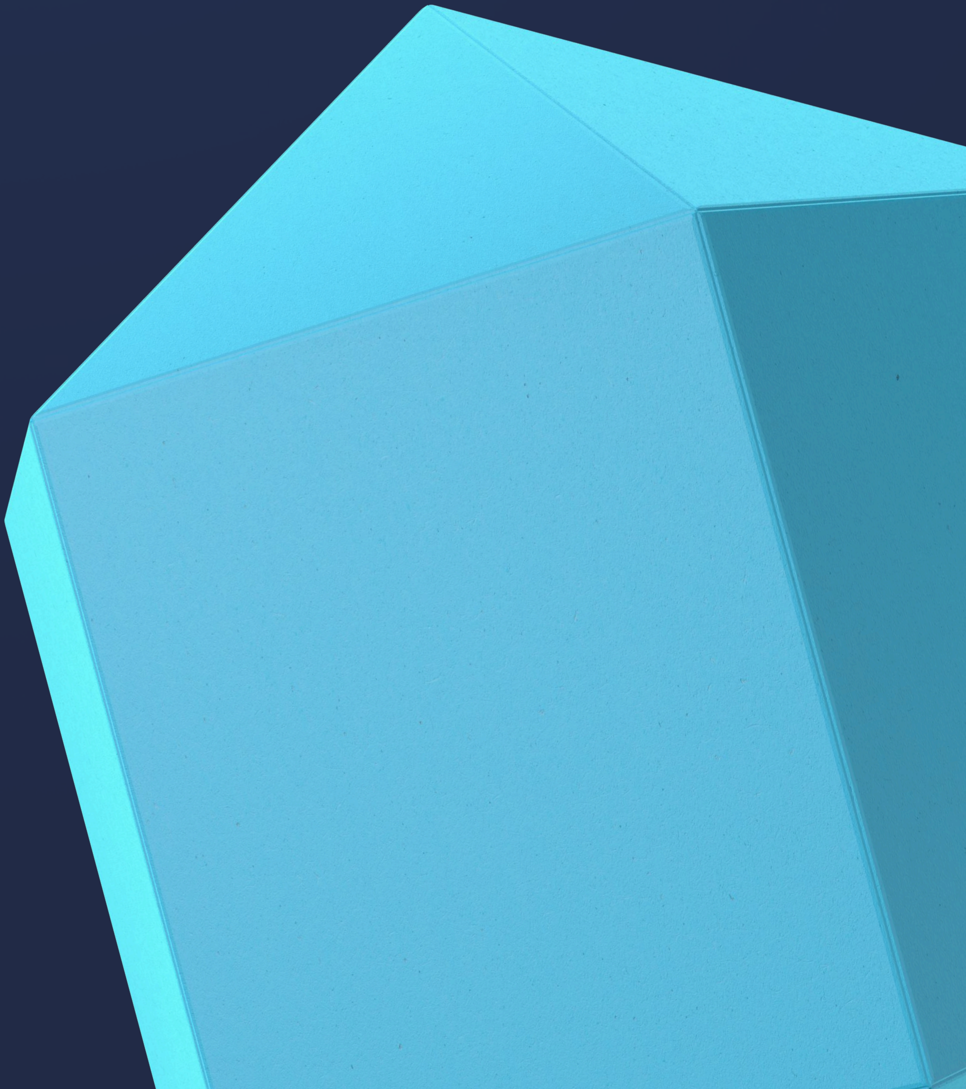
bell/soft | Liberica JDK Performance Edition

# Liberica JDK Performance Edition

INCREASE THE PERFORMANCE OF YOUR JAVA™ 8 & 11 WORKLOADS WITHOUT MIGRATION

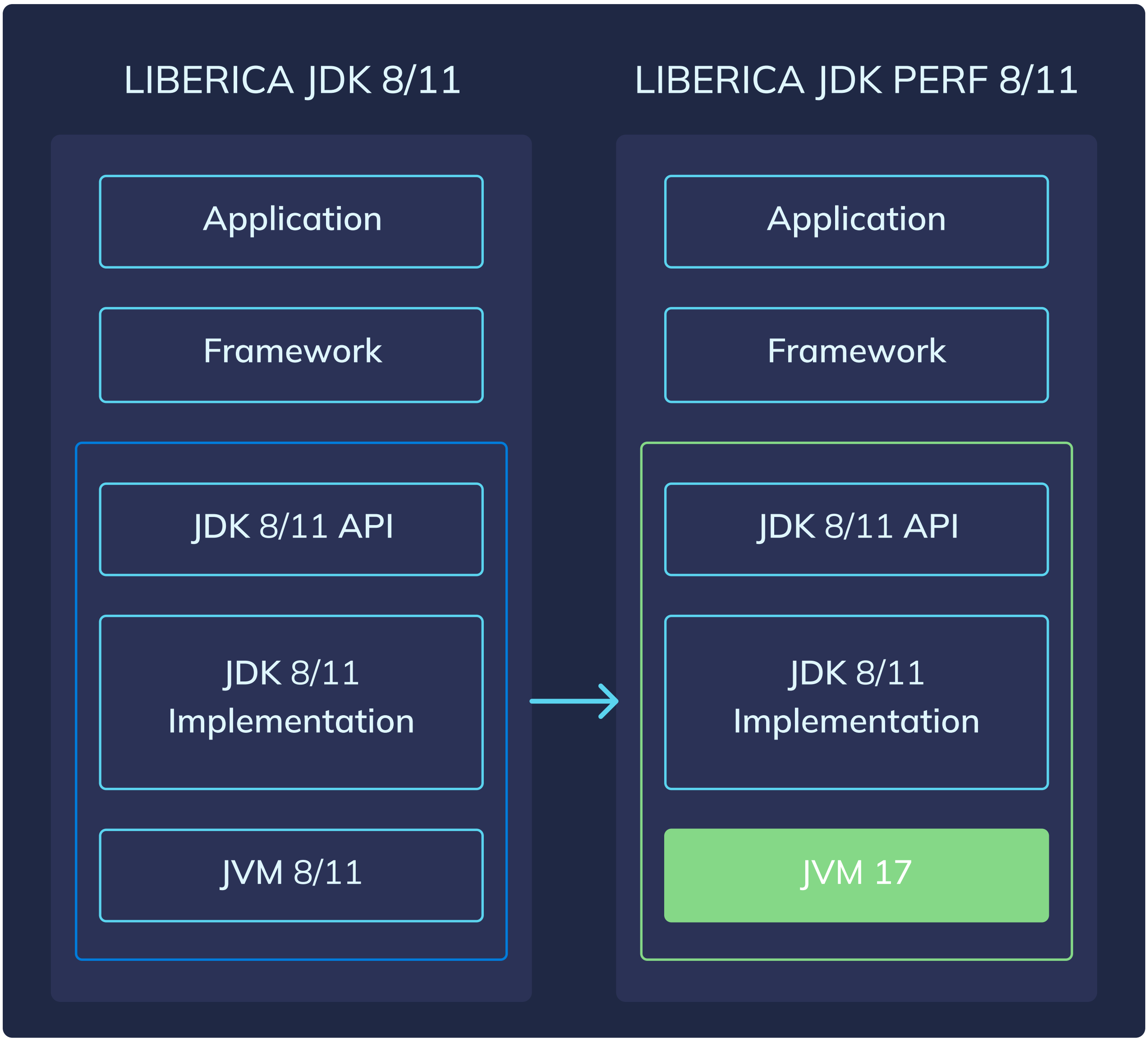


JULY 2024

# Liberica JDK Performance Edition

## INCREASE THE PERFORMANCE OF YOUR JAVA™ 8 & 11 WORKLOADS WITHOUT MIGRATING TO A NEWER VERSION OF JAVA™

Liberica JDK Performance Edition (liberica-perf for short) is a Java runtime that integrates HotSpot JVM 17 into the builds of JDK 8 or 11. The JVM includes multiple improvements introduced up to Java version 17, with a primary focus on garbage collection enhancements. These modifications have a considerable impact on optimizing application throughput, latency, and startup.


LIBERICA JDK 8/11

Application

Framework

JDK 8/11 API

JDK 8/11 Implementation

JVM 8/11

LIBERICA JDK PERF 8/11

Application

Framework

JDK 8/11 API

JDK 8/11 Implementation

JVM 17

Liberica JDK Performance Edition serves as a drop-in replacement for Java SE 11 or 8. Most of the APIs and features from JDK 8 or 11 remain the same, meaning that your developers don’t have to rewrite the application and update dependencies. Updating one element of your stack enables you to maintain compatibility and benefit from the enhancements of Java 17.

As a result, you will notice instant performance improvement even with the default liberica-perf settings without major code refactoring:

## For workloads running on x86_64 machines:

| JDK 8 | JDK 11 |
|---|---|
| –20% faster application response | –9.7% faster application response |
| –60% shorter G1 GC pauses | –77% shorter G1 GC pauses |
| –32% less RAM needed with G1 GC | –10% less RAM needed with G1 GC |
| –5.8% faster startup | –9% faster startup |

## For workloads running on AArch64 architecture:

| JDK 8 | JDK 11 |
|---|---|
| –16.9% faster application response | –10% faster application response |
| –70% shorter G1 GC pauses | –46% shorter G1 GC pauses |
| –12% less RAM needed with G1 GC | –25% less RAM needed with G1 GC |
| –7.7% faster startup | –7.6% faster startup |

# Which Enterprises Will Benefit from Liberica JDK Performance Edition

JDK 8 WAS RELEASED IN 2014, AND JDK 11 – IN 2018. THE JAVA PLATFORM RECEIVED MANY ENHANCEMENTS SINCE THEN, BUT AS JDK 8 AND 11 ARE IN DEEP MAINTENANCE, ONLY FEW IMPROVEMENTS ARE BACKPORTED TO THE RESPECTIVE UPDATE PROJECTS.

