Память JVM по полочкам

Андрей Паньгин
Одноклассники

2018

Обо мне

@AndreiPangin

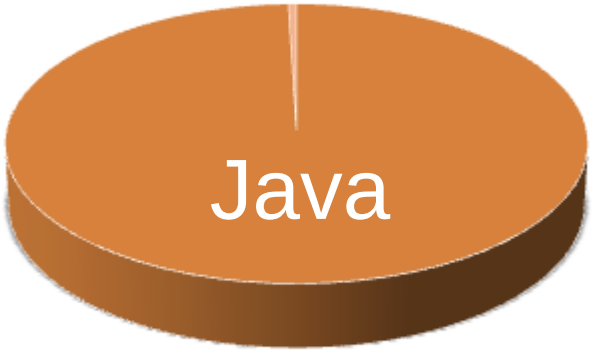
Ведущий разработчик

Специалист по HotSpot JVM

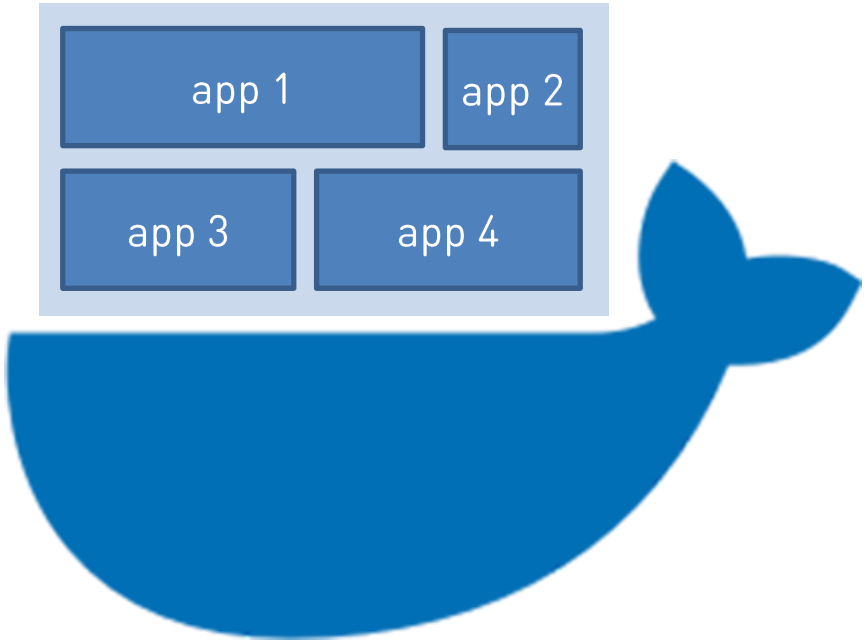
Автор one-nio и async-profiler

В Одноклассниках

- 71 M MAU
- 7000 серверов в 4 ЦОД
- 2 Тбит/с
- 200 (микро)сервисов
- до 100 K QPS на сервер



Приложения в контейнере




```
resources:
  limits:
    cpus: '4'
    memory: 10G
```

Про one-cloud: http://2017.jokerconf.com/2017/talks/zpbad8iurugyu644aoae2/

Лимиты и Java

- java -Xmx4G

```
resources:
  limits:
    cpus: '4'
    memory: 10G
```

Лимиты и Java

- java -Xmx4G

```
resources:
  limits:
    cpus: '4'
    memory: 10G
```

```
$ dmesg
[ 1078.873114] Memory cgroup out of memory: Kill process 1774 (java)
[ 1078.873187] Killed process 1774 (java) total-vm:9474252kB,
anon-rss:9361208kB, file-rss:4kB, shmem-rss:0kB
```

Java жрёт память

Q: Java consumes memory more than Xmx argument
Q: Java using up far more memory than allocated with -Xmx
Q: activemq memory usage 4 times greater than Maximum heap allocated
Q: How come Java can allocate more memory than the specified heap size
Q: Limiting Java 8 Memory Consumption
Q: Estimating maximum safe JVM heap size in 64-bit Java
Q: How do I keep memory used by JVM under control?
Q: How to find what is using physical memory in a Java process
Q: Difference between Resident Set Size (RSS) and Java total committed memory
Q: Is Linux RSS not equivalant to java Xmx + MaxMetaspaceSize?
Q: oom-killer kills java application in Docker - mismatch of memory usage reported

Память Java процесса

```
$ top -o %MEM
```

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
   83 boss      20   0 20.747g 9.022g   2104 S  39.7 56.8   8168:17 java
  416 nginx     20   0 1707516 1.256g  19200 S   0.5  8.0   5:09.09 nginx
  430 nginx     20   0 1653708 1.228g  19116 S   0.5  7.8   5:31.43 nginx
  424 nginx     20   0 1673828 1.073g  19112 S   0.5  6.8   5:15.85 nginx
  418 nginx     20   0 1622720 994.9m  19236 S   1.5  6.2   5:11.41 nginx
```

Память Java процесса

```
$ top -o %MEM
```

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
   83 boss      20   0 20.747g 9.022g   2104 S  39.7 56.8   8168:17 java
  416 nginx     20   0 1707516 1.256g  19200 S   0.5  8.0   5:09.09 nginx
  430 nginx     20   0 1653708 1.228g  19116 S   0.5  7.8   5:31.43 nginx
  424 nginx     20   0 1673828 1.073g  19112 S   0.5  6.8   5:15.85 nginx
  418 nginx     20   0 1622720 994.9m  19236 S   1.5  6.2   5:11.41 nginx
```

WTF?

Память Java процесса

```
$ top -o %MEM
```

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
   83 boss      20   0 20.747g 9.022g   2104 S  39.7 56.8   8168:17 java
  416 nginx     20   0 1707516 1.256g  19200 S   0.5  8.0   5:09.09 nginx
  430 nginx     20   0 1653708 1.228g  19116 S   0.5  7.8   5:31.43 nginx
  424 nginx     20   0 1673828 1.073g  19112 S   0.5  6.8   5:15.85 nginx
  418 nginx     20   0 1622720 994.9m  19236 S   1.5  6.2   5:11.41 nginx
```

Память Java процесса

```
$ top -o %MEM
```

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
   83 boss      20   0 20.747g 9.022g   2104 S  39.7 56.8   8168:17 java
  416 nginx     20   0 1707516 1.256g  19200 S   0.5  8.0   5:09.09 nginx
  430 nginx     20   0 1653708 1.228g  19116 S   0.5  7.8   5:31.43 nginx
  424 nginx     20   0 1673828 1.073g  19112 S   0.5  6.8   5:15.85 nginx
  418 nginx     20   0 1622720 994.9m  19236 S   1.5  6.2   5:11.41 nginx
```

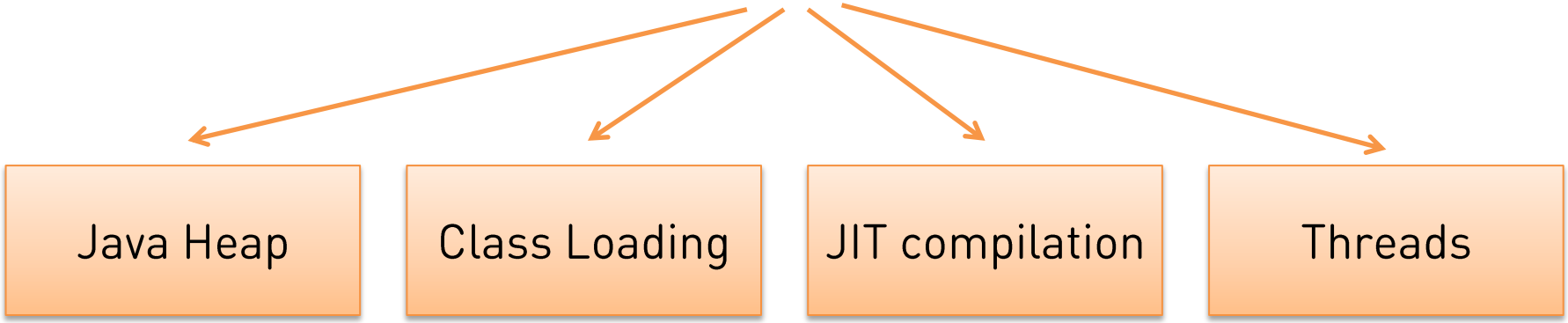
Linux о памяти

- Статистика памяти ОС
  /proc/meminfo
- Карта виртуальной памяти процесса
  /proc/PID/maps
  /proc/PID/smaps
- man procfs

pmap -x PID

```
Address           Kbytes       RSS     Dirty Mode  Mapping
0000000000400000       4         4         0 r-x-- java
0000000080000000 4194304   3898444   3898444 rw---   [ anon ]
0000000080000000       0         0         0 rw---   [ anon ]
0000000100280000 1046016         0         0 -----   [ anon ]
       ...
00007f5c07c05000      76        76         0 r--s- resources.jar
00007f5c07c18000      16        16         0 r--s- jsse.jar
00007f5c07c1e000    1888      1888         0 r--s- rt.jar
       ...
00007f5c1810e000    1024       192         0 r-x-- libm-2.17.so
00007f5c18410000   13104     11448         0 r-x-- libjvm.so
00007f5c19c78000    1752      1484         0 r-x-- libc-2.17.so
       ...
---------------- -------- -------  -------
total kB         13411948 9452628  9449116
```

JVM Runtime



Native Memory Tracking

```
java -XX:NativeMemoryTracking=[summary|detail]
```

- 5-10% perf overhead
- +2 words / malloc
- аллокации только внутри JVM

Native Memory Tracking

```
java -XX:NativeMemoryTracking=[summary|detail]
```

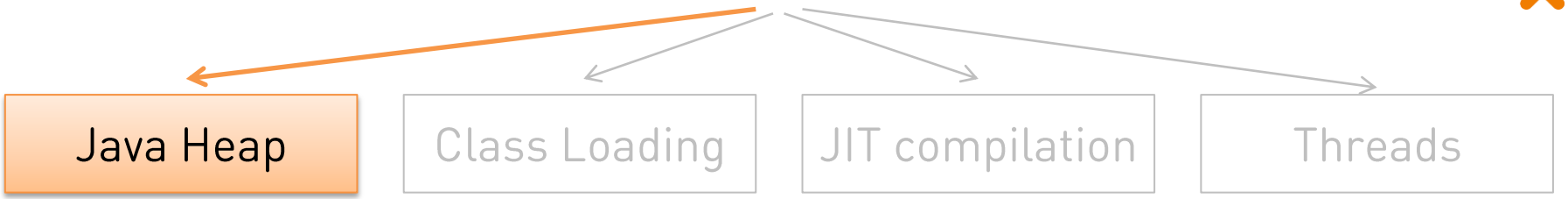
- 5-10% perf overhead
- +2 words / malloc
- аллокации только внутри JVM

```
jcmd PID VM.native_memory [detail]
```

