BiSheng JDK 24年规划

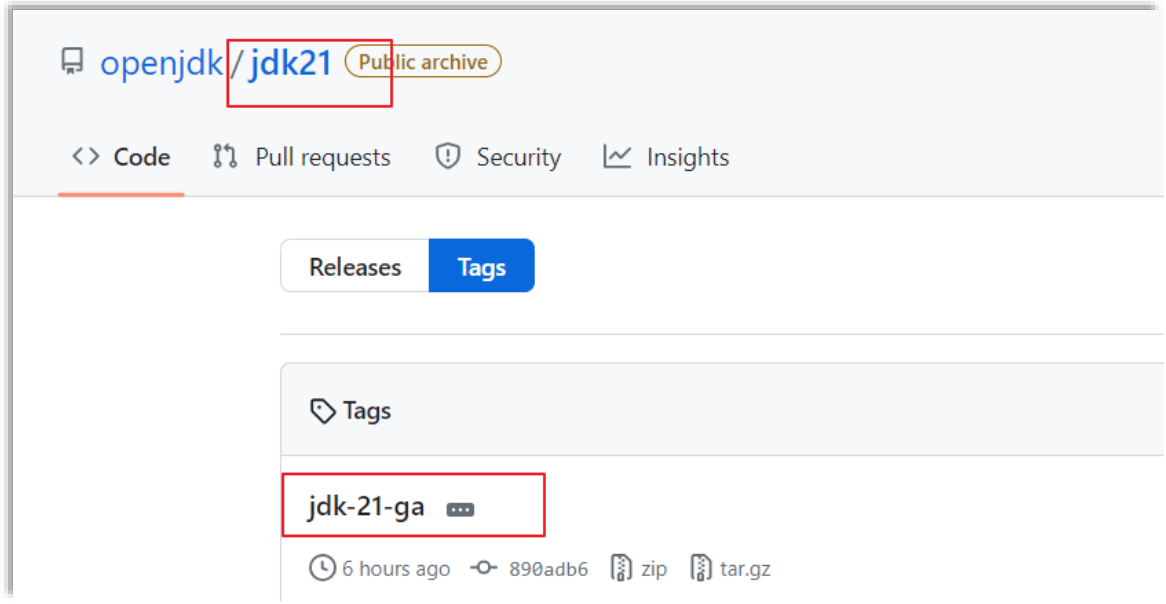
华为技术有限公司

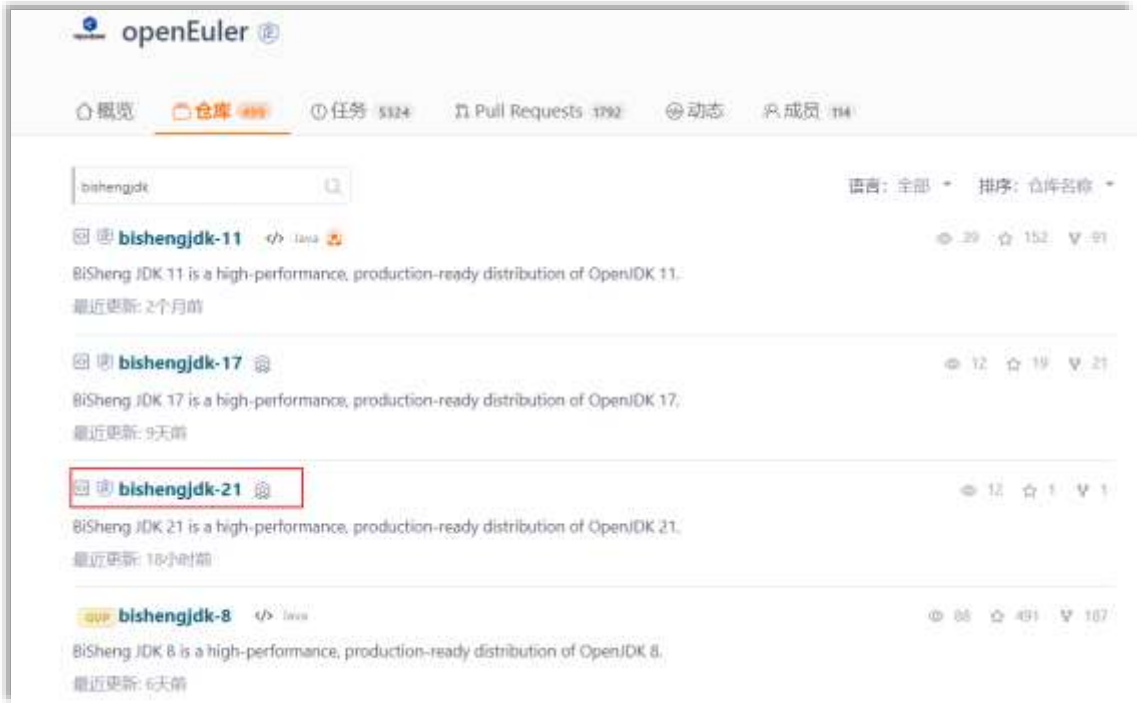
Compiler SIG Maintainer 周磊

目录

毕昇JDK（LTS）生命周期

OpenEuler Compiler SIG



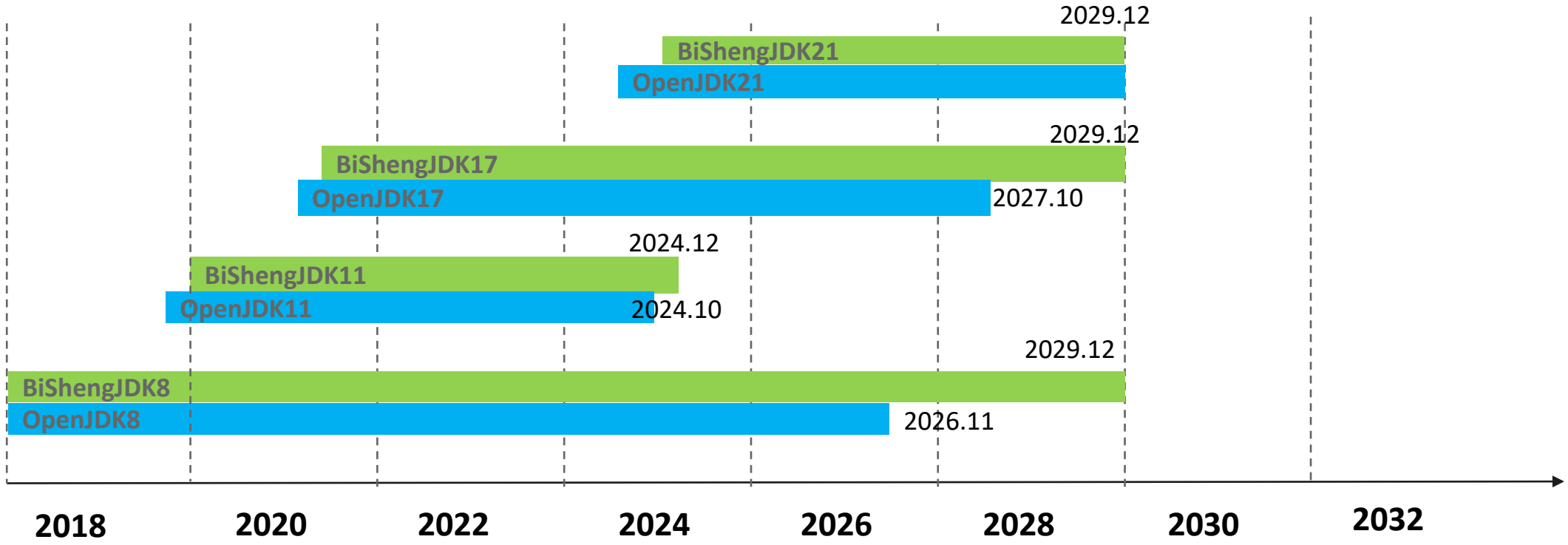


https://github.com/openjdk/jdk21/

https://gitee.com/organizations/openeuler/projects

JDK（LTS） LifeCycle

OpenEuler Compiler SIG



https://access.redhat.com/articles/1299013

https://gitee.com/openeuler/bishengjdk-8/wikis/%E4%B8%AD%E6%96%87%E6%96%87%E6%A1%A3/LifeCycle?sort_id=4848276

24年新特性规划

OpenEuler Compiler SIG

1、**JVM启动优化：**Aggressive CDS、Class Loader相关优化

2、**云原生分布式JIT编译探索：**提升启动速度，降低集群功耗

3、**安全隐私：**java堆栈数据Dump匿名化

4、**国密TLS1.3：**使能鲲鹏KAE

5、**发布JDK21（LTS）到鲲鹏社区**

启动优化 - Aggressive CDS

OpenEuler Compiler SIG

基于Dynamic CDS，但跳过更多步骤，进一步提升类加载速度

- 使用CDS加载类时，仍然会遍历所有classpath找到类的byte code，然后与CDS进行CRC校对
- 安全的同时，牺牲了不少性能
- 默认CDS archive可信，则可以跳过CRC校验，即无需读取byte code、也无需遍历classpath

对java agent的支持

- 开启agent时会将类的原始字节码传给agent修改，因此难以跳过“读取byte code”的步骤
- 大部分经agent注入后的类的字节码，在多次运行下均相同
- dump模式时收集原始类字节码 和注入后字节码的crc，一并dump入archive中
- load模式下进行agent注入，若注入后通过一致性校验则直接使用缓存加速

