

















■

—

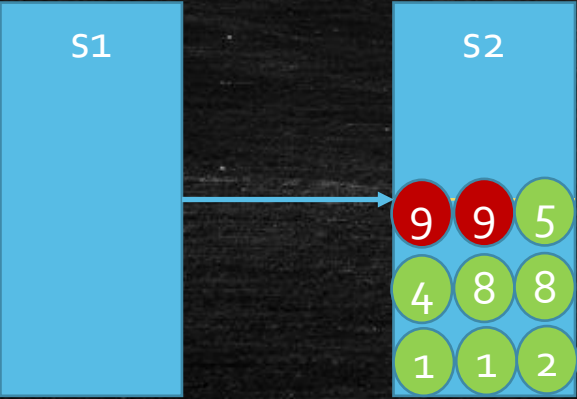
—

—

■

—

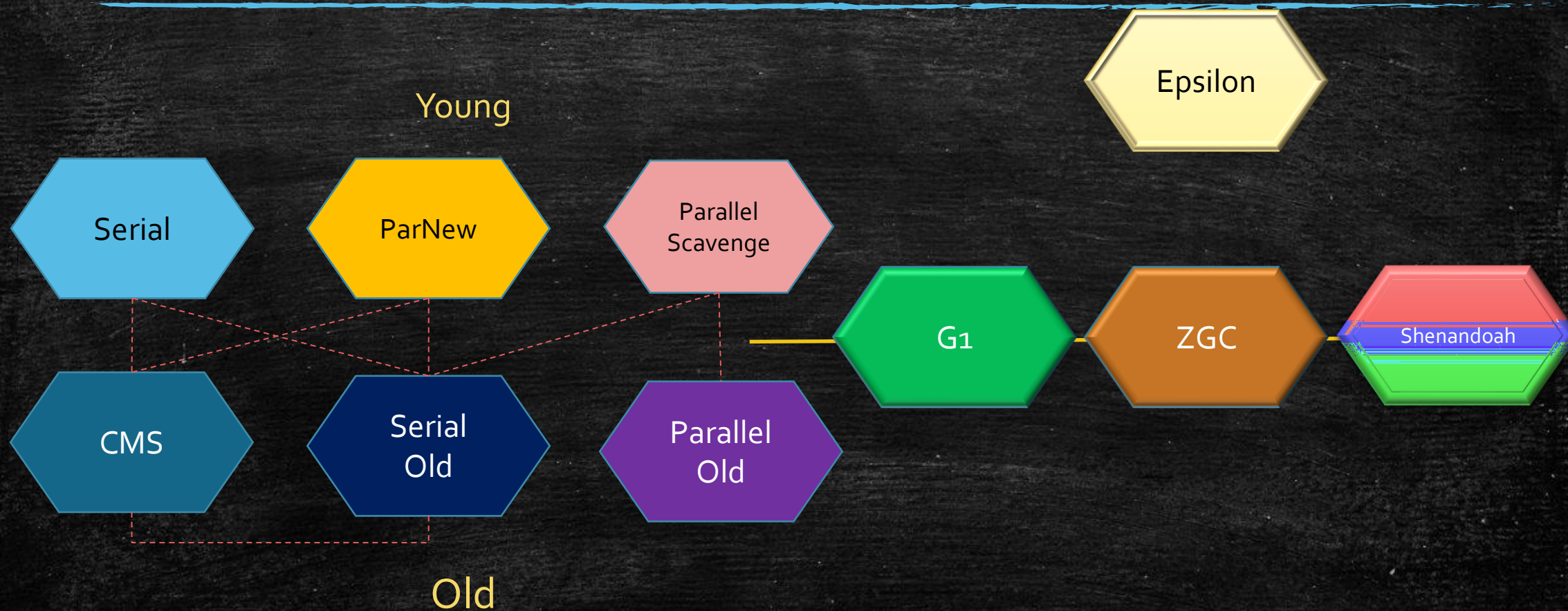
—



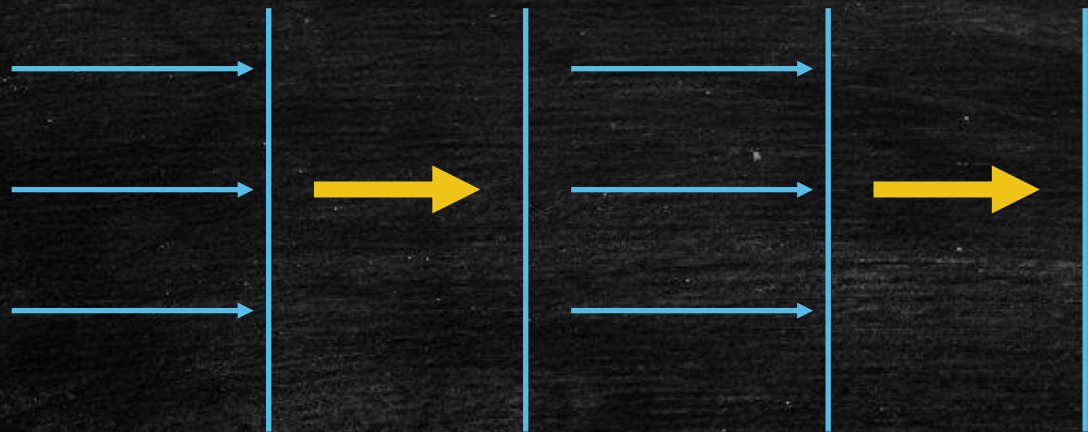