NMT summary

```
Total: reserved=6796641KB, committed=5508529KB
- Java Heap (reserved=2097152KB, committed=2097152KB)
            (mmap: reserved=2097152KB, committed=2097152KB)

-     Class (reserved=1068151KB, committed=20471KB)
            (classes #3348)
            (malloc=1143KB #11651)
            (mmap: reserved=1067008KB, committed=19328KB)

-    Thread (reserved=27346KB, committed=27346KB)
            (thread #75)
            (stack: reserved=26920KB, committed=26920KB)
            (malloc=243KB #377)
            (arena=184KB #149)
                                ...
```

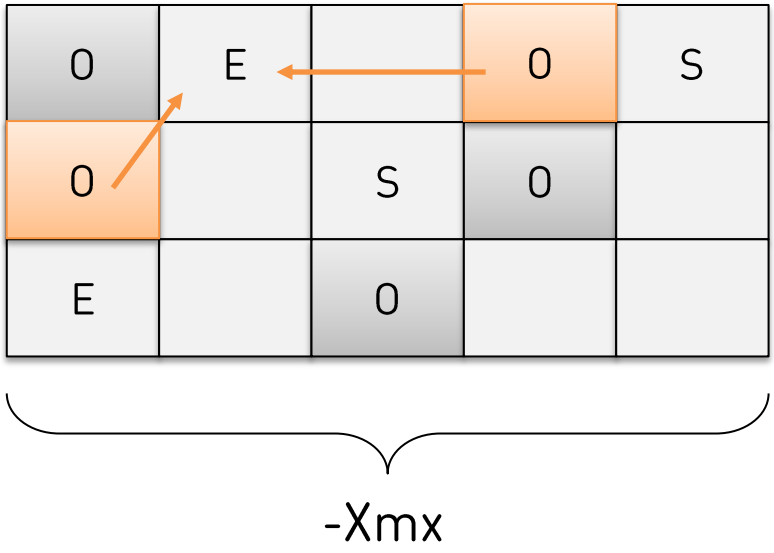
JVM Runtime




```
- Java Heap (reserved=2097152KB, committed=2097152KB)
            (mmap: reserved=2097152KB, committed=2097152KB)


-         GC (reserved=17289KB, committed=17289KB)
            (malloc=10689KB #184)
            (mmap: reserved=6600KB, committed=6600KB)
```

Java Heap & GC

G1 Heap



Накладные расходы на GC

- Mark bitmap
- Mark stacks
  -XX:MarkStackSizeMax
- Remembered sets

-XX:G1HeapRegionSize

Накладные расходы GC

GC	Overhead, MB	%
Serial	7	0,3%
Shenandoah	36	1,8%
Parallel	86	4,2%
CMS	141	6,9%
G1	165	8,1%
Z	206	10,1%

OpenJDK 11 | Heap size = 2 GB

Размер хипа

java -Xmx4G -Xms4G HelloWorld

```
  PID USER   PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
   86 boss   20   0 6544056  53868  16680 S   0.0  1.1   0:00.08 java
```

Размер хипа

java -Xmx4G -Xms4G -XX:+AlwaysPreTouch HelloWorld

```
  PID USER   PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
   86 boss   20   0 6544056   4.0g  16684 S   0.0 69.3   0:00.08 java
```

Размер хипа

```
java -Xmx4G -Xms4G HelloWorld
```

начальный размер,
а не минимальный

Размер хипа

java -Xmx4G -Xms4G HelloWorld

начальный размер,
а не минимальный

java -Xmx4G -Xms4G -XX:-AdaptiveSizePolicy HelloWorld

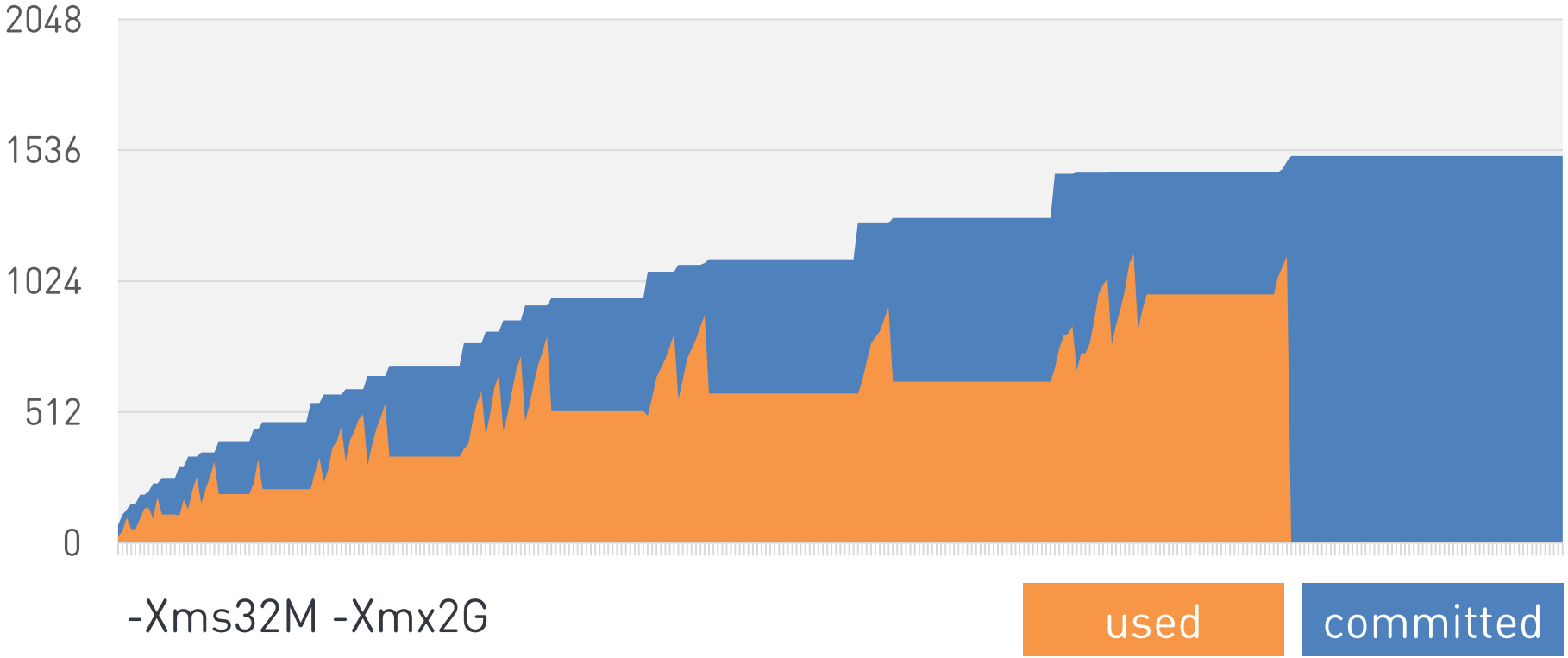
Adaptive size policy

- Подбирает размер хипа исходя из статистики

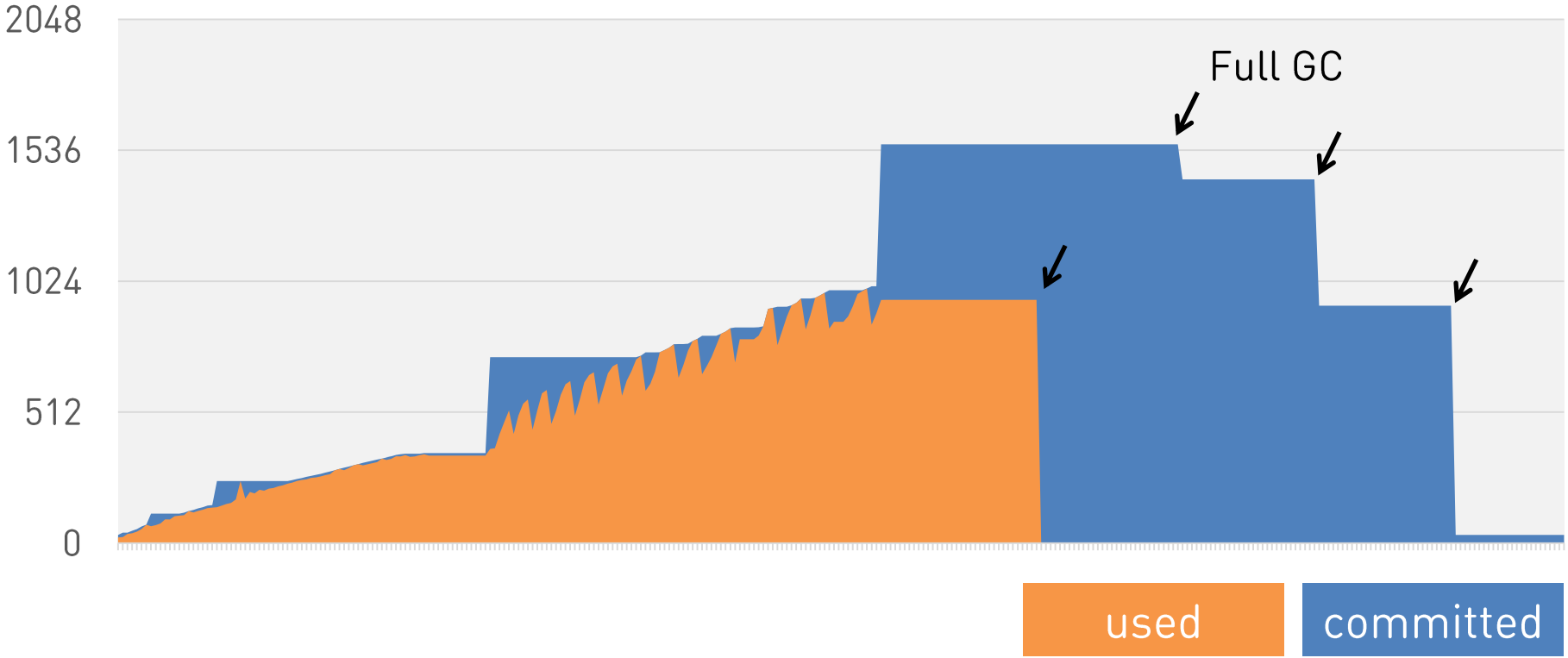
  -XX:MinHeapFreeRatio=40 (расширение)
  -XX:MaxHeapFreeRatio=70 (сжатие)

- Позволяет отдавать память системе!