Aggressive CDS类加载过程

分布式JIT 背景 - 云原生

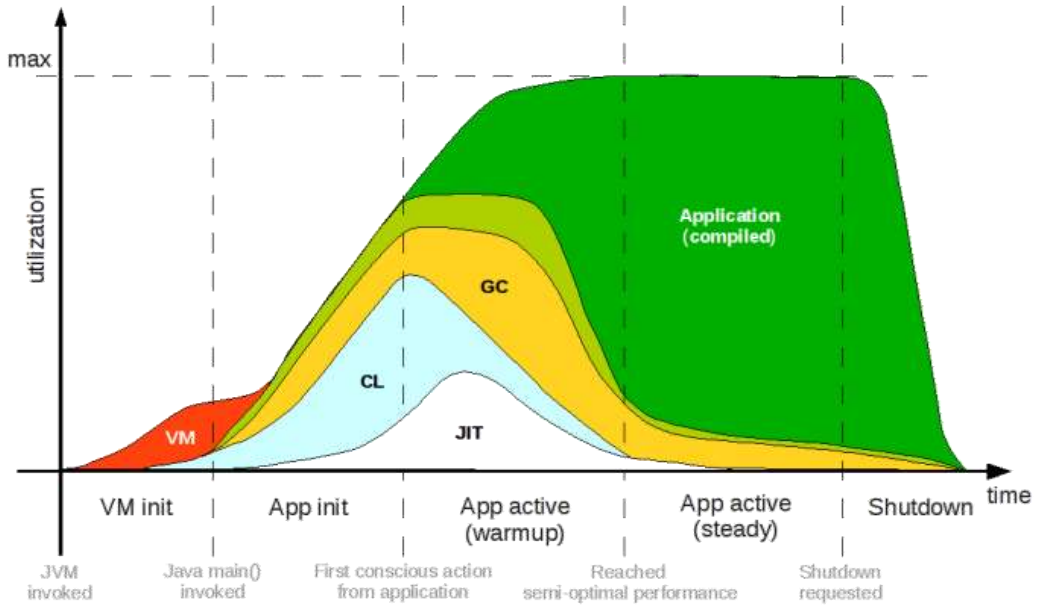
OpenEuler Compiler SIG

随着云计算的发展，越来越多的应用被部署在了云场景之下

- 云上单个实例分配的资源较少（CPU、内存等）
- 云上实例的资源分配十分灵活与精确，例如分配CPU可以精确到0.001核
- 为了节约成本，用户会选择尽可能缩减资源分配量
- 云应用可能会较为频繁地弹性伸缩
- 云应用的实例数量会随着业务压力的变化实时增加或减少
- 增加新实例时，程序会有个启动过程，程序启动完成才能接收服务请求

在云场景下，传统VM应用面临着较为严重的启动性能问题

- 传统VM如JVM，最初是面向长时间运行的应用所设计的，且占用资源较大
- JVM的类加载、预热、编译、对各种动态特性的支持等过程耗时长、资源消耗多
- 使用业界主流开发框架如Spring Boot开发的程序冷启动时间普遍偏长
- 框架本身就有很多模块需要初始化，且广泛使用了JVM提供的各种动态特性
- 业界常用Spring示例应用PetClinic在1.5核容器下需要15秒左右才能启动完毕



