Lecture 24:
Bloom Filters

Wednesday, June 2, 2010
Topics for the Final

- SQL
- Conceptual Design (BCNF)
- Transactions
- Indexes
- Query execution and optimization
- Cardinality Estimation
- Parallel Databases
Lecture on Bloom Filters

Not described in the textbook!

Lecture based in part on:

Example (from Pig Latin lecture)

Users(name, age)
Pages(user, url)

SELECT Pages.url, count(*) as cnt
FROM Users, Pages
WHERE Users.age in [18..25]
GROUP BY Pages.url
ORDER DESC cnt
Example

Problem: many pages, but only a few visited by users with age 18..25

• Pig’s solution:
  – MAP phase send *all* pages and *all* users to the reducers

• How can we reduce communication cost?
Hash Maps

• Let \( S = \{x_1, x_2, \ldots, x_n\} \) be a set of elements
• Let \( m > n \)

• Hash function \( h : S \rightarrow \{1, 2, \ldots, m\} \)

\[
S = \{x_1, x_2, \ldots, x_n\}
\]

\[
H = \begin{array}{cccccccc}
1 & 2 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
\end{array}
\]
Hash Map = Dictionary

The hash map acts like a dictionary

• Insert(x, H) = set bit h(x) to 1
  – Collisions are possible

• Member(y, H) = check if bit h(y) is 1
  – False positives are possible

• Delete(y, H) = not supported!
  – Extensions possible, see later
Example (cont’d)

- **Map-Reduce task 1**
  - Map task: compute a hash map $H$ for **users** with age in $[18..25]$. Several Map tasks in parallel.
  - Reduce task: combine all hash maps using OR. One single reducer suffices.

- **Map-Reduce task 2**
  - Map tasks 1: map each **user** to the appropriate region
  - Map tasks 2: map each **page** in $H$ to appropriate region
  - Reduce task: do the join

Why don’t we lose any pages?
Analysis

• Let $S = \{x_1, x_2, \ldots, x_n\}$

• Let $j = \text{a specific bit in } H (1 \leq j \leq m)$

• What is the probability that $j$ remains 0 after inserting all $n$ elements from $S$ into $H$?

• Will compute in two steps
Analysis

• Recall $|H| = m$
• Let’s insert only $x_i$ into $H$

• What is the probability that bit $j$ is 0?
Analysis

• Recall |H| = m
• Let’s insert only $x_i$ into H

• What is the probability that bit $j$ is 0?

• Answer: $p = 1 - 1/m$
Analysis

- Recall $|H| = m$, $S = \{x_1, x_2, \ldots, x_n\}$
- Let’s insert all elements from $S$ in $H$

- What is the probability that bit $j$ remains 0?
Analysis

- Recall $|H| = m$, $S = \{x_1, x_2, \ldots, x_n\}$
- Let’s insert all elements from $S$ in $H$

- What is the probability that bit $j$ remains 0?

- Answer: $p = (1 - 1/m)^n$
Probability of False Positives

- Take a random element $y$, and check $\text{member}(y, H)$
- What is the probability that it returns $\text{true}$?
Probability of False Positives

- Take a random element $y$, and check $\text{member}(y,H)$
- What is the probability that it returns $true$?

- Answer: it is the probability that bit $h(y)$ is 1, which is $f = 1 - (1 - 1/m)^n \approx 1 - e^{-n/m}$
Bloom Filters

• Introduced by Burton Bloom in 1970

• Improve the false positive ratio

• Idea: use $k$ independent hash functions
Bloom Filter = Dictionary

• Insert(x, H) = set bits $h_1(x), \ldots, h_k(x)$ to 1
  – Collisions are possible
• Member(y, H) = check if bits $h_1(y), \ldots, h_k(y)$ are 1
  – False positives are possible
• Delete(z, H) = not supported !
  – Extensions possible, see later
Example Bloom Filter $k=3$

$y_1 = \text{is not in } H \text{ (why ?)}$; $y_2 = \text{may be in } H \text{ (why ?)}$
Bloom Filter Principle

- Wherever a list or set is used, and space is at a premium, consider using a Bloom filter if the effect of false positives can be mitigated
Choosing $k$

Two competing forces:

- **If $k = \text{large}$**
  - Test more bits for $\text{member}(y,H) \Rightarrow$ lower false positive rate
  - More bits in $H$ are 1 $\Rightarrow$ higher false positive rate

- **If $k = \text{small}$**
  - More bits in $H$ are 0 $\Rightarrow$ lower positive rate
  - Test fewer bits for $\text{member}(y,H) \Rightarrow$ higher rate
Analysis

• Recall $|H| = m$, hash functions $= k$
• Let’s insert only $x_i$ into $H$

• What is the probability that bit $j$ is 0?
Analysis

• Recall $|H| = m$, #hash functions = $k$
• Let’s insert only $x_i$ into H

• What is the probability that bit $j$ is 0 ?