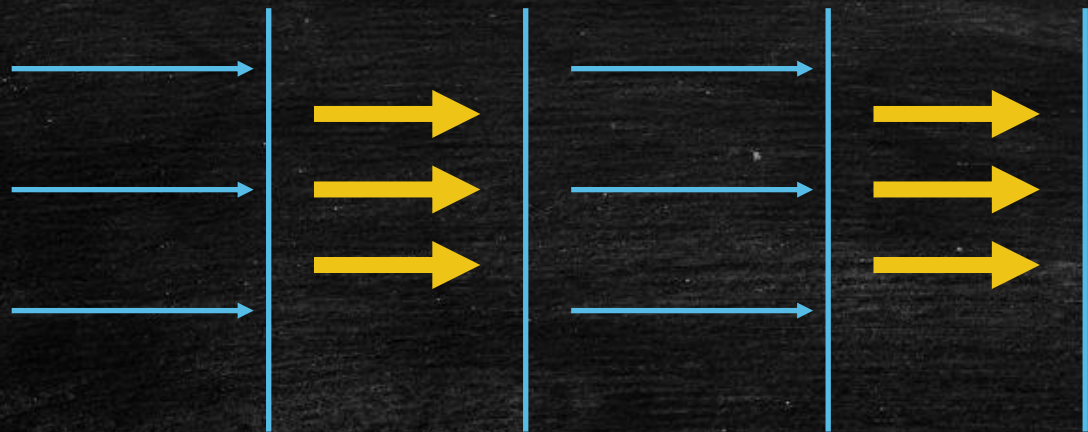




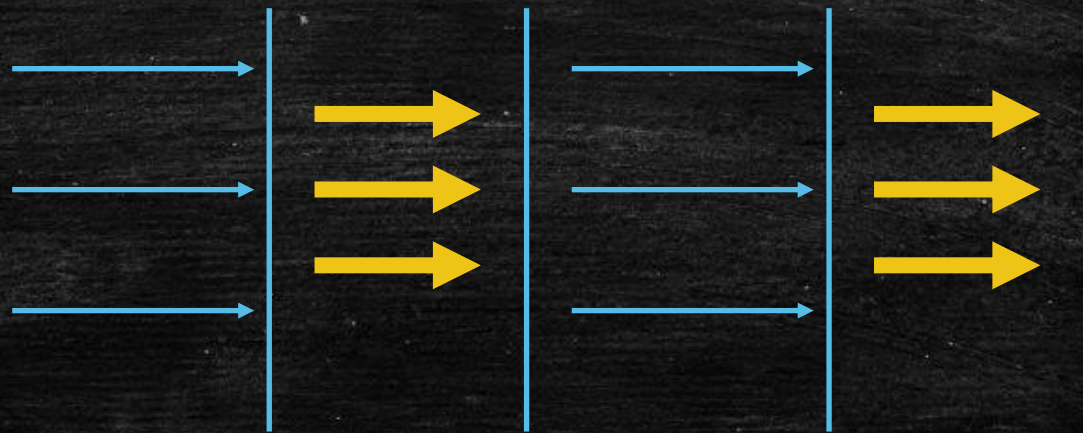


■



















- 
- 
-



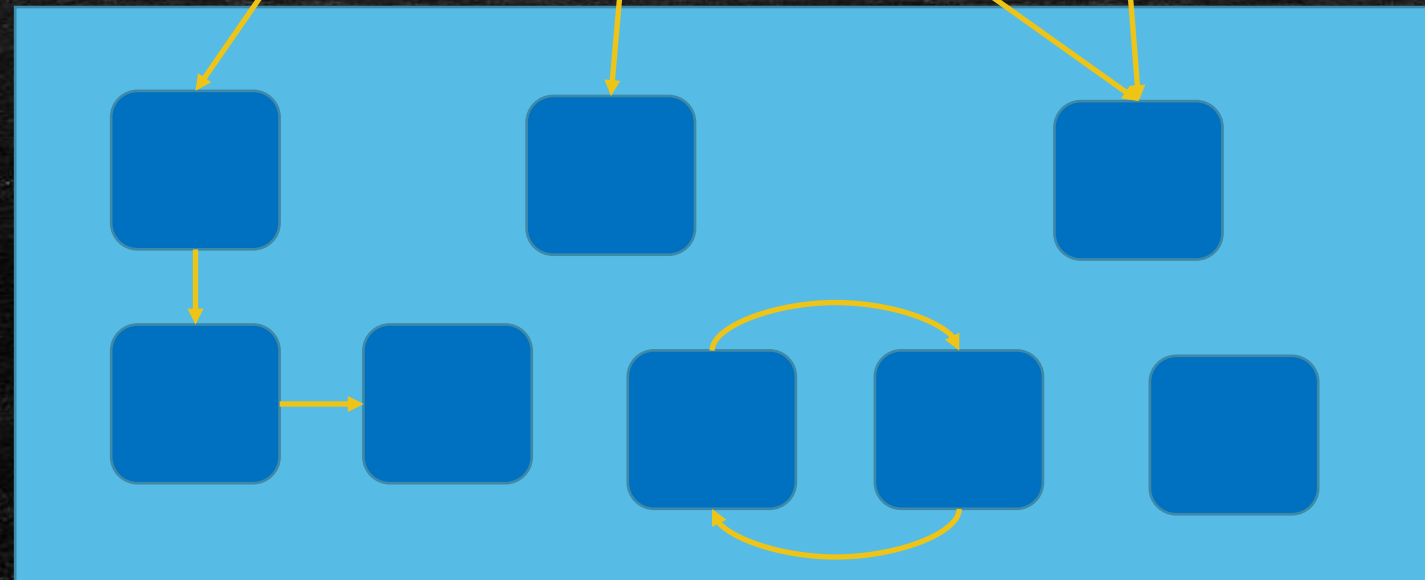