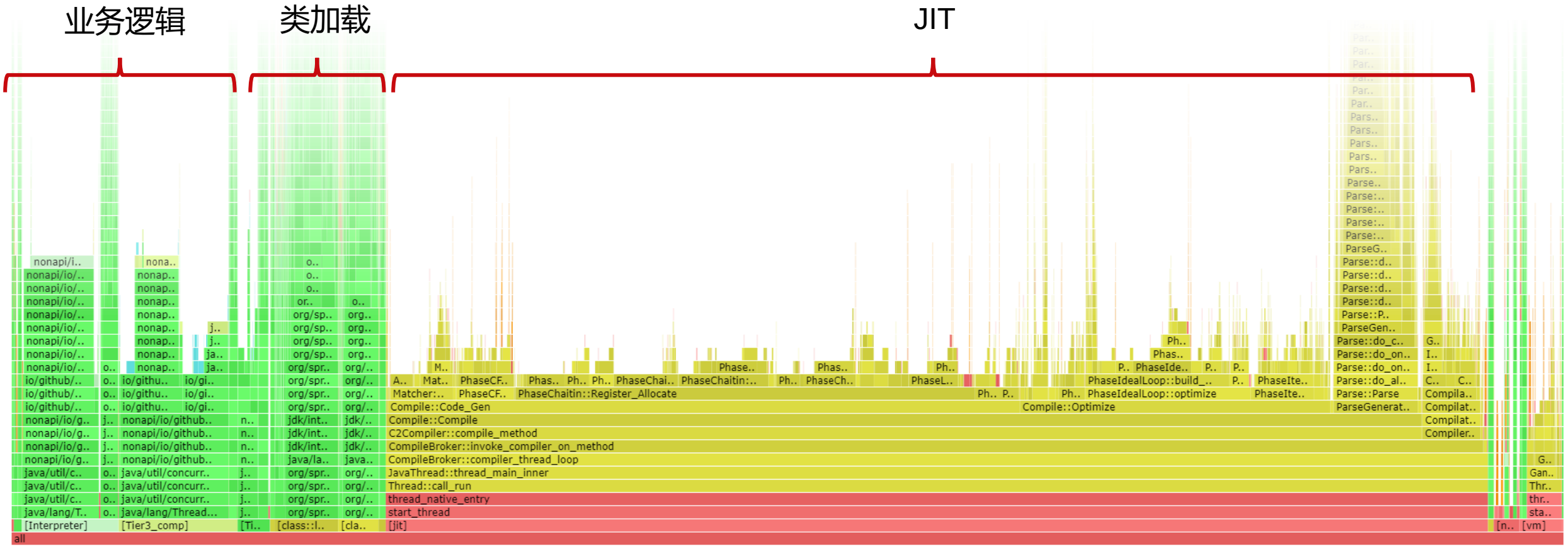
Author: Aleksey Shipilev

分布式JIT - 优化思路

OpenEuler Compiler SIG

火焰图：Spring经典应用PetClinic

- 启动过程中，真正的业务逻辑CPU占比并不大
- JIT占比很大

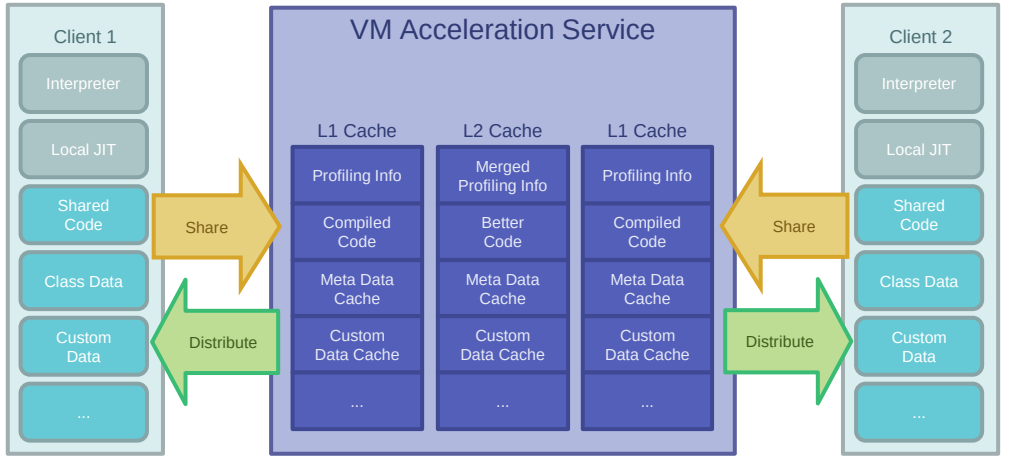


分布式JIT - 框架与流程

