Troubleshooting with Serviceability and the New Runtime Monitoring Tool: HeapStats

JavaOne 2014 BOF3018

Yuji KUBOTA (@sugarlife) / Shinji TAKAO
Nippon Telegraph and Telephone Corp.

Questions and comments

#heapstats

Questions on twitter are appreciated!

We will response correctly than via hearing-loss :)

What will we talk about?

Troubleshooting of Java systems especially :

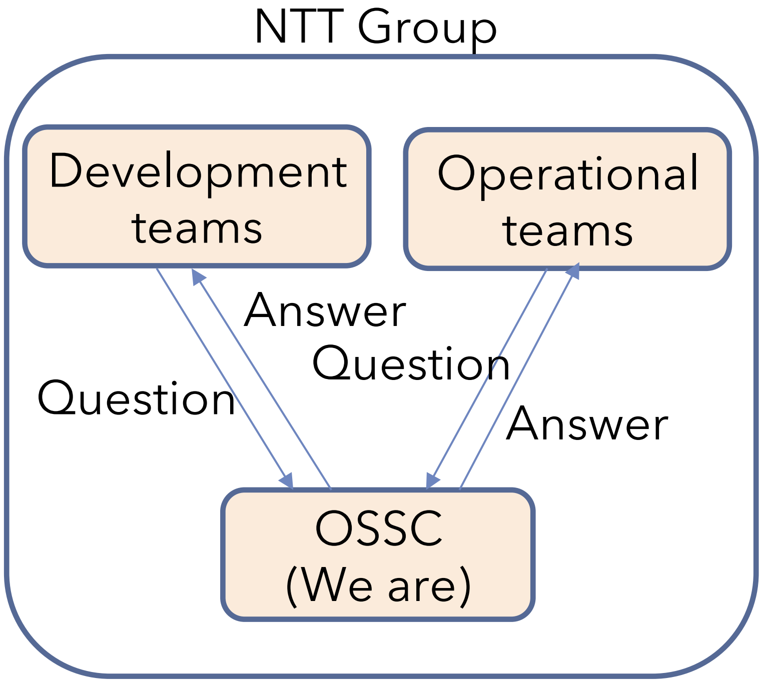
- in operating phase
- using free / open source tools for OracleJDK and OpenJDK on Linux for x86 (32/64)

Who are we?

- Yuji Kubota (hearing impaired, @sugarlife)
  Shinji Takao (me)
- IcedTea committers
- OpenJDK technical support engineers at Nippon Telegraph and Telephone Corp. (NTT) Open Source Software Center (OSSC) , Tokyo, Japan

Our history with OpenJDK

- The OSSC provides a technical advice for NTT group’s companies, handled 11,735 inquiries since 2006
- We handled **404** Java related inquiries including **259** Java system trouble shooting advice request since 2008