GC roots

线程栈变量

静态变量

常量池

JNI指针





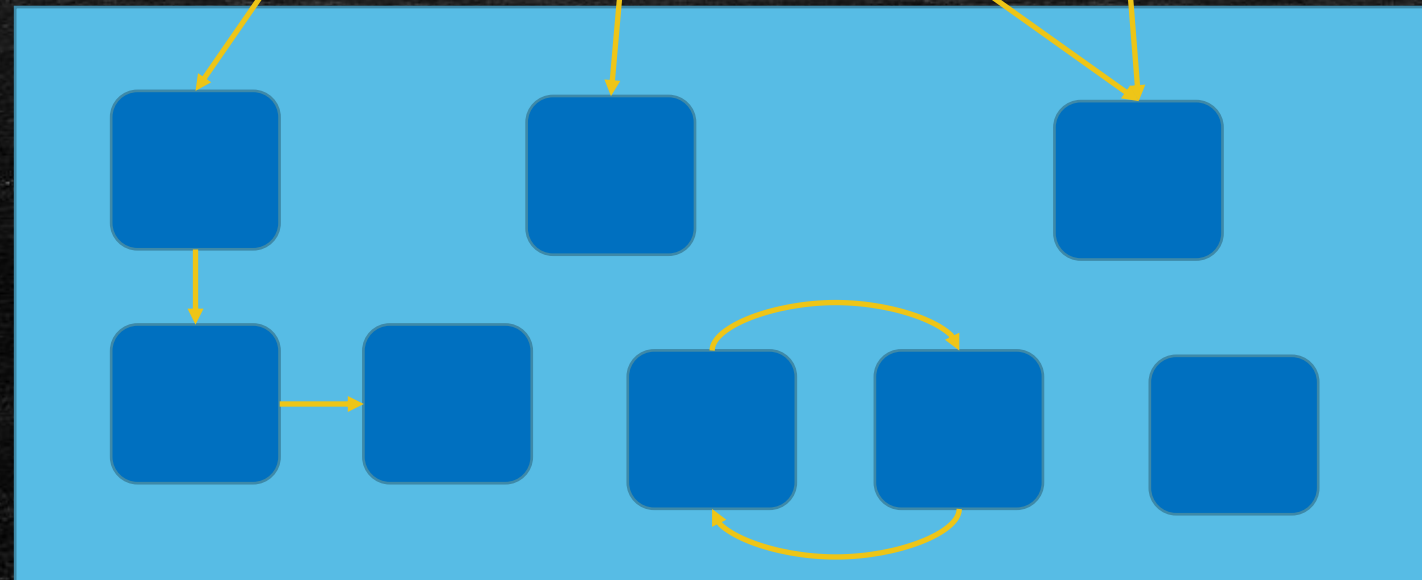
GC roots

线程栈变量

静态变量

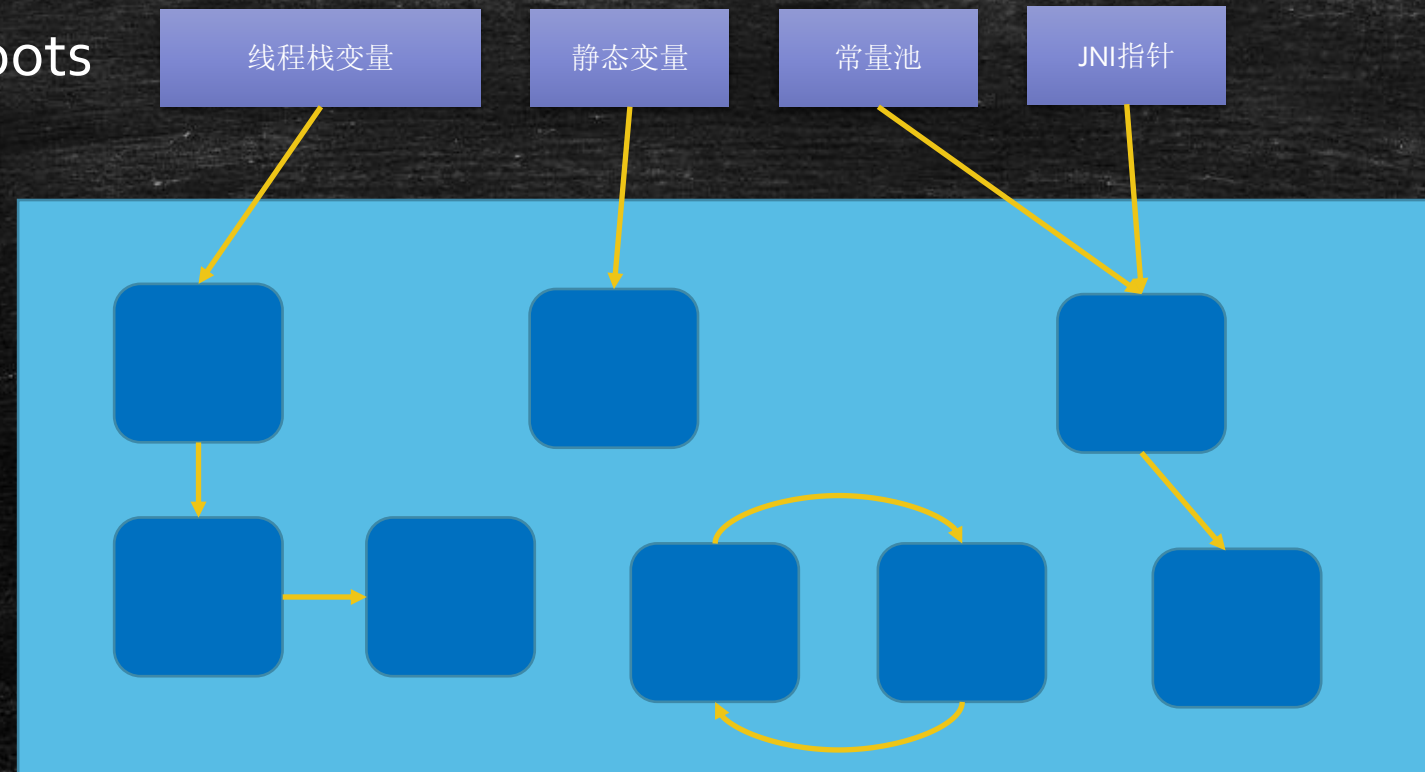
