Cardinality Estimation: An Experimental Survey

Hazar Harmouch and Felix Naumann

Information System Group
Hasso Plattner Institut
University of Potsdam
Germany
Finding Number of Distinct Values Problem
A Polyonymous Problem

How many distinct voice actors are there in our series?

- **Statistics**: number of species in a population.
- **DB**: „COUNT DISTINCT“
- **Streaming**: The zeroth-frequency moment of a multiset [Alon96]
Why cardinality is an important statistic?

How Many Distinct ...

... queries did I get?

... pairs (sourceIP, destinationIP) have I seen?

... distinct messages have I seen?

... values have I seen for this attribute x?

... connections have been established from same source?

... visitors to this website in order to advertise in it?
Is exact cardinality sufficient?
Big Data: Exact Counting is Not Easy!

Exact cardinality of multiset determined with storage proportional to dataset size
Is exact cardinality sufficient?
Big Data-Scale! Estimate!

- Scale-up the computation
  - Expensive (hardware, equipment, energy).
  - Not always fast.
- Scale-down the data
  - Create **synopsis**: data structure maintained by the estimation algorithm in main memory.
    - **Temporary**: static scenarios.
    - **Compact representation**: streaming applications
  - Need to fit the problem.
Cardinality Estimation Approaches (1-6)

**Exact cardinality:** Sorting

- Sorting eliminates duplicates.

**Problem:**
- Expensive operation.
- Synopsis size is at least as large as the dataset.
- Impractical for current big datasets.
Cardinality Estimation Approaches (2-6)

**Exact Cardinality: Bitmap**

- **Synopsis**: is a bitmap of size equals to universe size and initialized to 0s.
  - Scan dataset once and set the bit $i$ to 1 whenever an item with the $i$–th value of the universe is observed.
  - Cardinality = Number of 1s.

- **Problem**: The synopsis size is a function of the universe size $N$, which is potentially much larger than the size of the dataset itself.

![Bitmap Example]

Still used in another approached.

Cardinality Estimation
Hazar Harmouch
Cardinality Estimation Approaches (3-6)

**Exact Cardinality: Hashing**

- Hashing eliminates duplicates without sorting, scale-down synopsis size and requires one pass.
- Simple application of hashing can be worse than sorting in terms of memory consumption.

All exact approaches are expensive in both size and runtime.
Cardinality Estimation Approaches (4-6)

**Estimation**: Bitmap of hash values

- Scales down the synopsis size by don’t store the hash values.
- **Synopsis**: a bitmap keeps track of the hashed values.
  - The hash function maps each item to a bit in the bitmap.
  - Like Bloom filters

![Bitmap Diagram]

A prior knowledge of the maximum cardinality is required to choose a good bitmap size.
Cardinality Estimation approaches (5-6)

Estimation: Sampling

- Reduces the synopsis size

- Several negative results.
  - For every estimate based on a small-sample, there is a dataset where the ratio error can be made arbitrarily large [Charikar00].
  - Almost all the dataset needs to be sampled to bound the estimation error within a small constant [Haas95, Haas98].
Cardinality Estimation approaches (6-6)

Estimation: Observations in hash values

- **Bit pattern observables**
  - depend on the occurrence of particular bit patterns at the binary string representation.

- **Order statistic observables**
  - consider the hash values as real numbers.
  - The order statistic of rank \( k \) is the \( k \)-th smallest value in the dataset.

