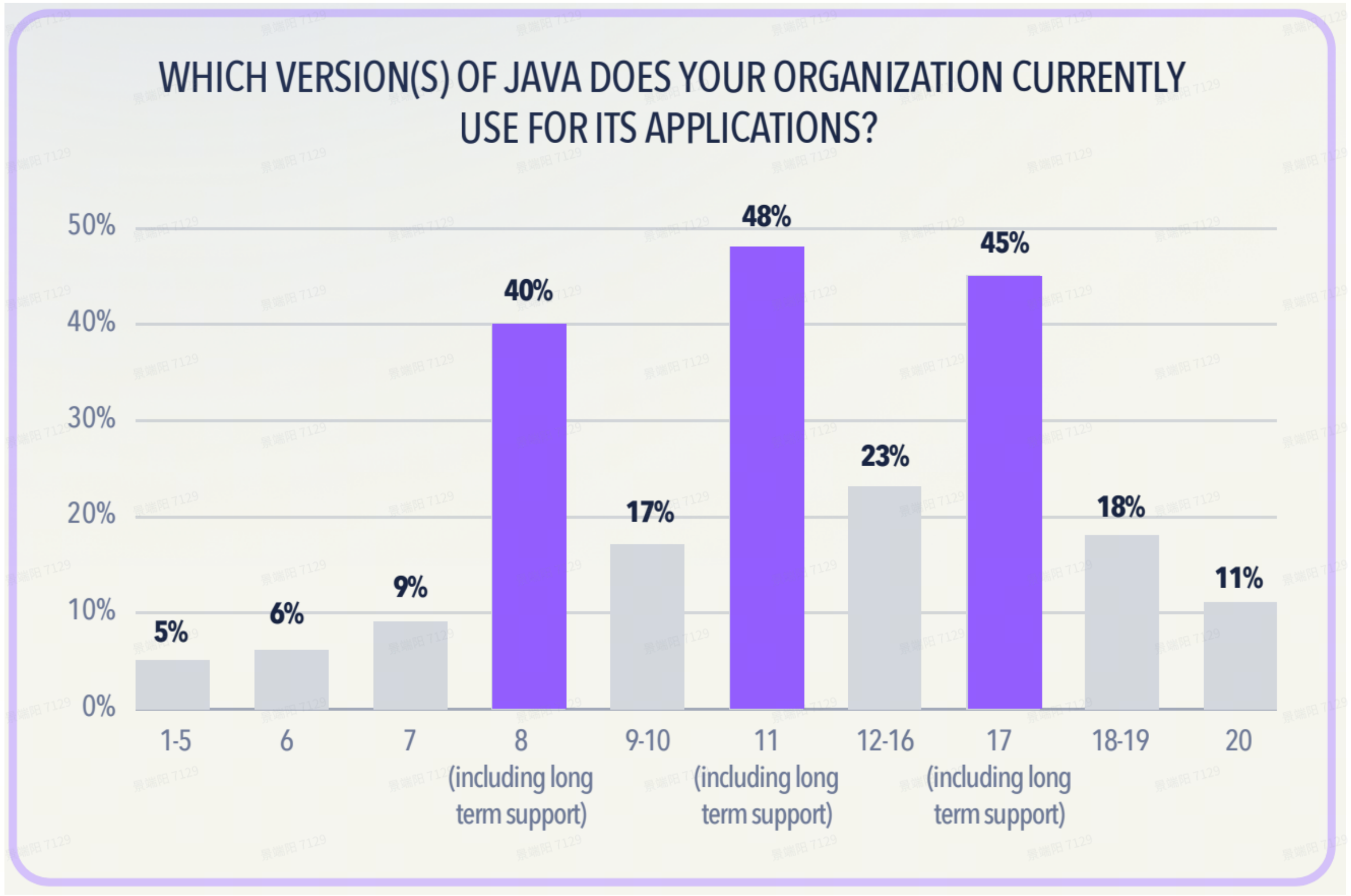
字节跳动

A Tale of Two VMs

-- Compound VM, 帮助Java业务快速升级的组合JDK

字节跳动 编译&语言团队
景端阳
GreenTea JUG 2024

Java业务升级难 – 现状



https://www.azul.com/wp-content/uploads/final-2023-state-of-java-report.pdf

Java语言发展迅速，但业务跟不上节奏，十年前的Java 8依然广泛使用

字节跳动

Java业务升级难 – 原因

目标	获得Bugfix，各种新特性，业务更快更安全
投入	需要投入人力分析各种兼容性问题，甚至重写代码
预期	性能收益 + 稳定运行
风险	性能回退 + 可能会挂

虽然技术升级是大势所趋，但升级似乎**ROI不高**

字节跳动

Java业务升级难 – 一条捷径?

