Desempenho de Tabela Hash

Bruno M. R. Klemz, Jorge S. T. Jordão

[1]Curso de Ciência da Computação – Pontifícia Universidade Católica do Paraná (PUCPR)
R. Imac. Conceição, 1155 - Prado Velho, Curitiba - PR, 80215-901

mezzomo.klemz@pucpr.edu.br, jorge.jordao@pucpr.edu.br

Abstract. *The aim of this study was to understand the performance of 27 possible combinations for the hash table based on three parameters: dataset size, the type of hash function used for both insertion and search, and the available hash table size. In addition to variations in size, the hash functions used were division, multiplication, and folding. The dataset sizes varied among 1 million, 5 million, and 20 million records to observe how each table handles insertions and searches, considering the time taken and the number of collisions. Finally, the results of all these combinations were compiled in tables and simplified into graphs to compare performance differences.*

Resumo. *O objetivo do trabalho em questão foi entender o desemepenho de 27 combinações possíveis para a tabela hash a partir de três parâmetros: o tamanho do conjunto de dados, o tipo de hash a ser utilizado tanto na inserção quanto na busca, e o tamanho disponível na tabela hash. Além das variações de tamanho, os hashes utilizads foram: divisão, multiplicação e dobramento. Para o tamanho de conjuntos de dados houve variação dos seguintes valores, 1 milhão, 5 milhões e 20 milhões de registros para ver como cada tabela lida com inserções e buscas, observando o tempo gasto e as colisões. Por fim, o resultado de todas essas combinações puderem ser armazenados em tabelas e simplificados em gráficos para comparar as diferenças de desempenho.*

1. Introdução

Uma tabela hash é uma estrutura de dados que permite armazenar e acessar dados de forma eficiente, associando chaves a valores. Seu funcionamento baseia-se em uma função chamada de função hash, que recebe uma chave (como um número ou uma palavra) e a transforma em um índice na tabela. Esse índice determina a posição onde o dado será armazenado ou acessado, tornando o processo de busca e inserção mais rápido em comparação com outras estruturas, como listas ou arrays.

Ela é muito útil para ocasiões em que a quantidade de dados é imensa, como é o caso dessa pesquisa que lida com conjuntos de dados dos seguintes tamanhos: 1 milhão, 5 milhões e 20 milhões, considerando que a variação de números em cada um ocorre entre zero e 1 bilhão.

A partir disso, foram gerados as tabelas hash com os seguintes valores: 500 mil, 750 mil e 999 mil, apresentando a quantidade necessária para armazenar so dados. Porém essa mapeamento foram utilizados três funções de hash: divisão, multiplicação e dobramento.

Hash por Divisão

- Baseia-se na operação de **resto da divisão** para determinar o índice na tabela.
- A fórmula básica é:
$$h(k) = km$$
onde k é a chave e m é o tamanho da tabela hash.
- É simples de implementar e eficiente, mas depende da escolha de um valor m adequado (preferencialmente um número primo) para reduzir colisões.

Hash por Multiplicação

- Utiliza uma **constante de multiplicação** para gerar um índice, independente do tamanho da tabela.
- A fórmula básica é:
$$h(k) = \lfloor m \cdot (k \cdot A1) \rfloor$$
onde A é uma constante entre 0 e 1. Uma escolha comum para A é $\frac{\sqrt{5}-1}{2} \approx 0.6180339887$.
- Garante uma distribuição uniforme dos índices, mas é mais complexa e exige uma constante A bem escolhida.

Hash por Dobramento

- Divide a chave em partes menores e **soma** essas partes para obter o índice.
- Exemplo: uma chave numérica 123456 pode ser dividida em 123 e 456, e o índice seria $123 + 456$.
- Ideal para chaves numéricas grandes e para reduzir padrões, mas pode ser menos eficiente com dados muito variados.

Nesse sentido, durante a inserção com os métodos apresentados enfrentamos um problema comum: colisões, que ocorrem quando duas chaves diferentes geram o mesmo índice. A partir disso algumas técnicas foram desenvolvidas para lidar com esse caso, dentre elas a de lista encadeada ou por rehashing. No nosso caso foi utilizado o tratamento de colisões por lista encadeada, onde cada posição da tabela hash armazena uma lista de elementos em vez de apenas um valor. Quando ocorre uma colisão, o novo elemento é adicionado ao final da lista existente naquela posição. Dessa forma, múltiplos valores que compartilham o mesmo índice podem coexistir sem sobrescrever uns aos outros.