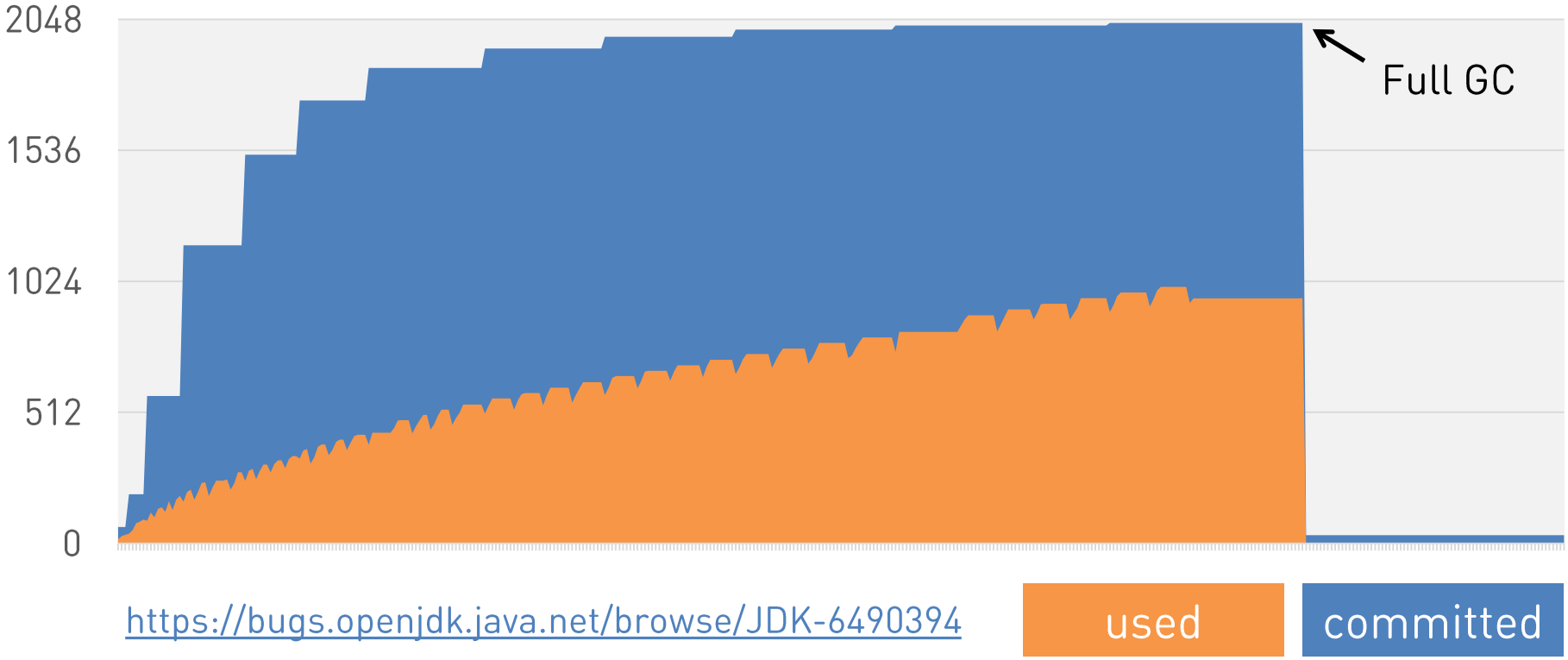
Uncommit: Parallel



Uncommit: CMS

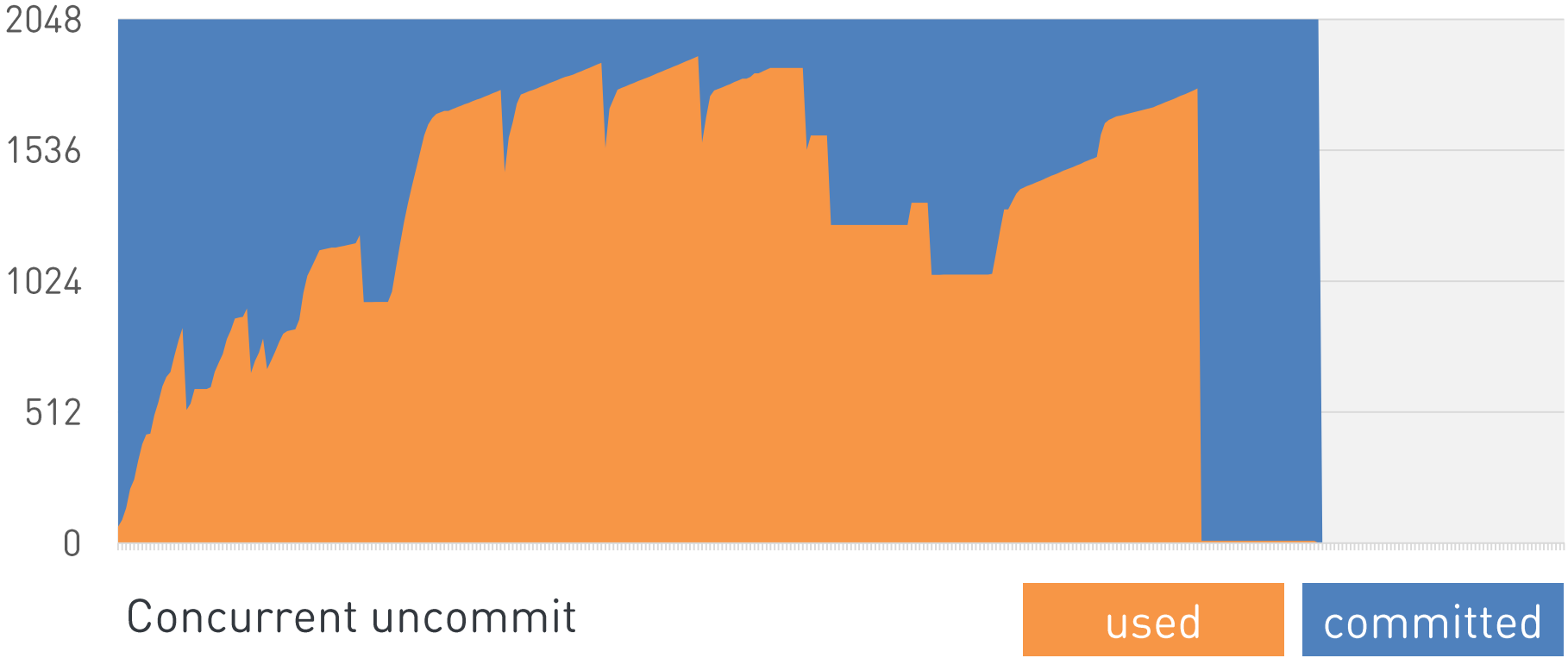


Uncommit: G1

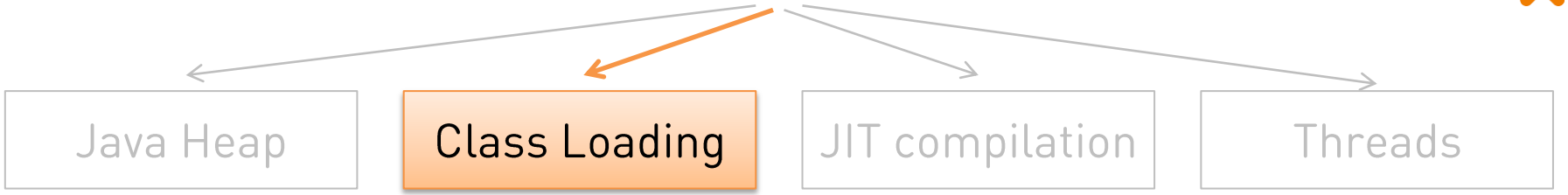


https://bugs.openjdk.java.net/browse/JDK-6490394

Uncommit: Shenandoah



JVM Runtime




```
-     Class (reserved=1068151KB, committed=20471KB)
            (classes #3348)
            (malloc=1143KB #11651)
            (mmap: reserved=1067008KB, committed=19328KB)
```

Class metadata | JDK 10+

```
-     Class (reserved=1073841KB, committed=28593KB)
            (classes #3967)
            (  instance classes #3694, array classes #273)
            (malloc=689KB #9228)
            (mmap: reserved=1073152KB, committed=27904KB)
            (  Metadata:   )
            (    reserved=24576KB, committed=24576KB)
            (    used=24131KB)
            (    free=445KB)
            (    waste=0KB =0.00%)
            (  Class space:)
            (    reserved=1048576KB, committed=3328KB)
            (    used=3003KB)
            (    free=325KB)
            (    waste=0KB =0.00%)
```

Class metadata

- Metaspace
  - классы
  - методы
  - constant pools
  - символы
  - аннотации…

```
-XX:MaxMetaspaceSize=[unlimited]
```

~~java.lang.OutOfMemoryError: PermGen space~~

UseCompressedClassPointers

- Metaspace
  - методы
  - constant pools
  - символы
  - аннотации…

```
-XX:MaxMetaspaceSize=[unlimited]
```

- Compressed class space
  - классы

```
-XX:CompressedClassSpaceSize=1G
                             max 3G
```

UseCompressedClassPointers

- Metaspace
  - методы
  - constant pools
  - символы
  - аннотации…

`-XX:MaxMetaspaceSize=[unlimited]`

- Compressed class space
  - классы

`-XX:CompressedClassSpaceSize=1G`
`max 3G`

```
java.lang.OutOfMemoryError:
Compressed class space
```

