Testing Lucene and Solr with various JVMs:

Bugs, Bugs, Bugs

Uwe Schindler

Apache Lucene Committer & PMC Member

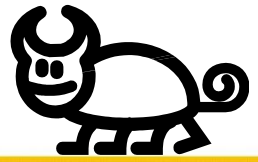
uschindler@apache.org

http://www.thetaphi.de, http://blog.thetaphi.de

@ThetaPh1

SD DataSolutions GmbH, Wätjenstr. 49, 28213 Bremen, Germany

Tel: +49 421 40889785-0, http://www.sd-datasolutions.de



My Background

- **Committer** and **PMC member** of **Apache Lucene and Solr** - main focus is on development of Lucene Java.
- Implemented **fast numerical search** and maintaining the **new attribute-based text analysis API**. Well known as *Generics and Sophisticated Backwards Compatibility Policeman*.
- Working as **consultant** and software architect for **SD DataSolutions GmbH** in Bremen, Germany. The main task is maintaining PANGAEA (Publishing Network for Geoscientific & Environmental Data) where I implemented the portal's geo-spatial retrieval functions with Apache Lucene Core.
- Talks about Lucene at various international conferences like the previous Berlin Buzzwords, ApacheCon EU/NA, Lucene Eurocon, Lucene Revolution, and various local meetups.

Agenda

- Some history
- **The famous bugs ☺**
- How to debug hotspot problems
- *Setting up Jenkins to test your software with lots of virtual machine vendors*
- **Bugs, Bugs, Bugs**

What happened?

SOME HISTORY…

Chronology

- **Java 7 Release Candidate** released July 6, 2011 as build 147 *(compiled and signed on June 27, 2011 – also the release date of OpenJDK 7 b147)*
- **Saturday, July 23, 2011:**
  - downloaded it to do some testing with Lucene trunk, core tests ran fine on my Windows 7 x64 box
  - Installation of FreeBSD package on Apache's Jenkins "Lucene" slave **=> heavy testing started: various crashes/failures:**

Issues found

- Jenkins revealed **SIGSEGV bug in Porter stemmer** (found when number of iterations were raised) [LUCENE-3335]
- New Lucene 3.4 facetting test sometimes **produced corrupt indexes** [LUCENE-3346]

WARNING !!!

- **Also Java 6 was affected!**
  (some time after the only stable version 1.6.0_18)
- Optimizations disabled by default, so:

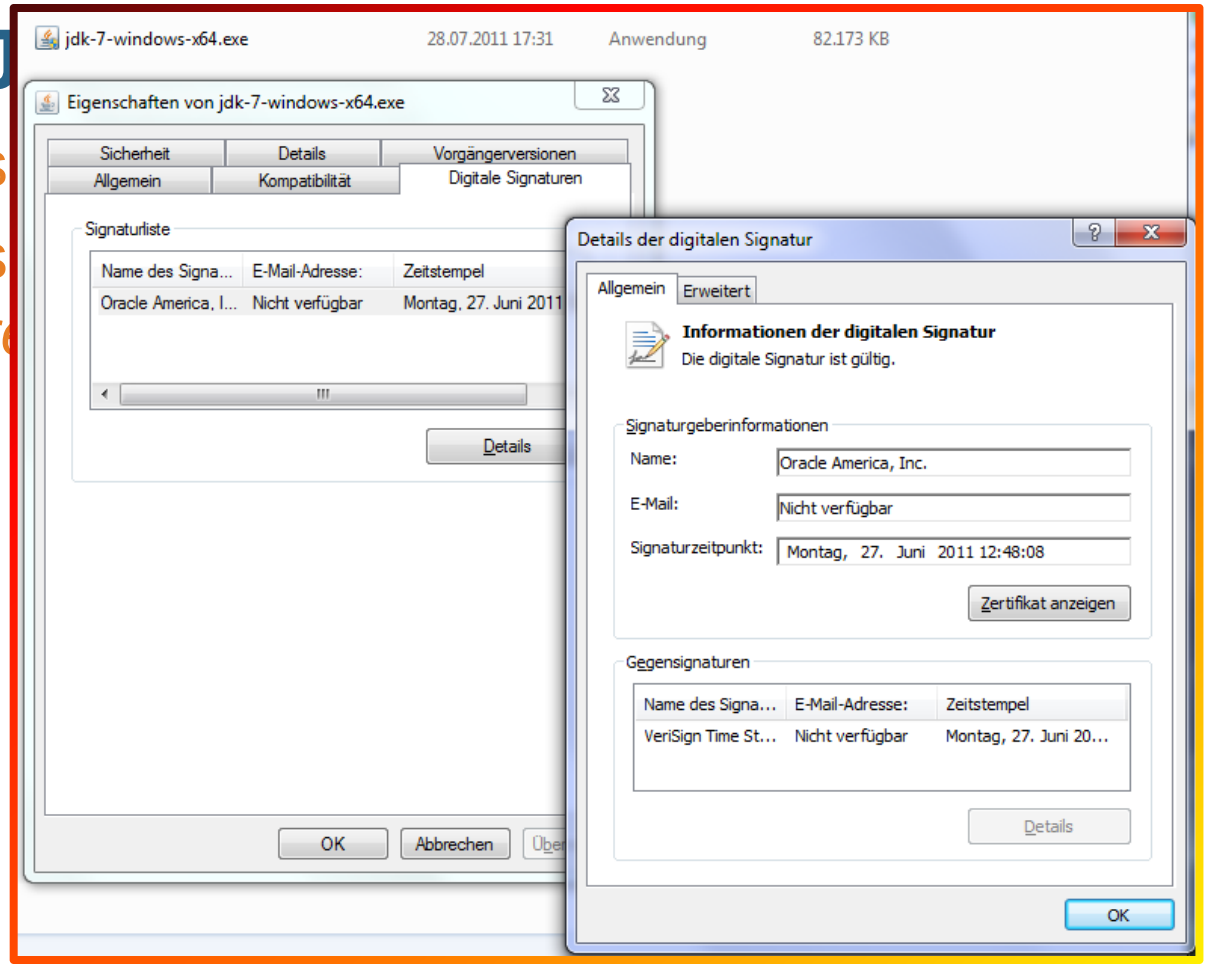
Don't use `-XX:+AggressiveOpts`
if you want your loops behave correctly!

Chronology

- **Thursday, July 28, 2011:**
  - Oracle released JDK 7 to public
  - Package was identical to release candidate *(Windows EXE signature dated June 27, 2011)*

Chronology

- **Thursday, J**
  - Oracle releas
  - Package was *EXE signatur*



Chronology

- **Thursday, July 28, 2011:**
  - Oracle released JDK 7 to public
  - Package was identical to release candidate *(Windows EXE signature dated June 27, 2011)*

- Apache Lucene PMC decided to warn users on web page and announce@apache.org mailing list

Chronology:
Friday, July 29, 2011

THE H OPEN

The H | open source | security | In association with

Last 7 days | News Archive | Features | Forums | Newsletter | RSS

29 July 2011, 12:58 « previous | next »

Java 7 paralyses Lucene and Solr

The hotspot compiler in the recently released Java 7 has a defective optimiser that can cause flawed loops, according to a warning published by the Apache Software Foundation. As a result, the Java Virtual Machine can crash, and calculations can produce incorrect results.

A number of Apache projects are affected, including every published version of Lucene and Solr. The Apache developers say that the indexing of documents on Solr causes Java to crash. Loops in Lucene can also be incorrectly compiled, thereby corrupting the indexes. In particular, the trunk version of Lucene with the pulsing codec is affected.

The bugs were discovered only five days before Java 7 was published; Oracle says it will correct them in the second service release of Java 7 at the latest; the first update to Java 7 was reserved solely for security fixes, but the issue may prompt Oracle to change that plan. Until then though, users of Lucene and Solr should refrain from using the new version of Java or at least use the JVM option `-XX:-UseLoopPredicate` to disable the optimisation and prevent the index from being damaged.

The Apache developers say that users of Java 6 could also be affected. However, the flaws only occur in Java 6 when the JVM is used with the options `-XX:+OptimizeStringConcat` or `-XX:+AggressiveOpts` which activate normally disabled Hotspot optimisations.

Oracle has registered the flaws under 7070134, 7044738 and 7068051. The first one causes JVM to crash when Martin Porter's stemmer algorithm is used, which traces English words back to their stems. This flaw currently is of "low priority" while the others are "medium".

(djwm)

ronology:
July 29, 2011

THE H OPEN

The H | open source | security | In association with

Last 7 days | News Archive | Features | Forums | Newsletter | RSS

29 July 2011, 12:58 « previous | next »

Java 7 paralyses Lucene and Solr

The hotspot compiler in the recently released Java 7 has a defective optimiser that can cause flawed loops, according to a warning published by the Apache Software Foundation. As a result, the Java Virtual Machine can crash, and calculations can produce incorrect results.

A number of Apache projects are affected, including every published version of Lucene and Solr. The Apache developers say that the indexing of documents on Solr causes Java to crash. Loops in Lucene can also be incorrectly compiled, thereby corrupting the indexes. In particular, the trunk version of Lucene with the pulsing codec is affected.

The bugs were discovered only five days before Java 7 was published; Oracle says it will correct them in the second service release of Java 7 at the latest; the first update to Java 7 was reserved solely for security fixes, but the issue may prompt Oracle to change that plan. Until then though, users of Lucene and Solr should refrain from using the new version of Java or at least use the JVM option `-XX:-UseLoopPredicate` to disable the optimisation and prevent the index from being damaged.

The Apache developers say that users of Java 6 could also be affected. However, the flaws only occur in Java 6 when the JVM is used with the options `-XX:+OptimizeStringConcat` or `-XX:+AggressiveOpts` which activate normally disabled Hotspot optimisations.

Oracle has registered the flaws under 7070134, 7044738 and 7068051. The first one causes JVM to crash when Martin Porter's stemmer algorithm is used, which traces English words back to their stems. This flaw currently is of "low priority" while the others are "medium".

(djwm)

jaxenter

NEWS | VIDEOS | BOOKS | EVENTS | JAVA TECH JOURNAL

HOME | NEWS | JAVA | EDITORS PICK | ECLIPSE | ANDROID | ARCHITECTURE | CLOUD

! Find out what's new in JBoss AS7 and OpenShift, in the new issue of Java Tech Journal!

