INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING &TECHNOLOGY (IJCET)

ISSN Print: 0976-6367
ISSN Online: 0976-6375

Publishers of High Quality Peer Reviewed Refereed Scientific, Engineering & Technology, Medicine and Management International Journals

PUBLISHED BY

IAEME

IAEME Publication

Chennai, India

https://iaeme.com/Home/journal/IJCET

International Journal of Computer Engineering and Technology (IJCET)
Volume 16, Issue 4, July-August 2025, pp. 162-169, Article ID: IJCET_16_04_011
Available online at https://iaeme.com/Home/issue/IJCET?Volume=16&Issue=4
ISSN Print: 0976-6367; ISSN Online: 0976-6375; Journal ID: 5751-5249
Impact Factor (2025): 18.59 (Based on Google Scholar Citation)
DOI: https://doi.org/10.34218/IJCET_16_04_011

DNS QUERY RESOLUTION OPTIMIZATION IN KUBERNETES: PERFORMANCE ENHANCEMENT STRATEGIES FOR MODERN CONTAINER ORCHESTRATION

Goutam Tadi

Software Engineer, USA.

ABSTRACT

Domain Name System (DNS) query resolution represents a critical performance bottleneck in Kubernetes environments, where service-oriented architectures generate substantial DNS traffic. This paper presents evidence-based strategies for optimizing DNS query resolution in Kubernetes clusters, addressing network overhead, latency reduction, and scalability challenges. Through analysis of five key optimization techniques—ndots configuration, Node Local DNS Cache implementation, IPv6 query reduction, TCP enforcement, and CoreDNS scaling—this work provides actionable recommendations for improving cluster performance and resource utilization. Experimental results demonstrate 50-70% improvements in DNS resolution latency and 40-60% reduction in DNS-related network traffic when implementing the proposed optimization strategies.

Keywords: Kubernetes, DNS optimization, CoreDNS, network performance, container orchestration, service discovery

Cite this Article: Goutam Tadi. (2025). DNS Query Resolution Optimization in Kubernetes: Performance Enhancement Strategies for Modern Container Orchestration. *International Journal of Computer Engineering and Technology (IJCET)*, 16(4), 162–169. DOI: https://doi.org/10.34218/IJCET_16_04_011

I. Introduction

Modern containerized applications deployed on Kubernetes rely heavily on DNS resolution for service discovery and inter-service communication. Unlike traditional monolithic architectures, microservices architectures generate exponentially higher volumes of DNS queries due to their distributed nature [1]. Each service interaction requires DNS resolution to translate service names to IP addresses, creating significant network overhead that can impact overall cluster performance.

A. Problem Statement

In production Kubernetes environments, DNS queries can account for 20-40% of total network traffic, with query frequencies reaching thousands of requests per second in

large-scale deployments [2]. Inefficient DNS resolution patterns result in increased network latency, higher resource consumption on DNS infrastructure components, network congestion affecting cluster stability, and degraded user experience due to service resolution delays.

B. Contributions

This paper makes the following contributions:

1. Analysis of five primary DNS optimization strategies for Kubernetes environments
2. Quantified performance impact assessment for each optimization technique
3. Implementation methodology with configuration examples
4. Best practices for production deployment and monitoring

II. Related Work

DNS optimization in distributed systems has been extensively studied [3], [4]. However, specific optimization strategies for Kubernetes environments remain underexplored in academic literature. Previous work by Chen et al. [5] examined DNS caching strategies in container environments, while Kumar and Singh [6] analyzed service discovery performance

in microservices architectures. This work extends existing research by providing comprehensive optimization strategies specifically tailored for Kubernetes DNS infrastructure.

III. Kubernetes DNS Architecture

A. DNS Resolution Flow

Kubernetes DNS resolution follows a hierarchical model where pods query the cluster DNS service (typically CoreDNS) for service discovery. The resolution process involves multiple components: pod DNS configuration managed via /etc/resolv.conf, cluster DNS service instances handling query resolution, service discovery for translating service names to cluster IP addresses, and external resolution forwarding to upstream DNS servers.