Статистика ClassLoader

```
$ jmap -clstats PID                  # JDK 8
$ jcmd PID VM.classloader_stats      # JDK 9+
```

Статистика ClassLoader

```
$ jmap -clstats PID                    # JDK 8
$ jcmd PID VM.classloader_stats        # JDK 9+
```

Classes	ChunkSz	BlockSz	Type
100000	88203264	88000488	one.nio.gen.BytecodeGenerator
4800	36528128	36459272	<boot class loader>
469	1307648	835744	+ unsafe anonymous classes
1	3072	2016	jdk.internal.reflect.DelegatingClassLoader
239	1826816	1797088	org.netbeans.MainImpl$BootClassLoader
11	147456	143744	jdk.internal.loader.ClassLoaders$PlatformClassLoader
Total = 125	101983	100848640	100388560

Статистика по всем классам

```
$ jcmd PID GC.class_stats
```

Index	Super	InstBytes	KlassBytes	annotations	CpAll	MethodCount	Bytecodes	MethodAll	ROAll	RWAll	Total	ClassName
1	-1	14935368	504	0	0	0	0	0	24	616	640	[I
2	-1	3649224	504	0	0	0	0	0	24	616	640	[B
3	25	1296960	616	128	14216	109	4577	67648	18640	65272	83912	java.lang.String
4	-1	1109248	504	0	0	0	0	0	24	616	640	[J
5	-1	1099112	504	0	0	0	0	0	32	616	648	[Ljava.util.HashMap$Node;
6	25	1032096	672	0	22112	139	5682	65928	24616	65992	90608	java.lang.Class
7	25	856736	584	0	1384	7	149	2280	1152	3416	4568	java.util.HashMap$Node
8	-1	726536	504	0	0	0	0	0	24	616	640	[Ljava.lang.Object;
9	-1	350744	504	0	0	0	0	0	24	616	640	[C
10	25	262752	608	0	1512	8	240	2112	1328	3232	4560	java.util.Hashtable$Entry
11	5725	239136	1024	0	7904	51	4065	35360	12672	32600	45272	java.util.HashMap
12	-1	143016	504	0	0	0	0	0	24	616	640	[D
13	7	124680	584	0	504	1	10	624	304	1640	1944	java.util.LinkedHashMap$Entry
14	5721	121512	1440	0	7000	64	2681	29776	11232	27976	39208	java.util.ArrayList
15	-1	106624	504	0	0	0	0	0	24	616	640	[Ljava.security.ProtectionDomain;
16	5468	95072	560	0	448	2	13	1144	272	2016	2288	java.lang.ref.WeakReference
17	7166	93296	968	0	8152	40	2051	24096	6600	27256	33856	org.netbeans.lib.profiler.heap.ClassDump

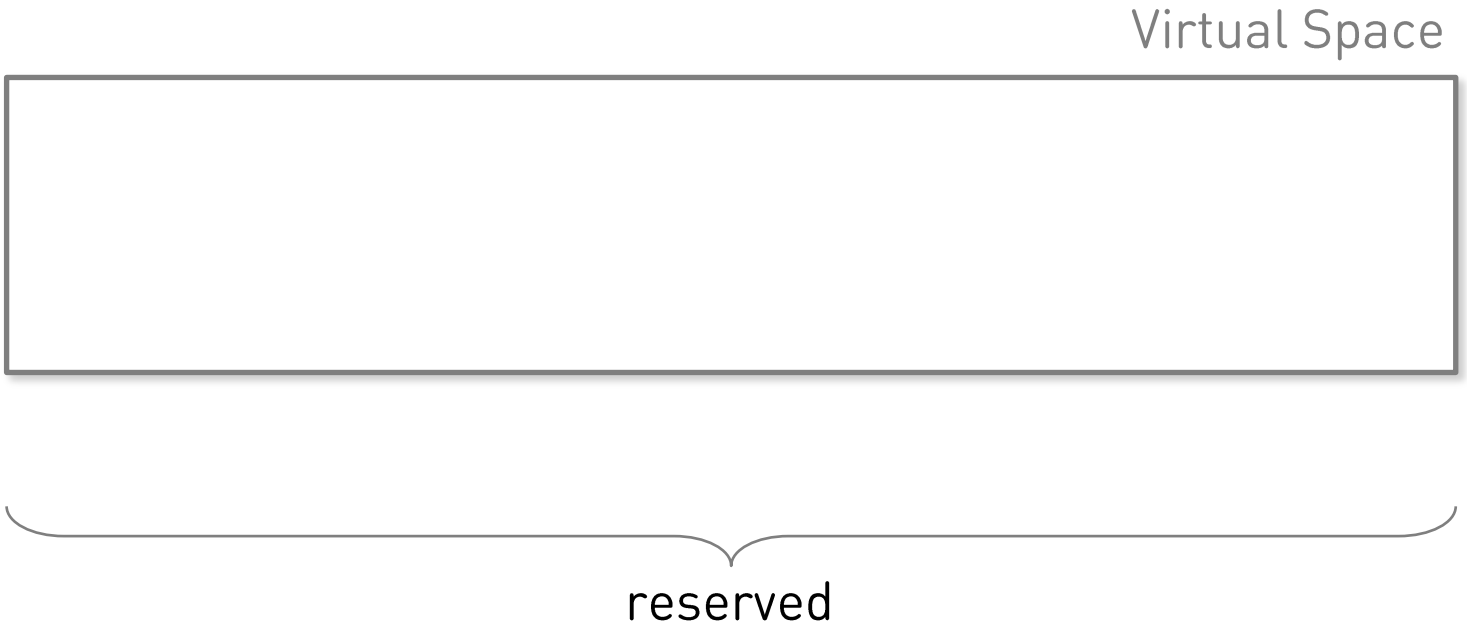
Размеры Metaspace

```
-XX:+PrintGCDetails | -Xlog:gc+heap
```

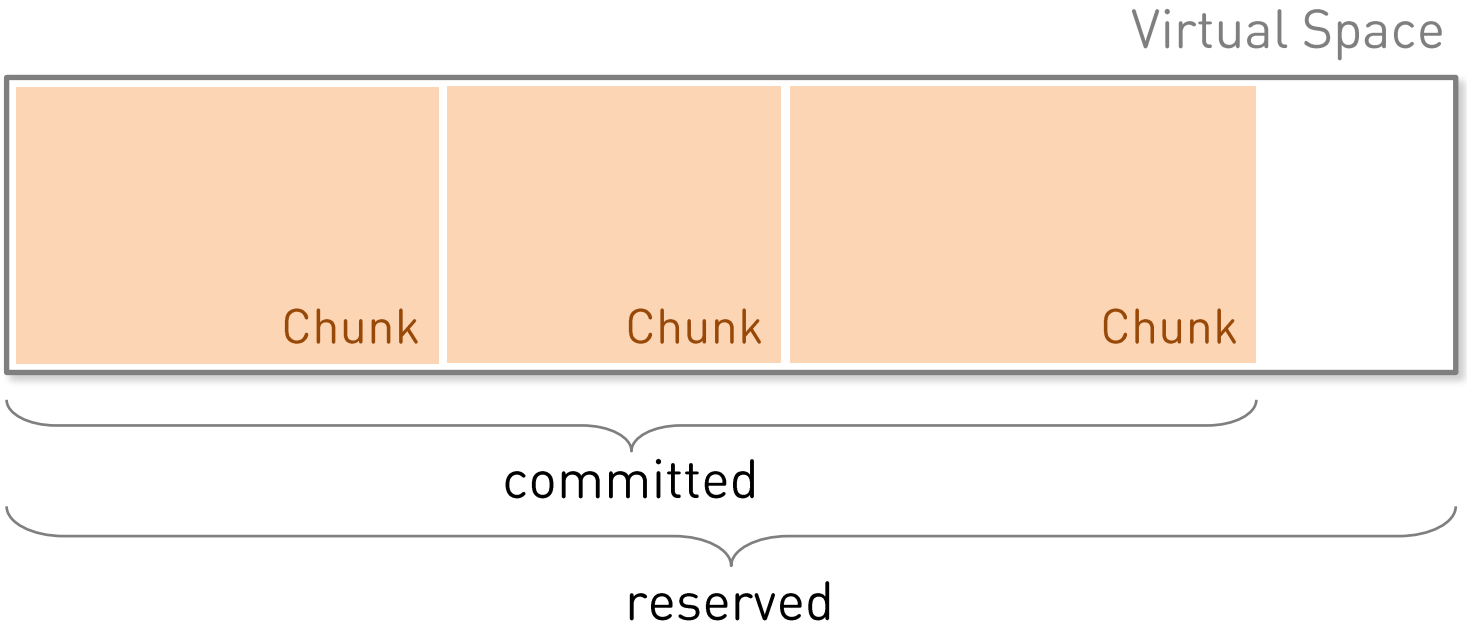
```
Metaspace     used 219005K, capacity 272729K, committed 273152K, reserved 1280000K
 class space  used 36068K, capacity 43069K, committed 43264K, reserved 1048576K
```

```
$ jcmd PID VM.metaspace    # JDK 9+
```

