Mining Ultra-Large-Scale Software Repositories and StackOverflow database to study *sun.misc.Unsafe* API usage patterns in Java applications

Luis Mastrangelo*, Matthias Hauswirth* and Nate Nystrom*
*Faculty of Informatics, University of Lugano

***Abstract*—**

The Java language and Java Virtual Machine (JVM) evolved over time to satisfy the needs for millons of developers world-wide. Performance is one of the main drivers why the Java language changed. One of this changes is the addition of an undocumented class, *sun.misc.Unsafe*, provided by Oracle. Although *sun.misc.Unsafe* allows the developer to access low-level programming features, its use is extremely dangerous. In some cases, its use can lead to a crash of the JVM.

In this paper, we study how effectively this API is used in industry. To carry out our study, we mine ultra-large-scale software repositories and the StackOverflow database. The goal of our study is to answer the following question: Why and how the Unsafe API is used in Java projects?

Our aim is to devise safer alternatives to *sun.misc.Unsafe* on the JVM to improve programmer productivity.

I. Introduction

The Java Virtual Machine (JVM) executes Java bytecode and provides other services for programs written in Java, Scala, Clojure, and many other languages. Although the JVM was designed to be portable, "write once run anywhere", many JVM implementations in wide use expose an API to allow access to low-level, non-portable features of the JVM. This API is provided through an undocumented [1] class, *sun.misc.Unsafe*, in the Java reference implementation produced by Oracle. The class allows the developer to access low-level programming features.

Other virtual machines provide similar functionality. For instance the language C# provides an `unsafe` construct on the .NET platform.[2]

Use of *sun.misc.Unsafe* has increased recently [FIXME citation]. The main reason for this trend *sun.misc.Unsafe* is *performance*. *sun.misc.Unsafe* provides methods to allow the programmer to access low-level details of the virtual machine and underlying hardware that would otherwise be impossible. For instance, *sun.misc.Unsafe* contains methods to do compare-and-swap (CAS) operations, needed to implement lock-free data structures.

The operations *sun.misc.Unsafe* provides can be dangerous. If misused they can cause performance problems, resource leaks, deadlock, data corruption, and even VM crashes.

Because of the danger of using *sun.misc.Unsafe*, we aim to study how the API is used in practice, with the longer term goal of providing safer alternatives to *sun.misc.Unsafe* on the JVM and on other virtual machines in general.

Sandoz [1] describe several usage patterns of *sun.misc.Unsafe*. Let's show of these examples.

Sandoz [1] did a survey to study how Unsafe is used [3].

In this paper, we go beyond that survey and look at how *sun.misc.Unsafe* is used in the wild. We systematically examine projects open source projects from the SourceForge repository using BOA [2] and analyze how these projects use *sun.misc.Unsafe*. Moreover, we study problems encountered using *sun.misc.Unsafe* by analyzing the StackOverflow question/answer database.

Trend using unsafe methods. But what for? Certainly you can do everithing without Unsafe API, so why using it?

Stackoverflow mining.

Measure error with boa.

Bugreport stackoverflow posts.

Overall, the main contributions of this paper are two-fold:

- We present a detailed study of how the Java *sun.misc.Unsafe* API is used and
- We constrast this information on why this API is used based on responses from Stackoverflow.

The rest of this paper is organized as follows: Section II presents related work. Section **??** explains the methodology and technologies used to get our results. Section V shows the results we obtained and Section VI concludes.

II. Related Work

GHTorrent [3] provide GitHub quering but for metadata, not source code mining, true?

Lean GHtorrent [4] provides mining of GitHub repositories, but for a limit impose by GitHub, only 1,000 repositories can be queried.

Boa For source code mining we searched in Boa [2].

[5]

Paul Sandoz estimates how Unsafe is used [1], but only provides a survey.

Several research work uses *sun.misc.Unsafe* in their implementation.