July 29, 2011 RELATED NEWS | BOOKS | VIDEOS | EVENTS

Apache Code Affected by Java 7

Java 7 Could Cause Bugs in Some Apache Projects

Share 4 | Tweet 5 | +1 0 | Share 2 | Comment | Email | Share

Uwe Schindler has posted that the just-released Java 7 contains hotspot compiler optimisations, which miscompile some loops, and this can **affect the code of "several" Apache projects**. This can potentially lead to JVM crashes, or the incorrect calculation of results, ultimately leading to bugs in applications. Currently, it is known that all versions of Lucene Core and Solr released today, are affected by these bugs. Java 6 users are also affected, if they use one of the JVM options that are not enabled by default:

-XX:+OptimizeStringConcat or
-XX:+AggressiveOpts

> "These problems were detected only 5 days before the official Java 7 release, so Oracle had no time to fix those bugs," states the announcement. "It is strongly recommended not to use any hotspot optimization switches in any Java version without extensive testing!"

Oracle have proposed to include fixes in service release u2, and eventually in service release u1.

Jessica Thornsby


THE H OPEN
The H
Last 7 days
News Archive
29 July 2011, 12:58
Java 7 paralyses Lucene
The hotspot compiler in the recently
defective optimiser that can cause f
warning published by the Apache S
result, the Java Virtual Machine can
produce incorrect results.
A number of Apache projects are af
Lucene and Solr. The Apache deve
Solr causes Java to crash. Loops in
thereby corrupting the indexes. In p
pulsing codec is affected.
The bugs were discovered only five
says it will correct them in the secon
first update to Java 7 was reserved
prompt Oracle to change that plan.
should refrain from using the new v
-XX:-UseLoopPredicate to disable th
being damaged.
The Apache developers say that us
However, the flaws only occur in Jav
XX:+OptimizeStringConcat or -XX:+
disabled Hotspot optimisations.
Oracle has registered the flaws und
first one causes JVM to crash when
which traces English words back to
priority" while the others are "mediu
(djwm)
cnet News
Reviews
News
Downloads
Video
How To
Latest News
Apple
Crave
Business Tech
Green Tech
Wireless
Home > News > Business Tech
Business Tech
Ad Info
Bis 12% und mehr
Endlich auch für Privatanleger: Teakholzinvestment ab 3.900€ bis zu 12% p.a. und mehr steuerfrei
Selbständig? Unter 55?
Private Krankenkasse ab nur 57,- Euro für Selbständige und Freiberufler unter 55 Jahren!
Gabelst
FILED UNDER: BUSINESS TECH
Oracle releases 'buggy' Java SE7
By: Ben Woods
JULY 29, 2011 10:25 AM PDT
Print
E-mail
Recommend 33
Tweet 84
+1 9
Share
7 comments
Oracle released its first full version of Java yesterday, but developers have reported bugs that can crash virtual machines, corrupt data, and cause errors in applications.
Java Standard Edition 7 (SE7) is the first milestone since Oracle bought Java's creator, Sun, which at the time prompted fears from some community members about the future of Java.
The release includes improved support for dynamic languages, multicore-compatible APIs, and additional networking and security features. Oracle said in a statement it is the culmination of "industry-wide development involving open review, weekly builds and extensive collaboration between Oracle engineers and members of the worldwide Java ecosystem."
However, the Apache Lucene search engine project management committee warned yesterday that Java SE7 contained bugs that could cause a Java Virtual Machine to crash or affect applications.
See also: Scoop: Oracle scrubs site of embarrassing Java blog
Read more of "Oracle releases 'buggy' Java SE7" at ZDNet UK.
JAVA TECH JOURNAL
CLIPSE
ANDROID
ARCHITECTURE
CLOUD
OpenShift, in the new issue of Java
ED
NEWS
BOOKS
VIDEOS
EVENTS
Some Apache Projects
Share 2
Comment
Email
Share
Java 7 contains hotspot compiler
nd this can affect the code of "several"
JVM crashes, or the incorrect calculation of
ns. Currently, it is known that all versions of
d by these bugs. Java 6 users are also
are not enabled by default:
y 5 days before the official Java 7 release, so
gs," states the announcement. "It is strongly
ot optimization switches in any Java version
release u2, and eventually in service release
Jessica Thornsby

InfoWorld

Sign in

CHANNELS | Application Development | Applications | Cloud Computing | Data Center

News | Blogs | Test Center | Technologies | Tech Watch | White P

InfoWorld Home / InfoWorld Tech Watch / Apache and Oracle warn of serious Java 7 compiler...

The First Word on Tech

INFOWORLD TECH WATCH

JULY 29, 2011

Apache and Oracle warn of serious Java 7 compiler bugs

The newly released Java upgrade suffers hotspot-compiler problems that affect Lucene and Solr

By Ted Samson | InfoWorld

Follow @tsamson_IW

Print | 4 Comments

Gefällt mir

It looks like a few bugs have crashed Oracle's Java 7 release party that can wreak havoc on Apache Project applications. The news likely will come as a disappointment to fans of Java, who've waited five long years for a major update to the language.

Released just today, Java 7 includes hotspot-compiler optimizations that miscompile certain loops, potentially affecting projects such as Apache Lucene Core, Apache Solr, and possibly others, according to a warning from the Apache Project. At best, the bugs only cause JVMs to crash; in other cases, they result in miscalculations that can lead to application bugginess.

Downloads | Video | How To

usiness Tech | Green Tech | Wireless | Se

Ad Info

elbständig? Unter 55?

Private Krankenkasse ab nur 57,- Euro für Selbständige und Freiberufler unter 55 Jahren!

Gabelst

Print | E-mail

Share | 7 comments

rs have reported bugs that can crash virtual

ought Java's creator, Sun, which at the time of Java.

multicore-compatible APIs, and additional e culmination of "industry-wide development etween Oracle engineers and members of the

committee warned yesterday that Java SE7 r affect applications.

va blog

JAVA TECH JOURNAL

CLIPSE | ANDROID | ARCHITECTURE | CLOUD

OpenShift, in the new issue of Java

ED | NEWS | BOOKS | VIDEOS | EVENTS

n Some Apache Projects

Share 2 | Comment | Email | Share

Java 7 contains hotspot compiler

nd this can affect the code of "several"

JVM crashes, or the incorrect calculation of

ns. Currently, it is known that all versions of

d by these bugs. Java 6 users are also

are not enabled by default:

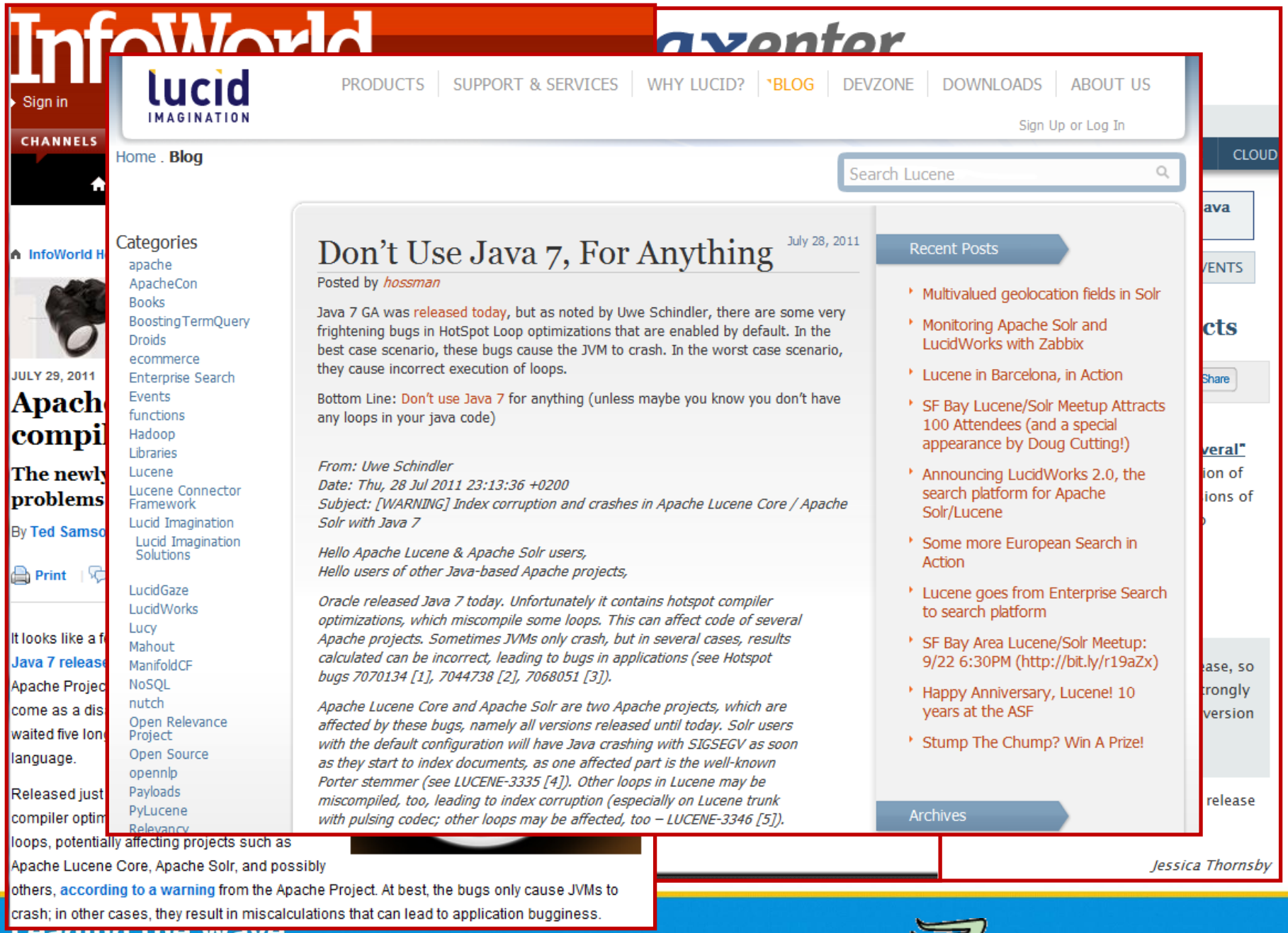
y 5 days before the official Java 7 release, so

