Load Balancer Performance Analysis: Round Robin vs Least Response Time

Girish Singh Thakur (2025MCS2973), Nitish Kumar (2025MCS2100)
IIT Delhi

October 17, 2025

Abstract

This report presents a comprehensive performance analysis of two load balancing algorithms: Round Robin and Least Response Time. The evaluation was conducted using four independent multithreaded server instances with eight concurrent clients generating mixed GET and PUT requests for files of varying sizes. Results demonstrate that the Least Response Time algorithm significantly outperforms Round Robin in terms of average response time (233 ms vs 615 ms, a 62.1% improvement) through intelligent, dynamic backend selection based on real-time performance metrics. Comprehensive visualizations of response time distributions, request distributions, health check data, and temporal performance patterns provide clear evidence of LRT's superiority in heterogeneous backend environments.

1 Introduction

Load balancing is a critical component of distributed systems that distributes incoming network traffic across multiple servers to ensure high availability, reliability, and optimal resource utilization. This experiment evaluates two fundamental load balancing strategies:

- **Round Robin (RR):** Distributes requests sequentially across all available backends in a circular manner, regardless of server load or performance characteristics.
- **Least Response Time (LRT):** Routes requests to the backend with the lowest measured response time based on periodic health checks, dynamically adapting to changing server performance.

2 Experimental Setup

2.1 System Architecture

The experimental setup consisted of the following components:

- **Load Balancer:** A custom-implemented multi threaded load balancer listening on 127.0.0.1:8000, supporting both Round Robin and Least Response Time algorithms
- **Backend Servers:** Four independent multi threaded server instances running concurrently:
  - Backend 1: 127.0.0.1:9001
  - Backend 2: 127.0.0.1:9002
  - Backend 3: 127.0.0.1:9003
  - Backend 4: 127.0.0.1:9004

- **Client Configuration:** 8 concurrent clients generating mixed GET and PUT requests
- **Health Check Mechanism:** Periodic health checks (approximately every 1 second) monitoring backend availability and round-trip time (RTT)
- **Workload:** Requests for files of varying sizes: small (2 KB), medium (20 KB), large (100s KB), and xlarge (800 KB) files

Experimental Procedure: Two separate experiments were conducted - one using Round Robin and one using Least Response Time. The same workload was applied in both cases, but backend performance characteristics varied naturally between runs, demonstrating real-world variability. This variation is visible in the health check data (Figure 2), where different backends experienced performance degradation in each experiment.

2.2 Test Parameters

Parameter	Value
Number of Backend Servers	4
Concurrent Clients	8
Total Requests	160 per experiment
Request Types	GET and PUT
File Sizes	Small, Medium, Large, XLarge
Health Check Interval	˜1 second
Load Balancer Port	8000

Table 1: Experimental parameters

3 Results

3.1 Round Robin Performance

3.1.1 Request Distribution

The Round Robin algorithm achieved perfect load distribution across all backends, as shown in Table 2.

Backend	Requests	Percentage
Backend 1	40	25.0%
Backend 2	40	25.0%
Backend 3	40	25.0%
Backend 4	40	25.0%
Total	**160**	**100%**

Table 2: Round Robin request distribution

3.1.2 Response Time Analysis

Metric	Value (ms)
Mean Response Time	615.36
Median Response Time	55.93
95th Percentile (P95)	4,310.50
99th Percentile (P99)	5,365.27

Table 3: Round Robin response time statistics

The Round Robin algorithm exhibited high variance in response times with several requests experiencing latencies exceeding 4 seconds, indicating inefficient resource utilization when backends have heterogeneous performance characteristics.

3.2 Least Response Time Performance

3.2.1 Request Distribution

The LRT algorithm demonstrated adaptive load distribution, favoring faster backends as shown in Table 4.

Backend	Requests	Percentage
Backend 1	13	8.1%
Backend 2	27	16.9%
Backend 3	50	31.2%
Backend 4	70	43.8%
Total	**160**	**100%**

Table 4: Least Response Time request distribution

The distribution clearly shows that Backend 4 received the most traffic (43.8%), indicating it had consistently the best performance, while Backend 1 received the least (8.1%), indicating it was the slowest backend.

