Глубже стек-трейсов
Шире хип-дампов

Andrei Pangin
andrey.pangin@corp.mail.ru

Questions on StackOverflow

1. http://stackoverflow.com/questions/36343209/which-part-of-throwing-an-exception-is-expensive
2. http://stackoverflow.com/questions/26140182/running-jmap-getting-unable-to-open-socket-file
3. http://stackoverflow.com/questions/35517934/why-does-the-count-of-calls-of-a-recursive-method-causing-a-stackoverflowerror-v
4. http://stackoverflow.com/questions/28631656/runnable-thread-state-but-in-object-wait
5. http://stackoverflow.com/questions/26795573/which-method-is-the-least-obtrusive-for-generating-thread-dumps-in-java
6. http://stackoverflow.com/questions/25701740/java-thread-dump-waiting-on-object-monitor-what-is-it-waiting-on
7. http://stackoverflow.com/questions/36588354/how-to-dump-java-objects-came-from-jvm-heap-old-generation
8. http://stackoverflow.com/questions/30530082/how-can-i-restart-jvm-on-outofmemoryerror-after-making-a-heap-dump
9. http://stackoverflow.com/questions/28767905/what-is-the-difference-between-xss-and-xxthreadstacksize
10. http://stackoverflow.com/questions/25309748/what-is-thread-stack-size-option-xss-given-to-jvm-why-does-it-have-a-limit-of
11. http://stackoverflow.com/questions/35792936/how-to-take-a-heap-dump-in-windows-with-minimum-downtime
12. http://stackoverflow.com/questions/34076680/java-out-of-memory-automatic-heap-dump-file-name
13. http://stackoverflow.com/questions/26548103/how-to-obtain-the-jvm-thread-name-id-through-the-known-pid-tid
14. http://stackoverflow.com/questions/26116181/access-stacktraces-with-good-performance
15. http://stackoverflow.com/questions/23552296/how-to-determine-hotspot-vm-default-thread-stack-size
16. http://stackoverflow.com/questions/36458490/jni-how-to-list-all-the-current-instances
17. http://stackoverflow.com/questions/36279207/get-a-heap-dump-from-a-jre-only-on-windows
18. http://stackoverflow.com/questions/32605962/locate-and-read-objects-of-a-specific-type-from-memory-of-a-running-java-program
19. http://stackoverflow.com/questions/30629014/how-to-get-object-id-as-used-in-heap-dump
20. http://stackoverflow.com/questions/26502836/jmap-fforce-option-doesnt-work
21. http://stackoverflow.com/questions/25007427/how-are-exceptions-caught-and-dealt-with-at-the-low-assembly-level
22. http://stackoverflow.com/questions/23632653/generate-java-heap-dump-on-uncaught-exception
23. http://stackoverflow.com/questions/32198820/why-jstack-not-working-when-the-tmp-java-pidnum-socket-file-has-been-deleted
24. http://stackoverflow.com/questions/23247947/how-to-automatically-dump-memory-when-old-gen-of-jvm-exceeds-some-ratio

Exception traces

```
Exception in thread "main" java.net.UnknownHostException: nonexistent.com
    at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:184)
    at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:172)
    at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)
    at java.net.Socket.connect(Socket.java:589)
    at java.net.Socket.connect(Socket.java:538)
    at java.net.Socket.<init>(Socket.java:434)
    at java.net.Socket.<init>(Socket.java:211)
    at TestConnection.main(TestConnection.java:6)
```

Demo 1

No stack trace

- Implicit exceptions
  - NullPointerException
  - ArrayIndexOutOfBoundsException
  - ArithmeticException
  - ArrayStoreException
  - ClassCastException

- -XX:-OmitStackTraceInFastThrow

Exception traces

```
@Benchmark
public Object date() {
    return new Date();
}

@Benchmark
public Object exception() {
    return new Exception();
}
```

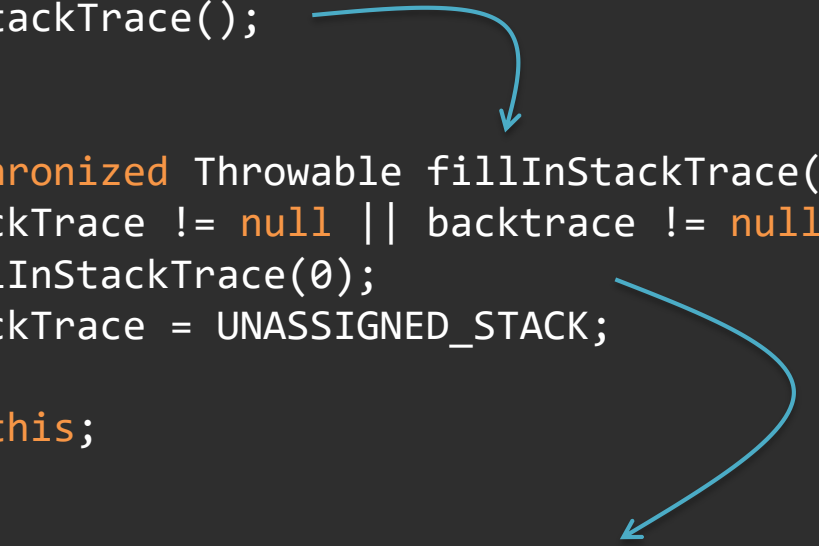
```
Benchmark              Mode  Cnt     Score    Error  Units
Exception.date         avgt   10     8,457 ±  0,040  ns/op
Exception.exception    avgt   10  1208,841 ± 15,569  ns/op
```