As far as we know, no prior work shows how the *sun.misc.Unsafe* is used in industry.

[1] http://www.oracle.com/technetwork/java/faq-sun-packages-142232.html

[2] http://msdn.microsoft.com/en-us/en-en/library/chfa2zb8(v=vs.90).aspx

[3] http://www.infoq.com/news/2014/02/Unsafe-Survey

III. Mining Boa infrastructure

In this section we describe our methodology to mine the data we used for our analysis. We first begin describing how we mined source code repositories. The next section describes how we analysed the Stackoverflow database.

The complete scripts and results used for our study are available online [4].

A. Static usage

For source code mining we searched in Boa [2]. The BOA infrastructure allows the user to navigate the parsed AST of source code. Notice that BOA does not provide type resolution analysis. Therefore we develop some heuristics to determine where and how *sun.misc.Unsafe* is used.

The following assumptions hold:

- *sun.misc.Unsafe* class is `final`
- inherits directly from `java.lang.Object`
- Its public methods (except for `getUnsafe`) are instance methods

Thus, our Boa script [5] looks for `sun.misc.Unsafe` as either an import or fully qualified name where a type may appear. In case that we found a use of *sun.misc.Unsafe* we proceed to determine which method is used. If *sun.misc.Unsafe* is found in a compilation unit, then all call sites in the compilation unit are analysed. If a call site target is one of the *sun.misc.Unsafe* methods, then we can conclude that *sun.misc.Unsafe* is used and that method is called.

B. Reflection usage

What happens if *sun.misc.Unsafe* is used through reflection? Sometimes this API is used through reflection to avoid the compilation dependency (remember that *sun.misc.Unsafe* is not part of the public API).

For instance, listing 1 shows how to take the address size of the host machine. First it obtains an instance of *sun.misc.Unsafe* by reflection. Then, again by reflection, invoke the method `addressSize`, which returns the address size of the host machine.

```
Class<?> cls = Class.forName("sun.misc.Unsafe");
Field theUnsafe = cls.getDeclaredField("theUnsafe");
theUnsafe.setAccessible(true);
Object unsafe = theUnsafe.get(null);
Method method = cls.getMethod("addressSize");
Object value = method.invoke(unsafe);
int addressSize = ((Number) value).intValue();
```

Listing 1. Use of sun.misc.Unsafe with reflection

We are aware of this situation and also look for the string literal `"sun.misc.Unsafe"` to see when it is used through reflection.

[4]https://bitbucket.org/acuarica/java-unsafe-analysis

[5]https://bitbucket.org/acuarica/java-unsafe-analysis/raw/master/unsafe-analysis/unsafe.boa

C. Avoiding duplicates

Usually a code repository has branches, to allow the team to introduce new features without interrupt the main source of development. In Boa, all *sun.misc.Unsafe* uses are reported, regardless if it is the same file in different branches. Sometimes it happens that the same file, with the same uses, has different package in different branches.

Therefore we need to take into account this situation. To determine if a file is a duplicate in another branch, we look for the same file name and uses, if we found the same file and the same uses, we assume that the file is in a branch and we exclude it.

D. JDK forks

Our script also retrieves the package of the compilation unit. With the package we can determine if the compilation unit belongs to a JDK implementation.

E. Metadata

Also we are interested in some metadata information. For each project that uses *sun.misc.Unsafe*, either statically or dinamically, our Boa script also retrieves :

- the size of the project (in number of AST nodes)
- the number of revisions in the repository.
- the lifetime of the repository (time from the first to the last commit)

It is possible to group methods in *sun.misc.Unsafe* by functionality. Table II shows all methods (without overloads) grouped by functionality.