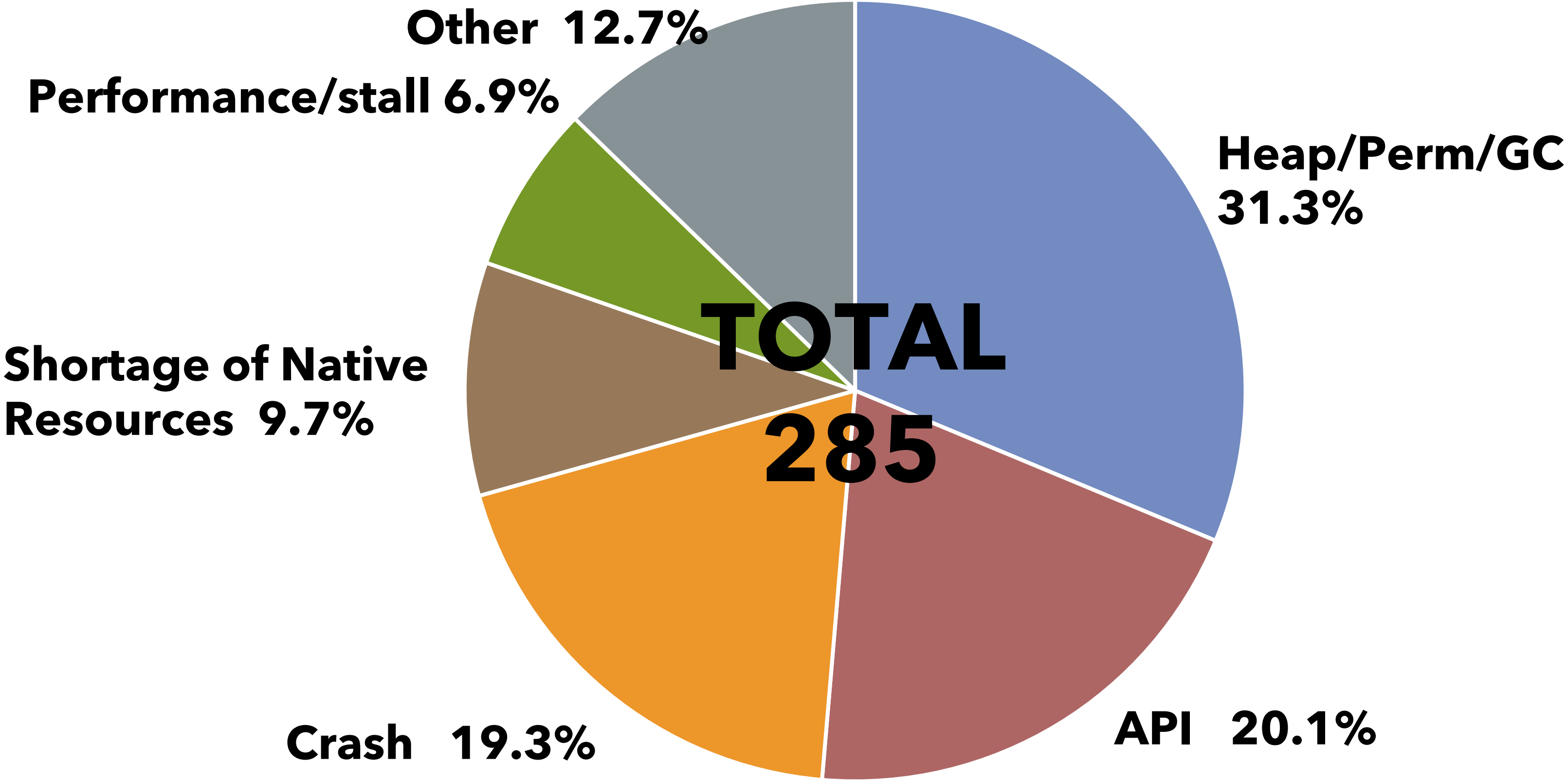


Topics

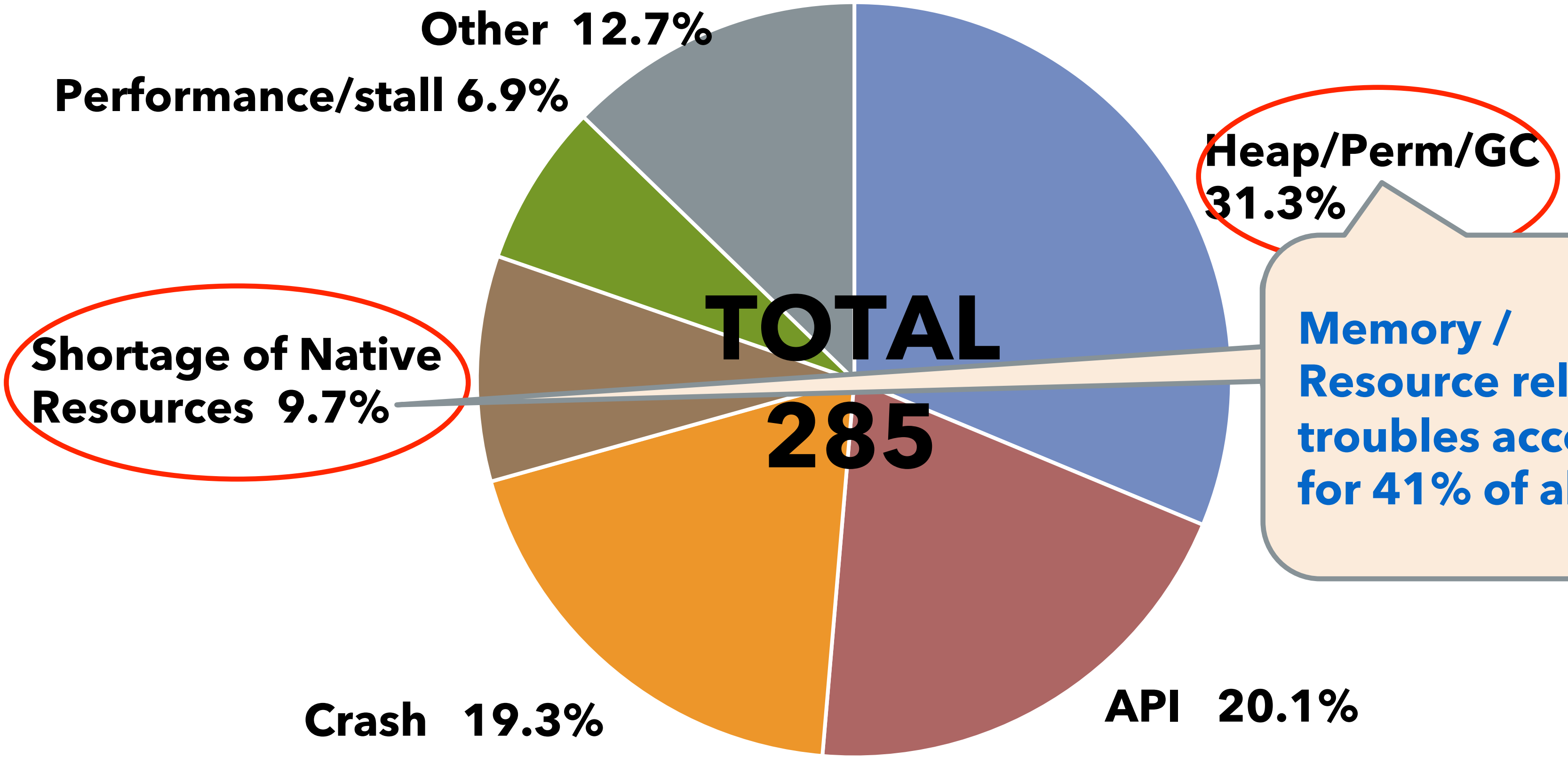
- Troubleshooting
  - Categorizing
  - Problems of current trouble shootings
- HeapStats
  - Design for "Good" troubleshooting tool
  - Demonstration with use-cases

Troubleshooting

Troubles we faced



Troubles we faced



Memory / Resource related troubles account for 41% of all

Case study : heap related

- Initial diagnosis
  - The amount of user access was not increased
  - GC worked fine and a sudden OOM had happened
- Set class histogram with OnOutOfMemory option
- Several months later, the OOME was reproduced

Case study : heap related

- Located the suspect class, but the operation team requested more concreate evidence
- Set HeapDumpOnOutOfMemory and waited another OOME
- Data elicited from the heapdump indicate that a database query for all of the data in a table at a time made the result eat up the heap

Shortcomings of the current way

- Collect Information : insufficient
- No suitable logging :
  Performance vs information-richness
- Presentation : inefficient
- Settings about logging : complicated

OK, so, what do I have to do?

128
HeapStats

HeapStats

- **Open source** Java tooling project in IcedTea community that aimed to better monitoring and **after-the-fact analysis**.
- This project provides a lightweight **JVMTI agent** which attaches HotSpot VM to collect clues, and a **GUI analyzer**.

What is “Good” troubleshooting tool?

#1 Collect required information.

#2 Lightweight.