Exception traces

```
public Throwable() {
    fillInStackTrace();
}

public synchronized Throwable fillInStackTrace() {
    if (stackTrace != null || backtrace != null) {
        fillInStackTrace(0);
        stackTrace = UNASSIGNED_STACK;
    }
    return this;
}

private native Throwable fillInStackTrace(int dummy);
```



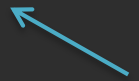
Exception traces

```
@Benchmark
public Object exceptionNoStackTrace() {
    return new Exception() {
        @Override
        public Throwable fillInStackTrace() {
            return this;
        }
    };
}
```

Benchmark	Mode	Cnt	Score	Error	Units
Exception.date	avgt	10	8,457	± 0,040	ns/op
Exception.exceptionNoStackTrace	avgt	10	7,839	± 0,034	ns/op

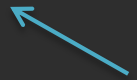
Exception traces

```
protected Exception(String message, Throwable cause,
                    boolean enableSuppression,
                    boolean writableStackTrace) {
```



```
public class LightException extends Exception {

    public LightException(String message, Throwable cause) {
        super(message, cause, true, false);
    }
}
```



Exception traces

```
@Param({"1", "2", "10", "100", "1000"})
int depth;

@Benchmark
public Object throwCatch() {
    try {
        return recursive(depth);
    } catch (Exception e) {
        return e;
    }
}

private Object recursive(int depth) throws Exception {
    if (depth == 0) {
        throw new LightException();
    }
    return recursive(depth - 1);
}
```

Exception traces

```
@Param({"1", "2", "10", "100", "1000"})
int depth;

@Benchmark
public Object throwCatch() {
    try {
        return recursive(depth);
    } catch (Exception e) {
        return e;
    }
}
```

Benchmark	(depth)	Mode	Cnt	Score		Error	Units
Exceptions.throwCatch	1	avgt	10	8,020	±	0,062	ns/op
Exceptions.throwCatch	2	avgt	10	126,724	±	0,697	ns/op
Exceptions.throwCatch	10	avgt	10	620,068	±	3,354	ns/op
Exceptions.throwCatch	100	avgt	10	5750,440	±	34,838	ns/op
Exceptions.throwCatch	1000	avgt	10	58465,378	±	251,693	ns/op

Exception traces

```
@Benchmark
public Object fillInStackTrace() {
    return new Exception();
}

@Benchmark
public Object getStackTrace() {
    return new Exception().getStackTrace();
}
```

Benchmark	Mode	Cnt	Score	Error	Units
Exceptions.fillInStackTrace	avgt	10	1207,092	± 10,150	ns/op
Exceptions.getStackTrace	avgt	10	13388,665	± 54,684	ns/op

Exception traces

```
int depth = getStackTraceDepth();
stackTrace = new StackTraceElement[depth];
for (int i = 0; i < depth; i++)
    stackTrace[i] = getStackTraceElement(i);
```


```
public final class StackTraceElement {
    private String declaringClass;
    private String methodName;
    private String fileName;
    private int    lineNumber;
    ...
}
```

Exception traces

```
public final class StackTraceElement {
    private String moduleName;
    private String moduleVersion;
    private String declaringClass;
    private String methodName;
    private String fileName;
    private int    lineNumber;
```

new in JDK 9

```
Exception in thread "main" java.net.UnknownHostException: nonexistent.com
    at java.net.PlainSocketImpl.connect(java.base@9-ea/PlainSocketImpl.java:172)
    at java.net.SocksSocketImpl.connect(java.base@9-ea/SocksSocketImpl.java:402)
    at java.net.Socket.connect(java.base@9-ea/Socket.java:591)
    at java.net.Socket.connect(java.base@9-ea/Socket.java:540)
    at java.net.Socket.<init>(java.base@9-ea/Socket.java:436)
    at java.net.Socket.<init>(java.base@9-ea/Socket.java:213)
    at TestConnection.main(TestConnection.java:6)
```

Exception performance

- new Exception() is cheap
  - fillInStackTrace() costs!
  - getStackTrace() costs even more!
  - *O(stack_depth)*
- throw is cheap
  - catch ≈ goto in the same frame
  - *O(stack_depth)*

Tips

- Use -XX:-OmitStackTraceInFastThrow
- No exceptions in normal control flow
- But if you really want
  - use no-stack-trace constructor

When do you need stack traces?