TABLE I
Functional groups of *sun.misc.Unsafe*

Group	Methods
Array	arrayBaseOffset arrayIndexScale
CAS	compareAndSwapInt compareAndSwapLong compareAndSwapObject
Class	defineAnonymousClass defineClass ensureClassInitialized
Get	getBoolean getByte getChar getDouble getFloat getInt getIntVolatile getLoadAverage getLong getLongVolatile getObject getObjectVolatile getShort getBooleanVolatile getDoubleVolatile getFloatVolatile getByteVolatile getCharVolatile getShortVolatile
Memory	addressSize allocateMemory copyMemory freeMemory getAddress pageSize putAddress reallocateMemory setMemory
Offset	fieldOffset objectFieldOffset staticFieldBase staticFieldOffset
Park	park unpark
Put	putBoolean putByte putChar putDouble putFloat putInt putIntVolatile putLong putLongVolatile putObject putObjectVolatile putOrderedInt putOrderedLong putOrderedObject putShort putCharVolatile putOrderedInt putBooleanVolatile putShortVolatile putFloatVolatile putByteVolatile putDoubleVolatile
Single	allocateInstance throwException
Monitor	monitorEnter monitorExit tryMonitorEnter

IV. Mining StackOverflow

In this section we describe our methodology to retrieve the data we used for our analysis. We first begin describing how

we mined source code repositories and then how we analysed the Stackoverflow database.

The complete scripts and results used for our study are available online [6].

A. Source Code repositories

For source code mining we searched in Boa [2]. The BOA infrastructure allows the user to navigate the parsed AST of source code.

Our Boa script looks for `sun.misc.Unsafe` as either an import or fully qualified name where a type may appear. In case that we found a use of *sun.misc.Unsafe* we proceed to determine which method is used.

It is possible to group methods in *sun.misc.Unsafe* by functionality. Table II shows all methods (without overloads) grouped by functionality.

TABLE II
FUNCTIONAL GROUPS OF *sun.misc.Unsafe*

Group	Methods
Array	arrayBaseOffset arrayIndexScale
CAS	compareAndSwapInt compareAndSwapLong compareAndSwapObject
Class	defineAnonymousClass defineClass ensureClassInitialized
Get	getBoolean getByte getChar getDouble getFloat getInt getIntVolatile getLoadAverage getLong getLongVolatile getObject getObjectVolatile getShort getBooleanVolatile getDoubleVolatile getFloatVolatile getByteVolatile getCharVolatile getShortVolatile
Memory	addressSize allocateMemory copyMemory freeMemory getAddress pageSize putAddress reallocateMemory setMemory
Offset	fieldOffset objectFieldOffset staticFieldBase staticFieldOffset
Park	park unpark
Put	putBoolean putByte putChar putDouble putFloat putInt putIntVolatile putLong putLongVolatile putObject putObjectVolatile putOrderedInt putOrderedLong putOrderedObject putShort putCharVolatile putOrderedInt putBooleanVolatile putShortVolatile putFloatVolatile putByteVolatile putDoubleVolatile
Single	allocateInstance throwException
Monitor	monitorEnter monitorExit tryMonitorEnter

Reflection: What happens with other uses such as reflection? It is not detected but it uses Unsafe. There should be a way to measure this kind of use.

Look for problematic uses of the API, and some use patterns.

B. Stackoverflow

Google search for sun.misc.unsafe site:stackoverflow.com returns about 1,360 results.

V. RESULTS

The Figure **??** shows the pie.

Total number of projects: 699331

Total number of Java projects: 50692 (7.25%)

Total number of projects: 49 (0.1%)

The figure 1 shows how many times a method is called. Grouped by functional group.

[6]https://bitbucket.org/acuarica/java-unsafe-analysis

The most called is `objectFieldOffset`. Because the result is then used by many other calls to Unsafe.

A. Stackoverflow

Searching for the term: ”unsafe java” on stackoverflow returns 1,241 results. While searching for only the term ”sun.misc.unsafe” returns 318 results.