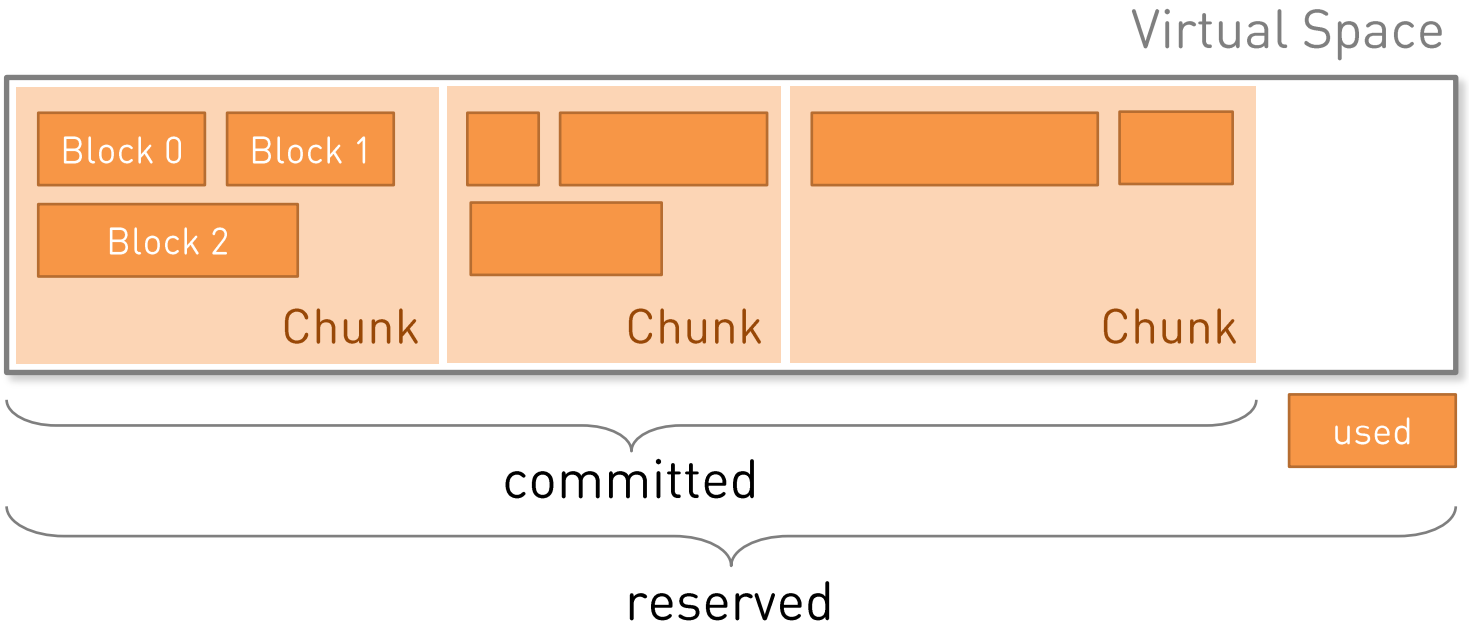
Устройство Metaspace



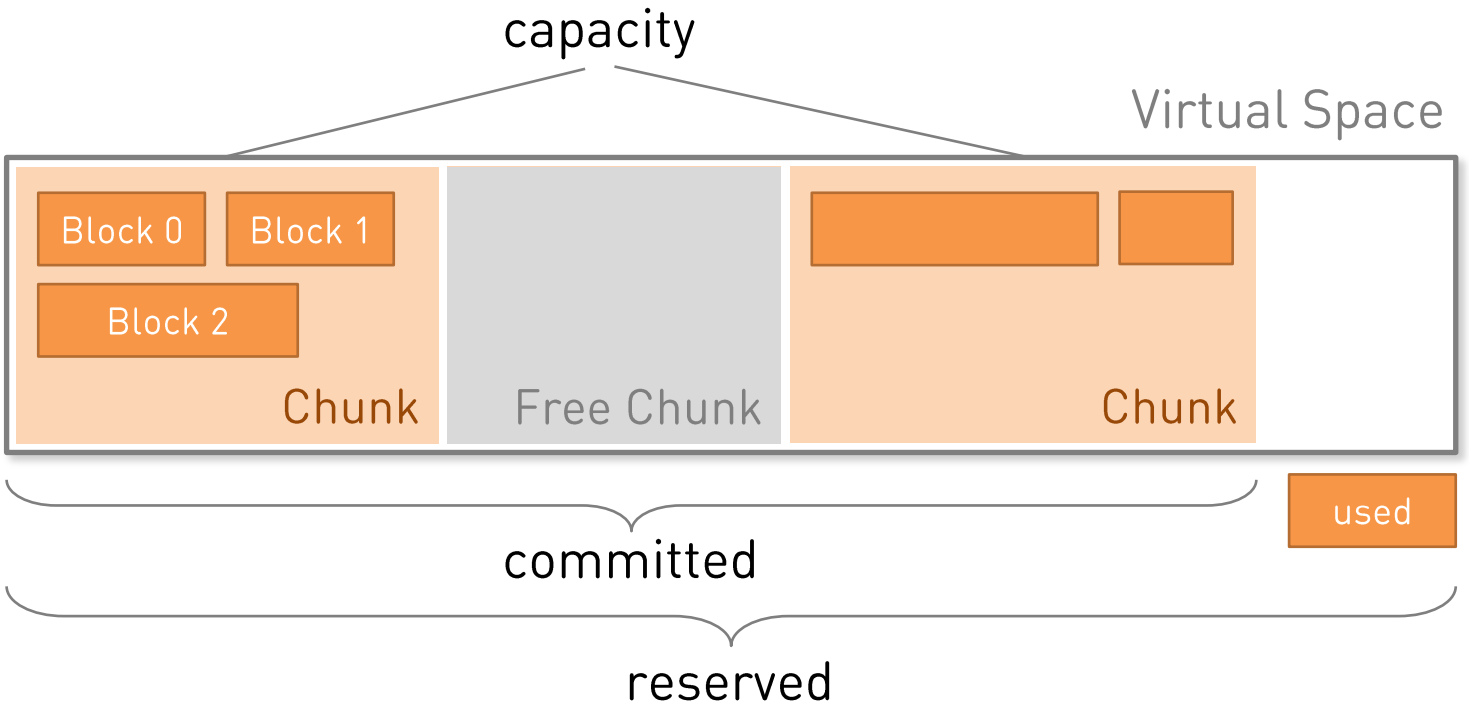
Устройство Metaspace



Устройство Metaspace



Устройство Metaspace



used ⩽ capacity ⩽ committed ⩽ reserved

Тюнинг Metaspace

-XX:MaxMetaspaceSize
-XX:CompressedClassSpaceSize

https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/considerations.html

Тюнинг Metaspace

-XX:MaxMetaspaceSize
-XX:CompressedClassSpaceSize

-XX:MetaspaceSize=20M ⟵ high-water mark

```
[Full GC (Metadata GC Threshold)  1032K->894K(198656K), 0.077995 secs]
```

https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/considerations.html

Тюнинг Metaspace

-XX:MaxMetaspaceSize
-XX:CompressedClassSpaceSize

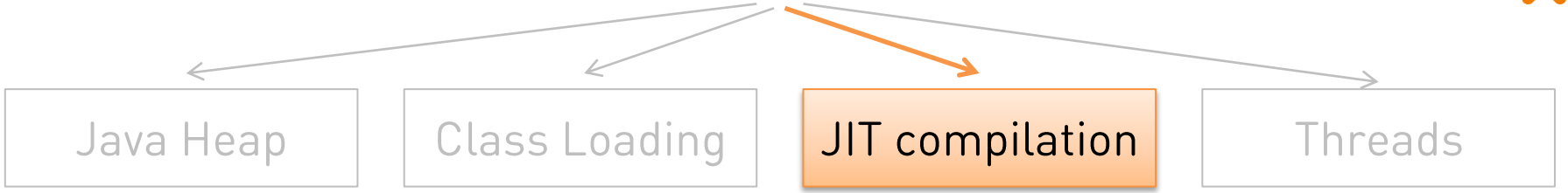
-XX:MetaspaceSize=20M ⟵ high-water mark

```
[Full GC (Metadata GC Threshold)  1032K->894K(198656K), 0.077995 secs]
```

-XX:MinMetaspaceFreeRatio=40
-XX:MaxMetaspaceFreeRatio=70 ⟵ expand / shrink

https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/considerations.html

JVM Runtime




```
-      Code (reserved=251072KB, committed=10640KB)
            (malloc=1472KB #2952)
            (mmap: reserved=249600KB, committed=9168KB)

-  Compiler (reserved=309KB, committed=309KB)
            (malloc=179KB #254)
            (arena=131KB #3)
```

Compiler

- Code Cache
  - nmethods
  - и не только: interpreter, stubs

```
-XX:InitialCodeCacheSize
-XX:ReservedCodeCacheSize
```

Compiler

- Code Cache
  - nmethods
  - и не только: interpreter, stubs

```
-XX:InitialCodeCacheSize
-XX:ReservedCodeCacheSize
```

- «Арены» компилятора
  - IR, структуры
  - × (N потоков)

Compiler

- Code Cache
  - nmethods
  - и не только: interpreter, stubs

`-XX:InitialCodeCacheSize`
`-XX:ReservedCodeCacheSize`

- «Арены» компилятора
  - IR, структуры
  - × (N потоков)

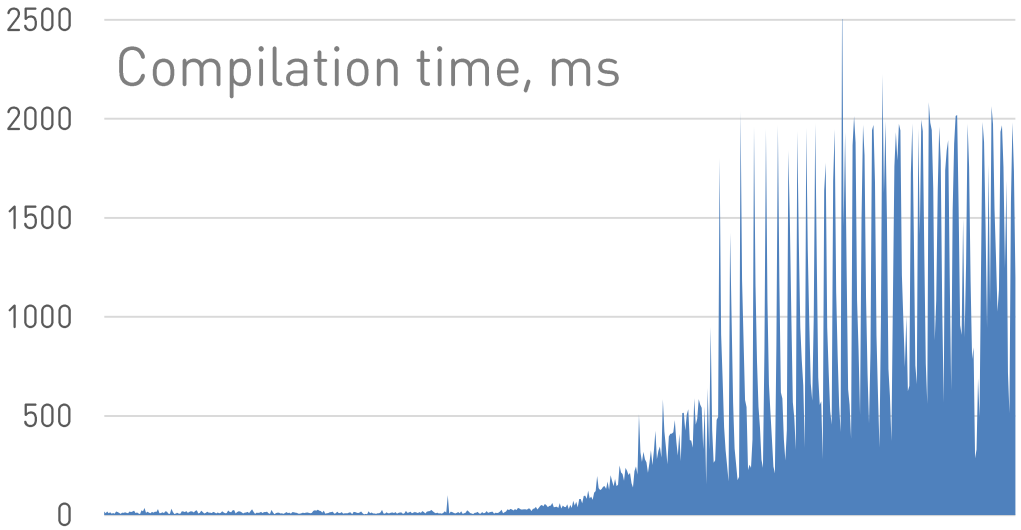
`Нет у GraalVM`

Tiered Compilation

- C1 + C2 = больше кода
- ReservedCodeCacheSize * 5 (48 → 240MB)

