Estudo do efeito de variações de *Bloom filters* no desempenho de algoritmos de hifenização de palavras

Matheus Barbosa Silva

14 de Dezembro de 2022

Supervisor: Prof. Dr. Guilherme Oliveira Mota
Supervisor: Prof. Dr. Yoshiharu Kohayakawa
Departamento de Ciência da Computação - IME USP

Sumário

Sumário

Introdução

- **O armazenamento e manipulação de grandes volumes de dados é essencial para muitos setores.**

Introdução

- **O armazenamento e manipulação de grandes volumes de dados é essencial para muitos setores.**
  - O Facebook armazena mais de 300PB de dados, de acordo com as estatísticas divulgadas pela empresa em 2020
  - Esses dados são, usualmente, armazenados em bancos de dados **distribuídos**

Introdução

- **O armazenamento e manipulação de grandes volumes de dados é essencial para muitos setores.**
  - O Facebook armazena mais de 300PB de dados, de acordo com as estatísticas divulgadas pela empresa em 2020
  - Esses dados são, usualmente, armazenados em bancos de dados **distribuídos**
- **Cada vez mais dispositivos estão conectados à internet**
  - Como encontrar um dado dispositivo conectado à rede?

Um dado elemento x é membro do conjunto S?
(teste de membresia)

Introdução

- **Não é trivial obter uma solução determinística eficiente em espaço para responder a testes de membresia.**

Introdução

- **Não é trivial obter uma solução determinística eficiente em espaço para responder a testes de membresia.**
  - Dicionários guardam mais informações do que o necessário

Introdução

- **Não é trivial obter uma solução determinística eficiente em espaço para responder a testes de membresia.**
  - Dicionários guardam mais informações do que o necessário
- **Alternativa: soluções probabilísticas**
  - Algumas dessas estruturas podem permitir **resultados falsos positivos**

Introdução

Um dado elemento x é membro do conjunto S?

- **Provavelmente sim (com razão de falsos positivos ϵ)**
- **Não**

Sumário

Bloom filter

O *Bloom filter* é uma estrutura de dados aleatorizada concebida por Bloom que tem o objetivo de responder a testes de membresia com **eficiência no consumo de espaço**.

Princípio do *Bloom filter*

Onde quer que seja usada uma lista ou conjunto e o espaço seja valioso, considere usar um Bloom filter se o efeito de falsos positivos puder ser mitigado. (Broder and Mitzenmacher, 2003, tradução nossa)

Bloom filter

O *Bloom filter* é uma estrutura de dados aleatorizada concebida por Bloom que tem o objetivo de responder a testes de membresia com **eficiência no consumo de espaço**.

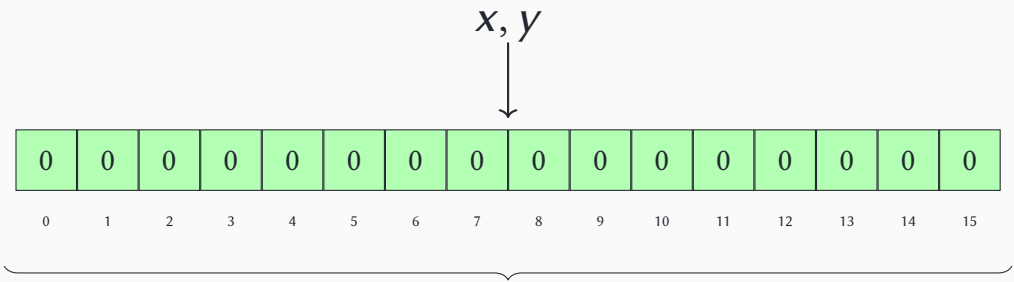
Princípio do *Bloom filter*

Onde quer que seja usada uma lista ou conjunto e o espaço seja valioso, considere usar um Bloom filter se o efeito de falsos positivos puder ser mitigado. (Broder and Mitzenmacher, 2003, tradução nossa)