Use cases

```
"Thread-1" #18 prio=5 os_prio=0 tid=0x1923a800 nid=0xdf8 waiting on condition [0x19faf000]
   java.lang.Thread.State: WAITING (parking)
     at sun.misc.Unsafe.park(Native Method)
     - parking to wait for <0xd94cec48> (a java.util.concurrent.locks.ReentrantLock$NonfairSync)
     at java.util.concurrent.locks.ReentrantLock.lock(ReentrantLock.java:285)
     at RpcTest.clientLoop(RpcTest.java:34)
     at java.lang.Thread.run(Thread.java:745)

"Thread-2" #19 prio=5 os_prio=0 tid=0x19243800 nid=0x3d40 runnable [0x1a0af000]
   java.lang.Thread.State: RUNNABLE
     at one.nio.net.NativeSocket.readFully(Native Method)
     at one.nio.rpc.RpcClient.invoke(RpcClient.java:76)
     at RpcTest.clientLoop(RpcTest.java:36)
     at java.lang.Thread.run(Thread.java:745)
```

Use cases

```
"Thread-1" #18 prio=5 os_prio=0 tid=0x1923a800 nid=0xdf8 waiting on condition [0x19faf000]
   java.lang.Thread.State: WAITING (parking)
     at sun.misc.Unsafe.park(Native Method)
     - parking to wait for <0xd94cec48> (a java.util.concurrent.locks.ReentrantLock$NonfairSync)
     at java.util.concurrent.locks.ReentrantLock.lock(ReentrantLock.java:285)
     at RpcTest.clientLoop(RpcTest.java:34)
     at java.lang.Thread.run(Thread.java:745)

   Locked ownable synchronizers:
     - None

"Thread-2" #19 prio=5 os_prio=0 tid=0x19243800 nid=0x3d40 runnable [0x1a0af000]
   java.lang.Thread.State: RUNNABLE
     at one.nio.net.NativeSocket.readFully(Native Method)
     at one.nio.rpc.RpcClient.invoke(RpcClient.java:76)
     at RpcTest.clientLoop(RpcTest.java:36)
     at java.lang.Thread.run(Thread.java:745)

   Locked ownable synchronizers:
     - <0xd94cec48> (a java.util.concurrent.locks.ReentrantLock$NonfairSync)
```

-XX:+PrintConcurrentLocks

Logging

```
logger.warning("Operation timed out");
```

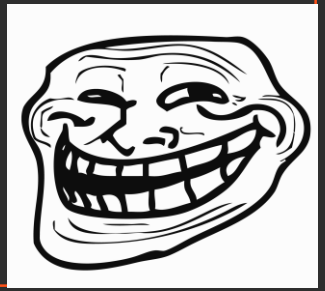
```
[Controller.java:123] WARNING: Operation timed out
```

```
public static String getLocation() {
    StackTraceElement s = new Exception().getStackTrace()[2];
    return s.getFileName() + ':' + s.getLineNumber();
}
```

A better way?

```
Thread.current().getStackTrace()
```

```
public StackTraceElement[] getStackTrace() {
    if (this != Thread.currentThread()) {
        // ... some magic ...
    } else {
        // Don't need JVM help for current thread
        return (new Exception()).getStackTrace();
    }
}
```



A better way!

```
public static String getLocation() {
    StackTraceElement s = sun.misc.SharedSecrets.getJavaLangAccess()
            .getStackTraceElement(new Exception(), 2);

    return s.getFileName() + ':' + s.getLineNumber();
}
```

Not in JDK 9

Use cases

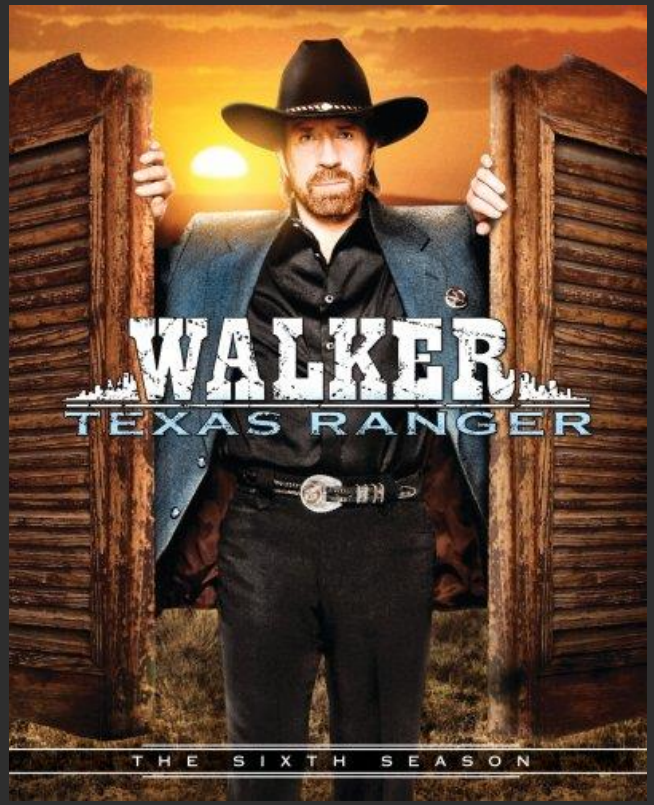
Caller sensitive methods

- @CallerSensitive
  - Class.forName
  - ResourceBundle.getBundle
  - System.loadLibrary
  - AccessController.doPrivileged
- sun.reflect.Reflection.getCallerClass

