Zing Virtual Machine Release Notes

This document provides release information for Zing Virtual Machine 18.02.0.0

February 22, 2018

Table of Contents

1 Zing Overview

The Azul Systems® Zing® platform uses the Zing Virtual Machine (ZVM) to run Java™ technology-based applications. In the Zing product, the Zing System Tools (ZST) component, installed on each ZVM host system, manages the elastic and highly scalable shared memory resources.

Document Title	Purpose	Format
Zing Virtual Machine Release Notes (this document)	Release information, including new ZVM features, resolved issues, and known issues.	PDF
Zing System Tools Release Notes	Release information about the latest available version of ZST.	PDF
Zing System Requirements and Compatibility References	List of Zing System Requirements including Operating System, CPU, Memory, and Hardware Disk Storage Space. Provides information about Zing Component Version Compatibility and ZST/ Zing API Compatibility.	PDF
Zing Getting Started Guide	Provides reference information about how to install Zing components, configure Zing memory management, and run your Java applications with Zing.	PDF
Zing User's Guide	Provides detailed description of Zing installation, memory configuration, using the Pool License Server, running Java applications with Zing, troubleshooting, and using additional tools and utilities to improve performance of your Java applications.	PDF
Zing Common Vulnerabilities and Exposures List	List of CVE fixes integrated in this release.	PDF
Zing MXBeans Javadoc	Javadoc documentation for Zing MXBeans.	PDF

2 Features and Updates

Advisory!	1. ZST 5.20.5 provides compatibility with the newly available fixes for the recently reported Intel CPU kernel side-channel security flaws and is required for Zing to operate on Linux distributions that have been updated to address these flaws by adding kernel page table isolation (KPTI). To avoid problems, you should upgrade your ZST to ZST 5.20.5 or higher as soon as possible. Our recommendation for best practice in isolating changes independently is to update Zing to use ZST 5.20.5 and verify your application launches and works, then apply your KPTI kernel update, and recheck that your application launches and still works. Azul is tracking industry-wide response to the recently reported side-channel security flaws. As changes to different operating system distributions emerge, Azul will continue to provide guidance on how to best accommodate those changes. 2. Azul recognizes the concern over a potential regression in Java's use of ZLib and subsequent data decompression errors, as discussed here: https://github.com/madler/zlib/issues/305. The Zing ZVM never shipped a public release exhibiting this problem. However, a preventive change included in ZVM 17.11.0 ensures the problem will not occur in future releases.

2.1 New Features and Updates

ZVM 18.02.0.0 is compatible with ZST 5.15.0 and above.

Below is the list of new features and updates introduced in ZVM 18.02.0.0:

Compiler Statistics Updates	This release introduces the following changes in compiler statistics reporting: • Adding total wait-in-queue time to compiler statistic reporting and GC log records. • Changing `TotalAcutalTimeMS` to `TotalCPUTimeMS` in GC log records and compiler statistics.
Performance and Stability Improvements	Zing 18.02.0.0 comes with several internal stability and performance fixes.

2.2 Features Added in Previous Releases

Zing 18.01.0.0

- Unlimited Cryptographic Policy is Enabled by Default.
- Providing Azul OEM Master License Key.

- Enhancements for gcLogAnalyser.
- Compiler Enhancements including Using Compile Stashing.
- Deprecation of Support for RHEL 5

Zing 17.12.0.0

- Enhance control of precompilation to allow control of each tier's precompliation separately.
- Add support for uncounted loop safepoint removal.
- Add a command-line option that changes the threshold at which the OSR compiles are triggered.
- Enhance the gcLogAnalyser tool's graph PNG file generator to automatically add html pages to:
  1. Easily navigate the graphs in a set of PNG files using a web browser
  2. Compare any two graphs in one view using a web browser

Zing 17.11.0.0

- Include JCE Jurisdiction Policy files.
- Set limit on array length.
- CodeCache flushing improvements.
- New Falcon inlining command-line option.
- ReadyNow! enhancements.

