JVM troubleshooting

Jean-Philippe BEMPEL
Performance Architect

@jpbempel
http://jpbempel.blogspot.com

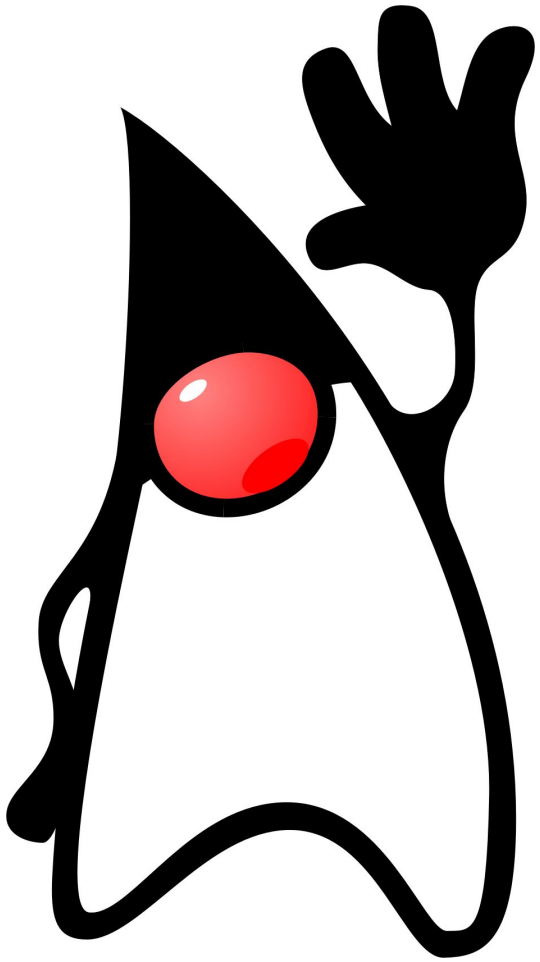
Ullink
CONNECT. TRADE.

Agenda

- JVM Startup
- CPU/Threads
- Memory/GC
- JVM crash

ullink

JVM Startup



JVM options

- Starts JVM with Options
- Lots of options available (~720-860)
  `-XX:+PrintFlagsFinal`
- Useful for setup, diagnostic & tuning
- Most options are only set at startup

jinfo

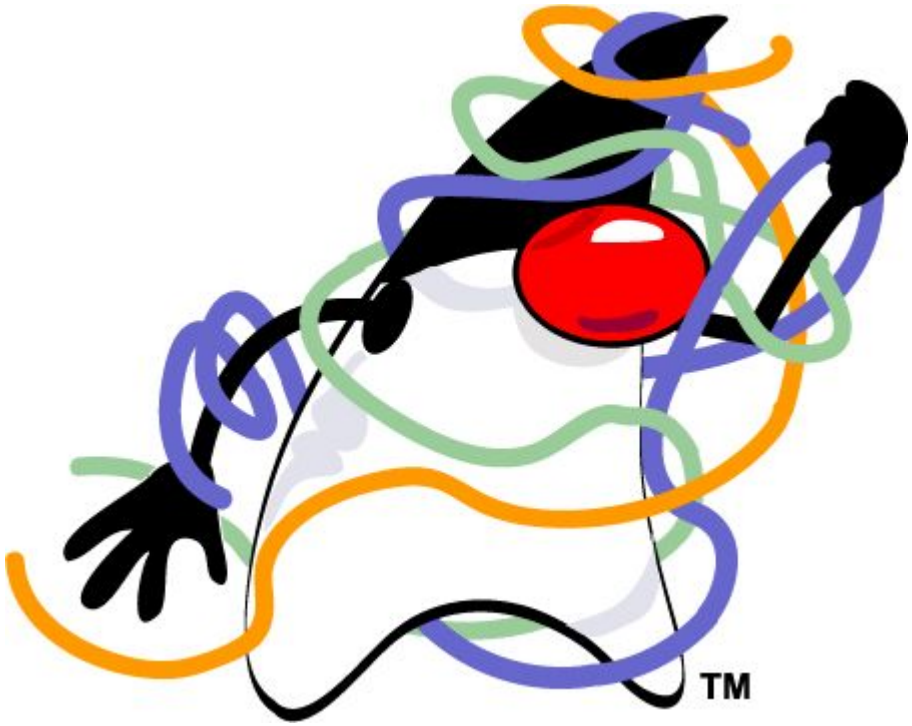
- tool included in the JDK
- provides configuration information of a running JVM
  - system properties
  - VM Flags
  - command line
- Can also be done with jcmd tool

```
jcmd <pid> VM.flags
jcmd <pid> VM.commandline
jcmd <pid> VM.systemproperties
```

