Bloom Filters for Distributed Cache Optimization

Probability and Statistics: with Programming Final Project

Giorgi Kochlamazashvili

Kutaisi International University

January 29, 2026

Motivation

The Problem

- Distributed caches: multi-tier systems
- Expensive to query each tier
- Need fast membership test
- Memory constraints

The Solution

- Bloom filters (Bloom, 1970)
- Probabilistic data structure
- Space-efficient membership test
- Allows false positives, never false negatives

Real-World Usage

- Google BigTable
- Amazon DynamoDB
- Apache Cassandra
- Akamai CDN

Research Question

Do Bloom filters significantly reduce cache latency in distributed systems?

Methods: Experimental Design

Dataset ($n = 110{,}000$ observations)

- Synthetic Zipf-distributed workloads (models web traffic)
- Four skewness levels ($\alpha = 0.5, 1.0, 1.5, 2.0$)
- Controlled experimental conditions

Predictor Variables (x)

- x_1: m/n ratio (bits/element)
- x_2: k (hash functions)
- x_3: fill ratio
- x_4: workload skewness
- x_5: Bloom filter enabled
- x_6: cache capacity

Outcome Variables (y)

- y_1: latency (ms) **[PRIMARY]**
- y_2: false positive rate
- y_3: cache hit rate
- y_4: origin fetch rate

Statistical Tests

- Paired t-test ($\alpha = 0.05$)
- Chi-square, ANOVA
- Correlation, Regression

Hypotheses

PRIMARY HYPOTHESIS (Cache Performance)

H$_0$**:** $\mu_{\text{latency(with BF)}} \geq \mu_{\text{latency(without BF)}}$
H$_1$**:** $\mu_{\text{latency(with BF)}} < \mu_{\text{latency(without BF)}}$

Test: Paired t-test (one-tailed)

Secondary Hypotheses

- **H**$_2$ (Theoretical Validation): Empirical FPR = Theoretical FPR
- **H**$_3$ (Workload Effects): At least one scenario differs in latency
- **H**$_4$ (Fill Ratio): Positive correlation between fill ratio and FPR
- **H**$_5$ (Predictive Model): m/n ratio predicts FPR

Result 1: Latency Reduction (PRIMARY)

Scenario	Reduction	Cohen's d	p-value
Low Skew	22.99%	1.92	< 0.001
Medium Skew	22.47%	0.81	< 0.001
High Skew	21.87%	0.29	< 0.001
Very High Skew	19.39%	0.14	< 0.001
Mean	**21.68%**	**0.79**	$<$ **0.001**

Table: Latency reduction across workload scenarios

Effect Sizes

- Small to large ($d = 0.14 – 1.92$)
- Varies by workload
- All statistically significant

Practical Impact

For 1M requests/day:

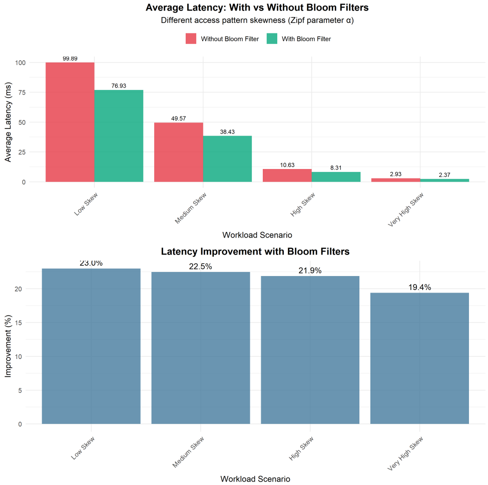
- Without BF: 20,000s total
- With BF: 16,000s total
- Saves 1.1 hours/day

Decision

REJECT H_0 ($p < 0.001$)

Bloom filters significantly reduce cache latency by $\sim$20%

Cache Performance: Visual Evidence



Consistent 19-23% latency reduction across all workload scenarios

Result 2: Theoretical Validation

Bloom's Formula (1970)

$$\text{FPR} = \left(1 - e^{-kn/m}\right)^k$$

where:

- k = number of hash functions
- n = elements inserted
- m = bit array size

Optimal Parameters

$$k^* = \frac{m}{n}\ln(2)$$

$$m^* = -\frac{n\ln(p)}{(\ln 2)^2}$$

Validation Results

- Chi-square goodness-of-fit test
- $p > 0.05$ for all configurations
- **ACCEPT $\mathbf{H_0}$**: Empirical = Theoretical