Zing 17.10.0.0

- Increase maximum heap size (`-Xmx`) supported by Zing from 2 TB to 8 TB.

Zing 17.08.0.0

- Eliminate a transaction latency problem.
- Support for ParallelClassLoaders.
- Multiple changes and improvements in the GC Log Analyser tool and ReadyNow!.
- Deprecation of the `GPGCConcurrentMarkGlobalHandles` command-line option.
- Add "update-alternatives" for Java and a symlink for easy access of Java binary.

Zing 17.06.0.0

- The ZST 5.20 that is included in the Zing Trial Program has two new behaviors. First, system-config-zing-memory is run automatically after the installation of the ZST to configure the System Zing Memory. Second, the default policy for reservation of memory used by the ZVM has been changed from reserve-at-config (reserved when system-config-zing-memory is run) to reserve-at-launch (reserved when the ZVM process is launched).
- Support for Stop-The-World (STW) garbage collection in the ZVM.
- Multiple changes and improvements in the GC Log Analyser tool.
- Enable CodeCache Flushing and support for ParallelClassLoaders.

Zing 17.03.0.0

- Falcon compiler becomes the default Tier 2 compiler for Zing (for Java SE 7 and 8) replacing C2.
- Multiple changes and improvements in the Zing monitoring tools.
- Many improvements of the ReadyNow! performance tuning tool.

Zing 16.12.0.0

- General Availability of the Falcon Compiler. Learn more about the Falcon specifics in the Zing User's Guide.
- Adding the `UseFastJNIAccessors` option to the list of the unchangeable options.
- Adding the ability to process files that have GC log lines without any timestamp preceding the log line label.

Zing 16.10.0.0

- Increased number of supported operating systems and kernels.
- Compiler Enhancements.
- Performance and Stability Improvements

Zing 16.07.0.0

- Intel TSX support
- Additions to the Garbage Collector output information
- Enhancements in the gcLogAnalyser tool
- Large number of supported operating systems and kernels
- Performance Improvements.

Zing 16.01.0.0

- Extended Java Heap Size up to 2 TB per JVM instance
- The Native Memory Tracking functionality includes invocation of Memory-tracking functions to record allocations.
- New graphs are available in the gcLogAnalyser tool.
- Bug fixes.

Refer to the *Zing System Requirements and Compatibility References* for more information about the supported operating systems and Zing component compatible versions.

3 Zing Virtual Machine Resolved Issues

The following table lists known issues that are resolved as of Zing Virtual Machine 18.02.0.0. The Bug IDs listed are Azul internal reference numbers.

Bug ID	Release Resolved	Description
13102	18.02.0.0	MapR default library loading incompatible with ReadyNow!. The fix ensures that MapR 4+ works with ReadyNow! with default settings.
13154	18.02.0.0	ZVM crashes with no `hs_err` and zero size core file when run in combination with `AZ_CHEAP_MEMORY_SANITIZER=1`.
10414	18.01.0.0	In the product version of the ZVM, restrict the set of command-line options shown in ZVision's HotSpot Flags window to the available set of `-XX` options. Previously, non-product options were also shown.
12521	18.01.0.0	Backport of JDK-8063086: `Math.pow` yields different results upon repeated calls.
12629	18.01.0.0	The application threads waits to be notified by the collector until the end of next new collection for allocations when they hit allocation failure. The use of the newly introduced `-XX:GPGCMutatorSleepBeforeAllocRetryMS` option (10 ms by default) makes the delayed to respond to the freed pages earlier and helps in reducing the length of allocation delays seen by the application threads.
12428	17.12.1.0	Expressions with nested `Math.pow()` fail with the result `Not-a-Number` error for some floating point values.
12767	17.12.1.0	Comodo root Certificate Authority is missing in cacerts files.
8606	17.12.0.0	Zing crashes if its command line contains only `-Xms` but no `-Xmx`, and `-Xms` is larger than default max heap size (currently 1G). The fix ensures no more such crashes happen.
11772	17.12.0.0	Zing tools do not work with an OEM license because the entry point JAR file is loaded by bootstrap.
12010	17.11.1.0	ZVM garbage collector related process abort at `C`