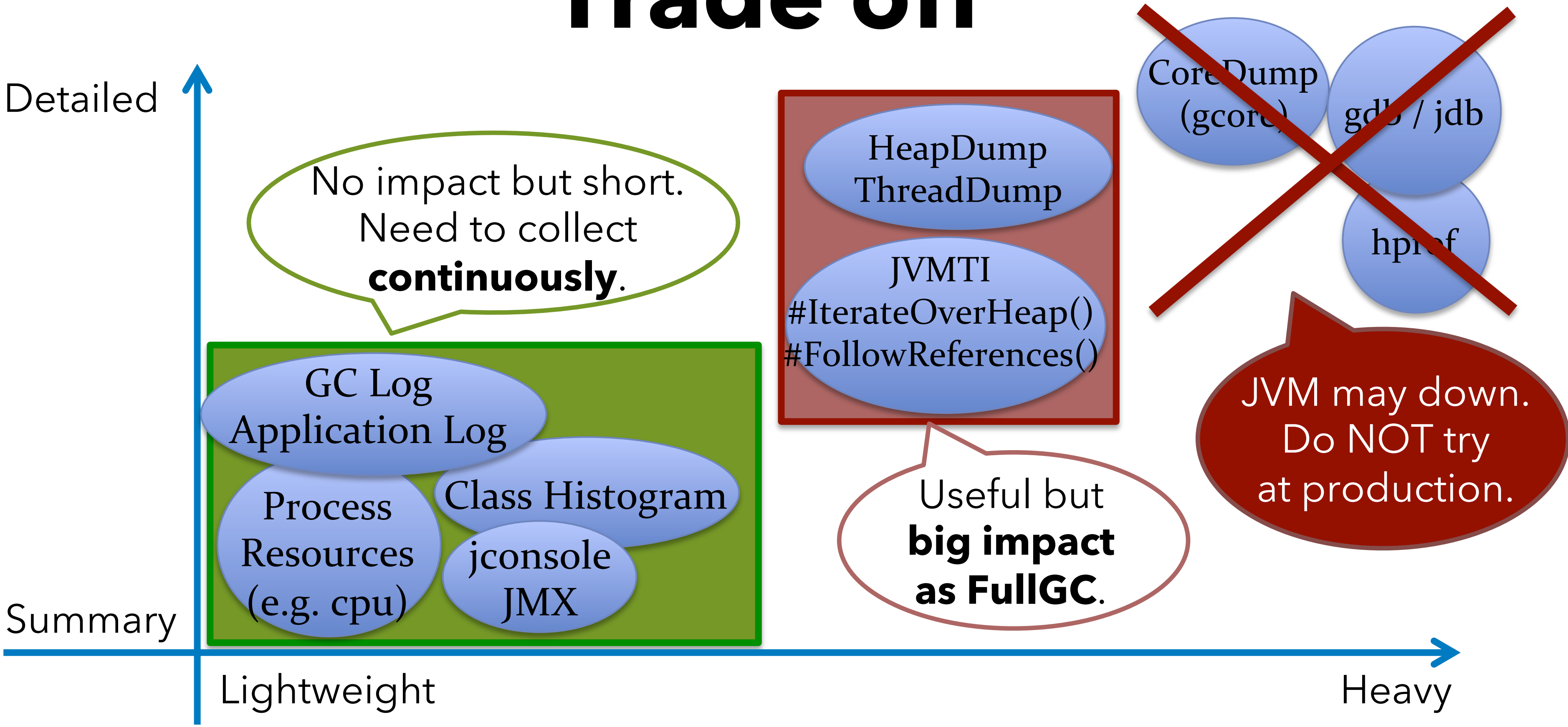
#3 Visualization.

#4 Ease of use.

#1 Collect required information.

Avoid the lack of information to troubleshoot even if a java expert is not assigned.

Trade off



Detailed

Summary

Lightweight

Heavy

GC Log
Application Log

Process
Resources
(e.g. cpu)

Class Histogram

jconsole
JMX

No impact but short.
Need to collect
continuously.

HeapDump
ThreadDump

JVMTI
#IterateOverHeap()
#FollowReferences()

Useful but
big impact
as FullGC.

CoreDump
(gcore)

gdb / jdb

hprof

JVM may down.
Do NOT try
at production.

Our strategy

- **Continuously collected resource data during JVM alives.**
  - Process resources (OS / JVM).
  - Heap and GC statistics.
- **Detailed dump with stop-the-world when JVM will die or face troubles.**
  - Heap snapshots, Thread dump
  - Other os log (syslog / journald, etc)

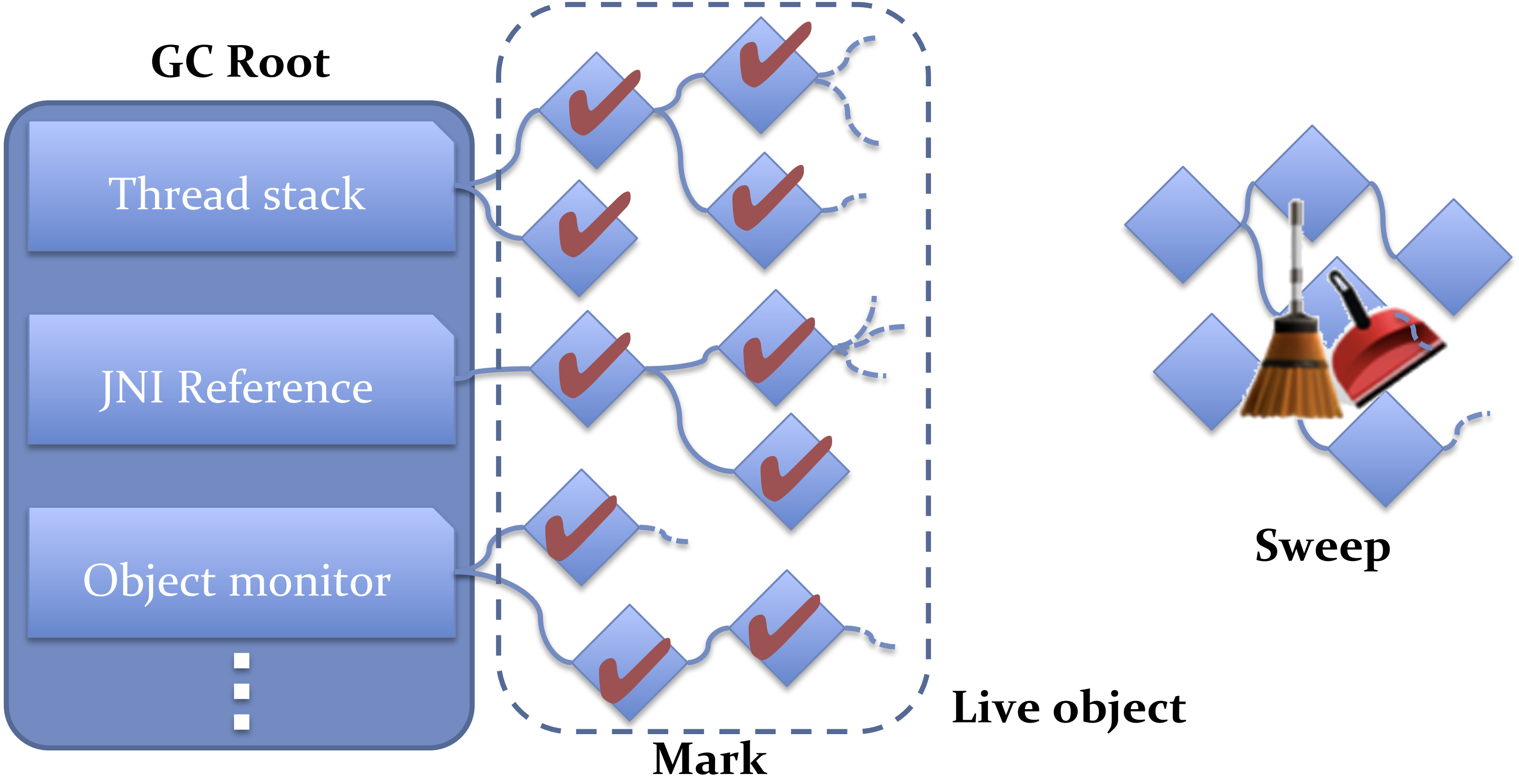
#2 Lightweight.

Do not stop your application.