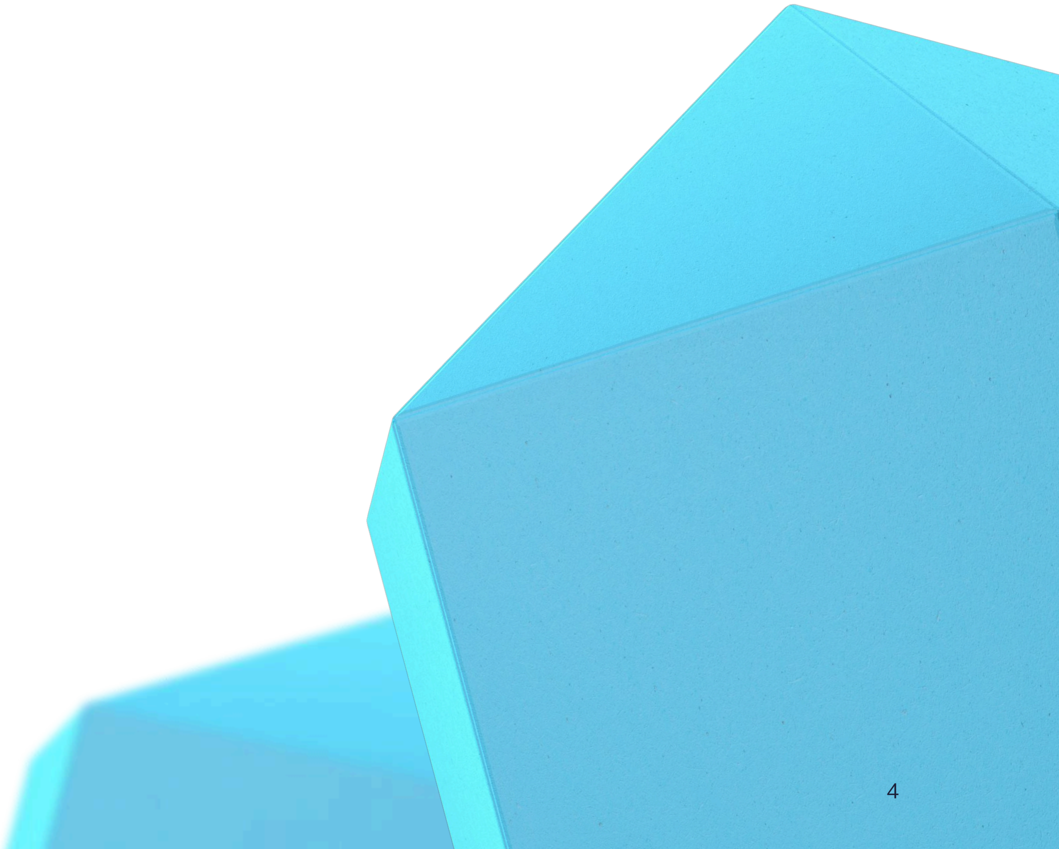
**As a result, running workloads on JDK 8 or 11 is associated with two major drawbacks:**

**Higher IT costs:** Java 8 & 11 require more resources, translating to increased expenses compared to Java 17 and 21 for completing the same tasks.

**User dissatisfaction:** Applications built on JDK 8 & 11 struggle to keep pace with user expectations due to increased latency and decreased throughput.

Upgrading to a newer JDK version can solve these issues. But migration is a laborious task requiring developers to rewrite the application code, upgrade dependencies, and solve compatibility issues. For companies who are not ready to upgrade Java right away, BellSoft created Liberica JDK Performance Edition.

**Liberica JDK Performance Edition will help companies with a development environment based on JDK 8 or 11 to increase essential KPIs without rewriting the application code or changing framework and library versions.**



# Features and Enhancements of Liberica JDK Performance Edition

LIBERICA JDK PERFORMANCE EDITION COUPLES JVM 17 AND JDK 8 OR 11 AND CONTAINS MULTIPLE IMPROVEMENTS INTRODUCED TO JAVA VIRTUAL MACHINE UP TO VERSION 17.

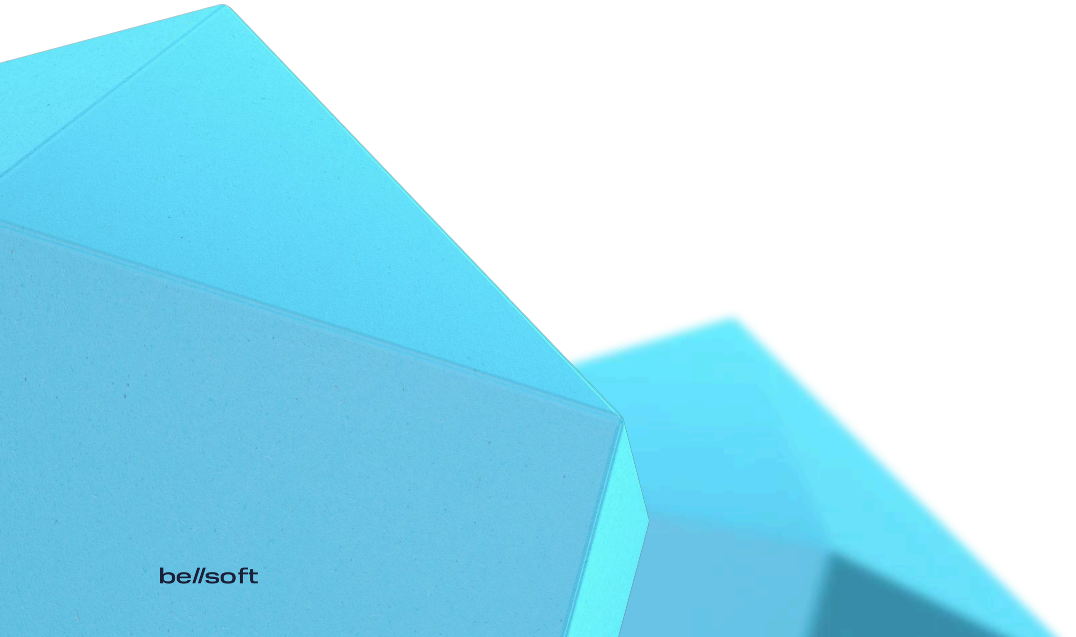
**Liberica JDK Performance Edition 8 includes the following new features:**

- **Z Garbage Collector** (new in JDK 8, improved in JDK 11): a scalable, low latency garbage collector.
- **Compact Strings** (new in JDK 8): a space-efficient internal representation of strings, which reduces memory footprint and garbage collection activity; it's enabled by default.
- **Unified JVM Logging** (new in JDK 8): replaces JDK options that print details about the JVM with `-Xlog` options.

**In addition, Liberica JDK Performance Edition 8 and 11 include the following enhancements:**

- **Garbage-First (G1) Garbage Collector:** targeted for multiprocessor machines scaling to a large amount of memory. This is the default garbage collector for all versions of Liberica JDK Performance Edition.
- **G1 String Deduplication:** reduces the memory footprint of String objects on the Java heap by taking advantage of the fact that many String objects are identical. It's disabled by default, but you can enable it with the `-XX:+UseStringDeduplication` option.
- **Class Data Sharing (CDS):** helps reduce the startup time and memory footprint between multiple JVMs. It's enabled by default in Liberica JDK Performance Edition. To disable it, see Manually Controlling Class Data Sharing.
- **Enhanced JDK Flight Recorder:** a tool for collecting diagnostic and profiling data for a running Java application.

*Note that Concurrent Mark Sweep Garbage Collector is absent from liberica-perf 8 and 11.*



# Performance Studies of Liberica JDK Performance Edition

WE VALIDATED THE PERFORMANCE OF LIBERICA JDK PERFORMANCE EDITION USING THE SPRING PETCLINIC APPLICATION AND LIBERICA JDK 11 AND 8 STANDARD FOR A COMPARISON.

**Experimental setup:**

**CPU:**
Intel(R) Xeon(R) CPU
X5675 @ 3.07GHz

**Memory:**
96GB

**OS:**
Linux Ubuntu
22.04.1

**For both liberica-perf 8 and 11, we used the SpecJBB benchmark for throughput evaluation. This benchmark measures the throughput of an application with two metrics:**

**max-jOPS**
represents the maximum transaction throughput of a system until failure;

**critical-jOPS**
is the mean stable transaction throughput in the long term.

Detailed description of studies specific to the liberica-perf version, as well as study results can be found in the corresponding sections below.

## LIBERICA JDK PERFORMANCE EDITION 11 STUDIES

To measure GC evacuation pause affecting application latency, we tested G1GC (a default GC in Java 11) with the BigRamTester benchmark. The tests evaluated G1 Evacuation Pause (where live objects are copied from one region into another), when all application threads are stopped for the period of garbage collection.