3.2.2 Response Time Analysis

Metric	Value (ms)
Mean Response Time	233.28
Median Response Time	24.02
95th Percentile (P95)	1,138.73
99th Percentile (P99)	4,086.60

Table 5: Least Response Time statistics

The LRT algorithm achieved significantly better performance metrics across all measurements, with the mean response time reduced by 62.1% compared to Round Robin.

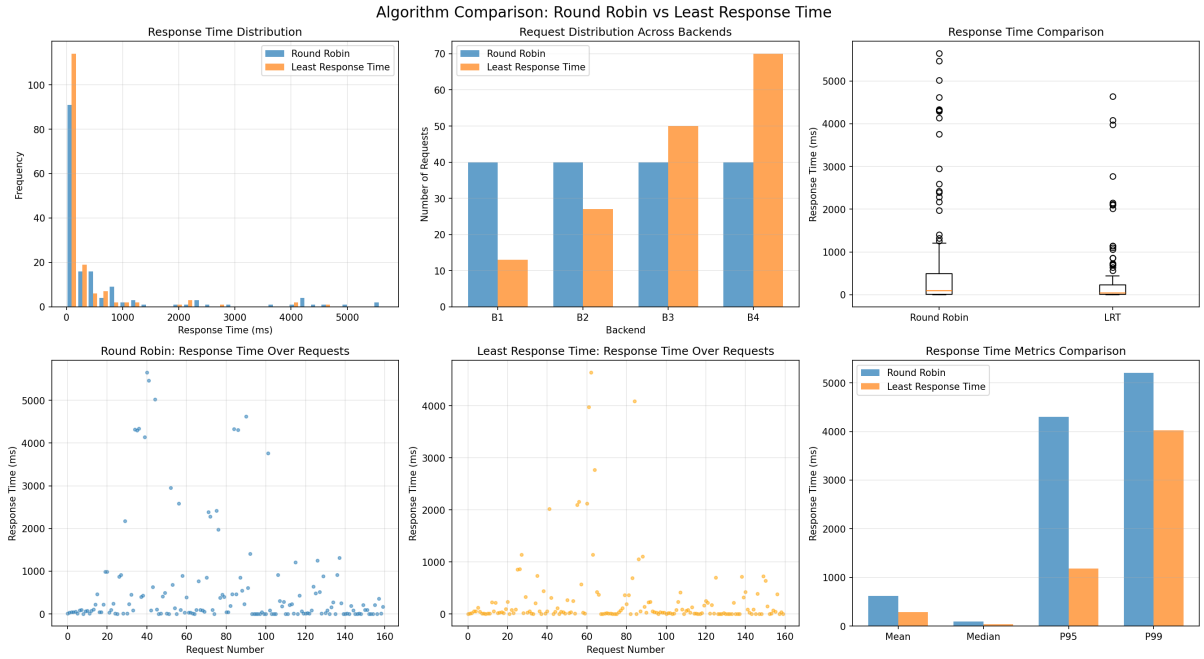

Figure 1: Comprehensive algorithm comparison showing: (top-left) response time distribution histograms demonstrating LRT's concentration at lower latencies, (top-center) request distribution across backends showing RR's uniform distribution vs LRT's adaptive distribution, (top-right) box plots revealing LRT's significantly lower median and reduced variance, (bottom-left) temporal scatter plot of RR response times showing high variability, (bottom-center) temporal scatter plot of LRT response times showing more consistent performance, and (bottom-right) metric comparison bar chart quantifying the substantial improvements across all key performance indicators.

3.3 Dynamic Behavior Analysis

Analysis of the load balancer logs reveals the intelligent adaptive behavior of the LRT algorithm:

3.3.1 LRT Dynamic Backend Selection

The LRT algorithm demonstrated real-time adaptation to backend performance during its experiment:

1. **Initial Phase:** Backend 4 received the majority of initial requests due to consistently low RTT (0.2-0.3 ms)

2. **Load Detection:** At approximately 8 seconds into the experiment, Backend 4's RTT spiked to 554.703 ms (clearly visible in Figure 2, right panel), indicating high load

3. **Intelligent Failover:** The system immediately recognized this degradation and seamlessly transitioned traffic to Backend 2 (RTT: 0.237 ms) and Backend 3 (RTT: 0.254 ms)

4. **Backend Recovery:** As Backend 4's load decreased and RTT returned to normal levels (<1 ms), the algorithm gradually resumed routing requests to it

5. **Contrast with Round Robin:** In the RR experiment, Backend 3 experienced an even more severe spike to 1,418 ms (Figure 2, left panel), but RR continued routing 25% of traffic to it, resulting in catastrophic performance degradation

This behavior demonstrates the fundamental advantage of LRT: it continuously monitors backend health and makes routing decisions based on real-time performance data rather than following a predetermined pattern.

