Java Performance Tuning Workshop (EXL-2025)

Duration: 4 Days

What you will learn

In this Extreme Learning expert-led courses, you will learn a blend of tuning methodology, performance theory and practical tips on solving difficult performance problems. You will have an opportunity to hone your skills on a series of labs that are derived from real problems found during our consulting experience. The tools learned are all freely available or open source and will equip you to immediately apply what you have learned in your workplace. No more suffering through slow, under performing applications. You will learn how to develop code with performance as the goal and the end user in mind.

About Extreme Learning: Get taught directly by the experts and interact with peers in an engaging informal environment. These are intensive, deep dive training sessions designed for advanced IT professionals which incorporate expert-led presentations.

Students who can benefit from this course
Java developers and Quality Assurance engineers

Prerequisites

Required Prerequisites

Familiarity with the Java Enterprise Edition (Java EE).
Intermediate to advanced level knowledge of Java Standard Edition (Java SE) versions 1.6 and beyond.

Course Objectives

understand the importance of the user experience
find the tools needed to conduct a performance investigation
administer benchmarking and micro-benchmarking
understand Java memory and execution models

Course Topics

Introduction
defining performance
identifying causes of poor performance
introduction to aspects of performance tuning

Performance Tuning Methodology
introduction to The Box, an abstraction of a computer system
hardware's role in performance
JVM and operating systems role in performance applications role in performance
actors role in performance
performance testing workflow
using the dominate consumer to identify bottlenecks
role of hardware and JVM/OS role of application
role of tooling

**Time**
responsiveness
how to measure time
logging to monitor performance
common performance problems

**Memory**
System level monitoring
Memory management
Garbage collection algorithms
Heap tuning

**Performance Testing**
ojectives, types and components
differences between closed and open test harnesses
performance testing environment performance testing pitfalls
performance testing goals
GUI testing
web based performance testing
introduction to Apache JMeter
test doubles for performance

**Monitoring Hardware**
important components of hardware to monitor
hardware monitoring tools for CLM
Unix/Linux monitoring
introduction to vmstat
case studies of how different performance problems show up in vmstat
relating vmstat output to application code
Windows monitoring

**Java Memory Management**
process organization
JVM heap organization (Sun and IBM)
reference counting and tracing collectors
hemispheric and generational collectors
parallel and concurrent collectors
G1 collector and GC ergonomics
effects of OS and hardware on GC
switches affecting JVM memory management

**Object Life-cycles**
 some terminology
finialization
reference objects and collection rules
soft
weak
phantom
ReferenceQueue
WeakHashMap

**Monitoring Garbage Collection**
JVM flags for monitoring GC (Sun, IBM)
GC log record format
Secondary information hidden in GC logs
GC sequential overhead
GCViewer
HPJMeter

**Interprocess communications**
RMI tracing
socket tracing
JDBC Monitoring

**Threading**
Java support for threading
Synchronization explained
Synchronized statement and keyword
Producer consume with performance implications of choices
Little's Law
reducing lock contention
Java 5 locking optimizations

**Java Management eXtentions**
review of JMX based tools
Defining your own MBeans for performance monitoring
defining an mbean
adding attributes and operations
hooking an mbean into the notification framework
AttributeBundles

**Profilers**
role of profiling
profiling techniques
Execution, memory, and thread profiling
Tools: prof, hprof, HPJMeter, NetBeans Profiler
when to use which profiler
effects of Object pooling
diagnosing memory problems: loitering object and leaking object
permspace leaks

**Performance Tactics**
effects of human perception
tuning strategy
influences on performance
latency
CPU, I/O, memory bottlenecks
garbage collection tactics
heap sizing
The Box as a profiling guide
Benchmarking
- macro and microbenchmarks
- dangers of benchmarking
- accounting for interfering factors
- developing a micro-benchmark
- performance tuning a benchmark
- benchmark validation
- macro-benchmarking
- test harness validation

Collections
- coding style for custom implied collections
- Review of standard 1.5 collection types
- introduction to non-blocking collections

Serialization
- purpose and performance implications
- review of serialization
- optimizing serialization