**The results of the studies demonstrate:**

**Shorter G1GC evacuation pause**

–96%

**Higher mean throughput**

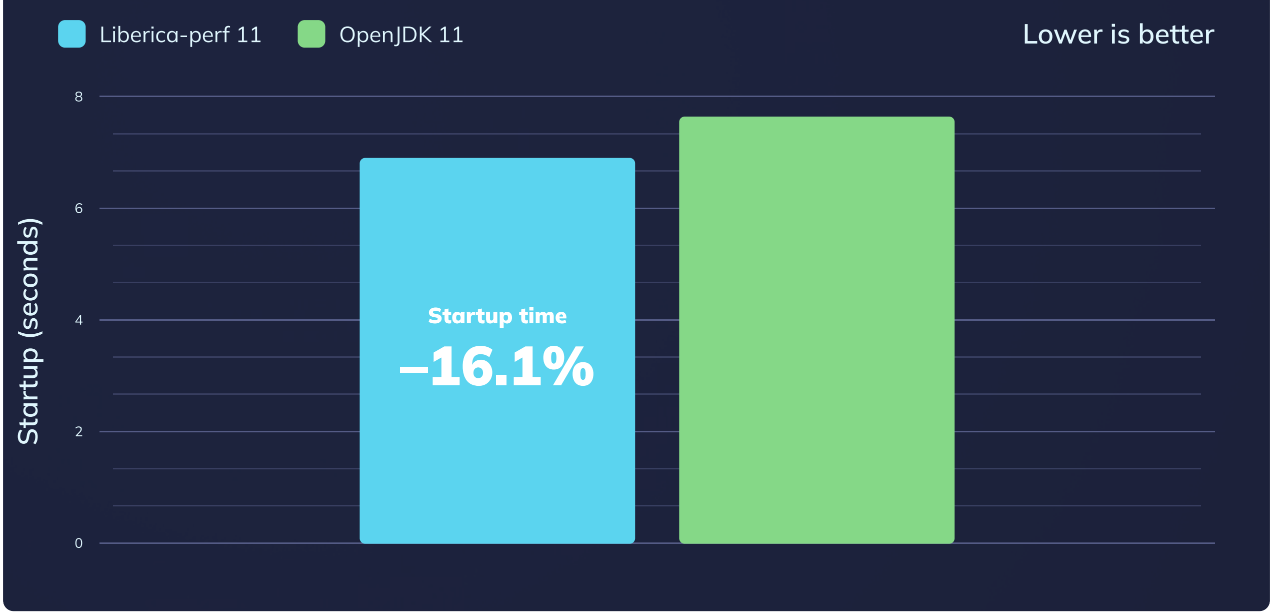
+120%

**Faster startup**

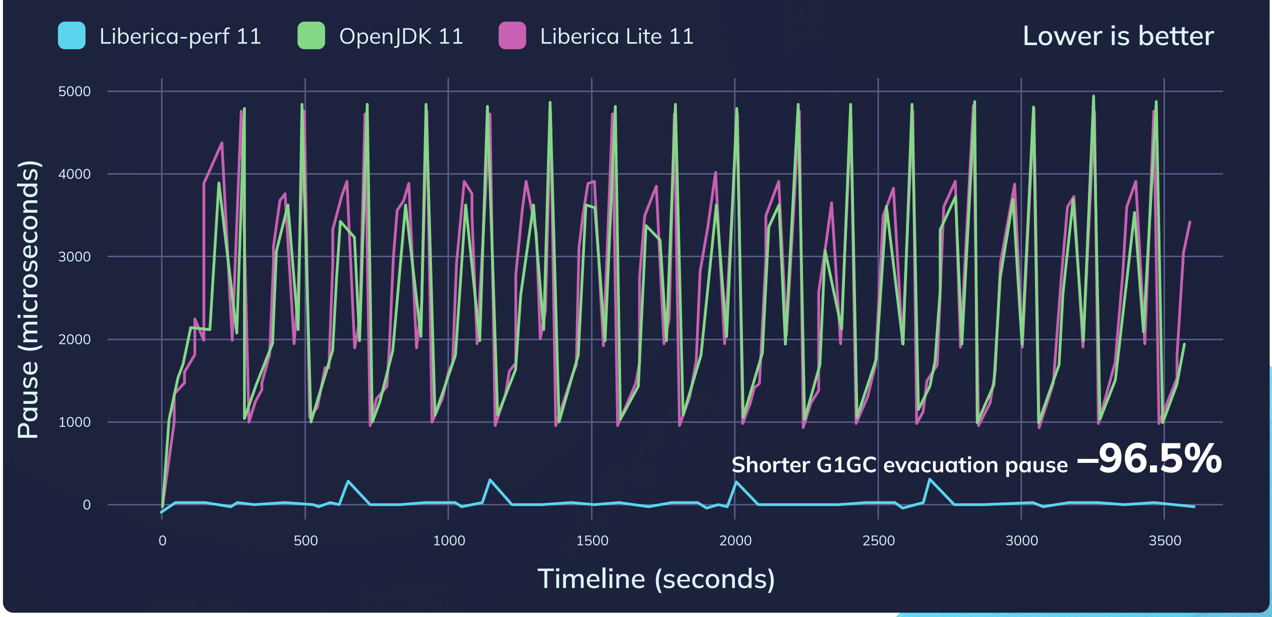
–16%

The graphs below demonstrate the results of startup, G1GC evacuation pause, and throughput studies.

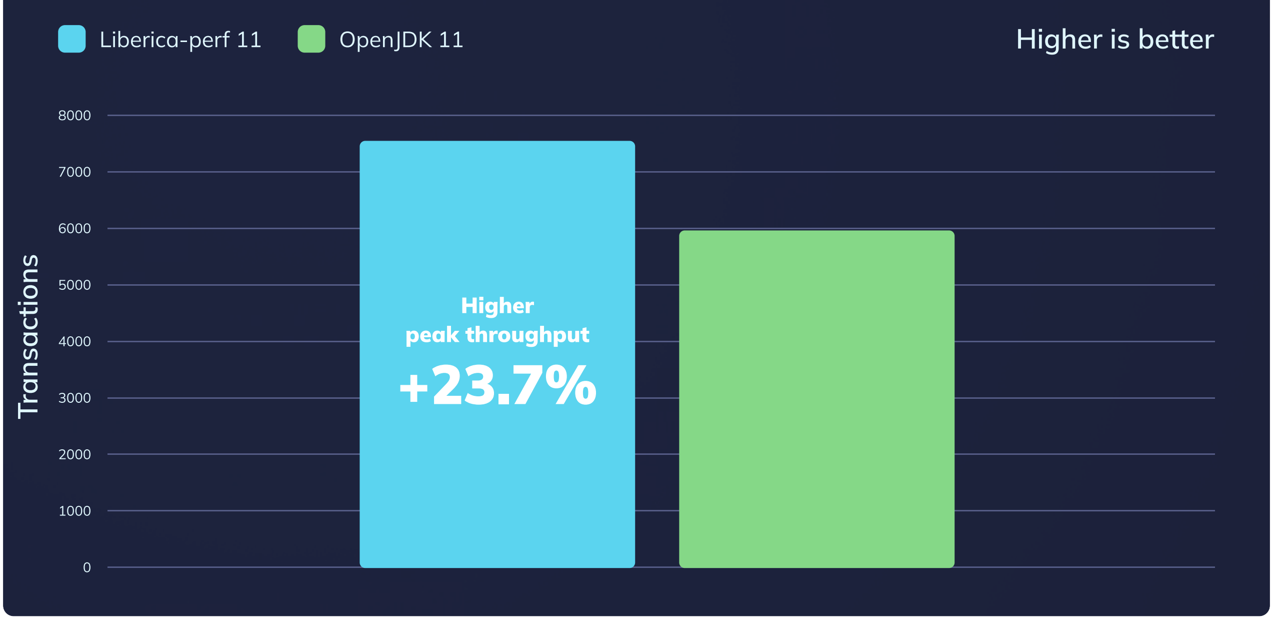
## PetClinic startup



## GC performance study: G1 Evacuation Pause



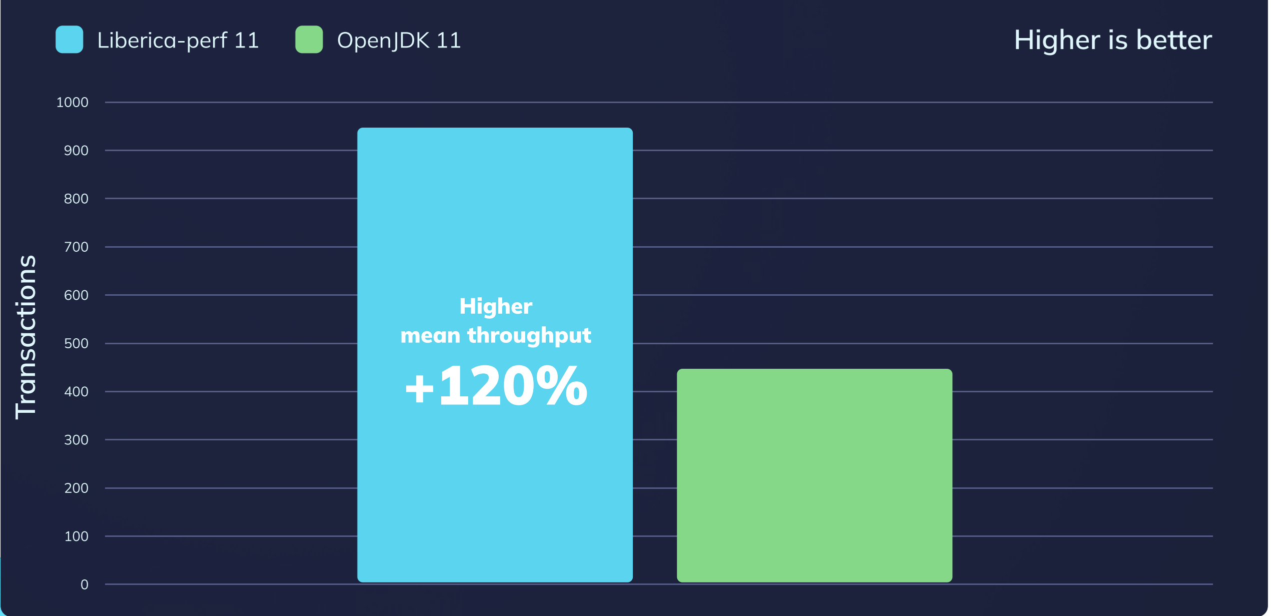
## Peak throughput (SpecJBB max-jOPS)