Quando feita a inserção a partir da combinação de um conjunto, tipo de hash e tabela hash, podemos fazer a chamada busca de hashing, o que permite acesso rápido e eficiente a dados a partir da própria função hash selecionada, e depois contabilizado o total de comparações feitas na lista encadeada da posição calculada para verificar onde está localizado a chave.

2. Objetivos

- Apresentar todas as combinações possíveis a partir dos parâmetros tamanho de conjuntos de dados, tipo de hash e tabela hash
- Calcular o tempo de execução para cada combinação a partir da inserção e busca
- Apresentar os resultados em gráficos e tabelas

3. Resultados

A seguir iremos apresentar os resultados em uma tabela mostrando todas as combinações e o tempo levado em milissegundos levado para cada ocorrência. Além disso, os valores buscados em cada combinação foram os mesmos para fazer uma comparação assertiva. A lista de valores escolhidos aleatoriamente pelo programa foi ”919061487”, ”656540772”, ”266198740”, ”923186538” e ”500235818”.

Importante salientar que esses valores foram escolhidos apenas durante a primeira combinação, com 1 milhão de dados no conjunto.

Tabela de combinações

Table 1. Comparação de funções hash com diferentes tamanhos de tabelas e quantidades de elementos

Tamanho da Tabela	Função Hash	Conjunto de Dados	Colisões	Tempo (ms)
500000	Divisão	1000000	567821	297
500000	Multiplicação	1000000	567953	454
500000	Dobramento	1000000	997083	16443
500000	Divisão	5000000	4500016	2542
500000	Multiplicação	5000000	4500032	3993
500000	Dobramento	5000000	4997039	402226
500000	Divisão	20000000	19500000	26420
500000	Multiplicação	20000000	19500000	29927
500000	Dobramento	20000000	19997019	2935538
750000	Divisão	1000000	447608	63
750000	Multiplicação	1000000	447837	406
750000	Dobramento	1000000	997083	23151
750000	Divisão	5000000	4250936	1251
750000	Multiplicação	5000000	4250972	3047
750000	Dobramento	5000000	4997039	731922
750000	Divisão	20000000	19250000	18109
750000	Multiplicação	20000000	19250000	22970
750000	Dobramento	20000000	19997019	2989880
999999	Divisão	1000000	367833	47
999999	Multiplicação	1000000	368422	312
999999	Dobramento	1000000	997083	23507
999999	Divisão	5000000	4006868	969
999999	Multiplicação	5000000	4006788	2673
999999	Dobramento	5000000	4997039	469284
999999	Divisão	20000000	19000001	13129
999999	Multiplicação	20000000	19000001	22871
999999	Dobramento	20000000	19997019	4317450

Agora iremos apresentar cinco tabelas, mostrando as 27 combinações e os valores buscados incluindo codigo, posição, numéro de comparações e o tempo de busca

Código	Posição	Nº de Comparações	Tempo de Busca (ms)
919061487	61487	1	9900
919061487	358509	0	2300
919061487	1467	163	50699
919061487	61487	1	7200
919061487	358509	0	2000
919061487	1467	163	18300
919061487	311487	1	2600
919061487	537764	0	2000
919061487	1467	163	35600
919061487	311487	1	2500
919061487	537764	0	2100
919061487	1467	163	17700
919061487	62406	0	1600
919061487	717018	0	1500
919061487	1467	163	26500
919061487	62406	0	35200
919061487	717018	0	600
919061487	1467	163	1500
919061487	311487	1	4699
919061487	537764	0	4000
919061487	1467	163	12700
919061487	62406	0	1400
919061487	717018	0	1300
919061487	1467	163	19700

