Alibaba Cloud | Worldwide Cloud Services Partner

JDK11 专题

JVM团队

2020-9-27

- JDK11 新技术
- AJDK11/DragonWell11专有特性
- 升级JDK11注意事项

Java 发展历史

- Java 8
  - 2019 年1 月以后，Oracle 不会将 Java SE 8 的更多更新发布到其公共下载站点用于商业用途
  - 2020 年 12 月之后，不再为个人桌面用户提供 Oracle JDK 8 的修复更新
- Java 11
  - 2018 年 9 月 25 日正式发布
  - 长期支持服务（LTS, Long-Term-Support）
  - 提供技术支持直至 2023 年 9 月
  - 对应的补丁和安全警告等支持将持续至 2026 年

Java 8 -> Java11 变化

- Java 9
  - Platform Module System
  - Interface Private Methods
  - 改进版 Try-With Resources
  - G1 as default GC
- Java 10
  - Application Data-class Sharing
  - Parallel FullGC for G1
  - 局部变量类型推断
  - 线程-局部管控
  - 基于Java的实验性JIT编译器Graal

Java 8 -> Java11 变化

- Java 11
  - 简单启动单个源代码文件的方法
  - 用于Lambda参数的局部变量语法
  - 支持TLS 1.3 协议
  - Epsilon：低开销垃圾回收器
  - ZGC：可伸缩低延迟垃圾收集器
  - 飞行记录器：JFR
  - 低开销的Heap Profiling
  - AArch64 master支持

Java 平台模块系统

- Project Jigsaw – 将模块化开发引入Java平台
- JDK 被重新组织成 94 个模块
- 新增 jlink 工具，创建出只包含所依赖的 JDK 模块的自定义运行时镜像
- 重要特征：描述模块的module-info.class <- module-info.java
  - 模块导出的包
  - 模块的依赖关系
  - 服务的提供和使用

语言特性

- 集合、Stream 和 Optional – Java 9
  - 集合增加 List.of、Set.of、Map.of 和 M ap.ofEntries等工厂方法来创建不可变集合
  - Stream新增方法 ofNullable、dropWhile、takeWhile 和 iterate
  - Optional类新增ifPresentOrElse、or、Stream等
- 局部变量类型推断
  - Java 10 中引入
    - var list = new ArrayList<String>(); // ArrayList<String>
    - var stream = list.stream(); // Stream<String>
  - Java 11 支持Lambda 表达式中使用 var 进行参数声明，
    - 局部变量和 Lambda 表达式的用法进行了统一，并且可以将注释应用于局部变量和 Lambda 表达式

GC

- G1 优化
  - G1 垃圾回收器是 Java 9 中 Hotspot 的默认垃圾回收器
  - Java 10 引入多线程并行 GC，同时使用与年轻代回收和混合回收相同的并行工作线程数量，从而减少了 Full GC 的发生算法

- ZGC 即Z Garbage Collector
  - 可伸缩的、低延迟的垃圾收集器，主要为了满足如下目标进行设计
    - GC 停顿时间不超过 10ms
    - 即能处理几百 MB 的小堆，也能处理几个 TB 的大堆
    - 应用吞吐能力不会下降超过 15%（与 G1 回收算法相比）
    - 方便在此基础上引入新的 GC 特性和利用 colord
    - 针以及 Load barriers 优化奠定基础

