AVALIANDO O IMPACTO DO DESDOBRAMENTO DE LAÇOS NA EFICIÊNCIA DO CÓDIGO EMBARCADO

SILVA, Wellisson G. P.
Universidade Federal de Pelotas

CORRÊA, Ulisses B.
Universidade Federal do Rio Grande do Sul

BRISOLARA, Lisane B.
Universidade Federal de Pelotas

1 INTRODUÇÃO

Sistemas embarcados complexos possuem restrições severas quanto ao desempenho, tamanho de memória e consumo de potência e de energia (GRAAF, *et al*. 2003). Essas propriedades estão normalmente mais ligadas ao hardware utilizado, porém a forma como o software interage com os recursos do sistema têm um impacto no consumo de potência e energia assim como no desempenho e na quantidade de memória requerida (SAXE, 2010). Além disso, a indústria de sistemas embarcados é movida por custos e pelo *time-to-market,* que é o tempo para que um projeto esteja pronto para entrar no mercado e competir, e, portanto, estes dois pontos influenciam as decisões que os desenvolvedores de software embarcados precisam tomar enquanto projetam o sistema.

Para lidar com a crescente complexidade dos sistemas embarcados e seus requisitos, o uso de linguagens orientadas a objetos têm se tornado mais importante, principalmente pela sua capacidade de modularização e reutilização de código, além da maior facilidade de manutenção. Porém, tais linguagens podem introduzir penalidades como aumentos no consumo de potência e energia do sistema embarcado e reduções no desempenho (CHATZIGEORGIOU; STEPHANIDES, 2002). Assim, os projetistas de software embarcado precisam lidar com a complexidade do sistema, e ainda produzir um software eficiente.

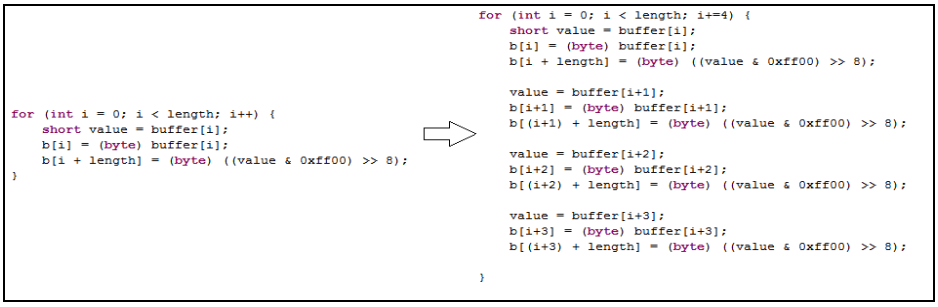
Refatoração é uma técnica de engenharia de software que modifica o código para melhorar a legibilidade e manutenibilidade do código sem alterar a computação por ele realizada (FOWLER, 2004). Dentre as refatorações e otimizações com potencial uso em sistemas embarcados podemos destacar o *inline* e o desdobramento de laços (em inglês, *loop unrolling*). No entanto, os impactos destas modificações nas métricas físicas do sistema embarcado precisam ser investigados a fim de auxiliar os projetistas na escolha das modificações que permitirão obter uma solução que atenda os requisitos do sistema. Em (SILVA, 2010) o uso de *inline* foi investigado e seu impacto em desempenho e consumo pode ser observado. Neste trabalho, o objetivo é avaliar qual é o impacto da aplicação de *loop unrolling*, um conhecido método de otimização de código, em uma aplicação Java embarcada.

2 METODOLOGIA

Este trabalho utiliza informações obtidas da análise do comportamento dinâmico das aplicações através de instrumentação de código. Desta análise,

obtém-se o número de chamadas dos métodos, o que pode ser usado para determinar aqueles métodos que mais consomem recursos durante sua execução. Então, são realizadas modificações no código de forma incremental, iniciando pelos métodos que mais consomem tempo de execução do aplicativo. Assim, o método mais chamado foi modificado na primeira iteração, após, o segundo método mais chamado sofreu alterações com base no código resultante da primeira iteração, e assim por diante. Estas modificações geram várias versões do código da aplicação que devem ser avaliadas quanto às métricas físicas a fim de observar o impacto da mudança.