常量池

JNI指针



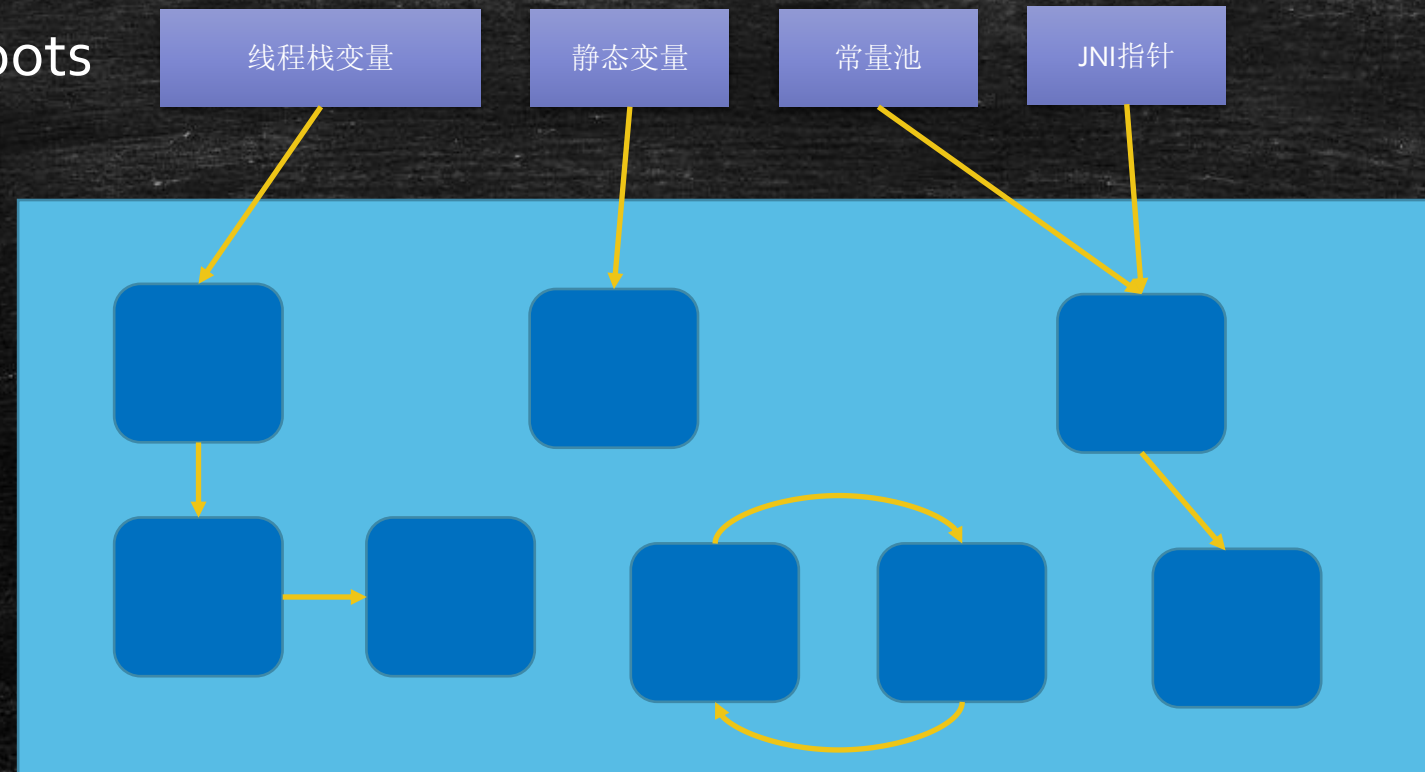


GC roots

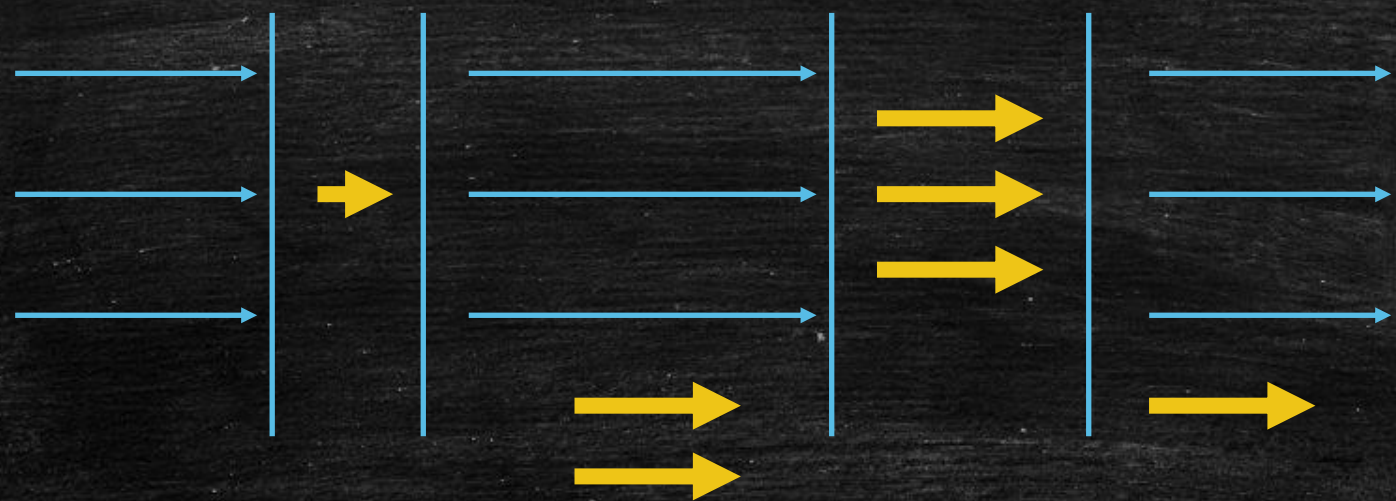




GC roots







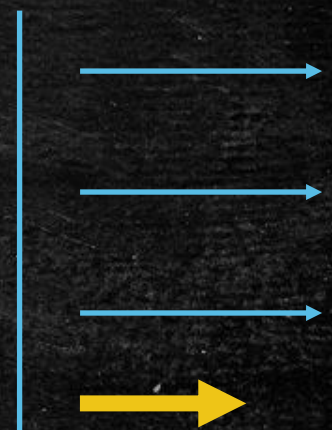