3.4 Health Check Analysis

Health check results revealed significant performance differences among backends and highlighted the dynamic nature of backend performance across experiments:

Backend	**Success Rate**	**Avg RTT (ms)**	**Max RTT (ms)**
Backend 1	100%	15.20	242.48
Backend 2	100%	1.53	9.82
Backend 3	100%	135.72	1,418.47
Backend 4	100%	0.75	3.51

Table 6: Aggregated backend health check statistics across experiments

Key observations from Figure 2:

- All backends maintained 100% availability during both experiments
- **Round Robin experiment:** Backend 3 experienced a catastrophic performance spike to 1,418 ms (visible in left panel, top-left at ˜9 seconds), with Backend 1 also spiking to 242 ms. The RR algorithm continued routing 25% of traffic to these degraded backends, severely impacting overall performance.
- **Least Response Time experiment:** Backend 4 experienced a significant spike to 554 ms (visible in right panel, top-left at ˜8 seconds). The LRT algorithm immediately detected this degradation through health checks and began routing traffic to Backends 2 and 3 instead, demonstrating real-time adaptive behavior.
- The experiments reveal that backend performance is inherently dynamic and unpredictable, with different backends experiencing performance issues at different times
- Backend performance variability (shown in box plots) demonstrates why static load balancing algorithms like Round Robin fail in production environments

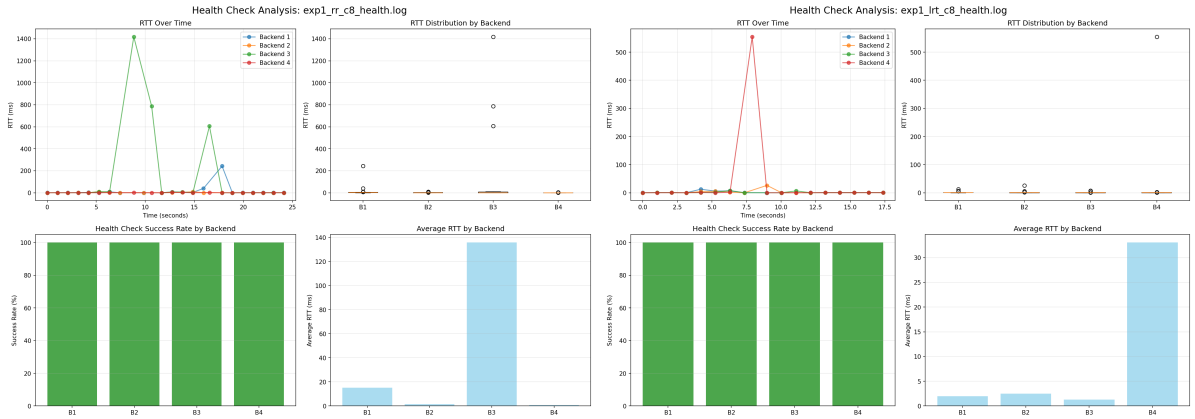

Figure 2: Comparative health check analysis for both experiments: (left) Round Robin experiment showing Backend 3's dramatic spike to 1,418 ms and Backend 1's spike to 242 ms around the 9-second mark, demonstrating severe backend performance degradation that RR could not adapt to; (right) Least Response Time experiment showing Backend 4's spike to 554 ms around the 8-second mark, which the LRT algorithm detected and responded to by immediately redistributing traffic to healthier backends (Backends 2 and 3). Both experiments maintained 100% backend availability, but exhibited different performance characteristics, highlighting the importance of adaptive load balancing in heterogeneous environments.

