Implementación de Bloom Filters

Ilse Córdova Sánchez 181901

Estructuras de datos avanzadas

Instituto Tecnológico Autónomo de México

12 de diciembre 2019

Los Bloom Filter son una estructura de datos probabilística creada por Burton Bloom que fue diseñada con el propósito de tener un manejo eficiente de espacio. Uno de los principales usos de esta estructura es el verificar si un elemento es miembro de un conjunto específico de datos.

Un Bloom Filter está compuesto por un arreglo de bits (m), que contiene diferentes cantidades de elementos (n). Los elementos son comprimidos por funciones de Hash, también llamadas funciones digest, para posteriormente usar las posiciones obtenidas para marcar el valor de un bit como verdadero o falso. Cabe mencionar que se puede tener k funciones de Hash. El valor de k puede ser determinado por el usuario o calculado mediante una fórmula para obtener el número óptimo de funciones de Hash por utilizar en un Bloom Filter.

Otro aspecto importante que debe ser tomado en consideración al tratar la estructura de datos Bloom Filter es que existe un porcentaje de falsos positivos (p) que pueden indicar que un dato se encuentra en el conjunto sin que efectivamente se encuentre contenido en este.

Los Bloom Filter tienen una amplia gama de usos, entre ellos se encuentran el detectar sitios web maliciosos, mejorar el rendimiento de cachés, realizar un seguimiento de artículos leídos e inclusive agilizar diferentes procesos al utilizar Bitcoins.

I. Calculando m

Mediante un argumento combinatorio, se puede mostrar que la probabilidad de obtener un falso positivo (p) puede aproximarse utilizando la siguiente fórmula:

$$p = \left(1 - e^{-\frac{k\,n}{m}}\right)^k$$

k = número de funciones de hash
n = tamaño del input
m = tamaño del Bloom Filter en bits

Al programar la fórmula para obtener p, se obtiene la siguiente línea de código:

```
p = pow(1 - exp(-k / (m / n)), k)
```

Para obtener la fórmula que nos permita calcular el tamaño óptimo de un Bloom Filter, es necesario despejar la m, de manera que:

$$m = \frac{n \log\left(\frac{1}{p}\right)}{\log^2(2)}$$

Al programar la fórmula para obtener p, se obtiene la siguiente línea de código:

```
m = ceil((n * log(p)) / log(1 / pow(2, log(2))));
```

II. Rol de k

De misma manera que con la fórmula para calcular el tamaño óptimo del Bloom Filter en bits (m), la fórmula para obtener el valor óptimo de k puede ser obtenido al despejarlo de la primera fórmula que fue enunciada. Al despejar el valor, se obtiene lo siguiente:

$$k = \frac{m \log(2)}{n}$$

k = número de funciones de hash
n = tamaño del input
m = tamaño del Bloom Filter en bits

Al programar la fórmula para obtener p, se obtiene la siguiente línea de código:

```
k = round((m / n) * log(2));
```

Además, cabe mencionar que en esta fórmula, el número de funciones de hash utilizadas crece linealmente con el número de bits por elemento.

III. Funciones de hash

Para la implementación del Bloom Filter, se utilizó el algoritmo MD5. Dicho algoritmo de hash actúa como una función criptográfica unidireccional que acepta un mensaje de cualquier longitud como entrada y devuelve como salida un valor digest. Originalmente, el algoritmo MD5 fue diseñado por Ronal Rivest para utilizarse como algoritmo criptográfico que autentificara diversas firmas digitales en la red; sin embargo, actualmente ya no se considera confiable en prácticas criptográficas.

La codificación del algoritmo MD5 es de 128 bits que típicamente se representa con 32 símbolos hexadecimales. Para obtener el hash de un elemento, es

necesario que este se convierta a String para que el algoritmo criptográfico pueda procesarlo en los siguientes pasos.

Adicionar bits

Se extiende el String dado a la función al añadir un bit “1” y bits “0” hasta que la longitud de esta sea congruente con 448, módulo 12. Es necesario resaltar que dicha extensión siempre es realizada a pesar de que la longitud de la cadena ya sea congruente.

Longitud del mensaje

Se concatena a la cadena un entero de 64 bits que represente la longitud del mensaje. Al realizar esto, la cadena cuenta con una longitud que es múltiplo de 512 bits.