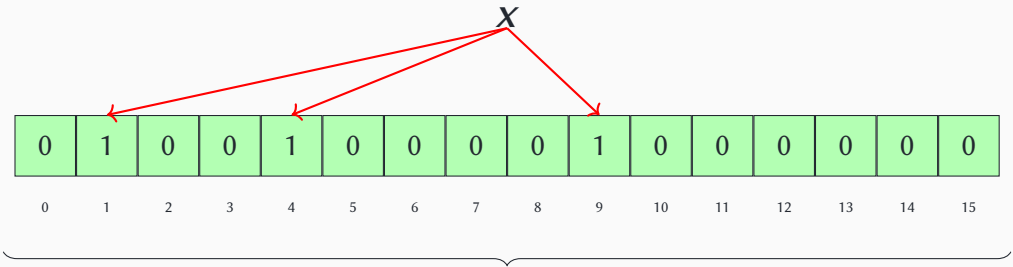
Operações

- **Inserção;**
- **Consulta (testes de membresia).**

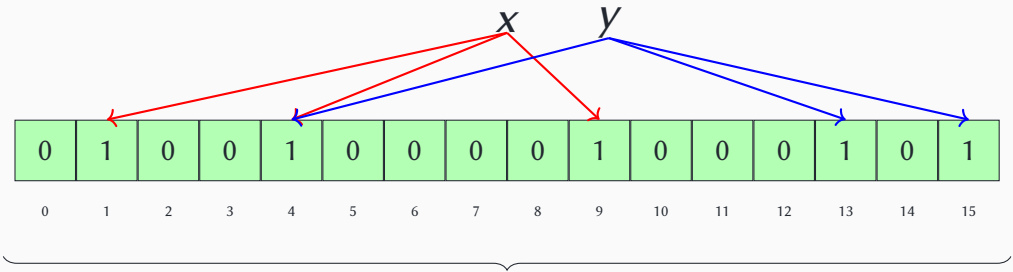
Bloom filters - Inserção



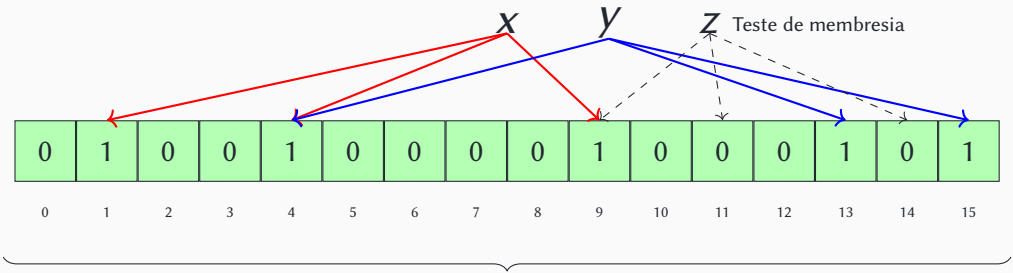
Bloom filters - Inserção



Bloom filters - Inserção



Bloom filters - Inserção



Sumário

Cuckoo filter

O *cuckoo filter*, descrito por Fan *et al.*, é uma alternativa ao *Bloom filter* para os cenários em que a **remoção de elementos** da estrutura é necessária.

- **Utiliza o *partial-key cuckoo hashing***
- **Cada elemento é mapeado para duas posições do vetor**

Cuckoo filter

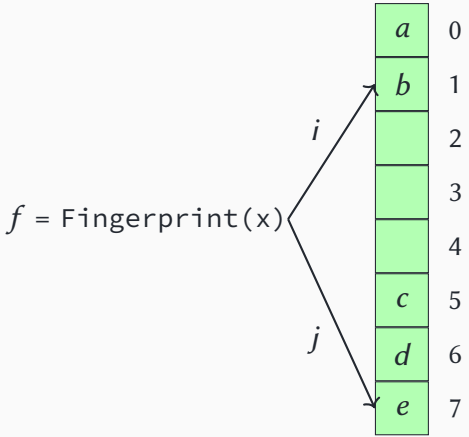
O *cuckoo filter*, descrito por Fan *et al.*, é uma alternativa ao *Bloom filter* para os cenários em que a **remoção de elementos** da estrutura é necessária.