manageable VM options

- some VM options are called manageable
  `java -XX:+PrintFlagsFinal -version | grep manageable`
- means you can change their value at runtime
- `jinfo -flag +PrintGCDetails <pid>`
- use JMX MBean
  com.sun.management.HotSpotDiagnostic setVMOption

Threads



Thread dump

- JVM stuck or CPU at 100% => get a thread dump
- `jstack <pid>`
- `kill -3 <pid>`
  (will print in stdout of the JVM!)
- `jcmd <pid> Thread.print`

Thread dump

1 2 3

```
"Thread-5" #61 prio=5 os_prio=0 tid=0x00007f5d19a69800 nid=0x262c
runnable [0x00007f5bbf6fd000]
   java.lang.Thread.State: RUNNABLE
        at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
        at sun.nio.ch.EPollArrayWrapper.poll(EPollArrayWrapper.java:269)
        at
sun.nio.ch.EPollSelectorImpl.doSelect(EPollSelectorImpl.java:79)
        at sun.nio.ch.SelectorImpl.lockAndDoSelect(SelectorImpl.java:86)
        - locked <0x0000000713cc35c8> (a sun.nio.ch.Util$2)
        - locked <0x0000000713cc35e0> (a
java.util.Collections$UnmodifiableSet)
        - locked <0x0000000712711f68> (a sun.nio.ch.EPollSelectorImpl)
        at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:97)
        at
sun.net.httpserver.ServerImpl$Dispatcher.run(ServerImpl.java:352)
        at java.lang.Thread.run(Thread.java:745)
```

4

Deadlocks

- Thread dump can automatically detects trivial deadlocks
- works with synchronized blocks & j.u.c.Lock
- can be called by JMX
  `java.lang.Threading findDeadlockedThreads`

```
Found one Java-level deadlock:
=============================
"Thread-0":
  waiting for ownable synchronizer 0x000000076b2f0fc8, (a
java.util.concurrent.locks.ReentrantLock$NonfairSync),
  which is held by "main"
"main":
  waiting for ownable synchronizer 0x000000076b2f0ff8, (a
java.util.concurrent.locks.ReentrantLock$NonfairSync),
  which is held by "Thread-0"
```