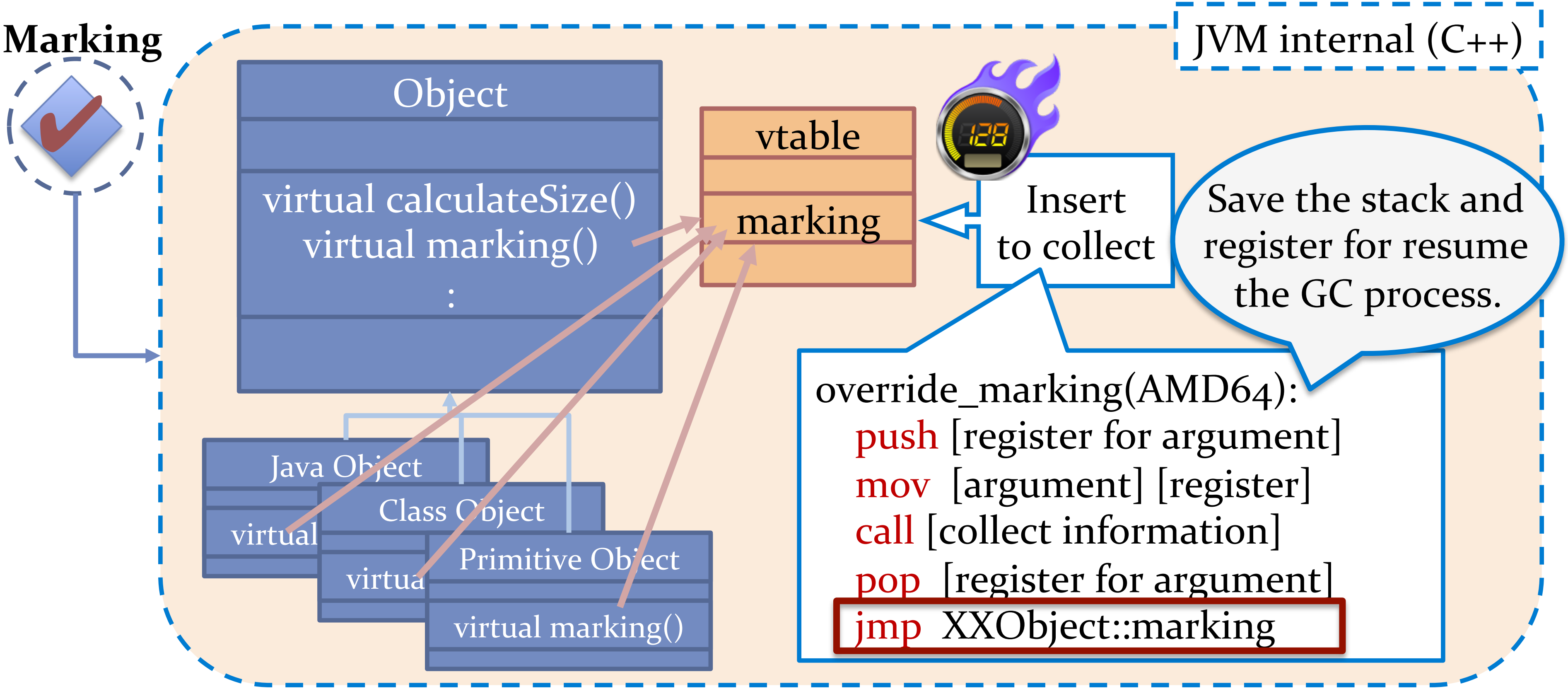
HeapStats collect data per Major GC.

- HeapStats agent hooks the marking or sweep phases of the garbage collector to collect java runtime information by oop (ordinary object pointer) **without extra stop the world**.

GC process of HotSpot VM



How to hook the marking



Save the stack and register for resume the GC process.

```
override_marking(AMD64):
   push [register for argument]
   mov  [argument] [register]
   call [collect information]
   pop  [register for argument]
   jmp  XXObject::marking
```

Additional optimization

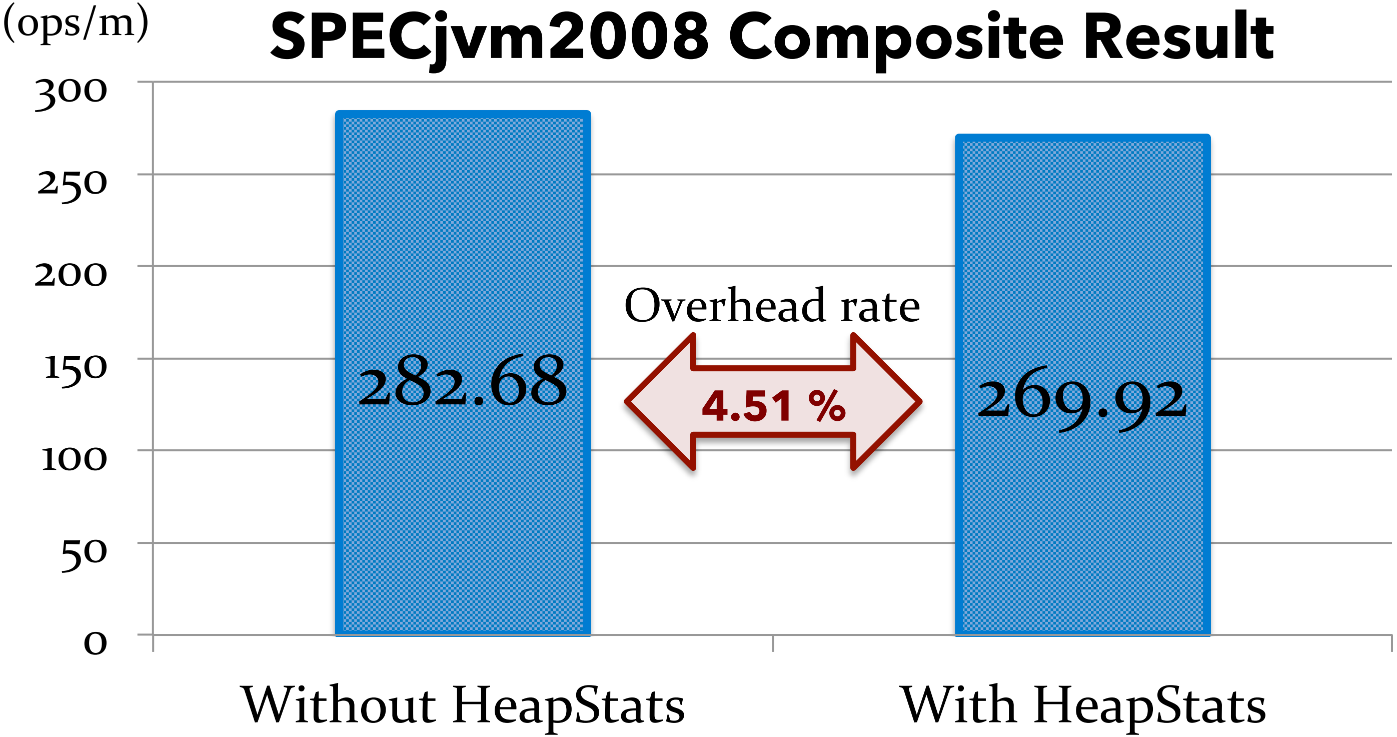
- Refactor the assembly of the override JVM inner functions to using SSE4 / AVX instruction sets.