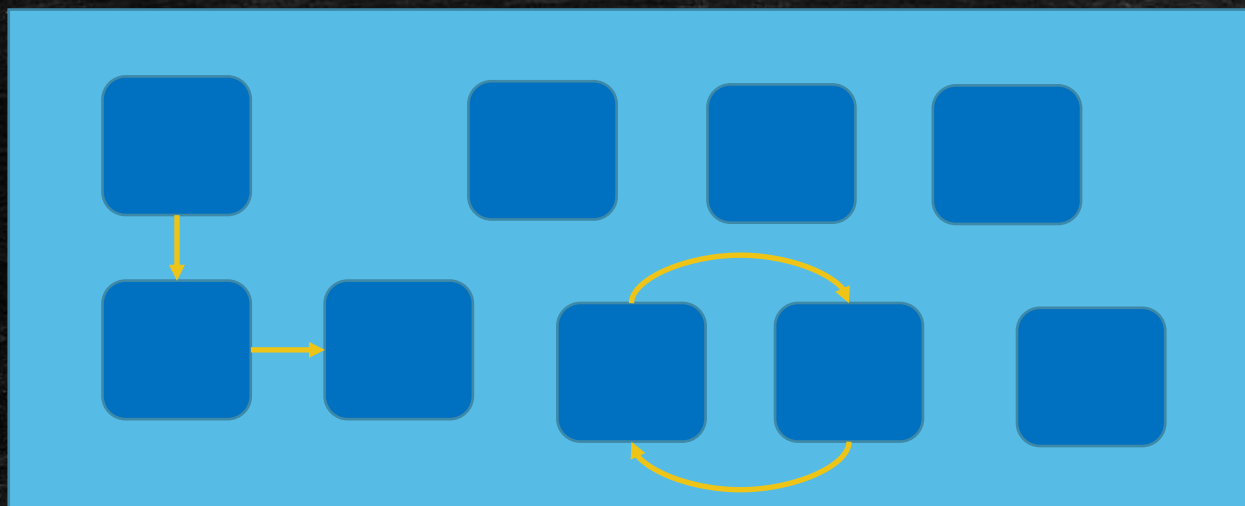
初始标记

并发标记

重新标记

并发清理





并发清理



■

■

■

—

■

■







■

■

—

■

—













■

—

—

■

—

■

—

■

—

■









[https://blogs.oracle.com/  
jonthecollector/our-collectors](https://blogs.oracle.com/jonthecollector/our-collectors)