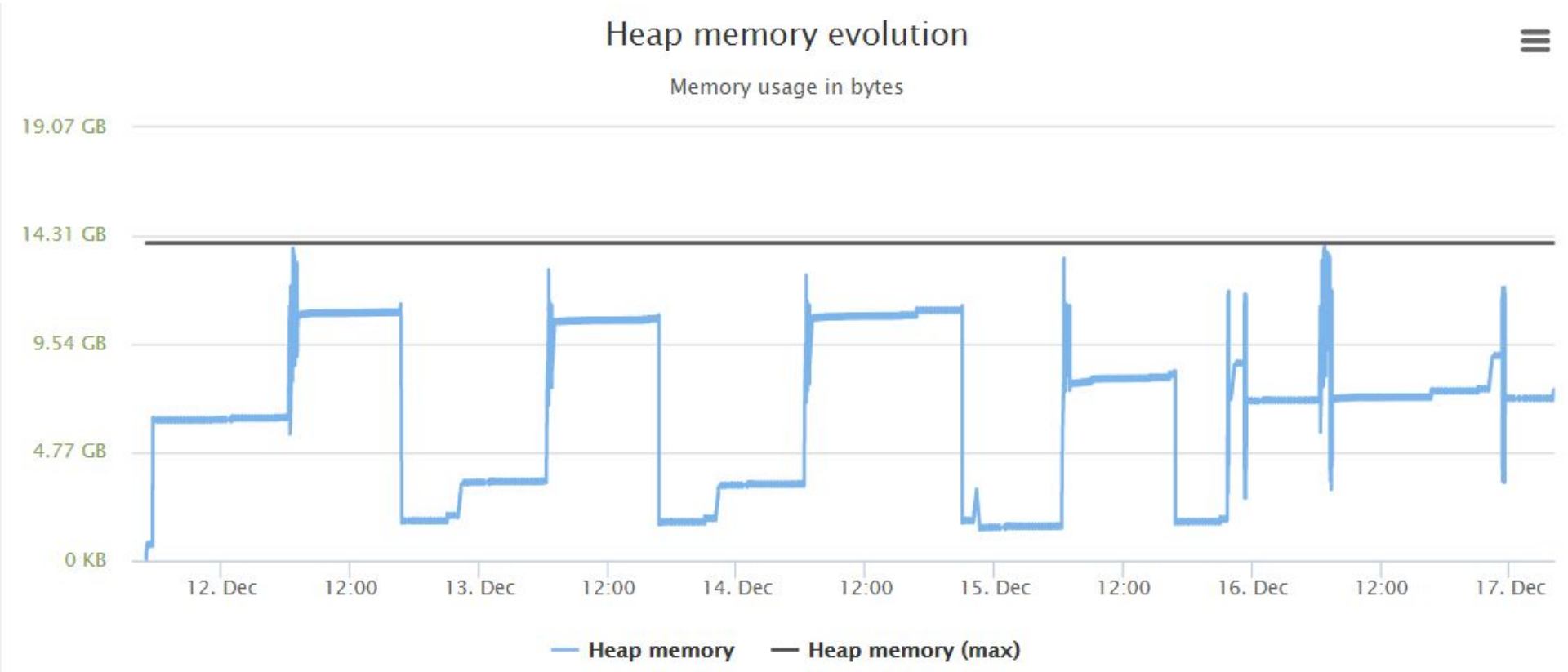
Memory/GC

Monitoring GC: logs

- Minimum JVM Options:
  - » `-Xloggc:gc.log`
  - » `-XX:+PrintGCDetails`
  - » `-XX:+PrintGCDateStamps`

```
2017-02-08T11:33:24.740+0100: 656.316: [Full
GC (System.gc()) [PSYoungGen:
60359K->0K(458752K)] [ParOldGen:
917600K->976339K(3670016K)]
977960K->976339K(4128768K), [Metaspace:
47136K->44073K(106496K)], 0.2618394 secs]
[Times: user=1.04 sys=0.03, real=0.26 secs]
```

GC logs: Plot them!



Monitoring GC: logs

Overlooked info: Times

```
[Times: user=1.17 sys=0.07, real=0.11 secs]

[Times: user=200.96 sys=85.32, real=60.68
secs]

[Times: user=2.58 sys=0.00, real=49.34 secs]
```

Monitoring GC: logs

- Information about aging in Survivor:
  `-XX:+PrintTenuringDistribution`
- Safepoint time:
  `-XX:+PrintGCApplicationStoppedTime`

```
2017-02-10T15:56:20.581+0100: 851.548: Total time for which application
threads were stopped: 0.3539306 seconds, Stopping threads took:
0.0000169 seconds
```