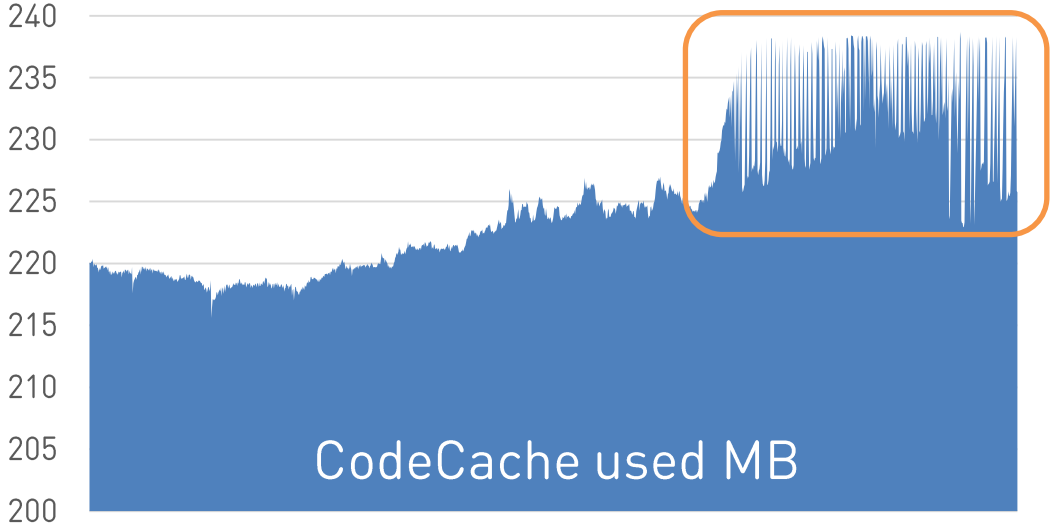
Tiered Compilation

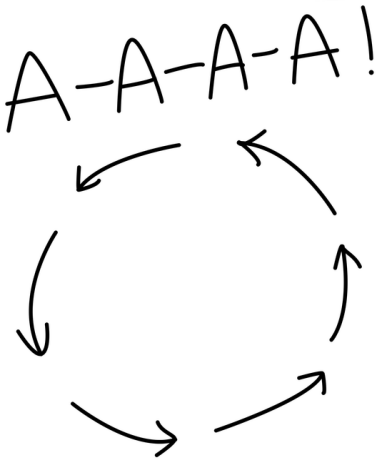
- C1 + C2 = больше кода
- ReservedCodeCacheSize * 5 (48 → 240MB)



Tiered Compilation

- C1 + C2 = больше кода
- ReservedCodeCacheSize * 5 (48 → 240MB)





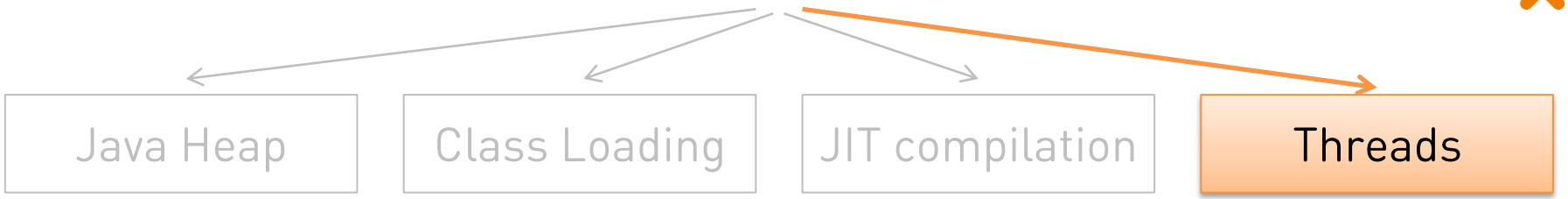
Мониторинг

Active Memory Pools

Pool Name	Type	Used	Max	Usage	Peak Used	Peak Max
Par Survivor Space	HEAP	32,7 MiB	102 MiB	31,9 %	32,7 MiB	102 MiB
Par Eden Space	HEAP	236 MiB	819 MiB	28,8 %	236 MiB	819 MiB
CMS Old Gen	HEAP	3,53 GiB	9 GiB	39,2 %	3,53 GiB	9 GiB
Code Cache	NON_HEAP	75,9 MiB	200 MiB	38 %	75,9 MiB	200 MiB
Compressed Class Space	NON_HEAP	36,1 MiB	1 GiB	3,53 %	36,1 MiB	1 GiB
Metaspace	NON_HEAP	221 MiB			221 MiB	

Overview | MBean Browser | Triggers | System | Memory | Threads | Diagnostic Commands

JVM Runtime




```
-    Thread (reserved=486826KB, committed=486826KB)
            (thread #475)
            (stack: reserved=486400KB, committed=486400KB)
            (malloc=243KB #377)
            (arena=184KB #149)
```

Thread stack

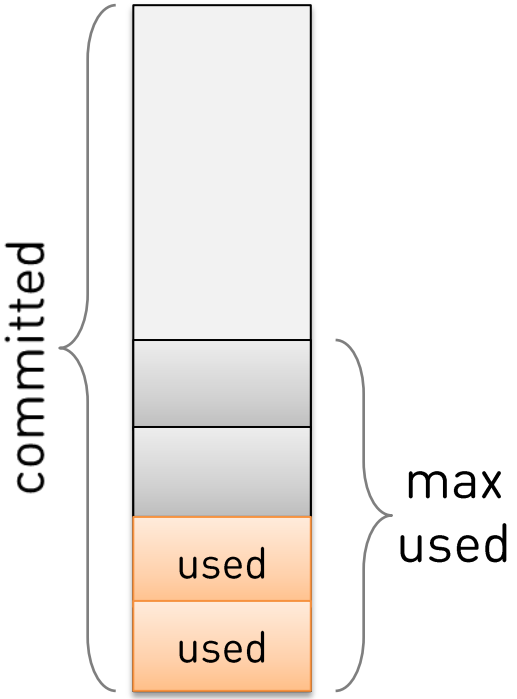
-Xss1M (min ~200K)

-XX:VMThreadStackSize
-XX:CompilerThreadStackSize

Thread stack

-Xss1M (min ~200K)

-XX:VMThreadStackSize
-XX:CompilerThreadStackSize



committed != resident
https://github.com/apangin/jstackmem

Native Memory: Symbols

```
-    Symbol (reserved=4633KB, committed=4633KB)
            (malloc=2994KB #19875)
            (arena=1638KB #1)
```

↓ **SymbolTable**
names, signatures, etc.

↓ **StringTable**
interned strings

Статистика таблиц символов

```
$ jcmd 15188 VM.stringtable | VM.symboltable
```

```
StringTable statistics:
Number of buckets       :     60013 =    480104 bytes, each 8
Number of entries       :  10097954 = 242350896 bytes, each 24
Number of literals      :  10097954 = 484773056 bytes, avg  48.007
Total footprint         :           = 727604056 bytes
Average bucket size     :   168.263
Variance of bucket size :    57.271
Std. dev. of bucket size:     7.568
Maximum bucket size     :       197
```

Статистика таблиц символов

```
$ jcmd 15188 VM.stringtable | VM.symboltable
```

```
StringTable statistics:
Number of buckets       :     60013 =    480104 bytes, each 8
Number of entries       :  10097954 = 242350896 bytes, each 24
Number of literals      :  10097954 = 484773056 bytes, avg  48.007
Total footprint         :           = 727604056 bytes
Average bucket size     :   168.263
Variance of bucket size :    57.271
Std. dev. of bucket size:     7.568
Maximum bucket size     :       197
```

Как распечатать содержимое таблиц:

https://stackoverflow.com/q/35238902/3448419

Сколько-сколько?!

```
-  Internal (reserved=4112230KB, committed=4112230KB)
            (malloc=4112230KB #10535448)
```

Сколько-сколько?!

```
-  Internal (reserved=4112230KB, committed=4112230KB)
            (malloc=4112230KB #10535448)
```

- Off-Heap
  - Direct ByteBuffers
  - Unsafe.allocateMemory
- JDK 11: Internal → Other

Direct ByteBuffers

ByteBuffer.allocateDirect

FileChannel.map

Direct ByteBuffers

ByteBuffer.allocateDirect

FileChannel.map

-XX:MaxDirectMemorySize

По умолчанию = Xmx ¯_(ツ)_/¯

Direct ByteBuffers

ByteBuffer.allocateDirect

-XX:MaxDirectMemorySize

По умолчанию = Xmx ¯_(ツ)_/¯

FileChannel.map

Не ограничивается

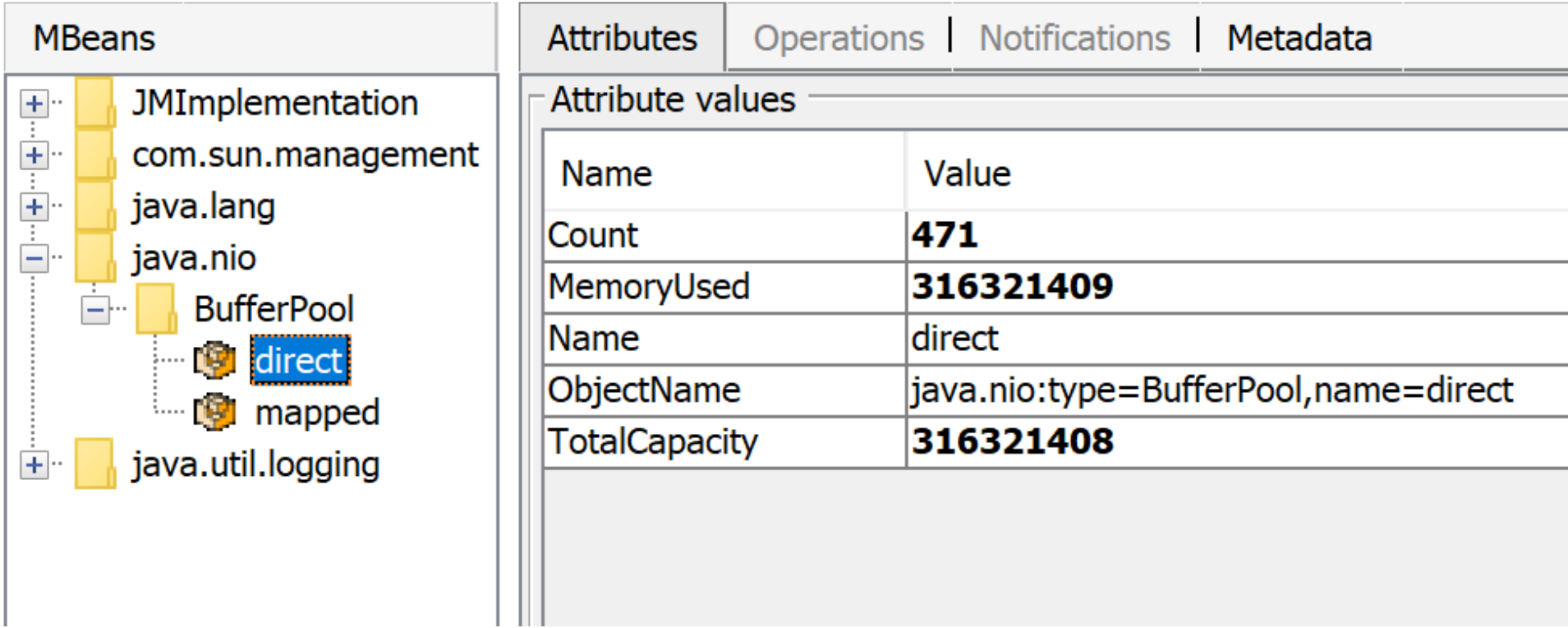
Не учитывается NMT

RSS?