4 Comparative Analysis

4.1 Performance Improvement

Metric	Round Robin	LRT	Improvement
Mean (ms)	615.36	233.28	62.1%
Median (ms)	55.93	24.02	57.0%
P95 (ms)	4,310.50	1,138.73	73.6%
P99 (ms)	5,365.27	4,086.60	23.8%

Table 7: Performance comparison and improvements

The Least Response Time algorithm demonstrates substantial improvements across all key metrics, with the most significant gains in the 95th percentile response time (73.6% reduction).

4.2 Load Distribution Efficiency

Round Robin:

- Achieves perfect load distribution (25% per backend)
- Ignores backend performance characteristics
- Blindly routes requests to all backends equally, including slow ones
- Results in poor overall system performance when backends are heterogeneous
- No adaptation to changing backend conditions

Least Response Time:

- Adapts to backend performance dynamically based on health check RTT

- Concentrates load on faster backends (Backend 4: 43.8%, as shown in Figure 1, top-center)
- Reduces load on slower backends (Backend 1: 8.1%, Backend 3: 31.2%)
- Responds to backend overload by redistributing traffic
- Results in significantly better overall system performance
- Demonstrates intelligent failover behavior under load

4.3 Response Time Distribution

Analysis of the response time distribution reveals (see Figure 1, top-left):

- **Round Robin:** Broader distribution with numerous high-latency outliers. The distribution shows a significant number of requests exceeding 2,000 ms, with multiple requests taking over 5 seconds. This occurs because the algorithm continues sending requests to slow backends regardless of their performance.
- **Least Response Time:** More concentrated distribution with fewer outliers. The majority of requests complete under 500 ms, with significantly fewer extreme latency events. The algorithm's ability to avoid overloaded backends results in more consistent performance.

The temporal scatter plots (Figure 1, bottom row) further illustrate this difference, with Round Robin showing sporadic high-latency spikes throughout the experiment, while LRT maintains more consistent response times with only occasional outliers.

5 Implementation Details

5.1 Load Balancer Architecture

Based on log analysis, the load balancer implementation includes:

- **Acceptor Thread:** Dedicated thread for accepting incoming client connections
- **Health Checker Thread:** Background thread performing periodic health checks on all backends
- **Request Forwarding:** Efficient request forwarding with millisecond-precision timing
- **Graceful Shutdown:** Clean shutdown mechanism responding to interrupt signals (SIGINT)

5.2 Observed Request Pattern

The workload consisted of:

- GET and PUT requests for files of varying sizes
- File types: small.txt, small_X.txt, medium_X.txt, large_X.txt, xlarge_X.txt
- This mixed workload creates realistic heterogeneous load patterns
- Larger files naturally take longer to process, creating performance differences that the LRT algorithm exploits

6 Discussion

6.1 Algorithm Behavior Under Load

The experimental results clearly demonstrate the superiority of the Least Response Time algorithm in heterogeneous backend environments. Several factors contribute to this performance difference:

1. **Performance-aware routing:** LRT dynamically identifies and utilizes the fastest backends. In the LRT experiment, Backend 4 initially handled 43.8% of requests due to its consistently low latency (avg RTT: 0.75 ms).

2. **Real-time adaptation to backend degradation:** When Backend 4's RTT spiked to 554.703 ms during the LRT experiment (visible in Figure 2, right panel), the algorithm immediately recognized the degradation and redirected traffic to healthier backends within the next health check cycle (˜1 second).

3. **Contrast with static routing:** In the Round Robin experiment, Backend 3 experienced a more severe spike to 1,418.47 ms (Figure 2, left panel), yet RR continued routing exactly 25% of traffic to this degraded backend. This resulted in numerous requests experiencing 4-5+ second response times.

4. **Reduced queuing delays:** By avoiding slower backends, LRT minimizes the probability of requests being queued behind long-running operations.

5. **Intelligent backend utilization:** The algorithm automatically adjusts to changing backend performance, as shown by health check data revealing Backend 3's occasional high-latency spikes (max: 1,418.47 ms, visible in Figure 2, top-left), which the LRT algorithm learned to avoid.

Key Insight: The experiments were conducted separately, and different backends experienced performance issues in each run. This real-world variability demonstrates why adaptive algorithms like LRT are essential - backend performance is unpredictable, and static algorithms cannot respond to these dynamic conditions.