Higher peak throughput

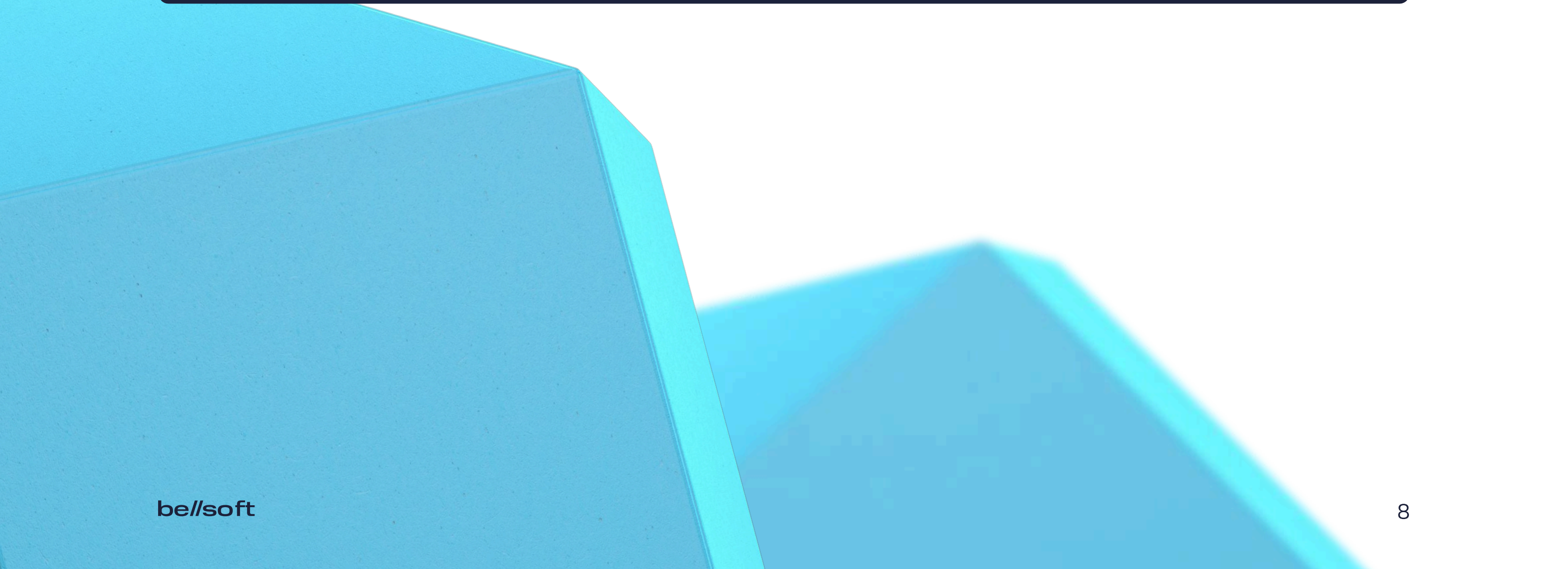
+23.7%

## Mean throughput (SpecJBB critical-jOPS)



Higher mean throughput

+120%



## LIBERICA JDK PERFORMANCE EDITION 8 STUDIES

When studying throughput and latency, we used two GC implementations: G1GC and ZGC. ZGC is a scalable low-latency garbage collector absent in OpenJDK 8. We measured latency using the BigRamTester benchmark (1h run with 50 GB).

To measure the overall performance, we used the DaCapo benchmark, a set of real-world Java applications with different memory loads.

The study of compression/decompression speed was done using two datasets, Large Calgary Corpus and Silesia Corpus.

**The results of the studies demonstrate:**

**Higher compression speed**

+85%

**Higher decompression speed**

+113%

**Higher mean throughput with G1GC**

+50%

**Higher peak throughput with ZGC**

+200%

**Lower latency with ZGC**

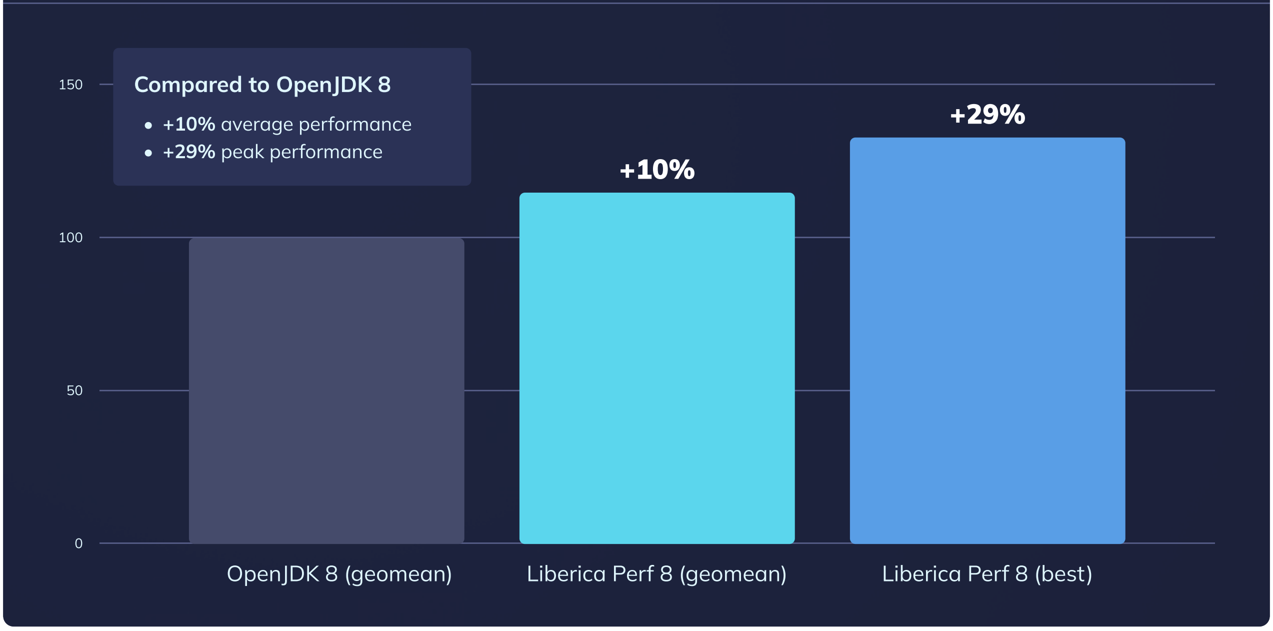
–1,000%

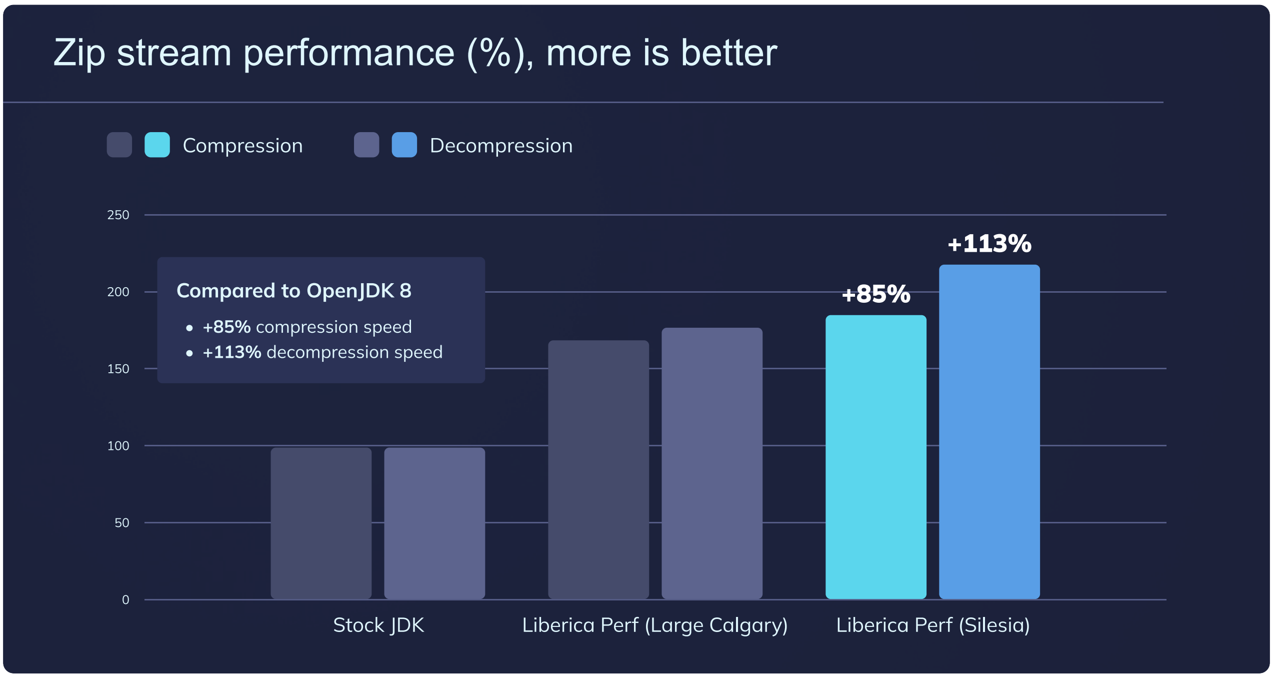
**Better average performance**

+10%

The graphs below demonstrate the results of throughput, latency, compression/decompression speed, and overall performance studies.