Representative SO ids:

http://stackoverflow.com/questions/13003871/how-do-i-get-the-instance-of-sun-misc-unsafe

http://stackoverflow.com/questions/18220435/using-sun-misc-unsafe-what-is-the-fastest-way-to-scan-bytes-from-a-direct-byteb

http://stackoverflow.com/questions/5761702/can-one-break-a-secury-manager-with-sun-misc-unsafe

http://stackoverflow.com/questions/22242836/strange-behaviour-of-sun-misc-unsafe-put-on-solaris-sparcv9

http://stackoverflow.com/questions/7934779/using-sun-misc-unsafe-to-get-address-of-java-array-items

http://stackoverflow.com/questions/9323416/using-memory-allocated-by-sun-misc-unsafe-allocatememory-in-native-code

http://stackoverflow.com/questions/26995856/strange-behavior-in-sun-misc-unsafe-compareandswap-measurement-via-jmh

http://stackoverflow.com/questions/21589159/java-sun-misc-unsafe-confusion

http://stackoverflow.com/questions/24241335/waiting-at-sun-misc-unsafe-parknative-method

http://stackoverflow.com/questions/20494387/how-unsafe-is-the-use-of-sun-misc-unsafe-actually

http://stackoverflow.com/questions/12972918/why-does-park-unpark-have-60-cpu-usage

http://stackoverflow.com/questions/12638761/getting-error-as-sun-misc-unsafe-cannot-be-resolved-while-modifying-library

http://stackoverflow.com/questions/18687243/using-sun-misc-usafe-as-off-heap-memory-and-writing-memory-managers

http://stackoverflow.com/questions/12226123/busted-how-to-speed-up-a-byte-lookup-to-be-faster-using-sun-misc-unsafe

http://stackoverflow.com/questions/16723244/dealing-with-16-bit-characters-using-sun-misc-unsafe

http://stackoverflow.com/questions/7823665/why-jni-call-to-native-method-is-slower-than-similar-in-sun-misc-unsafe

http://stackoverflow.com/questions/1490760/sun-misc-unsafe-how-to-get-the-bytes-from-an-address

http://stackoverflow.com/questions/25234679/why-is-sun-misc-unsafe-unpark-described-unsafe

http://stackoverflow.com/questions/17671066/java-direct-memory-using-sun-misc-cleaner-in-custom-classes

http://stackoverflow.com/questions/6042858/can-i-override-object-with-sun-misc-unsafe

http://stackoverflow.com/questions/8462200/examples-of-forcing-freeing-of-native-memory-direct-bytebuffer-has-allocated-us