gs," states the announcement. "It is strongly

ot optimization switches in any Java version

release u2, and eventually in service release

Jessica Thornsby


InfoWorld
Sign in
CHANNELS
JULY 29, 2011
By Ted Samso
Print
It looks like a f
Java 7 releas
Apache Projec
come as a dis
waited five lon
language.
Released just
compiler optim
loops, potentially affecting projects such as
Apache Lucene Core, Apache Solr, and possibly
others, according to a warning from the Apache Project. At best, the bugs only cause JVMs to
crash; in other cases, they result in miscalculations that can lead to application bugginess.
lucid IMAGINATION
PRODUCTS
SUPPORT & SERVICES
WHY LUCID?
BLOG
DEVZONE
DOWNLOADS
ABOUT US
Sign Up or Log In
Home . Blog
Search Lucene
Categories
apache
ApacheCon
Books
BoostingTermQuery
Droids
ecommerce
Enterprise Search
Events
functions
Hadoop
Libraries
Lucene
Lucene Connector Framework
Lucid Imagination
Lucid Imagination Solutions
LucidGaze
LucidWorks
Lucy
Mahout
ManifoldCF
NoSQL
nutch
Open Relevance Project
Open Source
opennlp
Payloads
PyLucene
Relevancy
Don't Use Java 7, For Anything
July 28, 2011
Posted by hossman
Java 7 GA was released today, but as noted by Uwe Schindler, there are some very frightening bugs in HotSpot Loop optimizations that are enabled by default. In the best case scenario, these bugs cause the JVM to crash. In the worst case scenario, they cause incorrect execution of loops.
Bottom Line: Don't use Java 7 for anything (unless maybe you know you don't have any loops in your java code)
From: Uwe Schindler
Date: Thu, 28 Jul 2011 23:13:36 +0200
Subject: [WARNING] Index corruption and crashes in Apache Lucene Core / Apache Solr with Java 7
Hello Apache Lucene & Apache Solr users,
Hello users of other Java-based Apache projects,
Oracle released Java 7 today. Unfortunately it contains hotspot compiler optimizations, which miscompile some loops. This can affect code of several Apache projects. Sometimes JVMs only crash, but in several cases, results calculated can be incorrect, leading to bugs in applications (see Hotspot bugs 7070134 [1], 7044738 [2], 7068051 [3]).
Apache Lucene Core and Apache Solr are two Apache projects, which are affected by these bugs, namely all versions released until today. Solr users with the default configuration will have Java crashing with SIGSEGV as soon as they start to index documents, as one affected part is the well-known Porter stemmer (see LUCENE-3335 [4]). Other loops in Lucene may be miscompiled, too, leading to index corruption (especially on Lucene trunk with pulsing codec; other loops may be affected, too – LUCENE-3346 [5]).
Recent Posts
Multivalued geolocation fields in Solr
Monitoring Apache Solr and LucidWorks with Zabbix
Lucene in Barcelona, in Action
SF Bay Lucene/Solr Meetup Attracts 100 Attendees (and a special appearance by Doug Cutting!)
Announcing LucidWorks 2.0, the search platform for Apache Solr/Lucene
Some more European Search in Action
Lucene goes from Enterprise Search to search platform
SF Bay Area Lucene/Solr Meetup: 9/22 6:30PM (http://bit.ly/r19aZx)
Happy Anniversary, Lucene! 10 years at the ASF
Stump The Chump? Win A Prize!
Archives
Jessica Thornsby

Further analysis the week after

Further

Share Report Abuse Next Blog»

Enterprise Software Development with Java

This is a blog about software development for the enterprise. It focuses on Java Enterprise Edition (J2EE/Java Beside this, I blog about Oracle WebLogic and GlassFish Server and other technologies that hit my road.

FRIDAY, JULY 29, 2011

Don't Use Java 7? Are you kidding me?

Tweet 40 +1 10

Java 7 was released yesterday and some guys from the Apache Lucene & Apache Solr community quickly came up with a couple of issues which lead them to the point where they are actively rejecting Java 7 and advice anybody else to to likewise. Even a general warning was issued by Apache Lucene PMC Member Uwe Schindler. But what exactly is wrong with Java 7 and why shouldn't you use it after waiting nearly five years for it? Let's look at this.

It's not about Java 7 but about the JVM

First of all, it's not about Java 7 in general but about the HotSpot JVM. The GA release contains three bugs (7070134, 7044738 and 7068051) which could affect the users with either JVM crashes or wrong calculations.

Hotspot crashes with sigsegv from PorterStemmer

The first one is about a wrong compiler optimization that changed the loop optimizations. The problem is, that this JVM feature is on by

ABOUT ME

myfear

German author and software architect

View my complete profile

Subscribe in a reader

MORE EISELE.NET

home | work | blog | personal | disclaimer

New Design

"Architecture is frozen music"
(Arthur Schoppenhauer)

LABELS

oracle weblogic server glassfish

Java.net The Source for Java Technology Collaboration

Login | Join Now | Help

Forums | Blogs | Projects | People | Search All

Home | Projects | Forums | People | Java User Groups | My Projects

Get Involved
- About Java.net
- Create a Project
- Java.net Enhancements

Get Informed
- Articles
- Blogs
- Site Wiki
- Events
- Oracle University

Cay Horstmann

In Praise of Language Specs | Main | Inner Classes in Scala and Java

Java 7 Unsafe at Any Speed?

Posted by cayhorstmann on July 29, 2011 at 7:32 PM EDT

Some people are nervous about everything—killer bees, poison oak, martian invaders, socialized medicine, you know the type. I try not to be like that. When JDK 7 went final yesterday, I boldly went into my `.bashrc` and changed `JAVA_HOME` to point to `jdk1.7.0`. Then I read this.

So, apparently, under some conditions, Hotspot messes up. It might crash, but that doesn't bother me so much—I'd notice that. But it might also silently produce the wrong result. I try not to be a scaredy squirrel about these things, but I must say that "rarely happening" bugs in a widely used platform bother me. When Toyota cars had random brake problems, the NHSTA ultimately concluded that floor mats, sticky pedals, and "pedal misapplication" were the culprits. But what if the electronics had a bug that only happens in a confluence of rare circumstances? They said no, but how can they really know?

It all reminds me of the Pentium bug from 1994. Intel had just released the first Pentium chip, and I immediately went and bought one at considerable expense to the management. Later I learned that a mathematics professor, Dr. Thomas Nicely of Lynchburg College, had run into a curious issue. On a small set of inputs, the multiplication was buggy. For example, 4195835- 4195835 / 3145727 × 3145727 yielded 256 instead of the expected 0. I tried it out on my new computer. Sure enough, I got 256. I tried it out on an older 486. I got 0.

It turned out that Intel had known about the bug but decided to ship the processor anyway. Intel claimed that under normal use, a typical consumer would only notice the problem once every 27,000 years. Unfortunately for Intel, Dr. Nicely had not been a normal user. Intel stonewalled for a while, but eventually they sent out replacement chips for everyone.

Fatal Exception

NEIL MCALLISTER

AUGUST 04, 2011

Oracle: Java's worst enemy

The buggy Java SE 7 release is only the latest misstep in a mounting litany of bad behavior

By Neil McAllister | InfoWorld

Oracle shipped Java SE 7 with a serious, showstopping bug, and who was the first to alert the Java community? The Apache Foundation. Oh, the irony.

This is the same Apache Foundation that resigned from the Java Community Process (JCP) executive committee in protest after Oracle repeatedly refused to give it access to the Java Technology Compatibility Kit (TCK).

[Neil McAllister reveals the most dangerous programming mistakes. | Get software development news and insights from InfoWorld's Developer World newsletter. | And sharpen your Java skills with the JavaWorld Enterprise Java newsletter.]

It's the same Apache Foundation that developed Harmony, an open source implementation of the Java platform. Google used Harmony to build its Android mobile OS, which is now the subject of a multi-billion-dollar lawsuit from Oracle alleging intellectual property violations. Oracle has subpoenaed documents from the Apache Foundation to help make its case. Nobody is sure what this means for other Harmony users.

Java.net

Home | Projects

Get Involved
- About Java.net
- Create a Project
- Java.net Enhancements

Get Informed
- Articles
- Blogs
- Site Wiki
- Events
- Oracle University

InfoWorld

WORLD

Sign in

News | Blog | White Pape

InfoWorld Home / Application Development /

Fatal Exceptio

NEIL MCAL

AUGUST 04, 2011

Oracle: Java's worst e

The buggy Java SE 7 release is o mounting litany of bad behavio

By Neil McAllister | InfoWorld

Print | 27 Comments

Oracle shipped Java SE 7 with a serious, show Java community? The Apache Foundation. Oh,

This is the same Apache Foundation that resig executive committee in protest after Oracle rep Technology Compatibility Kit (TCK).

[Neil McAllister reveals the most dangerous p development news and insights from InfoWor your Java skills with the JavaWorld Enterprise

It's the same Apache Foundation that developed Harmony, an open source implementation of the Java platform. Google used Harmony to build its Android mobile OS, which is now the subject of a multi-billion-dollar lawsuit from Oracle alleging intellectual property violations. Oracle has subpoenaed documents from the Apache Foundation to help make its case. Nobody is sure what this means for other Harmony users.

Dr.Dobb's

THE WORLD OF SOFTWARE DEVELOPMENT

Search

Search: Site Source Code

Home | Articles | News | Blogs | Source Code | Dobbs on DVD | Dobbs TV | Webinars

CHANNELS
- Architecture & Design
- C/C++
- Database
- Development Tools
- Embedded Systems
- High Performance Computing
- > Java
- Jolt Awards
- Mobility
- Open Source
- Security
- Web Development
- Windows/.NET
- Visual Studio 2010

INFO-LINK
- Threading? GPU load in Windows? Sound important?

JAVA

Tweet 89 | Gefällt mir | Share | +1 | Permalink

Sloppy Work at Oracle

By Andrew Binstock, August 01, 2011

5 Comments

Poor testing and bad decision-making mar an important release

Within a few days of my editorial suggesting that Java 7 be adopted quickly, news began to leak out that there were showstopper bugs in the Java 7 HotSpot compiler. I'll get into the defects shortly, but what really turned up the heat was Oracle's decision to ship the compiler aware that the known defects would cause one of two types of errors: hang the program or silently generate incorrect results. Given that Java 7 took five years to see light, it seems to me and many others that Oracle could have waited a bit longer to fix the bug before releasing the software. To a large extent, there is a feeling in the Java community that Oracle does not understand Java (despite the company's earlier acquisition of BEA). That may or may not be, but I would have expected it to understand enterprise software enough not to ship a compiler with defects that hang a valid program.

The problem, from what is known so far, derives from a command-line optimization switch on the Java compiler. This switch incorrectly optimized loops, resulting in the various reported errors.