- Safepoint details (stdout)
  `-XX:+PrintSafepointStatistics`
  `-XX:+PrintSafepointStatisticsCount=1`

Monitoring GC: logs JDK9

`-Xlog:gc*:file=gclog.txt:time:uptime,l,tg:`

```
[2016-05-20T11:17:44.887+0200][0.038s][info][gc] Using Parallel
[2016-05-20T11:17:44.887+0200][0.038s][info][gc,heap,coops] Heap address: 0x00000006fc000000, size: 4096 MB,
Compressed Oops mode: Zero based, Oop shift amount: 3
[2016-05-20T11:17:47.291+0200][2.441s][info][gc,start     ] GC(0) Pause Young (Allocation Failure) (2.441s)
[2016-05-20T11:17:47.321+0200][2.471s][info][gc,heap      ] GC(0) PSYoungGen: 393216K->29472K(458752K)
[2016-05-20T11:17:47.321+0200][2.471s][info][gc,heap      ] GC(0) ParOldGen: 0K->8K(3670016K)
[2016-05-20T11:17:47.321+0200][2.471s][info][gc,metaspace ] GC(0) Metaspace: 18073K->18073K(81920K)
[2016-05-20T11:17:47.321+0200][2.471s][info][gc           ] GC(0) Pause Young (Allocation Failure)
384M->28M(4032M) (2.441s, 2.471s) 29.691ms
[2016-05-20T11:17:47.321+0200][2.471s][info][gc,cpu       ] GC(0) User=0.12s Sys=0.00s Real=0.03s
[2016-05-20T11:17:50.817+0200][5.969s][info][gc,start     ] GC(1) Pause Young (Allocation Failure) (5.969s)
[2016-05-20T11:17:50.847+0200][5.994s][info][gc,heap      ] GC(1) PSYoungGen: 422688K->45128K(458752K)
[2016-05-20T11:17:50.847+0200][5.994s][info][gc,heap      ] GC(1) ParOldGen: 917512K->917520K(3670016K)
[2016-05-20T11:17:50.847+0200][5.994s][info][gc,metaspace ] GC(1) Metaspace: 28613K->28613K(92160K)
[2016-05-20T11:17:50.847+0200][5.994s][info][gc           ] GC(1) Pause Young (Allocation Failure)
1308M->940M(4032M) (5.969s, 5.994s) 24.988ms
[2016-05-20T11:17:50.847+0200][5.994s][info][gc,cpu       ] GC(1) User=0.12s Sys=0.00s Real=0.03s
```

Monitoring GC: JMX/MBean

- MBean:
  `java.lang:type=GarbageCollector,name=*`
  Name depends on Collectors

- Allow internal or external live monitoring + Notifications

- LastGCInfo
  - thread count
  - duration
  - mem usage before/after
  - start/end time

Leak or OutOfMemoryError?

- Memory issue? => get an histogram
- Beware this is a STW event (comparable to a FullGC)!
- `jmap -histo[:live] <pid>`
- `-XX:+PrintHistogramBeforeFullGC`

Histogram analysis

```
 num     #instances         #bytes  class name
----------------------------------------------
   1:    421751083    20244051984  java.util.concurrent.ConcurrentHashMap$HashEntry
   2:     23327688    12130397760  com.ullink.ulmonitoring.api.model.SimpleBMOrder
   3:     23428922     6694953456  [Ljava.util.concurrent.ConcurrentHashMap$HashEntry;
   4:     11574497     5007170576  [C
   5:       178973     2246627208  [B
   6:     23357783     1681760376  java.util.concurrent.ConcurrentHashMap
   7:     26115162     1253527776  java.util.HashMap$Entry
   8:     23386843     1122568464  java.util.concurrent.locks.ReentrantLock$NonfairSync
   9:     23381591     1122316368  java.util.concurrent.ConcurrentHashMap$Segment
  10:     27363594     1094543760  java.lang.String
  11:     22350856      894034240  com.ullink.ulmonitoring.api.model.IndentedHelper
  12:     22350849      894033960  com.ullink.ulmonitoring.api.model.OrderLine
  13:     23357784      747639528  [Ljava.util.concurrent.ConcurrentHashMap$Segment;
  14:     22350792      715225344  com.ullink.ulmonitoring.view.HierarchicalOrderLine
  15:     23356168      560548032  com.ullink.ulmonitoring.api.model.property.PropertyHolderImpl
...
817:           40           4480  com.ullink.ulmonitoring.view.HierarchicalViewDataContainer
```