Ex: Check vtable to hook

```
#ifdef AVX
 #define CHECK_VTABLE(oop_reg, valid_caller) \
    vmovd (oop_reg), %xmm0;            \
    vpshufd $0, %xmm0, %xmm1;          \
    mov valid_caller##@GOT(%ebx), %ecx; \
    vmovq (%ecx), %xmm1;               \
    vpcmpeqd %xmm0, %xmm1, %xmm1;      \
    vptest %xmm1, %xmm1;
```

```
#elif defined SSE4_1
 #define CHECK_VTABLE(oop_reg, valid_caller) \
    movd (oop_reg), %xmm0;             \
    pshufd $0, %xmm0, %xmm1;           \
    mov valid_caller##@GOT(%ebx), %ecx; \
    movq (%ecx), %xmm1;                \
    pcmpeqd %xmm0, %xmm1;              \
    ptest %xmm1, %xmm1;
```

Result



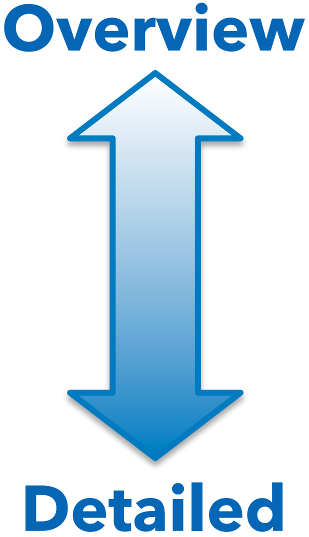
Measurement conditions
Benchmark tool: SPECjvm2008 1.01 Machine: DELL PowerEdge R810 (Xeon X7542, 32 GB memory)
OS: Red Hat Enterprise Linux Server 6.3 x86_64 Java: java-1.7.0-openjdk-1.7.0.25-2.3.10.4.el6_4.x86_64
Java execution options: -Xms4500m –Xmx4500m –XX:+UseG1GC –agentpath: <agent library>

#3 Visualization.

We do not need to hunt for treasure from noisy and huge log.