Bug ID	Release Resolved	Description
		`[libjvm.so+0x29672c] GPGC_Layout::addr_to_ BasePageForSpace`.
12461	17.11.1.0	ZVM Falcon compiler related process abort at `C [libjvm.so+0x58dca8] DolphinParser::reify_abstract_ state`.
12492	17.11.1.0	Interpreter or other runtime ZVM abort following a deoptimization of Falcon compiled code and an attempt to recompile and run the newly compiled method. Internal root cause is Falcon compiler's reuse of OopTable indices.
8341	17.11.0.0	ZVM will not start if `–Xmx` and `–Xms` are both specified and have values that are odd numbers.
11143	17.11.0.0	Unexpected use of the `System.nanoTime()` method in the Falcon compiler.
11850	17.11.0.0	ZVM crashed in `guarantee(secondary_supers()) failed: Unitialized secondary supers during typecheck`.
11896	17.11.0.0	Backport of JDK-6512830: `Error: assert(tag_at(which).is_ unresolved_klass(), "Corrupted constant pool")`.
12131	17.11.0.0	Backport JDK-8075484: `SocketInputStream.socketRead0` can hang even with `soTimeout` set.
12132	17.11.0.0	Backport JDK-8178536: OOM ERRORS + SERVICE-THREAD TAKES A PROCESSOR TO 100%.
10064	17.10.0.0	ZVM crashed due to incorrect deoptimization in clone intrinsic implementation.
10076	17.08.0.0	Creation of methodstubs (for C2i) for methods for all the loaded classes causes application to run out of code cache.
10567	17.08.0.0	ZVM crashed in `src/cpu/x86/vm/interpreterRT_x86.cpp:1085` with `Unimplemented()` error.

Bug ID	Release Resolved	Description
10882	17.08.0.0	Falcon compiler does not respect `DynamicBranchEliminationLevel` of 0 or 1. The fix makes Falcon to distinguish more levels.
10907	17.08.0.0	RMI Registry ignores depth limit pattern specified for the `registryFilter`.
11612	17.06.2.0	A bug in a function call in the optimized intrinsic code for AES.
10043	17.06.0.0	Using the `-usedatex` command-line option together with the Set Time Range option of the GC Log Analyser tool resulted in empty graphs.
10298	17.06.0.0	`INVOKEINTERFACE` called on the `java.lang.Object` method in ASM-generated byte code fails after several iterations.
10428	17.06.0.0	The use of the date/time X-axis of the GC Log Analyser tool encounters error caused by duplicate elapsed times for items in a dataset.
10440	17.06.0.0	Warning counters differ in jstat `-profile` and `-profileerrors` output of the ReadyNow! tool.
10499	17.03.3.0	Internal Error at `ciObject.hpp`: `compiler_assert(is_instance()) failed: bad cast :`
10353	17.03.3.0	Crash due to an unexpected error detected by Java Runtime Environment: `guarantee(lbl == _ex_labels->at(idx)) failed: single handler bci given 2 different rel_pc mappings.`
10334	17.03.3.0	Scala 2.12 fails with bytecode error.
10174	17.03.3.0	Crash with the error message: `LLVM fatal error: Unable to allocate section memory!`
10297	17.03.2.0	Crash with the following problematic frame: `J (C2) com.mchange.v2.c3p0.impl.NewPooledConnection.carefulCheckHoldability (Ljava/sql/Connection;)I`

