Università della Svizzera italiana

Faculty of Informatics

Use at Your Own Risk: The Java Unsafe API in the Wild

Luis Mastrangelo, Luca Ponzanelli, Andrea Mocci,
Michele Lanza, Matthias Hauswirth, Nathaniel Nystrom

Faculty of Informatics
Università della Svizzera Italiana (USI)
Switzerland

Java is a safe language.

```
$ java ch.usi.inf.MySafeJavaProgram
```

```
$ java ch.usi.inf.MySafeJavaProgram
#
# A fatal error has been detected by the Java Runtime Environment:
#
#  SIGSEGV (0xb) at pc=0x000000011170bf04, pid=43948, tid=3591
#
# JRE version: Java(TM) SE Runtime Environment (7.0_79-b15) (build 1
# Java VM: Java HotSpot(TM) 64-Bit Server VM (24.79-b02 mixed mode b
# Problematic frame:
# V  [libjvm.dylib+0x50bf04]  Unsafe_GetNativeLong+0x4c
```

```
$ java ch.usi.inf.MySafeJavaProgram
#
# A fatal error has been detected by the Java Runtime Environment:
#
#  SIGSEGV (0xb) at pc=0x000000011170bf04, pid=43948, tid=3591
#
# JRE version: Java(TM) SE Runtime Environment (7.0_79-b15) (build 1
# Java VM: Java HotSpot(TM) 64-Bit Server VM (24.79-b02 mixed mode b
# Problematic frame:
# V  [libjvm.dylib+0x50bf04]  Unsafe_GetNativeLong+0x4c
```

What is sun.misc.Unsafe?

Why Developers Should Not Write Programs That Call 'sun' Packages

The classes that JavaSoft includes with the JDK fall into at least two packages: java.* and sun.*. Only classes in java.* packages are a standard part of the Java Platform and will be supported into the future. In general, API outside of java.* can change at any time without notice, and so cannot be counted on either across OS platforms (Sun, Microsoft, Netscape, Apple, etc.) or across Java versions. Programs that contain direct calls to the sun.* API are not 100% Pure Java. In other words:

The java.* packages make up the official, supported, public Java interface.
If a Java program directly calls only API in java.* packages, it will operate on all Java-compatible platforms, regardless of the underlying OS platform.

The sun.* packages are *not* part of the supported, public Java interface.
A Java program that directly calls any API in sun.* packages is *not* guaranteed to work on all Java-compatible platforms. In fact, such a program is not guaranteed to work even in future versions on the same platform.

For these reasons, there is no documentation available for the sun.* classes. Platform-independence is one of the great advantages of developing in Java. Furthermore, JavaSoft, and our licensees of Java technology, are committed to maintaining the APIs in java.* for future versions of the Java platform. (Except for code that relies on bugs that we later fix, or APIs that we deprecate and eventually remove.) This means that once your program is written, the binary will work in future releases. That is, future implementations of the java platform will be backward compatible.

Each company that implements the Java platform will do so in their own private way. The classes in sun.* are present in the JDK to support the JavaSoft implementation of the Java platform: the sun.* classes are what make the classes in java.* work "under the covers" for the JavaSoft JDK. These classes will not in general be present on another vendor's Java platform. If your Java program asks for a class "sun.package.Foo" by name, it will likely fail with ClassNotFoundError, and you will have lost a major advantage of developing in Java.

Technically, nothing prevents your program from calling API in sun.* by name, but these classes are unsupported APIs, and we are not committed to maintaining backward compatibility for them. From one release to another, these classes may be removed,or they may be moved from one package to another, and it's fairly likely that the API (method names and signatures) will change. (From the JavaSoft point of view, since we are committed to maintaining the java.* APIs, we need to be able to change sun.* to enhance our products.) In this case, even if you are willing to run only on the JavaSoft implementation, you run the risk of a new version of the implementation breaking your program.

In general, writing java programs that rely on sun.* is risky: they are not portable, and the APIs are not supported.

Unsafe is hidden

Unsafe is hidden

```
1 Constructor<Unsafe> c = Unsafe.class.
      getDeclaredConstructor();
2 c.setAccessible(true);
3 Unsafe unsafe = c.newInstance();
```

JVM crash on 64 bit SPARC with Elasticsearch 1.2.2 due to unaligned memory access

26 posts by 6 authors G+1

David Roberts 22/07/2014

Hello,

After upgrading from Elasticsearch 1.0.1 to 1.2.2 I'm getting JVM core dumps on Solaris 10 on SPARC.

```
# A fatal error has been detected by the Java Runtime Environment:
#
#  SIGBUS (0xa) at pc=0xffffffff7e452d78, pid=15483, tid=263
#
# JRE version: Java(TM) SE Runtime Environment (7.0_55-b13) (build 1.7.0_55-b13)
# Java VM: Java HotSpot(TM) [illegible] VM (24.55-b03 mixed mode solaris-sparc compressed oops)
# Problematic f[illegible]
# V  [libjvm.so[illegible]52d78]  Unsafe_GetLong+0x[illegible]
```

I'm pretty sure the problem here is that Elasticsearch is making increasing use of "unsafe" functions in Java, presumably to speed things up, and some CPUs are more picky than others about memory alignment. In particular, x86 will tolerate misaligned memory access whereas SPARC won't.

Somebody has tried to report this to Oracle in the past and (understandably) Oracle has said that if you're going to use unsafe functions you need to understand what you're doing: http://bugs.java.com/bugdatabase/view_bug.do?bug_id=8021574

A quick grep through the code of the two versions of Elasticsearch shows that the new use of "unsafe" memory access functions is in the BytesReference, MurmurHash3 and HyperLogLogPlusPlus classes:

```
bash-3.2$ git checkout v1.0.1
Checking out files: 100% (2904/2904), done.

bash-3.2$ find . -name '*.java' | xargs grep UnsafeUtils
./src/main/java/org/elasticsearch/common/util/UnsafeUtils.java:public enum UnsafeUtils {
./src/main/java/org/elasticsearch/search/aggregations/bucket/BytesRefHash.java:            if (id == -1L || UnsafeUtils.equals(key, get(id, spare))) {
./src/main/java/org/elasticsearch/search/aggregations/bucket/BytesRefHash.java:            } else if (UnsafeUtils.equals(key, get(curId, spare))) {
./src/test/java/org/elasticsearch/benchmark/common/util/BytesRefComparisonsBenchmark.java:import org.elasticsearch.common.util.UnsafeUtils;
./src/test/java/org/elasticsearch/benchmark/common/util/BytesRefComparisonsBenchmark.java:                return UnsafeUtils.equals(b1, b2);
```

JVM crash on 64 bit SPARC with Elasticsearch 1.2.2 due to unaligned memory access

26 posts by 6 authors

David Roberts 22/07/2014

Hello,

After upgrading from Elasticsearch 1.0.1 to 1.2.2 I'm getting JVM core dumps on Solaris 10 on SPARC.

```
# A fatal error has been detecti
#
#  SIGBUS (0xa) at pc=0xffffffff
#
# JRE version: Java(TM) SE R
# Java VM: Java HotSpot(TM)
# Problematic fr
# V  [libjvm.so+0x52d78]  Uns
```

I'm pretty sure the problem is
picky than others about memor

Somebody has tried to report t
you're doing: http://bugs.java.c

A quick grep through the code
and HyperLogLogPlusPlus clas

```
bash-3.2$ git checkout v1.0.1
Checking out files: 100% (2904

bash-3.2$ find . -name '*.java' |
./src/main/java/org/elasticsearc
./src/main/java/org/elasticsearc
./src/main/java/org/elasticsearc
./src/test/java/org/elasticsearch
./src/test/java/org/elasticsearch
```

Hazelcast cust serialization with kryo fail in jdk 64bit in solaris-sparc #219

Closed **bwzhang2011** opened this issue on May 14, 2014 · 11 comments

bwzhang2011 commented on May 14, 2014

please take a look at hazelcast/hazelcast#2453

the kryo serialization seems fail in solaris sparc and I don't know whether kryo test cases has been completely tested in jdk solaris edition. and I hope someone could help me point out my mistake or something wrong with kryo itself for different o.s

romix commented on May 14, 2014 Collaborator

It seems like it has pr ems using Unsafe-backend Solaris.
But you can easily switch t based one, using

```
kryo.setAsmEnabled(true);
```

ASM-based backend should work on any platform.

JVM crash on 64 bit SPARC with Elasticsearch 1.2.2 due to unaligned memory access

26 posts by 6 authors

David Roberts

Hello,

After upgrading from Elasticsearch 1.0.1 to 1.2.2 I'm getting

A fatal error has been detected by the Java Runtime Envir
#
SIGBUS (0xa) at pc=0xffffffff7e452d78, pid=15483, tid=26
#
JRE version: Java(TM) SE Runtime Environment (7.0_55-
Java VM: Java HotSpot(TM) [illegible] VM (24.55-b03
Problematic fr[illegible]
V [libjvm.so+0x52d78] Unsafe_GetLong+0x[illegible]

I'm pretty sure the problem [illegible] Elasticsearch is maki
picky than others about memory alignment. In particular, x8

Somebody has tried to report this to Oracle in the past and (
you're doing: http://bugs.java.com/bugdatabase/view_bug.do

A quick grep through the code of the two versions of Elastics
and HyperLogLogPlusPlus classes:

bash-3.2$ git checkout v1.0.1
Checking out files: 100% (2904/2904), done.

bash-3.2$ find . -name '*.java' | xargs grep UnsafeUtils
./src/main/java/org/elasticsearch/common/util/UnsafeUtils.ja
./src/main/java/org/elasticsearch/search/aggregations/bucke
./src/main/java/org/elasticsearch/search/aggregations/bucke
./src/test/java/org/elasticsearch/benchmark/common/util/Byt
./src/test/java/org/elasticsearch/benchmark/common/util/Byt

Hazelcast cust serialization with kryo fail in jdk 64bit in

Snappy causes SIGBUS jvm crasher on HP-UX via UnsafeMemory.loadInt() #24

Open **ekeithkw** opened this issue on Jan 14 · 1 comment

1 comments

on't know whether kryo test cases has been
neone could help me point out my mistake or

Collaborator

Solaris.
ed one, using

ekeithkw commented on Jan 14

Steps to reproduce:

1. Install HP-UX Java 7 (e.g. from "Itanium_JDK_JRE_7.0.09_-_Feb_2014_Z7550-01326_java_17009_ia.depot") to a local folder.
2. Set JDK_HOME to the above folder.
3. Download CLion for Linux, from jetbrains.com > Products > CLion > Early Access Program (EAP); gunzip and "tar xvf" to unpack (e.g. to /opt).
4. Run "/opt/clion-140.569.17/bin/clion.sh". Deselect all plug-ins.
5. It appears necessary to create or open a project (failure occurs when opening a project).

Snippet from hs_err.log file:

Current thread (01370200): JavaThread "ApplicationImpl pooled thread 3" [_thread_in_vm, id=30, lwp_id=4383651, stack(2d601000,2d641000)]

```
siginfo:si_signo=SIGBUS: si_errno=0, si_code=1 (BUS_ADRALN), si_addr=2f4b204d

[illegible]ode, j=interpreted, Vv=VM code)
j sun.misc.Unsafe.getInt(L[illegible]ang/Object;J)I+0
[illegible].loadInt([BI)I+50
j org.[illegible].SnappyInternalUtils.loadInt([BI)I+5
j org.iq80.snappy.SnappyCompressor.findCandidate([BIIII[SI)I+19
j org.iq80.snappy.SnappyCompressor.compressFragment([BII[BI[S)I+157
j org.iq80.snappy.SnappyCompressor.compress([BII[BI)I+63
j org.iq80.snappy.Snappy.compress([BII[BI)I+6
j org.iq80.snappy.SnappyOutputStream.writeCompressed([BII)V+27
j org.iq80.snappy.SnappyOutputStream.flushBuffer()V+17
j org.iq80.snappy.SnappyOutputStream.flush()V+18
j sun.nio.cs.StreamEncoder.implFlush()V+15
j sun.nio.cs.StreamEncoder.flush()V+12
j java.io.OutputStreamWriter.flush()V+4
j org.jdom.output.XMLOutputter.output(Lorg/jdom/Element;Ljava/io/Writer;)V+12
j com.intellij.openapi.components.impl.stores.StateMap.a(Lorg/jdom/Element;)[B+85
j com.intellij.openapi.components.impl.stores.StateMap.getStateAndArchive(Ljava/lang/String;)Lorg/j
```

JVM crash on 64 bit SPARC with Elasticsearch 1.2.2 due to unaligned memory access

26 posts by 6 authors

David Roberts

Hello,

After upgrading from Elasticsearch 1.0.1 to 1.2.2 I'm gettin

A fatal error has been detected by the Java Runtime Env
#
SIGBUS (0xa) at pc=0xffffffff7e452d78, pid=15483, tid=
#
JRE version: Java(TM) SE Runtime Environment (7.0_5
Java VM: Java HotSpot(TM) [illegible] VM (24.55-b0
Problematic fr[illegible]
V [libjvm.so-[illegible]52d78] Unsafe_GetLong+0x

I'm pretty sure the problem [illegible] Elasticsearch is ma
picky than others about memory alignment. In particular, x

Somebody has tried to report this to Oracle in the past and
you're doing: http://bugs.java.com/bugdatabase/view_bug.