JAVA
SOA
EBIZQ
TheServerSide.COM
Your Enterprise Java Community
DESIGN/ARCHITECTURE
EJB
WEB SERVICES
WEB APPLICATIONS
SEARCH
Get the most from your data in the cloud with the Cloud Data Architecture Quick Guide from SearchSOA.com.
Home > Discussions > News > Lucene should just shut up about Java 7
Discussions
News: Lucene should just shut up about Java 7
SHARE
GET THREAD FEED
Lucene should just shut up about Java 7 (24 messages)
POSTED BY: Richard Mayhew POSTED ON: August 05 2011 12:00 EDT
When Java 7 was released last week, Lucene and Solr both issued a warning saying that you can't use Java 7 with Lucene or Solr. These are popular text search libraries (and Solr is an app) so this can be pretty severe, and the way the community was informed at first was pretty emotional. See Java 7 paralyses Lucene and Solr for an example of the kind of headline generated.
The actual warnings on the Apache sites are pretty mild and state the problem although not the history of the problem well, and give an expectation of when you'll be able to use Lucene with Java 7.
Some people have written some pretty good responses to the issue, see Don't Use Java 7? Are you kidding me? for one example.
These do a good job of hiding the emotions from the problem; in mailing lists and other forums, the word is that Java 7 is buggy, that Oracle's ignoring the problem (they're not, they are working on the bug and have a fix scheduled) and that Oracle didn't test enough.
I say that's crap. Sure, it'd be nice if Oracle never released a JVM with bugs, but it's silly to blame them for this.
There's enough blame for everyone, for Lucene and Oracle.
Oracle honestly did the right thing, though. They had some optimizations in Java 6 (-XX:+OptimizeStringConcat and -XX:+AggressiveOpts) that could break some code, like all things can. With Java 7, they made the optimizations on by default, starting a month before Java 7's release.
Search
Search: Site Source Code
Blogs Source Code Dobbs on DVD Dobbs TV Webinars
esri
Your Maps
Your World
Tweet 89 Gefällt mir Share Permalink
loppy Work at Oracle
Andrew Binstock, August 01, 2011
Comments
or testing and bad decision-making mar an important release
thin a few days of my editorial suggesting that Java 7 be opted quickly, news began to leak out that there were owstopper bugs in the Java 7 HotSpot compiler. I'll get into the fects shortly, but what really turned up the heat was Oracle's cision to ship the compiler aware that the known defects would use one of two types of errors: hang the program or silently nerate incorrect results. Given that Java 7 took five years to see ht, it seems to me and many others that Oracle could have aited a bit longer to fix the bug before releasing the software. To arge extent, there is a feeling in the Java community that Oracle es not understand Java (despite the company's earlier acquisition BEA). That may or may not be, but I would have expected it to derstand enterprise software enough not to ship a compiler with fects that hang a valid program.
e problem, from what is known so far, derives from a command- e optimization switch on the Java compiler. This switch correctly optimized loops, resulting in the various reported errors.
ntation of the
Java platform. Google used Harmony to build its Android mobile OS, which is now the subject of a multi-billion-dollar lawsuit from Oracle alleging intellectual property violations. Oracle has subpoenaed documents from the Apache Foundation to help make its case. Nobody is sure what this means for other Harmony users.
Leading the W
of Open Sourc
Lucene

Of communities, companies, and bugs (Or, "Dr Dobbs Journal is a slut!")

18 ZONE IT 4 DZone
Submitted by Ted Neward on Sat, 2011/08/06 - 11:25am
Tags: Java Oracle Agile

Andrew Binstock (Editor-in-Chief at DDJ) has taken a shot at Oracle's Java7 release, and I found myself feeling a need to respond.

In his article, Andrew notes that

> ... what really turned up the heat was Oracle's decision to ship the compiler aware that the known defects would cause one of two types of errors: hang the program or silently generate incorrect results. Given that Java 7 took five years to see light, it seems to me and many others that Oracle could have waited a bit longer to fix the bug before releasing the software. To a large extent, there is a feeling in the Java community that Oracle does not understand Java (despite the company's earlier acquisition of BEA). That may or may not be, but I would have expected it to understand enterprise software enough not to ship a compiler with defects that hang a valid program.

We Recommend These Resources
- Enterprise Integration Patterns: Past, Present and Future
- Open Source as a Catalyst for Innovation and Cultural Change featuring Yahoo!
- The Future of Camel
- Migrating to FUSE Mediation Router
- Apache Camel – How to go from EIPs to production

There's so many things in this paragraph alone I want to respond to, I feel it necessary to deconstruct it and respond individually:

this means for other Harmony users.

ppy Work at Oracle

ndrew Binstock, August 01, 2011

Comments

r testing and bad decision-making mar an important release

in a few days of my editorial suggesting that Java 7 be ted quickly, news began to leak out that there were vstopper bugs in the Java 7 HotSpot compiler. I'll get into the cts shortly, but what really turned up the heat was Oracle's sion to ship the compiler aware that the known defects would e one of two types of errors: hang the program or silently rate incorrect results. Given that Java 7 took five years to see , it seems to me and many others that Oracle could have ed a bit longer to fix the bug before releasing the software. To ge extent, there is a feeling in the Java community that Oracle not understand Java (despite the company's earlier acquisition EA). That may or may not be, but I would have expected it to rstand enterprise software enough not to ship a compiler with cts that hang a valid program.

problem, from what is known so far, derives from a command- ptimization switch on the Java compiler. This switch rrectly optimized loops, resulting in the various reported errors.

ation of the
subject of a
as
sure what

Of communities, companies, ar "Dr Dobbs Journal is a slut!")

18 ZONE IT 4

DZone

Submitted by Ted Neward on Sat, 2011/08/06 - 11:25am

Tags: Java Oracle Agile

Andrew Binstock (Editor-in-Chief at DDJ) has taken a shot at Oracle's Java7 release, and I found myself feeling a need to respond.

In his article, Andrew notes that

> ... what really turned up the heat was Oracle's decision to ship the compiler aware that the known defects would cause one of two types of errors: hang the program or silently generate incorrect results. Given that Java 7 took five years to see light, it seems to me and many others that Oracle could have waited a bit longer to fix the bug before releasing the software. To a large extent, there is a feeling in the Java community that Oracle does not understand Java (despite the company's earlier acquisition of BEA). That may or may not be, but I would have expected it to understand enterprise software enough not to ship a compiler with defects that hang a valid program.

There's so many things in this paragraph alone I want to respond to, I feel it necessary individually:

this means for other Harmony users.

We Recommend T
Enterprise Inte and Future
Open Source a Cultural Chang
The Future of C
Migrating to FU
Apache Camel production

August 10, 2011

RELATED NEWS BOOKS VIDEOS EVENTS

Java 7 Debate Rages On

Java 7 Bugs: Should the Release Have Been Delayed?

Java 7 may have brought with it some useful (and long-awaited) updates for the Java community, but it has also sparked controversy as the release shipped with some **bugs in the Java 7 HotSpot compiler**. These bugs affect all currently released versions of Apache Lucene Core and Apache Solr, but the problem could also affect Java 6 users, if they use one of the JVM options that are not enabled by default:

-XX:+OptimizeStringConcat or
-XX:+AggressiveOpts

These bugs were discovered five days before Java 7 was published, which has caused some to question whether the release should have been delayed. At his blog, Uwe Schindler, who first posted about the bug, has drawn attention to the fact that the final release of Java 7, is the same as the preview release, and has questioned the point of the preview release. "It was for sure not intended for public review and bug hunting!" he says. Others, such as Markus Eisele, have defended Oracle, stressing that: "these problems were detected only 5 days before the official Java 7 release, so Oracle had no time to fix those bugs."

Andrew Binstock, Executive Editor of Dr. Dobb's, has **posted his thoughts on the controversy**, referring to the HotSpot compiler problems as "showstopper bugs" and stating that Oracle should have delayed the release. He goes on to claim that there is a "feeling in the

Le
of Open Sourc

Java 7 Crashes Eclipse…

THE PORTER STEMMER SIGSEGV BUG

What's wrong with these methods?

```
private final void step4() {
  switch (b[k]) {
    case 'e': if (ends("icate")) { r("ic"); break; }
              if (ends("ative")) { r(""); break; }
              if (ends("alize")) { r("al"); break; }
              break;
    case 'i': if (ends("iciti")) { r("ic"); break; }
              break;
    case 'l': if (ends("ical")) { r("ic"); break; }
              if (ends("ful")) { r(""); break; }
              break;
    case 's': if (ends("ness")) { r(""); break; }
              break;
  }
}
```

```
private final boolean ends(String s) {
    int l = s.length();
    int o = k-l+1;
    if (o < 0) return false;
    for (int i = 0; i < l; i++) {
      if (b[o+i] != s.charAt(i)) return false;
    }
    j = k-l;
    return true;
}
```

Conclusion: Porter Stemmer Bug

- Less serious bug as your virtual machine simply crashes. You won't use it!
- Oracle made bug report "serious", as this affects their software, reproducible to everyone.
- Can be prevented by JVM option: `-XX:-UseLoopPredicate`

Loop Unwinding

THE VINT BUG

What's wrong with this method?

```
/** Reads an int stored in variable-length format.  Reads between one and
 * five bytes.  Smaller values take fewer bytes.  Negative numbers are not
 * supported.
 * @see IndexOutput#writeVInt(int)
 */
public int readVInt() throws IOException {
  byte b = readByte();
  int i = b & 0x7F;
  for (int shift = 7; (b & 0x80) != 0; shift += 7) {
    b = readByte();
    i |= (b & 0x7F) << shift;
  }
  return i;
}
```

What's wrong with this method?

```
/** Reads an int stored in variable-length format.  Reads between one and
 * five bytes.  Smaller values take fewer bytes.  Negative numbers are not
 * supported.
 * @see DataOutput#writeVInt(int)
 */
public int readVInt() throws IOException {
  byte b = readByte();
  if (b >= 0) return b;
  int i = b & 0x7F;
  b = readByte();
  i |= (b & 0x7F) << 7;
  if (b >= 0) return i;
  b = readByte();
  i |= (b & 0x7F) << 14;
  if (b >= 0) return i;
  b = readByte();
  i |= (b & 0x7F) << 21;
  if (b >= 0) return i;
  b = readByte();
  // Warning: the next ands use 0x0F / 0xF0 - beware copy/paste errors:
  i |= (b & 0x0F) << 28;
  if ((b & 0xF0) == 0) return i;
  throw new IOException("Invalid vInt detected (too many bits)");
}
```

Conclusion: Vint Bug

- **Serious data corruption:** Some methods using loops silently return wrong results!
- Bug already existed in Java 6
  - appeared some time after 1.6.0_18, enabled by default
  - is prevented since Lucene 3.1 by manual loop unwinding *(helps only in Java 6)*
- Cannot easily be reproduced, Oracle assigned “medium” bug priority – was never fixed in Java 6.
- **Problems got worse with Java 7**, only safe way to prevent is to disable loop unwinding completely, but that makes Lucene very slow.

3 Answers

active | oldest | votes

55

The problem with any hotspot bugs, is that you need to reach the compilation threshold (e.g. 10000) before it can get you: so if your unit tests are "trivial", you probably won't catch it.

For example, we caught the incorrect results issue in lucene, because this particular test creates 20,000 document indexes.