Bug ID	Release Resolved	Description
9970	17.03.2.0	Zing crashes with the following error: `LLVM fatal error: Do not know how to split the result of this operator!`
10156	17.03.1.0	Too many recompilations on a particular method results in the same being marked "do not-compile any longer".
4702	17.03.0.0	Use of the `OnOutOfMemoryError` flag triggers a crash.
8298	16.12.3.0	New generation relocation aborts can lead to Java heap live set growth. This affects ZVM versions 16.01.0.0 through ZVM 16.12.2.
9463	16.12.3.0	Frequent New GC cycles can lead to Java heap live set growth. When the New GC intercycle time is lower than the promotion threshold, objects can be retained in the new generation causing a growth in the live set. This affects ZVM versions 15.09.0.0 through ZVM 16.12.2.
7911	16.12.2.0	Crash with the following problematic frame: `C [libjvm.so+0x38d1c2] jvmti_GetTime+0x62`.
9050	16.12.2.0	On some RHEL 5 systems, where the command: `find /sys/devices/system/cpu -name thread_siblings` produces a list of files that is in descending numerical order by a CPU directory name, `ThreadOpt` will exit with an `IndexOutOfBoundsException`.
9086	16.12.1.0	Applications can hang when ZVM runs with the Falcon compiler.
8655	16.12.0.0	Crash due to usage of the `+UseFastJNIAccessors` option. The option has not been implemented yet. To avoid crashing the option marked as unchangeable.
6520	16.12.0.0	No timestamp generated with the `+PrintGCDetails` option alone on command line causes gcLogAnalyser "unable to read".
8561	16.10.1.0	ZVM can exit due to code cache exhaustion by Java monitors. Applications which make heavy use of Java monitors either due to lock contention or wait/notify mechanisms, can potentially see a high footprint related to monitors in the code cache.

Bug ID	Release Resolved	Description
8224	16.10.0.0	A stray C2 thread is running at 100% CPU without making any forward progress.
8071	16.07.1.0	C-Heap Leak happens in raw-monitor creation. `RawMonitors` are leaking semaphore objects which are eagerly created during `VMLock` creation. The fix includes releasing `sem_t` objects and calling `sem_destroy`.
7756	16.07.1.0	ZVM crashes with the following problematic frame: `[libjvm.so+0x318ab3] java_lang_Class::as_klassOop (oopDesc*)+0x23`.
2820	16.07.1.0	Nashorn benchmark stalls after encountering data error processing profile log error (error: `101`) while running Nashorn benchmarks with ReadyNow! enabled. The fix adds supports for dealing with Nashorn and eliminates this issue.

4758	16.07.0.0	Additional control flags introduced to prevent startup crashes due to usage of `ProfileLogIn`. Following are those control flags:
		`ProfileUsePersistedInstructionData`
		`ProfileProactivelyCompileC1`
		`ProfileProactivelyCompileC2`

4811	16.07.0.0	`jstat -profile` (and `-profileerrors`) returned unresolved symbols when `ProfileLogIn` is not specified. The fix now ensures that zeros will be printed instead of unresolved symbols.
5803	16.07.0.0	Wrong class was being loaded only when using the `ProfileLogIn` profile. The fix implements safe mode for ReadyNow to avoid an improper speculative load that might produce a misleading and worrisome report.
6300	16.07.0.0	Removed non-relevant data from graphs (Old Gen Collector: App Threads Delay and Pages Promoted).
6778	16.07.0.0	ReadyNow failed to recognize the generated Lambda classes as generated. The fix implements safe mode for ReadyNow to use when ReadyNow's generated class heuristics do not work for a class generator.
7422	16.07.0.0	Enabling `ProfileLiveObjects` seems to increase GC cycle durations

