Uma biblioteca para síntese automática de operações especializadas em funções de *hash* usando *value profiling*

João Vitor Fröhlich

6 de abril de 2025

1 Introdução

Funções de *hash* são muito importantes para a computação. Elas estão presentes em diversos sistemas computacionais, como por exemplo sistemas operacionais, sistemas de arquivos, servidores DNS, redes sociais e criptografia. Em especial, funções de *hash* não criptográficas são muito utilizadas por sua capacidade de buscar rapidamente por uma informação, independente da quantidade, e por sua escalabilidade conforme o crescimento da aplicação onde ela é utilizada.

Contudo, a busca por um elemento pode ser piorada dependendo da forma como a função de *hash* é implementada. Isto porque estas funções trabalham a partir do mapeamento de um valor de entrada, representado por meio de uma sequência de *bytes* de tamanho arbitrário, para um valor de saída, de tamanho fixo. Num mundo ideal, todos os valores seriam mapeados para um valor diferente mas, como a entrada pode ser infinita e a saída é de tamanho fixo, isto nem sempre acontece. Se uma função mapeia muitos valores de entrada para um mesmo valor de saída, então sua principal qualidade, a velocidade pela busca de informação, é perdida, pois será necessária uma busca sequencial pela informação.

Em alguns casos, os valores de entrada possuem padrões que podem ser explorados pelas funções de *hash*, de forma que permita sua otimização. Por exemplo, um CPF é representado no formato XXX.XXX.XXX-XX, onde X é um número inteiro de 0 a 9. Como '.' e '-' estão presentes em todos os CPFs, eles podem ser ignorados e apenas os 11 dígitos X serem considerados pela função. Dessa forma, a velocidade da função pode melhorar e o número de colisões diminuir.

Nesse sentido, é possível especializar essas funções através de um compilador por meio da técnica de *value profiling*. Um compilador é uma aplicação que traduz um código escrito em uma linguagem qualquer para instruções a serem executadas pelo computador, visando otimizar estas instruções. Já a técnica de *value profiling* consiste em verificar os valores que são modificados durante a execução do programa, o que permite encontrar padrões nos valores que são dados como entrada para o programa. O compilador seria capaz de analisar os padrões, identificar as funções de *hash* do programa original e produzir funções de *hash* especializadas.

Assim, o objetivo dese trabalho consiste em implementar e testar uma ferramenta capaz de sintetizar automaticamente funções de *hash* baseado no conhecimento sobre as chaves *hash* dessa função, afim de otimizar essa função para seu contexto na aplicação, seja aumentando a sua velocidade ou reduzindo o número de colisões.

2 Referencial Teórico

Uma função de *hash* é uma função que transforma um valor de entrada de tamanho arbitrário em um valor de *hash* de tamanho fixo. A entrada de uma função é chamada chave *hash*, enquanto que o resultado da função é chamado valor de *hash*. Estas funções podem ser utilizadas, por exemplo, para otimizar consultas em tabelas e em bancos de dados e funções de criptografia.

As funções de *hash* podem ser criptográficas (FHC) ou não criptográficas (FHNC), onde uma FHC tem como principal característica a dificuldade computacional de reverter o processo de *hashing*, enquanto uma FHNC tem como característica o baixo número de colisões. O foco deste trabalho serão as FHNCs.

Segundo [5], existem 4 critérios principais para medir a qualidade de uma FHNC, sendo eles: a resistência a colisões, que se refere à probabilidade de existirem duas chaves *hash* que produzam o mesmo valor de *hash*; a distribuição de resultados, que mede o quão próximo os valores de *hash* da função seguem uma distribuição uniforme; o efeito avalanche, que se refere à capacidade de produzir uma grande mudança no valor de *hash* a partir de uma pequena modificação em uma chave *hash*; e a velocidade, que considera o quão rápido a função executa. A ideia de otimizar uma função de *hash*, portanto, tem como foco melhorar um ou mais destes aspectos, a depender do contexto da aplicação.

Uma forma de se melhorar o desempenho de uma função de *hash* consiste em especializar uma função para um dado conjunto de dados. Essa especialização poderia ser realizada uma vez que o conjunto de dados de entrada fosse analisado

e alguns padrões fossem encontrados. Em [3], por exemplo, uma função de *hash* especializada para um conjunto de endereços IP é elaborada utilizando algoritmos evolutivos. Já em [6], um método é proposto e analisado utilizando *linear genetic programming* para especializar funções de *hash* visando o aumento na velocidade destas funções.

Na literatura são encontradas diversas funções de *hash* não criptográficas, cuja qualidade depende da aplicação onde esta função é utilizada. Algumas destas funções são: **FNV**, **lookup3**, **SuperFastHash**, **MurmurHash2**, **DJBX33A**, **BuzHash**, **DEK**, **BKDR** e **APartow**. Como a complexidade de desenvolver uma nova FHNC é muito alta, alguns trabalhos focam na evolução de funções já existentes para gerar novas FHNC. É o caso de trabalhos como [1], que produziu 2 FHNCs genéricas usando *grammatical evolution*, [4], que propôs e analisou um método de geração de FHNCs genéricas utilizando programação genética, e [7], que propôs e analisou um método de especialização de FHNCs baseado no conhecimento do conjunto de chaves, utilizando *linear genetic programming*.