In our tests we randomize different interfaces (e.g. different Directory implementations) and indexing parameters and such, and the test only fails 1% of the time, of course its then reproducable with the same random seed. We also run checkindex on every index that tests create, which do some sanity tests to ensure the index is not corrupt.

For the test we found, if you have a particular configuration: e.g. RAMDirectory + PulsingCodec + payloads stored for the field, then after it hits the compilation threshold, the enumeration loop over the postings returns incorrect calculations, in this case the number of returned documents for a term != the docFreq stored for the term.

We have a good number of stress tests, and its important to note the normal assertions in this test actually pass, its the checkindex part at the end that fails.

The big problem with this, is that lucene's incremental indexing fundamentally works by merging multiple segments into one: because of this, if these enums calculate invalid data, this invalid data is then *stored* into the newly merged index: aka corruption.

I'd say this bug is much sneakier than previous loop optimizer hotspot bugs we have hit (e.g. sign-flip stuff, https://issues.apache.org/jira/browse/LUCENE-2975). In that case we got wacky negative document deltas, which make it easy to catch. We also only had to manually unroll a single method to dodge it. On the other hand, the only "test" we had initially for that was a huge 10GB index of http://www.pangaea.de/, so it was painful to narrow it down to this bug.

In this case, I spent a good amount of time (e.g. every night last week) trying to manually unroll/inline various things, trying to create some workaround so we could dodge the bug and not have the possibility of corrupt indexes being created. I could dodge some cases, but there were many more cases I couldn't... and I'm sure if we can trigger this stuff in our tests there are more cases out there...

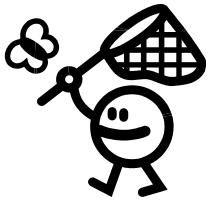
link | improve this answer

answered **Aug 1 at 4:27**

Robert Muir

940 ●4 ●8

Hands-On



HOW TO DEBUG HOTSPOT PROBLEMS

First…

- Fetch some beer!
- Tell your girlfriend that you will not come to bed!
- Forget about Eclipse & Co! We need a command line and our source code…

Hardcore: **Debugging *without* Debugger**

- Open `hs_err` file and watch for stack trace.
  (if your JVM crashed like in Porter stemmer)
- *Otherwise:* disable Hotspot to verify that it's not a logic error! (`-Xint` / `-Xbatch`)
- Start to dig around by adding `System.out.println`, assertions,...
  Please note: You cannot use a debugger!!!

Hardcore: **Debugging *without* Debugger**

- Open `hs_err` file and watch for stack trace.
  (if your JVM crashed like in Porter stemmer)
- *Otherwise:* disable Hotspot to verify that it's not a logic error! (`-Xint` / `-Xbatch`)
- Start to dig around by adding `System.out.println`, assertions,...
  Please note: You cannot use a debugger!!!

Digging…

- If you found a method that works incorrectly, disable Hotspot optimizations for only that one:
  `-XX:CompileCommand=exclude,your/package/Class,method`
  - If program works now, you found a workaround!
  - But this may not be the root cause - does not help at all!

- Step down the call hierarchy and replace exclusion by methods called from this one.

Take action!

Open a bug report at Oracle!

Inform

hotspot-compiler-dev@openjdk.java.net

mailing list.

Setting up Jenkins

TESTING SOFTWARE ON VARIOUS JVM VENDORS

Randomization everywhere

- **Apache Lucene & Solr** use randomization while testing:
  - Random codec settings
  - Random Lucene directory implementation
  - Random locales, default charsets,…
  - Random indexing data

Randomization everywhere

- **Apache Lucene & Solr** use randomization while testing:
  - Random codec settings
  - Random Lucene directory implementation
  - Random locales, default charsets,…
  - Random indexing data
- **Reproducible:**
  - Every test gets an initial random seed
  - Printed on test execution & included in stack traces

Missing parts

- JVM randomization
  - Oracle JDK 6 / 7
  - IBM J9 6 / 7
  - Oracle JRockit 6

Missing parts

- JVM randomization
  - Oracle JDK 6 / 7
  - IBM J9 6 / 7
  - Oracle JRockit 6
- JVM settings randomization
  - Garbage collector
  - Bitness: 32 / 64 bits
  - Server / Client VM
  - Compressed OOPs *(ordinary object pointer)*

Missing parts

- JVM randomization
  - Oracle JDK 6 / 7
  - IBM J9 6 / 7
  - Oracle JRockit 6
- JVM settings randomization
  - Garbage collector
  - Bitness: 32 / 64 bits
  - Server / Client VM
  - Compressed OOPs *(ordinary object pointer)*
- Platform
  - Linux, Windows, MacOS X, FreeBSD,…

Possibilities

- Define each Jenkins job with a different JVM:
  - Duplicates
  - Hard to maintain
  - Multiplied by additional JVM settings like GC, server/client, or OOP size

Possibilities

- Define each Jenkins job with a different JVM:
  - Duplicates
  - Hard to maintain
  - Multiplied by additional JVM settings like GC, server/client, or OOP size
- Make Jenkins server set build / environment variables with a (pseudo-)randomization script:
  - `$JAVA_HOME` → passed to Apache Ant
  - `$TEST_JVM_ARGS` → passed to test runner

Plugins needed

- Environment Injector Plugin
  - Executes Groovy script to do the actual work
  - Sets some build environment variables: `$JAVA_HOME, $TEST_JVM_ARGS, $JAVA_DESC`

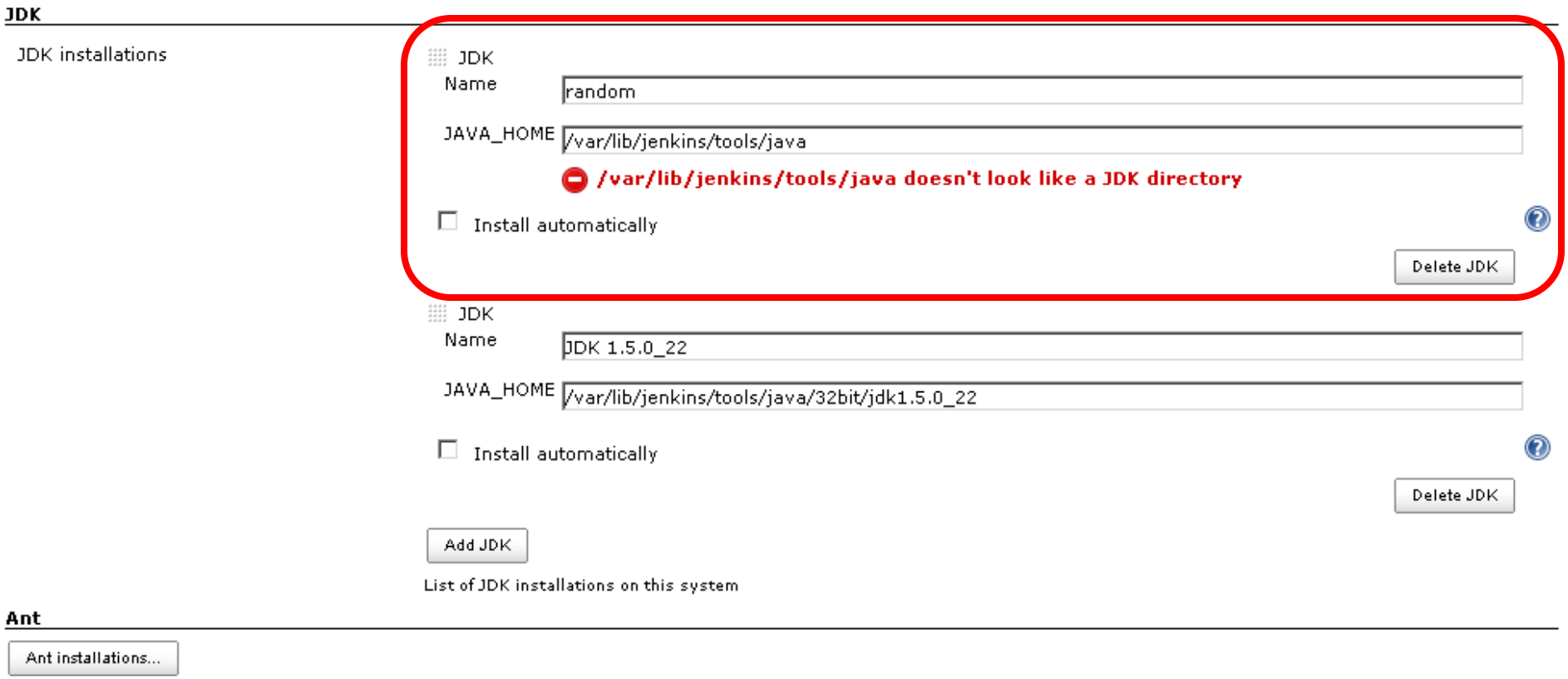
Plugins needed

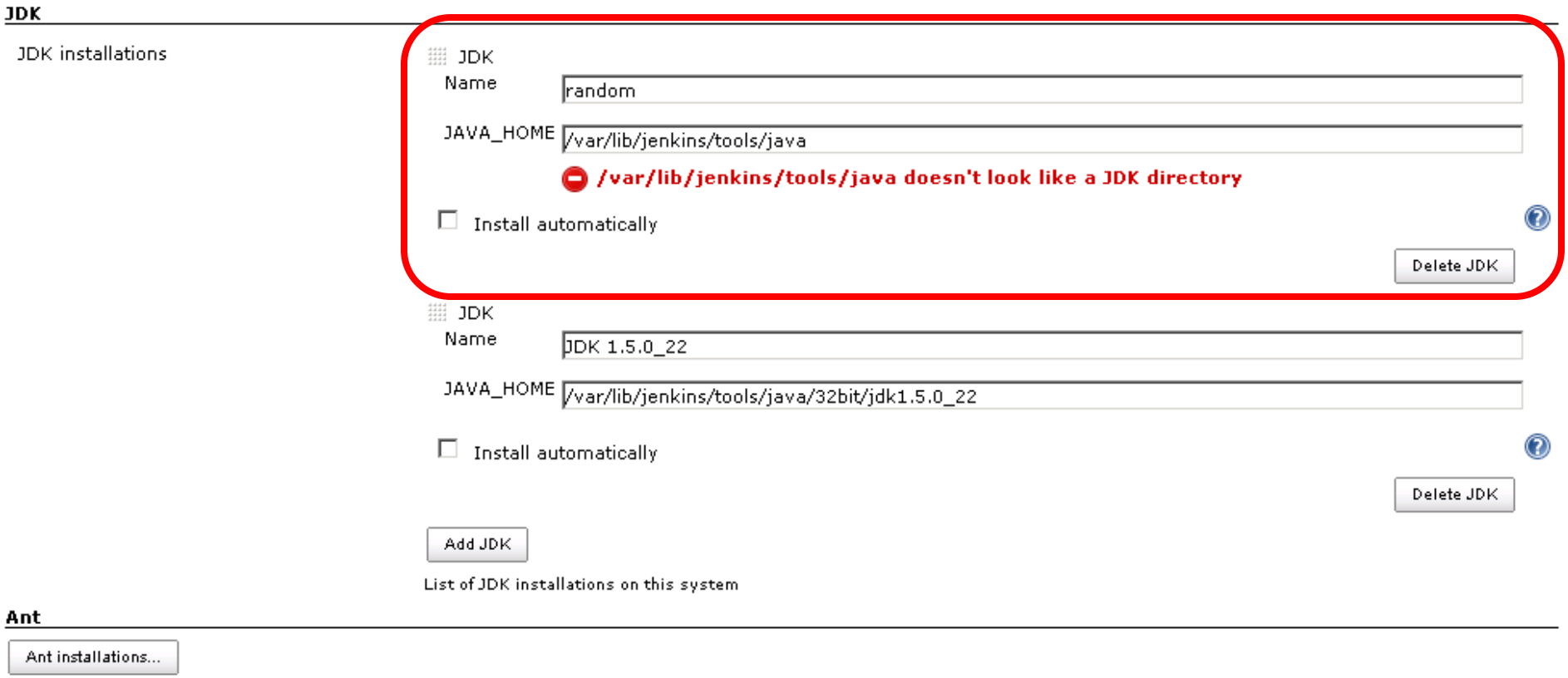
- Environment Injector Plugin
  - Executes Groovy script to do the actual work
  - Sets some build environment variables: `$JAVA_HOME, $TEST_JVM_ARGS, $JAVA_DESC`

