Explorando o reúso aproximativo em diferentes níveis

Leonardo Almeida da Silveira
Universidade Federal do Rio Grande do Sul
Instituto de Informática
Orientado por Prof. Dr. Antonio Carlos S. Beck Filho
lasilveira@inf.ufrgs.br

Motivação

Sistemas computacionais são cada vez mais utilizados não só para a automatização de diversas tarefas no dia a dia da população, mas também para aplicações específicas que exigem alto desempenho. Este trabalho explora uma alternativa para melhorar a performance de um nicho de aplicações que toleram pequenas falhas, como o processamento de imagens.

Reúso

Em organização de processadores, uma técnica existente para o aumento de performance é o reúso. Ele se baseia na ideia de que um segmento de código que recebe a mesma entrada produzirá sempre a mesma saída. Com isso, se o par de entradas e saídas para esse trecho for armazenado, não será necessário executar a sequência de instruções novamente, pois o resultado já estará disponível de uma execução anterior. Seguindo essa linha, a técnica pode ser aplicada em vários níveis, como instrução, basic block e função. O foco deste trabalho é a implementação de reúso de função em software.

Function Reuse em Software

O *function reuse* em software tem por objetivo substituir a execução de uma função inteira por uma consulta a uma tabela sempre que o reúso for possível, isto é, se as entradas da função já possuem suas saídas armazenadas de uma computação anterior. Porém, devido à grande variedade de entradas possíveis normalmente em uma função, essa tabela deveria ser extremamente grande para que a taxa de reúso fosse significativa, tornando inviável a prática do reúso em nível de função de forma **exata**. Ao aplicar técnicas de **aproximação**, é possível reduzir consideravelmente o tamanho da tabela necessário para obtermos uma taxa de reúso aceitável, à custa da precisão dos resultados. Em aplicações que toleram erros, como processamento de imagens, observa-se um potencial para aplicar a técnica de reúso aproximativo de função, **trocando a qualidade da saída pelo desempenho**. Por isso, foram analisadas duas aplicações, que realizam codificação JPEG e detecção de bordas: *jpeg* e *sobel*, respectivamente. Em cada uma, foi escolhida uma função que representa grande parte da execução total do programa.

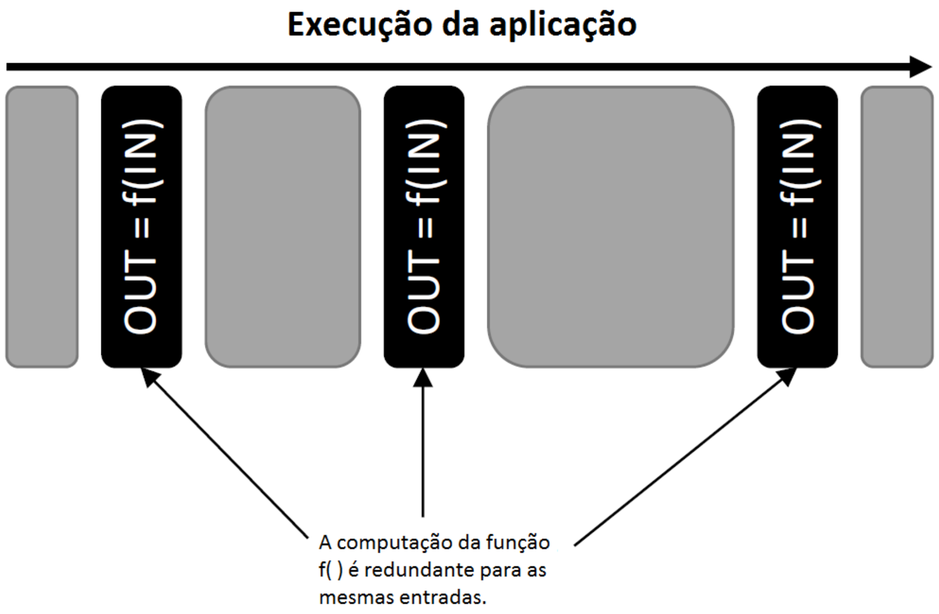

Fig. 1. Fluxo de uma aplicação em que a mesma função é executada com a mesma entrada várias vezes.

Implementação com Aproximação

- Foram empregados dois cálculos diferentes para a chave usada na tabela:
  - **Soma-todos**: a chave é a soma de todos os elementos da entrada da função.
  - **SHA1**: função criptográfica, com a característica de que pequenas variações nos parâmetros de entrada geram chaves completamente distintas.
- No caso da *SHA1*, diferentes níveis de quantização foram aplicados às entradas da função analisada em cada *benchmark*.