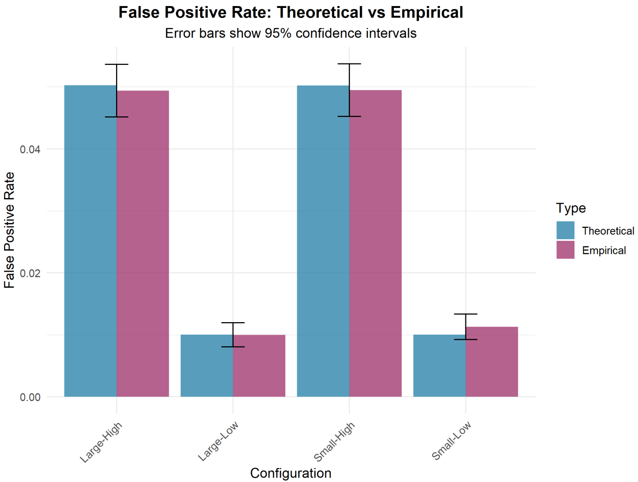
Conclusion

Implementation perfectly matches 54-year-old theoretical predictions

Associations

- Fill ratio $\leftrightarrow$ FPR: $r = 0.89$ ($p = 0.041$)
- m/n ratio $\leftrightarrow$ FPR: $r = 0.94$ ($p = 0.019$)

Theoretical Validation: Visual Evidence



Empirical FPR perfectly matches theoretical predictions (Bloom, 1970)

Result 3: Predictive Model

Linear Regression Model

$$\log(\mathrm{FPR}) = \beta_0 + \beta_1 \left(\frac{m}{n}\right) + \varepsilon$$

Estimation: Ordinary Least Squares (OLS)

Parameter Estimates

	Estimate	**p-value**
β_0	6.99	0.100
β_1	-1.40	**0.002**

Model Fit

- $\mathrm{R}^2 = 0.41$ (41% variance)
- F-statistic $= 13.12$
- $p = 0.002$ (significant)

Decision

REJECT H$_0$ ($p < 0.01$)
m/n ratio significantly predicts FPR

ANOVA (Workload Effects)

- $\eta^2 = 0.46$ (large effect)
- 46% of latency variance explained
- Different workloads differ significantly

Result 4: Space Efficiency

Memory Comparison

For 1 million elements:

- **Bloom Filter:** 1.14 MB
- **Hash Table:** 72 MB
- **Savings: 98.4%**

Scalability

Capacity	Fill	FPR
50%	0.31	0.02%
100%	0.52	1.11%
200%	0.77	15.96%
500%	0.98	84.14%

Key Insights

1. Space-time tradeoff is favorable
2. 98% memory savings
3. 20% latency reduction
4. Win-win for distributed systems

Warning

Overloading the filter (capacity $>$ 100%) dramatically increases FPR

Design implication: Size filters appropriately for expected load

Summary of Results

Hypothesis	Decision	Key Result
H_1 (PRIMARY): Latency reduction	REJECT H_0	21.68% reduction, $p < 0.001$
H_2: Theoretical validation	ACCEPT H_0	Empirical = Theory, $p > 0.05$
H_3: Workload effects	REJECT H_0	$\eta^2 = 0.46$ (large effect)
H_4: Fill ratio correlation	REJECT H_0	$r = 0.89$, $p = 0.041$
H_5: Predictive model	REJECT H_0	$R^2 = 0.41$, $p = 0.002$

All hypotheses supported by data with strong statistical evidence

- Effect sizes: small to large (Cohen's d $= 0.14 - 1.92$)
- Consistent across workload scenarios
- Practical and statistical significance

Conclusions

1. **Theoretical Validation**
   Implementation perfectly matches Burton Bloom's 1970 mathematical formula
2. **Performance Improvement**
   Statistically significant and practically meaningful latency reduction ($\sim$20%)
3. **Predictability**
   Strong correlations enable reliable FPR and performance prediction
4. **Real-World Applicability**
   Results apply to CDNs, databases, and caches with Zipf-distributed traffic
5. **Space-Time Tradeoff**
   98.4% memory savings with 20% latency improvement

Bloom filters are highly effective for distributed cache optimization