- Jenkins Description Setter Plugin / Jenkins Email Extension Plugin
  - Add JVM details / settings to build description and e-mails

Global Jenkins settings

- Extra JDK config in Jenkins (called “random”):
  - pointing to *dummy* directory *(we can use the base directory containing all our JDKs)*
  - Assigned to every job that needs a randomly choosen virtual machine


JDK
JDK installations
JDK
Name
random
JAVA_HOME
/var/lib/jenkins/tools/java
/var/lib/jenkins/tools/java doesn't look like a JDK directory
Install automatically
Delete JDK
JDK
Name
JDK 1.5.0_22
JAVA_HOME
/var/lib/jenkins/tools/java/32bit/jdk1.5.0_22
Install automatically
Delete JDK
Add JDK
List of JDK installations on this system
Ant
Ant installations...



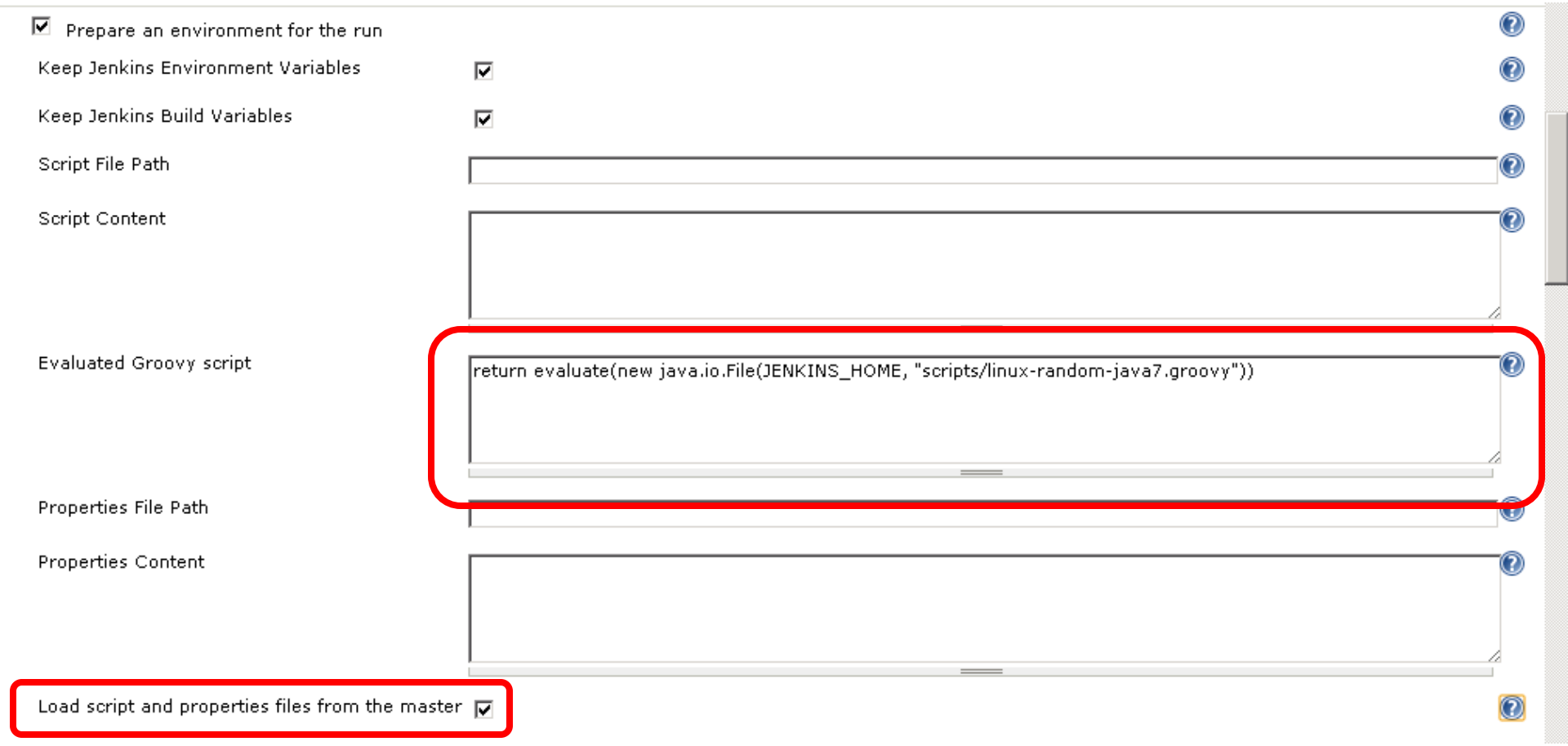
The warning displayed by Jenkins doesn't matter!

Job Config

- Standard **free style build** with plugins activated
  - Calls Groovy script file with main logic (sets `$JAVA_HOME` randomly,…)
  - List of JVM options as a „config file“
  - Job‘s JDK version set to „random“
  - Apache Ant configuration automatically gets `$JAVA_HOME` and test runner gets extra options via build properties