Hash values can be seen as:

- **Bit strings**
- **Range of real numbers**

Smallest value seen \( \approx \frac{1}{F_0} \)
Cardinality Estimation
Classification of 12 Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Observables</th>
<th>Intuition</th>
<th>Core method</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM</td>
<td>Bit-pattern</td>
<td>Logarithmic hashing</td>
<td>Count trailing 1s</td>
</tr>
<tr>
<td>PCSA</td>
<td>Bit-pattern</td>
<td>Logarithmic hashing</td>
<td>Count trailing 1s</td>
</tr>
<tr>
<td>AMS</td>
<td>Bit-pattern</td>
<td>Logarithmic hashing</td>
<td>Count leading 0s</td>
</tr>
<tr>
<td>BJKST</td>
<td>Order statistics</td>
<td>Bucket-based</td>
<td>Count leading 0s</td>
</tr>
<tr>
<td>LogLog</td>
<td>Bit-pattern</td>
<td>Logarithmic hashing</td>
<td>Count leading 0s</td>
</tr>
<tr>
<td>SuperLogLog</td>
<td>Bit-pattern</td>
<td>Logarithmic hashing</td>
<td>Count leading 0s</td>
</tr>
<tr>
<td>HyperLogLog</td>
<td>Bit-pattern (order statistics)</td>
<td>Logarithmic hashing</td>
<td>Count leading 0s</td>
</tr>
<tr>
<td>HyperLogLog++</td>
<td>Bit-pattern</td>
<td>Logarithmic hashing</td>
<td>Count leading 0s</td>
</tr>
<tr>
<td>MinCount</td>
<td>Order statistics</td>
<td>Interval-based</td>
<td>k-th minimum value</td>
</tr>
<tr>
<td>AKMV</td>
<td>Order statistics</td>
<td>Interval-based</td>
<td>k-th minimum value</td>
</tr>
<tr>
<td>LC</td>
<td>No observable</td>
<td>Bucket-based</td>
<td>Linear synopses</td>
</tr>
<tr>
<td>BF</td>
<td>No observable</td>
<td>Bucket-based</td>
<td>Linear synopses</td>
</tr>
</tbody>
</table>

Hazar Harmouch
Counting trailing 1`s Algorithm (1-2)
Flajolet-Martin (FM) [Flajolet-Martin85]

As I said over the phone, I started working on your algorithm when Kuo Young Whang considered implementing it and wanted explanations/estimations. I found it simple, clear and amazingly powerful.

Without analysis (original algorithm)

After all the values have been processed, then
if \( M(MAP) = 000 \), then \( RESULT = L0(MAP) - 1 \)
if \( M(MAP) = 111 \), then \( RESULT = L0(MAP) + 1 \)
otherwise \( RESULT = L0(MAP) \).

For example,
if \( MAP = 0000000000000000000000000000111111 \)
then \( L0(MAP) = 8 \) and \( M(MAP) = 000 \); \( RESULT = 7 \)
if \( MAP = 0000000000000000000000000000111111 \)
then \( L0(MAP) = 8 \) and \( M(MAP) = 000 \); \( RESULT = 7 \)
if \( MAP = 0000000000000000000000000000111111 \)
then \( L0(MAP) = 8 \) and \( M(MAP) = 000 \); \( RESULT = 7 \)
if \( MAP = 0000000000000000000000000000111111 \)
then \( L0(MAP) = 8 \) and \( M(MAP) = 010 \); \( RESULT = 8 \)

With analysis (Philippe)

Philippe determines that
\[
\mathbb{E}[2^p] \approx \phi n
\]
where \( \phi \approx 0.77351 \ldots \) is defined by
\[
\phi = \frac{e^\gamma \sqrt{2}}{3} \prod_{p=1}^{\infty} \left( \frac{(4p+1)(4p+2)}{(4p)(4p+3)} \right)^{(-1)^{p+1}}
\]
such that we can apply a simple correction and have unbiased estimator,
\[
Z := \frac{1}{\phi} 2^p \quad \mathbb{E}[Z] = n
\]

Cardinality Estimation
Hazar Harmouch

Photos: https://speakerdeck.com/timonk/philippe-flajolets-contribution-to-streaming-algorithms
Counting trailing 1`s Algorithm (2-2) Flajolet Martin (FM) [Flajolet-Martin85]

- Bitmap[$\rho$(hash(x))]=1
- $\rho$(y)=position of the LSB=1 in y.