A quick grep through the code of the two versions of Elasti
and HyperLogLogPlusPlus classes:

bash-3.2$ git checkout v1.0.1
Checking out files: 100% (2904/2904), done.

bash-3.2$ find . -name '*.java' | xargs grep UnsafeUtils
./src/main/java/org/elasticsearch/common/util/UnsafeUtils.j
./src/main/java/org/elasticsearch/search/aggregations/buck
./src/main/java/org/elasticsearch/search/aggregations/buck
./src/test/java/org/elasticsearch/benchmark/common/util/By
./src/test/java/org/elasticsearch/benchmark/common/util/By

Hazelcast cust serialization with kryo fail in jdk 64bit in

comments

n't know whether kryo test cases has been
eone could help me point out my mistake or

Collaborator

Solaris.
d one, using

Bug 229655 - StackOverflowError at sun.misc.Unsafe.defineClass

Status:	RESOLVED FIXED
Product:	debugger
Component:	Java
Version:	7.4
Hardware:	All All
Priority:	P3 (vote)
Target Milestone:	7.4
Assigned To:	Martin Entlicher
QA Contact:	issues@debugger
URL:	
Whiteboard:	EXCEPTIONS_REPORT
Keywords:	
Depends on:	
Blocks:	

Reported:	2013-05-13 12:56 UTC by Ondrej Vrabec
Modified:	2013-05-23 08:17 UTC (History)
CC List:	1 user (show)
See Also:	
Issue Type:	DEFECT
Exception Report :	20060

Show dependency tree / graph

Ondrej Vrabec 2013-05-13 12:56:07 UTC

```
Build: NetBeans IDE Dev (Build 20130510-fb9d6c10dcaa)
VM: Java HotSpot(TM) 64-Bit Server VM, 23.6-b04, Java(TM) SE Runtime Environment, 1.7.0_10-b18
OS: Linux

User Comments:
ovrabec: applying code changes



Stacktrace:
java.lan[illegible]
    at su[illegible]rc.Unsafe.defineClass([illegible]fe.java:0)
    at sun.[illegible]neClass(ClassDefiner.java:63)
    at sun.reflect.MethodAccessorGenerator$1.run(MethodAccessorGenerator.java:399)
    at sun.reflect.MethodAccessorGenerator$1.run(MethodAccessorGenerator.java:396)
    at java.security.AccessController.doPrivileged(AccessController.java:0)
    at sun.reflect.MethodAccessorGenerator.generate(MethodAccessorGenerator.java:395)
```

Snappy causes SIGBUS j

UnsafeMemory.loadInt() #

Open ekeithkw opened this issue on Jan 14 · 1

ekeithkw commented on Jan 14

Steps to reproduce:
1. Install HP-UX Java 7 (e.g. from "Itaniur
01326_java_17009_ia.depot") to a local fo
2. Set JDK_HOME to the above folder.
3. Download CLion for Linux, from jetbrai
gunzip and "tar xvf" to unpack (e.g. to /op
4. Run "/opt/clion-140.569.17/bin/clion.sh
5. It appears necessary to create or open

Snippet from hs_err.log file:
Current thread (01370200): JavaThread "A
lwp_id=4383651, stack(2d601000,2d641

```
siginfo:si_signo=SIGBUS: si_errno=0, s
[illegible]
j sun.misc.Unsafe.getInt(Ljava/lang/Ob
[illegible].loadInt
j org.iq80.snappy.SnappyInternalUtils.
j org.iq80.snappy.SnappyCompressor.fi
j org.iq80.snappy.SnappyCompressor.co
j org.iq80.snappy.SnappyCompressor.co
j org.iq80.snappy.Snappy.compress([BI
j org.iq80.snappy.SnappyOutputStream.
j org.iq80.snappy.SnappyOutputStream.
j org.iq80.snappy.SnappyOutputStream.
j sun.nio.cs.StreamEncoder.implFlush(
j sun.nio.cs.StreamEncoder.flush()V+1
j java.io.OutputStreamWriter.flush()V
j org.jdom.output.XMLOutputter.output
j com.intellij.openapi.components.imp
j com.intellij.openapi.components.imp
```

JVM crash on 64 bit SPARC with Elasticsearch 1.2.2 due to unaligned memory access

26 posts by 6 authors

David Roberts 22/07/2014

Hello,

After upgrading from Elasticsearch 1.0.1 to 1.2.2 I'm getting JVM core dumps on Solaris 10 on SPARC.

```
# A fatal error has been detected by the Java Runtime Environment:
#
#  SIGBUS (0xa) at pc=0xffffffff7e452d78, pid=15483, tid=263
#
# JRE version: Java(TM) SE Runtime Environment (7.0_55-b13) (build 1.7.0_55-b13)
# Java VM: Java HotSpot(TM) [illegible] VM (24.55-b03 mixed mode solaris-sparc compressed oops)
# Problematic fr[illegible]
# V  [libjvm.so+0x52d78]  Unsafe_GetLong+0x[illegible]
```

I'm pretty sure the problem is that Elasticsearch is making increasing use of "unsafe" functions in Java, presumably to speed things up, and some CPUs are more picky than others about memory alignment. In particular, x86 will tolerate misaligned memory access whereas SPARC won't.

Somebody has tried to report this to Oracle in the past and (understandably) Oracle has said that if you're going to use unsafe functions you need to understand what you're doing: http://bugs.java.com/bugdatabase/view_bug.do?bug_id=8021574

A quick grep through the code of the two versions of Elasticsearch shows that the new use of "unsafe" memory access functions is in the BytesReference, MurmurHash3 and HyperLogLogPlusPlus classes:

```
bash-3.2$ git checkout v1.0.1
Checking out files: 100% (2904/2904), done.

bash-3.2$ find . -name '*.java' | xargs grep UnsafeUtils
./src/main/java/org/elasticsearch/common/util/UnsafeUtils.java:public enum UnsafeUtils {
./src/main/java/org/elasticsearch/search/aggregations/bucket/BytesRefHash.java:            if (id == -1L || UnsafeUtils.equals(key, get(id, spare))) {
./src/main/java/org/elasticsearch/search/aggregations/bucket/BytesRefHash.java:            } else if (UnsafeUtils.equals(key, get(curId, spare))) {
./src/test/java/org/elasticsearch/benchmark/common/util/BytesRefComparisonsBenchmark.java:import org.elasticsearch.common.util.UnsafeUtils;
./src/test/java/org/elasticsearch/benchmark/common/util/BytesRefComparisonsBenchmark.java:                return UnsafeUtils.equals(b1, b2);
```

Hazelcast cust serialization with kryo fail in jdk 64bit in solaris-sparc #219

Closed **bwzhang2011** opened this issue on May 14, 2014 · 11 comments

bwzhang2011 commented on May 14, 2014

please take a look at hazelcast/hazelcast#2453

the kryo serialization seems fail in solaris sparc and I don't know whether kryo test cases has been completely tested in jdk solaris edition. and I hope someone could help me point out my mistake or something wrong with kryo itself for different o.s

romix commented on May 14, 2014 Collaborator

It seems like it has problems using Unsafe-backend on Solaris.
But you can easily switch to ASM-based one, using

```
kryo.setAsmEnabled(true);
```

ASM-based backend should work on any platform.

Snappy causes SIGBUS jvm crasher on HP-UX via UnsafeMemory.loadInt() #24

Open **ekeithkw** opened this issue on Jan 14 · 1 comment

ekeithkw commented on Jan 14

Steps to reproduce:
1. Install HP-UX Java 7 (e.g. from "Itanium_JDK_JRE_7.0.09_-_Feb_2014_Z7550-01326_java_17009_ia.depot") to a local folder.
2. Set JDK_HOME to the above folder.
3. Download CLion for Linux, from jetbrains.com > Products > CLion > Early Access Program (EAP); gunzip and "tar xvf" to unpack (e.g. to /opt).
4. Run "/opt/clion-140.569.17/bin/clion.sh". Deselect all plug-ins.
5. It appears necessary to create or open a project (failure occurs when opening a project).

Snippet from hs_err.log file:
Current thread (01370200): JavaThread "ApplicationImpl pooled thread 3" [_thread_in_vm, id=30, lwp_id=4383651, stack(2d601000,2d641000)]

```
siginfo:si_signo=SIGBUS: si_errno=0, si_code=1 (BUS_ADRALN), si_addr=2f4b204d

[illegible]code, j=interpreted, Vv=VM code)
j sun.misc.Unsafe.getInt(Ljava/lang/Object;J)I+0
[illegible].loadInt([BI)I+50
j org.iq80.snappy.SnappyInternalUtils.loadInt([BI)I+5
j org.iq80.snappy.SnappyCompressor.findCandidate([BIIII[SI)I+19
j org.iq80.snappy.SnappyCompressor.compressFragment([BII[BI[S)I+157
j org.iq80.snappy.SnappyCompressor.compress([BII[BI)I+63
j org.iq80.snappy.Snappy.compress([BII[BI)I+6
j org.iq80.snappy.SnappyOutputStream.writeCompressed([BII)V+27
j org.iq80.snappy.SnappyOutputStream.flushBuffer()V+17
j org.iq80.snappy.SnappyOutputStream.flush()V+18
j sun.nio.cs.StreamEncoder.implFlush()V+15
j sun.nio.cs.StreamEncoder.flush()V+12
j java.io.OutputStreamWriter.flush()V+4
j org.jdom.output.XMLOutputter.output(Lorg/jdom/Element;Ljava/io/Writer;)V+12
j com.intellij.openapi.components.impl.stores.StateMap.a(Lorg/jdom/Element;)[B+85
j com.intellij.openapi.components.impl.stores.StateMap.getStateAndArchive(Ljava/lang/String;)Lorg/j
```

Bug 229655 - StackOverflowError at sun.misc.Unsafe.defineClass

Status:	RESOLVED FIXED	Reported:	2013-05-13 12:56 UTC by Ondrej Vrabec
Product:	debugger	Modified:	2013-05-23 08:17 UTC (History)
Component:	Java	CC List:	1 user (show)
Version:	7.4	See Also:	
Hardware:	All All	Issue Type:	DEFECT
Priority:	P3 (vote)	Exception Report :	20060-
Target Milestone:	7.4		
Assigned To:	Martin Entlicher		
QA Contact:	issues@debugger		
URL:			
Whiteboard:	EXCEPTIONS_REPORT		
Keywords:			
Depends on:			
Blocks:			

Show dependency tree / graph

Ondrej Vrabec 2013-05-13 12:56:07 UTC

```
Build: NetBeans IDE Dev (Build 20130510-fb9d6c10dcaa)
VM: Java HotSpot(TM) 64-Bit Server VM, 23.6-b04, Java(TM) SE Runtime Environment, 1.7.0_10-b18
OS: Linux

User Comments:
ovrabec: applying code changes



Stacktrace:
java.lang.[illegible]
   at sun.misc.Unsafe.defineClass([illegible]fe.java:0)
   at sun.[illegible]eClass(ClassDefiner.java:63)
   at sun.reflect.MethodAccessorGenerator$1.run(MethodAccessorGenerator.java:399)
   at sun.reflect.MethodAccessorGenerator$1.run(MethodAccessorGenerator.java:396)
   at java.security.AccessController.doPrivileged(AccessController.java:0)
   at sun.reflect.MethodAccessorGenerator.generate(MethodAccessorGenerator.java:395)
```

Research Question 1

Does *Unsafe* **impact** common application code?


The Central Repository
Apache
maven

The Central Repository

Apache maven

959,300 artifacts

1.7 TB

The Central Repository

Apache maven

959,300 artifacts

1.7 TB

Last version

The Central Repository

Apache maven

959,300 artifacts

1.7 TB

Last version

106,574 artifacts

The Central Repository

Apache maven

959,300 artifacts — 1.7 TB

Last version

106,574 artifacts

.jar, .war, .ear, .ejb

The Central Repository

Apache maven

959,300 artifacts — 1.7 TB

↓ Last version

106,574 artifacts

↓ .jar, .war, .ear, .ejb

86,479 artifacts — 74 GB

The Central Repository

Apache maven

959,300 artifacts — 1.7 TB

↓ Last version

106,574 artifacts

↓ .jar, .war, .ear, .ejb

86,479 artifacts — 74 GB

↓

The Central Repository

Apache maven

959,300 artifacts — 1.7 TB

↓ Last version

106,574 artifacts

↓ .jar, .war, .ear, .ejb

86,479 artifacts — 74 GB

↓

817 *Unsafe* artifacts (1%)

The Central Repository

Apache maven

959,300 artifacts — 1.7 TB

↓ Last version

106,574 artifacts

↓ .jar, .war, .ear, .ejb

86,479 artifacts — 74 GB

↓

817 *Unsafe* artifacts (1%)

48,139 call sites
351 field accesses

The Central Repository

Apache maven

Languages

Scala

JRuby

The Central Repository

Apache maven

Languages

Scala

JRuby

Frameworks

spring

HIBERNATE

Grails

mockito

The Central Repository

Apache maven

Languages

Scala

JRuby

Frameworks

spring

HIBERNATE

Grails

mockito

Serialization

OBJENESIS

XStream

KRYO

google-gson

The Central Repository

Apache maven

Languages

Scala

JRuby

Frameworks

spring

HIBERNATE

Grails

mockito

Serialization

OBJENESIS

XStream

KRYO

google-gson

Database

Cassandra

neo4j

HBASE

MapDB

The Central Repository

Apache maven

Languages

Scala

JRuby

Frameworks

spring

HIBERNATE

Grails

mockito

Serialization

OBJENESIS

XStream

KRYO

google-gson

Database

Cassandra

neo4j

HBASE

MapDB

Concurrency

akka

Netty

LMAX Disruptor

kafka

The Central Repository

Apache maven

Languages

Scala

JRuby

Frameworks

spring

HIBERNATE

Grails

mockito

Serialization

OBJENESIS

XStream

KRYO

google-gson

Database

Cassandra

neo4j

HBASE

MapDB

Concurrency

akka

Netty

LMAX Disruptor

kafka

Big Data

hadoop

Spark

Storm

Solr

The Central Repository

Apache maven

959,300 artifacts — 1.7 TB

↓ Last version

106,574 artifacts

↓ .jar, .war, .ear, .ejb

86,479 artifacts — 74 GB

↓

817 *Unsafe* artifacts (1%)