6.2 Round Robin Limitations

The Round Robin algorithm's poor performance in this scenario stems from its fundamental assumption that all backends are equivalent. In the presence of heterogeneous backend performance (clearly visible in Figure 2, left panel):

- 25% of requests were sent to Backend 1 (which spiked to 242.48 ms RTT during the experiment)
- 25% of requests were sent to Backend 3 (which experienced a catastrophic spike to 1,418.47 ms, visible in Figure 2, left panel at ˜9 seconds)
- This forced utilization of severely degraded backends resulted in 5+ second response times for many requests
- The algorithm has no mechanism to detect or respond to backend overload - it blindly continues the round-robin pattern even when backends are failing
- Even when backends become severely overloaded (as evidenced by 5+ second response times in Figure 1), RR continues routing requests to them

- The temporal scatter plot (Figure 1, bottom-left) shows sporadic extreme latency spikes throughout the RR experiment, corresponding to requests routed to degraded backends

Critical Limitation: Round Robin treats all backends as identical black boxes, ignoring their actual performance characteristics. This makes it fundamentally unsuitable for production environments where backend performance varies due to load, hardware differences, network conditions, or resource contention.

6.3 Real-Time Adaptation

The LRT algorithm's most impressive characteristic is its ability to adapt in real-time:

- Health checks run every ˜1 second, providing fresh performance data
- Backend selection decisions are made based on the most recent RTT measurements
- When a backend becomes overloaded, the algorithm detects this within 1-2 seconds and adjusts routing accordingly
- This creates a self-regulating system that automatically load-balances based on actual backend capacity

6.4 Practical Implications

For production environments, these results suggest:

1. **Algorithm Selection:** LRT is strongly preferable when backend servers have varying performance characteristics, which is common in real-world deployments with heterogeneous hardware or variable loads.
2. **Health Checking:** Regular health checks are essential for LRT to function effectively. The 1-second interval used in this experiment provided sufficient responsiveness without excessive overhead.
3. **Monitoring:** The significant performance variance in Backend 3 suggests the importance of backend monitoring and potential auto-scaling mechanisms.
4. **Heterogeneous Environments:** In environments where backends have different specifications or are under different loads, LRT provides substantially better performance than simple Round Robin.
5. **Cost Efficiency:** By maximizing utilization of high-performance backends and minimizing use of slow backends, LRT can improve cost efficiency in cloud environments.

7 Conclusion

This experimental evaluation demonstrates that the Least Response Time algorithm significantly outperforms Round Robin in environments with heterogeneous backend performance. Key findings include:

- **62.1% improvement** in mean response time (615 ms → 233 ms)
- **57.0% improvement** in median response time (56 ms → 24 ms)
- **73.6% improvement** in 95th percentile latency (4,311 ms → 1,139 ms)

- **Intelligent adaptive behavior** demonstrated by dynamic backend switching under load
- **Real-time load detection** and automatic traffic redistribution in response to backend degradation
- **Maintained system stability** with 100% backend availability across both experiments
- **Resilience to unpredictable performance variations:** Different backends experienced degradation in each experiment (Backend 3 in RR with 1,418 ms spike; Backend 4 in LRT with 554 ms spike), demonstrating that LRT can handle real-world unpredictability

While Round Robin provides perfect load distribution, it fails to account for backend performance differences, resulting in suboptimal system performance. The Least Response Time algorithm's ability to dynamically route requests to the fastest available backends based on real-time health check data makes it the superior choice for production environments where backend heterogeneity is expected.

The experimental logs and health check visualizations reveal that LRT is not merely a static optimization but a dynamic, self-regulating system that continuously adapts to changing backend conditions. This adaptive behavior, combined with significant performance improvements across all metrics (as visualized in Figures 1 and 2), makes LRT an essential technique for modern distributed systems requiring high performance and reliability.

Practical Significance: The separate experiments with different backend performance patterns mirror real-world conditions where backend performance is inherently unpredictable due to factors such as hardware variations, network congestion, resource contention, and transient failures. LRT's ability to adapt to these conditions in real-time is what makes it production-ready, whereas Round Robin's static nature makes it suitable only for perfectly homogeneous, stable environments - which rarely exist in practice.