Inicializar el búfer

Se utiliza un búfer de cuatro palabras para calcular el resumen del mensaje, dichos registros se inicializan con valores hexadecimales en los cuales los bytes de menos peso se utilizan primero.

Procesar el mensaje

Se definen cuatro funciones auxiliares que toman como entrada tres palabras de 32 bits y regresan una palabra de 32 bits. En dichas funciones se utilizan operadores como XOR, AND, OR y NOT. En este paso se utiliza una tabla de 64 elementos construida por la función Seno.

Salida

Se obtiene el resumen del mensaje de salida producido por las cuatro funciones producidas por el búfer.

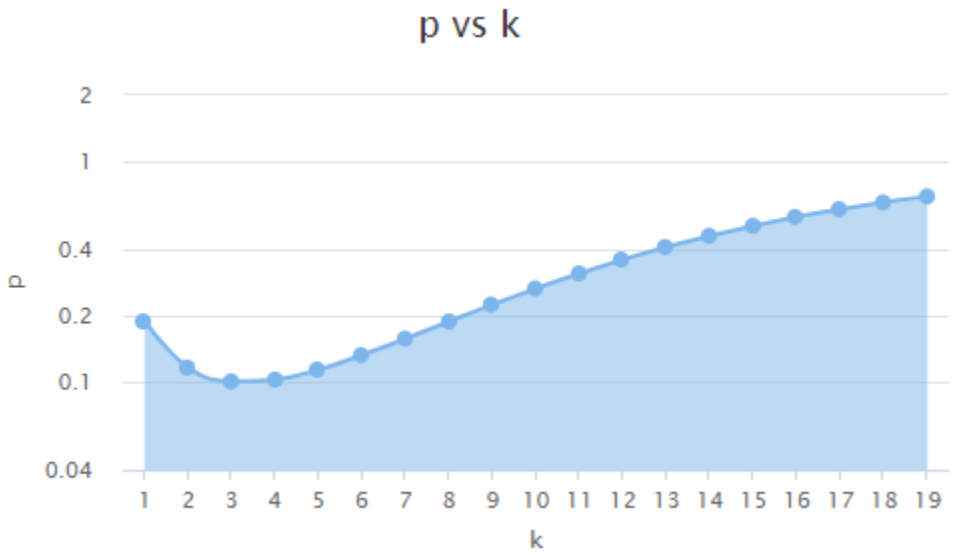
IV. Relación entre k y p

El utilizar varias funciones digest se hace con el propósito de evitar colisiones y disminuir el porcentaje de falsos positivos. A continuación se presentan diferentes experimentos realizados con diversos datos para poder apreciar la relación entre el número de funciones de hash y el porcentaje de falsos positivos.

Primera gráfica

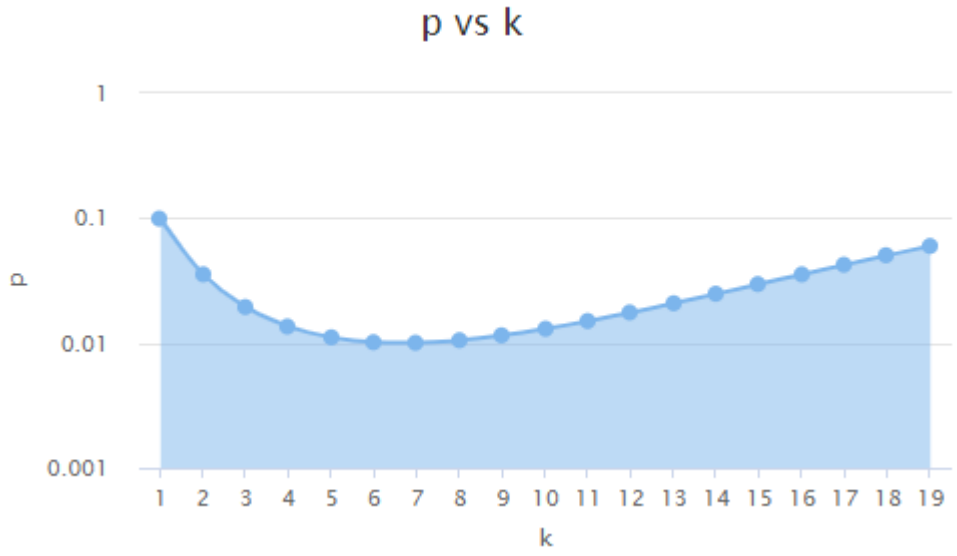
n = 1000 p = 0.1006919 (1 en 10)