		问题来源	尝试解决
投入	需要投入人力分析各种兼容性问题，甚至重写代码	主要来源于Java层class library不兼容	使用低版本class library
预期	性能收益 + 稳定运行	主要来源于VM层GC算法、JIT、高效Runtime实现	使用高版本VM
风险	性能回退 + 可能会挂		选项使能、快速回退

组合JDK/性能增强包：低版本classlib + 高版本VM

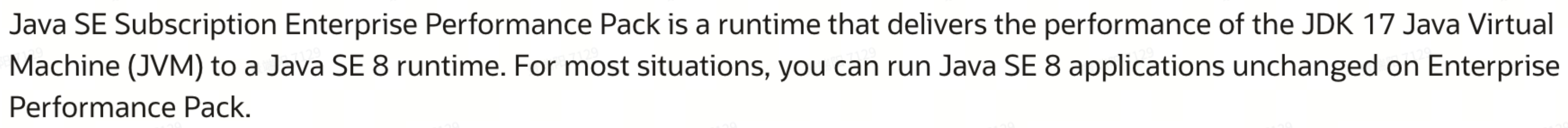
字节跳动

组合JDK – 目标

- 提供 JVM-17 + JDK-8 的组合版 JDK
- 通过 JDK-8 类库、命令行工具为现有基于Java 8的程序提供强兼容性
- 通过 JVM-17 为这些程序提供更高性能

组合JDK – 业界先例

Oracle: Java SE Subscription Enterprise Performance Pack


Java SE Subscription Enterprise Performance Pack is a runtime that delivers the performance of the JDK 17 Java Virtual Machine (JVM) to a Java SE 8 runtime. For most situations, you can run Java SE 8 applications unchanged on Enterprise Performance Pack.