■  
■  
—  
—  
■  
—



面试：CPU突然飙高如何解决？



## top -Hp 1122

```
Tasks: 61 total, 1 running, 60 sleeping, 0 stopped, 0 zombie
Cpu(s): 43.8%us, 1.8%sy, 0.0%ni, 54.4%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1004412k total, 396844k used, 607568k free, 8496k buffers
Swap: 2047992k total, 0k used, 2047992k free, 61408k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1124	root	20	0	2185m	233m	11m	S	20.6	23.8	0:03.65	java
1172	root	20	0	2185m	233m	11m	S	1.0	23.8	0:03.67	java
1133	root	20	0	2185m	233m	11m	R	0.7	23.8	0:03.52	java
1135	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.65	java
1136	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.63	java
1137	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.59	java
1139	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.65	java
1142	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.58	java
1143	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.57	java
1144	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.57	java
1147	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.62	java
1148	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.61	java
1150	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.56	java
1153	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.57	java
1156	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.67	java
1157	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.68	java
1158	root	20	0	2185m	233m	11m	S	0.7	23.8	0:03.63	java



```
top - 15:46:53 up 19 min,  2 users,  load average: 0.82, 0.40, 0.16
Tasks:  61 total,   1 running,  60 sleeping,   0 stopped,   0 zombie
Cpu(s):100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1004412k total,   397340k used,   607072k free,    8496k buffers
Swap:  2047992k total,        0k used,  2047992k free,   61468k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1124	root	20	0	2185m	234m	11m	R	95.2	23.9	1:01.69	java
1138	root	20	0	2185m	234m	11m	S	0.3	23.9	0:04.07	java
1151	root	20	0	2185m	234m	11m	S	0.3	23.9	0:04.14	java
1155	root	20	0	2185m	234m	11m	S	0.3	23.9	0:04.11	java
1156	root	20	0	2185m	234m	11m	S	0.3	23.9	0:04.25	java
1158	root	20	0	2185m	234m	11m	S	0.3	23.9	0:04.20	java
1162	root	20	0	2185m	234m	11m	S	0.3	23.9	0:04.23	java
1164	root	20	0	2185m	234m	11m	S	0.3	23.9	0:04.16	java
1172	root	20	0	2185m	234m	11m	S	0.3	23.9	0:04.23	java
1173	root	20	0	2185m	234m	11m	S	0.3	23.9	0:04.27	java
1178	root	20	0	2185m	234m	11m	S	0.3	23.9	0:04.16	java
1181	root	20	0	2185m	234m	11m	S	0.3	23.9	0:04.22	java
1122	root	20	0	2185m	234m	11m	S	0.0	23.9	0:00.04	java
1123	root	20	0	2185m	234m	11m	S	0.0	23.9	0:01.56	java
1125	root	20	0	2185m	234m	11m	S	0.0	23.9	0:00.00	java
1126	root	20	0	2185m	234m	11m	S	0.0	23.9	0:00.00	java
1127	root	20	0	2185m	234m	11m	S	0.0	23.9	0:00.00	java



## jstack 1122

```
at java.lang.Object.wait(Native Method)
- waiting on <0x00000000f8ad4378> (a java.lang.ref.ReferenceQueue$Lock)
at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:144)
- locked <0x00000000f8ad4378> (a java.lang.ref.ReferenceQueue$Lock)
at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:165)
at java.lang.ref.Finalizer$FinalizerThread.run(Finalizer.java:216)
```

"Reference Handler" #2 daemon prio=10 os\_prio=0 tid=0x00007faae4075800 nid=0x465 in Object.wait() [0x00007faae9284000]

```
java.lang.Thread.State: WAITING (on object monitor)
at java.lang.Object.wait(Native Method)
- waiting on <0x00000000f8ad4530> (a java.lang.ref.Reference$Lock)
at java.lang.Object.wait(Object.java:502)
at java.lang.ref.Reference.tryHandlePending(Reference.java:191)
- locked <0x00000000f8ad4530> (a java.lang.ref.Reference$Lock)
at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:153)
```

"VM Thread" os\_prio=0 tid=0x00007faae406d800 nid=0x464 runnable → 十进制1122

"VM Periodic Task Thread" os\_prio=0 tid=0x00007faae40cb000 nid=0x46b waiting on condition