OpenEuler Compiler SIG

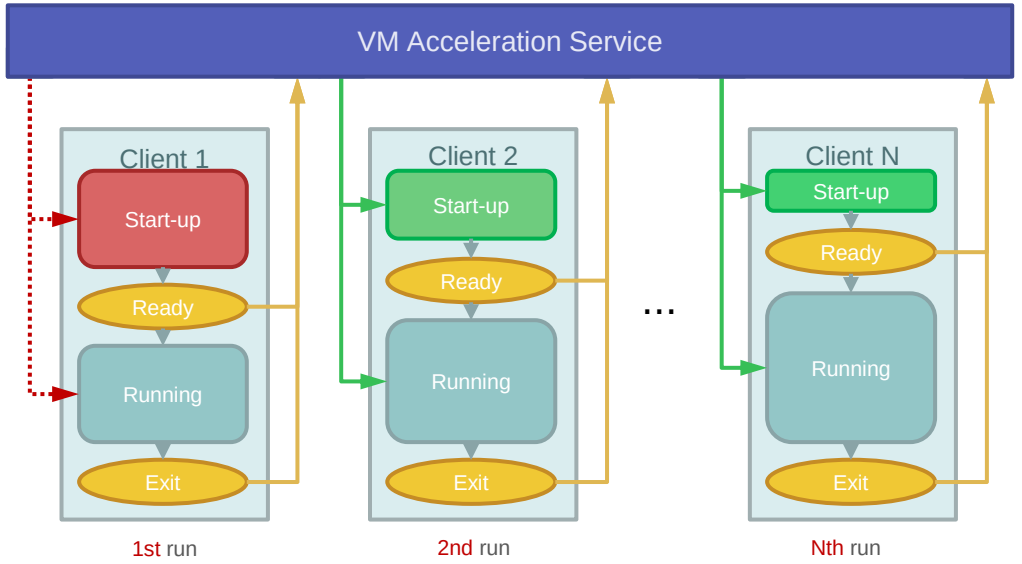
总体框架

- 客户端向服务端分享VM元数据，包括profiling information、meta data、custom data等信息，可分享的数据类型丰富
- 服务端缓存、汇聚、共享和持续迭代多客户端数据，缓存生成加速包，并向所有同类型客户端按需分发加速包
- 不同程序可能使用相同底层框架和第三方库，服务端整合数据并生成更通用的、分层分类的缓存