已成熟商用

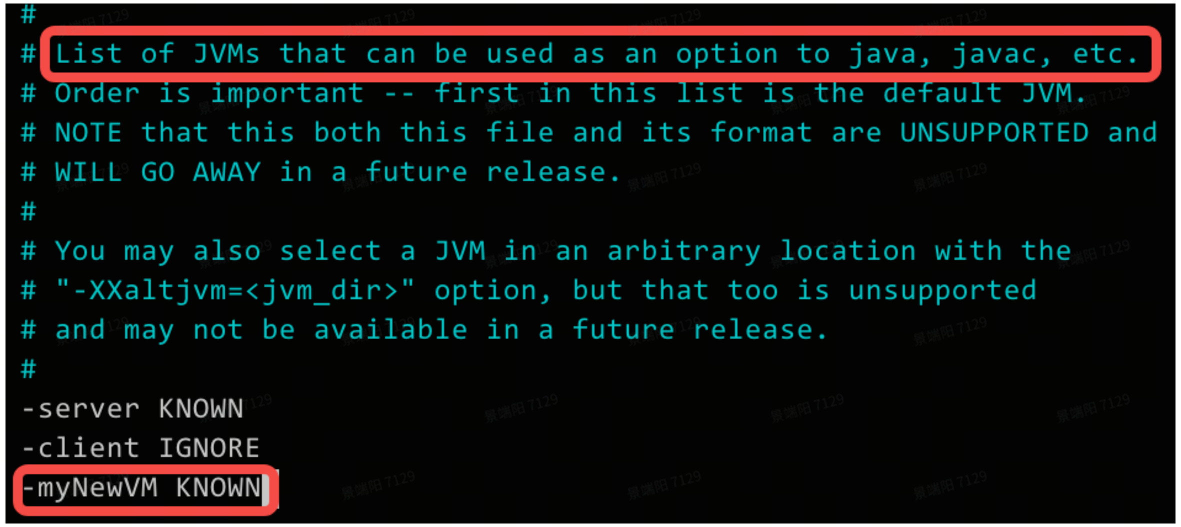
OpenJDK社区：HotSpot Express

已废弃

其他

组合JDK – 可行性与挑战

- 一个JDK本来就支持多个JVM
- 高版本JVM向前兼容低版本bytecode

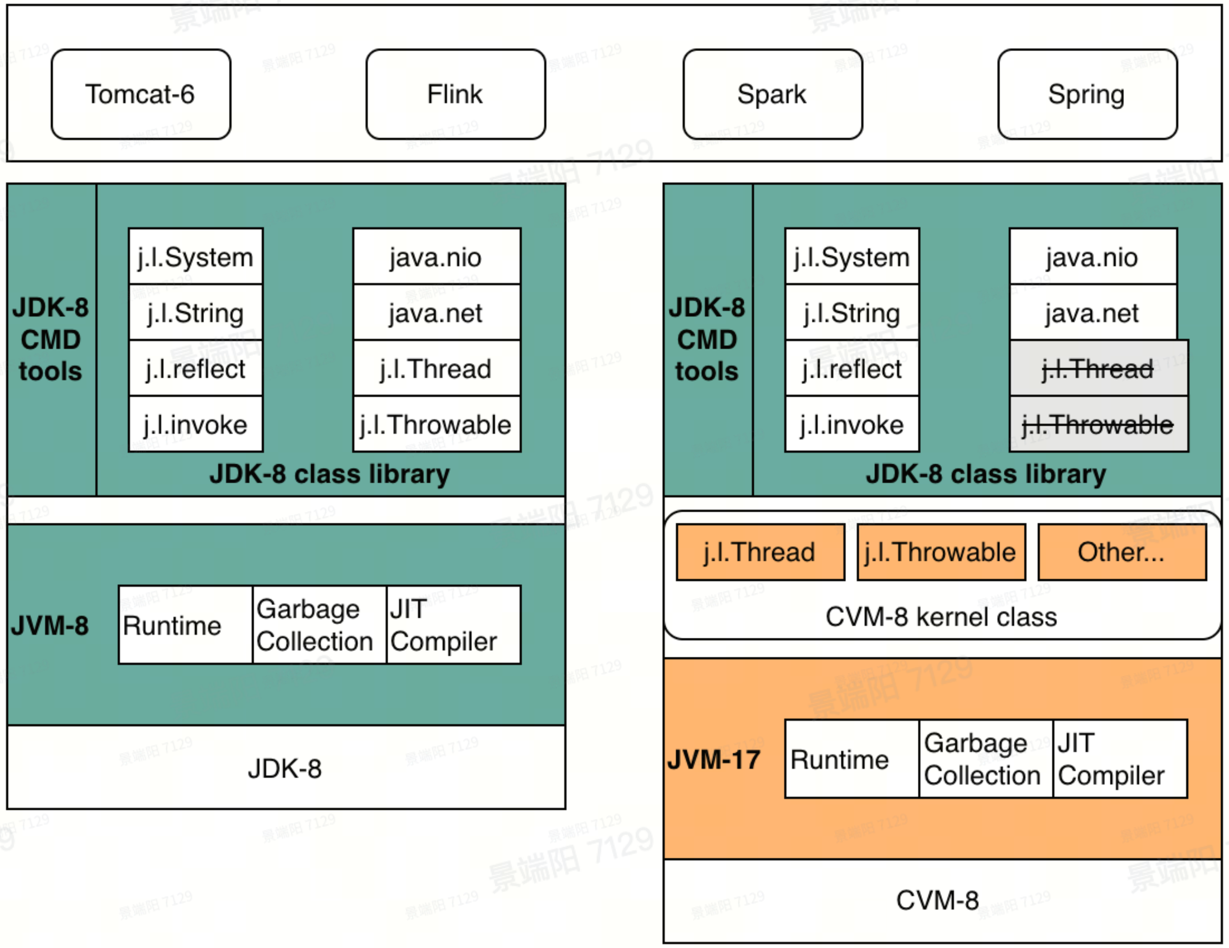


```
#
# List of JVMs that can be used as an option to java, javac, etc.
# Order is important -- first in this list is the default JVM.
# NOTE that this both this file and its format are UNSUPPORTED and
# WILL GO AWAY in a future release.
#
# You may also select a JVM in an arbitrary location with the
# "-XXaltjvm=<jvm_dir>" option, but that too is unsupported
# and may not be available in a future release.
#
-server KNOWN
-client IGNORE
-myNewVM KNOWN
```

jdk8/jre/lib/amd64/jvm.cfg

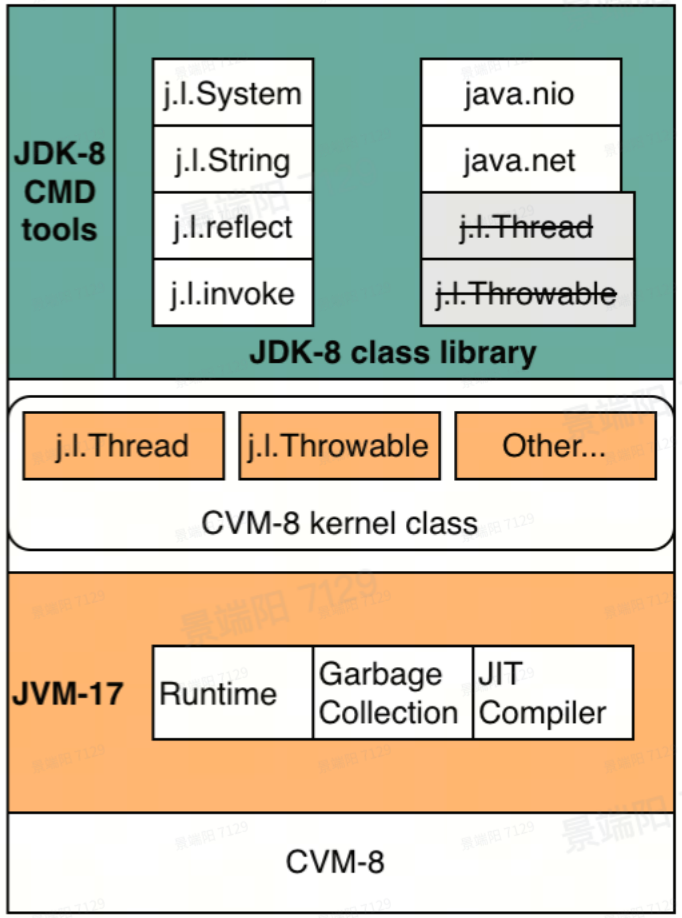
- classlib不一致，API修改、删除等
- JVM实现不一致，如JNI_XX, JVM_XX
- VM options不一致，如-Xbootclasspath
- 其他行为不一致，如classloader, modularity