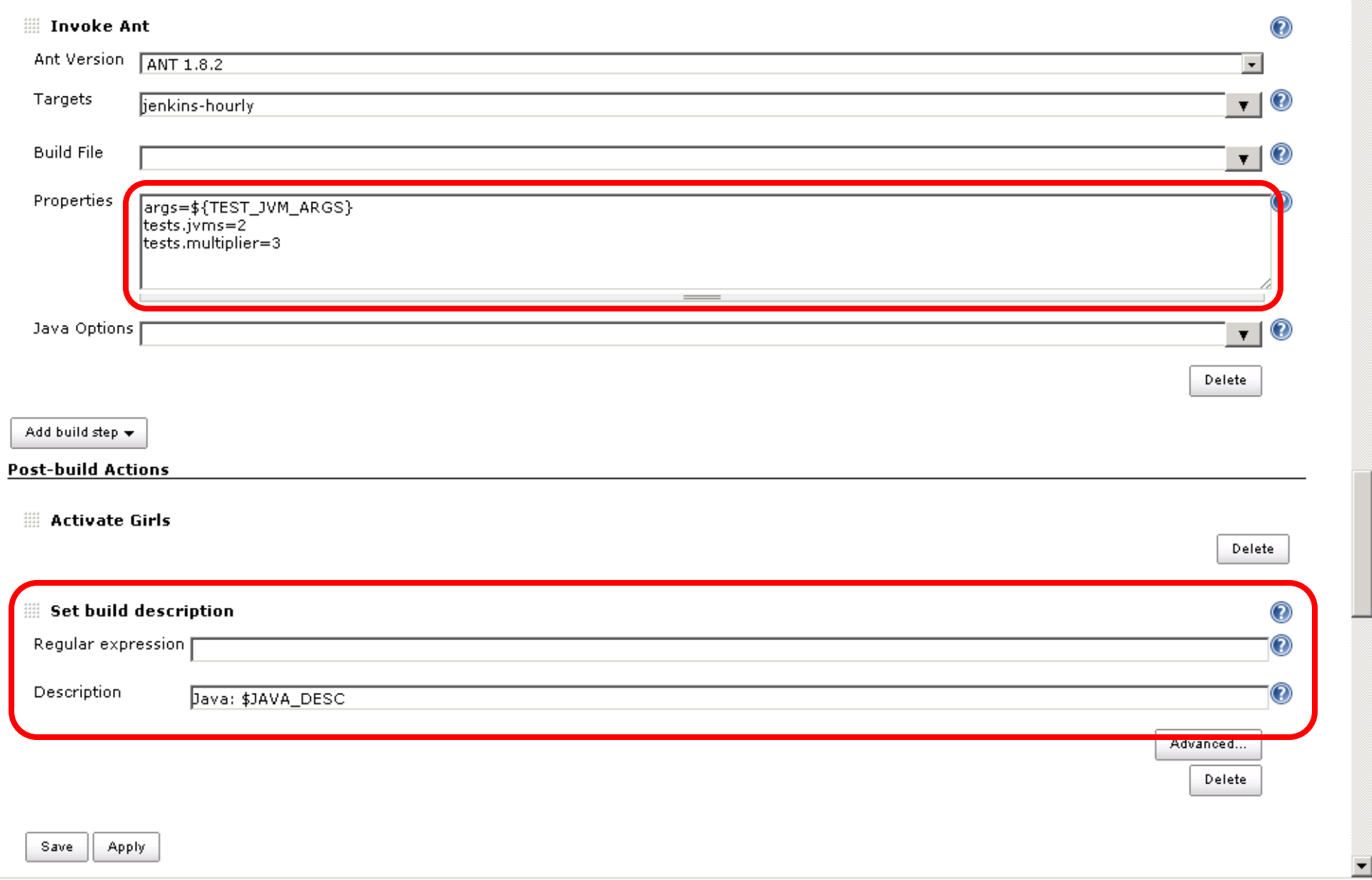
Job Config

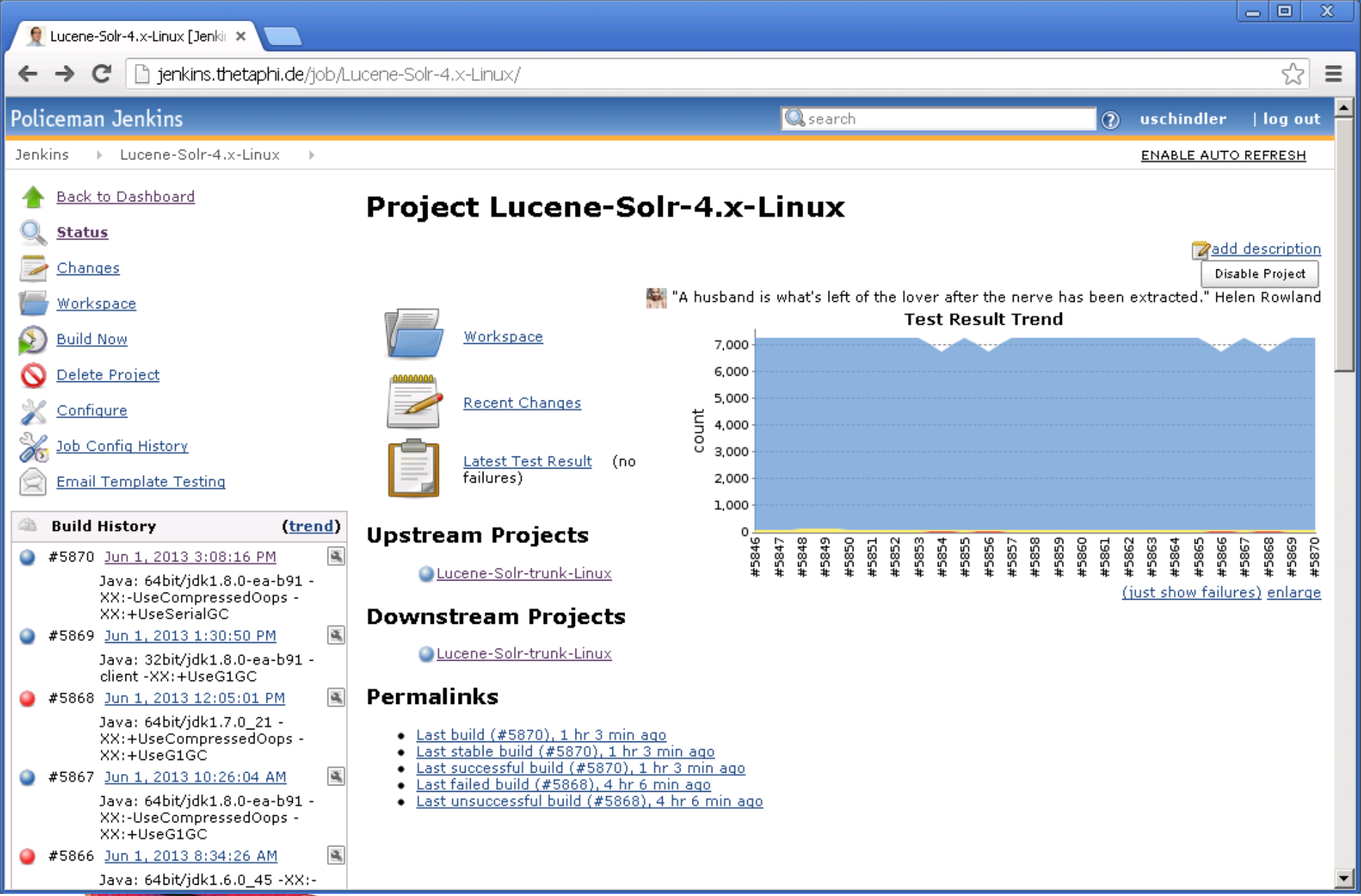
- Standard **free style build** with plugins activated
  - Calls Groovy script file with main logic (sets `$JAVA_HOME` randomly,…)
  - List of JVM options as a „config file“
  - Job‘s JDK version set to „random“
  - Apache Ant configuration automatically gets `$JAVA_HOME` and test runner gets extra options via build properties
- Should work with Maven builds, too!


Prepare an environment for the run
Keep Jenkins Environment Variables
Keep Jenkins Build Variables
Script File Path
Script Content
Evaluated Groovy script
return evaluate(new java.io.File(JENKINS_HOME, "scripts/linux-random-java7.groovy"))
Properties File Path
Properties Content
Load script and properties files from the master

linux-random-java7.groovy

```groovy
def separator = "/"
def JDKs = [
  [JAVA: "32bit/jdk1.7.0_21", TEST_JVM_ARGS: "-client -XX:+UseSerialGC"],
  [JAVA: "32bit/jdk1.7.0_21", TEST_JVM_ARGS: "-server -XX:+UseSerialGC"],
  [JAVA: "64bit/jdk1.7.0_21", TEST_JVM_ARGS: "-XX:+UseCompressedOops -XX:+UseSerialGC"],
  [JAVA: "64bit/jdk1.7.0_21", TEST_JVM_ARGS: "-XX:-UseCompressedOops -XX:+UseSerialGC"],
  [JAVA: "32bit/jdk1.7.0_21", TEST_JVM_ARGS: "-client -XX:+UseParallelGC"],
  [JAVA: "32bit/jdk1.7.0_21", TEST_JVM_ARGS: "-server -XX:+UseParallelGC"],
  [JAVA: "64bit/jdk1.7.0_21", TEST_JVM_ARGS: "-XX:+UseCompressedOops -XX:+UseParallelGC"],
  [JAVA: "64bit/jdk1.7.0_21", TEST_JVM_ARGS: "-XX:-UseCompressedOops -XX:+UseParallelGC"],
  [JAVA: "32bit/jdk1.7.0_21", TEST_JVM_ARGS: "-client -XX:+UseConcMarkSweepGC"],
  [JAVA: "32bit/jdk1.7.0_21", TEST_JVM_ARGS: "-server -XX:+UseConcMarkSweepGC"],
  [JAVA: "64bit/jdk1.7.0_21", TEST_JVM_ARGS: "-XX:+UseCompressedOops -XX:+UseConcMarkSweepGC"],
  [JAVA: "64bit/jdk1.7.0_21", TEST_JVM_ARGS: "-XX:-UseCompressedOops -XX:+UseConcMarkSweepGC"],
  [JAVA: "32bit/jdk1.7.0_21", TEST_JVM_ARGS: "-client -XX:+UseG1GC"],
  [JAVA: "32bit/jdk1.7.0_21", TEST_JVM_ARGS: "-server -XX:+UseG1GC"],
  [JAVA: "64bit/jdk1.7.0_21", TEST_JVM_ARGS: "-XX:+UseCompressedOops -XX:+UseG1GC"],
  [JAVA: "64bit/jdk1.7.0_21", TEST_JVM_ARGS: "-XX:-UseCompressedOops -XX:+UseG1GC"],

  [JAVA: "32bit/jdk1.8.0-ea-b91", TEST_JVM_ARGS: "-client -XX:+UseSerialGC"],
  // ...

  [JAVA: "32bit/jrockit-jdk1.6.0_37-R28.2.5-4.1.0", TEST_JVM_ARGS: "-XnoOpt"],
  [JAVA: "64bit/jrockit-jdk1.6.0_37-R28.2.5-4.1.0", TEST_JVM_ARGS: "-XnoOpt"],

  [JAVA: "64bit/ibm-j9-jdk6", TEST_JVM_ARGS: "-Xjit:exclude={org/apache/lucene/util/fst/FST.pack(IIF)Lorg/apache/lucene/util/fst/FST;}"],
  [JAVA: "32bit/ibm-j9-jdk6", TEST_JVM_ARGS: "-Xjit:exclude={org/apache/lucene/util/fst/FST.pack(IIF)Lorg/apache/lucene/util/fst/FST;}"],
  [JAVA: "64bit/ibm-j9-jdk7", TEST_JVM_ARGS: "-Xjit:exclude={org/apache/lucene/util/fst/FST.pack(IIF)Lorg/apache/lucene/util/fst/FST;}"],
  [JAVA: "32bit/ibm-j9-jdk7", TEST_JVM_ARGS: "-Xjit:exclude={org/apache/lucene/util/fst/FST.pack(IIF)Lorg/apache/lucene/util/fst/FST;}"],
]

def randomJdk = JDKs.get(new Random().nextInt(JDKs.size()))
def javaHome = JAVA_HOME + separator + randomJdk.get("JAVA").replace((char)'/', (char)separator)
randomJdk.put("JAVA_HOME", javaHome)
randomJdk.put("JAVA_DESC", randomJdk.get("JAVA") + " " + randomJdk.get("TEST_JVM_ARGS"))
randomJdk.put("PATH+JDK", javaHome + separator + "bin")
return randomJdk
```


Invoke Ant
Ant Version
ANT 1.8.2
Targets
jenkins-hourly
Build File
Properties
args=${TEST_JVM_ARGS}
tests.jvms=2
tests.multiplier=3
Java Options
Delete
Add build step
Post-build Actions
Activate Girls
Delete
Set build description
Regular expression
Description
Java: $JAVA_DESC
Advanced...
Delete
Save
Apply


Lucene-Solr-4.x-Linux [Jenki
jenkins.thetaphi.de/job/Lucene-Solr-4.x-Linux/
Policeman Jenkins
search
uschindler | log out
Jenkins ▸ Lucene-Solr-4.x-Linux ▸
ENABLE AUTO REFRESH
Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure
Job Config History
Email Template Testing
Build History (trend)
#5870 Jun 1, 2013 3:08:16 PM
Java: 64bit/jdk1.8.0-ea-b91 -XX:-UseCompressedOops -XX:+UseSerialGC
#5869 Jun 1, 2013 1:30:50 PM
Java: 32bit/jdk1.8.0-ea-b91 -client -XX:+UseG1GC
#5868 Jun 1, 2013 12:05:01 PM
Java: 64bit/jdk1.7.0_21 -XX:+UseCompressedOops -XX:+UseG1GC
#5867 Jun 1, 2013 10:26:04 AM
Java: 64bit/jdk1.8.0-ea-b91 -XX:-UseCompressedOops -XX:+UseG1GC
#5866 Jun 1, 2013 8:34:26 AM
Java: 64bit/jdk1.6.0_45 -XX:-
Project Lucene-Solr-4.x-Linux
add description
Disable Project
"A husband is what's left of the lover after the nerve has been extracted." Helen Rowland
Workspace
Recent Changes
Latest Test Result (no failures)
Test Result Trend
count
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
#5846 #5847 #5848 #5849 #5850 #5851 #5852 #5853 #5854 #5855 #5856 #5857 #5858 #5859 #5860 #5861 #5862 #5863 #5864 #5865 #5866 #5867 #5868 #5869 #5870
(just show failures) enlarge
Upstream Projects
Lucene-Solr-trunk-Linux
Downstream Projects
Lucene-Solr-trunk-Linux
Permalinks
Last build (#5870), 1 hr 3 min ago
Last stable build (#5870), 1 hr 3 min ago
Last successful build (#5870), 1 hr 3 min ago
Last failed build (#5868), 4 hr 6 min ago
Last unsuccessful build (#5868), 4 hr 6 min ago