ZGC概览

- 传统GC的问题：暂停时间长

CMS

问题：碎片化严重导致FullGC
（并发标记清除，不整理）

G1

问题：整理需暂停
（支持整理）

ZGC概览

ZGC

(Almost) pauseless GC

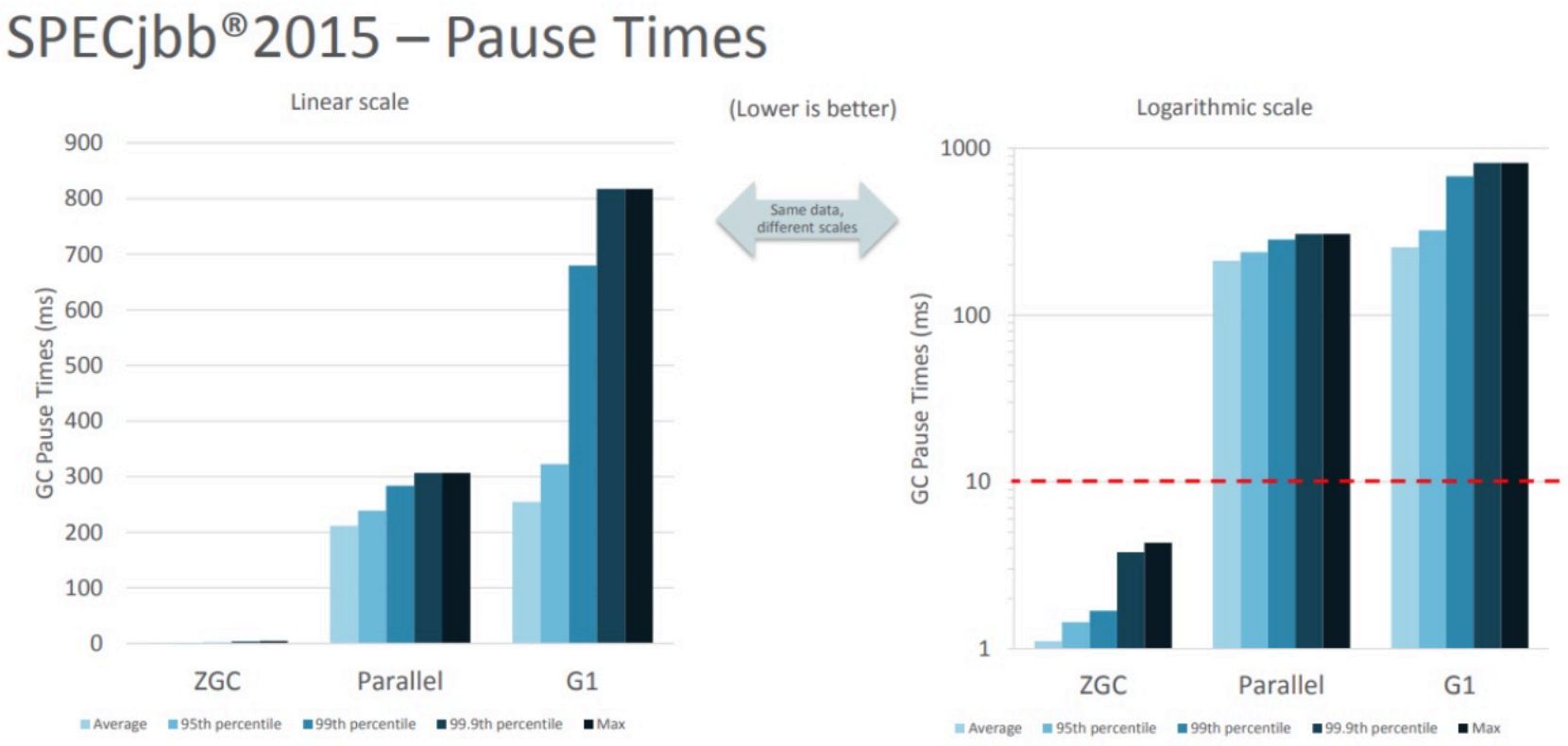
支持并发整理

（绝大部分任务都可以并发完成）

- JDK11 (experimental)
- JDK15 (product)

