Caracterização do perfil de carga a partir de programas binários

Renê S. Pinto[1], **Alexandre C. B. Delbem**[1], **Francisco J. Monaco**[1]

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (USP)
Av. Trabalhador São Carlense, 400 – Caixa Postal 668 – São Carlos – SP – Brazil

{rene,acbd,monaco}@icmc.usp.br

Abstract. *Knowledge of the platform resources demanded by a computer application is a relevant piece of information for performance engineering purposes. Well-known available methods are based either on measurement, requiring instrumentation and execution of applications, or on analytical source code analysis, which may turn difficult for complex application. This study presents the use of DAMICORE technique for clustering executable codes regarding their resource consumption profile. The results showed that this approach was applied successfully over a repository with 80 executable programs to distinguish between CPU-Intensive and IO-Intensive applications.*

Resumo. *O conhecimento do perfil de consumo de recursos pelos aplicativos de computadores é uma informação importante para propósitos de engenharia de performance. Os principais métodos disponíveis são baseados tanto em aferição, necessitando instrumentação e execução dos programas avaliados, quanto em análise estática de código fonte, o que pode tornar-se de difícil avaliação em aplicações complexas. Este artigo apresenta o uso da técnica DAMICORE para promover agrupamento de códigos executáveis (programas) de acordo com o perfil de carga dos mesmos. Os resultados gerados mostraram a viabilidade do uso da técnica, permitindo a sua aplicação em um repositório de 80 programas executáveis para distingui-los entre uso intenso de CPU ou Entrada/Saída.*

1. Introdução

O conhecimento dos recursos demandados por um programa durante sua execução é uma informação útil em engenharia de desempenho a qual pode ser utilizada para diversos propósitos, desde o projeto de políticas de Qualidade de Serviço (QoS), balanceamentos de carga, alocação elástica de recursos até análise de consumo energético[Noureddine et al. 2016]. Os principais métodos para caracterização do perfil de carga de softwares inserem-se basicamente em duas categorias: baseados em *identificação* ou *inferência*. A primeira compreende as técnicas que enxergam o software como uma caixa-preta e avaliam somente os dados obtidos através do monitoramento do consumo de recursos e outros parâmetros durante a execução do mesmo. A aferição em tempo de execução requer cuidados na instrumentação do sistema (em alguns casos do próprio software) para garantir que o impacto gerado pela aferição não influencie nos resultados ou possa ser bem determinado. A segunda categoria compreende técnicas de

inferência que tentam fazer a predição de perfil de carga através de abordagens alternativas à aferição. A análise estática do código fonte, seja por métodos manuais ou automatizados, é um dos meios adotados nestas abordagens. Entretanto, avaliar o consumo de recursos somente pela análise do código fonte torna-se uma tarefa difícil, especialmente em códigos grandes e complexos. Apesar da possibilidade de aplicar análises estáticas em códigos objeto, a análise estática é majoritariamente desenvolvida para códigos fontes e, portanto, dependem da linguagem de programação adotada além de não considerarem otimizações de compilação que poderão estar presentes no código binário da aplicação.

1.1. Objetivos

Este trabalho tem como objetivo introduzir o uso do método de aprendizagem de máquina não supervisionada DAMICORE como uma abordagem alternativa para análise e caracterização do perfil de carga de programas executáveis somente através da análise de seus respectivos códigos binários. Os aspectos que distinguem a abordagem proposta de demais existentes para o mesmo propósito são os fatos de que: o método DAMICORE opera sobre código binário em lugar de código-fonte; b) a técnica não depende de seleção de características via conhecimento a priori para determinar correlações entre código e desempenho (as características são determinadas implicitamente pelo próprio método); c) o resultado do método é uma filogenia que descreve relações hierárquicas de similaridade entre programas.