JNI global references: 183



吞吐量 = 用户代码执行时间 / (用户代码执行时间 + 垃圾收集执行时间)  
响应时间快 = 用户线程停顿的时间短

确定调优之前，应该确定到底是哪个优先，是计算型任务还是响应型任务



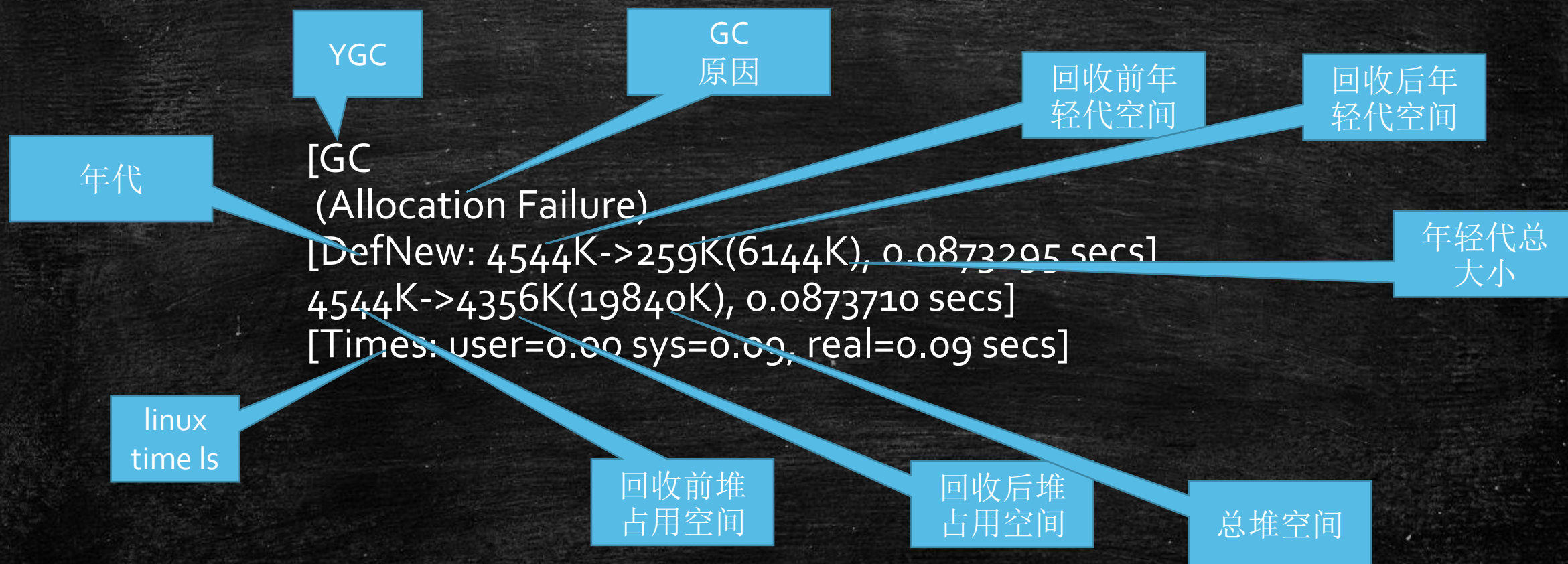








# GC日志详解





## Heap

```
def new generation    total 6144K, used 5504K [0x00000000fec00000, 0x00000000ff2a0000,
0x00000000ff2a0000)
  eden space 5504K, 100% used [0x00000000fec00000, 0x00000000ff160000, 0x00000000ff160000)
  from space 640K,   0% used [0x00000000ff160000, 0x00000000ff160000, 0x00000000ff200000)
  to   space 640K,   0% used [0x00000000ff200000, 0x00000000ff200000, 0x00000000ff2a0000)
tenured generation    total 13696K, used 13312K [0x00000000ff2a0000, 0x0000000100000000,
0x0000000100000000)
  the space 13696K, 97% used [0x00000000ff2a0000, 0x00000000fffa0148, 0x00000000fffa0200,
0x0000000100000000
) Metaspace          used 2538K, capacity 4486K, committed 4864K, reserved 1056768K
  class space        used 275K, capacity 386K, committed 512K, reserved 1048576K
```

已经使用

总容量

虚拟内存  
占用

虚拟内存  
保留



## 案例

有一个50万PV的资料类网站（从磁盘提取文档到内存）原服务器32位，1.5G的堆，用户反馈网站比较缓慢，因此公司决定升级，新的服务器为64位，16G的堆内存，结果用户反馈卡顿十分严重，反而比以前效率更低了

为什么？

如何优化？



案例：  
开源软件Xfire缺陷

[https://blog.csdn.net/qq\\_15037231/article/details/80689905](https://blog.csdn.net/qq_15037231/article/details/80689905)