- **Utiliza o *partial-key cuckoo hashing***
- **Cada elemento é mapeado para duas posições do vetor**

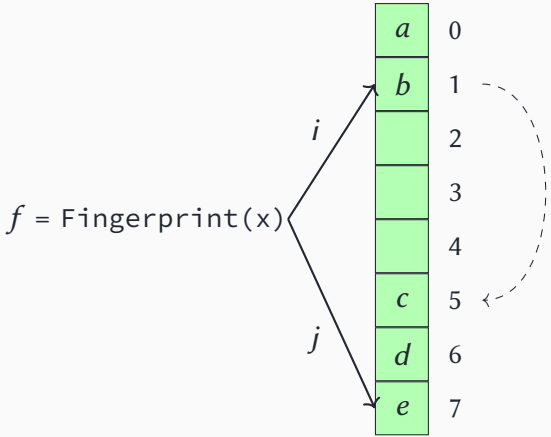
Operações

- **Inserção;**
- **Consulta (testes de membresia);**
- **Remoção de elementos.**

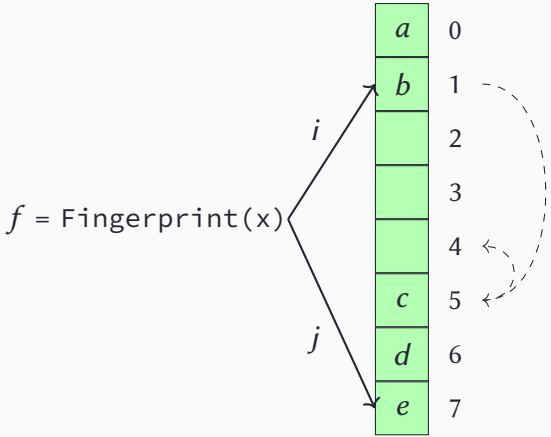
Cuckoo filters - Inserção



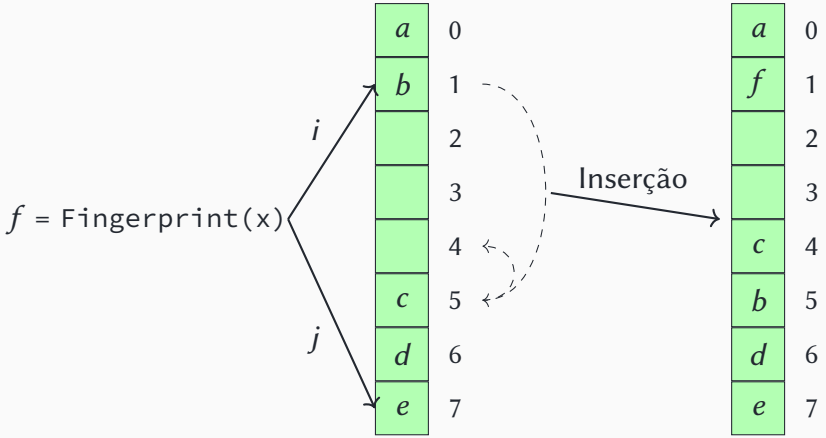
Cuckoo filters - Inserção



Cuckoo filters - Inserção



Cuckoo filters - Inserção



Partial-key cuckoo hashing

O *partial-key cuckoo hashing* é proposto por Fan et al. (2014) como uma forma de obter um alto nível ocupação da tabela, o que evita falhas na execução de inserções.

Partial-key cuckoo hashing

O *partial-key cuckoo hashing* é proposto por Fan et al. (2014) como uma forma de obter um alto nível ocupação da tabela, o que evita falhas na execução de inserções.

