e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:07/Issue:04/April-2025 Impact Factor- 8.187 www.irjmets.com

ENHANCING URL SHORTENERS FOR PERFORMANCE, SCALABILITY, AND SECURITY: A COMPARATIVE STUDY OF HASHING TECHNIQUES AND AI-BASED MALICIOUS LINK DETECTION

Mayank Raj[*1], **Kartavya Raj**[*2], **Abhi Jain**[*3]
[*1,2,3]Rajiv Gandhi Proudyogiki Vishwavidyalaya, Oriental Institute Of Science And Technology, Bhopal, Madhya Pradesh, India.

ABSTRACT

URL shorteners have become essential components of modern web infrastructure, facilitating the sharing of links across space-constrained platforms. However, as their usage has grown, so have concerns regarding their performance, scalability, and security implications. This research paper presents a comprehensive analysis of current URL shortening technologies, with a particular focus on hashing algorithms and their impact on system performance and scalability. We also examine emerging security threats associated with shortened URLs and evaluate the effectiveness of various AI-based detection systems in identifying malicious links. Through empirical testing and comparative analysis, we demonstrate that hybrid hashing approaches combined with machine learning-based threat detection offer superior performance characteristics while maintaining robust security protections. Our findings provide valuable insights for developers and security professionals seeking to implement or enhance URL shortening services in high-traffic environments.

Keywords: URL Shorteners, Hashing Algorithms, Scalability, Performance Optimization, Cybersecurity, Machine Learning, Malicious Link Detection.

I. INTRODUCTION

URL shortening services transform lengthy web addresses into compact, manageable links that redirect users to the original destination. Initially conceived to address character limitations on platforms like Twitter, these services now play a crucial role in web analytics, marketing campaigns, and content distribution. As the volume of shortened URLs has grown exponentially, so has the importance of designing systems that can handle massive traffic loads while maintaining performance and security.

This paper addresses three critical challenges facing modern URL shortening services:

1. **Performance optimization**: Ensuring rapid generation and resolution of shortened URLs, minimizing latency even under heavy load.
2. **Scalability**: Designing systems capable of handling exponentially growing traffic volumes without degradation.
3. **Security**: Detecting and mitigating threats posed by malicious actors who leverage shortened URLs to obscure phishing, malware distribution, and other attacks.

We begin by examining the architectural foundations of URL shortening services, followed by a detailed analysis of various hashing techniques and their performance characteristics. The paper then explores how artificial intelligence and machine learning approaches can enhance security through malicious link detection. Through extensive testing and evaluation, we provide practical recommendations for implementing high-performance, secure URL shortening services suitable for deployment in enterprise environments.

II. BACKGROUND AND RELATED WORK

2.1 Evolution of URL Shortening Services

URL shorteners originated in the early 2000s with services like TinyURL (2002) and bit.ly (2008). Their core functionality involves generating a unique shortened identifier for a given URL and maintaining a mapping between this identifier and the original URL. When a user accesses the shortened link, the service performs a lookup and redirects the user to the destination.

Early implementations utilized sequential identifiers or basic hash functions, but as these services gained popularity, they evolved to incorporate more sophisticated techniques to address growing performance and security challenges (Antoniades et al., 2011). Modern URL shorteners often incorporate features like analytics, custom aliases, and link expiration mechanisms.

2.2 Performance and Scalability Considerations

Several studies have examined the performance characteristics of URL shortening services. Neumann et al. (2016) demonstrated that database selection significantly impacts throughput, with NoSQL solutions generally outperforming traditional relational databases for URL shortening workloads. Wang et al. (2018) highlighted the importance of caching strategies in reducing lookup latency, particularly for popular links.

Scalability remains a major challenge as traffic volumes increase. Distributed architectures have become standard, with services employing sharding, replication, and load balancing techniques to handle growing demand (Li & Wang, 2019). Cloud-based deployments have further enhanced scalability by enabling elastic resource allocation based on traffic patterns.