Leak or OutOfMemoryError?

- Still Memory issue? => get a heap dump
- Beware this is a STW event and can be very very long!
- `jmap -dump[:live,]format=b,file=heap.hprof <pid>`
- `-XX:+HeapDumpOnOutOfMemoryError`
  `-XX:HeapDumpPath=../logs`
  `-XX:OnOutOfMemoryError=sh OOME_script.sh`
- JMX:
  `com.sun.management:type=HotSpotDiagnostic.dumpHeap(filename, liveonly)`

Heap Dump analysis

Class	Objects		Shallow Size		Retained Size	
javolution.util.FastMap$Entry	199 119	0 %	6 371 808	0 %	≈ 4 633 102 805	100 %
java.util.HashMap$Entry	3 674 013	5 %	117 568 416	3 %	≈ 2 310 444 936	50 %
java.util.HashMap	697 175	1 %	33 464 400	1 %	≈ 2 256 358 160	49 %
java.util.HashMap$Entry[]	707 506	1 %	77 521 848	2 %	≈ 2 221 700 720	48 %
com.ullink.ultools.collections.ConcurrentMultiMap$HashEntry	52 456 171	70 %	1 678 597 472	36 %	≈ 1 915 161 984	41 %
com.ullink.oms.uliris.manager.aggregation.MetaContainer	1	0 %	88	0 %	1 749 167 048	38 %
java.util.Collections$SynchronizedMap	1 556	0 %	49 792	0 %	≈ 1 724 803 696	37 %
com.ullink.oms.uliris.manager.aggregation.data.LinkedControlItem	75 624	0 %	13 309 824	0 %	≈ 1 682 646 264	36 %
com.ullink.ultools.collections.ConcurrentMultiMap$HashEntry[]	264 459	0 %	516 515 056	11 %	≈ 1 680 235 664	36 %
com.ullink.oms.uliris.manager.aggregation.container.UserControlItemContainer	1	0 %	64	0 %	1 602 239 792	35 %
com.ullink.oms.uliris.manager.aggregation.data.AggregatedValue	320 063	0 %	5 121 008	0 %	≈ 1 574 052 120	34 %
com.ullink.ultools.collections.ConcurrentMultiMap$Segment	264 400	0 %	10 576 000	0 %	≈ 1 530 397 344	33 %

Name	Retained Size	Shallow Size
com.ullink.oms.uliris.manager.aggregation.MetaContainer	1 749 167 048	88
userControlItemContainer ⇨ com.ullink.oms.uliris.manager.aggregation.container.UserControlI	1 602 239 792	64
outputObjectMap ⇨ java.util.Collections$SynchronizedMap	1 595 939 272	32
mutex ⇨ **<this>** [Self Reference]	1 595 939 272	32
m ⇨ java.util.HashMap **size** = 34555	1 595 939 240	48
table ⇨ java.util.HashMap$Entry[65536]	1 595 939 192	262 160
[4] ⇨ java.util.HashMap$Entry **chunk size** = 2	4 224	32
next ⇨ java.util.HashMap$Entry	2 152	32
value ⇨ com.ullink.oms.uliris.manager.aggregation.data.LinkedControlItem	2 040	176
aggregationOrderCheck ⇨ com.ullink.oms.uliris.manager.aggregation.c	408	24
aggregationOrderLoad ⇨ com.ullink.oms.uliris.manager.aggregation.da	288	40
aggregationPath ⇨ com.ullink.oms.uliris.manager.aggregation.data.Agg	256	16
aggregationDailyKeys ⇨ com.ullink.oms.uliris.manager.aggregation.dat	168	16
aggregationAuthorization ⇨ com.ullink.oms.uliris.manager.aggregatior	104	16
aggregationBorrowAvailability ⇨ com.ullink.oms.uliris.manager.aggre	80	16