Stack Walking API

- Limit depth
- Filter frames
- Lazy construction
- Class instances instead of Class names

java.lang.StackWalker



java.lang.StackWalker

- StackWalker.getInstance()
- StackWalker.getInstance(Set<Option> options)
  - RETAIN_CLASS_REFERENCE
  - SHOW_REFLECT_FRAMES
  - SHOW_HIDDEN_FRAMES
- StackWalker.getInstance(Set<Option> options,
  int estimateDepth)

Stack Walking API

```
StackWalker sw = StackWalker.getInstance();
sw.forEach(System.out::println);
```

```
StackWalking.dumpStackTrace(StackWalking.java:7)
StackWalking.main(StackWalking.java:12)
com.intellij.rt.execution.application.AppMain.main(AppMain.java:144)
```

Stack Walking API

```
StackWalker sw = StackWalker.getInstance(StackWalker.Option.SHOW_REFLECT_FRAMES);
sw.forEach(System.out::println);
```

```
StackWalking.dumpStackTrace(StackWalking.java:7)
StackWalking.main(StackWalking.java:12)
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
java.lang.reflect.Method.invoke(Method.java:531)
com.intellij.rt.execution.application.AppMain.main(AppMain.java:144)
```

Stack Walking API

```
StackWalker sw = StackWalker.getInstance(StackWalker.Option.SHOW_HIDDEN_FRAMES);
sw.forEach(System.out::println);
```

```
StackWalking.dumpStackTrace(StackWalking.java:7)
StackWalking$$Lambda$1/1287712235.run(Unknown Source bci:0)
StackWalking.main(StackWalking.java:12)
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
java.lang.reflect.Method.invoke(Method.java:531)
com.intellij.rt.execution.application.AppMain.main(AppMain.java:144)
```

Stack Walking API

```
public <T> T walk(Function<? super Stream<StackFrame>, ? extends T> function)
```

```
public static StackFrame getCallerFrame() {
    return StackWalker.getInstance()
            .walk(stream -> stream.skip(2).findFirst())
            .orElseThrow(NoSuchElementException::new);
}
```

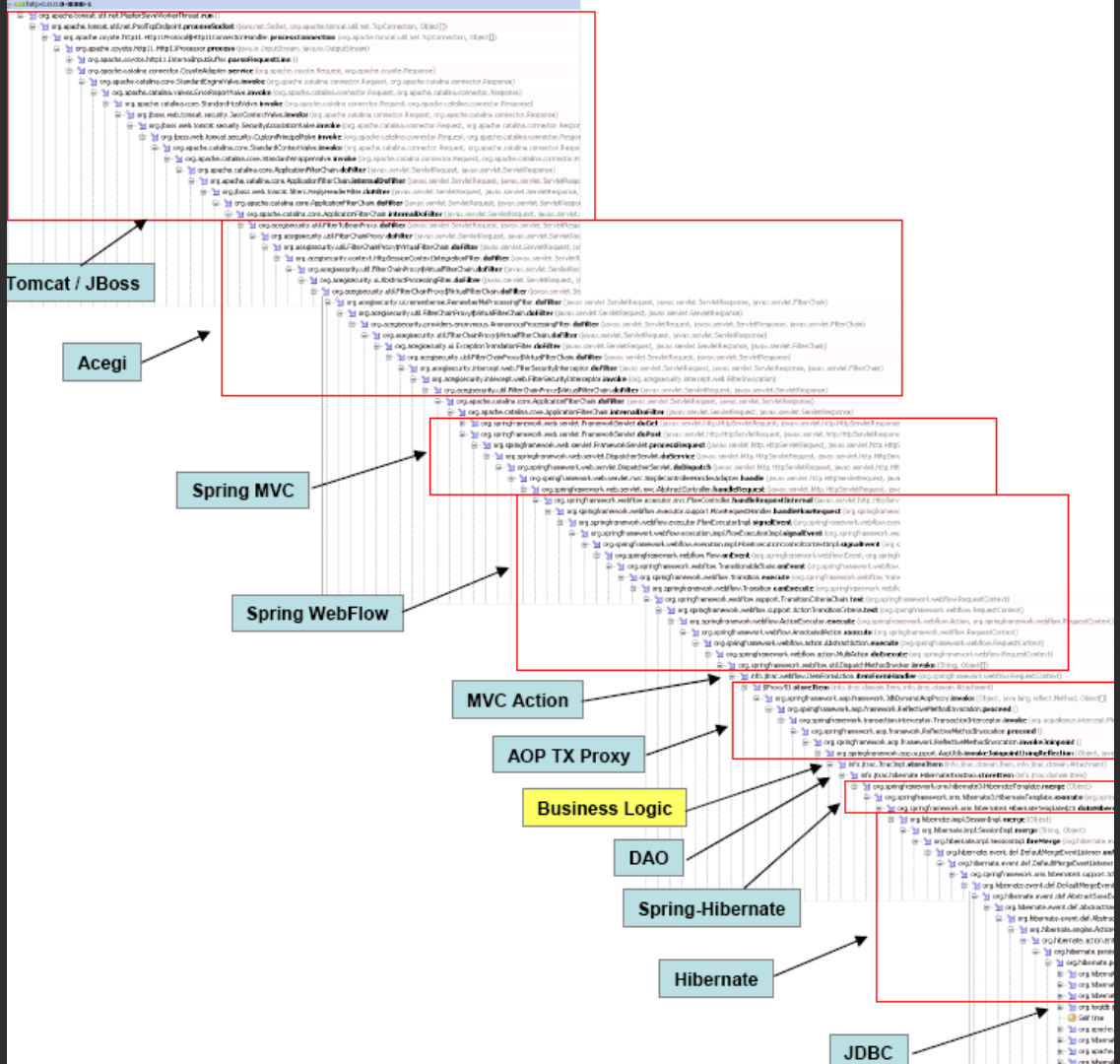
```
return StackWalker.getInstance(RETAIN_CLASS_REFERENCE).getCallerClass();
```