Seja f = `fingerprint`(x), então escolhem-se as seguintes funções de *hashing*:

$$h_1(x) = h(x) = i,$$
$$h_2(x) = h_1(x) \oplus h(f) = j.$$

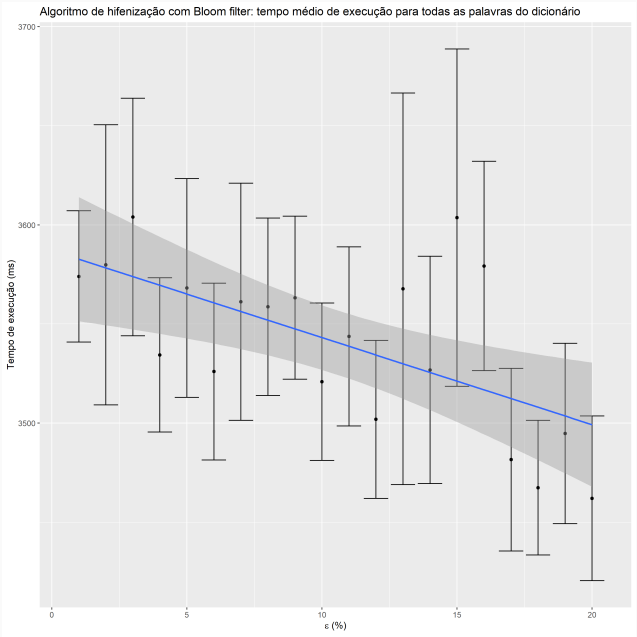
Sumário

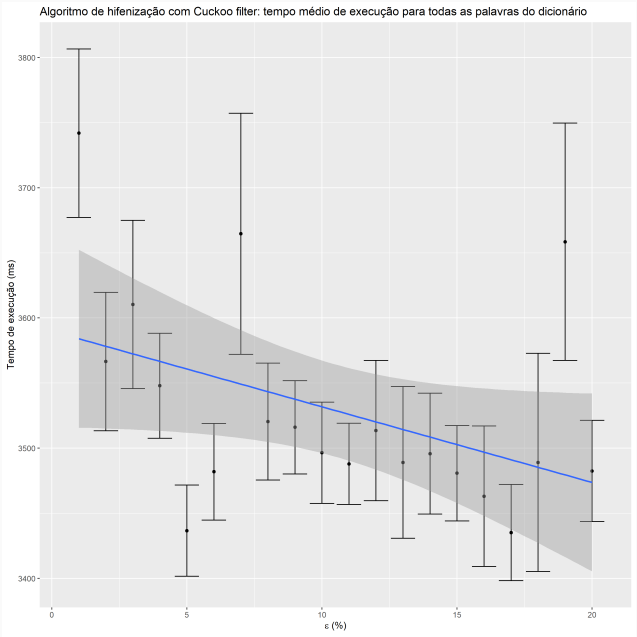
Metodologia

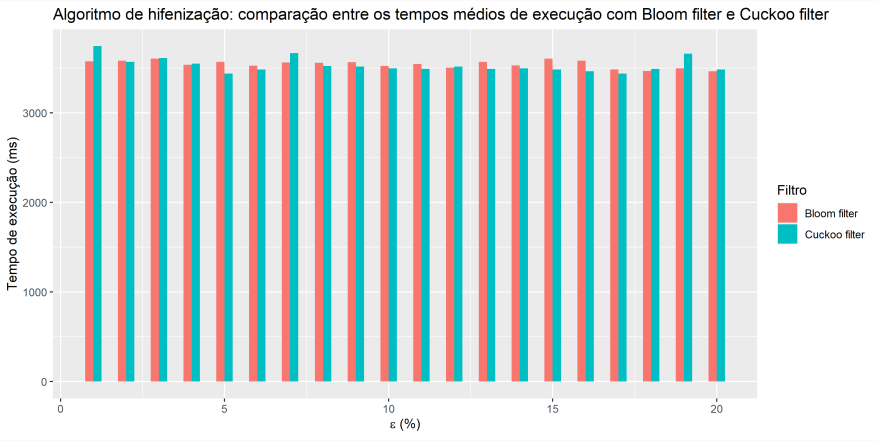
- **Pesquisa experimental e quantitativa**
- **Utiliza-se uma implementação de hifenizador de palavras de código aberto (https://github.com/bramstein/hypher)**
- **A implementação é modificada para utilizar um *Bloom filter* ou *cuckoo filter* (https://github.com/Callidon/bloom-filters)**