Bug ID	Release Resolved	Description
		significantly. The fix implements an improved hash function along with few other optimizations resulting in reduced GC cycle durations.
7434	16.07.0.0	The default value of `MlockLevel` has been changed to 1 from 0: `MlockLevel=0` – do not attempt any mlocks `MLockLevel=1` – quietly attempt to mlock `libjvm.so`
7451	16.07.0.0	Crash with the following problematic frame: `StubRoutines::find_SEGV_continuation_address`.
7654	16.07.0.0, 16.01.7.0	Resolving an orphaned `FinalLive` objects through JNI weak handle during `ConcurrentRefProcessing` could lead to the following crash: `guarantee(loop_count < size) failed: should have found the relocation record`
7422	16.01.7.0	Enabling `ProfileLiveObjects` seems to increase GC cycle durations significantly. The fix implements an improved hash function along with few other optimizations resulting in reduced GC cycle durations.
5547	16.01.7.0	When running on some Cassandrs nodes, the following enexpected error has been detected: `guarantee(GPGC_Marks::is_any_marked_strong_live (obj)) failed: NewGen oop at final clear not strong live`
6907	16.01.6.0	Crash due to an internal error: `guarantee(loop_count < size) failed: should have found the relocation record` The fix ensures that a class of a Java object is treated as `StrongLive` even if that object is itself only `FinalLive`. This guarantees that the classes of `FinalLive` objects cannot be orphaned before the `StrongLive` mark-through of `FinalLive` referents, and will ensure their relocation.
6058, 6108	16.01.5.0	High GC pauses due to JVM code being paged out. The fix introduces a new command-line option `MlockLevel`, which can be tuned. If the flag is set without a value, the default ensures no page JVM code page-outs. The option specifies one of four mlock strategies:

Bug ID	Release Resolved	Description
		1. Do not attempt any mlock (EARLY) 2. mlock libjvm.so text region (LATE) 3. mlockall (EARLY) 4. mlockall (LATE) Example: use `-XX:+MlockLevel=1001` to attempt strategy 1 while reporting verbosely. **This option should be used only with guidance of Azul Support**.
6258	16.01.5.0	A random error code is being returned from the JDK method: `java.util.prefs.FileSystemPreferences.lockFile0().`
6436	16.01.5.0	In ZVision, clicking on the **tty_lock** entry in **zvision** -> **threads** -> **contention** causes a crash.
6631	16.01.5.0	`+MinimizeJNICriticalLock` causes an `IllegalArgumentException` in `java.util.zip.Deflater.deflate`.
6898	16.01.5.0	Nginx-Clojure does not work with Zing because the linker option `-Wl,-soname,libjvm.so` has not been passed and the `libjvm.so` library cannot be found.
7071	16.01.5.0	Zing VM can crash at very early stages due to lack of system resources.
6542	16.01.3.0	Applications that use large chunks(>100s of MBs) of `DirectByteBuffer` could potentially face high TTSP times in New/OldGC `pause3` because of the deallocation of these buffers.
6791	16.01.3.0	`NullPointerException` caused by wrong code generated from C2.
6938	16.01.3.0	`Thread.sleep` can sleep ~30% longer than specified when the Intel pstate driver is enabled. The pstate driver can be disabled to workaround this problem with older ZVMs. You can determine whether the pstate driver has been enabled by checking `/sys/devices/system/cpu/cpu0/cpufreq/scaling_driver` for the string “intel_pstate”.
6956	16.01.3.0	A race condition in `CodeCache::GPGC_unlink` can cause a crash

Bug ID	Release Resolved	Description
		when there are multiple `GenPauselessOldThreads`. This affects ZVM versions 16.01.0.0, 16.01.1.0, and 16.01.2.0. For these ZVMs, the workaround is to set `-XX:GenPauselessOldThreads=1` but this can cause a spike in Old GC cycle times for applications with a large live set.
6736	16.01.2.0	Fatal error in native method: `JDWP PushLocalFrame: Unable to push JNI frame, jvmtiError=AGENT_ERROR_OUT_OF_MEMORY (188).`
6124	16.01.1.0	Crash in generated code of the following method: `it.unimi.dsi.fastutil.ints.IntRBTreeSet.add`.
6370	16.01.1.0	GC log file SYSINFO line reports Page Cache active(file) value for the Page Cache active(anon) value.
2699	16.01.0.0	New intrinsic for `BigInteger.multiplToLen()` for improved `BigInteger` performance.
3178	16.01.0.0	The new `UseCRC32Intrinsics` option for improved CRC32 performance.
3910	16.01.0.0	New intrinsics for the `squareToLen` and `mulAdd` methods in `BigInteger` for improved `BigInteger` performance.
5002	16.01.0.0	The `UseSuperWord` ZVM option is turned on by default. It enables up to 16-byte vectorization.
5437	16.01.0.0	Memory-tracking functions to record the use of the native memory. For more details, see the Native Memory Tracking section of the *Zing 16.01.0.0 User Guide*.
5478	16.01.0.0	As of ZVM 14.09.0.0, a difference in the version of libstdc++ on the system and the libstdc++ version that is statically linked into libjvm could cause a crash due to malloc corruption.
5626	16.01.0.0	The `printir` compiler command dumps out C2 node IR phase by phase for a specific Java method.
5663	16.01.0.0	Crash with the following problematic frame: `AddNode::Ideal`