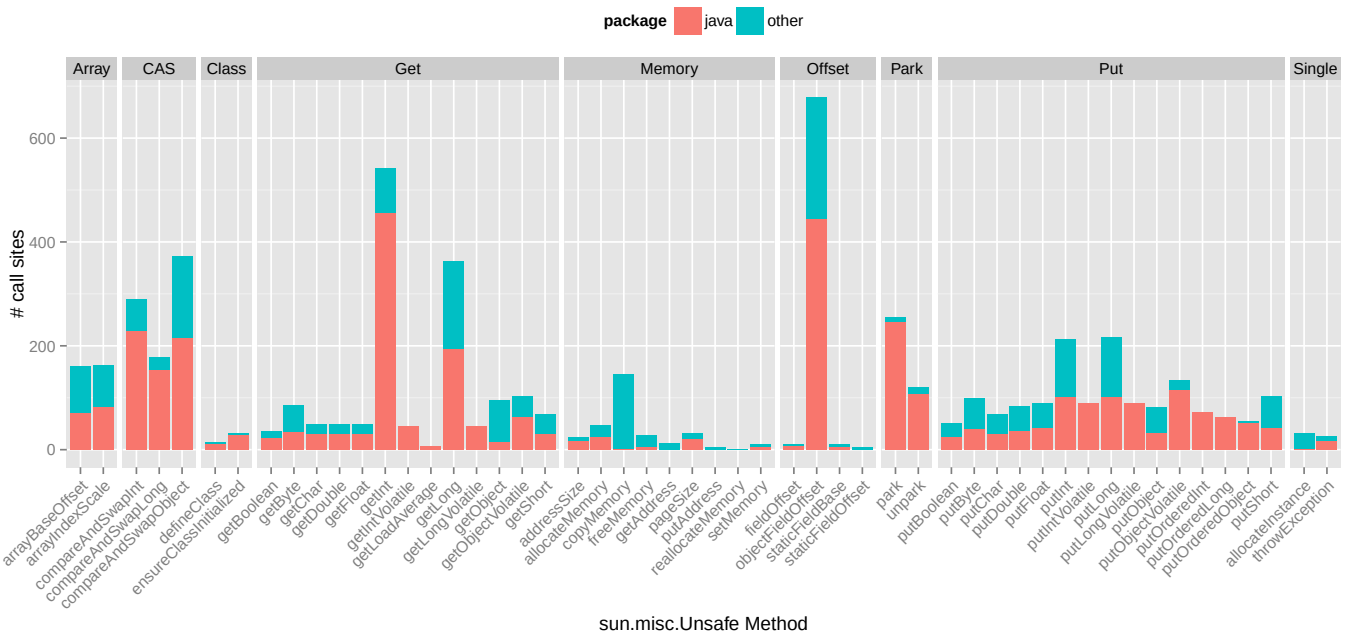

Fig. 1. sun.misc.Unsafe methods usage

http://stackoverflow.com/questions/22846538/correct-use-of-arraybaseoffset-and-arrayindexscale

http://stackoverflow.com/questions/20157508/is-there-a-way-to-force-unload-a-class-by-using-the-sun-misc-unsafe-class

http://stackoverflow.com/questions/23709378/modifying-memory-via-unsafe-causing-exception-access-violation

VI. Conclusions

Although the current use of *sun.misc.Unsafe* seems low in SourceForge, it is important to notice the snapshot is from September 2013. It would be interesting to apply the same analysis but to the current GitHub source code database. Unfortunately at the moment we could not find any full dataset from GuitHub.

We strongly believe that this study will help us to develop our language.

Acknowledgments

The first author was supported by Swiss National Science Foundation grant CRSII2_136225.

References

[1] P. Sandoz, "Safety Not Guaranteed: sun.misc.Unsafe and the quest for safe alternatives," http://cr.openjdk.java.net/~psandoz/dv14-uk-paul-sandoz-unsafe-the-situation.pdf, 2014, Oracle Inc. [Online; accessed 29-January-2015].

[2] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen, "Boa: A language and infrastructure for analyzing ultra-large-scale software repositories," in *Proceedings of the 35th International Conference on Software Engineering*, ser. ICSE'13, May 2013, pp. 422–431.

[3] G. Gousios, "The ghtorrent dataset and tool suite," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 233–236. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487132

[4] G. Gousios, B. Vasilescu, A. Serebrenik, and A. Zaidman, "Lean ghtorrent: Github data on demand," in *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*, M. Pinzger, S. Kim, and P. Devanbu, Eds. ACM, 2014, pp. 384–387. [Online]. Available: http://dx.doi.org/10.1145/2597073.2597126

[5] R. Dyer, H. Rajan, H. A. Nguyen, and T. N. Nguyen, "Mining billions of AST nodes to study actual and potential usage of Java language features," in *36th International Conference on Software Engineering*, ser. ICSE'14, June 2014, pp. 779–790.