Stack Walking API

```
StackWalker sw = StackWalker.getInstance();

List<StackFrame> frames = sw.walk(stream ->
        stream.filter(sf -> sf.getClassName().startsWith("com.jpoint."))
              .limit(10)
              .collect(Collectors.toList()));
```

```
public Stream<StackFrame> stream();
```


Tomcat / JBoss
Acegi
Spring MVC
Spring WebFlow
MVC Action
AOP TX Proxy
Business Logic
DAO
Spring-Hibernate
Hibernate
JDBC

Fun with recursion

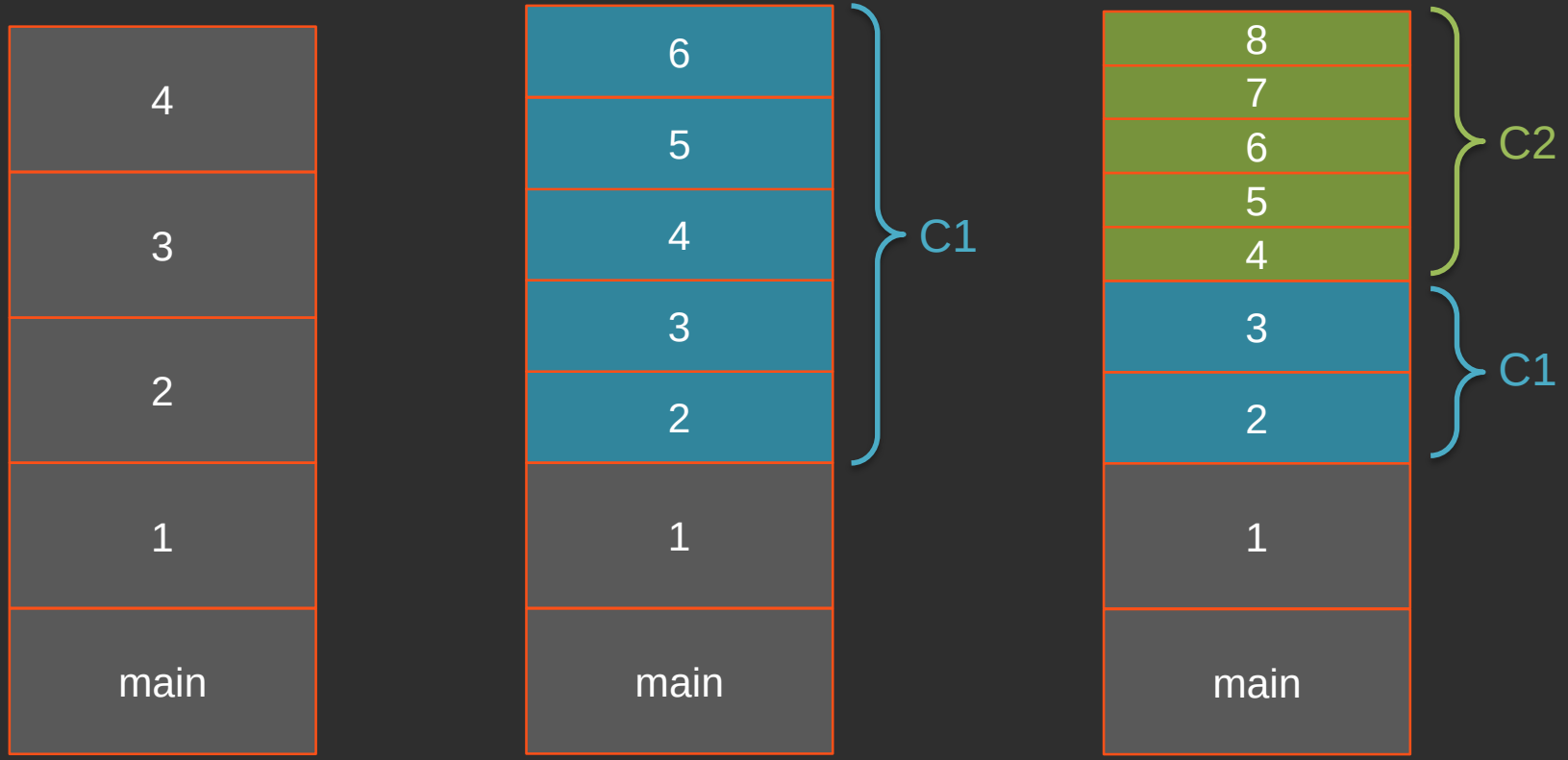
```
static int depth;

static void recursion() {
    depth++;
    recursion();
}

public static void main(String[] args) {
    recursion();
}
```