48,139 call sites
351 field accesses

The Central Repository

Apache maven

959,300 artifacts — 1.7 TB

Last version

106,574 artifacts

.jar, .war, .ear, .ejb

86,479 artifacts — 74 GB

48,139 call sites
351 field accesses

817 *Unsafe* artifacts (1%)

The Central Repository

Apache maven

959,300 artifacts — 1.7 TB

↓ Last version

106,574 artifacts

↓ .jar, .war, .ear, .ejb

86,479 artifacts — 74 GB

48,139 call sites
351 field accesses

817 *Unsafe* artifacts (1%)

Artifact dependencies

The Central Repository

Apache maven

959,300 artifacts — 1.7 TB

Last version

106,574 artifacts

.jar, .war, .ear, .ejb

86,479 artifacts — 74 GB

48,139 call sites
351 field accesses

817 *Unsafe* artifacts (1%)

Artifact dependencies

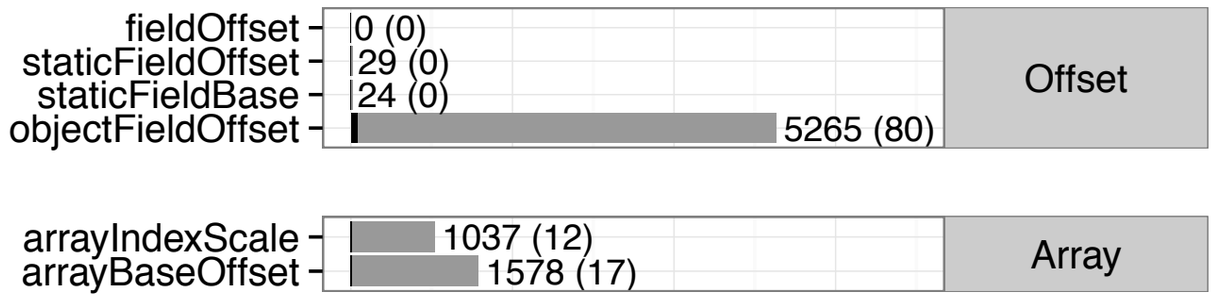
21,297 artifacts depend on *Unsafe* (25%)

Research Question 2

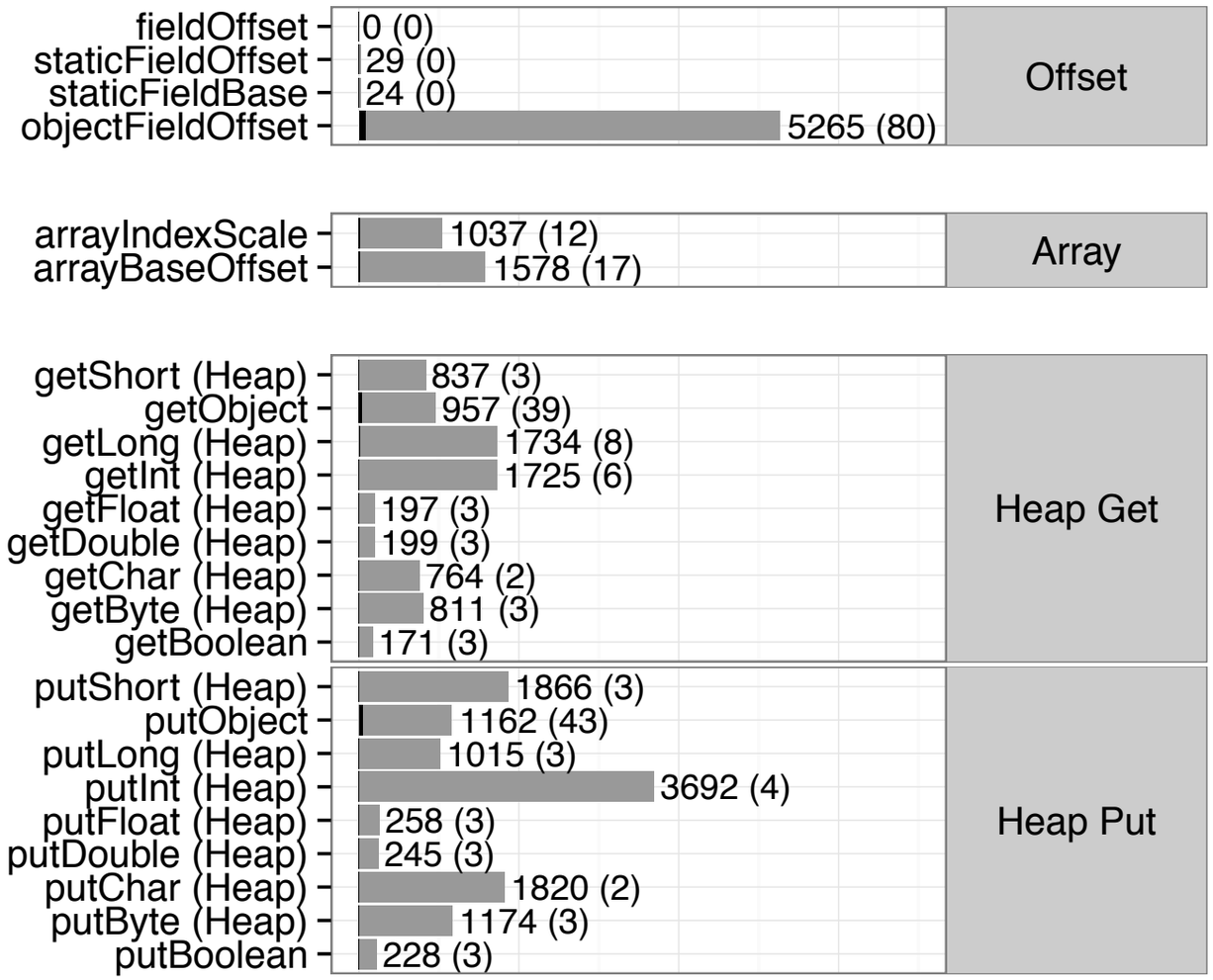
Which **features** of *Unsafe* are used?