m = 4793 (599B) k = 3



Segunda gráfica

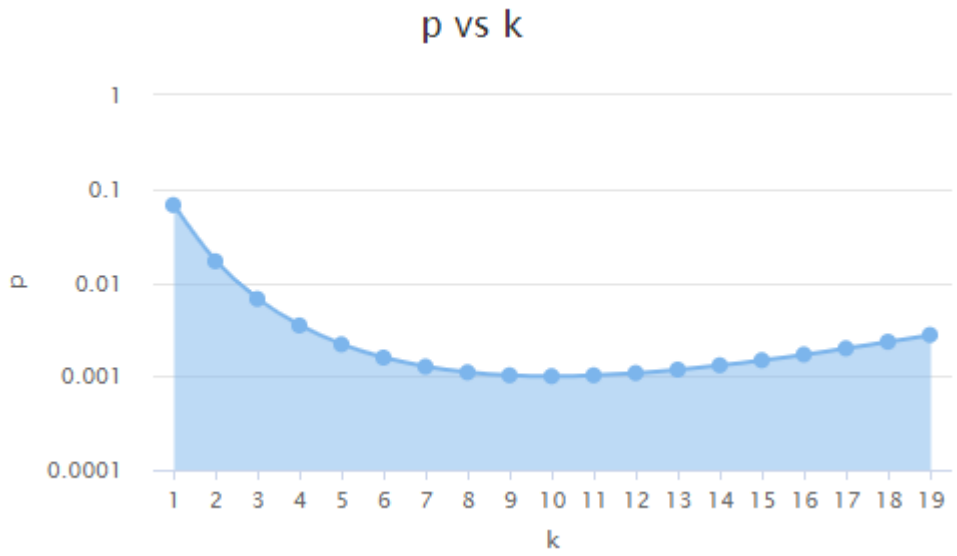
n = 1000 p = 0.010034532 (1 en 100)
m = 9586 (1.17KiB) k = 7



Tercera gráfica

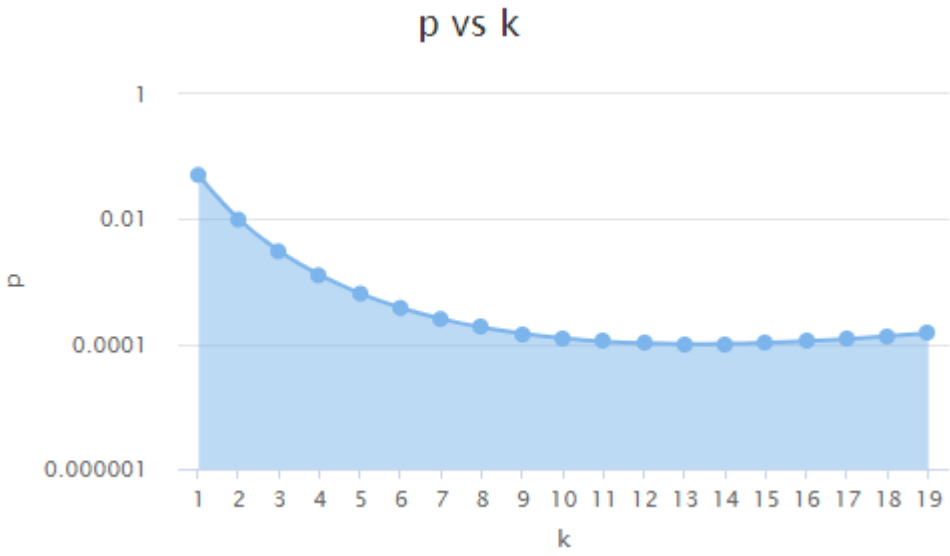
n = 1000 p = 0.000999826 (1 en 1000)