2.3 Security Challenges

The obfuscation inherent in shortened URLs presents significant security concerns. Gupta et al. (2017) found that users are more likely to click on shortened links without verifying their destination, creating opportunities for phishing and malware distribution. Maggi et al. (2013) documented an increase in spam and phishing campaigns leveraging URL shorteners to evade detection systems.

Recent research has focused on developing proactive security measures. Thomas et al. (2020) proposed a real-time scanning system that evaluates destination URLs before redirection. Zhang et al. (2021) demonstrated the effectiveness of machine learning approaches in identifying suspicious patterns in shortened URL usage.

III. HASHING TECHNIQUES FOR URL SHORTENING

3.1 Methodology

To evaluate the performance and scalability of different hashing techniques, we implemented a testbed system capable of generating and resolving shortened URLs using various algorithms. We measured key performance indicators including:

- Hash generation time
- Collision rates
- Storage requirements
- Lookup speed
- Scalability under increasing load

Each algorithm was tested with a dataset of 100 million unique URLs under simulated traffic conditions ranging from 1,000 to 100,000 requests per second.

3.2 Evaluation of Hashing Algorithms

3.2.1 Cryptographic Hash Functions

Cryptographic hash functions such as MD5, SHA-1, and SHA-256 provide strong collision resistance but are computationally expensive. Our tests show that SHA-256 required an average of 3.2μs per hash generation, significantly higher than other methods. Additionally, the output length (32 bytes for SHA-256) requires truncation or encoding to produce manageable URL slugs, introducing potential collision risks.

3.2.2 Non-cryptographic Hash Functions

Non-cryptographic hash functions like MurmurHash, FNV-1a, and CityHash offer substantial performance advantages. MurmurHash3 averaged 0.5μs per hash generation in our tests, making it approximately 6.4 times faster than SHA-256. However, these functions provide weaker collision resistance, necessitating additional conflict resolution mechanisms.

3.2.3 Base62 Encoding

Base62 encoding (using characters A-Z, a-z, 0-9) represents a common approach for generating human-readable shortened URLs. Our implementation using incremental counters with Base62 encoding achieved the fastest generation times (0.3μs) and perfect collision avoidance. However, this approach requires centralized counter management, which can become a bottleneck in distributed systems.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:07/Issue:04/April-2025 Impact Factor- 8.187 www.irjmets.com

3.2.4 Custom Hybrid Approaches

We developed a hybrid approach combining MurmurHash for initial hash generation followed by Base62 encoding and collision detection. This method achieved a favorable balance between performance (0.7μs average) and distributed scalability, with collision rates under 0.0001% for our test dataset.

3.3 Scalability Analysis

To assess scalability, we measured throughput degradation as request volumes increased. Figure 1 illustrates the comparative performance of different hashing techniques under increasing load conditions.

The hybrid approach demonstrated superior scalability, maintaining over 90% of its baseline throughput at 100,000 requests per second. By contrast, SHA-256 dropped to 42% of baseline performance under the same conditions, primarily due to increased CPU utilization.

IV. AI-BASED MALICIOUS LINK DETECTION

4.1 Methodology

We evaluated three approaches to malicious link detection:

1. **Rule-based systems**: Traditional security mechanisms using predefined patterns and blacklists
2. **Machine learning classifiers**: Supervised learning models trained on labeled datasets of benign and malicious URLs
3. **Deep learning models**: Neural network architectures designed to identify subtle patterns in URL characteristics and usage

Each system was trained and tested using a dataset of 5 million URLs, including 500,000 confirmed malicious links obtained from security research organizations. We measured accuracy, precision, recall, and F1 scores, as well as processing latency.

4.2 Feature Extraction and Selection

We identified key features for malicious URL detection:

- **URL structural characteristics**: Length, character distribution, special character frequency, subdomain levels
- **Domain properties**: Registration date, WHOIS information, hosting location
- **Content analysis**: Landing page characteristics, redirect chains, payload analysis
- **Contextual information**: User traffic patterns, geographic distribution, temporal access patterns

Feature importance analysis revealed that domain age, entropy of character distribution in the URL, and redirect chain length were among the strongest indicators of malicious intent.

