GIAC

全球互联网架构大会

GLOBAL INTERNET ARCHITECTURE CONFERENCE

JDK 10深度解读

杨晓峰，Java平台部门首席工程师，Oracle

日程

- JDK 10一览
- 特性分析
  - 开源的商业特性：AppCDS…
  - 完善的容器支持
  - GC领域改进
  - 本地变量类型推断实践参考

JDK 10

JDK 10：http://openjdk.java.net/projects/jdk/10/

- 第一个半年期版本！
- ~~特性优先~~→ 时间优先
- 12个新特性
- 更多选择：
  - OpenJDK build， GPL V2 with Classpath Exception
  - Oracle JDK

日程

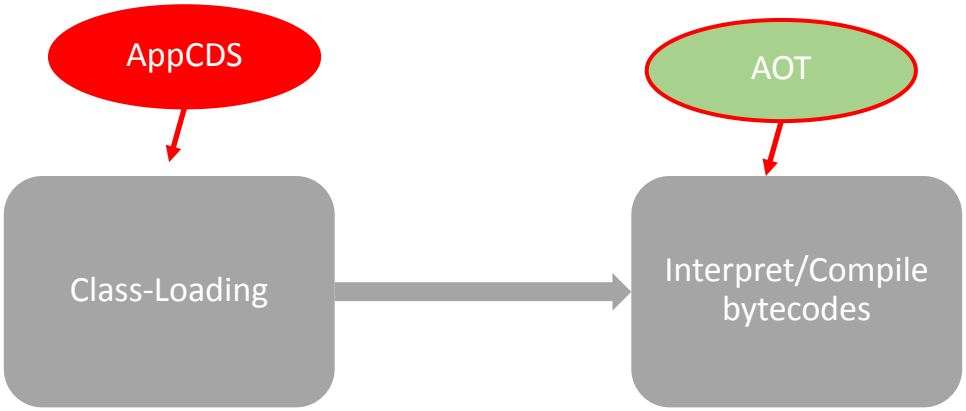
- JDK 10一览
- 特性分析
  - 开源的商业特性：AppCDS…
  - 完善的容器支持
  - GC领域改进
  - 本地变量类型推断实践参考

Application Class-Data Sharing

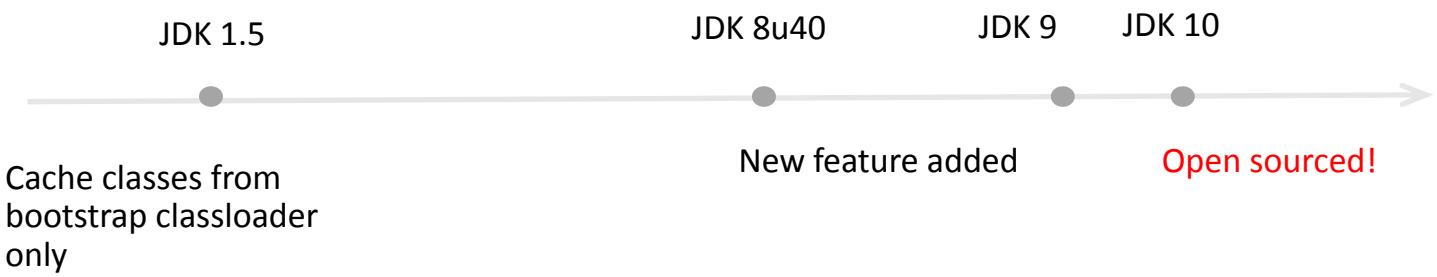
- 将CDS扩展到所有类文件原来只能archive
- 改进启动速度
- 降低Footprint

First Oracle JDK commercial feature Open Sourced !

Java启动为什么更耗时？



CDS和APPCDS演进



Dump阶段

- Classes被JVM解析成为元数据（ metadata ）
- Metaspace被分为 read-only(RO)和read-write (RW)两部分
- 写入缓存文件

```
Java -Xshare:dump -XX:+UseAppCDS -XX:SharedArchiveFile=<jsa> \
      -XX:SharedClassListFile=<classlist> -XX:SharedArchiveConfigFile=<config_file>
```

Java运行时阶段

- 直接将文件内存映射到JVM地址空间
- RO在JVM进程间共享
- RW以 copy-on-write形式共享
- 映射的信息只需要非常少的处理

```
Java -Xshare:dump -XX:+UseAppCDS -XX:SharedArchiveFile=<jsa>
```

启动改善情况

- 简单实施，明显收益

Software	Startup time Reduced	Footprint(memory) Reduced
WebLogic	19 ~ 37%	(average) 5%
Apache Spark with KMeans workload and 20 slaves	11%	(average) 10%