2. DAMICORE

O método DAMICORE (DAta MIning of Code REpositories)[Sanches et al. 2011] compreende um arcabouço teórico composto por várias técnicas que combinadas são capazes de fornecer relações hierárquicas entre objetos de dados não estruturados. O processo é efetuado através de três passos: a) construir uma matriz de distâncias comparando cada dois objetos do repositório de acordo com uma métrica de similaridade; b) converter a matriz de distâncias em uma rede que conecta os objetos de acordo com suas similaridades; c) aplicar um processo de detecção de comunidades para detecção de grupos de elementos similares dentro da rede. Originalmente três algoritmos são utilizados nestes processos: *Normalized Compression Distance* (NCD) para a construção da matriz de distâncias, Neighbor Joining (NJ)para produzir uma árvore filogenética a partir da matriz de distâncias e o método Fast Newman (FN) para detecção de comunidades na árvore gerada. O cálculo da NCD é o passo do DAMICORE que viabiliza sua aplicação a qualquer tipo de dado sem requerer informação semântica sobre o mesmo. A NCD é uma aproximação computável da complexidade de Kolmogorov, definida pela equação 1.

$$NCD_z(a,b) = \frac{C_z(ab) - min\{C_z(a), C_z(b)\}}{max\{C_z(a), C_z(b)\}} \quad (1)$$

Os dois objetos a serem comparados são denominados por a e b, sendo ab a concatenação de ambos e $C_z(x)$ é o tamanho da versão compactada do objeto x obtido através de um algoritmo de compressão Z. Para um compressor ideal e dois arquivos idênticos, $C_z(ab) = C_z(a) = C_z(b)$ e, portanto, $NCD_z(a,b) = 0$, enquanto para dois arquivos totalmente diferentes, $C_z(ab) = C_z(a) + C_z(b)$ e logo $NCD_z(a,b) = 1$. Uma vez que a compressão é efetuada em nível de cada byte dos dados, o método DAMICORE

torna-se agnóstico quanto à semântica dos dados. A aplicação do algoritmo NJ na matriz de distâncias produz uma árvore filogenética dos objetos do repositório tornando visual relações não evidentes na matriz de distâncias. Por fim, o método é completado pela aplicação do FN para detecção de agrupamentos (comunidades) dos elementos semelhantes na árvore, destacando as relações mais fortes entre os objetos em estudo.

3. Experimentos

Para avaliar a aplicação do DAMICORE em programas binários foram considerados o consumo de CPU (Processamento) e E/S (Operações de Entrada/Saída). É importante observar que esta abordagem pode ser generalizada para outros recursos, tais como memória, consumo de energia, etc. Para compor o repositório de dados, 80 programas foram coletados de diversas fontes (trabalhos acadêmicos, repositórios de softwares livres ou implementados) sendo 40 programas determinados *CPU-Intensive*, ou seja, aplicações consumindo majoritariamente apenas recursos de processamento, e 40 programas determinados *IO-Intensive*, que fazem majoritariamente diversas operações de E/S especialmente em arquivos.

Todos os programas foram obtidos respectivamente com seus códigos fonte (Linguagem C), compilados e executados sob o mesmo ambiente: Sistema Operacional Linux 64 bits, hardware composto por uma placa mãe Intel DG31PR, processador Intel Core2Quad, 2GB de memória RAM e disco SATA de 5400 RPM. Posteriormente foi executada uma extensiva análise dinâmica (aferições e *benchmarks*) em cada um dos programas escolhidos para assegurar se seus respectivos perfis realmente possuíam características de uma das categorias escolhidas (*CPU ou IO-Intensive*). O DAMICORE foi aplicado sobre todo o repositório dos programas binários e sobre amostras compostas de forma aleatória a partir do repositório original. Para avaliar a robustez do método, o DAMICORE também foi aplicado a repositórios compostos pelos mesmos programas binários, porém compilados com parâmetros de otimizações diferentes (-O0, -O1, -O2, -O3 e -Os) e diferentes compiladores (gcc 5.4.0, icc 15.0.6 e llvm 3.8.1).Para cada árvore gerada e cada um de seus elementos, foi calculada a distância d_{raiz} que representa a distância entre um elemento da árvore e a raiz. Assim, se as árvores geradas para cada nível de otimização forem semelhantes, o desvio padrão das distâncias será baixo, mostrando a robustez do agrupamento independente dos níveis de otimização ou compiladores utilizados.