HeapStats analyzer

- HeapStats provides analyzer to show…

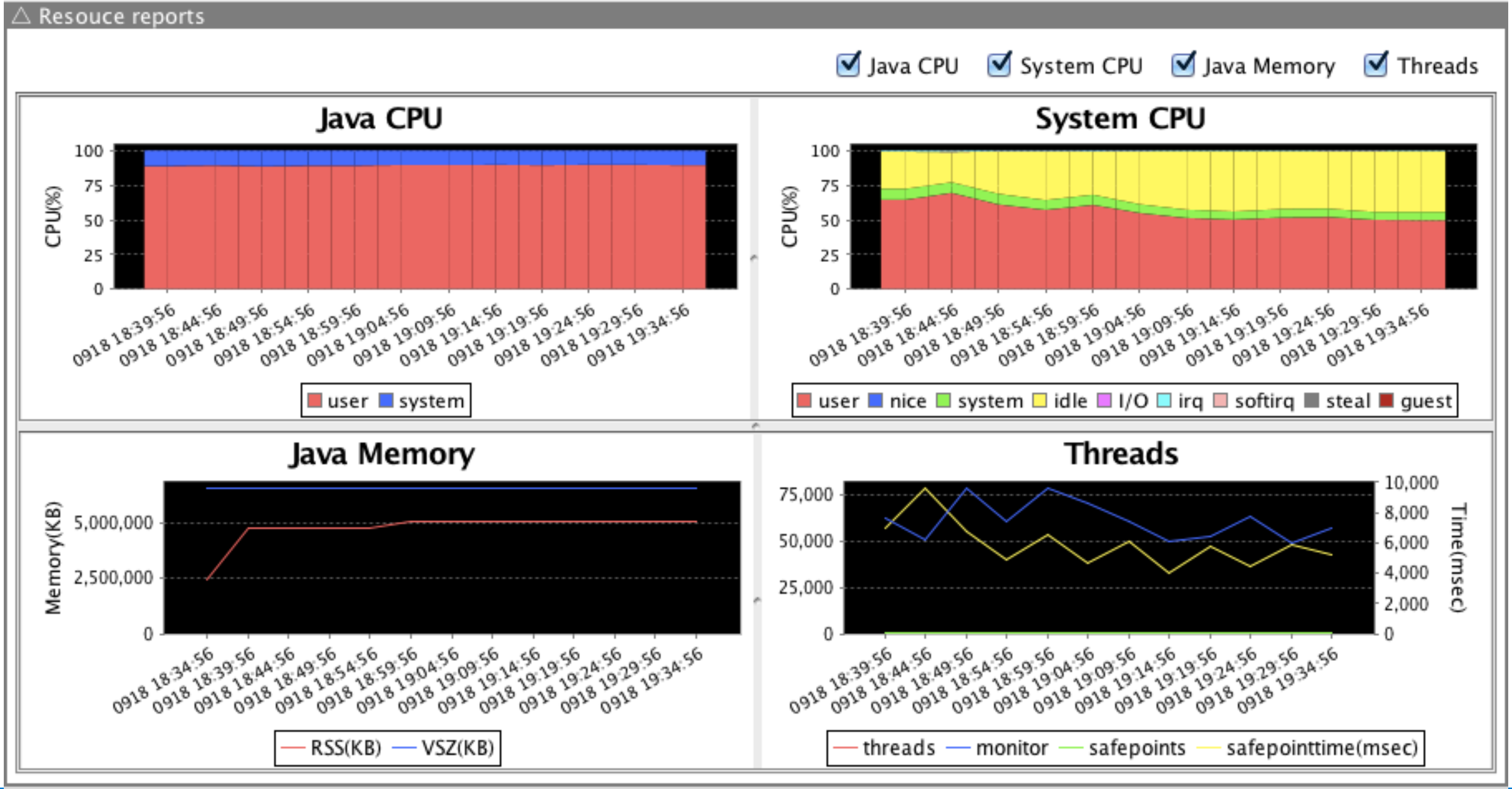


Overview

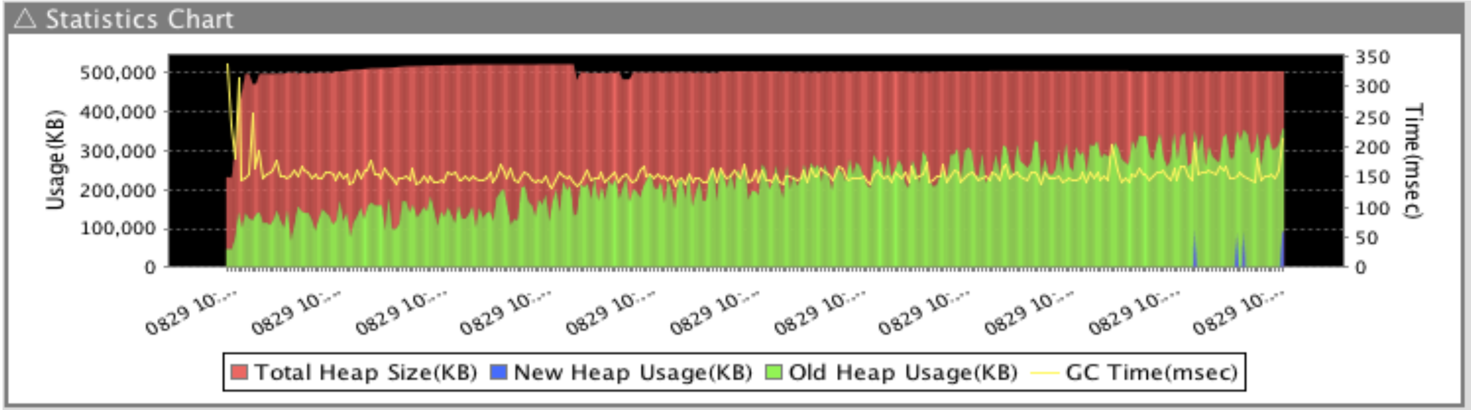
Detailed

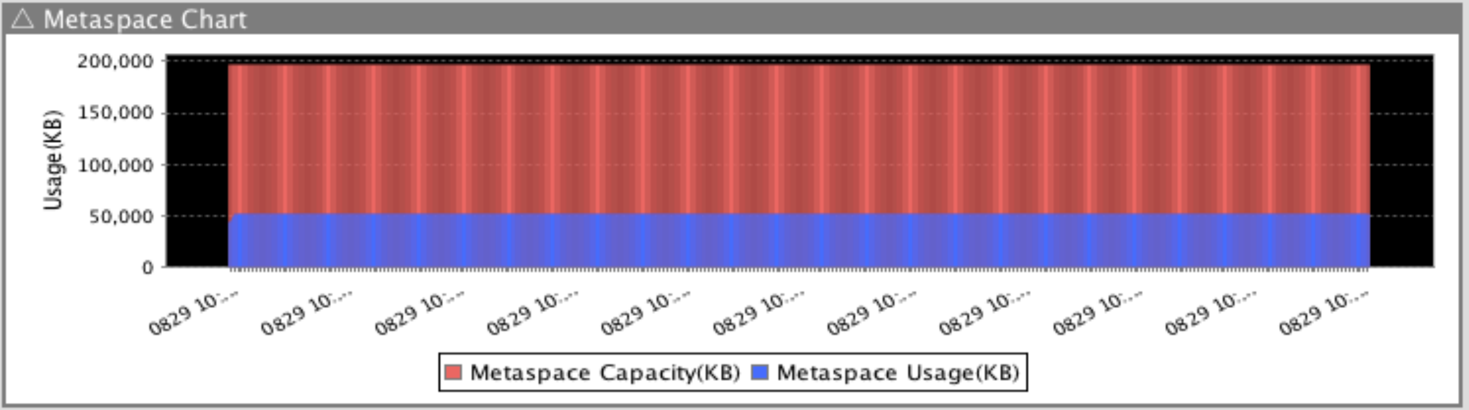
  - Process Resources (OS / JVM)
  - GC and Heap / Metaspace
  - Class Histogram
    - Timeline and Summary
  - Object reference

Process Resources (OS / JVM)

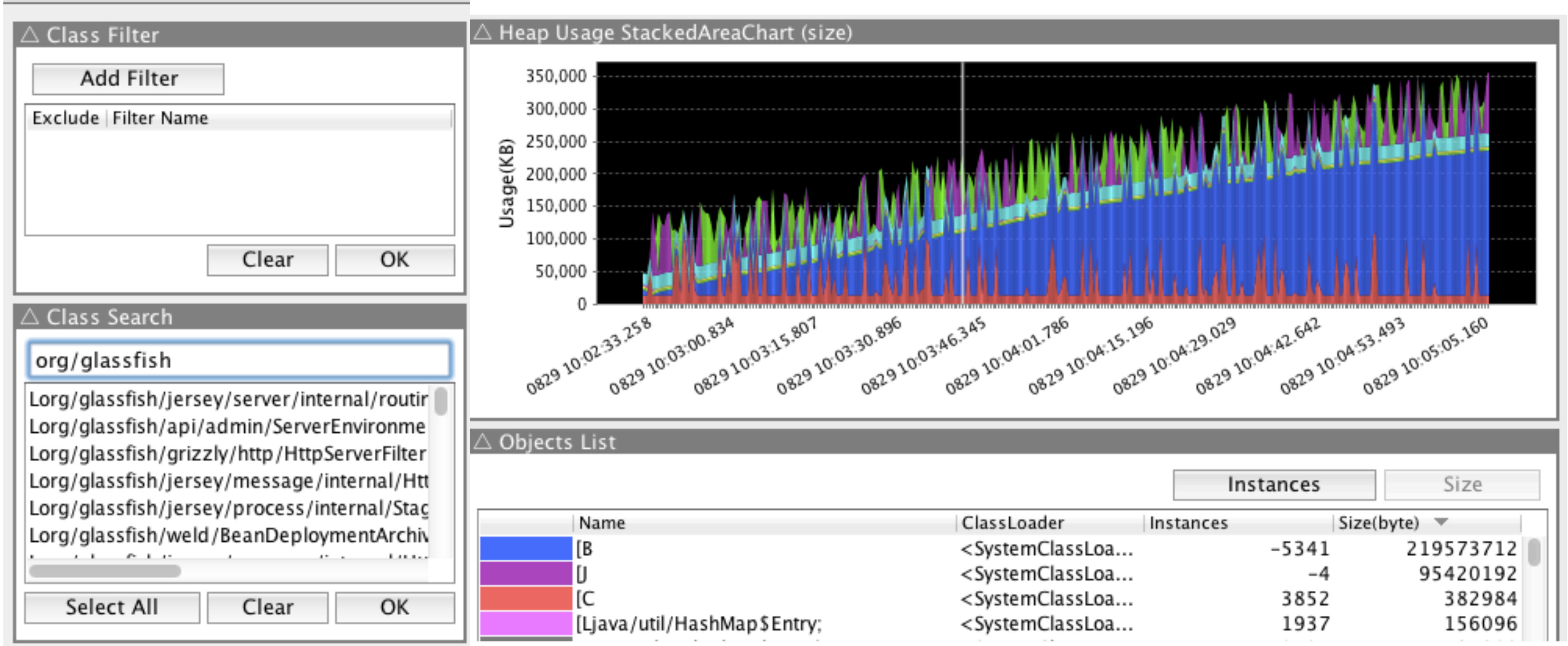


GC and Heap / Metaspace





Class Histogram (timeline)