Bug ID	Release Resolved	Description
		`(PhaseGVN*, bool)`.
5858	16.01.0.0	When the JVM core dumps, the hs error file is incorrectly reporting `SCHED_IDLE` threads as `UNKNOWN`.
5754	16.01.0.0	Crash with the following problematic frame: `jvm_exception_handler`.
5791	16.01.0.0	VM takes a long time to shutdown due to background page scrubbing.
6142	16.01.0.0	Crash with the following problematic frame: `GraphKit::add_ exception_states_from(JVMState*)`.
5467	15.09.1.0	Crash in `Java_sun_font_ FreetypeFontScaler_ disposeNativeScaler()`.

4 Zing Virtual Machine Known Issues

The following table lists known issues that are known issues as of Zing Virtual Machine 18.02.0.0. The Bug IDs listed are Azul internal reference numbers.

Bug ID	Release Known	Description
6318	15.09.0.0	When using `reserve-at-launch`, you may see an exception such as "`MemoryUsage ERROR: initialReserved 0 size -4738568192 used -4736471040`" in Zing MXBean calls that use `MemoryUsage` objects for System Linux Memory. As of ZVM 16.01.0.0, the probability of seeing this error has been reduced. If the error is seen, it can safely be ignored.
5348	15.09.0.0	WebSphere fails to launch when using `-XX:+UseZingMXBeans`. **Workaround**: Set the following by using `server.xml` or WAS admin console (note the empty value for the system property): `genericJvmArguments="-XX:+UseZingMXBeans -Djavax.management.builder.initial= "`
2924	all Zing releases	Loop predicate and loop limit check code problem. (1) The java spec says explicitly that operations on integers can overflow and no exception will be thrown. Indeed, unlike the C++ spec that says integer overflow is undefined, java says integer overflow is required to happen. (2) Java coding guidelines always point out that using Integer.`MAX_VALUE` in comparison expressions is dangerous programming. Especially if it is used in a loop bound. (2a) This loop will terminate because eventually i will be equal to Integer.`MAX_VALUE`: `for (int i=0; i<Integer.MAX_VALUE; i+=1) { ... }` (2b) This loop will never terminate, because i will overflow: `for (int i=0; i<Integer.MAX_VALUE; i+=2) { ... }` (3) Zing will sometimes terminate the loop in (2b). Note: `-XX:+DisableLoopOptimizations` will often avoid the problem but is is only recommend as a workaround on a per-method basis. -`XX:CompileCommand='disableloopopts,classname::method'`
3958	15.09.0.0	JBoss7 throws an exception on startup with -`XX:+UseZingMXBeans` flag. **Workaround**: Add the following lines to the `standalone.conf` file: `JAVA_OPTS="$JAVA_OPTS -Djava.util.logging.manager=org.jboss.logmanager.LogManager"`

Bug ID	Release Known	Description
		`JAVA_OPTS="$JAVA_OPTS -Xbootclasspath/p:../modules/org/jboss/logmanager/main/jboss-logmanager-1.2.0.GA.jar:../modules/org/jboss/logmanager/log4j/main/jboss-logmanager-log4j-1.0.0.GA.jar:../modules/org/apache/log4j/main/log4j-1.2.16.jar"`
2602	14.09.0.0	ZVM running in environment with multiple scheduling policies, RR, BATCH, and OTHER, encounters checkpoint sync timeout in thread running with BATCH policy.

Legal Notice

Published, February 22, 2018