JavaBugs

Java Bugs in various JVMs affecting Lucene / Solr

Sometimes Lucene runs amok of bugs in JVM implementations from different vendors. In certain cases we whittle it down to a small test case, open an issue with the vendor, and hopefully it gets fixed. In other cases we know the bug is in the JVM but we haven't narrowed it enough to open a bug with the vendor. Sometimes we can work out a simple workaround in Lucene.

We try to open a Lucene mirror bug to provide details on how Lucene is affected, iterate on a compact test case, etc.

Oracle Java / Sun Java / OpenJDK Bugs

If you are affected by one of these issues that Oracle's Java has yet to accept or resolve, or simply have some spare votes, please consider adding your vote to the bug (on Oracle's bug page):

Do not, under any circumstances, run Lucene with the G1 garbage collector. Lucene's test suite fails with the G1 garbage collector on a regular basis, including bugs that cause index corruption. There is no person on this planet that seems to understand such bugs (see https://bugs.openjdk.java.net/browse/JDK-8038348, open for over a year), so don't count on the situation changing soon. This information is not out of date, and don't think that the next oracle java release will fix the situation.

Oracle bug	Lucene mirror issue	Impact to Lucene	State, Priority	Workaround
6265734	LUCENE-573	`NIOFSDirectory` has very poor performance on Windows	Accepted, Low	Use `FSDirectory` or `MMapDirectory` on Windows
6478546	LUCENE-1566	You hit a false `OutOfMemoryException` when loading norms in an index with many docs	Accepted, Low	Locally patch Lucene to load large contiguous byte sequences in chunks
6707044	LUCENE-1282	Index corruption	Fixed as of 1.6.0_10, High	Lucene code base has a workaround in it
4724038	LUCENE-1658 LUCENE-1669	`MMapDirectory` won't close files until GC (higher transient disk usage than other `FSDir`); `MMapDirectory` returns 0 bytes when used for read/write access in remote (`SMB/CIFS`) mount	Cause Known, Low	Use a different Directory implementation
(No Oracle bug yet)	LUCENE-1342	SEGV during indexing, with Java 1.6 64 bit	(No compact test case yet)	None known
(No Oracle bug yet)	Discussed here and here	On 64 bit JREs, reading from files may hang (??)	(Still characterizing)	None known
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1" ac:macro-id="87221b11-cc1e-468d-b506-07ae14c7a6f8"><ac:plain-text-body><![CDATA[	[6588260	https://bugs.openjdk.java.net/browse/JDK-6588260]	[LUCENE-2449	https://issues.apache.org/jira/browse/LUCENE-2449]	No impact as of 3.1/4.0 – we now avoid calling new String(int[], int, int)	Fixed, Delivered, Bug		]]></ac:plain-text-body></ac:structured-macro>
6240963	LUCENE-2685	Fix Delivered, High	xml-query-parser's XSLT transforms fail under some locales	Upgrade to Java 6, use a different JRE vendor, plug in an alternate, more up-to-date XSL engine like Apache XALAN, or set system property telling TransformerFactory to use the non byte-code generating bundled XSL engine in Java 5
5091921	LUCENE-2975	readVInt() returns wrong results	not sure	Lucene code base has a workaround in it
7044738	LUCENE-3346	readVInt() returns wrong results	8-Fix Available, Medium	use Java 6 or Java 7u1
7070134	LUCENE-3335	Porter Stemmer crashes JRE	8-Fix Available, Low	-XX:-UseLoopPredicate or Java 7u1
7104012	LUCENE-3301	BreakIterators (e.g. in analyzers) crash on certain inputs	Accepted, Low	Lucene code base has a workaround in it
6822370	LUCENE-3235	Concurrent Classes suffer from a race in LockSupport park() arising from weak memory models	Fixed, High	Use -XX:+UseMembar if you are running on a JVM < 1.6 u18 see this writeup for details
8024830	LUCENE-5212	SEGV or serious index corruption	Fixed, Critical	Use -XX:-UseSuperWord if you are running on 7u40 <= JVM < 7u55

Oracle JRockit Bugs

Oracle JRockit is no longer available for Java 7+, but it is still used in legacy environments (e.g. Oracle/BEA WebLogic application server). Oracle seems to no longer accept bugs / only accepts bugs from customers with a paid support contract, so there is no chance for the Lucene Contributors to report bugs upstream.

As stated by Oracle, it is recommended to upgrade to Oracle Java 7 or OpenJDK 7 (both update 1 or later). JRockit key features were merged into OpenJDK / Oracle Java.

Lucene issue	Impact to Lucene	Severity	Workaround
(No issue yet)	"If" statements are not being executed	Serious (index corruption)	Use `-XnoOpt` JVM parameter

IBM J9 Bugs

IBM J9 is available for Java 5, 6, and 7. Unfortunately, there is no public bugtracker available, so the bugs listed below cannot be reported. IBM only has a bug tracker for paying customers with a support contract.

Although some Lucene committers are IBM employees, its impossible to report bugs involving the whole Lucene community. Because of this the status of all bugs listed below is unknown, we have no information about possible fixes.

If you are using SUSE Enterprise Linux which has IBM J9 as their default Java implementation, it is strongly recommend to use Oracle Java 7 or OpenJDK 7 (minimum update 1).

The following bugs are known, but since a while all nightly tests using the IBM J9 were disabled on Lucene's Jenkins servers, because a lot of test fail randomly.

Lucene issue	Impact to Lucene	Severity	Workaround
(No issue yet)	FST.pack() produces corrupt index (Lucene 4) because a loop is miscompiled	Serious (index corruption)	Use `-Xjit:exclude={org/apache/lucene/util/fst/FST.pack(IIF)Lorg/apache/lucene/util/fst/FST;}` JVM parameter to disable optimizations in FST.pack()
LUCENE-4987	a synchronized method is completely optimized away	tests only	the test framework was updated to version 2.0.10 which contains a workaround