Call Sites / sun.misc.Unsafe methods

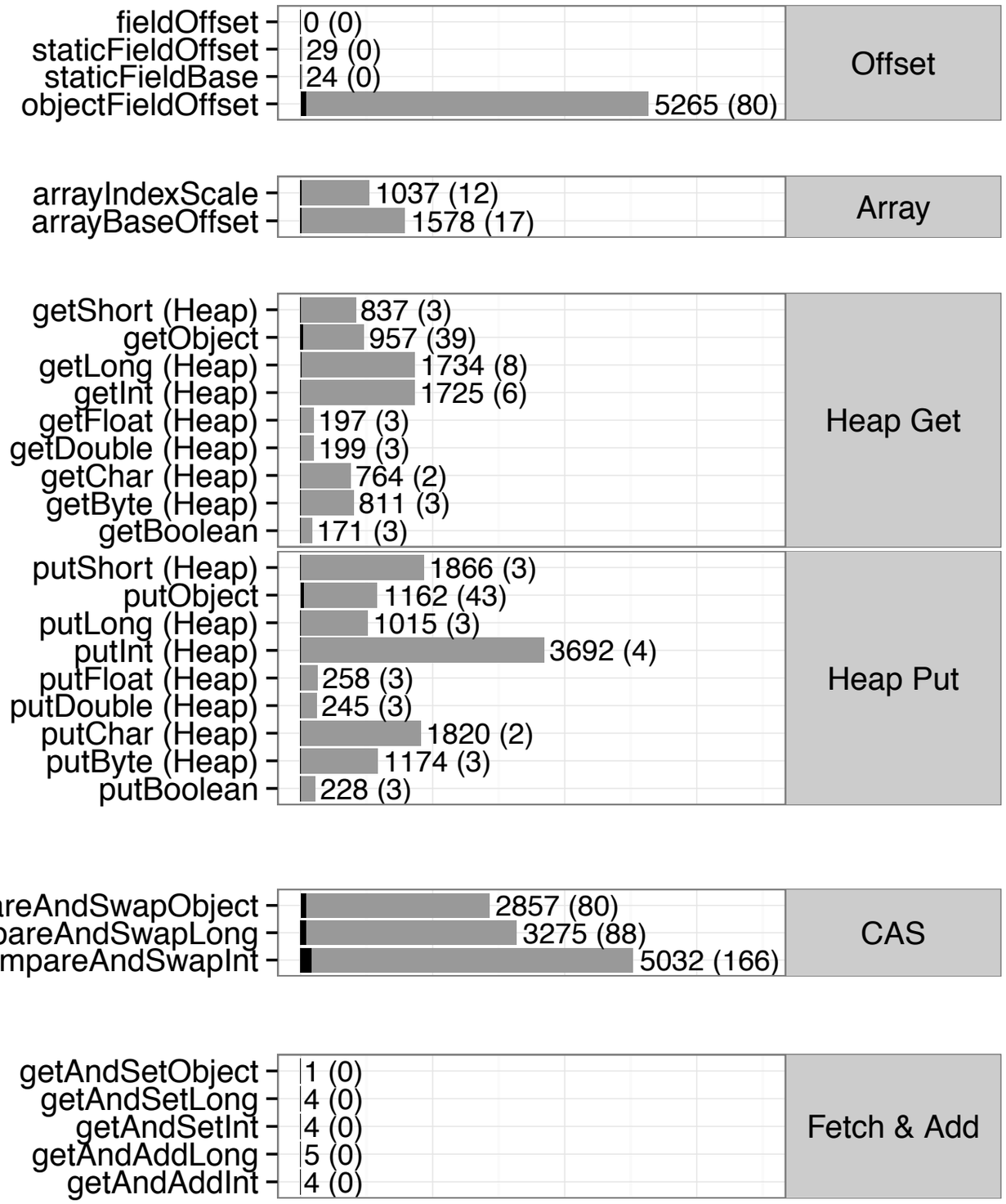
Call Sites / sun.misc.Unsafe methods



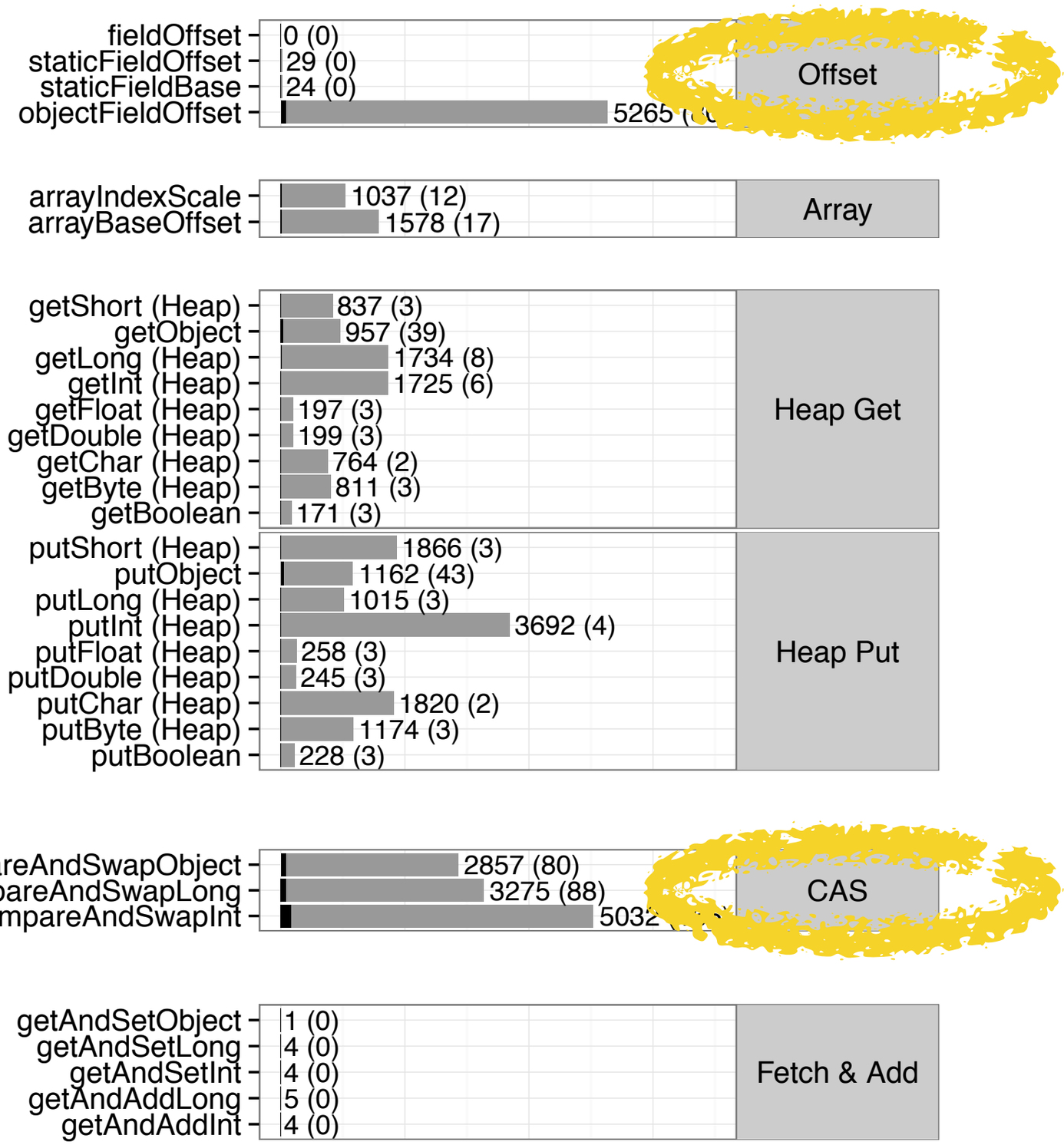
Call Sites / sun.misc.Unsafe methods



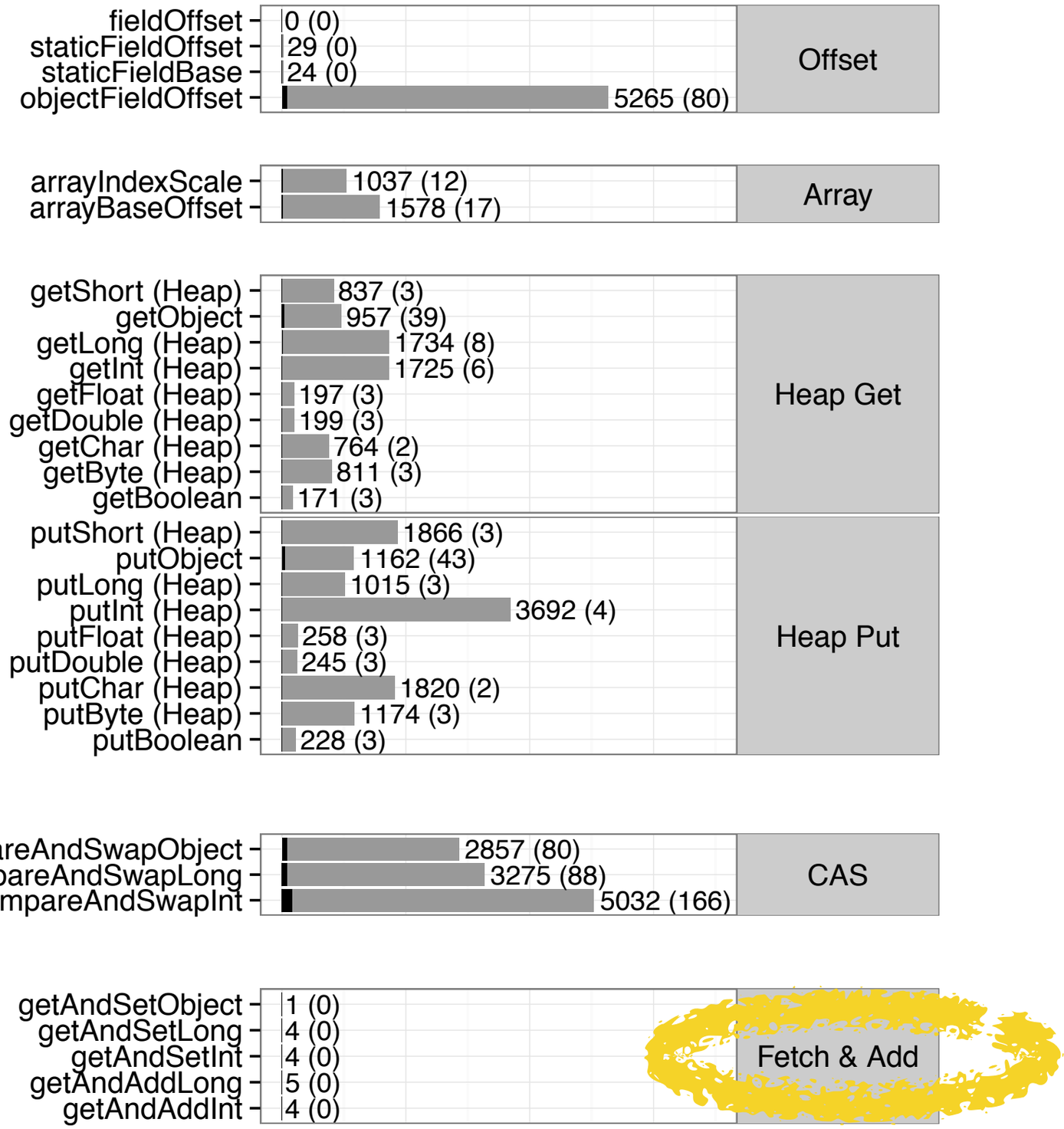
Call Sites / sun.misc.Unsafe methods



Call Sites / sun.misc.Unsafe methods

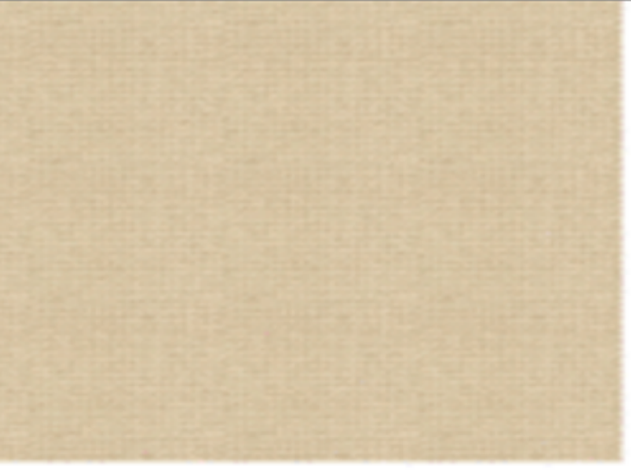


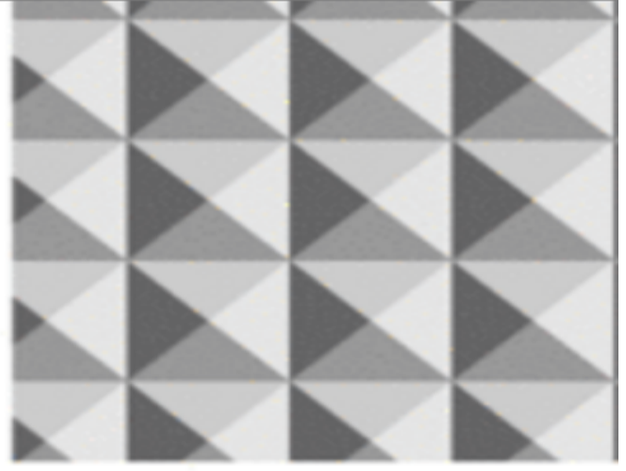
Call Sites / sun.misc.Unsafe methods



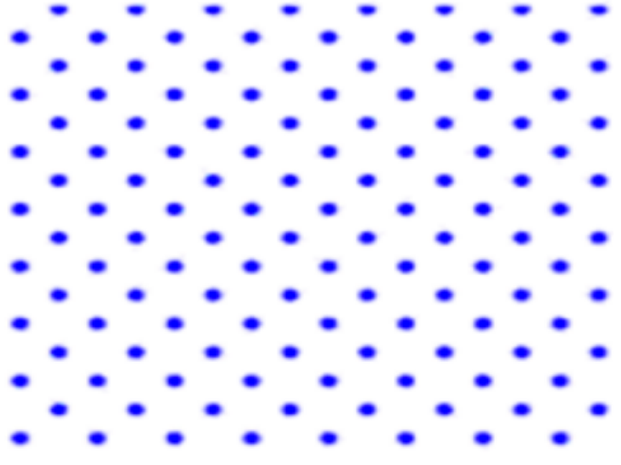
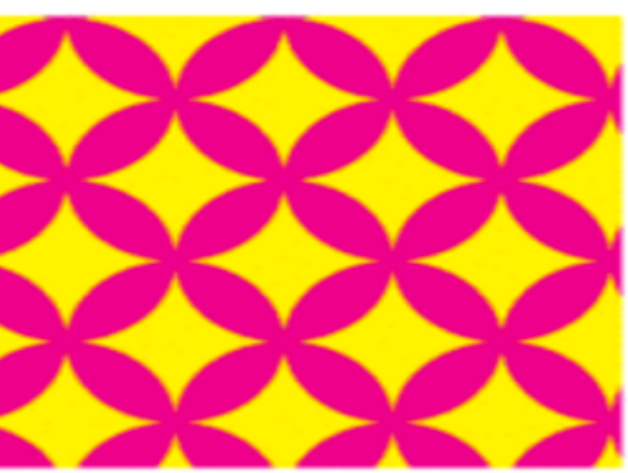
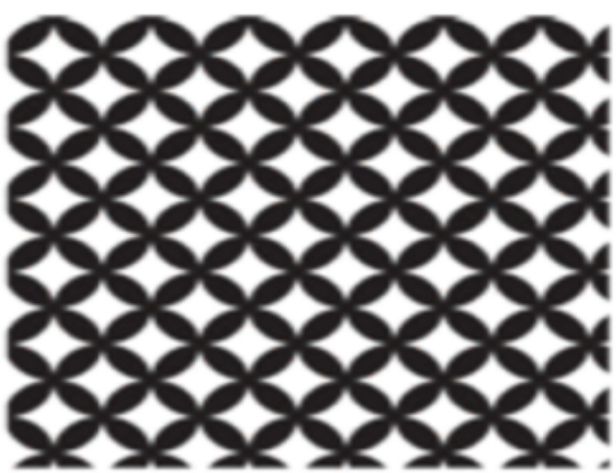
Research Question 3

Why are *Unsafe* features used?

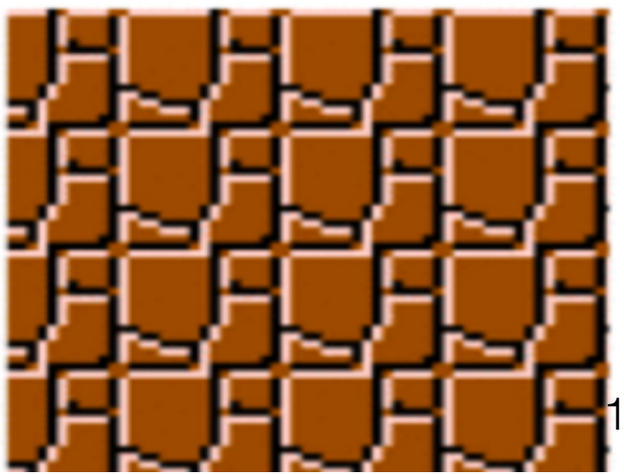
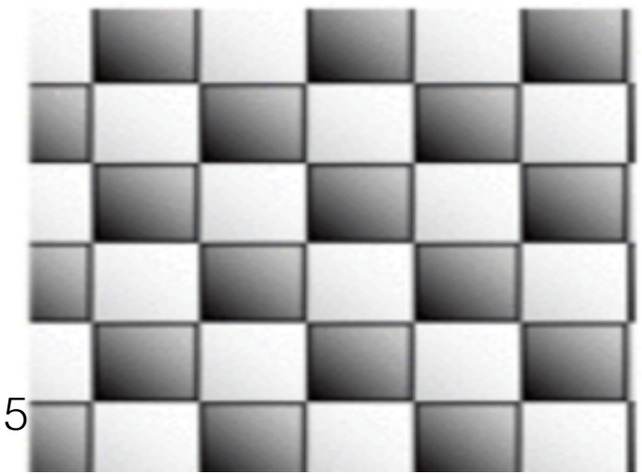
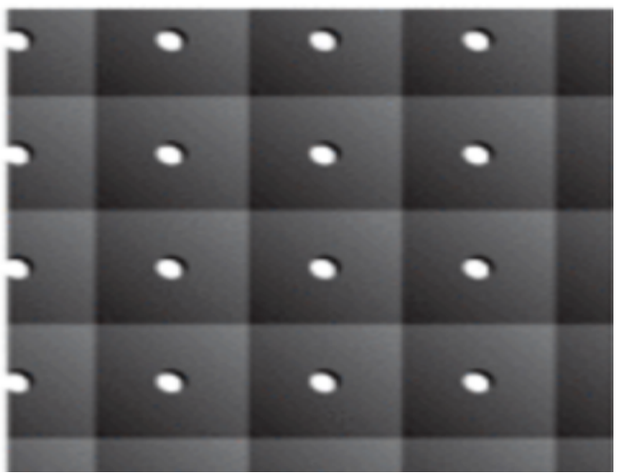



14 patterns



Process Byte Arrays in Block

Process Byte Arrays in Block

```
for (int i = 0; i < minWords * Longs.BYTES; i += Longs.BYTES) {
  long lw = theUnsafe.getLong(left, BYTE_ARRAY_BASE_OFFSET + (long) i);
  long rw = theUnsafe.getLong(right, BYTE_ARRAY_BASE_OFFSET + (long) i);
  if (lw != rw) {
    if (BIG_ENDIAN) {
      return UnsignedLongs.compare(lw, rw);
    }

    /*
     * We want to compare only the first index where left[index] != right[index].
     * This corresponds to the least significant nonzero byte in lw ^ rw, since lw
     * and rw are little-endian.  Long.numberOfTrailingZeros(diff) tells us the least
     * significant nonzero bit, and zeroing out the first three bits of L.nTZ gives us the
     * shift to get that least significant nonzero byte.
     */
    int n = Long.numberOfTrailingZeros(lw ^ rw) & ~0x7;
    return (int) (((lw >>> n) & UNSIGNED_MASK) - ((rw >>> n) & UNSIGNED_MASK));
  }
}
```