Demo 2

Stack size

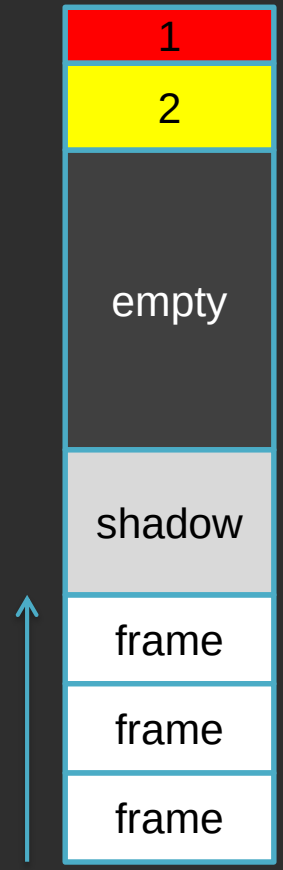


Stack size

- -Xss, -XX:ThreadStackSize
- Thread(threadGroup, target, name, stackSize)
- java.lang.OutOfMemoryError: Unable to create new native thread

Stack size

- Default stack size?
  - x86: 320 KB
  - x64: 1MB
- Minimum stack size?
  - ~228 KB



Profiling

Types of profiling

- Instrumentation

```
public void someMethod(String... args) {
    Profiler.onMethodEnter("myClass.someMethod");
    // method body
    Profiler.onMethodExit("myClass.someMethod");
}
```

- Sampling
  - Take stack traces every 10 – 100 ms
  - Suitable for production

Demo 3

Sampling bias

- Stack traces are taken at VM safepoint
- Running and idle threads are sampled equally
- Blocking native calls appear as Runnable
- Victims: VisualVM, JProfiler, YourKit…

Fair profiler?

- setitimer + SIGPROF
- SIGPROF handler collects stack trace of signaled thread
- HotSpot internal API: AsyncGetCallTrace
  - Used in Oracle Solaris Studio

AsyncGetCallTrace

```
ASGCT_CallFrame frames[MAX_FRAMES];
ASGCT_CallTrace trace = {env, MAX_FRAMES, frames};

AsyncGetCallTrace(&trace, trace.num_frames, ucontext);
```

from signal handler

https://github.com/apangin/async-profiler

How to dump?