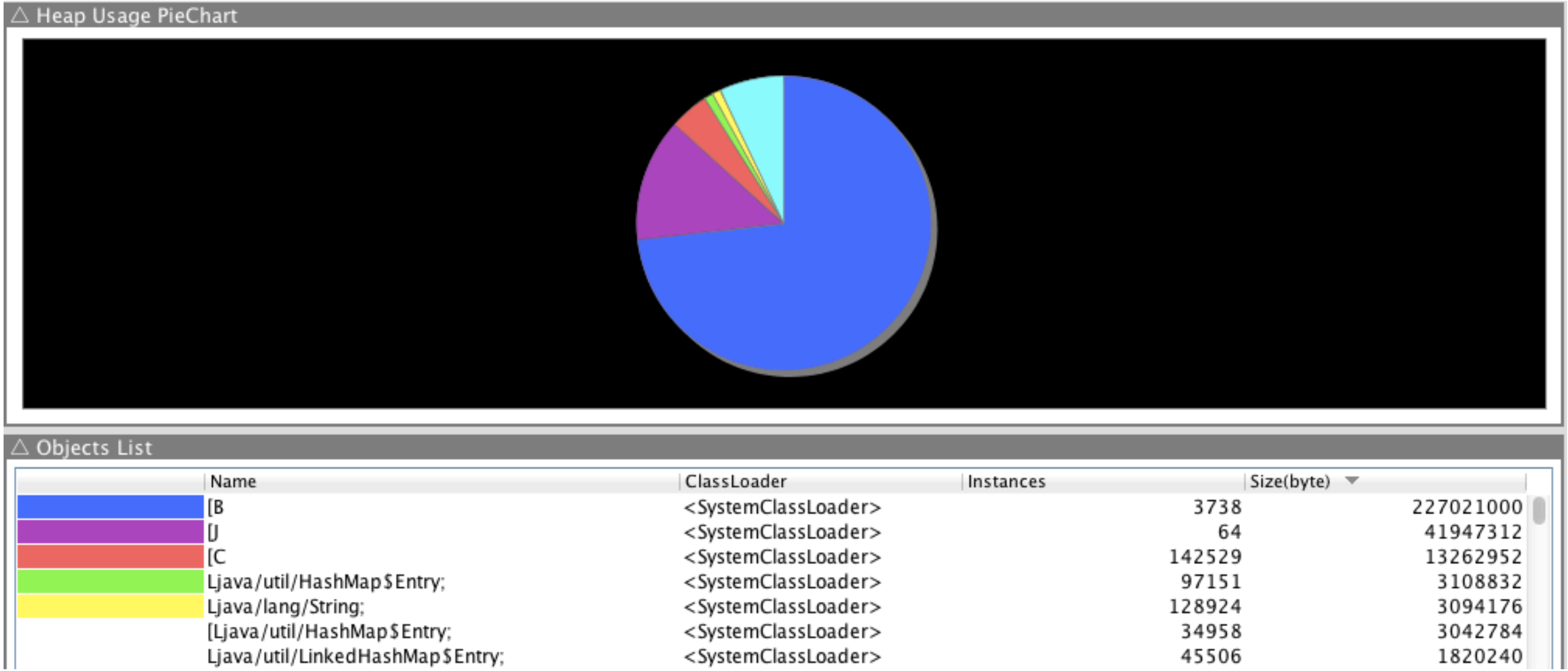


org/glassfish

Lorg/glassfish/jersey/server/internal/routir
Lorg/glassfish/api/admin/ServerEnvironme
Lorg/glassfish/grizzly/http/HttpServerFilter
Lorg/glassfish/jersey/message/internal/Htt
Lorg/glassfish/jersey/process/internal/Stag
Lorg/glassfish/weld/BeanDeploymentArchiv

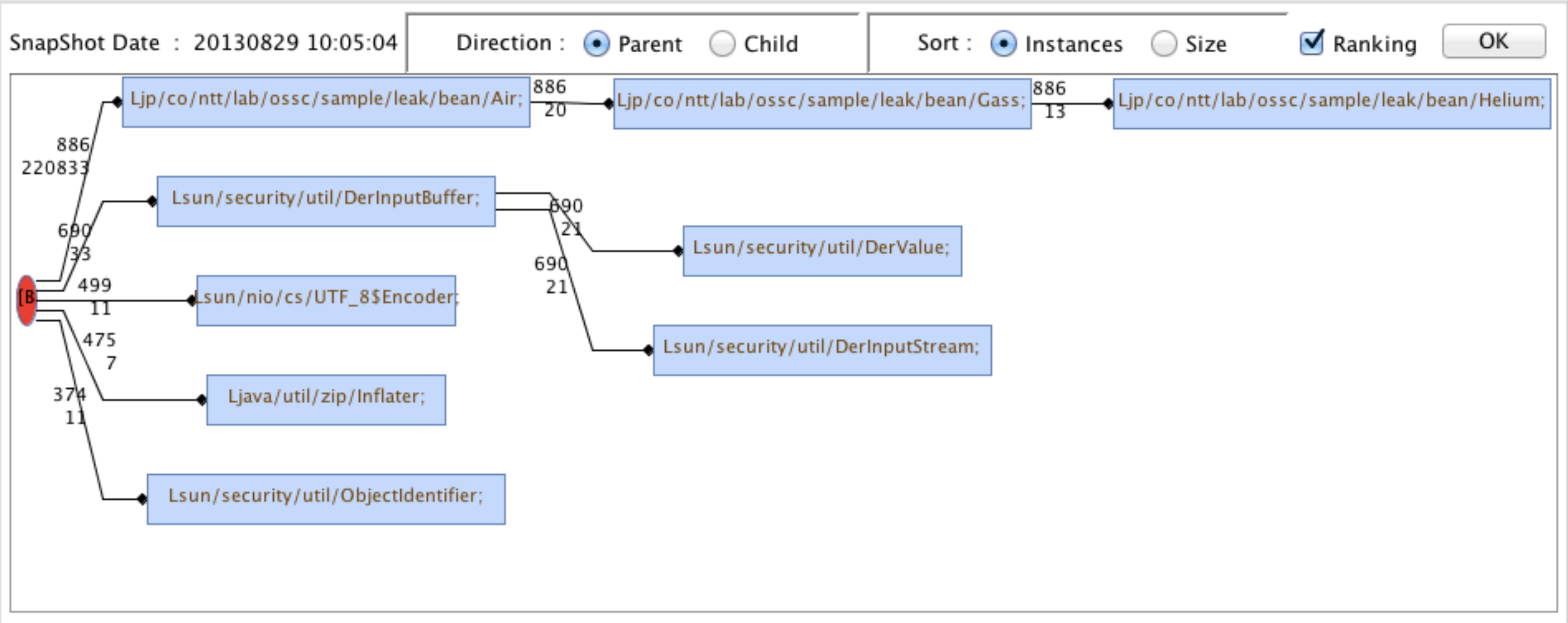
	Name	ClassLoader	Instances	Size(byte)
🟦	[B	<SystemClassLoa...	-5341	219573712
🟪	[J	<SystemClassLoa...	-4	95420192
🟥	[C	<SystemClassLoa...	3852	382984
🟪	[Ljava/util/HashMap$Entry;	<SystemClassLoa...	1937	156096