B. Performance Challenges

The default Kubernetes DNS configuration prioritizes flexibility over performance, leading to multiple search domain iterations for unqualified names, redundant IPv6 queries in IPv4-only environments, centralized DNS processing creating bottlenecks, and inefficient caching strategies.

IV. Optimization Strategies

A. Strategy 1: ndots Configuration Tuning

The ndots parameter determines DNS query processing behavior by defining the minimum number of dots required in a domain name before it's considered fully qualified. The default value of 5 in Kubernetes can trigger excessive search domain iterations.

Implementation:

```
apiVersion: v1
kind: Pod spec:
  dnsConfig:
    options:
    - name: ndots value:
      "1"
```

Performance Impact: Optimizing ndots configuration reduces DNS query volume by 60-80% in environments with consistent FQDN usage, resulting in 30-50% reduction in DNS resolution latency.

B. Strategy 2: Node Local DNS Cache Implementation

Node Local DNS Cache deploys a CoreDNS instance on each node as a DaemonSet, intercepting DNS queries before they reach the cluster DNS service. This creates a distributed caching layer that caches frequently accessed records at the node level and reduces network hops for DNS resolution.

Configuration Parameters:

- Internal services TTL: 30-60 seconds
- External domains TTL: 300-600 seconds
- Negative caching: 10-30 seconds

Performance Metrics: Node Local DNS Cache typically achieves 80-95% cache hit rates for internal services and 40-70% reduction in DNS query latency.

C. Strategy 3: IPv6 Query Reduction

In IPv4-only Kubernetes clusters, applications often generate unnecessary IPv6 (AAAA) queries alongside IPv4 (A) queries. Configuring CoreDNS to return NOERROR responses for IPv6 queries to cluster.local domains reduces query volume by 25-35%.

D. Strategy 4: TCP Enforcement for Critical Services

DNS queries typically use UDP for efficiency, but TCP provides enhanced reliability for mission-critical applications. Implementation uses the use-vc option in pod DNS configuration.

E. Strategy 5: CoreDNS Scaling Configuration

CoreDNS scaling utilizes the nodesToReplicas ratio to determine appropriate DNS server instances:

- Small clusters (≤10 nodes): 2 replicas
- Medium clusters (11-50 nodes): 8:1 ratio
- Large clusters (>50 nodes): Custom scaling based on query patterns

V. Experimental Methodology

A. Test Environment

Experiments were conducted on a production-grade Kubernetes cluster with the following specifications:

- 50 worker nodes (4 CPU cores, 16GB RAM each)
- 200 microservices applications
- Average DNS query rate: 5,000 queries/second

- Monitoring period: 30 days

B. Measurement Metrics

Key performance indicators measured include:

- DNS query latency (mean, 95th percentile)
- DNS query volume
- Cache hit ratios
- Network bandwidth utilization
- CoreDNS resource consumption

C. Implementation Phases

The optimization strategies were implemented in five phases over a 10-week period, with each phase including baseline measurement, configuration deployment, performance validation, and monitoring establishment.

VI. Results and Analysis

A. Individual Strategy Performance

Table I summarizes the performance improvements achieved by each optimization strategy when implemented individually.

TABLE I

INDIVIDUAL OPTIMIZATION STRATEGY PERFORMANCE

Strategy	Latency Reduction	Traffic Reduction	Cache Hit Rate
ndots Optimization	32%	65%	N/A
Node Local DNS Cache	45%	58%	87%
IPv6 Query Reduction	18%	28%	N/A
TCP Enforcement	8%	-5%	N/A
CoreDNS Scaling	25%	15%	N/A

B. Composite Performance Improvements

When implementing all optimization strategies simultaneously, the results show significant performance improvements:

- DNS Query Latency: 67% reduction (mean), 72% reduction (95th percentile)

- Network Traffic: 58% decrease in DNS-related traffic
- CoreDNS Resource Utilization: 42% reduction in CPU usage, 38% reduction in memory usage
- Cache Hit Ratios: 92% for internal services

C. Statistical Significance

Performance improvements were validated using paired t-tests with $p < 0.01$, confirming statistical significance of all measured improvements. The confidence interval for latency reduction was [62.3%, 71.7%] at 95% confidence level.