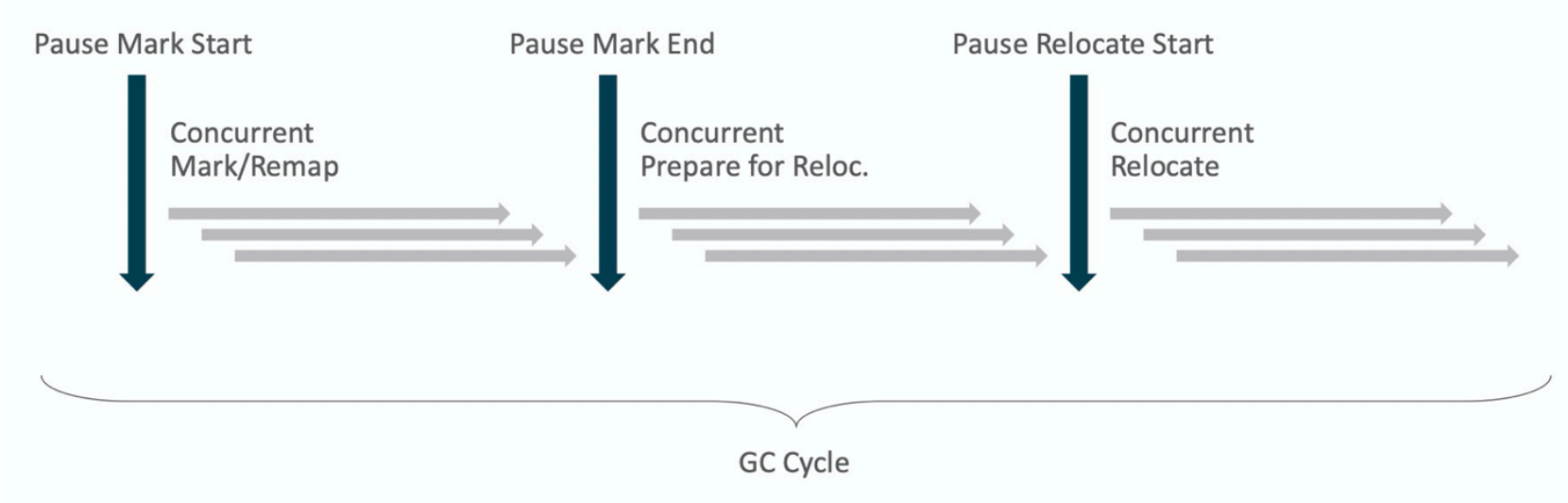
- 暂停：10ms以内
- 支持大堆：4TB (16TB, JDK13+)
- 负面：吞吐量下降15%以内（官方）

ZGC 回收算法停顿时间对比



ZGC概览

- 集团某应用压力高峰的ZGC数据
  - GC Cycle (280ms), 每5.5s一次 （占5%）
  - 暂停1 (1.5ms), 暂停2 (0.5ms), 暂停3 (1ms)
  - 并发1 (200ms), 并发2 (26ms), 并发3(50ms)



Runtime

- AppCDS 应用程序类数据共享 – Java 10
  - Java 5 引入了类数据共享机制 (Class Data Sharing，简称 CDS)，允许将一组类预处理为共享归档文件
  - JDK10 在原来的 bootstrap 类基础之上，扩展加入了应用类的 CDS (Application Class-Data Sharing) 支持
- 线程-局部管控
  - Java 10 中引入 JVM 安全点的概念，将允许在不运行全局 JVM 安全点的情况下实现线程回调，由线程本身或者 JVM 线程来执行，同时保持线程处于阻塞状态
- 统一 JVM 日志
  - 选项-Xlog 来控制 JVM 上 所有组件的日志记录

为什么需要AppCDS

- Java的类加载机制
  - 应用启动时会加载大量的类（启动慢）
  - 类加载需要占用一定的内存
- AppCDS可以帮助解决这些问题
  - 应用只需启动一次，dump可以Java进程共享使用
  - 多个Java进程可以共享AppCDS mmap的内存