Z: Number of trailing 1s in the bitmap
L: length of the hash bit string (e.g. 32 bit)

Estimate $F_0 = \lfloor 2^Z / 0.77351 \rfloor$

Intuition:
- Seeing $\rho = k$ means there are at least $2^{k+1}$ different bit strings.
- Find the largest $\rho$ and estimate the cardinality by $2^\rho$.
Why comparative experiments is needed?

- Some applications require a very accurate estimation. However, others accept a less accurate estimation.
  - The number of distinct visitors of a website = money.
  - The number of distinct connections \( \approx \) Denial of service.

Why re-evaluation is good?
- Is theoretical error analysis matches real-world?
- What is hidden in the Big-O notation in space bound?
- Different hash function assumptions
- Different error metric
Comparative experiments

Experimental setup

**Implementations** (Unified test environment):
- Implemented for Metanome
- MurmurHash 64-bit. (32-bit for AKMV and MinCount)
- All algorithms were configured to produce theoretical (standard/relative) errors of 1%.

**Datasets**: 90 synthetic datasets. The exact cardinalities made to be the powers of 10, starting with 10 up to $10^9$.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Attributes</th>
<th># Tuples</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCVoter</td>
<td>25 (of 71)</td>
<td>7,560,886</td>
</tr>
<tr>
<td>Openaddresses-Europe</td>
<td>11</td>
<td>93,849,474</td>
</tr>
</tbody>
</table>
Guaranteed Error Estimator (GEE) [Charikar00] uses frequency of the values within the sampled data.

We used Reservoir sampling without replacement.

- 1% relative error requires sampling more than 90% of the dataset.
- Minimum heap size of at least 13 GByte and 35 GByte is needed to guarantee an estimation error below 1% on NCVoter and Openaddress-Europe, respectively.
- Runtime noticeably increases with the size of the dataset, but only slightly with the sampling rate.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Sampling rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20%</td>
</tr>
<tr>
<td>Synthetic</td>
<td>0.54</td>
</tr>
<tr>
<td>NCVoter</td>
<td>0.26</td>
</tr>
<tr>
<td>Openaddresses</td>
<td>0.28</td>
</tr>
</tbody>
</table>
Comparative experiments
Accuracy experiments - synthetic datasets

Cardinality Estimation
Hazar Harmouch
Comparative experiments
Accuracy experiments - synthetic datasets

Cardinality Estimation
Hazar Harmouch

![Graph showing comparative experiments for accuracy in synthetic datasets]

- PCSA
- FM
- AMS
- BJKST
- LogLog
- SuperLogLog
- HyperLogLog
- HyperLogLog++
- LC
- BF1
- BF2
- Mincount
- AKMV
Comparative experiments
Accuracy experiments-real-world datasets

Cardinality Estimation
Hazar Harmouch
Comparative experiments
Runtime behavior experiments - synthetic datasets

Main factors:
- Dataset size
- Nb. of hash functions
- Synopsis type

Cardinality Estimation
Hazar Harmouch
Comparative experiments
Runtime behavior experiments-real-world datasets

Cardinality Estimation
Hazar Harmouch

NCVoter
Openaddress-Europe
Comparative experiments
Memory Consumption experiments-synthetic datasets

Cardinality Estimation
Hazar Harmouch
Summary
Counting like Einstein

- Cardinality estimation is a widely studied problem
- Some preliminary solutions, such as sampling and hash tables, are valid only when one can scale up the available computational resources
- For a given accuracy, dataset size is obviously the main factor, affecting all the algorithms' runtime and memory consumption.
- FM: extremely high runtime
- BJKST and Bloom filter have a high memory consumption.
- HyperLogLog, AKMV, and LC are efficient over all cardinality ranges by all means.
How many distinct voice actors?

The Answer

https://en.wikipedia.org/wiki/List_of_The_Simpsons_cast_members
References


References


References