pmap -x PID

```
Address           Kbytes     RSS   Dirty Mode  Mapping

0000000540000000 8388608 8388608 8388608 rw---   [ anon ]
00007f2991909000 1822932    2920       0 r--s- photos-uploads-jb-1290-Data.db
00007f2a00d3e000 2097152    5104       0 r--s- photos-uploads-jb-1290-Data.db
00007f2aa345f000   62368    6660       0 r--s- photos-refs-jb-3161-Index.db
00007f2aa7147000   62604    5508       0 r--s- photos-refs-jb-3160-Index.db
00007f2b3e557000   39592   32956       0 r--s- photos-photos-jb-17405-Index.db
00007f2b40c01000   39600   33092       0 r--s- photos-photos-jb-17404-Index.db
00007f2ce9954000   39600   33048       0 r--s- photos-photos-jb-17266-Index.db
00007f2ffaec9000   55744   52268       0 r--s- photos-albums-jb-1705-Index.db
00007f2ffe539000   55756   52184       0 r--s- photos-albums-jb-1704-Index.db
00007f30a8035000    7904    7904       0 r--s- photos-counters-jb-1896-Index.db
```

BufferPool MBean



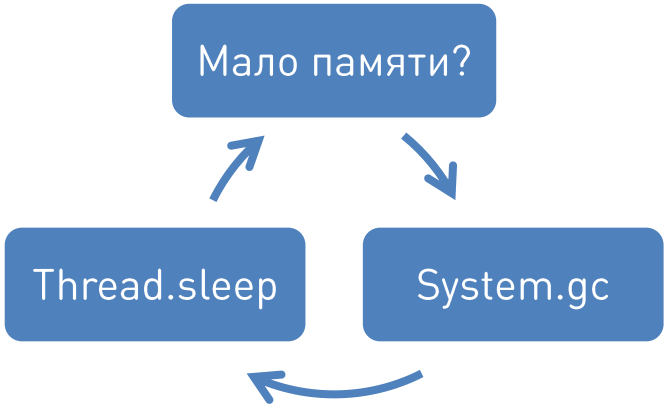
Освобождение Direct ByteBuffers

- Автоматически после GC

Освобождение Direct ByteBuffers

- Автоматически после GC

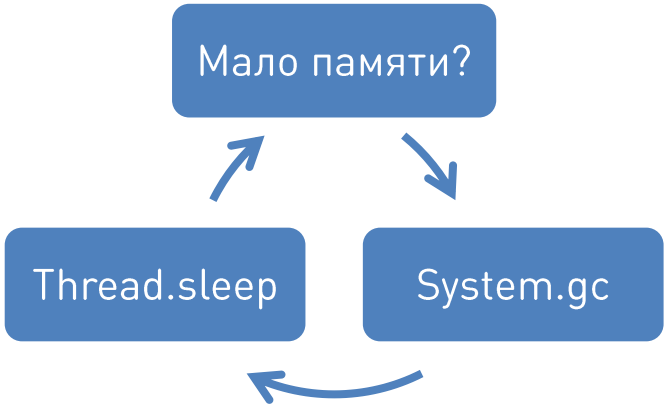
java.nio.Bits.reserveMemory



Освобождение Direct ByteBuffers

- Автоматически после GC

java.nio.Bits.reserveMemory



✖ -XX:+DisableExplicitGC

✔ -XX:+ExplicitGCInvokesConcurrent

Может, лучше Heap ByteBuffer?

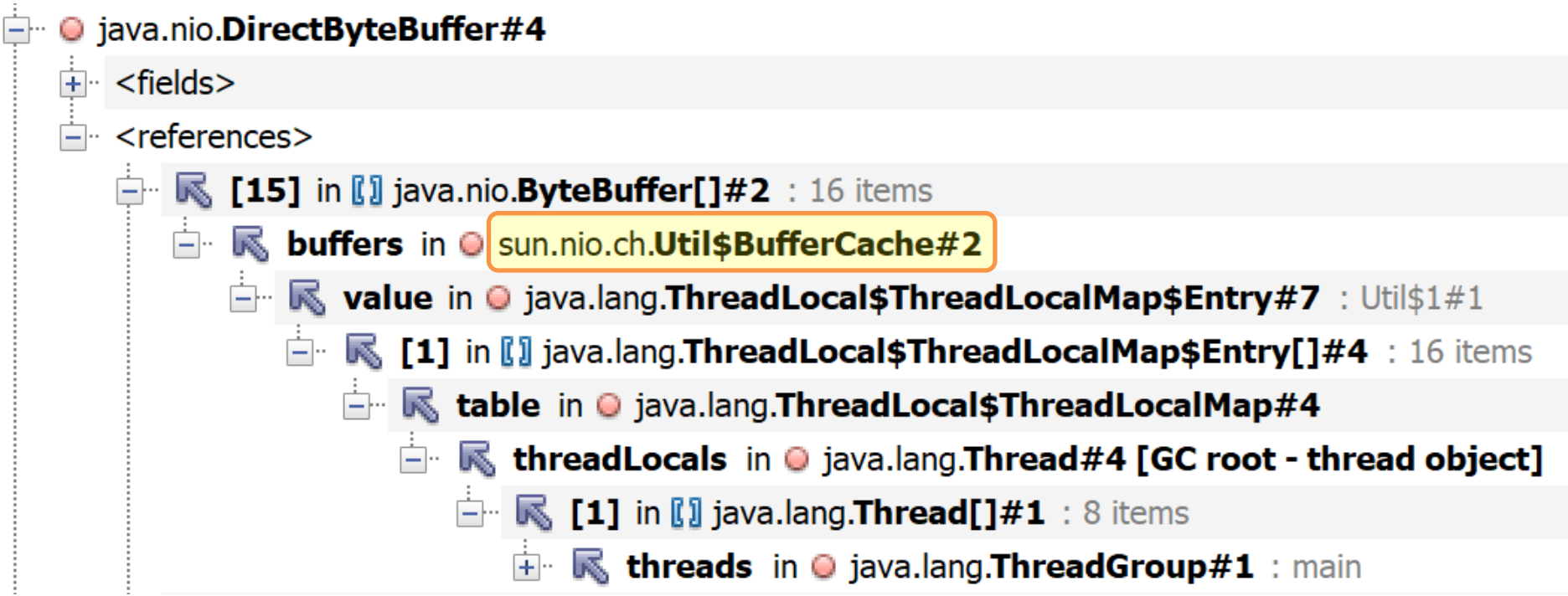
```
byte[] array = new byte[8192];

// ... fill array ...

socketChannel.write(ByteBuffer.wrap(array));
```

Может, лучше Heap ByteBuffer?

```
byte[] array = new byte[8192];

// ... fill array ...

socketChannel.write(ByteBuffer.wrap(array));
```

allocate temp direct buffer → write(direct buffer) → release temp direct buffer

DirectByteBuffer в хип дампе



sun.nio.ch.Util.BufferCache

- Никогда не уменьшается
- ThreadLocal
  https://bugs.openjdk.java.net/browse/JDK-8202788
- -Djdk.nio.maxCachedBufferSize=
  с JDK 8u102

- Видишь утечку?
- Нет.
- А она есть!

```
while (true) {
    ByteBuffer.allocateDirect(random.nextInt(10000));
}
```

-Xmx1G -XX:MaxDirectMemorySize=2G -XX:+UseG1GC

Утечка нативной памяти

```
$ top
```

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   72 boss      20   0 9198868 7.186g  11948 S 766.7 45.85   2:22.87 java
```

7 GB

Утечка нативной памяти

```
$ top
```

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   72 boss      20   0 9198868 7.186g  11948 S 766.7 45.85   2:22.87 java
```

7 GB

```
$ jcmd 72 VM.native_memory
```

```
Native Memory Tracking:

Total: reserved=4152240KB, committed=2851224KB
```