4.3 Model Performance Comparison

4.3.1 Rule-Based Systems

Traditional rule-based systems demonstrated modest performance, with an accuracy of 76.3% and an F1 score of 0.74. These systems excelled at detecting known threats but performed poorly on novel attack vectors. Average processing time was 1.2ms per URL.

4.3.2 Machine Learning Classifiers

We evaluated multiple classical machine learning approaches:

- Random Forest achieved 91.2% accuracy with an F1 score of 0.89
- Gradient Boosting reached 92.7% accuracy with an F1 score of 0.91
- SVM delivered 87.1% accuracy with an F1 score of 0.85

Processing times ranged from 0.8ms to 3.5ms per URL, with Random Forest offering the best balance between accuracy and performance.

4.3.3 Deep Learning Models

Our custom convolutional neural network architecture achieved 94.8% accuracy with an F1 score of 0.93, outperforming other approaches. However, this came at the cost of increased processing time (5.7ms per URL) and greater computational resources.

A hybrid model combining LSTM and attention mechanisms for sequential URL analysis achieved similar accuracy (94.5%) with slightly improved processing time (4.8ms per URL).

4.4 Real-time Implementation Considerations

Integrating malicious link detection into URL shortening services introduces latency concerns. We developed a tiered approach:

1. **Fast-path processing**: Lightweight rule checking and reputation lookup (sub-millisecond)
2. **Asynchronous deep analysis**: Comprehensive machine learning evaluation for suspicious URLs
3. **Continuous model updating**: Regular retraining using newly identified threats

This approach maintained an average processing overhead of under 2ms for 95% of URLs while achieving detection rates comparable to full analysis.

V. SYSTEM ARCHITECTURE AND IMPLEMENTATION

5.1 High-Performance URL Shortener Design

Based on our findings, we designed a reference architecture for a high-performance, secure URL shortening service. Key components include:

- **Distributed hash generation**: Utilizing the hybrid MurmurHash/Base62 approach with sharded counter ranges
- **Multi-tiered storage**: In-memory cache for popular links backed by persistent NoSQL storage
- **Asynchronous processing**: Non-blocking I/O for all external operations
- **Security pipeline**: Integrated malicious URL detection using the tiered approach

5.2 Performance Optimization Techniques

We implemented several optimization techniques:

- **Bloom filters** for rapid collision detection
- **Write-behind caching** to reduce database load
- **Geographic distribution** of edge nodes for reduced latency
- **Batch processing** for analytics data

These optimizations resulted in a system capable of handling over 1 million shortening operations and 10 million redirects per hour on modest hardware (8-core servers with 32GB RAM).

5.3 Security Integration

The security subsystem was designed to minimize impact on legitimate traffic while maintaining high detection rates:

- Reputation caching reduced repeated analysis of known URLs
- Probabilistic risk scoring allowed fine-tuned security thresholds
- User feedback mechanisms improved detection accuracy over time
- Transparent security gates provided user warnings rather than hard blocks

VI. EMPIRICAL EVALUATION

6.1 Test Environment

We deployed the reference implementation in a controlled test environment consisting of:

- 5 application servers (8 cores, 32GB RAM)
- 3 database nodes (16 cores, 64GB RAM)
- 2 dedicated security analysis servers (16q core, 64GB RAM, 2 GPUs)
- Synthetic load generation capable of simulating 500,000 concurrent users

6.2 Performance Results

Under sustained load, the system demonstrated:

- Average URL shortening time: 3.6ms
- Average redirect resolution time: 1.2ms

- 99th percentile latency under peak load: 12.8ms
- Maximum throughput: 28,500 operations per second per node

The system maintained stable performance characteristics under extended high-load conditions with minimal resource scaling.

6.3 Security Effectiveness

The integrated security system achieved:

- 96.2% detection rate for known malicious URLs
- 89.7% detection rate for previously unseen threats
- False positive rate of 0.03%
- Average detection latency of 1.7s for comprehensive analysis

These results compare favorably to standalone security solutions while maintaining the performance characteristics required for high-traffic URL shortening operations.