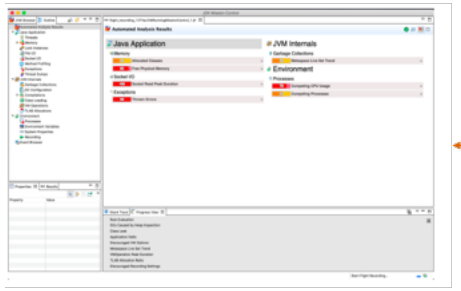
AppCDS工作方式

- 共享文件在运行时被memory mapped
  - JVM在类加载时会先从mapped数据里面查找类，而不是去jar包里面去searching/reading/parsing
  - 减少启动时间
- read only的区域可以被多个java进程共享
  - 减少footprint

Tools

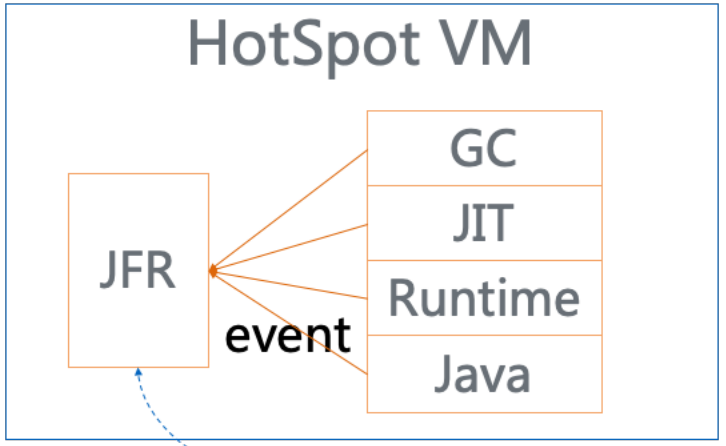
- JFR 飞行记录器
  - Java飞行记录仪：轻量级的事件采集
  - 之前是商业版 JDK 的一项分析工具，在 Java 11 中，其代码被包含到公开代码库中
  - 低开销的事件信息收集框架，用于对应用程序和 JVM 进行故障检查、分析。
  - 飞行记录器记录的主要数据源于应用程序、JVM 和 OS，这些事件信息保存在单独的事件记录文件中，故障发生后，能够从事件记录文件中提取出有用信息对故障进行分析

性能诊断技术：JFR



JMC

.jfr



jcmd

```
    // depends on the InitializeNode
    _igvn.replace_node(init_ctrl, ctrl);
    _igvn.replace_node(init_mem, mem);
  }
}

if (EnableJFR && JfrOptionSet::sample_object_allocations()) {
  jfr_sample_fast_object_allocation(alloc, fast_oop, fast_oop_
}

if (C->env()->dtrace_extended_probes()) {
  // Slow-path call
  int size = TypeFunc::Parms + 2;
  CallLeafNode *call = new (C) CallLeafNode(OptoRuntime::dtrac
                                            CAST_FROM_FN_PTR(
                                            "dtrace_object_all
```

Tools

- 低开销的 Heap Profiling
  - Java 11 中提供一种低开销的 Java 堆分配采样方法，能够得到堆分配的 Java 对象信息，并且能够通过 JVMTI 访问堆信息
  - 开销足够低，可以默认一直开启
  - 对所有堆分配区域进行采样
  - 给出正在和未被使用的Java对象信息
  - 帮助客户定位高内存分配的确切位置

参考文档

- Java9 新特性介绍
  - https://www.ibm.com/developerworks/cn/java/the-new-features-of-Java-9/index.html
- Java10 新特性介绍
  - https://www.ibm.com/developerworks/cn/java/the-new-features-of-Java-10/index.html
- Java11新特性介绍
  - https://www.ibm.com/developerworks/cn/java/the-new-features-of-Java-11/index.html

AJDK11 专有特性 （一）

- EagerAppCDS
  - 针对阿里的应用场景，对AppCDS进行优化
  - Pandora容器启动时间从9秒 -> 2秒启动画像
- JWarmup2
  - 在JWarmup1基础上开发
  - 使用更便捷，性能更优