A modificação investigada é conhecida como desdobramento de laço (em inglês *loop unrolling*, ou *loop unwinding*), e consiste no desdobramento de laços visando reduzir a sobrecarga de controle dos laços de repetição. O desdobramento pode ser total ou parcial. Neste último, o número de iterações de um laço é reduzido através da execução de mais operações por passo, diminuindo assim a quantidade de instruções de controle requeridas. Esta técnica possibilita uma otimização, por que reduz as instruções de controle que são mais custosas em termos de tempo na execução. A Figura 1 ilustra um exemplo de *loop unrolling* parcial.



```
for (int i = 0; i < length; i++) {
    short value = buffer[i];
    b[i] = (byte) buffer[i];
    b[i + length] = (byte) ((value & 0xff00) >> 8);
}
```

```
for (int i = 0; i < length; i+=4) {
    short value = buffer[i];
    b[i] = (byte) buffer[i];
    b[i + length] = (byte) ((value & 0xff00) >> 8);

    value = buffer[i+1];
    b[i+1] = (byte) buffer[i+1];
    b[(i+1) + length] = (byte) ((value & 0xff00) >> 8);

    value = buffer[i+2];
    b[i+2] = (byte) buffer[i+2];
    b[(i+2) + length] = (byte) ((value & 0xff00) >> 8);

    value = buffer[i+3];
    b[i+3] = (byte) buffer[i+3];
    b[(i+3) + length] = (byte) ((value & 0xff00) >> 8);

}
```

Figura 1 - Exemplo da aplicação do *loop unrolling*.

Para obter as informações de desempenho e consumo de energia, principais métricas do software embarcado, para as várias versões do código geradas pela aplicação do *loop unrolling,* foi utilizada uma ferramenta de estimativa chamada DESEJOS (MATTOS; CARRO, 2007). Como estes dados só podem ser obtidos com relativa precisão quando uma arquitetura alvo é definida, o processador FemtoJava (ITO et al. 2001) foi considerado como plataforma alvo. Este processador é uma implementação da máquina virtual Java baseada em pilha, que executa Java *bytecodes* nativamente. Nos experimentos, foi utilizada a versão multiciclo do processador FemtoJava, que é mais indicada para aplicações embarcadas devido a seu baixo consumo de potência. Este cenário de investigação foi escolhido devido à crescente atenção que a linguagem Java vem recebendo na comunidade de sistemas embarcados.

3 RESULTADOS E DISCUSSÕES

A aplicação alvo usada nos experimentos, um decodificador de áudio no formato *MPEG layer-3* (MP3), é disponibilizada no conjunto de *benchmarks* SPECJVM2008, que é um conjunto de aplicações desenvolvidas para medir o desempenho de uma máquina virtual JAVA e o hardware que a executa (SPECJVM2008).

A Figura 2 mostra os resultados de desempenho obtidos após cada alteração realizada no código, considerando os ciclos necessários para a execução do aplicativo. Como esperado, a aplicação do *loop unrolling* reduziu o número de ciclos do processador requeridos pela aplicação, já que, com a modificação realizada, mais instruções simples foram executadas por passo e foram diminuídas as instruções de controle do laço que são mais complexas e tomam mais tempo de processamento. Com a redução dos ciclos, a refatoração proposta causa uma redução no consumo de energia, o que pode ser observado na Figura 3. Esta redução é equivalente, em cada iteração, à redução obtida em número de ciclos.

