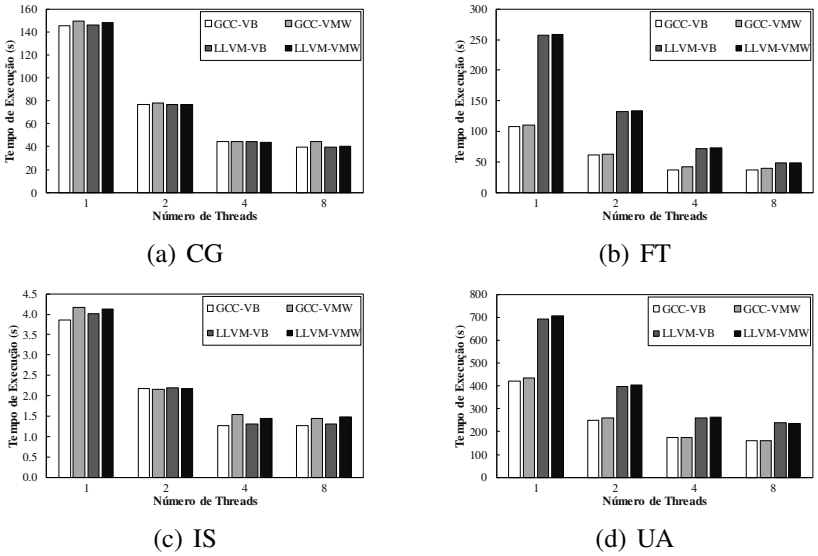
Comparando o desempenho dos monitores de máquinas virtuais, na maioria dos casos, o VirtualBox apresentou desempenho superior (i.e., tempo de execução menor) ao obtido pelo VMWare. Considerando a média geométrica de todas as aplicações, o VirtualBox executou as aplicações compiladas com o GCC 6% mais rápido que o VMWare. Por outro lado, esta diferença é reduzida para apenas 2% quando o compilador LLVM é utilizado.

Quando comparamos o desempenho dos dois compiladores, na maioria dos casos o GCC apresentou menor tempo de execução. Por outro lado, pode-se observar na Figura 1, que quanto maior o número de *threads*, menor é a diferença entre GCC e LLVM. Por exemplo, na execução com apenas 1 *thread* da aplicação FT (Figura 1b, GCC é 58% mais rápido que LLVM. No entanto, na execução com 8 *threads*, a diferença diminui para 20%, quando considerado o MMV VMWare. Isto mostra que, embora GCC possua melhores resultados, o LLVM apresenta melhor escalabilidade quando ocorre o aumento do número de *threads*.

Por fim, os resultados mostram que, enquanto LLVM fornece desempenho similar independente do MMV utilizado, o GCC apresenta melhores resultados no VirtualBox. Portanto, se o programador estiver utilizando o VirtualBox para virtualizar um sistema operacional para execução de aplicações paralelas, o indicado é o uso do compilador GCC.

5. Conclusão

Este trabalho realizou uma comparação de desempenho entre dois compiladores amplamente utilizados pela comunidade acadêmica, em ambientes virtualizados. A partir da execução de um conjunto de benchmarks paralelos, mostrou-se que o GCC apresenta melhor resultado que LLVM na maioria dos casos, independente do MMV utilizado. No entanto, observou-se que o LLVM apresenta melhor escalabilidade que o GCC. Como trabalhos futuros, pretende-se estudar as diferentes otimizações de compiladores e seus efeitos em ambientes virtualizados. Também será expandido o ambiente de execução para compreender outros compiladores (i.e., Intel *compiler*) e outros monitores de máquinas virtuais, tais como o Xen.



(a) CG (b) FT

(c) IS (d) UA

Figura 1. Resultados obtidos (VB: VirtualBox; VMW: VMWare)

Referências

Aho, A. V., Lam, M. S., Sethi, R., and Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools (2Nd Edition)*.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. In *ACM Symp. on Operating Systems Principles*, pages 164–177.

Kim, J. J., Lee, S. Y., Moon, S. M., and Kim, S. (2010). Comparison of llvm and gcc on the arm platform. In *Int. Conf. on Embedded and Multimedia Computing*, pages 1–6.

Krause, A. M., Moro, G. B., and Schnorr, L. M. (2017). Análise do consumo energético de aplicações paralelas com diferentes versões de compiladores. In *Escola Regional de Alto Desempenho - RS*.

Lattner, C. and Adve, V. (2004). Llvm: A compilation framework for lifelong program analysis & transformation. In *Int. Symp. on Code Generation and Optimization: Feedback-directed and Runtime Optimization*, CGO '04.

Li, P. (2010). Selecting and using virtualization solutions: Our experiences with vmware and virtualbox. *J. Comput. Sci. Coll.*, 25(3):11–17.

Lorenzon, A., P., D., and Rossi, F. D. (2009). Um estudo sobre o xen e a portabilidade de seu escalonamento para arquiteturas paralelas. In *Escola Regional de Alto Desempenho*.

Park, C., Han, M., Lee, H., and Kim, S. W. (2014). Performance comparison of gcc and llvm on the eisc processor. In *Int. Conf. on Electronics, Information and Communications*, pages 1–2.

Stallman, R. M. (1988). Using the gnu compiler collection.

Younge, A. J., Henschel, R., Brown, J. T., von Laszewski, G., Qiu, J., and Fox, G. C. (2011). Analysis of virtualization technologies for high performance computing environments. In *IEEE Int. Conf. on Cloud Computing*, pages 9–16.