AJDK11 专有特性 （二）

- AppAOT
  - AOT(ahead of time)运行前提前将部分代码编译
  - AppAOT增加动态加载AOT library的功能，可以动态加载/卸载
- VectorAPI
  - Vector API是OpenJDK project Panama的一个重要组成部分
  - 目标：让更灵活调用CPU强大的SIMD指令，让一条指令处理多条数据，从而获得成倍的性能提升
  - 在大数据，AI计算和多媒体处理等诸多方面有非常广阔的应用前景
- ZGC
  - 解决最大RT过大的问题
  - Back port 上游的优化

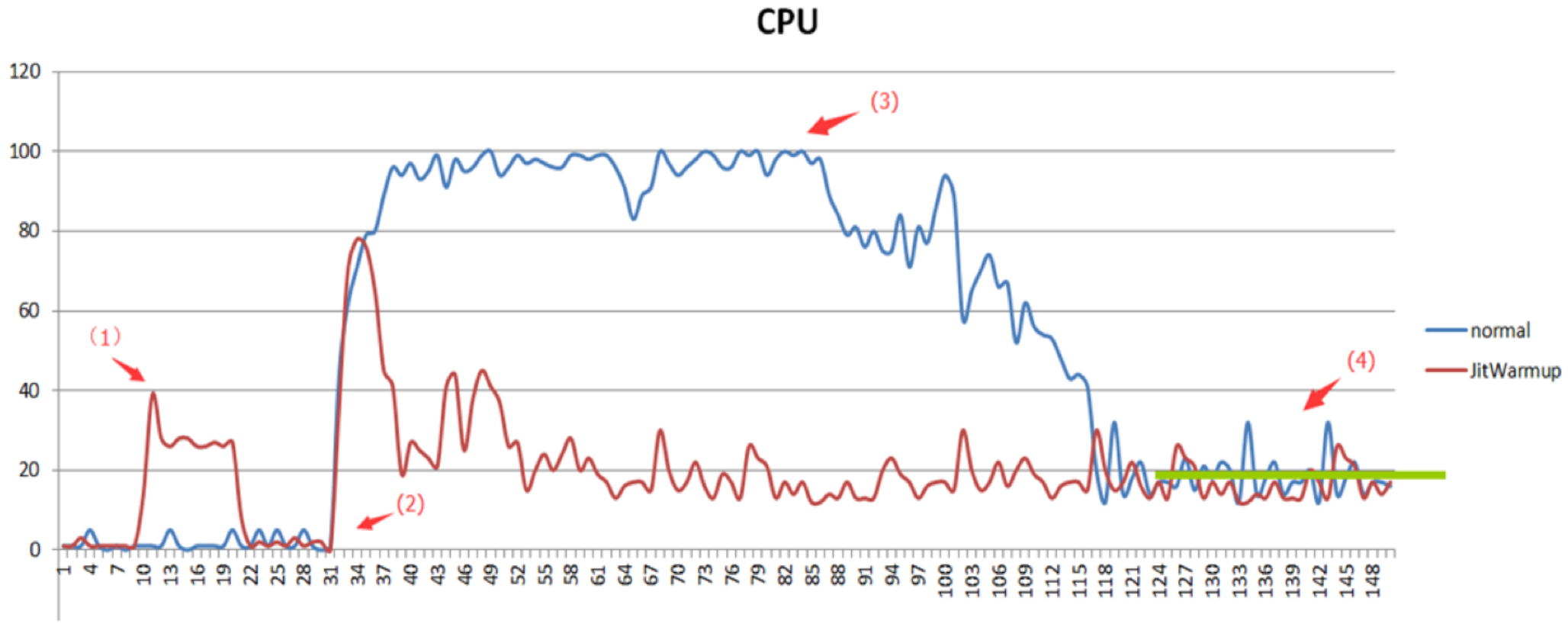
AJDK11 专有特性（三）

- ElasticHeap
  - 结合社区Elastic技术开发
- SGX 支持
  - 提供特定版本的JDK11支持
  - 结合occlum,支持Java应用在可信执行环境下运行
- RDMA支持
  - 基于JEP337: RDMA Network Sockets
  - 增强JDK 网络API，来支持remote direct memory access(RDMA)