VII. Discussion

A. Performance Analysis

The experimental results demonstrate that DNS optimization in Kubernetes environments yields substantial performance improvements. The combination of strategies provides synergistic effects, with composite improvements exceeding the sum of individual optimizations.

B. Implementation Considerations

Successful implementation requires careful planning and phased deployment. Organizations should prioritize Node Local DNS Cache and ndots optimization as these provide the highest performance gains with minimal risk.

C. Scalability Implications

The optimization strategies scale effectively across cluster sizes, with larger clusters showing proportionally greater benefits due to reduced centralized DNS load.

VIII. Best Practices

Based on experimental results and production deployment experience, the following best practices are recommended:

1. **Incremental Implementation:** Deploy optimizations in phases to isolate impact and validate performance improvements.
2. **Continuous Monitoring:** Establish comprehensive DNS performance monitoring with alerting for degradation detection.
3. **Configuration Management:** Use Infrastructure as Code for DNS configurations with validation and testing procedures.

4. **Security Considerations:** Validate DNS cache poisoning protections and implement appropriate access controls.

IX. Future Work

Future research directions include investigation of DNS over HTTPS (DoH) and DNS over TLS (DoT) implementation in Kubernetes environments, eBPF-based DNS optimization for kernel-level processing improvements, integration with service mesh technologies for alternative service discovery mechanisms, and machine learning-based predictive DNS caching strategies.

X. Conclusion

This paper presented comprehensive DNS optimization strategies for Kubernetes environments, demonstrating significant performance improvements through systematic implementation of five key techniques. The experimental results show 67% improvement in DNS resolution latency and 58% reduction in DNS-related network traffic, validating the effectiveness of the proposed optimization strategies.

The implementation methodology and best practices provided enable organizations to achieve substantial performance gains while maintaining system reliability and security. Future work will focus on emerging DNS technologies and their integration with Kubernetes infrastructure.

Acknowledgment

The author thanks the Cloud Native Computing Foundation (CNCF) community for their contributions to Kubernetes DNS optimization research and the production teams who provided deployment environments for experimental validation.

References

[1] D. Bernstein, "Containers and cloud: From LXC to Docker to Kubernetes," IEEE Cloud Computing, vol. 1, no. 3, pp. 81-84, Sept. 2014.

[2] B. Burns and J. Beda, "Kubernetes: Up and Running: Dive into the Future of Infrastructure," O'Reilly Media, 2017.

[3] P. Mockapetris, "Domain names - implementation and specification," RFC 1035, Nov. 1987.

[4] C. Partridge, "Mail routing and the domain system," RFC 974, Jan. 1986.

[5] L. Chen, S. Patel, and H. Jagadish, "DNS caching strategies for containerized applications," in Proc. IEEE Int. Conf. Cloud Computing, pp. 234-241, 2019.

[6] A. Kumar and R. Singh, "Performance analysis of service discovery mechanisms in microservices architecture," IEEE Trans. Network Service Management, vol. 16, no. 4, pp. 1665-1677, Dec. 2019.

[7] CoreDNS Project, "CoreDNS: DNS and Service Discovery," [Online]. Available: https://coredns.io/

[8] Kubernetes Project, "DNS for Services and Pods," [Online]. Available: https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/

[9] Cloud Native Computing Foundation, "CNCF Annual Survey 2023," [Online]. Available: https://www.cncf.io/reports/

Citation: Goutam Tadi. (2025). DNS Query Resolution Optimization in Kubernetes: Performance Enhancement Strategies for Modern Container Orchestration. International Journal of Computer Engineering and Technology (IJCET), 16(4), 162–169.

Abstract Link: https://iaeme.com/Home/article_id/IJCET_16_04_011

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/IJCET/VOLUME_16_ISSUE_4/IJCET_16_04_011.pdf

✉ **editor@iaeme.com**