2.7 GB

Карта памяти

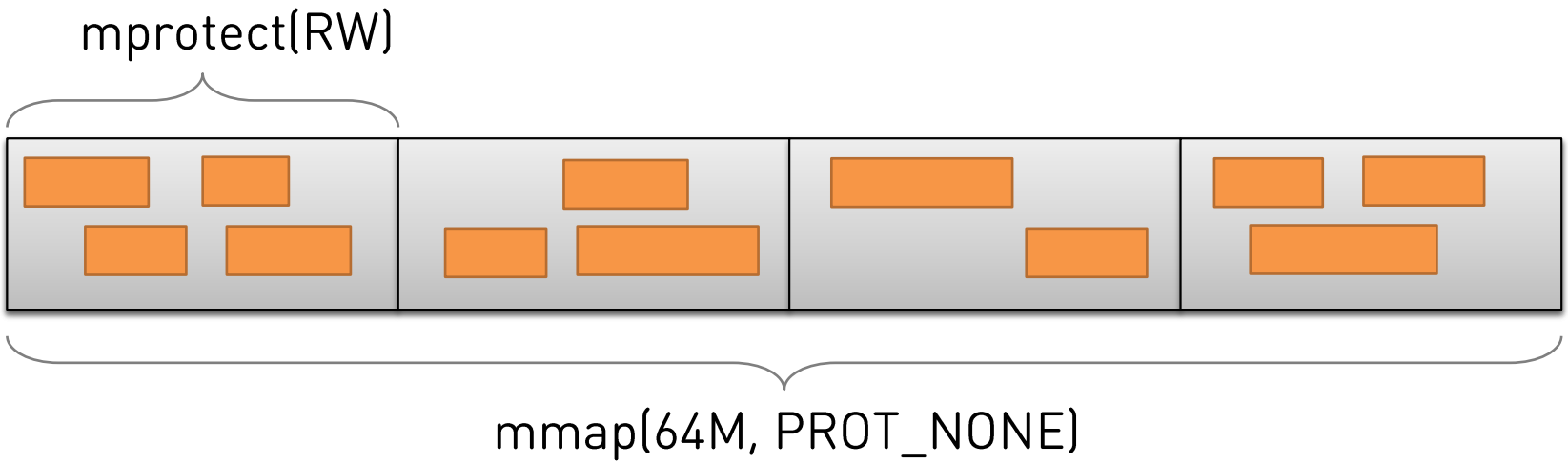
```
$ pmap -x 72
```

```
Address           Kbytes     RSS   Dirty Mode  Mapping

00007fb64c000000   65508   65508   65508 rw---   [ anon ]
00007fb64c000000       0       0       0 rw---   [ anon ]
00007fb64fff9000      28       0       0 -----   [ anon ]
00007fb64fff9000       0       0       0 -----   [ anon ]
00007fb650000000   65536   65536   65536 rw---   [ anon ]
00007fb650000000       0       0       0 rw---   [ anon ]
00007fb654000000   65536   65536   65536 rw---   [ anon ]
00007fb654000000       0       0       0 rw---   [ anon ]
00007fb658000000   65536   65536   65536 rw---   [ anon ]
00007fb658000000       0       0       0 rw---   [ anon ]
```

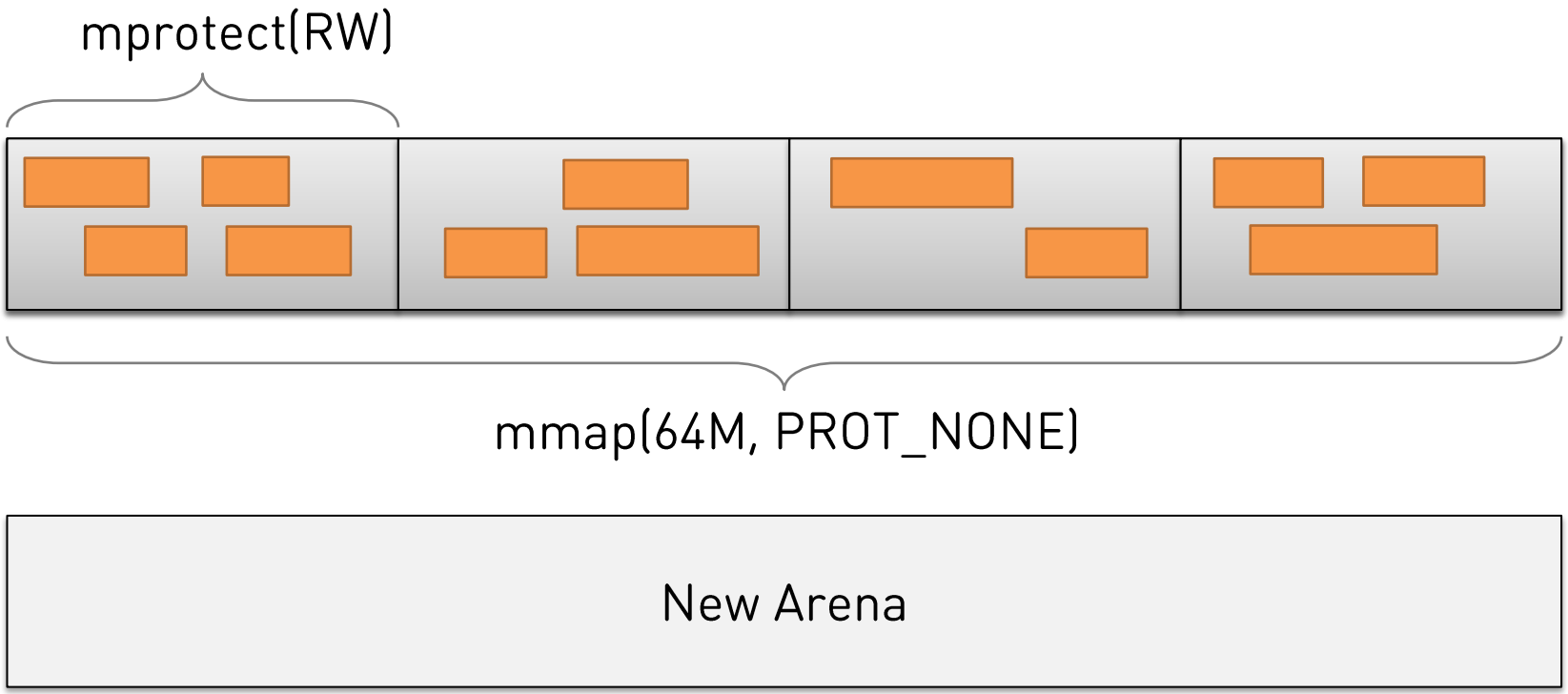
malloc

malloc

malloc



malloc



Альтернативные аллокаторы

- jemalloc
  http://jemalloc.net
- tcmalloc
  https://github.com/gperftools/gperftools

Альтернативные аллокаторы

- jemalloc
  http://jemalloc.net
- tcmalloc
  https://github.com/gperftools/gperftools

- LD_PRELOAD=/usr/lib64/libjemalloc.so

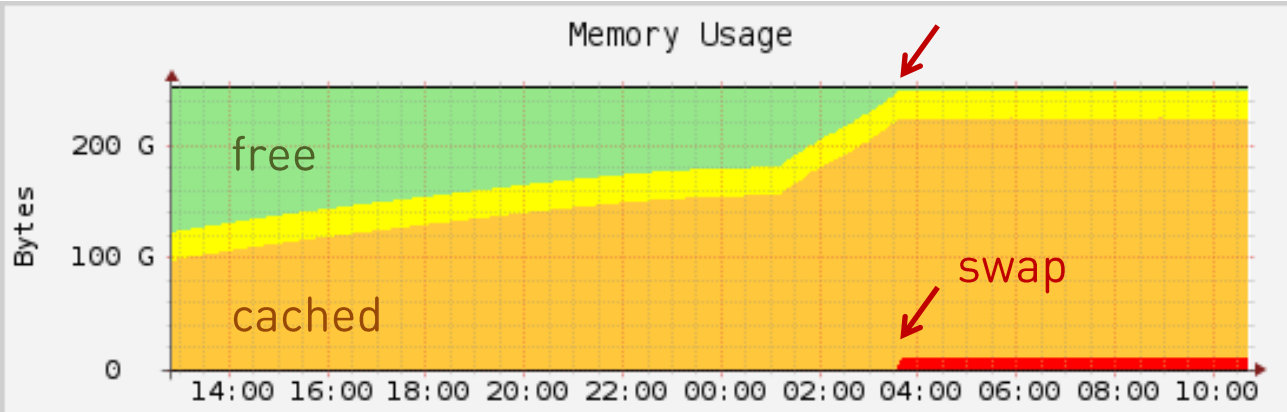
- RSS: 7 → 2.8 GB

jemalloc profiler

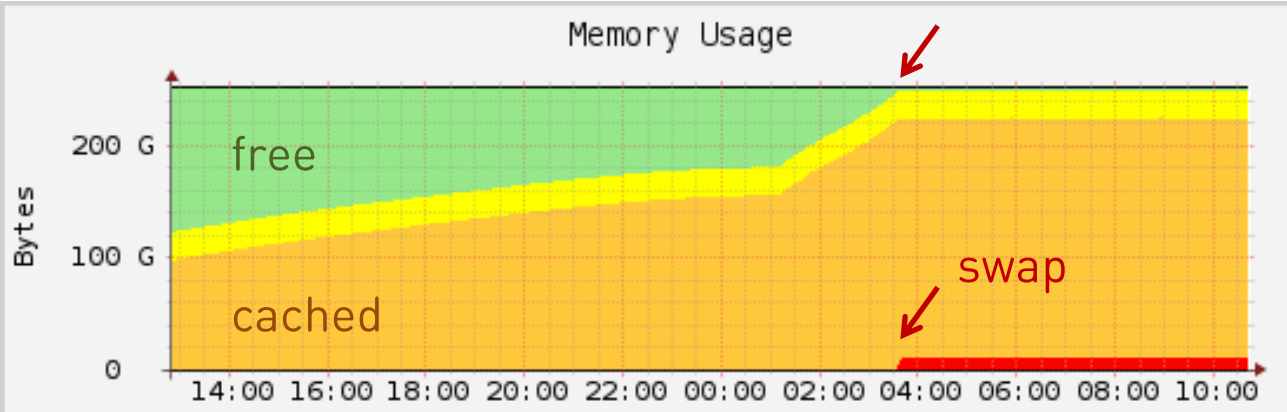
```
./configure --enable-prof

export MALLOC_CONF="prof:true,prof_prefix:jeprof.out,lg_prof_interval:30"

jeprof --svg /path/to/java jeprof.out.* > out.svg
```

Demo

JVM уходит в swap



JVM уходит в swap



- Выключить swap
- -XX:+UseLargePages
- `mlockall()`

  https://github.com/lucidworks/mlockall-agent

Checklist

- MemoryPool: Metaspace, CodeCache и т. д.
- Direct Buffers
  - MaxDirectMemorySize
  - jdk.nio.maxCachedBufferSize
- jemalloc / tcmalloc
- OS memory stats
- NativeMemoryTracking
- async-profiler

andrey.pangin@corp.mail.ru

https://v.ok.ru/vacancies.html