• Answer: $p = (1 - 1/m)^k$
Analysis

• Recall $|H| = m$, $S = \{x_1, x_2, \ldots, x_n\}$
• Let’s insert all elements from $S$ in $H$

• What is the probability that bit $j$ remains 0?
Analysis

• Recall $|H| = m$, $S = \{x_1, x_2, \ldots, x_n\}$
• Let’s insert all elements from $S$ in $H$

• What is the probability that bit $j$ remains 0?

• Answer: $p = (1 - 1/m)^{kn} \approx e^{-kn/m}$
Probability of False Positives

• Take a random element $y$, and check $\text{member}(y,H)$
• What is the probability that it returns $true$?
Probability of False Positives

• Take a random element y, and check member(y,H)

• What is the probability that it returns true?

• Answer: it is the probability that all k bits $h_1(y), \ldots, h_k(y)$ are 1, which is:

\[ f = (1-p)^k \approx (1 - e^{-kn/m})^k \]
Optimizing $k$

- For fixed $m$, $n$, choose $k$ to minimize the false positive rate $f$
- Denote $g = \ln(f) = k \ln(1 - e^{-kn/m})$
- Goal: find $k$ to minimize $g$

\[
\frac{\partial g}{\partial k} = \ln \left(1 - e^{-\frac{kn}{m}}\right) + \frac{kn}{m} \frac{e^{-\frac{kn}{m}}}{1 - e^{-\frac{kn}{m}}}
\]

\[k = \ln 2 \times \frac{m}{n}\]
Bloom Filter Summary

Given \( n = |S|, \ m = |H|, \)
choose \( k = \ln 2 \times m / n \) hash functions

- Probability that some bit \( j \) is 1
  \[ p \approx e^{-kn/m} = \frac{1}{2} \]

- Expected distribution
  \[ \text{m/2 bits 1, m/2 bits 0} \]

- Probability of false positive
  \[ f = (1-p)^k \approx \left(\frac{1}{2}\right)^k = \left(\frac{1}{2}\right)^{\frac{(\ln 2)m}{n}} \approx \left(0.6185\right)^{\frac{m}{n}} \]
Bloom Filter Summary

• In practice one sets \( m = cn \), for some constant \( c \)
  – Thus, we use \( c \) bits for each element in \( S \)
  – Then \( f \approx (0.6185)^c = \text{constant} \)

• Example: \( m = 8n \), then
  \[
  k = 8(\ln 2) = 5.545 \quad \text{(use 6 hash functions)}
  \]
  \[
  f \approx (0.6185)^{m/n} = (0.6185)^8 \approx 0.02 \quad (2\% \text{ false positives})
  \]
  Compare to a hash table: \( f \approx 1 - e^{-n/m} = 1 - e^{-1/8} \approx 0.11 \)
Set Operations

Intersection and Union of Sets:
- Set S $\Rightarrow$ Bloom filter H
- Set S’ $\Rightarrow$ Bloom filter H’

- How do we computed the Bloom filter for the intersection of S and S’?
Set Operations

Intersection and Union:

- Set S $\Rightarrow$ Bloom filter H
- Set S’ $\Rightarrow$ Bloom filter H’

- How do we computed the Bloom filter for the intersection of S and S’?
- Answer: bit-wise AND: $H \wedge H’$
Counting Bloom Filter

Goal: support delete(z, H)
Keep a counter for each bit j
• Insertion $\rightarrow$ increment counter
• Deletion $\rightarrow$ decrement counter
• Overflow $\rightarrow$ keep bit 1 forever
Using 4 bits per counter:

Probability of overflow $\leq 1.37 \times 10^{-15} \times m$
Application: Dictionaries

Bloom originally introduced this for hyphenation

- 90% of English words can be hyphenated using simple rules
- 10% require table lookup
- Use “bloom filter” to check if lookup needed
Application: Distributed Caching

• Web proxies maintain a cache of (URL, page) pairs
• If a URL is not present in the cache, they would like to check the cache of other proxies in the network
• Transferring all URLs is expensive!
• Instead: compute Bloom filter, exchange periodically