JVM crash

Crash

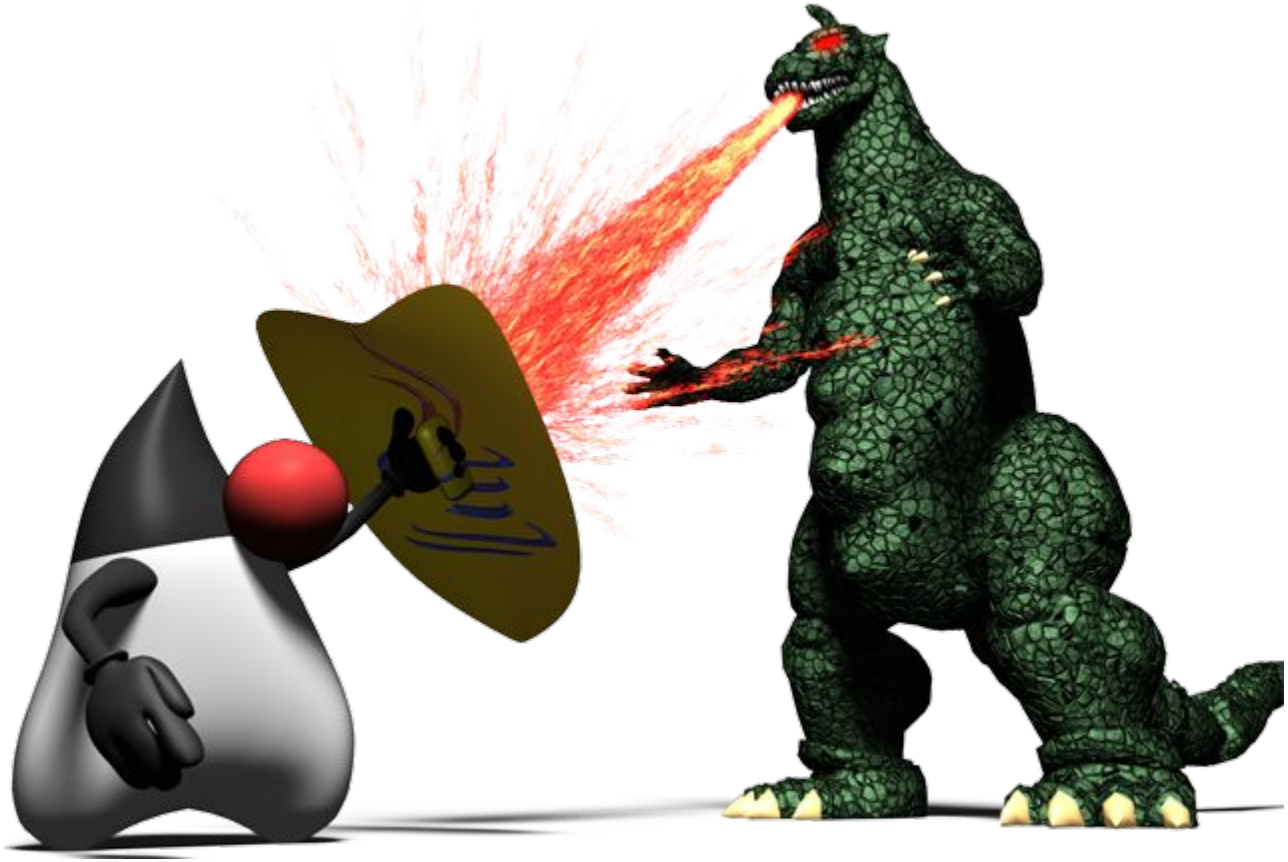
- Usually crash intercepted and hs_err_pid<pid>.log created in working directory