已经或正在开源的Java商业特性或项目

- Graal
- Application Class Data Sharing
- ZGC
- Java Mission Control
- Java Flight Recorder
- Java Usage Tracker
- Infrastructure
- …

真实案例：利用JFR进行线上诊断

- 生产环境出现内存泄漏，运行时开启JFR

$: *jcmd {pid} JFR.start duration=1m filename=/tmp/App.jfr*
Started recording 1. The result will be written to:
/tmp/App.jfr

$: *jcmd {pid} JFR.start*
Started recording 2. No limit (duration/maxsize/maxage) in use.
Use JFR.dump recording=2 filename=FILEPATH to copy recording data to file.
$: *jcmd {pid} JFR.dump recording=2 filename=App.2.jfr*

日程

- JDK 10一览
- 特性分析
  - 开源的商业特性：AppCDS…
  - 完善的容器支持
  - GC领域改进
  - 本地变量类型推断实践参考

JVM + 容器

- Java的优势仍然明显：
  - 硬件和OS中立
  - JVM保证了安全、可靠、兼容性
  - JVM屏蔽了不必要的复杂性，业界证明的可靠性
  - 广泛的生态…
- 承诺将保持Java作为容器第一部署选择

JVM + 容器 - 不足之处

- 容器不是虚拟机
- 容器底层技术对于历史版本Java是陌生的:
  - 文件系统、进程信息等方面微妙差异
  - 新的内存和CPU限制方式
  - 新的应用场景

在容器中使用历史版本JDK

- 目前的常见实践：
- 设置 Xmx

```
# Docker
Docker run …  -m 800m -e JAVA_OPTIONS='-Xmx300m'

# Kubernetes
kubectl run ... --limits='memory=800Mi' --env="JAVA_OPTIONS='-Xmx300m'"
```

- 修正Ergonomics：
  - -XX:MaxRAM=`cat /sys/fs/cgroup/memory/memory.limit_in_bytes`
- Hack系统环境，修改DLL欺骗JVM

JVM容器支持 - **JDK 8u131, JDK 9**

- 内存参数：
  - -XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLimitForHeap
- CPU：
  - 有限支持，JVM能够解析 --cpuset-cpus
  - 如果指定了-XX:ParallelGCThreads -XX:CICompilerCount，优先级更高
- 注意：
  - 参数顺序敏感
  - 仅Linux

JVM容器支持 - **JDK 10**

- JVM自动检测cgroups设置，基本上不需要额外配置

- 也支持明确指定CPU限制，适用于容器和非容器环境：
  -XX:ActiveProcessorCount=N
- 反转设置：
  -XX:-UseContainerSupport
- Deprecate内存参数：
  -XX:+UseCGroupMemoryLimitForHeap