TABLE III
JAVA PROJECTS USING *sun.misc.Unsafe*

#	Name	Description	# AST Nodes	# Revisions	Lifetime	# smU Calls	# smU Literal
1	adtools	Amiga Development Tools (adtools)	14911 k	466	6 years	421	0
2	amino	Concurrent Building Block	255 k	691	4 years	53	0
3	amock	Java Mock libarary for static method	13 k	3	1 month	14	0
4	android	Android on PXA270	4836 k	146	1 year	77	0
5	aojunit	An aspect-oriented extension to JUnit	1 k	5	1 day	1	0
6	archaiosjava	Scalable and fast libraries for Java	11 k	17	14 days	102	0
7	beanlib	Java Bean Library	119 k	854	6 years	4	0
8	caloriecount	Track what you eat	352 k	202	5 months	10	0
9	cegcc	CeGCC - Cross development for Pocket PC	2203 k	1449	4 years	101	0
10	cgnu	CGNU (Clean GNU)	2135 k	60	1 month	101	0
11	classreach	Identifies unused Java classes and methods	196 k	69	2 years	10	0
12	clipc	Library for IPC	228 k	278	6 months	10	0
13	concutest	Tools to test concurrent Java programs	550 k	14	2 years	185	0
14	ec	ec-gin Europe China Grid InterNetworking	635 k	9	2 months	10	1
15	essence	Essence Java Framework	157 k	293	2 years	75	0
16	essentialbudget	Essential Budget	60 k	55	4 months	20	2
17	glassbox	Troubleshooting and monitoring agent	99 k	458	4 years	1	0
18	grinder	Load testing framework	770 k	4334	11 years	6	0
19	high	Highly Scalable Java	37 k	78	2 years	37	0
20	hlv	Collection of high level view plugins for eclipse	33 k	278	1 year	0	4
21	ikvm	JVM for .NET Framework and Mono	531 k	3980	10 years	123	0
22	jadoth	abstraction utils and frameworks	619 k	2922	3 years	949	0
23	janetdev	Ja.NET - Java Development Tools for .NET	10034 k	366	2 years	280	0
24	janux	Java directly on the Linux Kernel	564 k	25	1 month	10	0
25	java	Lightweight Java Game Library	571 k	3841	10 years	6	0
26	javapathfinder	Verifies Java bytecode programs	9952 k	4038	6 years	0	4
27	javapayload	Payloads to be used for post-exploitation	74 k	92	2 years	28	1
28	jaxlib	Platform independent Java library	5405 k	3208	11 years	42	3
29	jigcell	Computational biology problem solving	3573 k	5286	8 years	0	3
30	jikesrvm	The Jikes Research Virtual Machine (RVM)	9026 k	16068	10 years	32	16
31	jnode	JNode: new Java Operating System	44401 k	11972	10 years	2104	0
32	jon	Java Object Notation	29 k	118	1 year	3	0
33	jprovocateur	RAD for Ajax applications in Java	197 k	934	2 years	10	2
34	junitrecorder	Record test cases	34 k	18	2 months	1	0
35	katta	Lucene in the cloud	169 k	478	1 year	31	0
36	l2next	L2 Private Server code	39 k	22	1 month	26	0
37	lockss	Lots of Copies Keep Stuff Safe	11551 k	23048	11 years	0	2
38	neurogrid	P2P Bookmark Organiser	337 k	738	5 years	2	0
39	osfree	osFree operating system	119 k	1124	5 years	96	0
40	ps2toolchain	Toolchain for the Playstation 2's	4298 k	8	1 day	202	0
41	simulaeco	Semester project	136 k	66	4 months	10	2
42	snarej	Snare's Not A Risc OS Emulator in Java	111 k	82	1 month	19	0
43	statewalker	Graph traversing library	477 k	432	3 years	36	2
44	takatuka	TakaTuka Java Virtual Machine	1176 k	2637	3 years	107	0
45	timelord	A tool for estimating and tracking time	697 k	546	2 years	40	0
46	ucl	A final year project by UCL students	1639 k	70	3 months	0	1
47	vcb	Component Based Development tool	602 k	2446	3 years	11	0
48	x10	Experimental language for DARPA/HPCS	12292 k	25432	9 years	279	0
49	xbeedriver	Driver for the ZigBee network	119 k	6	3 days	10	2