加速流程

- 自动共享
- 客户端适时地向服务端自动发送服务端缺失或需要更新的信息
- 服务端补全、合并、更新缓存数据
- 缓存可以由服务端生成，也可以由客户端生成后共享
- 服务端也可以最大化缓存的适用范围，持续优化缓存性能，并节约磁盘、CPU等资源
- 自动分发
- 客户端告知服务端可接受加速等级与运行时信息，与服务端协商加速范围，服务端寻找可用加速包发送给客户端
- 同时也支持客户端从本地直接加载已有加速包文件
- 用户可以以可接受的最小的修改，获取最大的加速收益



分布式JIT - Lazy AOT

OpenEuler Compiler SIG

基于JAOTC，但拥有运行时信息优化

- JAOTC编译完全静态编译，不考虑运行时信息，因此编译出的代码性能并不理想
- 完善invoke dynamic的编译支持
- 支持编译由custom class loader加载的类
- 支持部分PGO优化
- 基于invocation counter，仅编译热点方法
- 减小了code size，进而减小了网络传输压力
- 性能还有所提升

远程编译

- 服务端类数据重建
- JAOTC为Java实现的编译器，待编的类需要以Java类的形式传入
- 将客户端的类加载器、类、方法等在服务端进行重建
- 当客户端启动阶段结束后（或运行结束后），即可触发远程编译
- 增量发送服务端缺失的类、需要编译的方法等数据

