Ceramist: Certifying Certainty and Uncertainty

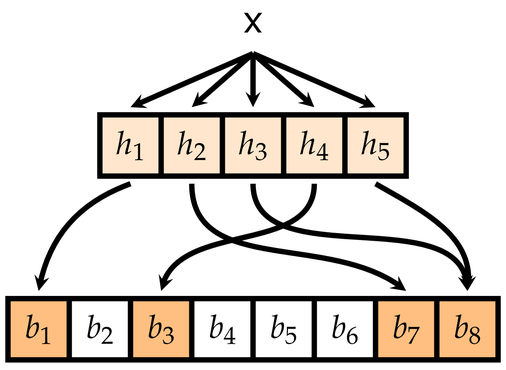
Kiran Gopinathan (Slack: @kiran gopinathan), Ilya Sergey
National University of Sinagpore

1 Bloomfilters & AMQs

What are they?

Approximate Membership Query structures (AMQs) are a general class of probabilistic data structures that provide compact encodings of sets of objects, trading **increased space efficiency** for a **weaker membership test**.

A Bloomfilter is an example of an AMQ, consisting of a bit vector and a sequence of hash functions.



To insert a value x**:** Apply the hashes to x, and raise the selected bits.

To test membership for x**:** Apply the hashes to x, and check that the corresponding bits are raised.

Properties of Bloomfilters

The operations on Bloomfilters work together to ensure two key properties that are common to all AMQs:

- **No False Negatives** - any item that has been previously inserted will always correctly be tested as being in the Bloomfilter.
- **False Positive Rate** - there is a small probability that a value not in the Bloomfilter may still test positve.

Practical applications

The properties of AMQs make them a particularly suitable component for providing caching-based optimizations.

Bloomfilters themselves are very widely used, and can be found in many modern software tools, collectively reaching over millions of users:

- Google Chrome [1]
- Apache Cassandra [2]
- Venti Network Storage System [3]

2 A history of errors...

Prior Theoretical Analysis

Reasoning about probability is hard, and it is easy to make mistakes by overlooking interdependencies.

This is the case with the analysis of Bloomfilters, which have had a history of incorrect proofs:

- **Bloom's original paper** [4] failed to account for subtle dependencies leading to an **incorrect false positive rate**.
- A later study [5] found the error, but their correction also used **incorrect definitions**, marring the analysis.
- Many subsequent papers, and **even textbooks** [6], still reference the incorrect bound.

3 Evening the "odds" in Coq!

Eliminating errors with proof assistants

Following Coq's established history of being used to correct invalid reasoning, we developed a Coq framework for reasoning about AMQs, modelling hash operations as **random oracles**, and using the **probability monad** to encode random outcomes.

We find that we can decompose the behaviours and proofs of Bloomfilters into **deterministic** and **probabilistic** subcomponents:

- **Deterministic** - setting and raising bits of a bit vector given the hash outputs.
- **Probabilistic** - hashing the input over a series of hash functions to get a list of indices.

Through this technique, we provide the first certified proof of the **true false positive rate** of a Bloomfilter.

$$\frac{1}{m^{k(l+1)}} \sum_{i=1}^{m} i^k i! \binom{m}{i} \left\{ {kl \atop i} \right\},$$

where $\left\{ {s \atop t} \right\}$ stands for the *Stirling number of the second kind*.

We show this using an analogy of throwing balls into bins, and as a bonus produce the first mechanised proof of Stirling numbers of the 2nd kind.

Looks quite complicated, what about other similar structures?

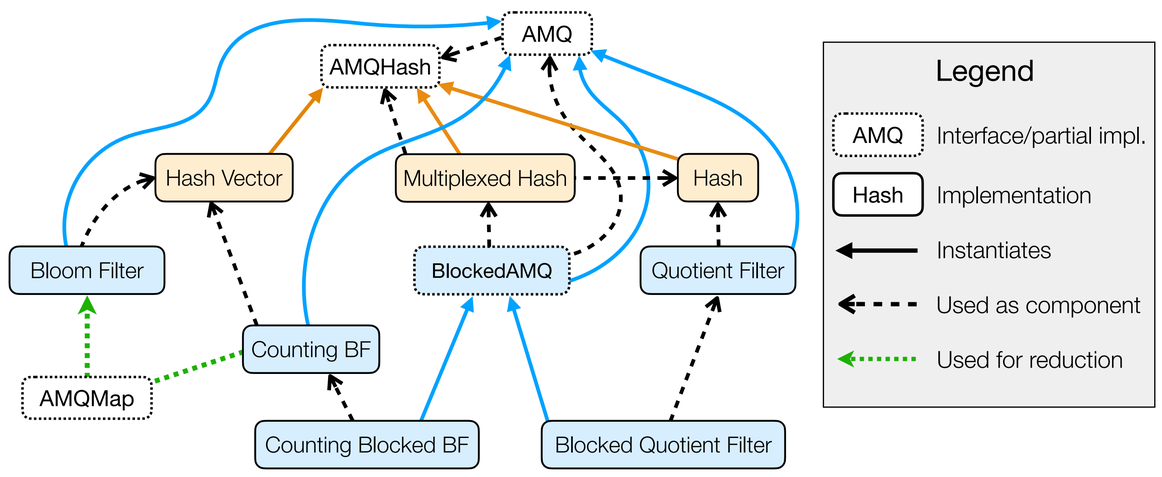
4 Certified AMQs for free

Decomposing AMQs into modular subcomponents

Discoveries:

- The previous decomposition is actually **common to most AMQs**
- Many of the patterns used in AMQs can be generalized into a **higher-order family** of abstract "blocked" AMQs.

Using this, we construct a collection of **reusable subcomponents**, which can be mixed and matched together to build custom AMQs:



The following are some of the AMQs we construct:

- **Counting Bloom filter** - a variant of the Bloomfilter that uses counters instead of bits to support removing elements.
- **Quotient filter** - a filter structure that uses fingerprinting and quotienting to optimize for cache usage.
- **Blocked Bloom filter** - a filter using hash functions to multiplex queries between several constituent Bloomfilters.

Future work: Synthesis of AMQ-based optimizations.

Interested? Ping @kiran gopinathan on PLDI slack to have a chat.

References

[1] *Google chrome safe browsing.* [Online]. Available: https://src.chromium.org/viewvc/chrome/trunk/src/chrome/browser/safe_browsing.
[2] *Bloomfilters - Apache Cassandra.* [Online]. Available: https://cassandra.apache.org/doc/latest/operating/bloom_filters.html.
[3] S. Quinlan and S. Dorward, "Venti: A new approach to archival storage.," in *FAST*, vol. 2, 2002, pp. 89–101.
[4] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.
[5] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid, and Y. Tang, "On the false-positive rate of Bloom filters," *Information Processing Letters*, 2008.
[6] M. Mitzenmacher, "Compressed Bloom filters," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 604–612, 2002.