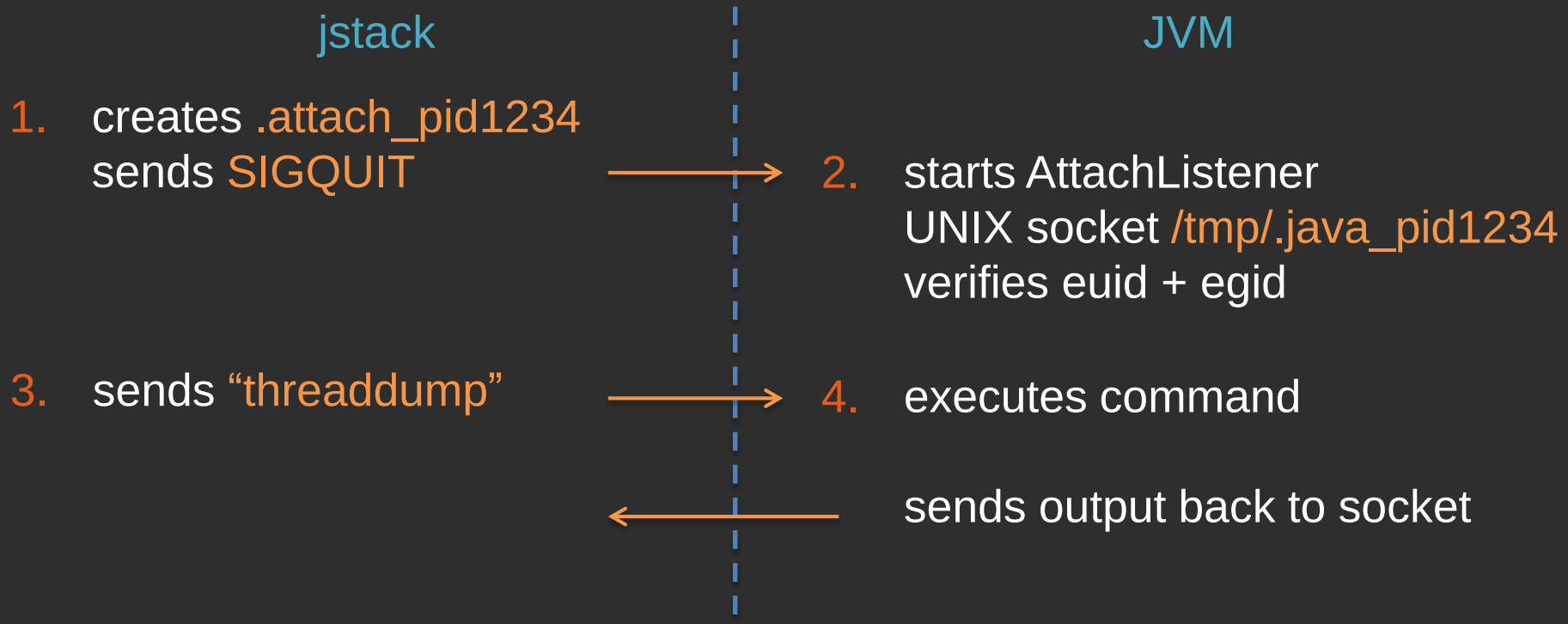
In-process

- Java
  - Thread.getAllStackTraces → StackTraceElement[]
  - ThreadMXBean.getThreadInfo
- JVMTI
  - GetAllStackTraces
  - Compact representation (jmethodID)
  - Used by profilers

External

- SIGQUIT (kill -3, Ctrl + \)
  - Processed by JVM at max. speed
  - No intermediate structures
  - Prints to stdout
- jstack
  - Dynamic attach
- jstack -F
  - Serviceability Agent

Dynamic attach



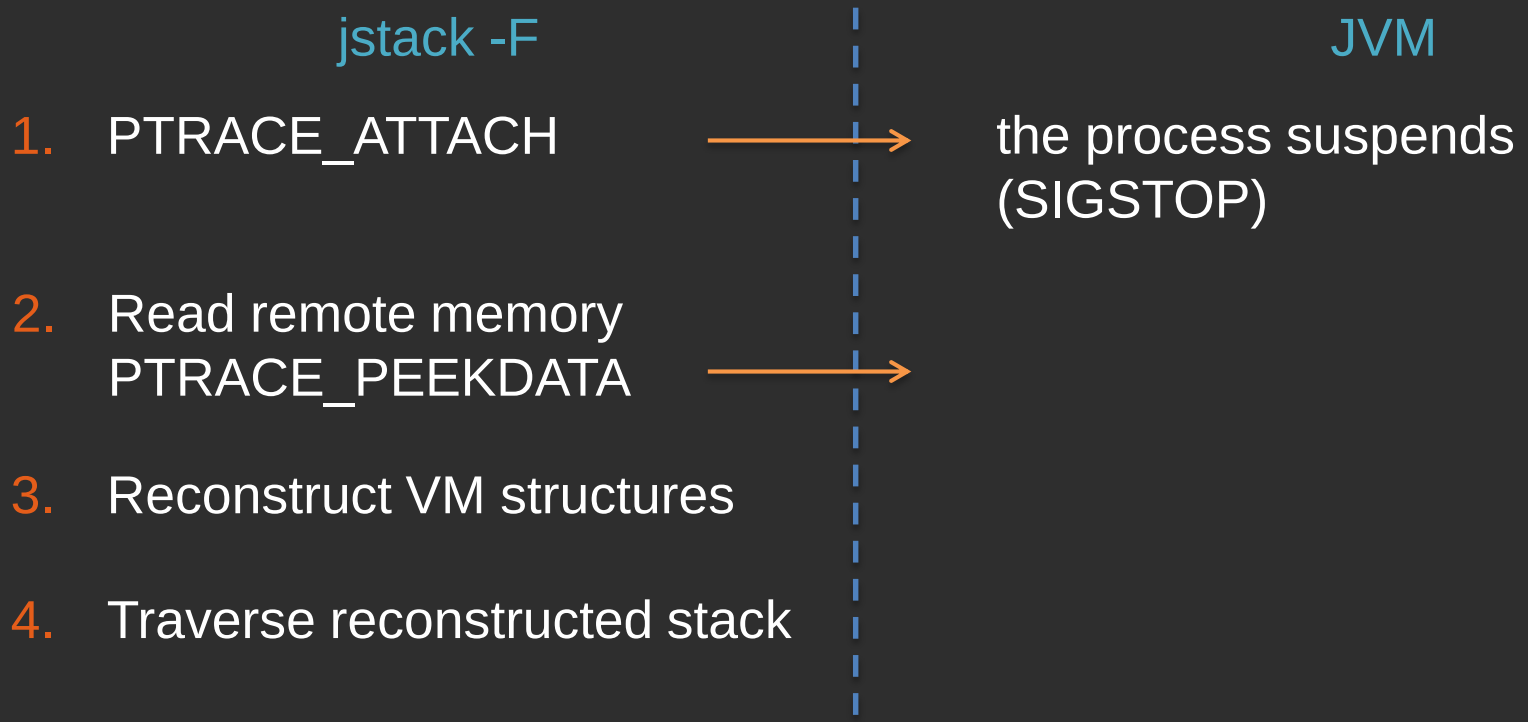
jstack

- Pros.
  - Performed by JVM at max. speed
  - Compatibility with different JVM versions