Class Histogram (summary)



	Name	ClassLoader	Instances	Size(byte) ▼
🟦	[B	<SystemClassLoader>	3738	227021000
🟪	[J	<SystemClassLoader>	64	41947312
🟥	[C	<SystemClassLoader>	142529	13262952
🟩	Ljava/util/HashMap$Entry;	<SystemClassLoader>	97151	3108832
🟨	Ljava/lang/String;	<SystemClassLoader>	128924	3094176
	[Ljava/util/HashMap$Entry;	<SystemClassLoader>	34958	3042784
	Ljava/util/LinkedHashMap$Entry;	<SystemClassLoader>	45506	1820240

Object Reference



SnapShot Date : 20130829 10:05:04

Direction : ● Parent ○ Child

Sort : ● Instances ○ Size ☑ Ranking

OK
[B
Ljp/co/ntt/lab/ossc/sample/leak/bean/Air;
Ljp/co/ntt/lab/ossc/sample/leak/bean/Gass;
Ljp/co/ntt/lab/ossc/sample/leak/bean/Helium;
Lsun/security/util/DerInputBuffer;
Lsun/security/util/DerValue;
Lsun/nio/cs/UTF_8$Encoder;
Lsun/security/util/DerInputStream;
Ljava/util/zip/Inflater;
Lsun/security/util/ObjectIdentifier;
886
220833
886
20
886
13
690
33
690
21
690
21
499
11
475
7
374
11

#4 Ease of use.

Keep it simple, stupid

Ease of use.

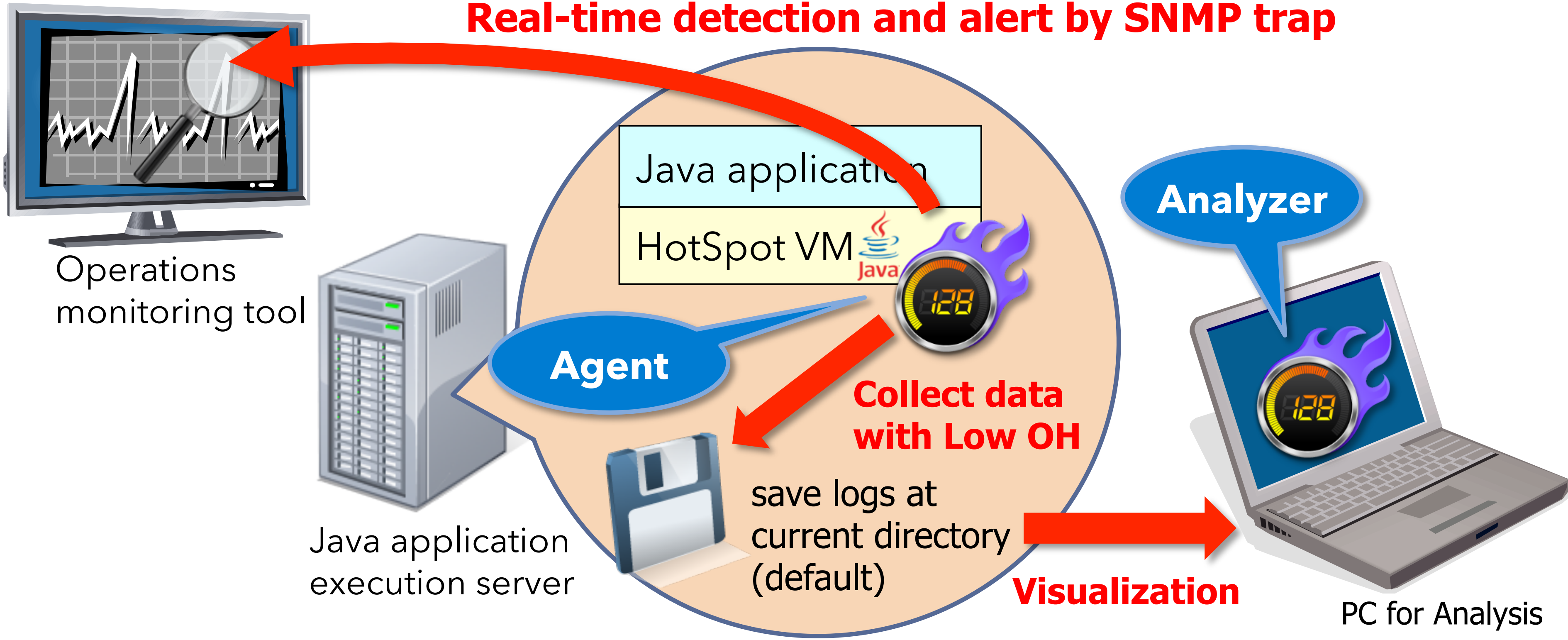
- You need only 2 steps to start

```
rpm -ivh heapstats*rpm
java -agentlib:heapstats YourApp
```

Add this JVM option

then, HeapStats go live!

Overview of HeapStats' behavior


Real-time detection and alert by SNMP trap
Java application
HotSpot VM
Java
Agent
Collect data with Low OH
save logs at current directory (default)
Visualization
Analyzer
Operations monitoring tool
Java application execution server
PC for Analysis

Requirement(s)

- HotSpot JVM
  - IcedTea, OpenJDK, OracleJDK
  - JDK 6 and later.
  - Debug symbol (e.g. debuginfo)
- Linux x86 or AMD64

Troubleshoot with HeapStats

Demonstration

One ~~fine~~ day

- Customer「Please teach us why our application got Out of Memory.」
- We must answer…
  - What happen?: leak or shortage or…
  - If leak happened, what object is the cause and how to solve it.
- Let's try troubleshoot with HeapStats.

Demonstration

Future Work

- Collect thread statistic as GC/Heap.
- More simple to install by yum/apt.
- More rich interface of analyzer.
  - Remake analyzer by JavaFX.
- Adapt JDK9 and new pause less GC "Shenandoah". (in review)

Join us!

- HeapStats:
  http://icedtea.classpath.org/wiki/HeapStats
- Mailing List: heapstats@icedtea.classpath.org
  http://icedtea.classpath.org/mailman/listinfo/heapstats
- Repository:
  hg clone http://icedtea.classpath/hg/heapstats
- Exhibition: 5013