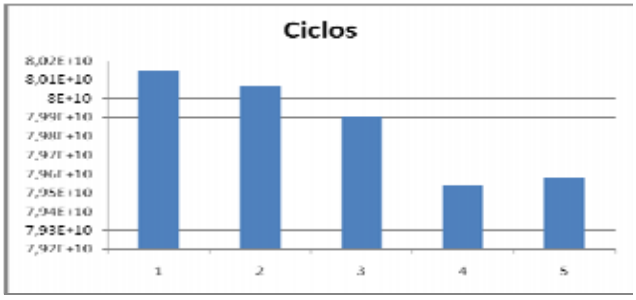


Figura 2 - Número de ciclos para a execução dos códigos resultantes

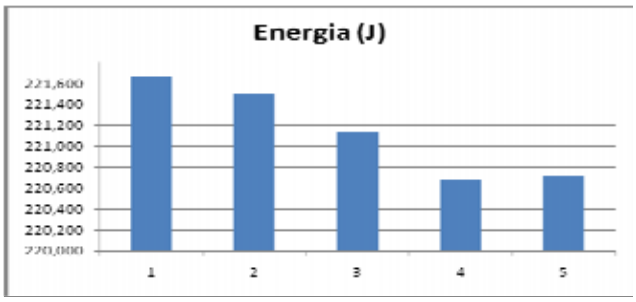


Figura 3 - Resultados de consumo de energia para código resultante

Porém, como pode ser observado nos resultados, a tendência de redução, tanto no consumo de energia quanto no número de ciclos, não ocorre na modificação do quarto código para o quinto. Isso se deve ao fato de que a aplicação de *loop unrolling,* apresentada na Figura 1, realizada para gerar a versão 5 causou uma sobrecarga para manter a semântica do programa. Nesta modificação, um acesso a uma posição de um arranjo descrito por buffer[i] foi substituído por duas instruções extras, quando modificado para buffer[i+1]. Enquanto a expressão buffer[i] é convertida em uma única instrução (*iload*) que carrega seu conteúdo da memória para o topo da pilha de operandos, a expressão buffer[i+1] é convertida em uma série de instruções, que compreendem a inserção de uma constante na pilha (*iconst_1*), a soma da constante com o índice i (*iadd*) e só então a carga da posição adequada do arranjo para o topo da pilha.

4 CONCLUSÕES

Este trabalho investigou a aplicação de uma técnica de refatoração, o *loop unrolling*, ao código Java de uma aplicação embarcada. O conhecimento do impacto de mudanças no código é importante para guiar os projetistas durante a

otimização do software, auxiliando na busca por uma solução que atenda os requisitos da aplicação e respeite as restrições da plataforma alvo.

Com base nos resultados encontrados, foi possível concluir que o *loop unrolling* tem um impacto positivo na redução do número de ciclos para a execução de um aplicativo, além disso, possibilita uma redução no consumo de energia do mesmo, que é uma característica importante a ser considerada no desenvolvimento de um sistema embarcado. No entanto, assim como outras modificações no código, o *loop unrolling* deve ser aplicado cuidadosamente, pois em alguns casos pode causar aumento em consumo de ciclos e de energia. Isso demonstra que existe espaço para a pesquisa de heurísticas específicas, para a linguagem Java, de aplicação de otimizações, visto que o seu compilador não realiza otimizações importantes para o domínio de sistemas embarcados.

5 AGRADECIMENTO

Agradecimento à FAPERGS pela bolsa de iniciação que suportou a realização deste trabalho de investigação.

6 REFERÊNCIAS

CHATZIGEORGIOU, A.; STEPHANIDES, G. Evaluating Performance and Power of Object-Oriented vs. Procedural Programming in Embedded Processors. In: **INTERNATIONAL CONFERENCE ON RELIABLE SOFTWARE TECHNOLOGIES**, Vienna, Austria, 2002. ADA-2002, Proceedings…

FOWLER, M. **Refactoring: Improving the Design of Existing Code**, Addison Wesley, 2004.

GRAAF, B.; LORMANS, M.; TOETENEL, H. Embedded Software Engineering: the State of the Practice. **IEEE Software**, v. 20, n. 6, p. 61- 69, 2003.

ITO, S. A.; CARRO, L.; JACOBI, R. P. Making java work for microcontroller applications. **IEEE Design & Test of Computers**, v.18, n. 5, p.100-110, 2001.

MATTOS, J.C.B.; CARRO, L. Object and Method Exploration for Embedded Systems Applications. In: **SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN**, Rio de Janeiro, Brazil, 2007. Symposium on integrated circuits and system design, Proceedings…

SAXE, E. Power Efficient Software. **Communication of the ACM**. v. 53, n. 02, 2010.

SILVA, W. G. P. ; CORREA, U. B. ; BRISOLARA, L. B. ; CARRO, L. Evaluation of the impact of code refactoring on embedded software efficiency. **WORKSHOP DE SISTEMAS EMBARCADOS**, Gramado, Brasil, 2010. Workshop de sistemas embarcados, Proceedings…

SPECJVM2008 (Java Virtual Machine Benchmark), http://www.spec.org/jvm2008.