Table 2. Resultados de busca para o código 919061487

Código	Posição	Nº de Comparações	Tempo de Busca (ms)
656540772	40772	1	3200
656540772	48047	1	1800
656540772	1968	76	7200
656540772	40772	1	899
656540772	48047	1	1300
656540772	1968	76	5500
656540772	40772	1	1300
656540772	48047	1	2700
656540772	1968	76	8200
656540772	290772	0	500
656540772	72071	0	1000
656540772	1968	76	14100
656540772	290772	0	800
656540772	72071	0	1500
656540772	1968	76	5599
656540772	290772	0	800
656540772	72071	0	1000
656540772	1968	76	4500
656540772	541428	0	600
656540772	96095	0	700
656540772	1968	76	1556600
656540772	541428	0	500
656540772	96095	0	600
656540772	1968	76	900
656540772	541428	0	200
656540772	96095	0	700
656540772	1968	76	1000

Table 3. Resultados de busca para o código 656540772

Código	Posição	Nº de Comparações	Tempo de Busca (ns)
266198740	198740	0	900
266198740	41198	1	1100
266198740	1204	206	22201
266198740	198740	0	800
266198740	41198	1	1100
266198740	1204	206	13200
266198740	698740	0	500
266198740	61797	1	1200
266198740	1204	206	40100
266198740	698740	0	900
266198740	61797	1	1600
266198740	1204	206	15099
266198740	199006	0	500
266198740	82396	1	900
266198740	1204	206	31900
266198740	199006	0	300
266198740	82396	1	600
266198740	1204	206	15800
266198740	61487	1	9900
266198740	40772	1	899
266198740	48047	1	1300
266198740	1968	76	5500
266198740	62406	0	35200
266198740	717018	0	600
266198740	311487	1	4699
266198740	537764	0	4000

Table 4. Resultados de busca para o código 266198740

Código	Posição	Nº de Comparações	Tempo de Busca (ns)
923186538	186538	0	7300
923186538	220173	1	1200
923186538	1647	203	18000
923186538	186538	0	701
923186538	220173	1	1000
923186538	1647	203	13000
923186538	686538	1	1600
923186538	330259	0	600
923186538	1647	203	45000
923186538	686538	1	13200
923186538	330259	0	700
923186538	1647	203	13399
923186538	187461	0	500
923186538	440346	1	700
923186538	1647	203	24900
923186538	187461	0	300
923186538	440346	1	500
923186538	1647	203	1400
923186538	61487	1	7200
923186538	40772	1	899
923186538	48047	1	1300
923186538	1968	76	5500
923186538	1467	163	26500
923186538	62406	0	35200
923186538	717018	0	600
923186538	311487	1	4699
923186538	537764	0	4000

Table 5. Resultados de busca para o código 923186538

Código	Posição	Nº de Comparações	Tempo de Busca (ns)
500235818	235818	1	1200
500235818	457053	0	1100
500235818	1553	505	62201
500235818	235818	1	900
500235818	457053	0	700
500235818	1553	505	31300
500235818	735818	1	900
500235818	685579	0	800
500235818	1553	505	84700
500235818	735818	1	1000
500235818	685579	0	1100
500235818	1553	505	33301
500235818	236318	1	1000
500235818	914105	0	900
500235818	1553	505	62500
500235818	236318	1	500
500235818	914105	0	600
500235818	1553	505	26400
500235818	61487	1	7200
500235818	40772	1	899
500235818	48047	1	1300
500235818	1968	76	5500
500235818	62406	0	35200
500235818	717018	0	600
500235818	311487	1	4699
500235818	537764	0	4000

Table 6. Resultados de busca para o código 500235818

4. Gráficos de desempenho

Logo abaixo apresentamos os resultados referentes a execução de cada tipo de hash

