马士兵教育

ZGC

zero paused GC ?

马士兵

Revision

- 2019年8月2日 – 马士兵

参考资料

- https://wiki.openjdk.java.net/display/zgc/Main

设计目标

- 支持TB级别（4T，据说已经扩展到16T）
- 最大GC停顿10ms
- 内存增大，停顿时间不长
- throughput不超过15%的影响
  - SPECjbb 2015基准测试，128G堆内存，单次GC停顿最大1.68ms, 平均1.09ms

特性

- Colored Pointers
- LoadBarrier
  - 完全不同于CPU中的Load Barrier
  - 指的是从堆中加载对象后，执行一段儿逻辑
- SingleGeneration
- PageAllocation
  - 三种大小不同的动态页面
- Partial Compaction
- Mostly Concurrent

GC设计方法论

1. 是否分代
2. GC算法
    1. throughput
    2. latency
        1. 减少STW时间
            1. 尽量concurrent
            2. 并行标记后的整理阶段通过内存屏障规避STW
                1. 读屏障 ZGC获取堆变量地址的时候
                2. 写屏障 G1并发标记阶段


http://mashibing.com

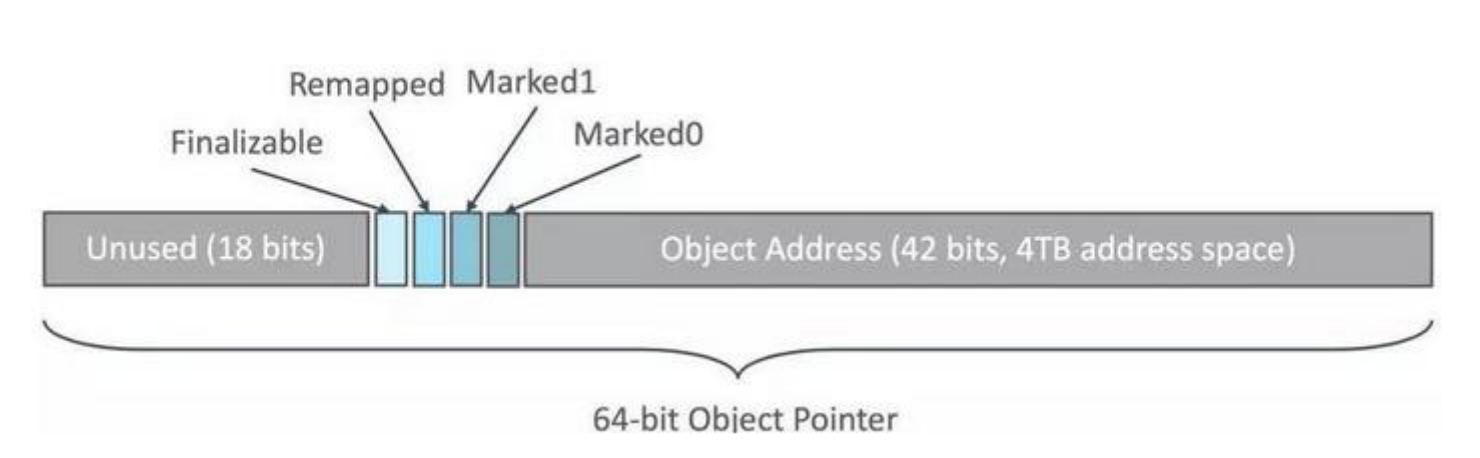
对比

128G 的堆，复合模式下的性能，看 GC 停顿时间：

```
ZGC
                avg: 1.091ms (+/-0.215ms)
    95th percentile: 1.380ms
    99th percentile: 1.512ms
  99.9th percentile: 1.663ms
 99.99th percentile: 1.681ms
                max: 1.681ms


G1
                avg: 156.806ms (+/-71.126ms)
    95th percentile: 316.672ms
    99th percentile: 428.095ms
  99.9th percentile: 543.846ms
 99.99th percentile: 543.846ms
                max: 543.846ms
```

Colored Pointer + Load Barrier



不支持32位，不支持指针压缩
Load Barrier根据指针颜色决定是否做一些事情

region更灵活

2M 32M n*2M
动态分配
256k以下在Small Page
4M以下在medium Page
以上在Large Page

支持NUMA

- non – uniform memory access
- 在距离CPU最近的内存分配和回收，更高效

并行

在ZGC 官网上有介绍，前面基准测试中的32核服务器，128G堆的场景下，它的配置是：

20条ParallelGCThreads，在那三个极短的STW阶段并行的干活 - mark roots， weak root processing（StringTable, JNI Weak Handles,etc）和 relocate roots；

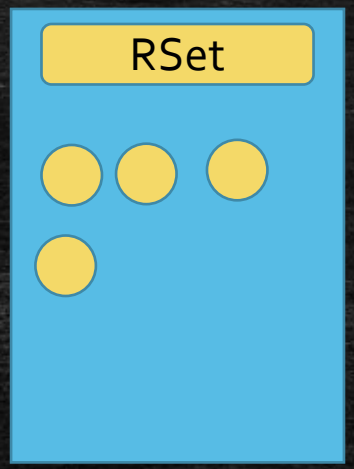
4条ConcGCThreads，在其他阶段与应用并发地干活 - Mark，Process Reference，Relocate。 仅仅四条，高风亮节地尽量不与应用争抢CPU 。

ConcCGCThreads开始时各自忙着自己平均分配下来的Region，如果有线程先忙完了，会尝试“偷”其他线程还没做的Region来干活，非常勤奋。

简单不分代

回收压缩

- 整个region mark – compact
- 而G1是incremental copying collector
  - 不会回收整个Region
  - 通过RSet记录 + 写屏障



没有RSet

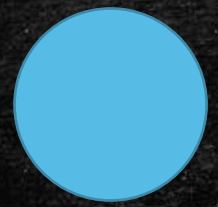
- 没有G1占内存的RememeredSet

zgc phases

1. pause mark start
2. concurrent mark
3. relocate
4. remap

ZGC JDK11引入
-XX:+UseZGC
specjbb zgc测试结果
竟然达到惊人的1ms
b站上的视频

ZGC的并发标记算法



http://mashibing.com