Realworld application performance - Dacapo (%), more is better

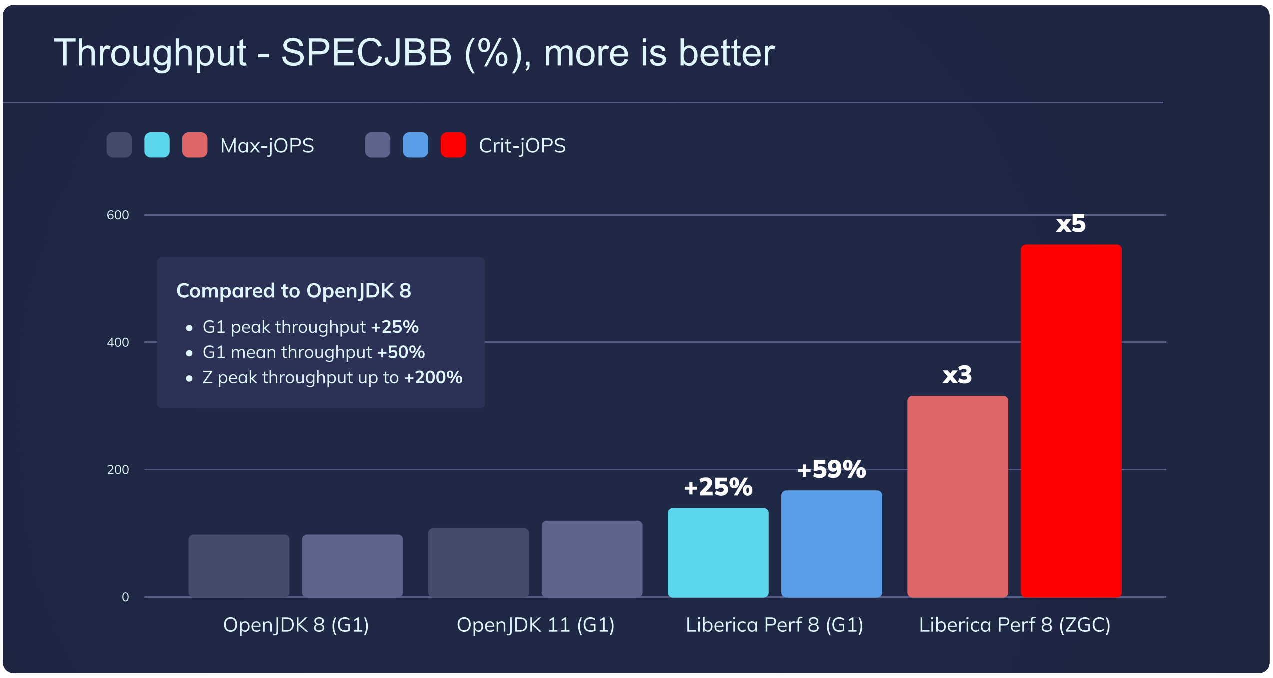




## Zip stream performance (%), more is better

**Compared to OpenJDK 8**

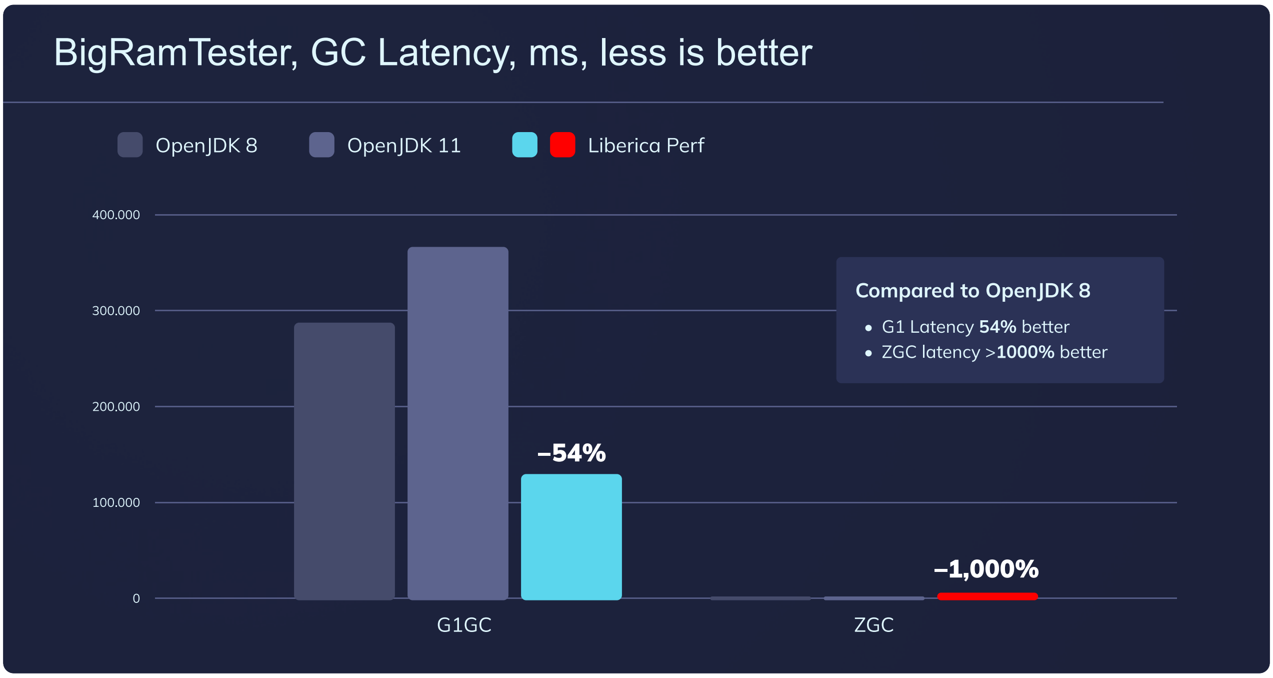
- **+85%** compression speed
- **+113%** decompression speed



## Throughput - SPECJBB (%), more is better

**Compared to OpenJDK 8**

- G1 peak throughput **+25%**
- G1 mean throughput **+50%**
- Z peak throughput up to **+200%**



## BigRamTester, GC Latency, ms, less is better

**Compared to OpenJDK 8**

- G1 Latency **54%** better
- ZGC latency >**1000%** better

## INCREASED PERFORMANCE OF COMPRESSION / DECOMPRESSION OPERATIONS WITH ALPAQUITA LINUX

Liberica JDK Performance Edition includes zlib-ng, a zlib data compression library for the next generation systems. Coupled with Alpaquita, a lightweight Linux distribution with many optimizations, it takes the performance of compression/decompression operations to a new level as compared to Alpine Linux that includes OpenJDK packages with a standard zlib:

**+85%** faster compression than OpenJDK 11/Alpine Linux 3.17

**+106%** faster decompression than OpenJDK 11/Alpine Linux 3.17

**Zlib-ng performance measurement was done with:**

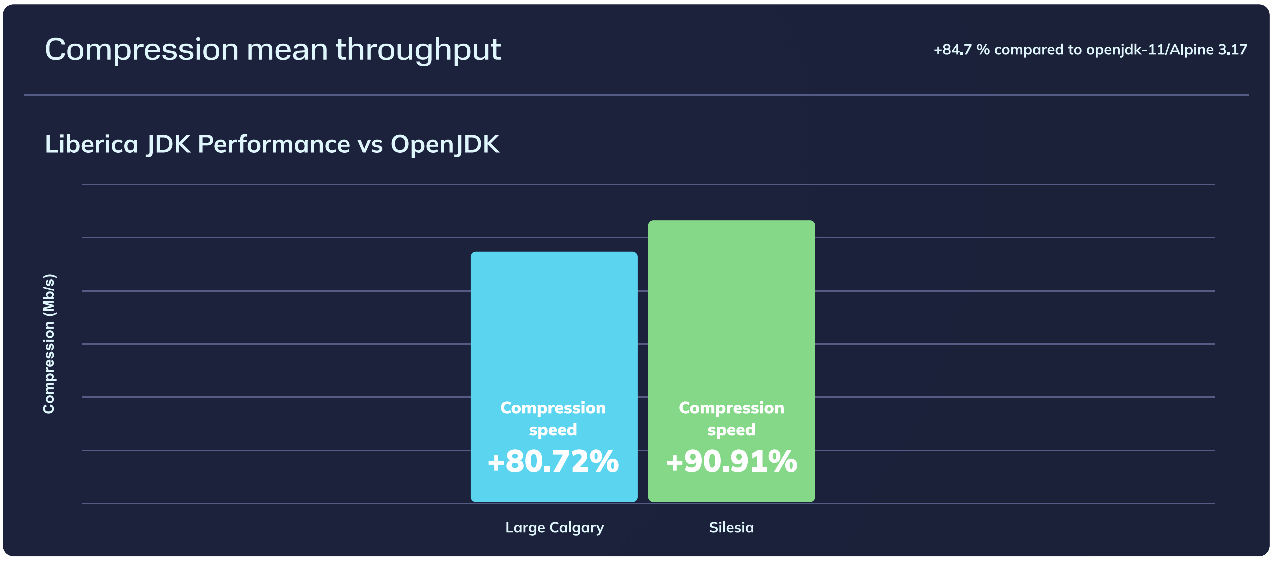
**CPU:**
Intel(R) Core(TM) i5-6600 CPU @ 3.30GHz

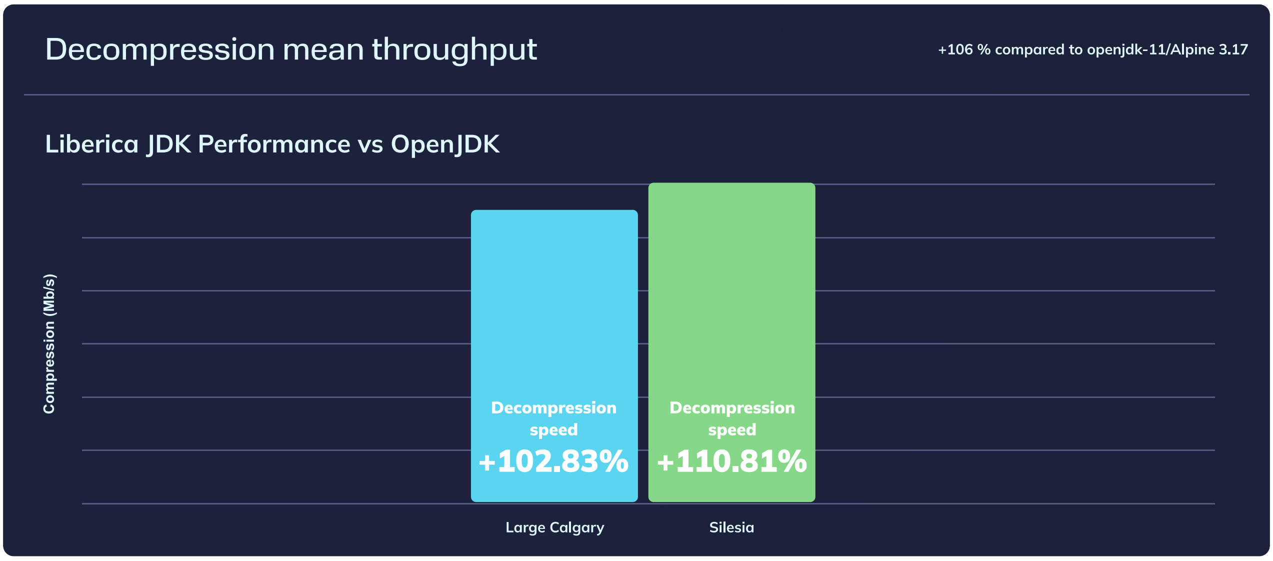
**Memory:**
32GB

**OS images:**
Alpaquita 23, Alpine 3.17

**Data sets:**
Large Calgary Corpus and Silesia Corpus

The graphs below demonstrate the results of compression/decompression rate studies.





# Installing Liberica JDK Performance Edition

STEP 1

### DEVELOP A MIGRATION PLAN

Consult Appendix I to compile a thorough inventory of JVM options and features introduced or removed in JDK 17. Use this information to create a detailed list of components requiring replacement.

STEP 2

### INSTALL LIBERICA JDK PERFORMANCE EDITION

Download the tar.gz package containing a Liberica JDK Performance Edition bundle. Install it as you would any other Java version.

STEP 3

### TEST YOUR APPLICATION

Run tests to ensure a smooth operation. If you encounter any issues, reach out to our support team.

STEP 4

### OPTIMIZE JVM AND RESOURCE UTILIZATION

Fine-tune JVM settings to reduce resource consumption or enhance the performance of your JDK 8 & 11-based services. Our engineers are committed to helping you optimize the performance of your services and minimize resource usage.