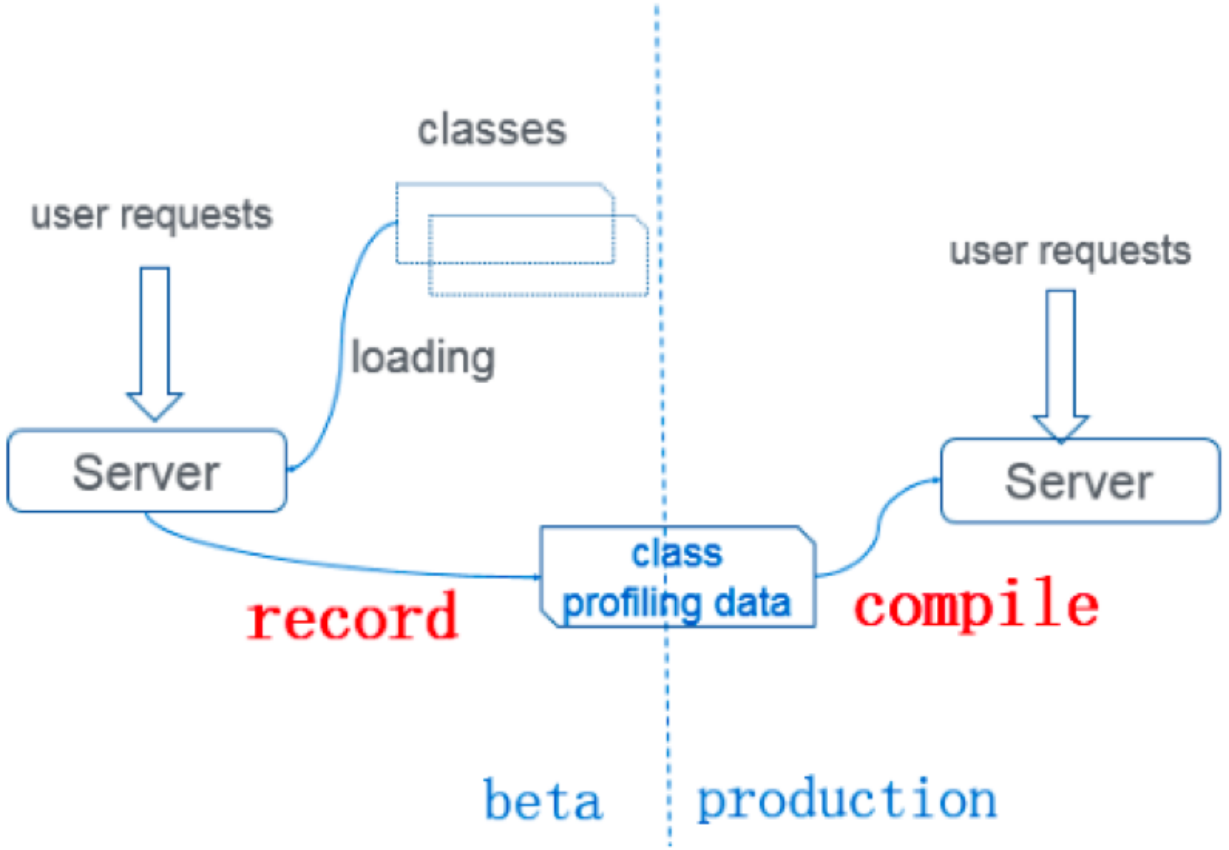
EagerAppCDS 介绍

- 目标： 优化通过customer class loader加载类的过程
- 方案：
  - 重点customer class loader 注册
  - 流程优化，实现类似于App class loader
  - Not found class 优化
  - 流程简化：将三步简化为两步，协助用户在线上部署