O diferencial deste trabalho consiste no desenvolvimento de uma biblioteca de otimização para compiladores, capaz de sintetizar automaticamente uma FHNC especializada se baseando no conhecimento do valor das chaves de *hash*, sendo que esse conhecimento é adquirido através de *value profiling*. Segundo [2], a técnica de *value profiling* consiste em identificar variáveis que sofrem poucas mudanças durante a execução de programas. Um dos possíveis usos dessa técnica envolve otimizar o código gerado no processo de compilação a partir da especialização do código para os valores encontrados.

3 Metodologia

A proposta para a realização deste trabalho consiste em implementar e modificar as ferramentas destacadas em negrito a seguir, para realizar o seguinte procedimento.

Inicialmente, a partir de uma aplicação P, o **compilador** geraria uma aplicação P', que registraria os valores de chaves *hash* através de *value profiling*.

A partir de uma análise desses valores, um **analisador de código** agruparia as chaves em uma estrutura a ser definida. Esta estrutura precisa ser capaz de agrupar as chaves de um formato específico a um formato mais geral.

Então, um **sintetizador de código**, focado em especializar funções de *hash*, produziria versões das funções de *hash* contidas originalmente na aplicação P, com foco em otimizar algum dos aspectos descritos em [5].

Por fim, o **compilador** modifica P com as funções de *hash* geradas no passo anterior, produzindo uma aplicação otimizada P''.

Para permitir a síntese, será necessário criar também um catálogo de especializações de funções de *hash* comuns. Alguns exemplos de especialização, baseados em alguns padrões de chaves *hash* são: substrings comuns a todas as chaves *hash*; dado uma codificação das chaves *hash* para um valor inteiro, a distribuição dessas chaves se concentra majoritariamente em um intervalo pequeno; e se todas as chaves são constituídas por sequências de dígitos ASCII.

4 Cronograma

Para cumprir os objetivos propostos, o trabalho será dividido em X etapas, cada uma focando em um tema distinto, sendo estas etapas e temas:

(E1) Teoria e implementação de funções de *hash*

(E2) Análise de padrões em conjuntos de dados distintos

(E3) Catalogação de funções de *hash* especializadas

(E4) Implementação da biblioteca de geração de código

(E5) Escrita da dissertação

Assim, supondo um prazo de um semestre para a execução de cada etapa, prazo este que pode ser maior ou menor, o cronograma proposto para a execução do projeto de mestrado é o seguinte:

Etapas	**Primeiro Ano**		**Segundo Ano**	
	1º Semestre	**2º Semestre**	**1º Semestre**	**2º Semestre**
E1				
E2				
E3				
E4				
E5				

Além disso, para adquirir o conhecimento necessário para a execução desta proposta de mestrado, e visando a oferta recente de disciplinas do programa,

pensa-se em cursar as seguintes disciplinas: Análise Estática de Programas, Cibersegurança e Métodos Formais. Além disso, será cursada a disciplina obrigatória e Projeto e Análise de Algoritmos e, por fim, a disciplina de Programação Competitiva, que mesmo não tendo muita relação direta com o conteúdo da dissertação, acredita-se que ajudará a expandir o conhecimento sobre algoritmos avançados da computação e auxiliará em entrevistas de emprego.

Não será apresentado um cronograma para a realização das disciplinas, pois isto depende da oferta de disciplinas em cada semestre. Além disso, a lista de disciplinas pode ser alterada durante o período do mestrado.

Referências

[1] BERARDUCCI, P., JORDAN, D., MARTIN, D., AND SEITZER, J. Gevosh: Using grammatical evolution to generate hashing functions. In Proceedings of the Fifteenth Midwest Artificial Intelligence and Cognitive Sciences Conference, MAICS (2004), Citeseer, pp. 16–18.

[2] CALDER, B., FELLER, P., AND EUSTACE, A. Value profiling. In Proceedings of 30th Annual International Symposium on Microarchitecture (1997), IEEE, pp. 259–269.

[3] DOBAI, R., KORENEK, J., AND SEKANINA, L. Evolutionary design of hash function pairs for network filters. Applied Soft Computing 56 (2017), 173–181.

[4] ESTEBANEZ, C., SAEZ, Y., RECIO, G., AND ISASI, P. Automatic design of noncryptographic hash functions using genetic programming. Computational Intelligence 30, 4 (2014), 798–831.

[5] ESTÉBANEZ, C., SAEZ, Y., RECIO, G., AND ISASI, P. Performance of the most common non-cryptographic hash functions. Software: Practice and Experience 44, 6 (2014), 681–698.

[6] GROCHOL, D., AND SEKANINA, L. Evolutionary design of fast high-quality hash functions for network applications. In Proceedings of the Genetic and Evolutionary Computation Conference 2016 (2016), pp. 901–908.

[7] SAEZ, Y., ESTEBANEZ, C., QUINTANA, D., AND ISASI, P. Evolutionary hash functions for specific domains. Applied Soft Computing 78 (2019), 58–69.