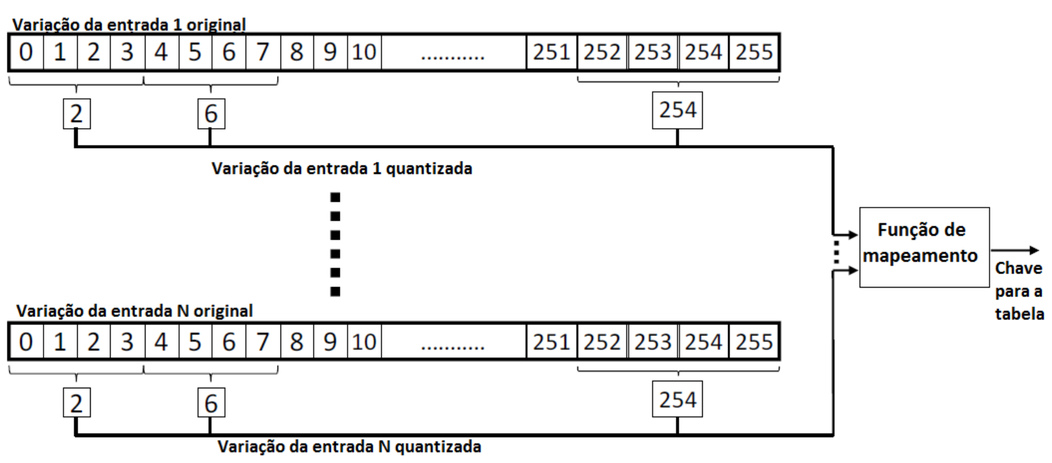

Fig. 2. Quantização do conjunto de entradas antes do cálculo da chave pela função de mapeamento.

Resultados

A Fig. 3 mostra que é possível ganhar em desempenho se o tempo de cálculo da chave e de acesso à tabela for menor que o custo da função original. Em suas versões originais, *jpeg* leva 1450 ciclos e *sobel* leva 300.

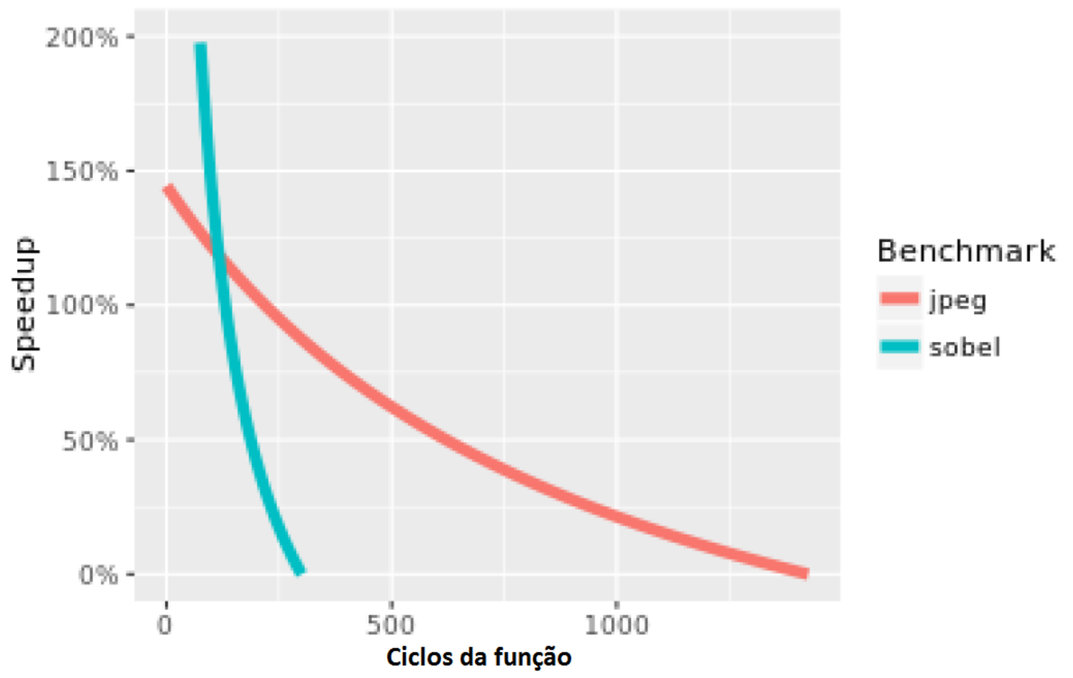

Fig. 3. Potencial *speedup* das aplicações ao empregar a técnica do *function reuse*.