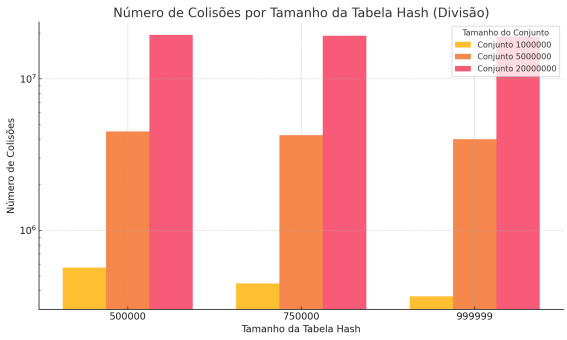

Figure 1. Número De Colisões Por Tamanho Da Tabela Hash DIVISAO

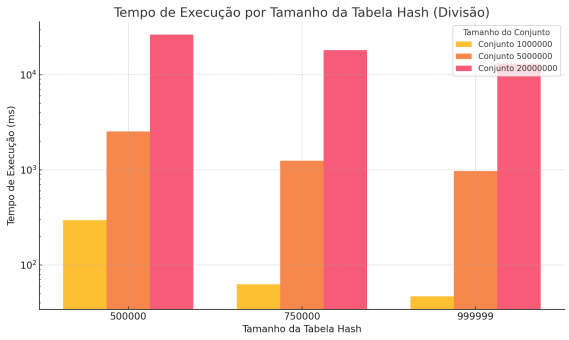

Figure 2. Tempo De Execução Por Tamanho Da Tabela Hash DIVISAO

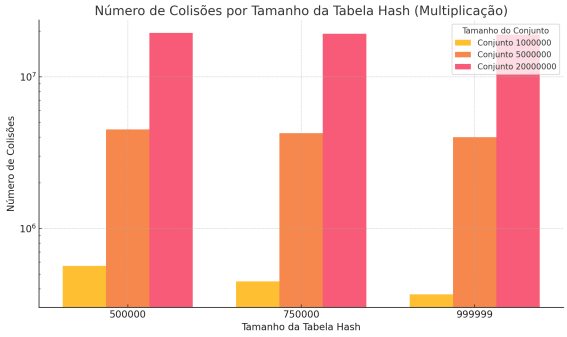

Figure 3. Número De Colisões Por Tamanho Da Tabela Hash MULTIPLICAÇÃO

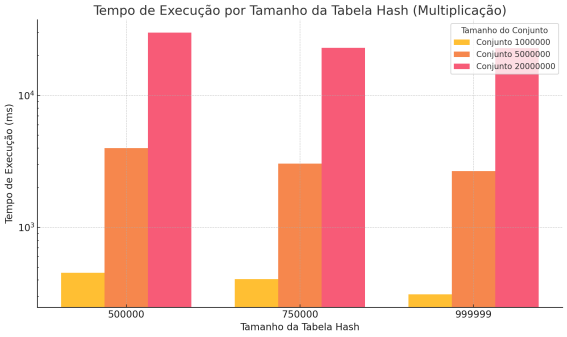

Figure 4. Tempo De Execução Por Tamanho Da Tabela Hash MULTIPLICAÇÃO

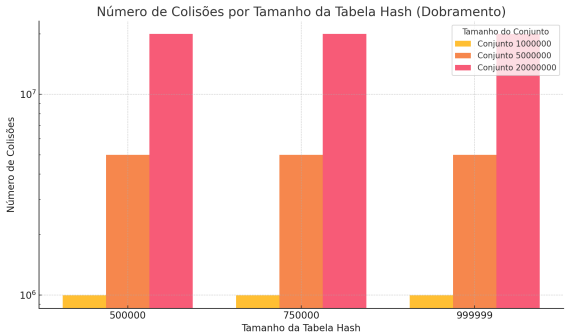

Figure 5. Número De Colisões Por Tamanho Da Tabela Hash DOBRAMENTO

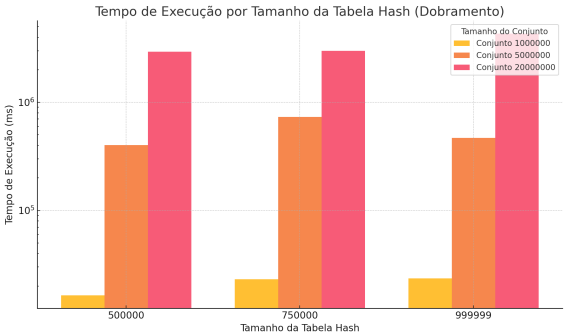

Figure 6. Tempo De Execução Por Tamanho Da Tabela Hash DOBRAMENTO

5. Conclusão

Em suma, percebemos que a função de hash de Divisão se destacou como a mais eficiente para os cenários que avaliamos. Ela se mostrou especialmente eficaz em situações onde rapidez e controle de colisões são essenciais. A função de Multiplicação também apresentou resultados razoáveis, mas ainda ficou atrás da Divisão em termos de eficiência. Já a função de Dobramento teve o pior desempenho entre as três, gerando um número alto de colisões e levando mais tempo para processar. Esses resultados indicam que a função de Dobramento não é a melhor escolha para aplicações que exigem alto desempenho e um baixo índice de colisões.