安全隐私保护：JVM Heapdump匿名化

OpenEuler Compiler SIG

目标：在进行heapdump分析的时候，使得dump出的文件中不带敏感信息

命令行参数：

-XX:HeapDumpRedact指定模式：

1) names：屏蔽敏感symbols，需要用户指定映射表

2) basic：屏蔽int/char/byte数组，全部清零

3) full：names+basic

4) off: 默认关闭

如果不指定HeapDumpRedact则默认为off。

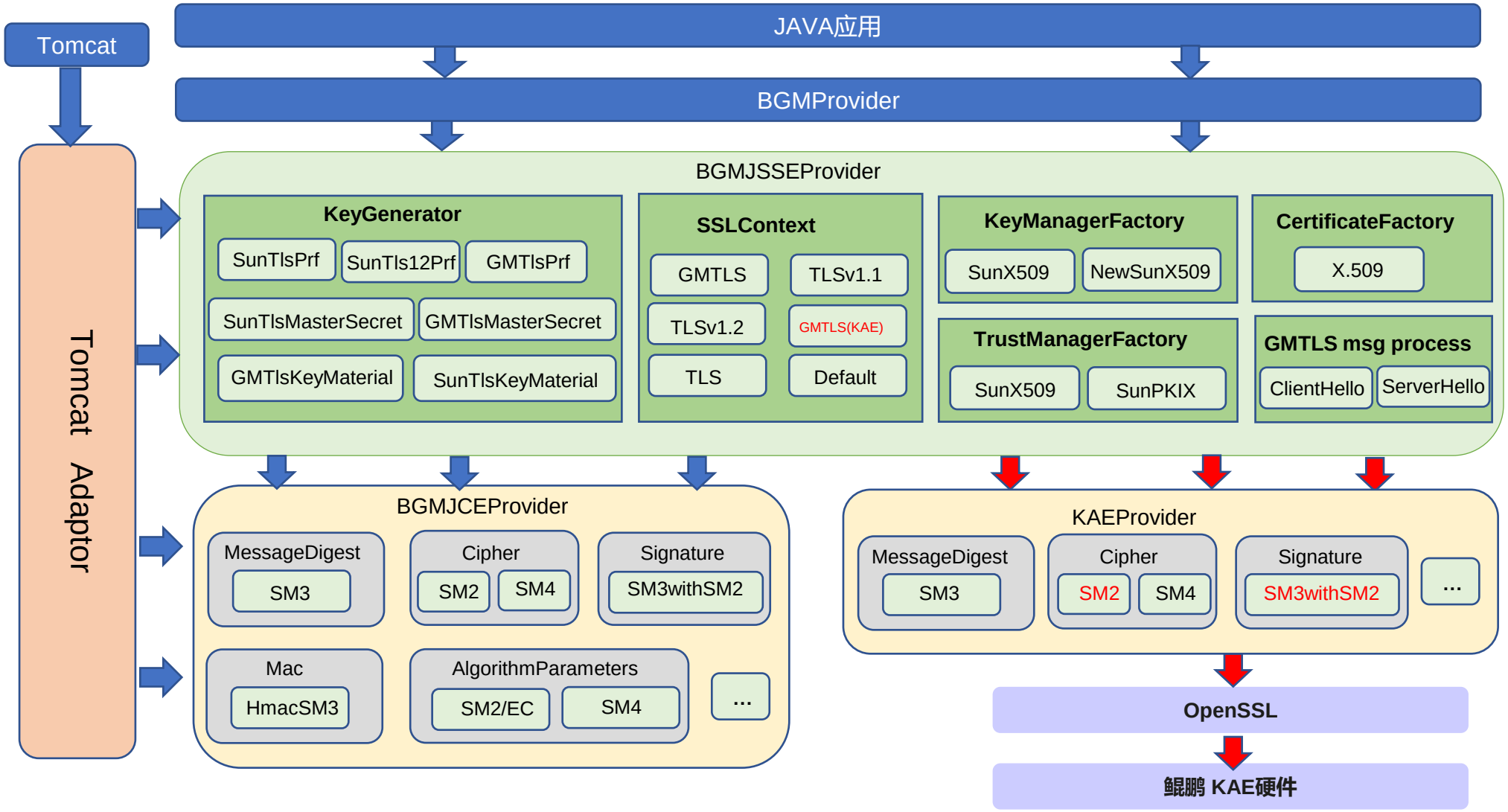
-XX:RedactMap可以在直接在命令行中指定屏蔽敏感名字映射关系对，以逗号作为组之间的分隔，以冒号作为映射对kay/value的分隔。例如："key1:value1,key2:value2……“

使用举例：

java -Xmx10M -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpRedact=full -XX:RedactMap="password:abc,encrypt:cde" MyClass

性能提升: TLS1.3 KAE硬算

OpenEuler Compiler SIG



支持Java版本：
8u302+、
17.0.8+版本。

支持Tomcat:
8.5.2+，9.0.1+，
10.0.0+版本。

红色为需要打通的路径

JDK21 Features

OpenEuler Compiler SIG

JEP430：	字符串模板（预览）
JEP431：	有序集合
JEP439：	分代ZGC
JEP440：	记录模式
JEP441：	开关的模式匹配
JEP442：	外部函数和内存 API（第三次预览版）
JEP443：	未命名模式和变量（预览）
JEP444：	虚拟线程
JEP445：	未命名类和实例主要方法（预览）
JEP446：	范围值（预览）
JEP448：	Vector API（第六个孵化器）
JEP449：	弃用 Windows 32 位 x86 端口以进行删除
JEP451：	准备禁止动态加载代理
JEP452：	密钥封装机制API
JEP453：	结构化并发（预览版）

OpenEuler x Compiler SIG

Thank You.

Compiler SIG 专注于编译器领域技术交流探讨和分享，包括 GCC/LLVM/OpenJDK 以及其他的程序优化技术，聚集编译技术领域的学者、专家、学术等同行，共同推进编译相关技术的发展。

毕昇编译公众号

Compiler 交流群小助手