Ao reduzir os níveis de quantização, a taxa de reúso aumentou significativamente, ao passo que a qualidade das imagens geradas não foi severamente afetada. No *jpeg* foi possível alcançar 50% de reúso com pouco mais de 5% de perda de qualidade. No *sobel*, esses valores são de quase 100% e 27%, respectivamente.

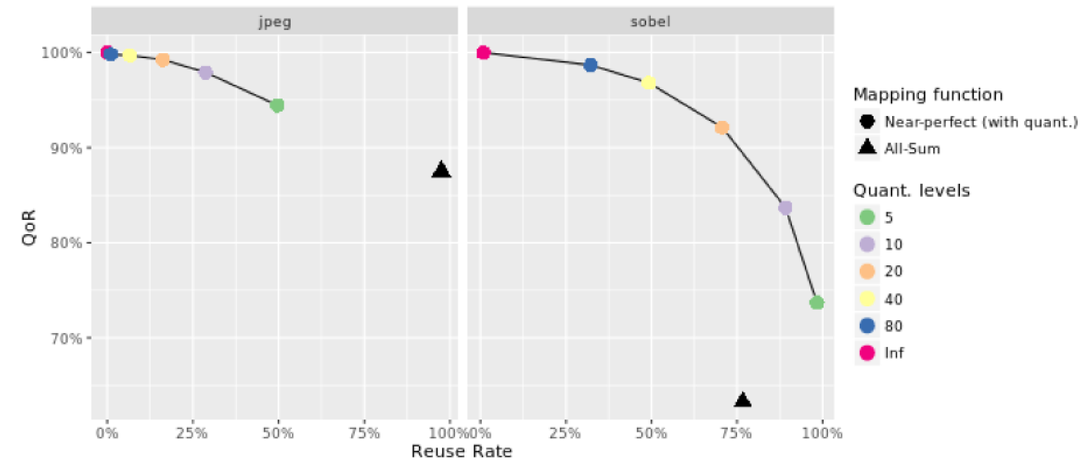

Fig. 4. *Tradeoff* entre a taxa de reúso e a qualidade das imagens geradas variando a função de mapeamento e os níveis de quantização das entradas.

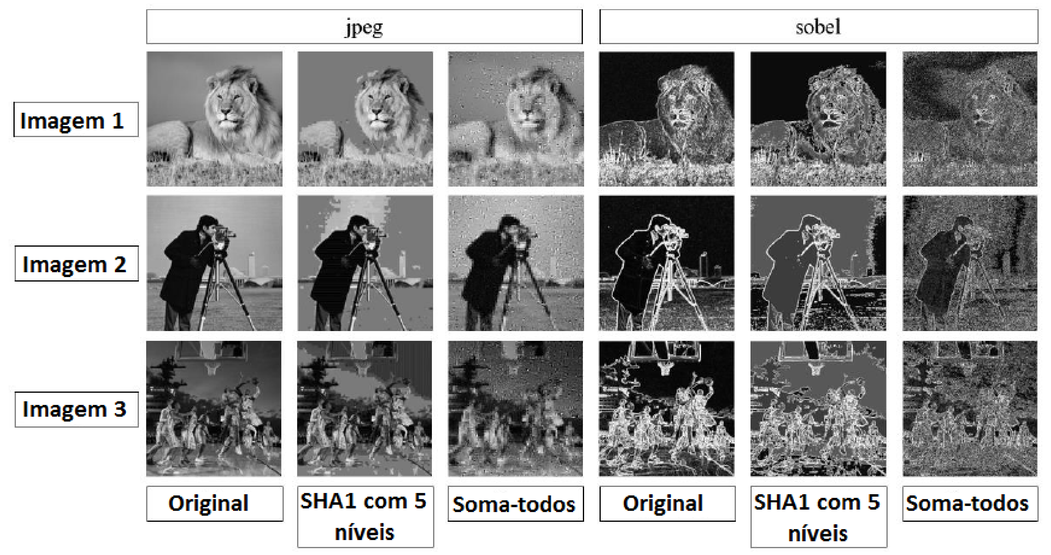

Fig. 5. Imagens produzidas em três versões distintas: original (sem reúso) e utilizando as duas funções de mapeamento propostas - *Soma-todos* e *SHA1* - com 5 níveis de quantização.

Sobre o trabalho

- **Conclusões**:
  O emprego de técnicas aproximativas para o reúso possibilita um potencial ganho de desempenho em aplicações que toleram falhas em seus resultados, reduzindo a complexidade e o tamanho da tabela necessária em comparação ao reúso **sem aproximação**.
- **Trabalhos futuros**:
  Observar o impacto de outras funções de mapeamento.
  Analisar o custo de acesso à tabela em software e/ou hardware com diferentes implementações.