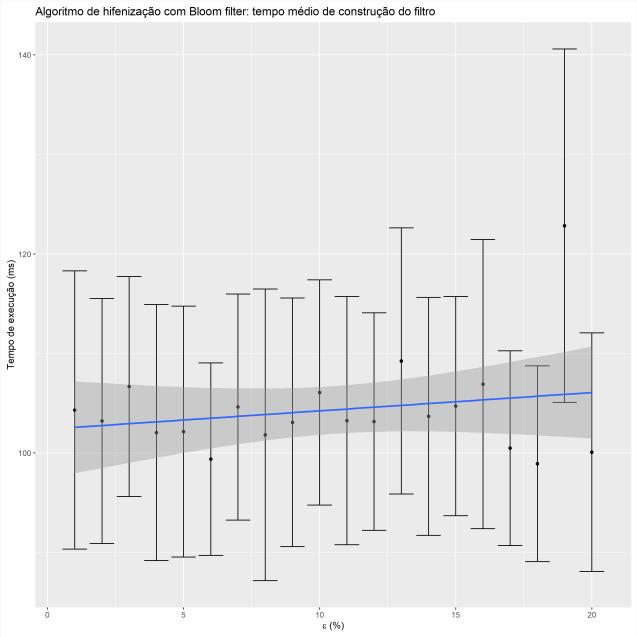
- **Objetivos**
  - Estudar o efeito da variação de ϵ
  - Comparar o desempenho de consultas e tempo de construção do algoritmo com cada um dos filtros

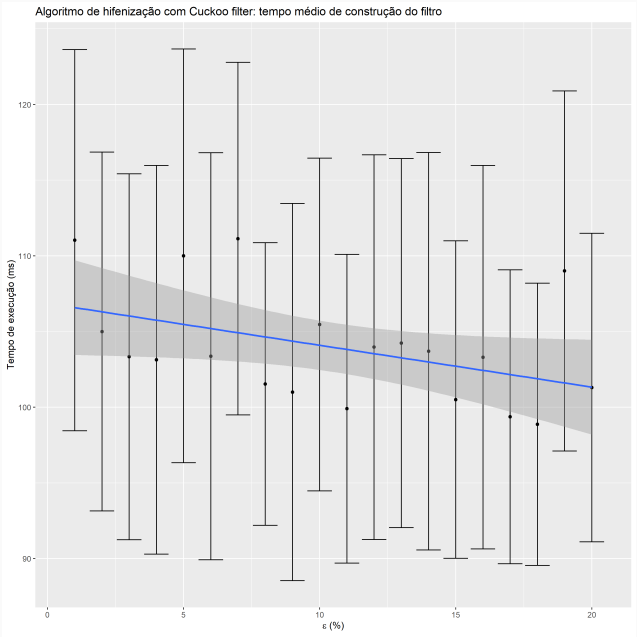
Sumário

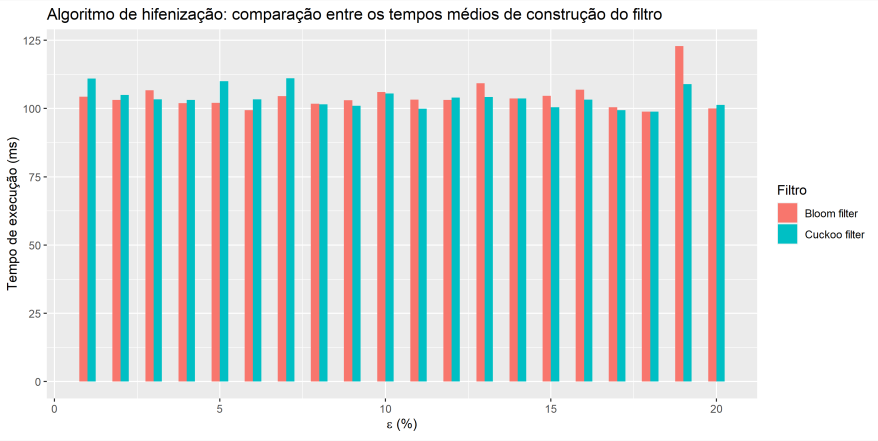

Algoritmo de hifenização com Bloom filter: tempo médio de execução para todas as palavras do dicionário
Tempo de execução (ms)
3700
3600
3500
0
5
10
15
20
ε (%)


