Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática
Bacharelado em Ciência da Computação

Trabalho de Conclusão de Curso - TCC 2010

Tcl JIT

Área de Concentração: Compiladores
Aluno: Guilherme Henrique Polo Gonçalves
Orientador: Prof. Dr. Anderson Faustino da Silva

1 Objetivos gerais

Melhorar o desempenho de programas escritos na linguagem de programação `Tcl` (*Tool Command Language*) através da criação e implantação de um compilador JIT (*Just-In-Time*) na mesma.

2 Justificativas

Diversos trabalhos já demonstraram a possibilidade de aumentar o desempenho de linguagens de programação com a aplicação de compilação dinâmica. Alguns desses projetos serão mencionados, e acredita-se que a linguagem de programação `Tcl` possa também obter resultados relevantes.

A implementação da linguagem `Smalltalk-80` descrita em [Deutsch and Schiffman 1984] traduz dinamicamente bytecodes da máquina virtual da `Smalltalk-80` para código nativo, obtendo, dependendo da estratégia utilizada, entre 1,45 e quase 2,0 vezes melhor tempo de execução quando comparado a utilização de interpretação pura.

Um outro projeto, o Psyco [Rigo 2004], demonstra resultados excelentes em casos específicos de aritmética de inteiros, reduzindo em 109 vezes o tempo de execução, e também em aritmética de ponto flutuante, obtendo melhoria de 10,9 vezes no tempo gasto, para a linguagem de programação `Python`.

Já a linguagem `Java` tem recebido considerável atenção em relação ao tópico discutido e vários projetos utilizam um compilador dinâmico juntamente com a máquina virtual com o objetivo de melhorar seu desempenho. Um exemplo é a Jalapeño JVM (*Java Virtual Machine*), onde o trabalho descrito em [Alpern et al. 2000] apresenta uma comparação entre seu tempo de execução utilizando um compilador otimizador dinâmico contra interpretação da JVM IBM Developer Kit. Nessa comparação, a Jalapeño demonstra vantagens de 7 a 29 vezes no tempo de execução.

3 Breve fundamentação

Compilação JIT se refere a tradução de código sob-demanda. No caso desse trabalho, estamos interessados na tradução de *bytecodes* da máquina virtual `Tcl` [Ousterhout 1989] para código de máquina durante o tempo de execução.

Algumas linguagens, como `SELF` [Ungar and Smith 1987] ou `Tcl`, são difíceis de se-

rem analisadas e compiladas estaticamente de forma eficiente. Uma das causas pode ser a utilização de tipagem dinâmica. O resultado é que tais linguagens não conseguem alcançar bom desempenho devido, em partes, ao *overhead* causado pela interpretação. São nessas linguagens de programação onde um compilador JIT se mostra mais necessário, pois a utilização de informações coletadas durante a execução do programa torna possível a geração de código mais eficiente.

Um sistema que aplica compilação dinâmica funciona, basicamente, de uma das formas seguintes. No caso do `SELF-93` [Hölzle 1994], um método é convertido em código nativo antes de se sua invocação. Neste caso, a interpretação nunca é utilizada. Essa tradução é feita de forma rápida, fazendo com que o tempo de execução total não seja muito prejudicado, porém o código não é tão eficiente. Entretanto, métodos que executarem muito frequentemente notificarão o sistema de recompilação que, então, determina se o compilador otimizador deve ser utilizado ou não. Uma outra forma é a utilização de um modo misto de execução, onde interpretação e execução de código nativo se alternam. O `YAPc` [da Silva and Costa 2007] funciona dessa forma, o que, no caso do `Prolog`, significa que apenas as cláusulas utilizadas mais frequentemente serão compiladas.

Independente da forma de funcionamento implementada, um sistema JIT elimina parte do *overhead* da interpretação para que o programa obtenha melhor desempenho. Tempo de decodificação e interpretação de *bytecodes* podem ser significantemente reduzidos, enquanto que tarefas relacionadas a, por exemplo, sincronização ou alocação de memória não são diretamente afetados pelo JIT.

Referências

[Alpern et al. 2000] Alpern, B., Attanasio, C. R., Barton, J. J., Burke, M. G., Cheng, P., Choi, J.-D., Cocchi, A., Fink, S. J., Grove, D., Hind, M., Hummel, S. F., Lieber, D., Litvinov, V., Mergen, M. F., Ngo, T., Russell, J. R., Sarkar, V., Serrano, M. J., Shepherd, J. C., Smith, S. E., Sreedhar, V. C., Srinivasan, H., and Whaley, J. (2000). The jalapeño virtual machine. *IBM Syst. J.*, 39(1):211–238.

[da Silva and Costa 2007] da Silva, A. F. and Costa, V. S. (2007). Design, implementation, and evaluation of an dynamic compilation framework for the yap system. In Dahl, V. and Niemelä, I., editors, *Proceedings of the 23rd International Conference on Logic Programming*, volume 4670 of *Lecture Notes in Computer Science*. Springer.

[Deutsch and Schiffman 1984] Deutsch, L. P. and Schiffman, A. M. (1984). Efficient implementation of the Smalltalk-80 system. In *Conference Record of the Eleventh*

Annual ACM Symposium on Principles of Programming Languages, pages 297–302, Salt Lake City, Utah.

[Hölzle 1994] Hölzle, U. (1994). *Adaptive Optimization for SELF: Reconciling High Performance with Exploratory Programming*. PhD thesis, Stanford, CA, USA.

[Ousterhout 1989] Ousterhout, J. K. (1989). Tcl: An embeddable command language. Technical Report UCB/CSD-89-541, EECS Department, University of California, Berkeley.

[Rigo 2004] Rigo, A. (2004). Representation-based just-in-time specialization and the psyco prototype for python. In *PEPM '04: Proceedings of the 2004 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation*, pages 15–26, New York, NY, USA. ACM.

[Ungar and Smith 1987] Ungar, D. and Smith, R. B. (1987). Self: The power of simplicity. In *OOPSLA '87: Conference proceedings on Object-oriented programming systems, languages and applications*, pages 227–242.

Guilherme Henrique Polo Gonçalves

Prof. Dr. Anderson Faustino da Silva

Maringá, 15 de julho de 2010