incusdata (pty) ltd. 1995/04149/07
Coding · Matters
PO Box 148
Wierda Park, 0149
tel : +27 12 666 2020
info@incusdata.com
www.incusdata.com

Price: R9,600 excl. VAT
Duration: 3 days
Delivery: Virtual classroom or on-site training

JVM Performance

Available on request for group bookings only

Description

The JVM Performance course takes you beyond Java syntax and into the internal mechanisms that determine how Java applications actually run. You learn how the JIT compiler optimises code execution, how different garbage collectors manage memory, and how to choose appropriate heap sizes and GC strategies for your environment. The course also explores modern threading models and their impact on throughput. By the end, you will understand the key components of JVM internals, memory management and diagnostic techniques that help you identify and resolve performance bottlenecks.

Objectives

After you have completed the JVM Performance Tuning course, you will be able to:

- Understand how the JVM interpreter and Just-In-Time (JIT) compilers execute and optimise Java code.
- Understand JVM memory management and garbage collection.
- Configure JVM memory settings and garbage collectors using command line options.
- Measure JVM performance and perform diagnostics using standard tools.
- Identify, analyse and resolve common memory leaks in Java applications.

Intended Audience

This course is useful if:

- You are an advanced Java programmer and you need to understand JVM internals and how to optimise memory usage and garbage collection.
- You are a Java performance engineer and you need to optimise the execution of Java applications.
- You are a DevOps engineer and you need to configure and tune JVMs in containerised environments.

Prerequisites

- You should have attended our Advanced Java Programming course or have equivalent knowledge.
- You should have at least one year of practical experience developing Java applications.
- You should have strong proficiency in Java SE programming and some familiarity with Java concurrency basics.

Course Contents

Introduction to JVM Performance

- Overview of JVM internals and performance considerations.
- JVM architecture and execution model.
- The role of the JIT compiler.
- Garbage collection and memory management concepts.

- Performance measurement and diagnostics.

JVM Architecture and Execution Model

- JVM components and execution pipeline.
- Execution engines: interpreter vs JIT compilation.
- HotSpot architecture.
- Hot spot detection and dynamic compilation.
- C1 (client) and C2 (server) compilers.
- Tiered compilation.
- JVM code cache.
- JIT compilation logs and diagnostics.

JVM Implementations

- HotSpot vs OpenJ9.
- GraalVM and native image.
- Just-In-Time vs Ahead-Of-Time compilation.

Memory Management

- Java memory model overview.
- Heap structure and object allocation.
- Young generation: Eden and survivor spaces.
- Old (tenured) generation.
- Metaspace and class metadata.
- Thread stacks and stack memory.

Garbage Collection

- Stop-the-World (STW) events.
- Garbage collection algorithms: mark-sweep, copying and compaction.
- Generational garbage collection.
- Modern collectors: Serial, Parallel, G1, ZGC and Shenandoah.
- Choosing a garbage collector: latency vs throughput.

JVM Tuning

- Heap sizing and memory configuration.
- Tuning metaspace and thread stacks.
- Garbage collector selection and tuning.
- Tuning pause time goals and throughput.
- JIT tuning and compilation thresholds.
- On-Stack Replacement (OSR).
- Key JVM command line options.

Diagnostics and Performance Analysis

- Monitoring and measuring JVM performance.
- Analysing GC behaviour and logs.
- Diagnostic tools: jcmd, jstat, jmap, jconsole.
- Java Flight Recorder and VisualVM.
- Heap dump analysis.

Diagnosing Memory Leaks

- Understanding memory leaks in Java.
- Common causes: static collections, unclosed resources, ThreadLocal misuse.
- Detecting leaks using profiling tools.
- Using weak and soft references.
- Preventing leaks with proper lifecycle management.

Concurrency and Thread Performance

- Platform threads and thread scheduling.
- Thread pools and tuning strategies.
- Project Loom and virtual threads.
- Pinned virtual threads and performance implications.

Writing High-Performance Java Code

- Algorithmic complexity and real-world performance.
- Choosing appropriate data structures.
- Avoiding unnecessary object allocation.
- Object pooling: advantages and disadvantages.
- Data locality and CPU cache effects.
- The cost of boxing and unboxing.
- Microbenchmarking with JMH.

** *The lecturer reserves the right to modify the contents of the course to suit the needs of the delegates.*