m = 14378 (1.75KiB) k = 10



Cuarta gráfica

n = 1000 p = 0.000100093 (1 en 9991)
m = 19171 (2.34KiB) k = 13



Para realizar las gráficas, se definió n como 1000 y se utilizaron las fórmulas anteriormente mencionadas para calcular el valor óptimo de m y k. Al observar las gráficas se puede apreciar que existe un área “óptima” en la que el valor de p se acerca al número ingresado por el usuario. Cabe notar que el agregar muchas funciones de hash al Bloom Filter no garantiza que se evitará obtener falsos positivos

completamente. En algunos casos, el agregar varias funciones digest puede resultar contraproducente ya que el porcentaje de falsos positivos incrementa.

A continuación se presenta una tabla que contiene el porcentaje de falsos positivos que se puede obtener de diferentes combinaciones de m, n y k:

m/n	*k*	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5	*k*=6	*k*=7	*k*=8
2	1.39	0.393	0.400						
3	2.08	0.283	0.237	0.253					
4	2.77	0.221	0.155	0.147	0.160				
5	3.46	0.181	0.109	0.092	0.092	0.101			
6	4.16	0.154	0.0804	0.0609	0.0561	0.0578	0.0638		
7	4.85	0.133	0.0618	0.0423	0.0359	0.0347	0.0364		
8	5.55	0.118	0.0489	0.0306	0.024	0.0217	0.0216	0.0229	
9	6.24	0.105	0.0397	0.0228	0.0166	0.0141	0.0133	0.0135	0.0145
10	6.93	0.0952	0.0329	0.0174	0.0118	0.00943	0.00844	0.00819	0.00846
11	7.62	0.0869	0.0276	0.0136	0.00864	0.0065	0.00552	0.00513	0.00509
12	8.32	0.08	0.0236	0.0108	0.00646	0.00459	0.00371	0.00329	0.00314
13	9.01	0.074	0.0203	0.00875	0.00492	0.00332	0.00255	0.00217	0.00199
14	9.7	0.0689	0.0177	0.00718	0.00381	0.00244	0.00179	0.00146	0.00129
15	10.4	0.0645	0.0156	0.00596	0.003	0.00183	0.00128	0.001	0.000852
16	11.1	0.0606	0.0138	0.005	0.00239	0.00139	0.000935	0.000702	0.000574
17	11.8	0.0571	0.0123	0.00423	0.00193	0.00107	0.000692	0.000499	0.000394
18	12.5	0.054	0.0111	0.00362	0.00158	0.000839	0.000519	0.00036	0.000275
19	13.2	0.0513	0.00998	0.00312	0.0013	0.000663	0.000394	0.000264	0.000194
20	13.9	0.0488	0.00906	0.0027	0.00108	0.00053	0.000303	0.000196	0.00014
21	14.6	0.0465	0.00825	0.00236	0.000905	0.000427	0.000236	0.000147	0.000101
22	15.2	0.0444	0.00755	0.00207	0.000764	0.000347	0.000185	0.000112	7.46e-05
23	15.9	0.0425	0.00694	0.00183	0.000649	0.000285	0.000147	8.56e-05	5.55e-05
24	16.6	0.0408	0.00639	0.00162	0.000555	0.000235	0.000117	6.63e-05	4.17e-05
25	17.3	0.0392	0.00591	0.00145	0.000478	0.000196	9.44e-05	5.18e-05	3.16e-05

V. Lista de referencias

(1998). Cao P. Bloom Filters – the math. Wisconsin University. Consultado el 11 de diciembre de 2019. Disponible en: http://pages.cs.wisc.edu/~cao/papers/summary-cache/node8.html

(2010). Cortesi A. 3 Rules of thumb for Bloom Filters. Consultado el 11 de diciembre de 2019. Disponible en: https://corte.si/posts/code/bloom-filter-rules-of-thumb/index.html

(2018). Hurt T. Bloom Filter Calculator. Consultado el 11 de diciembre de 2019. Disponible en: https://hur.st/bloomfilter/?n=1000&p=0.0001&m=&k=7%20

(2017). Warren C. Bloom Filters An Overview. Consultado el 11 de diciembre de 2019. Disponible en: https://github.com/Claudenw/BloomFilter/wiki/Bloom-Filters----An-overview