GC AWARE CONTAINERIZED DEPLOYMENT TO ELIMINATE GC LATENCIES

Premarathna E.H.A.D.S.D.

219384L

MSc in Computer Science

Department Of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

07 2025

GC AWARE CONTAINERIZED DEPLOYMENT TO ELIMINATE GC LATENCIES

Premarathna E.H.A.D.S.D.

219384L

Dissertation submitted in partial fulfillment of the requirements for the degree
MSc in Computer Science

Department Of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

07 2025

DECLARATION

I declare that this is my own work and this Dissertation 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. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Shanaka Date: 2025/07/03

The supervisor should certify the Dissertation with the following declaration.

The above candidate has carried out research for the MSc in Computer Science Dissertation under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of Supervisor: Prof. G.I.U.S. Perera

Signature of the Supervisor: Date: 2025/07/04

ABSTRACT

Garbage collection in Java applications continues to be a major source of performance overhead, often causing disruptive pauses in latency sensitive workloads and reducing overall throughput. Although various techniques have been developed ranging from advanced garbage collection algorithms to finely tuned parameters, these strategies do not completely eliminate stop the world interruptions.

This thesis presents a containerized deployment strategy aimed at achieving pause free execution for Java applications, leveraging Kubernetes orchestration and Kubernetes memory monitoring. The proposed methodology employs the Epsilon garbage collector, which inherently disables the garbage collection mechanism, thereby avoiding traditional garbage collection pause events. In place of conventional memory reclamation within a container, the system monitors heap usage and gracefully terminates containers approaching predefined memory thresholds, subsequently launching new containers to handle incoming traffic. This process ensures that application threads do not experience forced suspension for the purpose of heap compaction.

Central to the design is the use of Kubernetes for container lifecycle management, complemented by Kubernetes metrics that detect memory pressure before critical levels are reached. When a container's memory utilization exceeds a configurable limit, traffic is seamlessly redirected to a newly initiated container, allowing the retiring container to shut down without affecting ongoing requests or service availability. By shifting memory reclamation to container restarts, this approach prevents stop the world garbage collection pauses in a manner that reduces operational complexity.

A comprehensive performance evaluation compares the container rotation strategy against the traditional Garbage collection algorithms. Experimental results indicate that the proposed solution consistently delivers lower latency profiles and higher throughput under varying workload conditions. Furthermore, offloading the memory reclamation task to container restarts lessens the need for intricate GC tuning, thus simplifying operational overhead. This thesis demonstrates a viable method of achieving truly pause free Java application execution, with implications for high availability systems and microservices that require minimal disruption from garbage collection.

Keywords: Java, Garbage collection, Containerized deployment, Kubernetes

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
CMS	Concurrent Mark Sweep
CPU	Central Processing Unit
CRD	Kubernetes Custom Resource Definition
EJB	Enterprise Java Beans
GC	Garbage Collection
GCI	Garbage Collection Control Interceptor
GUI	Graphical User Interface
I/O	Input/Output
JMX	Java Management Extensions
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MITMEM	Mitigating Memory-induced Delays
NoSQL	NoSQL
NUMA	Non-Uniform Memory Access
OOM	Out of Memory
OS	Operating System
PaaS	PaaS
RBAC	Role-Based Access Control
STW	Stop The World
THP	Transparent Huge Pages
TLB	Translation Lookaside Buffer