

**COMPARING THE PERFORMANCE OF
CONCURRENCY MODELS OF LOAD BALANCER
ARCHITECTURES**

Ragavan Thiruchittampalam

189341P

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2021

**COMPARING THE PERFORMANCE OF
CONCURRENCY MODELS OF LOAD BALANCER
ARCHITECTURES**

Ragavan Thiruchittampalam

189341P

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2021

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature :

Date :

The above candidate has carried out research for the Masters thesis under my supervision.

Name of the supervisor:

Signature of the supervisor :

Date :

ABSTRACT

The performance of load balancers is increasingly essential to distribute application traffic across multiple instances efficiently while maintaining a large number of concurrent users and application reliability. There is a myriad of factors that influence the performance of a load balancer, and in this study, the impact of the concurrency model of server architectures in performance is investigated in detail. In this research, how different server architectures - thread-per-connection, reactor and disruptor - can be used to build load balancers was studied and the strength and weaknesses of their concurrency model under a heavy concurrent workload was analysed. Two different reference implementations for thread-per-connection, with and without thread pool, were created to understand the impact of a thread pool. The reactor architecture was implemented utilizing the HTTPCore-NIO library and the disruptor architecture was developed using Netty transport and LMAX java library.

Besides, each implementation was extensively tuned and the performance of the best performing load balancer in each server architecture was compared. Each chosen architecture of load balancer has a distinct set of properties that control the performance, therefore, tuning each implementation was treated as a separate effort. Through the benchmarking tool, JMeter, extensive experiments will be conducted, and response time, throughput, CPU and memory usage were measured to analyse the impact of server architecture on performance. This study produced a comprehensive survey on several concurrency models of load balancer architectures, an experimental illustration and a detailed analysis of load balancers, in terms of performance, under high concurrent load. The results show with proper tuning, the peak throughput of each load balancer is increased by more than 15% compared to its baseline configuration.

Reactor based architecture with the configuration, 4 threads per reactor and 512 worker threads produce the peak throughput of 4362 requests per second which is the highest throughput produced in the experiments. In terms of throughput, the peak throughput produced by reactor-based architecture is 19% better than the maximum throughput generated by disruptor based architecture. The average response time increases exponentially for all load balancer architectures but at different rates. Reactor based architecture produces the best response time for all concurrent connections used in the experiments. A long-tail problem is completely visible in thread-per-connection architecture under high concurrency, where it has a long and thick tail in the response time distribution. The key factors affecting the performance of these architectures are the non-blocking nature of I/O operations, CPU usage, handling contention for shared resources, memory footprint and supporting a high number of concurrent connections.

Keywords: Load Balancer, Performance, Architecture, Concurrency

ACKNOWLEDGEMENT

It is my pleasure to thank everyone who has supported to do this research successfully. I especially wish to thank my research supervisor, Dr Indika Perera, for his valuable advice, support and guidance throughout this research.

My sincere thanks to the entire panel of lecturers of the Master's Program. Without their guidance, I would not have gained this vast knowledge. It is important to mention here that their guidance from the first day brought me to continue my studies.

Finally, I extend my thanks to my family for encouraging and supporting me to follow this program.

Contents

Declaration	ii
Abstract	iii
Acknowledgement	iv
Table of Contents	vi
List of Figures	ix
List of Tables	x
1 CHAPTER 1 - INTRODUCTION	1
1.1 Background and Context	2
1.1.1 Load Balancer Types	4
1.1.2 Load Balancer Features	5
1.1.3 Load Balancing Strategies	6
1.2 Problem Statement	7
1.3 Research Objectives	8
1.4 Scope and Limitation	9
1.5 Document Outline	9
2 CHAPTER 2 - LITERATURE REVIEW	11
2.1 Concurrency Models	12
2.1.1 Bounded Thread Pool	13
2.1.2 Actor Model	14
2.1.3 Disruptor	15
2.2 Server Architectures	17
2.2.1 Thread Based Server Architecture	17
2.2.2 Event Driven Server Architecture	19
2.2.2.1 Single Process Event-Driven Architecture	21

2.2.2.2	SYmmetric Multi-Process Event-Driven Architecture	21
2.2.2.3	Asymmetric Multi-Process Event-Driven Architecture	22
2.2.3	Combined Server Architectures	22
2.2.3.1	Staged Event Driven Architecture	22
2.3	Previous Benchmarks	23
3	CHAPTER 3 - METHODOLOGY	26
3.1	Research Methodology	26
3.2	Proposed High-Level Architecture	29
3.3	Benchmark Setup	30
3.4	Workload	30
3.5	Tuning	31
3.6	Verification	31
3.7	Measuring Metrics	32
4	CHAPTER 4 - ARCHITECTURE IMPLEMENTATION	33
4.1	Thread Per-Connection Architecture	33
4.2	Reactor Based Event Driven Architecture	37
4.3	Disrupter Based Event Driven Architecture	43
5	CHAPTER 5 - IMPLEMENTATION TUNING	46
5.1	General Tuning Steps	46
5.1.1	OS and Network	46
5.1.2	Keep Alive Connections	47
5.2	Common Configuration	48
5.3	Per-Connection Thread Based Architecture	49
5.4	Reactor Based Event Driven Architecture	52
5.5	Disrupter Based Event Driven Architecture	54
6	CHAPTER 6 - RESULTS AND DISCUSSION	57
6.1	Throughput	57
6.2	Response Time	59

7	CHAPTER 7 - CONCLUSION	62
7.1	Research Contribution	63
7.2	Limitation	64
7.3	Future Work	64

List of Figures

2.1	Thread Pool	13
2.2	Disruptor Model	16
2.3	Threaded Server Design	18
2.4	Event Driven Server Design	19
2.5	SEDA Stage	23
3.1	Research Methodology	26
3.2	Load Balancer Architecture	29
3.3	Experiment System Architecture	30
4.1	Blocking I/O Model	34
4.2	Reference Implementation - A new thread for each connection	35
4.3	Reference Implementation - A thread in the thread pool handles connection	36
4.4	Class Diagram of Transport Listener of Reactor Architecture	38
4.5	I/O Multiplexing	40
4.6	Reactor Architecture Based Implementation	41
4.7	Disruptor Based Reference Implementation Transport Listener	43
4.8	Reference Implementation of Disruptor Based Event Driven Architecture	45
5.1	Throughput of Different Configurations of Thread-Per-Connection Load Balancer	50
5.2	Throughput of Different Configurations of Thread Pool Load Balancer	51
5.3	Throughput of Different Configurations of Reactor Based Implementation	53
5.4	Throughput of Different Configurations of Disruptor Based Reference Implementation	55
6.1	Throughput of Different Architecture Implementations	58
6.2	Average Response Time of Different Load Balancer Architectures	59
6.3	Probability Density Function of Different Architectures	60

List of Tables

2.1	I/O Models	12
3.1	Factors and Levels	27