Algoritmo de hifenização com Cuckoo filter: tempo médio de execução para todas as palavras do dicionário
Tempo de execução (ms)
3800
3700
3600
3500
3400
0
5
10
15
20
ε (%)


Algoritmo de hifenização: comparação entre os tempos médios de execução com Bloom filter e Cuckoo filter
Tempo de execução (ms)
0
1000
2000
3000
0
5
10
15
20
ε (%)
Filtro
Bloom filter
Cuckoo filter


Algoritmo de hifenização com Bloom filter: tempo médio de construção do filtro
Tempo de execução (ms)
140
120
100
0
5
10
15
20
ε (%)


Algoritmo de hifenização com Cuckoo filter: tempo médio de construção do filtro
Tempo de execução (ms)
120
110
100
90
0
5
10
15
20
ε (%)


Algoritmo de hifenização: comparação entre os tempos médios de construção do filtro
Tempo de execução (ms)
125
100
75
50
25
0
0
5
10
15
20
ε (%)
Filtro
Bloom filter
Cuckoo filter

Sumário

Conclusão

- ***Cuckoo filters*** **podem substituir** ***Bloom filters*** **em algoritmos hifenizadores de palavras, de modo que o consumo de tempo se mantenha similar, mas permita remoções de elementos;**

Conclusão

- ***Cuckoo filters*** **podem substituir** ***Bloom filters*** **em algoritmos hifenizadores de palavras, de modo que o consumo de tempo se mantenha similar, mas permita remoções de elementos;**
- **O tempo de construção dessas estruturas é, também, semelhante;**

Conclusão

- ***Cuckoo filters*** **podem substituir** ***Bloom filters*** **em algoritmos hifenizadores de palavras, de modo que o consumo de tempo se mantenha similar, mas permita remoções de elementos;**
- **O tempo de construção dessas estruturas é, também, semelhante;**
- **Há uma tendência de menor consumo de tempo para valores de ϵ maiores.**

Sumário

References i

- Bloom, Burton H. (July 1970). “Space/Time Trade-Offs in Hash Coding with Allowable Errors”. *Commun. ACM* 13.7, pp. 422–426. ISSN: 0001-0782. DOI: 10.1145/362686.362692. URL: https://doi.org/10.1145/362686.362692.
- Broder, Andrei and Michael Mitzenmacher (Nov. 2003). “Survey: Network Applications of Bloom Filters: A Survey.”. *Internet Mathematics* 1. DOI: 10.1080/15427951.2004.10129096.
- Fan, Bin et al. (2014). “Cuckoo Filter: Practically Better Than Bloom”. In: *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*. CoNEXT ’14. Sydney, Australia: Association for Computing Machinery, pp. 75–88. ISBN: 9781450332798. DOI: 10.1145/2674005.2674994. URL: https://doi.org/10.1145/2674005.2674994.

References ii

- Mosharraf, Sharafat Ibn Mollah and Muhammad Abdullah Adnan (Jan. 2022). "Improving lookup and query execution performance in distributed Big Data systems using Cuckoo Filter". *Journal of Big Data* 9.1, p. 12. ISSN: 2196-1115. DOI: 10.1186/s40537-022-00563-w. URL: https://doi.org/10.1186/s40537-022-00563-w.
- Reynolds, Patrick and Amin Vahdat (2003). "Efficient Peer-to-Peer Keyword Searching". In: *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*. Middleware '03. Rio de Janeiro, Brazil: Springer-Verlag, pp. 21–40. ISBN: 3540403175.
- szabowexler, Elias Szabo-Wexler (2014). "Approximate Membership of Sets : A Survey". In.
- Tarkoma, Sasu et al. (2012). "Theory and Practice of Bloom Filters for Distributed Systems". *IEEE Communications Surveys & Tutorials* 14.1, pp. 131–155. DOI: 10.1109/SURV.2011.031611.00024.

Estudo do efeito de variações de *Bloom filters*

https://linux.ime.usp.br/~mbsilva/mac0499/