- Cons.
  - Should be run by the same user (euid / egid)
  - Works only on healthy JVM
  - Can be disabled (-XX:+DisableAttachMechanism)

Thread dump

Serviceability Agent



jstack -F

- Pros.
  - No cooperation from JVM required
  - root can dump any process
  - Mixed Java+Natives frames (-m)

- Cons.
  - Slow
  - Requires the same version of JVM
  - No thread names, no lock info

How to dump?

```
public static void dump() throws AttachNotSupportedException, IOException {
    String vmName = ManagementFactory.getRuntimeMXBean().getName();
    String pid = vmName.substring(0, vmName.indexOf('@'));

    HotSpotVirtualMachine vm = (HotSpotVirtualMachine) VirtualMachine.attach(pid);
    try {
        vm.localDataDump();
    } finally {
        vm.detach();
    }
}
```

https://github.com/odnoklassniki/one-nio/blob/master/src/one/nio/mgt/

Heap dump

jmap

- jmap -dump:live,format=b,file=heap.bin PID
- jmap -histo PID

```
 num     #instances         #bytes  class name
----------------------------------------------
   1:          1943      404550984  [J
   2:          5622       41517944  [B
   3:        306128       33720768  [C
   4:        292548        7021152  java.lang.String
   5:        131072        4194304  one.nio.lock.RWLock
   6:        103634        3316288  java.util.HashMap$Node
   7:         89096        2138304  java.lang.Long
   8:         17503        1850168  [Ljava.lang.Object;
   9:         50688        1622016  [Lone.cassandra12.remoting.Host;
```

jmap vs. jmap -F

jmap

+ Speed
+ Version tolerance
+ Supports “live” option

- Requires the same user
- Works only on healthy JVM

jmap -F

+ Can be used on hung process
+ Can be used on core dump
+ root can dump everything

- Slow
- Requires the same JDK
- Heap may be inconsistent

Offline heap dumps

```
$ sudo gcore 1234

$ jmap -dump:format=b,file=heap.bin /path/to/java core.1234
```

Windows: WinDbg.exe

```
jmap -dump:format=b,file=heap.bin C:\path\to\java.exe core.mdmp
```

Automatic heap dumps

- -XX:+HeapDumpOnOutOfMemoryError

- -XX:+HeapDumpBeforeFullGC
- -XX:+HeapDumpAfterFullGC

- -XX:HeapDumpPath=/tmp (or /tmp/dump.bin)

Manageable options

```
$ jinfo -flag +HeapDumpOnOutOfMemoryError 1234
$ jinfo -flag HeapDumpPath=/tmp 1234
```

```
HotSpotDiagnosticMXBean bean = ManagementFactory.newPlatformMXBeanProxy(
            ManagementFactory.getPlatformMBeanServer(),
            "com.sun.management:type=HotSpotDiagnostic",
            HotSpotDiagnosticMXBean.class);

bean.setVMOption("HeapDumpOnOutOfMemoryError", "true");
bean.setVMOption("HeapDumpPath", fileName);
```

Kill JVM after heap dump

- -XX:+ExitOnOutOfMemoryError
- -XX:+CrashOnOutOfMemoryError


```
$ java -XX:+HeapDumpOnOutOfMemoryError
       -XX:OnOutOfMemoryError="kill -9 %p"
       com.jpoint.TestApp
```

Dump heap programmatically

JMX

```
HotSpotDiagnosticMXBean bean = ManagementFactory.newPlatformMXBeanProxy(
            ManagementFactory.getPlatformMBeanServer(),
            "com.sun.management:type=HotSpotDiagnostic",
            HotSpotDiagnosticMXBean.class);

bean.dumpHeap("/tmp/heap.bin", true);
```

JVMTI

```
jvmtiIterationControl JNICALL
HeapObjectCallback(jlong class_tag, jlong size,
                   jlong* tag_ptr, void* user_data) {
    *tag_ptr = 1;
    return JVMTI_ITERATION_CONTINUE;
}
```

```
jvmti->IterateOverInstancesOfClass(targetClass, JVMTI_HEAP_OBJECT_EITHER,
                                   HeapObjectCallback, NULL);
```

```
jlong tag = 1;
jint count;
jobject* instances;
jvmti->GetObjectsWithTags(1, &tag, &count, &instances, NULL);
```

Serviceability Agent

```
SystemDictionary dictionary = VM.getVM().getSystemDictionary();
Klass klass = dictionary.find("java/lang/String", null, null);

VM.getVM().getObjectHeap().iterateObjectsOfKlass(new DefaultHeapVisitor() {
    public boolean doObj(Oop obj) {
        obj.printValue();
        System.out.println();
        return false;
    }
}, klass);
```

$JAVA_HOME/lib/sa-jdi.jar

Conclusions

- Use thread dumps and heap dumps
  - In PROD!

- In-process
  - Java (JMX), JVMTI
- Externally
  - Dynamic attach
  - Serviceability Agent

Thank you

@AndreiPangin