# Support and Pricing

You can use liberica-perf for developing and running Java applications on a headless or GUI system on Linux. The builds are supported on Intel and ARM64 processors.

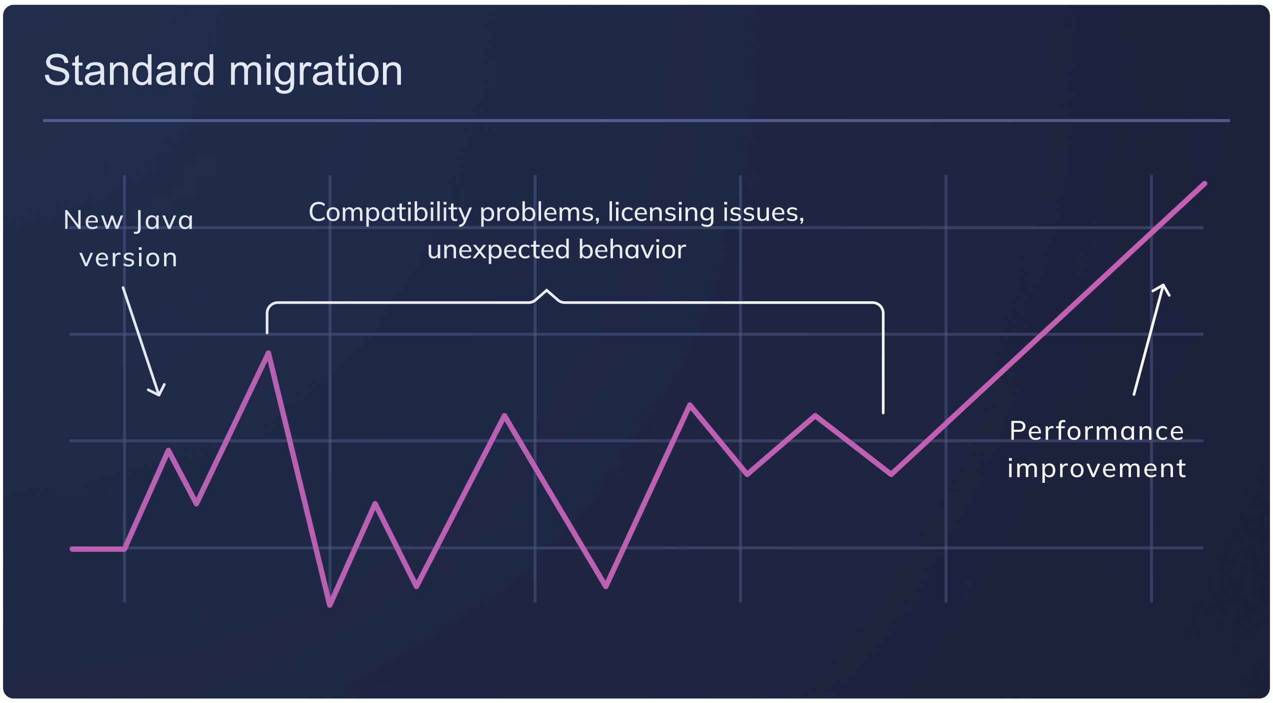
Quarterly updates and long-term support for Liberica JDK Performance Edition is in line with Liberica JDK Standard. Receiving regular security updates and fixes for JVM 17 and JDK 8 or 11 will help you keep your Java runtime stable and secure at all times.

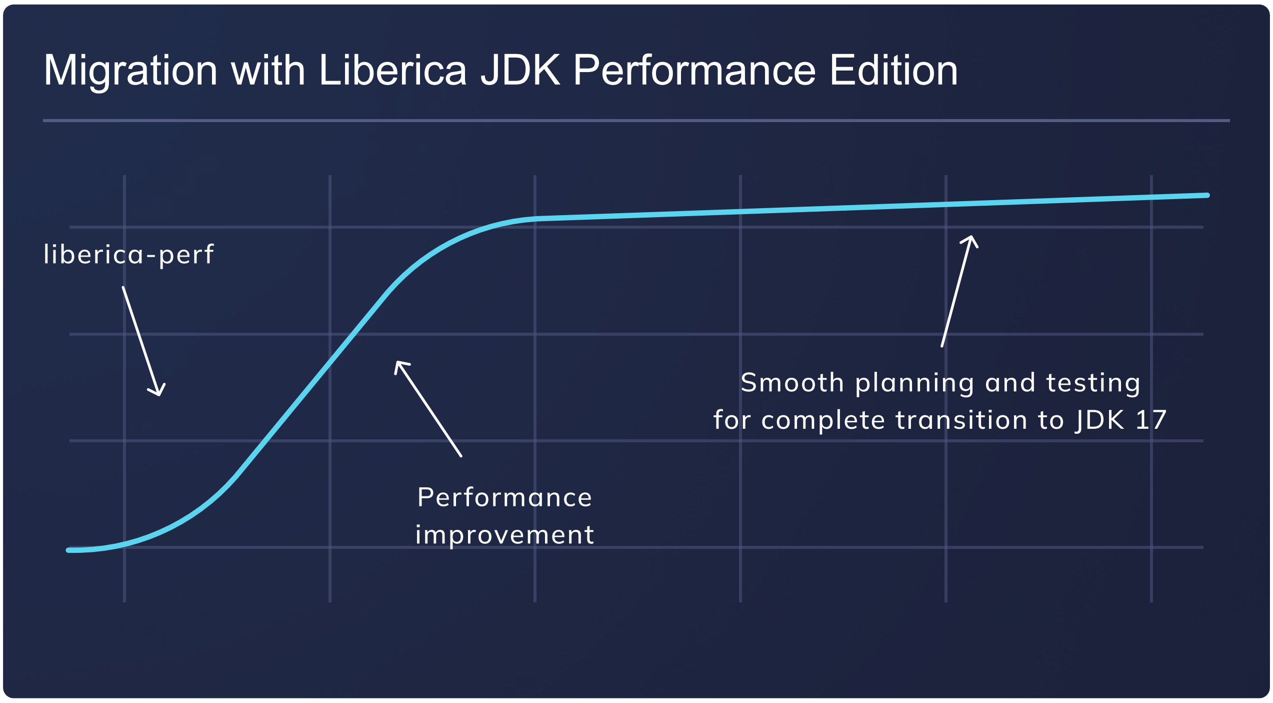
You can learn more about support plans and roadmap for Liberica JDK here. In addition, BellSoft provides a Software Bill of Materials for all Liberica JDK and Liberica JDK Performance Edition builds.

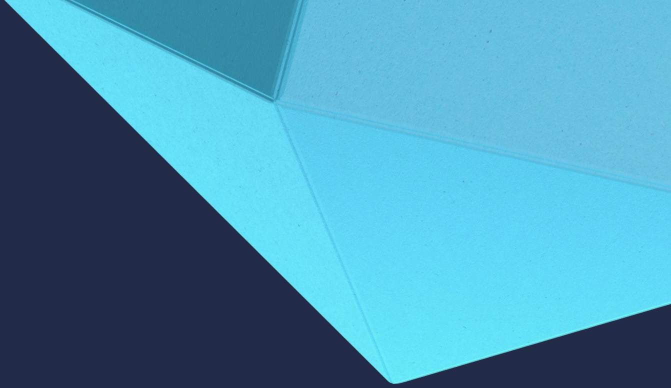
Liberica JDK Performance Edition is included in the Liberica JDK Subscription together with other solutions for Java development. Please contact our sales team to ask for a quote.

# Enjoy the Performance Boost Now and Make Migration to JDK 17 in the Future Easier

Integration of Liberica JDK Performance Edition will make subsequent migration to JDK 17 easier because you can strategically plan the steps to finalize the version upgrade at your preferred pace. This may involve migrating less critical services, conducting tests, identifying necessary library versions for JDK 17, and more.







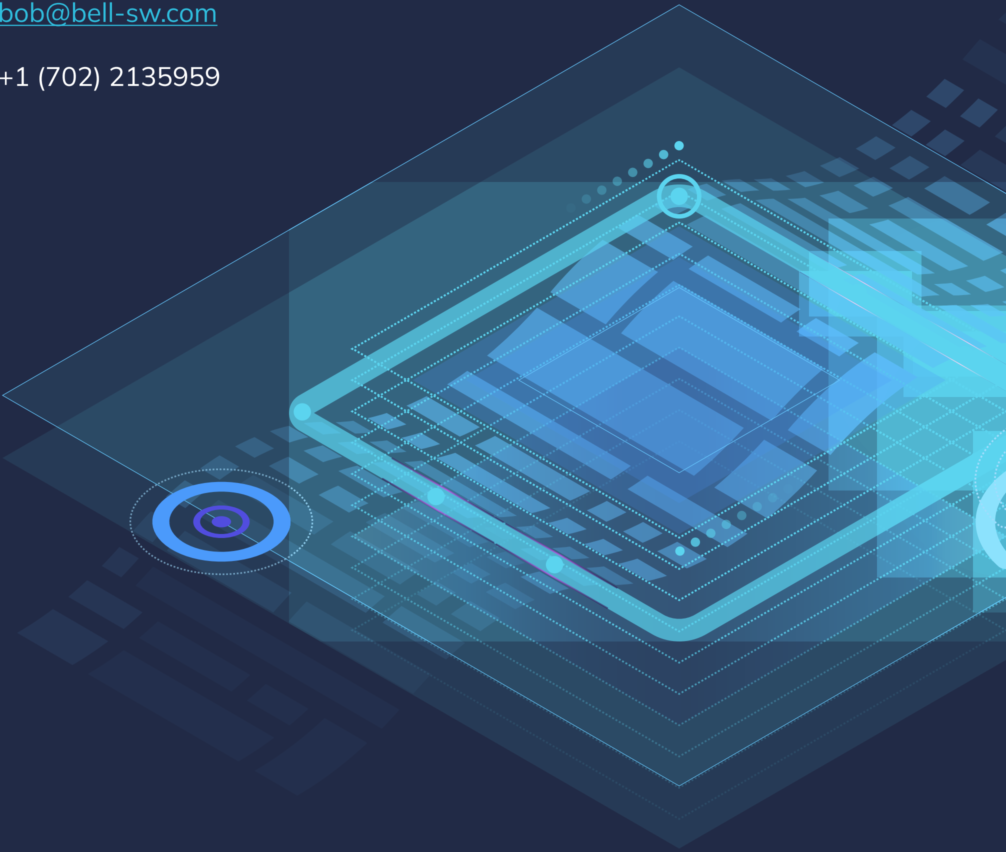
bell/soft

# CONTACT US TODAY

Have more questions about Liberica JDK, our other products, or enterprise support plans? Our sales representative, Bob Boshehri, will provide you with the assistance you need.