Process Byte Arrays in Block

```
for (int i = 0; i < minWords * Longs.BYTES; i += Longs.BYTES) {
  long lw = theUnsafe.getLong(left, BYTE_ARRAY_BASE_OFFSET + (long) i);
  long rw = theUnsafe.getLong(right, BYTE_ARRAY_BASE_OFFSET + (long) i);
  if (lw != rw) {
    if (BIG_ENDIAN) {
      return UnsignedLongs.compare(lw, rw);
    }

    /*
     * We want to compare only the first index where left[index] != right[index].
     * This corresponds to the least significant nonzero byte in lw ^ rw, since lw
     * and rw are little-endian.  Long.numberOfTrailingZeros(diff) tells us the least
     * significant nonzero bit, and zeroing out the first three bits of L.nTZ gives us the
     * shift to get that least significant nonzero byte.
     */
    int n = Long.numberOfTrailingZeros(lw ^ rw) & ~0x7;
    return (int) (((lw >>> n) & UNSIGNED_MASK) - ((rw >>> n) & UNSIGNED_MASK));
  }
}
```

guava-libraries
Guava: Google Core Libraries for Java 1.6+

`com.google.common.primitives.UnsignedBytes`

Process Byte Arrays in Block

```
for (int i = 0; i < minWords * I...                                    {
  long lw = theUnsafe.getLong(...left, BYTE_ARRAY_BASE_OFFSET + (long) i);
  long rw = theUnsafe.getLong(...                                  ...long) i);
  if (lw != rw) {
    if (BIG_ENDIAN) {
      return UnsignedLongs.compare(lw, rw);
    }

    /*
     * We want to compare only the first index where left[index] != right[index].
     * This corresponds to the least significant nonzero byte in lw ^ rw, since lw
     * and rw are little-endian.  Long.numberOfTrailingZeros(diff) tells us the least
     * significant nonzero bit, and zeroing out the first three bits of L.nTZ gives us the
     * shift to get that least significant nonzero byte.
     */
    int n = Long.numberOfTrailingZeros(lw ^ rw) & ~0x7;
    return (int) (((lw >>> n) & UNSIGNED_MASK) - ((rw >>> n) & UNSIGNED_MASK));
  }
}
```

guava-libraries
Guava: Google Core Libraries for Java 1.6+

`com.google.common.primitives.UnsignedBytes`

Process Byte Arrays in Block

```
for (int i = 0; i < [illegible]ongs.BYTES; i += Longs.BYTES) {
  long lw = theUn[illegible]fe.getLong(lef[illegible]BYTE_ARRAY_BASE_OFFSET + (long) i);
  long rw = theUnsa[illegible]ght, BYTE_ARRAY_BASE_OFFSET + (long) i);
  if (lw != rw) {
    if (BIG_ENDIAN) {
      return UnsignedLongs.compare(lw, rw);
    }

    /*
     * We want to compare only the first index where left[index] != right[index].
     * This corresponds to the least significant nonzero byte in lw ^ rw, since lw
     * and rw are little-endian.  Long.numberOfTrailingZeros(diff) tells us the least
     * significant nonzero bit, and zeroing out the first three bits of L.nTZ gives us the
     * shift to get that least significant nonzero byte.
     */
    int n = Long.numberOfTrailingZeros(lw ^ rw) & ~0x7;
    return (int) (((lw >>> n) & UNSIGNED_MASK) - ((rw >>> n) & UNSIGNED_MASK));
  }
}
```

guava-libraries
Guava: Google Core Libraries for Java 1.6+

com.google.common.primitives.UnsignedBytes

Process Byte Arrays in Block

```
for (int i = 0; i < minWords * Longs.BYTES; i += Longs.BYTES) {
  long lw = theUnsafe.getLong(left, BYTE_ARRAY_BASE_OFFSET + (long) i);
  long rw = theUnsafe.getLong(right, BYTE_ARRAY_BASE_OFFSET + (long) i);
  if (lw != rw) {
    if (BIG_ENDIAN) {
      return UnsignedLongs.compare(lw, rw);
    }
        // sanity check - this should never fail
    /*    if (theUnsafe.arrayIndexScale(byte[].class) != 1) {
     * T      throw new AssertionError();
     * a }                                                    st
     * s                                                       us the
     * shift to get that least significant nonzero byte.
     */
    int n = Long.numberOfTrailingZeros(lw ^ rw) & ~0x7;
    return (int) (((lw >>> n) & UNSIGNED_MASK) - ((rw >>> n) & UNSIGNED_MASK));
  }
}
```

guava-libraries
Guava: Google Core Libraries for Java 1.6+

`com.google.common.primitives.UnsignedBytes`

Process Byte Arrays in Block

```
for (int i = 0; i < minWords * Longs.BYTES; i += Longs.BYTES) {
  long lw = theUnsafe.getLong(left, BYTE_ARRAY_BASE_OFFSET + (long) i);
  long rw = theUnsafe.getLong(right, BYTE_ARRAY_BASE_OFFSET + (long) i);
  if (lw != rw) {
    if (BIG_ENDIAN) {
      return UnsignedLongs.compare(lw, rw);
    }
        // sanity check                    never fail
    /*  if (theUnsafe.arrayIndexScale(byte[].class) != 1) {
     * W
     * T     throw new Asse                    ();
     * a   }                                                    st
     * s                                                        us the
     * shift to get that least significant nonzero byte.
     */
    int n = Long.numberOfTrailingZeros(lw ^ rw) & ~0x7;
    return (int) (((lw >>> n) & UNSIGNED_MASK) - ((rw >>> n) & UNSIGNED_MASK));
  }
}
```

guava-libraries
Guava: Google Core Libraries for Java 1.6+

`com.google.common.primitives.UnsignedBytes`

Process Byte Arrays in Block

- Stored contiguously

guava-libraries
Guava: Google Core Libraries for Java 1.6+

`com.google.common.primitives.UnsignedBytes`

Process Byte Arrays in Block

- Stored contiguously
- Endianness

guava-libraries
Guava: Google Core Libraries for Java 1.6+

`com.google.common.primitives.UnsignedBytes`

Process Byte Arrays in Block

- Stored contiguously
- Endianness
- Performance issues

guava-libraries
Guava: Google Core Libraries for Java 1.6+

`com.google.common.primitives.UnsignedBytes`

Atomic Operations

Atomic Operations

```
do
{
    currentValue = get();
    newValue = currentValue + increment;
}
while (!compareAndSet(currentValue, newValue));
```

Atomic Operations

```
do
{
    currentValue = get();
    newValue = currentValue + increment;
}
while (!compareAndSet(currentValue, newValue));
```

LMAX Disruptor `com.lmax.disruptor.Sequence`

Atomic Operations

```
do
{
    currentValue = get();
    newValue = currentValue + increment;
}
while (!compareAndSet(currentValue, newValue));
```

LMAX Disruptor `com.lmax.disruptor.Sequence`

Atomic Operations

```
do
{
    currentValue = get();
    newValue = currentValue + increment;
}
while (!compareAndSet(currentValue, newValue));
```

```
return UNSAFE.compareAndSwapLong(this, VALUE_OFFSET, expectedValue, newValue);
```

LMAX Disruptor `com.lmax.disruptor.Sequence`

Atomic Operations

```
do
{
    currentValue = get();
    newValue = currentValue + increment;
}
while (!compareAndSet(currentValue, newValue));
```

```
return UNSAFE.compareAndSwapLong(this, VALUE_OFFSET, expectedValue, newValue);
```

LMAX Disruptor com.lmax.disruptor.Sequence

Atomic Operations

```
do
{
    currentValue = get();
    newValue = currentValue + increment;
}
while (!compareAndSet(currentValue, newValue));
```

```
return UNSAFE.compareAndSwapLong(this, VALUE_OFFSET, expectedValue, newValue);
```

LMAX Disruptor `com.lmax.disruptor.Sequence`

Atomic Operations

```
do
{
    currentValue = get();
    newValue = currentValue + increment;
}
while (!compareAndSet(currentValue, newValue));
```

```
return UNSAFE.compareAndSwapLong(this, VALUE_OFFSET, expectedValue, newValue);
```

```
VALUE_OFFSET = UNSAFE.objectFieldOffset(Value.class.getDeclaredField("value"));
```

LMAX Disruptor com.lmax.disruptor.Sequence

Atomic Operations

```
do
{
    currentValue = get();
    newValue = currentValue + increment;
}
while (!compareAndSet(currentValue, newValue));
```

```
return UNSAFE.compareAndSwapLong(this, VALUE_OFFSET, expectedValue, newValue);
```

```
VALUE_OFFSET = UNSAFE.objectFieldOffset(Value.class.getDeclaredField("value"));
```

LMAX Disruptor com.lmax.disruptor.Sequence

Atomic Operations

```
do
{
    currentValue = get();
    newValue = currentValue + increment;
}
while (!
```

- Hard to get it right

```
Value));

return UNSAFE.compar
tedValue, newValue);

VALUE_OFFSET = U SAFE.objectFieldOffset(Va e.class.getDeclaredField("value"));
```

LMAX Disruptor com.lmax.disruptor.Sequence

Atomic Operations

```
do
{
    currentValue = get();
    newValue = currentValue + increment;
}
while (!
```

- Hard to get it right
- Java Memory Model

```
value));
```

```
return UNSAFE.compar
```

```
tedValue, newValue);
```

```
VALUE_OFFSET = U SAFE.objectFieldOffset(Va e.class.getDeclaredField("value"));
```

LMAX Disruptor `com.lmax.disruptor.Sequence`

Update Final Fields

Update Final Fields

```
if (WRITE_TO_FINAL_FIELDS) {
    String string = new String();
    UNSAFE.putObject(string, STRING_VALUE_FIELD_OFFSET, chars);
    UNSAFE.putInt(string, STRING_COUNT_FIELD_OFFSET, chars.length);
    return string;
} else {
    return new String(chars);
}
```

Update Final Fields

```
if (WRITE_TO_FINAL_FIELDS) {
    String string = new String();
    UNSAFE.putObject(string, STRING_VALUE_FIELD_OFFSET, chars);
    UNSAFE.putInt(string, STRING_COUNT_FIELD_OFFSET, chars.length);
    return string;
} else {
    return new String(chars);
}
```

`groovy.json.internal.FastStringUtils`

Update Final Fields

```
if (WRITE_TO_FINAL_FIELDS) {
    String [illegible] new String();
    UNSAFE.putObject(string, STRING_VALUE_FIELD_OFFSET, chars);
    UNSAFE.putInt(string, STRING_COUNT_FIELD_OFFSET, chars.length);
    return [illegible]
} else {
    return new String(chars);
}
```

`groovy.json.internal.FastStringUtils`

Update Final Fields

```
if (WRITE_TO_FINAL_FIELDS) {
    String string = new Str...
    UNSAFE.putObject(st..ng, STRING_VALUE_FIELD_OFFSET, c...rs);
    UNSAFE.putInt(stri.., STRING_COUNT_FIELD_OFFSET, chars.length);
    return string;
} else {
    return new String(chars);
}
```

`groovy.json.internal.FastStringUtils`

Update Final Fields

```
if (WRITE_TO_FINAL_FIELDS) {
    String string = new String();
    UNSAFE.putObject(string, STRING_VALUE_FIELD_OFFSET, chars);
    UNSAFE.putInt(string, STRING_COUNT_FIELD_OFFSET, chars.length);
    return string;
} else {
    return new String(chars);
}
```

```
STRING_VALUE_FIELD_OFFSET = getFieldOffset("value");
STRING_COUNT_FIELD_OFFSET = getFieldOffset("count");
```

`groovy.json.internal.FastStringUtils`

Update Final Fields

```
if (WRITE_TO_FINAL_FIELDS) {
    String string = new String();
    UNSAFE.putObject(string, STRING_VALUE_FIELD_OFFSET, chars);
    UNSAFE.putInt(string, STRING_COUNT_FIELD_OFFSET, chars.length);
    return string;
} else {
    return new String(chars);
}
```

```
STRING_VALUE_FIELD_OFFSET = getFieldOffset("value");
STRING_COUNT_FIELD_OFFSET = getFieldOffset("count");
```

```
return UNSAFE.objectFieldOffset(String.class.getDeclaredField(fieldName));
```

groovy.json.internal.FastStringUtils

Update Final Fields

```
if (WRITE_TO_FINAL_FIELDS) {
    String string = new String();
    UNSAFE.putObject(string, STRING_VALUE_FIELD_OFFSET, chars);
    UNSAFE.putInt(string, STRING_COUNT_FIELD_OFFSET, chars.length);
    return string;
} else {
    return new String(chars);
}
```

```
STRING_VALUE_FIELD_OFFSET = getFieldOffset("value");
STRING_COUNT_FIELD_OFFSET = getFieldOffset("count");
```

```
return UNSAFE.objectFieldOffset(String.class.getDeclaredField(fieldName));
```

Groovy

groovy.json.internal.FastStringUtils

Update Final Fields

```
if (WRITE_TO_FINAL_FIELDS) {
    String string = new String();
    UNSAFE.putObject(string, STRING_VALUE_FIELD_OFFSET, chars);
    UNSAFE.putInt(string, STRING_COUNT_FIELD_OFFSET, chars.length);
    return string
} else {
    return new St
}

    STRING_VALU                                              ue");
    STRING_COUNT_FIELD_OFFSET = getFieldOffset("count");

return UNS E.objectFieldOffset(S ring.class.getDeclaredField(fieldName));
```

- Compiler optimizations

Groovy

groovy.json.internal.FastStringUtils

Update Final Fields

```
if (WRITE_TO_FINAL_FIELDS) {
    String string = new String();
    UNSAFE.putObject(string, STRING_VALUE_FIELD_OFFSET, chars);
    UNSAFE.putInt(string, STRING_COUNT_FIELD_OFFSET, chars.length);
    return string
} else {
    return new St
}

    STRING_VALU                                              ue");
    STRING_COUNT_FIELD_OFFSET = getFieldOffset("count");

return UNS  E.objectFieldOffset(S ring.class.getDeclaredField(fieldName));
```

- Compiler optimizations
- Thread visibility

Groovy

`groovy.json.internal.FastStringUtils`

Research Question 4

What **problems** do developers who use *Unsafe* encounter?