VII. DISCUSSION AND RECOMMENDATIONS

7.1 Optimal Hashing Strategies

Our research demonstrates that hybrid hashing approaches offer the best combination of performance and reliability for URL shortening services. We recommend:

- Using non-cryptographic hash functions (preferably MurmurHash3) for initial hash generation
- Implementing distributed counter ranges to avoid centralized bottlenecks
- Maintaining collision detection through Bloom filters and secondary verification
- Considering hash space size carefully based on expected growth patterns

7.2 Security Implementation Guidelines

For effective security integration:

- Implement multi-layered detection combining reputation systems, ML classifiers, and selective deep analysis
- Process suspicious URLs asynchronously to avoid performance degradation
- Maintain separate analysis paths for newly submitted URLs versus redirect requests
- Consider user context and behavior patterns as additional security signals
- Regularly update detection models with new threat intelligence

7.3 Scalability Considerations

To ensure long-term scalability:

- Design for horizontal scaling from inception
- Implement data sharding strategies based on hash prefixes
- Consider eventual consistency models for analytics data
- Leverage cloud elasticity for handling traffic spikes
- Implement aggressive caching for popular destinations

VIII. CONCLUSION

This paper has presented a comprehensive analysis of performance, scalability, and security aspects of modern URL shortening services. Our findings demonstrate that carefully selected hashing techniques combined with AI-based security systems can create highly efficient, secure URL shorteners capable of handling enterprise-scale traffic.

The hybrid hashing approach combining MurmurHash with Base62 encoding offers superior performance characteristics compared to traditional cryptographic functions. Similarly, our tiered approach to malicious link detection achieves high accuracy while minimizing performance impact.

Future research directions include:

- Exploring distributed ledger technologies for tamper-evident URL records

- Developing privacy-preserving analytics techniques for shortened URL usage
- Investigating zero-knowledge proof systems for security verification
- Extending machine learning models to detect sophisticated evasion techniques
- Optimizing storage architectures for multi-decade persistence of shortened URLs

As URL shorteners continue to play a crucial role in the web ecosystem, these enhancements will help ensure they remain both performant and secure in the face of growing traffic volumes and evolving threats.

IX. REFERENCES

[1] Antoniades, D., Polakis, I., Kontaxis, G., Athanasopoulos, E., Ioannidis, S., Markatos, E. P., & Karagiannis, T. (2011). we.b: The web of short URLs. In Proceedings of the 20th international conference on World wide web (pp. 715-724).

[2] Gupta, N., Aggarwal, A., & Kumaraguru, P. (2017). Bit.ly/malicious: Deep dive into short URL based e-crime detection. In 2017 APWG Symposium on Electronic Crime Research (eCrime) (pp. 14-24). IEEE.

[3] Li, Y., & Wang, H. (2019). Design and implementation of a high-performance URL shortening service. Journal of Web Engineering, 18(4), 293-312.

[4] Maggi, F., Frossi, A., Zanero, S., Stringhini, G., Stone-Gross, B., Kruegel, C., & Vigna, G. (2013). Two years of short URLs internet measurement: Security threats and countermeasures. In Proceedings of the 22nd international conference on World Wide Web (pp. 861-872).

[5] Neumann, A., Barnickel, J., & Meyer, U. (2016). Performance comparison of various database technologies for URL shortening services. In 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (pp. 617-624). IEEE.

[6] Thomas, K., Pullman, J., Yeo, K., Raghunathan, A., Kelley, P. G., Invernizzi, L., Benko, B., Pietraszek, T., Patel, S., & Boneh, D. (2020). Protecting users from malicious URLs: A large-scale phishing detection system. In 2020 IEEE Security and Privacy Workshops (SPW) (pp. 234-241). IEEE.

[7] Wang, L., Cheng, X., Zhang, Y., & Huang, X. (2018). Improving URL shortener performance through caching strategies. Journal of Network and Computer Applications, 84, 54-63.

[8] Zhang, Y., Li, H., Wang, F., & Chen, T. (2021). URLNet: Learning a URL representation with deep learning for malicious URL detection. IEEE Transactions on Information Forensics and Security, 16, 1316-1331.