Console Output

```
Started by upstream project "Lucene-Solr-trunk-Linux" build number 5934
originally caused by:
 Started by timer
 Started by upstream project "Lucene-Solr-4.x-Linux" build number 5869
 originally caused by:
  Started by upstream project "Lucene-Solr-trunk-Linux" build number 5933
  originally caused by:
   Started by timer
   Started by timer
[EnvInject] - Loading node environment variables.
[EnvInject] - Preparing an environment for the build.
[EnvInject] - Keeping Jenkins system variables.
[EnvInject] - Keeping Jenkins build variables.
[EnvInject] - Evaluation the following Groovy script content:
return evaluate(new java.io.File(JENKINS_HOME, "scripts/linux-random-java.groovy"))

[EnvInject] - Injecting contributions.
Building on master in workspace /var/lib/jenkins/workspace/Lucene-Solr-4.x-Linux
Cleaning up /var/lib/jenkins/workspace/Lucene-Solr-4.x-Linux/.


Updating http://svn.apache.org/repos/asf/lucene/dev/branches/branch_4x at revision '2013-06-01T15:08:16.123 +0000'
At revision 1488532
no change for http://svn.apache.org/repos/asf/lucene/dev/branches/branch_4x since the previous build
No emails were triggered.
[Lucene-Solr-4.x-Linux] $ /bin/sh -xe /tmp/hudson7861606418982274965.sh
+ echo Using JDK: 64bit/jdk1.8.0-ea-b91 -XX:-UseCompressedOops -XX:+UseSerialGC
Using JDK: 64bit/jdk1.8.0-ea-b91 -XX:-UseCompressedOops -XX:+UseSerialGC
+ /var/lib/jenkins/tools/java/64bit/jdk1.8.0-ea-b91/bin/java -XX:-UseCompressedOops -XX:+UseSerialGC -version
java version "1.8.0-ea"
Java(TM) SE Runtime Environment (build 1.8.0-ea-b91)
Java HotSpot(TM) 64-Bit Server VM (build 25.0-b33, mixed mode)
[Lucene-Solr-4.x-Linux] $ /var/lib/jenkins/tools/hudson.tasks.Ant_AntInstallation/ANT_1.8.2/bin/ant "-Dargs=-XX:-UseCompressedOops -
XX:+UseSerialGC" -Dtests.multiplier=3 -Dtests.jvms=2 jenkins-hourly
Buildfile: /mnt/ssd/jenkins/workspace/Lucene-Solr-4.x-Linux/build.xml

jenkins-hourly:

clean:

clean:
     [echo] Building solr...

clean:

-test-with-heapdumps-enabled:
     [echo] Java HotSpot(TM) 64-Bit Server VM: Enabling heap dumps on OutOfMemoryError to dir '/mnt/ssd/jenkins/workspace/Lucene-Solr-4.x-
Linux/heapdumps'.
```

Console Output

```
Started by upstream project "Lucene-Solr-trunk-Linux" build number 5934
originally caused by:
 Started by timer
 Started by upstream project "Lucene-Solr-4.x-Linux" build number 5869
 originally caused by:
  Started by upstream project "Lucene-Solr-trunk-Linux" build number 5933
  originally caused by:
   Started by timer
   Started by timer
[EnvInject] - Loading node environment variables.
[EnvInject] - Preparing an environment for the build.
[EnvInject] - Keeping Jenkins system variables.
[EnvInject] - Keeping Jenkins build variables.
[Lucene-Solr-4.x-Linux] $ /bin/sh -xe /tmp/hudson7861606418982274965.sh
+ echo Using JDK: 64bit/jdk1.8.0-ea-b91 -XX:-UseCompressedOops -XX:+UseSerialGC
Using JDK: 64bit/jdk1.8.0-ea-b91 -XX:-UseCompressedOops -XX:+UseSerialGC
+ /var/lib/jenkins/tools/java/64bit/jdk1.8.0-ea-b91/bin/java -XX:-UseCompressedOops -XX:+UseSerialGC -version
java version "1.8.0-ea"
Java(TM) SE Runtime Environment (build 1.8.0-ea-b91)
Java HotSpot(TM) 64-Bit Server VM (build 25.0-b33, mixed mode)
[Lucene-Solr-4.x-Linux] $ /var/lib/jenkins/tools/hudson.tasks.Ant_AntInstallation/ANT_1.8.2/bin/ant "-Dargs=-X
XX:+UseSerialGC" -Dtests.multiplier=3 -Dtests.jvms=2 jenkins-hourly
Buildfile: /mnt/ssd/jenkins/workspace/Lucene-Solr-4.x-Linux/build.xml

jenkins-hourly:
java version "1.8.0-ea"
Java(TM) SE Runtime Environment (build 1.8.0-ea-b91)
Java HotSpot(TM) 64-Bit Server VM (build 25.0-b33, mixed mode)
[Lucene-Solr-4.x-Linux] $ /var/lib/jenkins/tools/hudson.tasks.Ant_AntInstallation/ANT_1.8.2/bin/ant "-Dargs=-XX:-UseCompressedOops -
XX:+UseSerialGC" -Dtests.multiplier=3 -Dtests.jvms=2 jenkins-hourly
Buildfile: /mnt/ssd/jenkins/workspace/Lucene-Solr-4.x-Linux/build.xml

jenkins-hourly:

clean:

clean:
     [echo] Building solr...

clean:

-test-with-heapdumps-enabled:
     [echo] Java HotSpot(TM) 64-Bit Server VM: Enabling heap dumps on OutOfMemoryError to dir '/mnt/ssd/jenkins/workspace/Lucene-Solr-4.x-
Linux/heapdumps'.
```

Results

BUGS FOUND

Oracle (Hotspot) JVM

- Various issues with JIT compilation around all OpenJDK / Oracle JDK versions:
  - Miscompiled loops
  - Segmentation faults
  - `System.nanotime()` brokenness on MacOSX
  - Double `free()`
- Lucene bugs with memory allocations if compressed oops are disabled on 64bit JVMs
  - happens only with large heaps > 32 GB

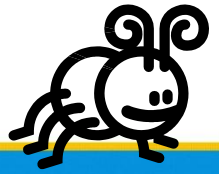
Java 8 prereleases

- G1 garbage collector deadlock due to marking stack overflow *(fixed)*
- Compile failures with `-source 1.7` related to default interface methods ("`isAnnotationPresent`") *(fixed)*
- Javadoc bugs
  - new doclint feature did not work *(fixed)*
  - doc-files folders were not copied *(fixed)*

Java 8 prereleases

- G1 garbage collector deadlock due to marking stack overflow *(fixed)*
- Compile failures with `-source 1.7` related to default interface methods ("`isAnnotationPresent`") *(fixed)*
- Javadoc bugs
  - new doclint feature did not work *(fixed)*
  - doc-files folders were not copied *(fixed)*

- Solr test bugs with cool new Nashorn Javascript engine *(fixed in Solr tests)*



Oracle JRockit

- `TestPostingsOffsets#testBackwardsOffsets` fails in assertion in core Lucene code
  - JVM “ignores” an `if`-statement
  - `IndexWriter` later hits assertion
- No fix available by Oracle
  - Impossible to open a bug report without support contract!
  - JRockit seems unsupported
  - No Java 7 version available anymore => *discontinued*
- Workaround: `-XnoOpt`
  - Slowdown => **better use supported Oracle Java 7**

Oracle JRockit

- `TestPostingsOffsets#testBackwardsOffsets` fails in assertion in core Lucene code
  - JVM “ignores” an `if`-statement
  - `IndexWriter` later hits assertion
- No fix available by Oracle
  - Impossible to open a bug report without support contract!
  - JRockit seems unsupported
  - No Java 7 version available anymore => *discontinued*
- Workaround: `-XnoOpt`
  - Slowdown => **better use supported Oracle Java 7**

Don’t use JRockit or WebLogic App Server

IBM J9

- `GrowableWriter#ensureCapacity()` fails in assertion in core Lucene code
  - FST#pack() passes wrong argument
- **Cause completely unknown!**
- Hard to debug
  - Happens with JIT, AOT and without any optimizer
  - Only happens if test is executed in whole test suite
- Workaround:

```
-Xjit:exclude={org/apache/lucene/util/fst/FST.pack(IIF)Lorg/apache/lucene/util/fst/FST;}
```

IBM J9

- `GrowableWriter#ensureCapacity()` fails in assertion in core Lucene code
  - FST#pack() passes wrong argument
- **Cause completely unknown!**
- Hard to debug
  - Happens with JIT, AOT and without any optimizer
  - Only happens if test is executed in whole test suite
- Workaround:

```
-Xjit:exclude={org/apache/lucene/util/fst/FST.pack(IIF)Lorg/apache/lucene/util/fst/FST;}
```

Don't use IBM J9

(Warning: Installed on SUSE Enterprise Linux by default)

How about OpenJDK?

- Version numbers are inconsistent to official Oracle Java!
- Ubuntu 12 still installs **OpenJDK 7b147**, *but patched!*
- **OpenJDK 6** is **very** different to **Oracle JDK 6**:
  - Forked from early Java 7!
  - Not all patches applied: e.g., `ReferenceQueue#poll()` does not use double checked locking

How about OpenJDK?

- Version numbers are inconsistent to official Oracle Java!
- Ubuntu 12 still installs **OpenJDK 7b147**, *but patched!*
- **OpenJDK 6** is **very** different to **Oracle JDK 6**:
  - Forked from early Java 7!
  - Not all patches applied: e.g., `ReferenceQueue#poll()` does not use double checked locking

You may use OpenJDK 7

(if you understand version numbers and their relation to Oracle's update packages)

How about OpenJDK?

- Version numbers are inconsistent to official Oracle Java!
- Ubuntu 12 still installs **OpenJDK 7b147**, *but patched!*
- **OpenJDK 6** is **very** different to **Oracle JDK 6**:
  - Forked from early Java 7!
  - Not all patches applied: e.g., `ReferenceQueue#poll()` does not use double checked locking

You may use OpenJDK 7

(if you understand version numbers and their relation to Oracle's update packages)

Don't use OpenJDK 6

Inform yourself about further bugs:

http://wiki.apache.org/lucene-java/JavaBugs