4. Resultados

A Figura 1 contém o gráfico com o desvio padrão da distância d_{raiz} para cada binário compilado com diferentes níveis de otimização e compilador. O máximo desvio padrão encontrado foi da ordem de 0,04 (4%). Já a Figura 2 contém uma árvore filogenética resultante da aplicação do DAMICORE para um subconjunto de 20 binários do repositório original escolhidos aleatoriamente (denotados por p_n). A linha tracejada foi traçada manualmente para demarcar os programas *CPU-Intensive* e *IO-Intensive*. Todos os programas *IO-Intensive* foram agrupados na árvore (acima da linha tracejada) assim como todos os programas *CPU-Intensive* foram agrupados abaixo da linha tracejada. Outras aplicações do DAMICORE sobre todo o repositório dos binários e diferentes amostras alcançaram resultados semelhantes, distribuindo as aplicações nas árvores de acordo com seus respectivos perfis de carga.

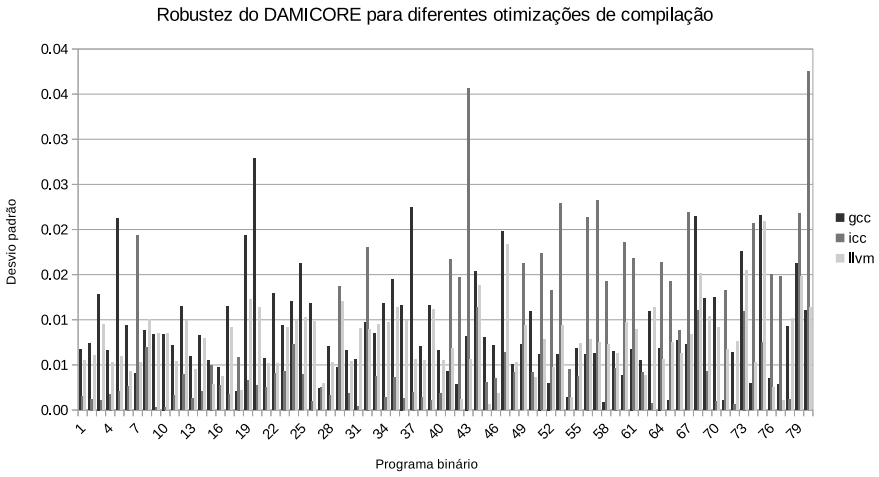

Figura 1. Desvio padrão de d_{raiz} para os compiladores gcc, icc e llvm.

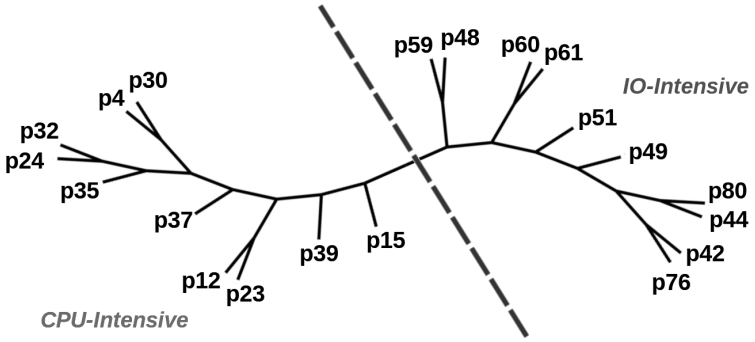

Figura 2. Árvore filogenética com análise de um subconjunto de 20 binários do repositório.

5. Conclusões

Dado que programas binários são uma representação em instruções de máquina de seus respectivos códigos fontes e algoritmos implementados, programas que apresentem comportamento semelhantes (em relação ao consumo de recursos) tenderão a utilizar instruções semelhantes. Neste contexto a metodologia imposta pela técnica DAMICORE pode ser utilizada uma vez que é agnóstica à semântica dos dados analisados. Os resultados confirmaram a viabilidade da técnica considerando consumo de recursos de CPU e Entrada/Saída com robustez para diferentes níveis de otimizações e compiladores. Este resultado é um passo importante para desenvolvimento de estratégias para construção de classificadores e estimadores de consumo de recurso.

Referências

Noureddine, A., Islam, S., and Bashroush, R. (2016). Jolinar: analysing the energy footprint of software applications. In *The International Symposium on Software Testing and Analysis*, pages Pages–445.

Sanches, A., Cardoso, J. M., and Delbem, A. C. (2011). Identifying merge-beneficial software kernels for hardware implementation. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, pages 74–79. IEEE.