Lack of Documentation

Using sun.misc.Unsafe in real world [closed]

Of course the Unsafe class is undocumented, but how can I use it in a real world scenario

141

java

memcpy function in C++ to Java equivalent

5

I have in C++

```
memcpy (&wkpm, (PMSK *)pr_masks + (long)(x - 1), sizeof(PMSK));
```

where `PMSK` is a struct. It will be a class in Java.

Now assuming that here I am copying the whole chunk of memory into pr_masks i.e creating an additional instance of the `PMSK` class. How to do this in Java.

Example: In a java code at line 20 I want capture the class instance and then again use that same instance in line 100. In between there may be many modifications.

Hope I am clear with my question.

Thanks

java c++

Misdirected uses

How to free memory using Java Unsafe, using a Java reference?

0

1

Java Unsafe class allows you to allocate memory for an object as follows, but using this method how would you free up the memory allocated when finished, as it does not provide the memory address...

```
Field f = Unsafe.class.getDeclaredField("theUnsafe"); //Internal reference
f.setAccessible(true);
Unsafe unsafe = (Unsafe) f.get(null);

//This creates an instance of player class without any initialization
Player p = (Player) unsafe.allocateInstance(Player.class);
```

Is there a way of accessing the memory address from the object reference, maybe the integer returned by the default hashCode implementation will work, so you could do...

```
unsafe.freeMemory(p.hashCode());
```

doesn't seem right some how...

java | memory-management | jvm | directmemory

Process Byte Arrays in Block

BUSTED How to speed up a byte[] lookup to be faster using sun.misc.Unsafe?

4

1

I am experimenting with Unsafe to iterate over memory instead of iterating over the values in a byte[]. A memory block is allocated using unsafe. The memory is sufficient to hold 65536 byte values.

I AM TRYING THIS:

```
char aChar = some character

if ((byte) 0 == (unsafe.getByte(base_address + aChar) & mask)){
 // do something
}
```

INSTEAD OF:

```
char aChar = some character

if ((byte) 0 == ( lookup[aChar] & mask )){
 // do something
}
```

asked 3 years a
viewed 2396 tim
active 2 years a

Jobs n

Senior Java Deve
lastminute.com
Chiasso, Switzerl

java

Agile Coach - Jav
lastminute.com
Chiasso, Switzerl

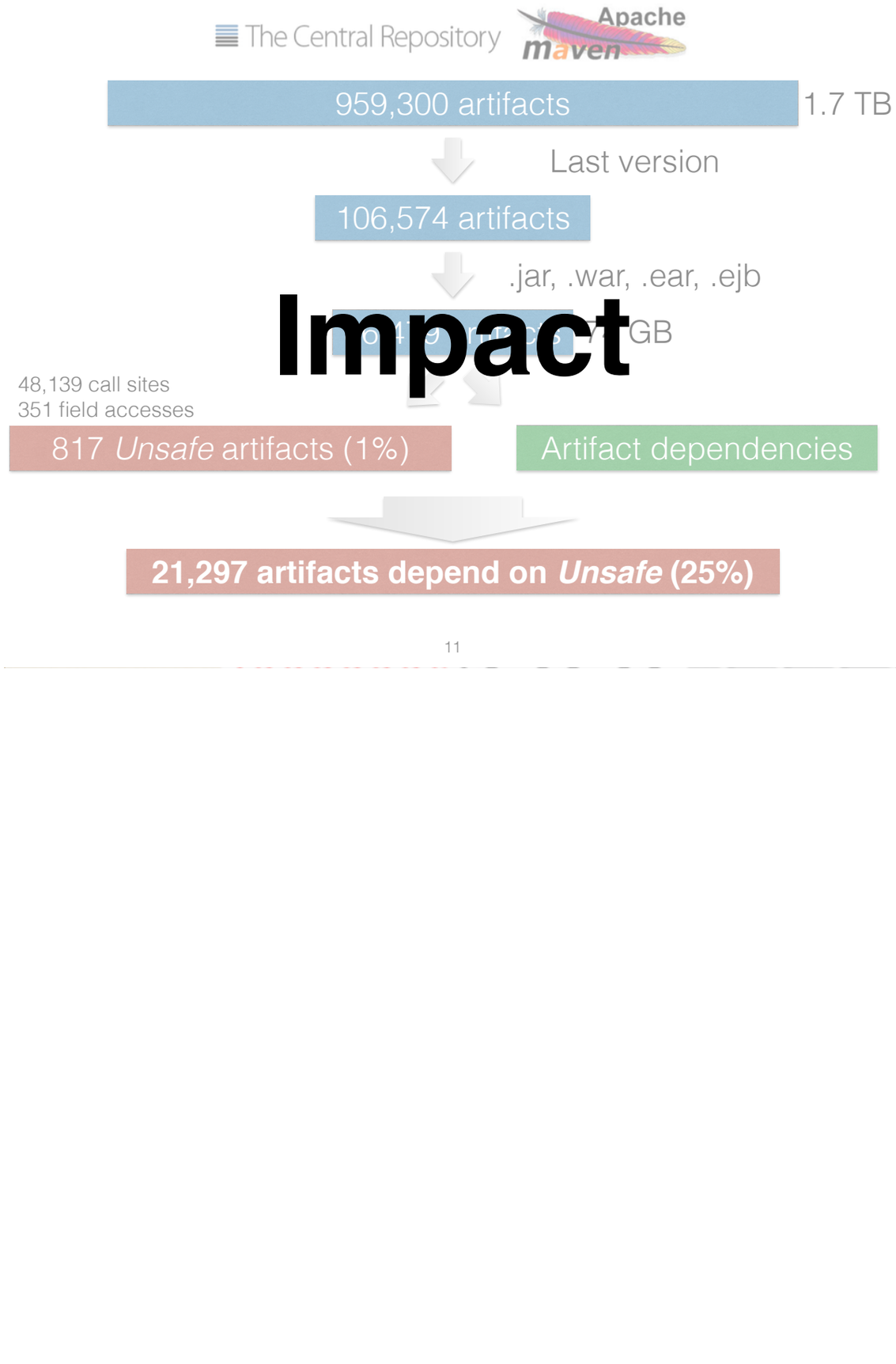
More jobs near L

https://bitbucket.org/acuarica/java-unsafe-analysis

luis.mastrangelo@usi.ch

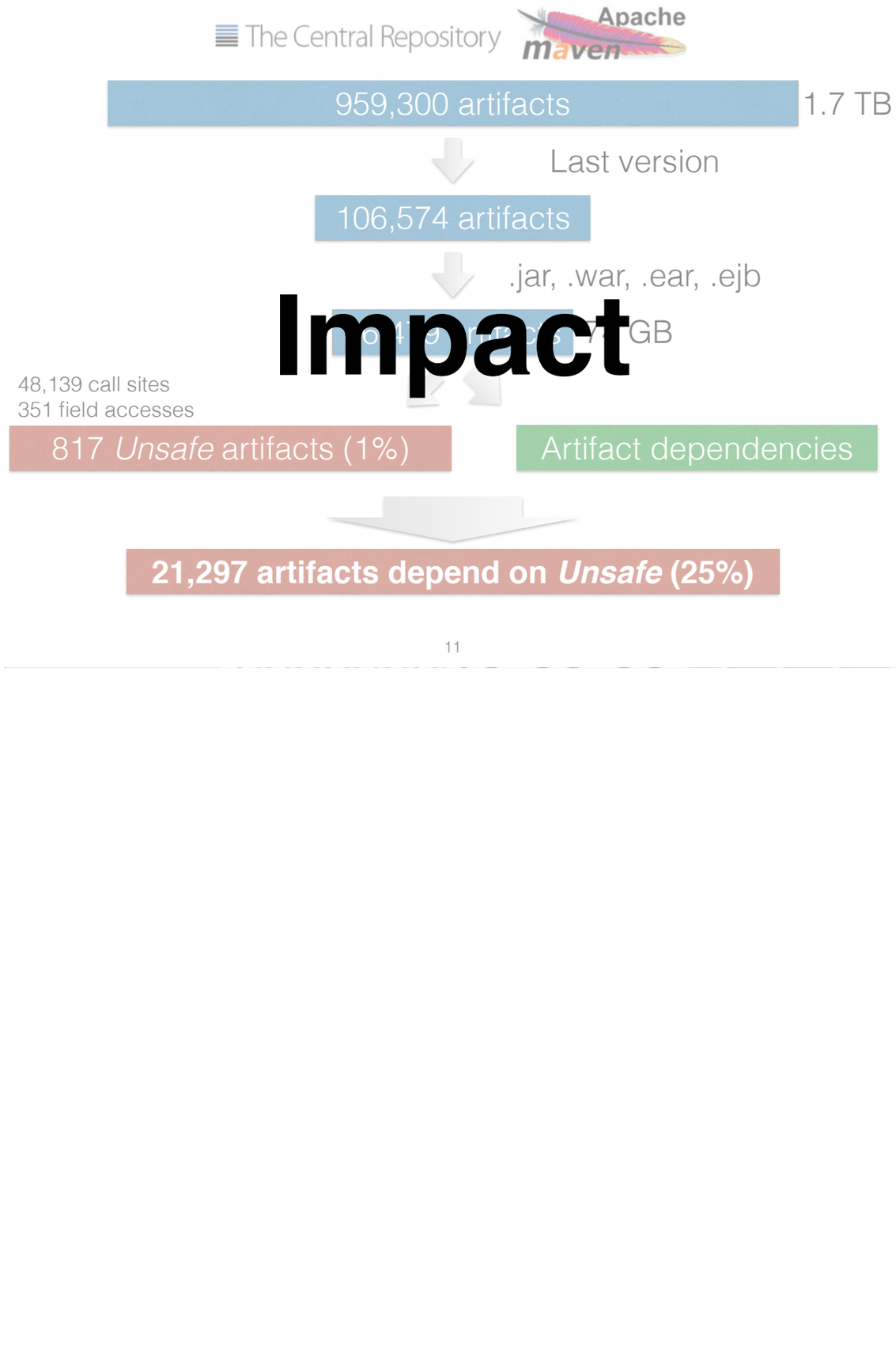
https://bitbucket.org/acuarica/java-unsafe-analysis