- `-XX:ErrorFile=/tmp/hs_err_pid%p.log`
  `-XX:OnError=sh crash_script.sh`

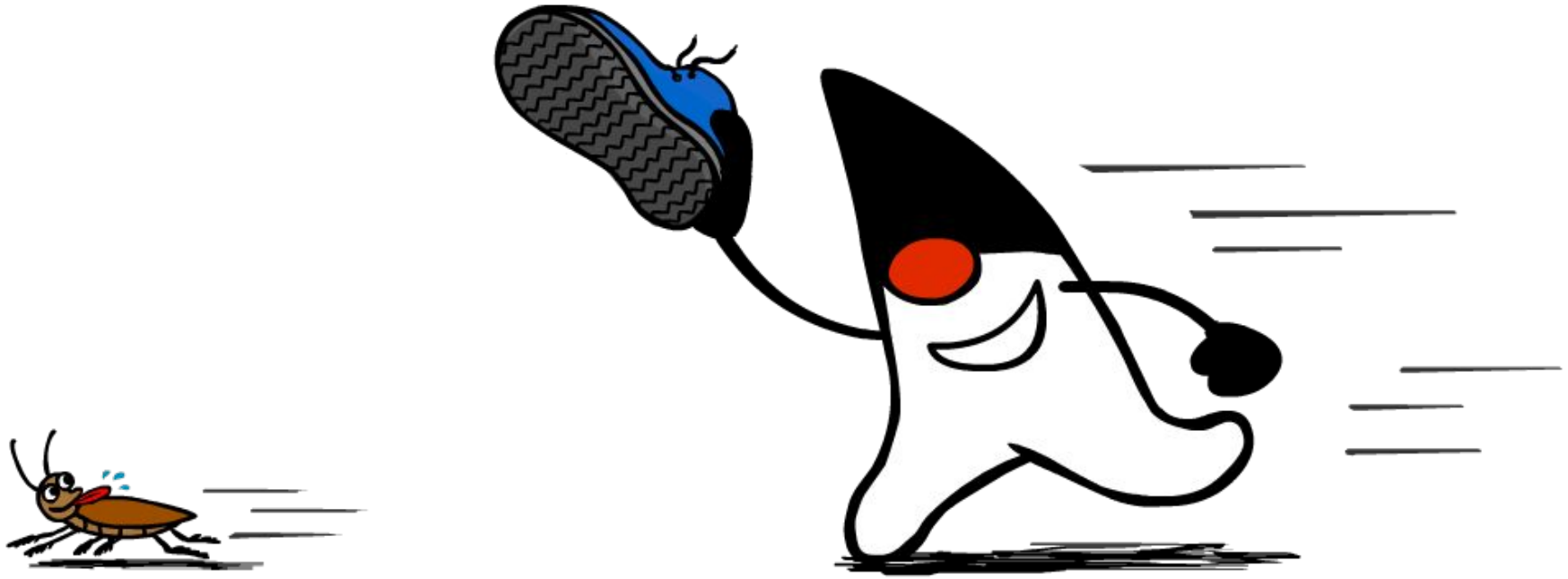
- Usually a JVM bug
  => a pure Java program should **never** crash a JVM

- Use latest JDK update

War stories



War stories



References

- FastThread.io Web Thread dump analyzer
- GCEasy.io Web GC analyzer
- GC Causes distilled

Q&A

Jean-Philippe BEMPEL
Performance Architect

@jpbempel
http://jpbempel.blogspot.com

Ullink
CONNECT. TRADE.