Feel free to reach out to Bob using the contact details below or schedule a meeting with him.

Bob Booshehri
Java Expert Group
bob@bell-sw.com
+1 (702) 2135959



# Appendix I. Changes to Some Tools, Libraries, and JVM Runtime Options in Liberica JDK Performance Edition

Most workloads can be migrated to the new functionality without significant changes, but in some cases, additional configuration is required.

Tools and libraries that are not supported on or behave differently in Liberica JDK Performance Edition:

- JVM compiler interface (JVMCI) is not supported, as well as Graal JIT and AOT that depend on JVMCI. GraalVM is now developed as a separate project. Builds for JDK 11 are available as part of GraalVM CE. You can also use a GraalVM CE-based Liberica Native Image Kit as a native-image compiler.
- Concurrent Mark Sweep Garbage Collector was removed from JDK 14 (JEP 363) and therefore is not supported.
- JFR events are based on JVM 17 capabilities. Minor changes in the amount and format of supported events are expected.
- Some VM log messages were changed. More logging categories are available via the `-Xlog` option (part of JVM 17).
- Only the Server VM is available.

In addition, several JVM runtime options were added from JDK 17, some were removed.
See the tables below for additional information.

## CHANGES TO RUNTIME OPTIONS FOR JDK 11

### List of added runtime options

Below is the list of options available in liberica-perf 11, but absent in JDK 11. For more information on these parameters, see the official documentation.

| New options in liberica-perf 11 |
|---|
| AdjustStackSizeForTLS |
| AllowRedefinitionToAddDeleteMethods |
| ArchiveClassesAtExit |
| AsyncLogBufferSize |
| C1InlineStackLimit |
| C1MaxInlineLevel |

| |
|---|
| C1MaxInlineSize |
| C1MaxRecursiveInlineLevel |
| C1MaxTrivialSize |
| CompilationMode |
| DynamicDumpSharedSpaces |
| G1PeriodicGCInterval |
| G1PeriodicGCInvokesConcurrent |
| G1PeriodicGCSystemLoadThreshold |
| G1RSetRegionEntries |
| G1RSetSparseRegionEntries |
| HeapDumpGzipLevel |
| MetaspaceReclaimPolicy |
| MinHeapSize |
| RecordDynamicDumpInfo |
| ShowCodeDetailsInExceptionMessages |
| SweeperThreshold |
| UseContainerCpuShares |
| UseEmptySlotsInSupers |
| UseNotificationThread |
| ZMarkStackSpaceLimit |
| ZProactive |
| ZUncommit |
| ZUncommitDelay |

### List of removed and renamed runtime options

Below is the list of options not available in liberica-perf 11 and the options that were renamed.

| JDK 11 options | Resolution in liberica-perf 11 |
|---|---|
| AggressiveOpts | Removed |
| AllowJNIEnvProxy | Removed |
| AllowNonVirtualCalls | Removed |
| AssertOnSuspendWaitFailure | Removed |
| AssumeMP | Removed |
| BindGCTaskThreadsToCPUs | Removed |
| BranchOnRegister | Removed |

| | |
|---|---|
| BytecodeVerificationLocal | Moved to the diagnostic flags category. Use with `-XX:+UnlockDiagnosticVMOptions` |
| BytecodeVerificationRemote | Moved to the diagnostic flags category. Use with `-XX:+UnlockDiagnosticVMOptions` |
| CMS* | CMS GC is not supported in liberica-perf 11 |
| CalculateClassFingerprint | AOT is not supported in liberica-perf 11 |
| CompactFields | Removed |
| CompilationPolicyChoice | Removed |
| CompilerThreadHintNoPreempt | Removed |
| Debugging | Removed |
| DeoptimizeRandom | Moved to the non-product category. Not available in the release build. |
| ErrorReportServer | Removed |
| FLSAlwaysCoalesceLarge | CMS GC is not supported in liberica-perf 11 |
| FLSCoalescePolicy | CMS GC is not supported in liberica-perf 11 |
| FLSLargestBlockCoalesceProximity | CMS GC is not supported in liberica-perf 11 |
| FailOverToOldVerifier | Removed |
| FieldsAllocationStyle | Removed |
| ForceNUMA | Removed |
| G1RSetRegionEntries | Removed |
| G1RSetScanBlockSize | Removed |
| G1RSetSparseRegionEntries | Removed |
| GCLockerInvokesConcurrent | Removed |
| GCTaskTimeStampEntries | Removed |
| InitialBootClassLoaderMetaspaceSize | Removed |
| InsertMemBarAfterArraycopy | Removed |
| LIRFillDelaySlots | Removed |
| MonitorBound | Removed |
| MonitorInUseLists | Removed |
| NeedsDeoptSuspend | Removed |
| OldPLABWeight | Removed |
| ParGCDesiredObjsFromOverflowList | Removed |
| ParGCTrimOverflow | Removed |
| ParGCUseLocalOverflow | Removed |
| PrintJNIResolving | Functionality moved to unified logging. Use `-Xlog:jni+resolve` instead |

| | |
|---|---|
| PrintSafepointStatistics | Functionality moved to unified logging.<br>Use `-Xlog:safepoint+stats` instead |
| PrintSafepointStatisticsCount | Functionality moved to unified logging.<br>Use `-Xlog:safepoint+stats` instead |
| PrintSafepointStatisticsTimeout | Functionality moved to unified logging.<br>Use `-Xlog:safepoint+stats` instead |
| PrintVMQWaitTime | Removed |
| ProfileIntervals | Removed |
| ProfileIntervalsTicks | Removed |
| ProfileVM | Removed |
| ProfilerPrintByteCodeStatistics | Removed |
| ProfilerRecordPC | Removed |
| ResizeOldPLAB | Removed |
| ShenandoahSoftMaxHeapSize | Known as `SoftMaxHeapSize` in liberica-perf 11 |
| StressLdcRewrite | Moved to the diagnostic flags category.<br>Use with `-XX:+UnlockDiagnosticVMOptions` |
| SuspendRetryCount | Removed |
| SuspendRetryDelay | Removed |
| ThreadLocalHandshakes | Removed |
| Tier3AOTBackEdgeThreshold | AOT is not supported in liberica-perf 11 |
| Tier3AOTCompileThreshold | AOT is not supported in liberica-perf 11 |
| Tier3AOTInvocationThreshold | AOT is not supported in liberica-perf 11 |
| Tier3AOTMinInvocationThreshold | AOT is not supported in liberica-perf 11 |
| TraceSuspendWaitFailures | Removed |
| TransmitErrorReport | Removed |
| UnlinkSymbolsALot | Removed |
| UseAdaptiveGCBoundary | Removed |
| UseCMSBestFit | CMS GC is not supported in liberica-perf 11 |
| UseCMSInitiatingOccupancyOnly | CMS GC is not supported in liberica-perf 11 |
| UseConcMarkSweepGC | CMS GC is not supported in liberica-perf 11 |
| UseGCTaskAffinity | Removed |
| UseLWPSynchronization | Removed |
| UseLargePagesInMetaspace | Removed |
| UseLegacyJNINameEscaping | Removed |
| UseMembar | Removed |
| UseOSErrorReporting | Removed |
| UseRDPCForConstantTableBase | Removed |

| | |
|---|---|
| VMThreadHintNoPreempt | Removed |
| VerifyMergedCPBytecodes | Removed |
| ZMarkStacksMax | Removed |
| Zpath | Removed |
| ZStallOnOutOfMemory | Removed |
| ZStatisticsInterval | Removed |

## CHANGES TO RUNTIME OPTIONS FOR JDK 8

### List of added runtime options

Below is the list of options available in liberica-perf 8, but absent in JDK 8.

| New options in liberica-perf 8 |
|---|
| AdjustStackSizeForTLS |
| AllocateHeapAt |
| AllowRedefinitionToAddDeleteMethods |
| AllowVectorizeOnDemand |
| ArchiveClassesAtExit |
| ArrayCopyLoadStoreMaxElem |
| AsyncLogBufferSize |
| C1InlineStackLimit |
| C1MaxInlineLevel |
| C1MaxInlineSize |
| C1MaxRecursiveInlineLevel |
| C1MaxTrivialSize |
| CompactStrings |
| CompilationMode |
| CompileThresholdScaling |
| CreateCoredumpOnCrash |
| DoReserveCopyInSuperWord |
| DynamicDumpSharedSpaces |
| EnableDynamicAgentLoading |
| ErrorLogTimeout |
| ExecutingUnitTests |
| ExtensiveErrorReports |
| G1PeriodicGCInterval |