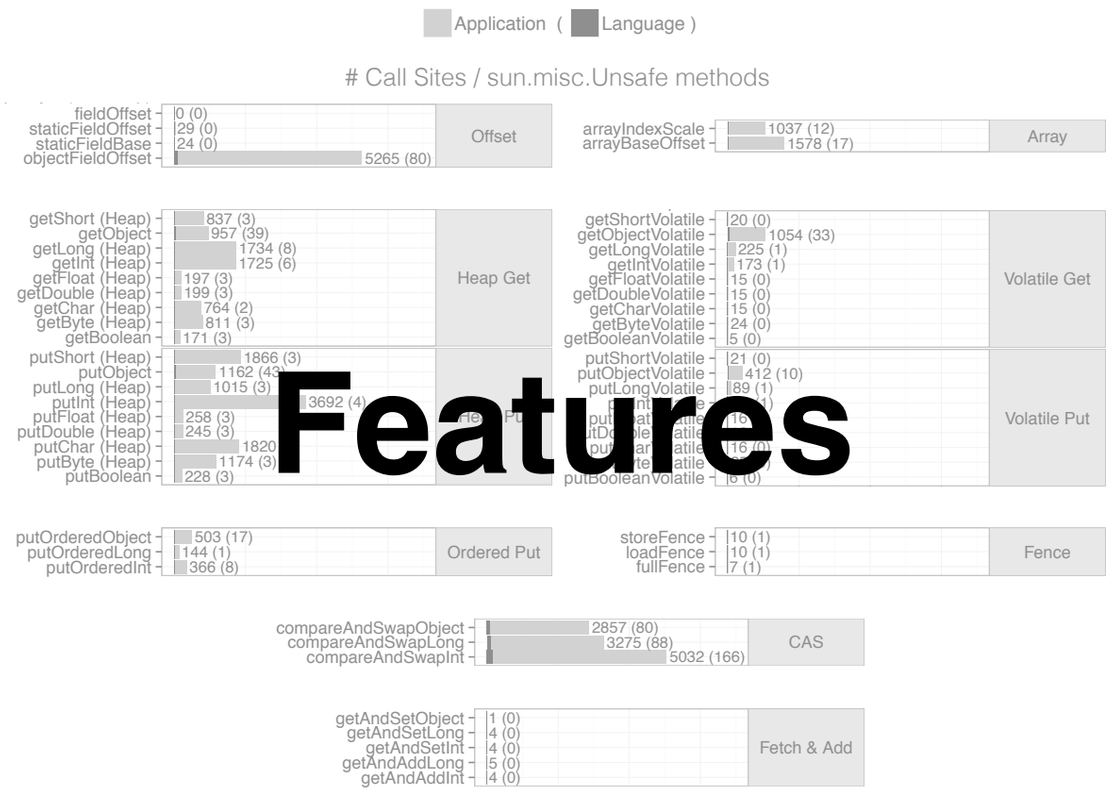
luis.mastrangelo@usi.ch



https://bitbucket.org/acuarica/java-unsafe-analysis

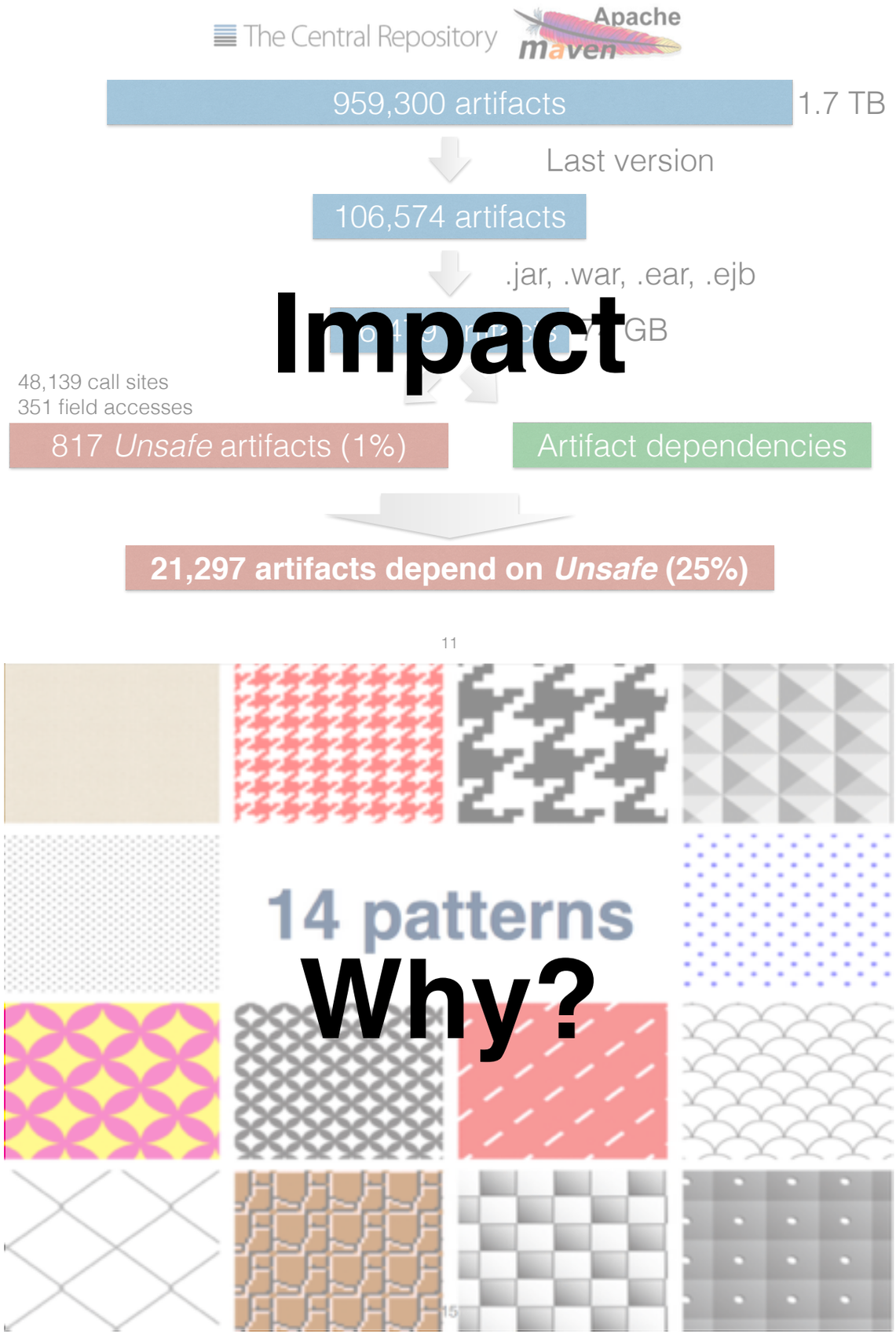
luis.mastrangelo@usi.ch

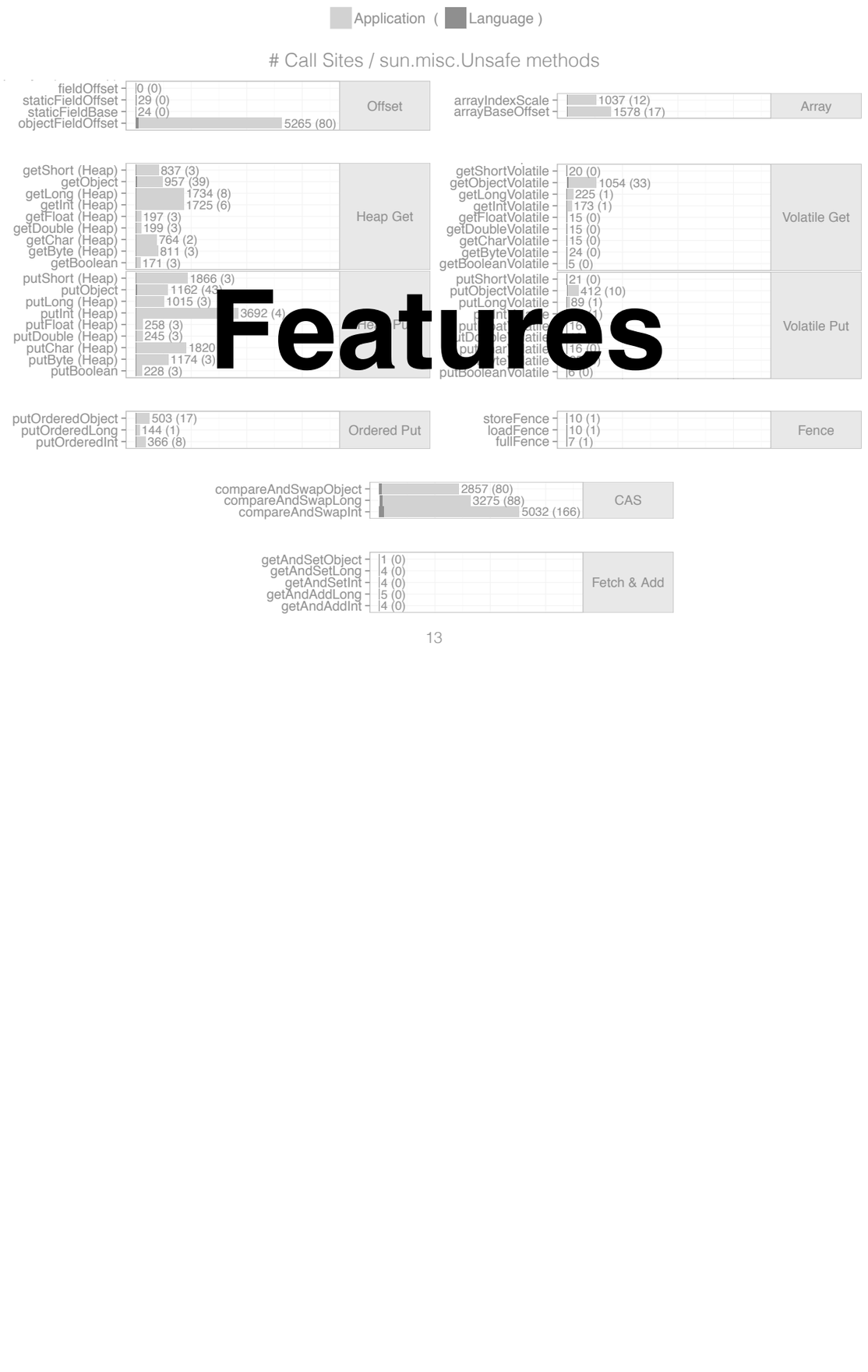




https://bitbucket.org/acuarica/java-unsafe-analysis

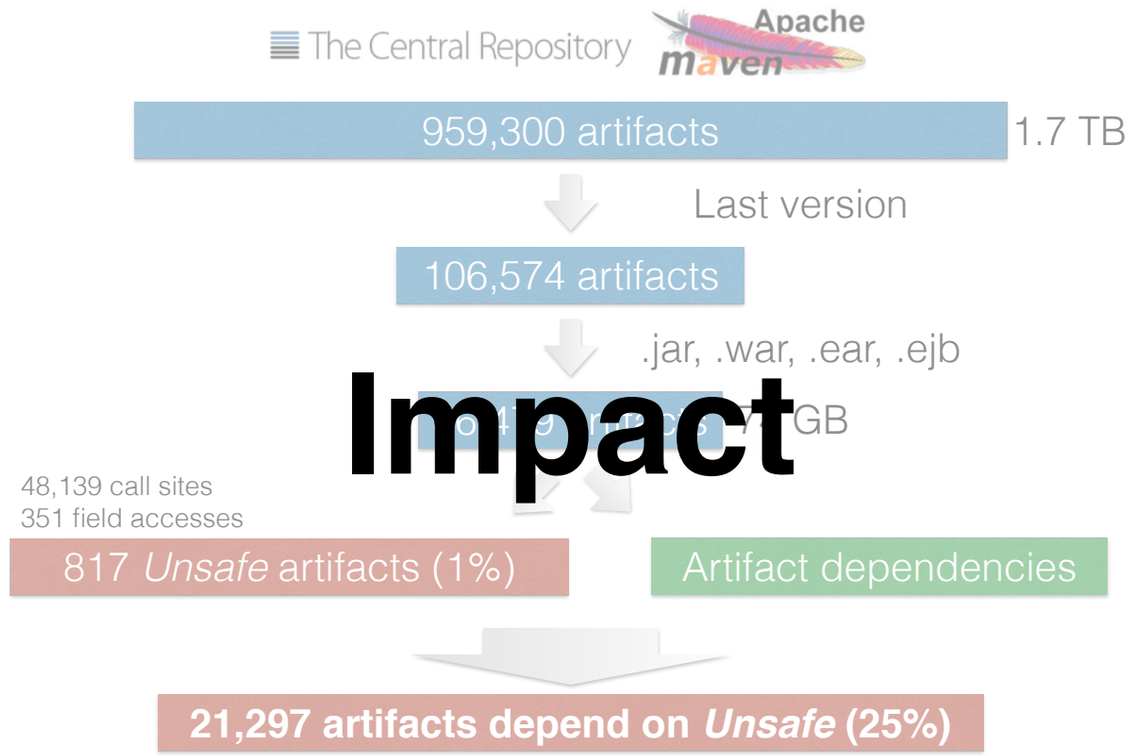
luis.mastrangelo@usi.ch

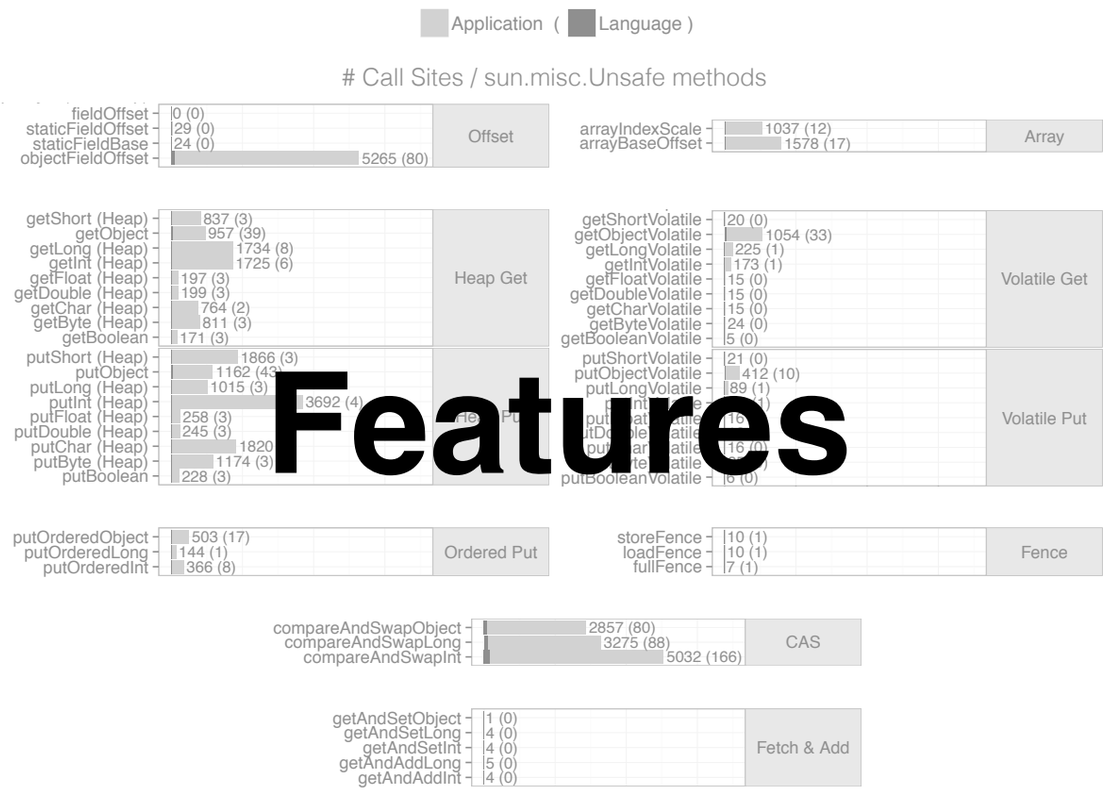


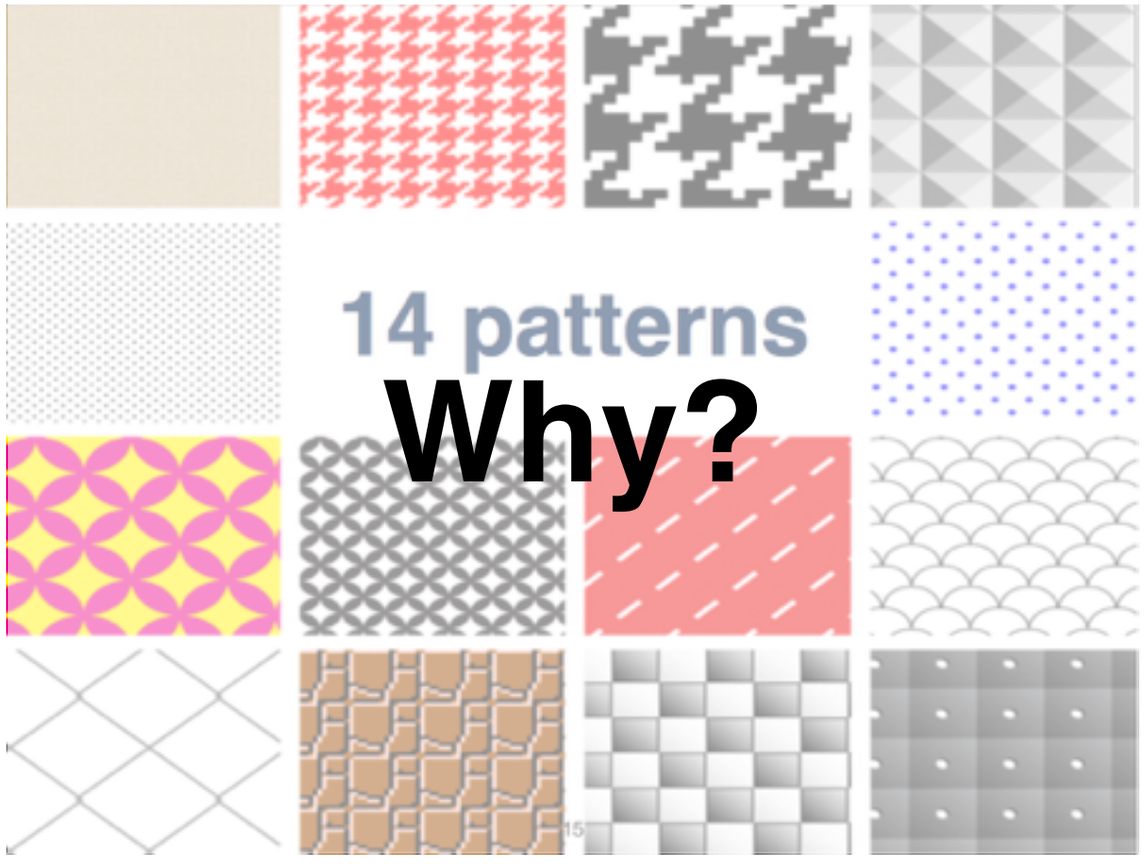


https://bitbucket.org/acuarica/java-unsafe-analysis

luis.mastrangelo@usi.ch







Misdirected uses

How to free memory using Java Unsafe, using a Java reference?

0

1

Java Unsafe class allows you to allocate memory for an object as follows, but using this method how would you free up the memory allocated when finished, as it does not provide the memory address...

```
Field f = Unsafe.class.getDeclaredField("theUnsafe"); //Internal reference
f.setAccessible(true);
Unsa e = (Unsafe) f.ge ull);

//Th
Play p = (Pl er) saf
```

Problems

Is there a way of accessing the memory address from the object reference, maybe the integer returned by the default hashCode implementation will work, so you could do...

```
unsafe.freeMemory(p.hashCode());
```

doesn't seem right some how...