代码预热技术： JWarmup2



代码预热技术：JWarmup2



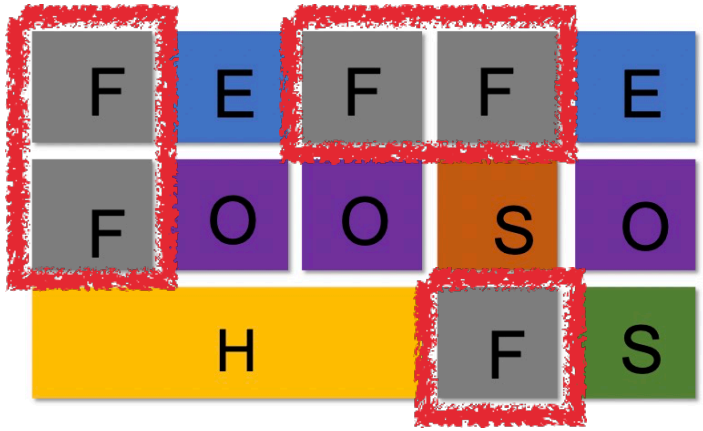
ElasticHeap的实现

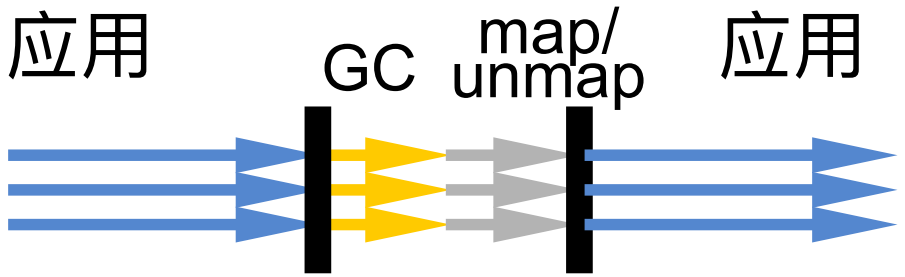
- 为什么JVM一般不归还物理内存?
  - 保证应用的吞吐量和响应时间
  - 避免耗时操作，减少暂停时间
    - map/unmap速度慢
    - 用户线程可能发生page fault

- ElasticHeap解决了上述问题

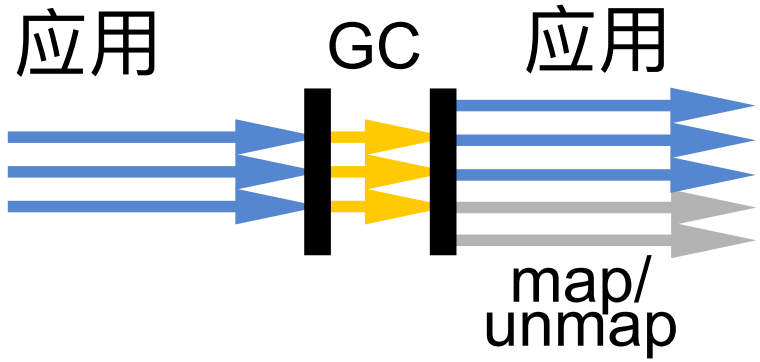
ElasticHeap的实现

- 关键步骤：并发地归还内存
  - 无需暂停就能执行map/unmap操作





JDK12/13的处理方式



ElasticHeap的处理方式

JDK11 升级注意事项

- 常用软件的升级：
  - **IntelliJ IDEA**: 2018.2
  - **Eclipse**: Photon 4.9RC2 with Java 11 plugin
  - **Maven**: 3.5.0
    - **compiler plugin**: 3.8.0
    - **Surefire** and **failsafe**: 2.22.0
  - **Gradle**: 5.0

编译阶段问题

- 模块化相关问题
  - 模块内的类不允许被模块外访问
  - --add-exports module/package=$readingmodule
- JDK文件夹层次变化
  - 取消jre文件夹
  - 删除类tools.jar 和 rt.jar
- 被移除的package/方法
- JavaEE相关模块被移除

JDK11 升级指南

- JDK11 升级：https://yuque.antfin.com/aone355606/gfqllg/ef23u6

Alibaba Cloud | Worldwide Cloud Services Partner

谢　谢