AdjustStackSizeForTLS

AllocateHeapAt

AllowRedefinitionToAddDeleteMethods

AllowVectorizeOnDemand

ArchiveClassesAtExit

ArrayCopyLoadStoreMaxElem

AsyncLogBufferSize

C1InlineStackLimit

C1MaxInlineLevel

C1MaxInlineSize

C1MaxRecursiveInlineLevel

C1MaxTrivialSize

CompactStrings

CompilationMode

CompileThresholdScaling

CreateCoredumpOnCrash

DoReserveCopyInSuperWord

DynamicDumpSharedSpaces

EnableDynamicAgentLoading

ErrorLogTimeout

ExecutingUnitTests

ExtensiveErrorReports

G1PeriodicGCInterval

G1PeriodicGCInvokesConcurrent

G1PeriodicGCSystemLoadThreshold

G1UseAdaptiveIHOP

HeapDumpGzipLevel

HeapSearchSteps

LoopPercentProfileLimit

LoopStripMiningIter

LoopStripMiningIterShortLoop

MetaspaceReclaimPolicy

MinHeapSize

NonNMethodCodeHeapSize

NonProfiledCodeHeapSize

OptoRegScheduling

| |
|---|
| PreTouchParallelChunkSize |
| PrintExtendedThreadInfo |
| PrintFlagsRanges |
| ProfiledCodeHeapSize |
| RecordDynamicDumpInfo |
| RestrictReservedStack |
| SegmentedCodeCache |
| SharedArchiveConfigFile |
| SharedArchiveFile |
| SharedSymbolTableBucketSize |
| ShenandoahGCHeuristics |
| ShenandoahGCMode |
| ShowCodeDetailsInExceptionMessages |
| ShrinkHeapInSteps |
| SoftMaxHeapSize |
| StackReservedPages |
| StartAggressiveSweepingAt |
| SuperWordLoopUnrollAnalysis |
| SuperWordReductions |
| SweeperThreshold |
| UseBASE64Intrinsics |
| UseCMoveUnconditionally |
| UseCodeAging |
| UseContainerCpuShares |
| UseDynamicNumberOfCompilerThreads |
| UseEmptySlotsInSupers |
| UseFMA |
| UseNotificationThread |
| UseProfiledLoopPredicate |
| UseShenandoahGC |
| UseSubwordForMaxVector |
| UseVectorCmov |
| UseXMMForObjInit |
| UseZGC |
| ZAllocationSpikeTolerance |
| ZCollectionInterval |

| |
|---|
| ZFragmentationLimit |
| ZMarkStackSpaceLimit |
| ZProactive |
| ZUncommit |
| ZUncommitDelay |

### List of removed and renamed runtime options

Below is the list of options not available in liberica-perf 8 and the options that were renamed.

| JDK 8 options | Resolution in liberica-perf 8 |
|---|---|
| AdaptiveSizePausePolicy | Removed |
| AdjustConcurrency | Removed |
| AggressiveOpts | Removed |
| AllowJNIEnvProxy | Removed |
| AllowNonVirtualCalls | Removed |
| AssertOnSuspendWaitFailure | Removed |
| AssumeMP | Removed |
| AutoGCSelectPauseMillis | Removed |
| BackEdgeThreshold | Removed.<br>Use `-XX:OnStackReplacePercentage` |
| BindGCTaskThreadsToCPUs | Removed |
| BranchOnRegister | Removed |
| BytecodeVerificationLocal | Is now  a diagnostic option |
| BytecodeVerificationRemote | Is now  a diagnostic option |
| CheckEndorsedAndExtDirs | Removed |
| ClearFPUAtPark | Removed |
| CMS* | CMS GC removed as well as its flags |
| CodeCacheMinimumFreeSpace | Removed |
| CollectGen0First | Removed |
| CompactFields | Removed |
| CompilationPolicyChoice | Removed |
| CompilerThreadHintNoPreempt | Removed |
| ConvertSleepToYield | Removed |
| ConvertYieldToSleep | Removed |
| CreateMinidumpOnCrash | Removed |
| Debugging | Removed |

| | |
|---|---|
| DefaultMaxRAMFraction | Removed |
| DefaultThreadPriority | Removed |
| DeferPollingPageLoopCount | Removed |
| DeferThrSuspendLoopCount | Removed |
| DeoptimizeRandom | Removed |
| EmitSync | Removed |
| EnableTracing | Removed |
| ErrorReportServer | Removed |
| ExplicitGCInvokesConcurrentAndUnloadsClasses | Removed |
| FailOverToOldVerifier | Removed |
| FastTLABRefill | Removed |
| FenceInstruction | Removed |
| FieldsAllocationStyle | Removed |
| FLSAlwaysCoalesceLarge | Removed |
| FLSCoalescePolicy | Removed |
| FLSLargestBlockCoalesceProximity | Removed |
| ForceNUMA | Removed |
| G1RSetScanBlockSize | Removed |
| GCLockerInvokesConcurrent | Removed |
| GCLogFileSize | Removed |
| GCTaskTimeStampEntries | Removed |
| InitialBootClassLoaderMetaspaceSize | Removed |
| InsertMemBarAfterArraycopy | Removed |
| JNIDetachReleasesMonitors | Removed |
| LazyBootClassLoader | Removed |
| LIRFillDelaySlots | Removed |
| LogJFR | Removed |
| MonitorBound | Removed |
| MonitorInUseLists | Removed |
| MustCallLoadClassInternal | Removed |
| NeedsDeoptSuspend | Removed |
| NmethodSweepCheckInterval | Removed |
| NmethodSweepFraction | Removed |
| NumberOfGCLogFiles | Removed |
| OldPLABWeight | Removed |
| ParallelGCVerbose | Removed |

| | |
|---|---|
| ParGCDesiredObjsFromOverflowList | Removed |
| ParGCTrimOverflow | Removed |
| ParGCUseLocalOverflow | Removed |
| PreInflateSpin | Removed |
| PrintAdaptiveSizePolicy | Removed |
| PrintClassHistogramAfterFullGC | Removed |
| PrintClassHistogramBeforeFullGC | Removed |
| PrintCMSInitiationStatistics | Removed |
| PrintCMSStatistics | Removed |
| PrintFLSCensus | Removed |
| PrintFLSStatistics | Removed |
| PrintGCApplicationConcurrentTime | Removed |
| PrintGCApplicationStoppedTime | Removed |
| PrintGCCause | Removed |
| PrintGCDateStamps | Removed |
| PrintGCID | Removed |
| PrintGCTaskTimeStamps | Removed |
| PrintGCTimeStamps | Removed |
| PrintHeapAtGC | Removed |
| PrintHeapAtGCExtended | Removed |
| PrintJNIGCStalls | Removed |
| PrintJNIResolving | Removed |
| PrintOldPLAB | Removed |
| PrintOopAddress | Removed |
| PrintParallelOldGCPhaseTimes | Removed |
| PrintPLAB | Removed |
| PrintPromotionFailure | Removed |
| PrintReferenceGC | Removed |
| PrintSafepointStatistics | Removed |
| PrintSafepointStatisticsCount | Removed |
| PrintSafepointStatisticsTimeout | Removed |
| PrintSharedSpaces | Removed |
| PrintStringDeduplicationStatistics | Removed |
| PrintTenuringDistribution | Removed |
| PrintTLAB | Removed |
| PrintVMQWaitTime | Removed |

| | |
|---|---|
| ProfileIntervals | Removed |
| ProfileIntervalsTicks | Removed |
| ProfilerPrintByteCodeStatistics | Removed |
| ProfilerRecordPC | Removed |
| ProfileVM | Removed |
| ReadPrefetchInstr | Removed |
| ReflectionWrapResolutionErrors | Removed |
| ResizeOldPLAB | Removed |
| SafepointPollOffset | Removed |
| SafepointSpinBeforeYield | Removed |
| SharedMiscCodeSize | Removed |
| SharedMiscDataSize | Removed |
| SharedReadOnlySize | Removed |
| SharedReadWriteSize | Removed |
| SpecialEncodeISOArray | Moved to diagnostic flags |
| StarvationMonitorInterval | Removed |
| StressLdcRewrite | Moved to diagnostic flags |
| SuspendRetryCount | Removed |
| SuspendRetryDelay | Removed |
| SyncFlags | Removed |
| SyncKnobs | Removed |
| SyncVerbose | Removed |
| ThreadSafetyMargin | Removed |
| TraceBiasedLocking | Removed |
| TraceClassLoading | Removed |
| TraceClassLoadingPreorder | Removed |
| TraceClassPaths | Removed |
| TraceClassResolution | Removed |
| TraceClassUnloading | Removed |
| TraceDynamicGCThreads | Removed |
| TraceExceptions | Removed |
| TraceGen0Time | Removed. Functionality moved to unified logging. |
| TraceGen1Time | Removed. Functionality moved to unified logging. |
| TraceLoaderConstraints | Removed |
| TraceMetadataHumongousAllocation | Removed |
| TraceMonitorInflation | Removed |

| | |
|---|---|
| TraceParallelOldGCTasks | Removed |
| TraceRedefineClasses | Removed |
| TraceSafepointCleanupTime | Removed |
| TraceSuspendWaitFailures | Removed |
| TransmitErrorReport | Removed |
| UnlinkSymbolsALot | Removed |
| UnlockCommercialFeatures | Removed |
| Use486InstrsOnly | Removed |
| UseAdaptiveGCBoundary | Removed |
| UseAESIntrinsics | Moved to diagnostic flags |
| UseAltSigs | Removed |
| UseAutoGCSelectPolicy | Removed |
| UseBoundThreads | Removed |
| UseCMSBestFit | Removed |
| UseCMSCollectionPassing | Removed |
| UseCMSCompactAtFullCollection | Removed |
| UseCMSInitiatingOccupancyOnly | Removed |
| UseCompilerSafepoints | Removed |
| UseConcMarkSweepGC | Removed |
| UseCRC32Intrinsics | Moved to diagnostic flags |
| UseFastAccessorMethods | Removed |
| UseFastEmptyMethods | Removed |
| UseGCLogFileRotation | Removed |
| UseGCTaskAffinity | Removed |
| UseGHASHIntrinsics | Moved to diagnostic flags |
| UseLargePagesInMetaspace | Removed |
| UseLegacyJNINameEscaping | Removed |
| UseLockedTracing | Removed |
| UseLWPSynchronization | Removed |
| UseMathExactIntrinsics | Moved to diagnostic flags |
| UseMembar | Removed |
| UseMontgomeryMultiplyIntrinsic | Moved to diagnostic flags |
| UseMontgomerySquareIntrinsic | Moved to diagnostic flags |
| UseMulAddIntrinsic | Moved to diagnostic flags |
| UseMultiplyToLenIntrinsic | Moved to diagnostic flags |
| UseOSErrorReporting | Removed from linux build |

| | |
|---|---|
| UseParallelOldGC | Removed |
| UseParNewGC | Removed |
| UseRDPCForConstantTableBase | Removed |
| UseSHA1Intrinsics | Moved to diagnostic flags |
| UseSHA256Intrinsics | Moved to diagnostic flags |
| UseSHA512Intrinsics | Moved to diagnostic flags |
| UseSquareToLenIntrinsic | Moved to diagnostic flags |
| UseVMInterruptibleIO | Removed |
| VerifyMergedCPBytecodes | Removed |
| VMThreadHintNoPreempt | Removed |
| WorkAroundNPTLTimedWaitHang | Removed |