java memory-management jvm directmemory

https://bitbucket.org/acuarica/java-unsafe-analysis

luis.mastrangelo@usi.ch

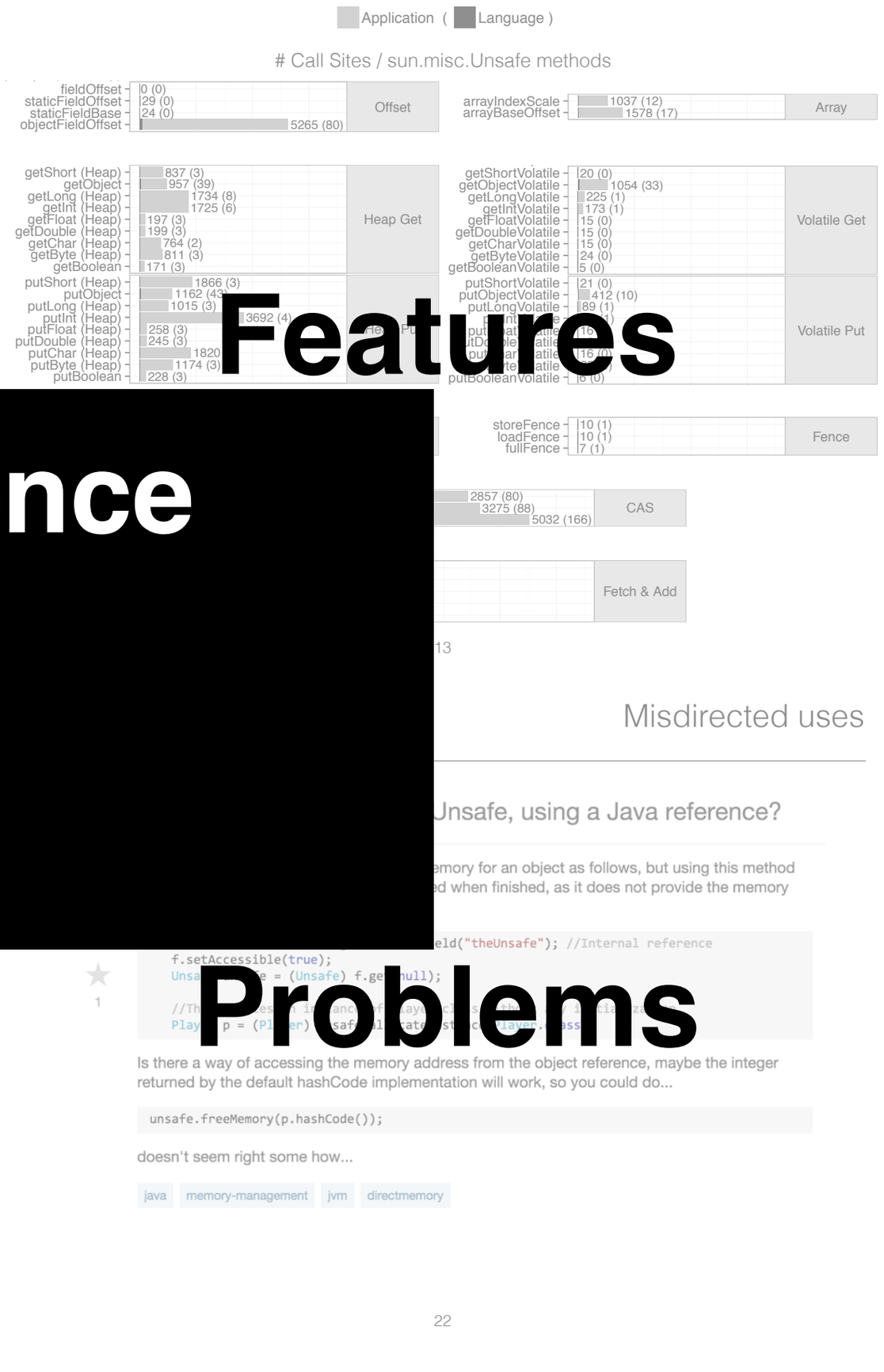
Impact

Features

Evidence

Why?

Problems



Misdirected uses

Unsafe, using a Java reference?

...emory for an object as follows, but using this method ...ed when finished, as it does not provide the memory

```
...eld("theUnsafe"); //Internal reference
f.setAccessible(true);
Unsa...e = (Unsafe) f.ge...ull);
//Th... ...
Play... p = (Pl...er) ...saf... ...ate... ...Player.class...
```

Is there a way of accessing the memory address from the object reference, maybe the integer returned by the default hashCode implementation will work, so you could do...

```
unsafe.freeMemory(p.hashCode());
```

doesn't seem right some how...

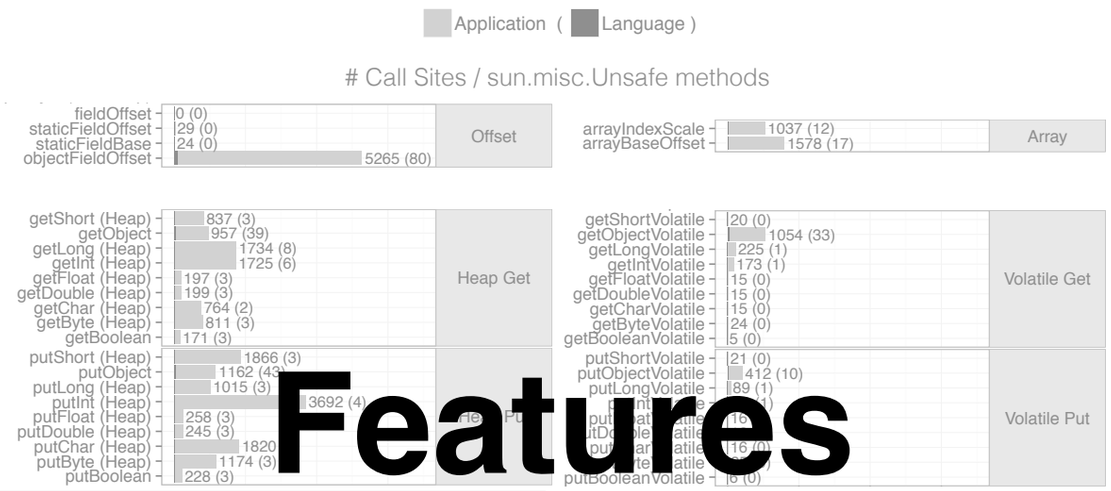
java memory-management jvm directmemory

https://bitbucket.org/acuarica/java-unsafe-analysis

luis.mastrangelo@usi.ch

Impact

Features

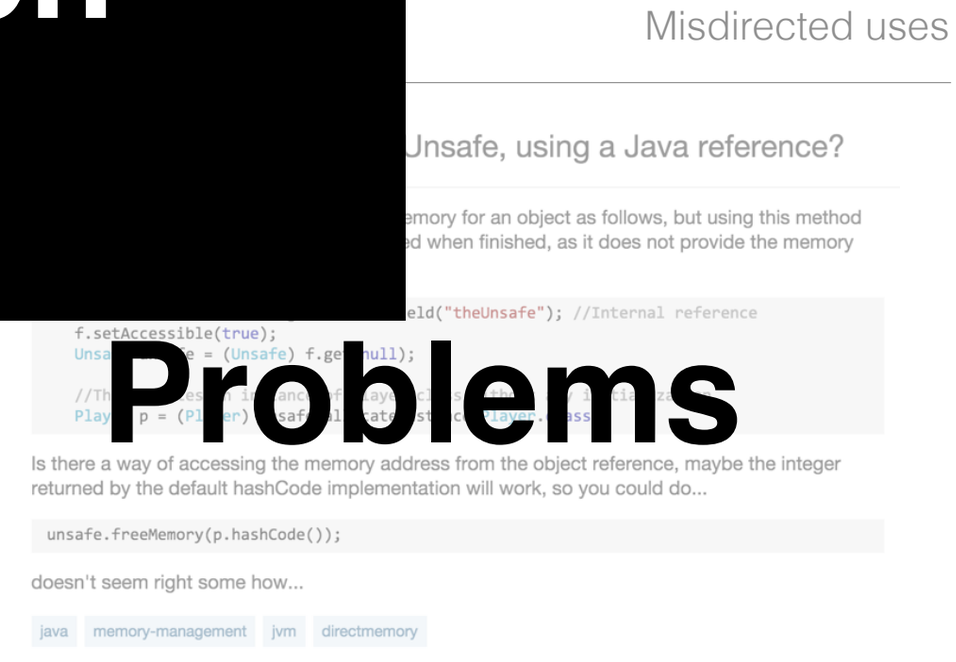


Evidence

Evolution

Why?

Problems



https://bitbucket.org/acuarica/java-unsafe-analysis

luis.mastrangelo@usi.ch

Impact

Features

Evidence

Evolution

~~Un~~safe

Why?

Problems