日程

- JDK 10一览
- 特性分析
  - 开源的商业特性：AppCDS…
  - 完善的容器支持
  - GC领域改进
  - 本地变量类型推断实践参考

G1 Parallel Full GC

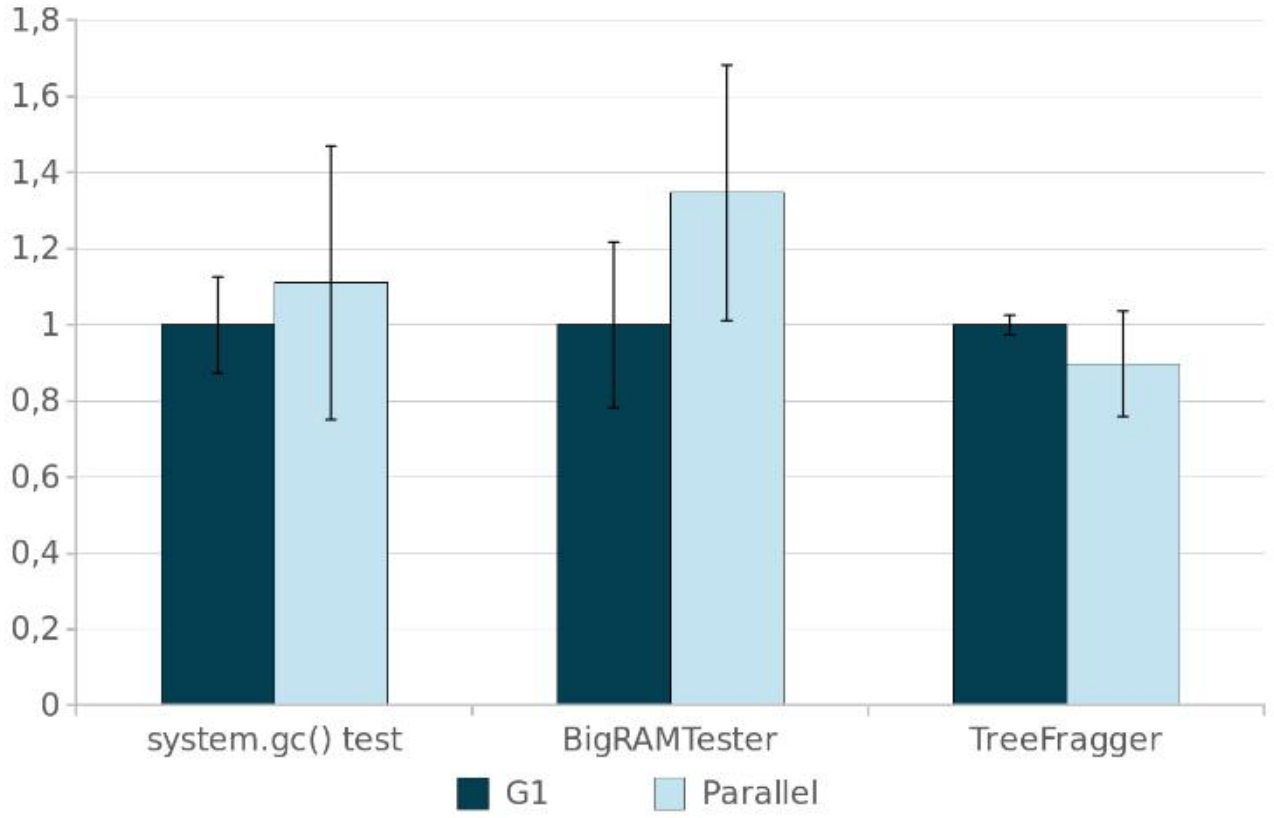
- 目前的G1 Full GC，基于
  - 单线程
  - mark-sweep-compact算法
- 所以，在最坏情况下：
  - 高延迟
  - 低吞吐量
- JDK 10改进的目标：
  - 达到Parallel GC Full GC的表现

G1 Parallel Full GC实现

- 改进为并行Full GC,
  - 线程数与Young GC或者Mixed-GC相同
- 意味着，我们可以设置参数调整并行级别：
  - -XX:ParallelGCThreads
  - 注意，调整它意味着调整 Young和Mixed
  - 可能带来一定程度的空间浪费

改进效果

- 数据源自Thomas Schatzl的演讲
- 从左到右，分别对应不同堆大小和数据特征（live set，小，大，中）：
  - 5G
  - 10G
  - 20G



日程

- JDK 10一览
- 特性分析
  - 开源的商业特性：AppCDS…
  - 完善的容器支持
  - GC领域改进
  - 本地变量类型推断实践参考

本地变量类型推断

- 原则：
  - 阅读代码比写代码重要
  - 代码在有限范围内可理解很重要
  - 可读性不要指望IDE
- 使用得当：
  - 减少工作量
  - 突出重要信息
- 但是，滥用也会导致很差的代码

类型推断建议(1)

- 变量名尽量直观有意义

```
// Good
var custList = dbconn.executeQuery(query)

// Not-suggested
var c = dbconn.executeQuery(query)
```

类型推断建议(2)

- 尽量限制var变量作用范围

```
// Not-suggested
var items = new HashSet<Item>(...);

// 100行其他逻辑
…

items.add(MUST_BE_PROCESSED_LAST);
for (var item : items) ...
```

类型推断建议(3)

- 避免类型争议

```
// Good
var itemQueue = new PriorityQueue<Item>();

// Not-suggested
var itemQueue = new PriorityQueue<>();
var list = List.of();
// 上面的代码其实是被推断为 PriorityQueue<Object>， List<Object>

// OK → 能够推断出正确类型，但可读性一般
Comparator<String> comp = ... ;
var itemQueue = new PriorityQueue<>(comp);
var list = List.of(BigInteger.ZERO);
```

类型推断建议(4)

- 小心处理literal

```
// Good for boolean, character, long, and string
// ORIGINAL
boolean ready = true;
char ch = '\ufffd';
long sum = 0L;
String label = "wombat";

// GOOD
var ready = true;
var ch    = '\ufffd';
var sum   = 0L;
var label = "wombat";
```

类型推断建议(5)

- 各种数字类型最容易混淆

```
// ORIGINAL
byte flags = 0;
short mask = 0x7fff;
long base = 17;

// 危险：全部被推断为int
var flags = 0;
var mask = 0x7fff;
var base = 17;
```

GIAC 全球互联网架构大会
GLOBAL INTERNET ARCHITECTURE CONFERENCE

关注公众号获得
更多案例实践