组合JDK – Compound VM



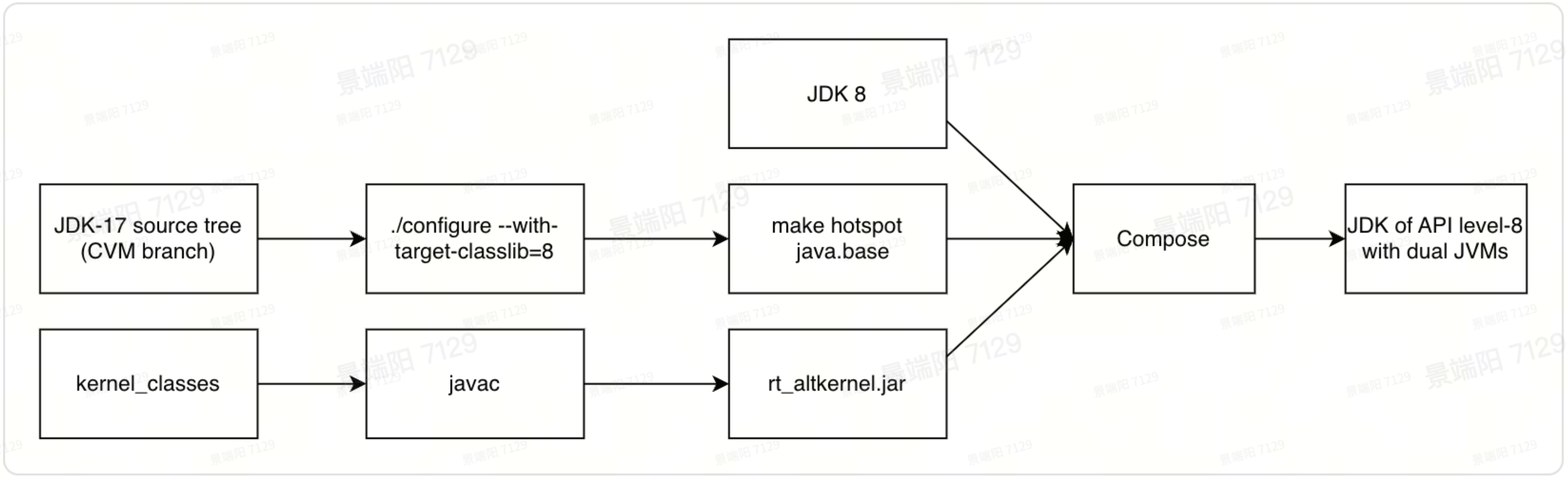
- Kernel class：与JVM 耦合比较紧密的classes，基于JDK-17改造并覆盖JDK8的rt.jar
- 改造JVM-17：与classlib-8的signature一致，补全缺失的API，支持"well known classes"，支持常用options

组合JDK – Compound VM



- 产品定位：组合JDK/现有JDK的性能增强包
- 使用场景：
  - 对现有的JDK进行增强
  - 也可直接作为一个JDK使用
- Usage: java –server17
- 主要特性
  - 无需业务代码改动，快速升级，快速回滚
  - JVM-17的诸多新特性，GC、JIT的改进
- 对比业界先例：可以直接对现有的JDK二进制进行无修改扩展

CVM实现 – 构建流程


JDK 8
JDK-17 source tree (CVM branch)
./configure --with-target-classlib=8
make hotspot java.base
Compose
JDK of API level-8 with dual JVMs
kernel_classes
javac
rt_altkernel.jar

- 原始 JDK-8 + JVM-8 可正常运行
- JVM-17作为候选JVM存在
- JVM-17需要依赖的java实现在rt_altkernel.jar中

CVM-8: JDK-8 + JVM-17 组合JDK

CVM实现 – CompactString

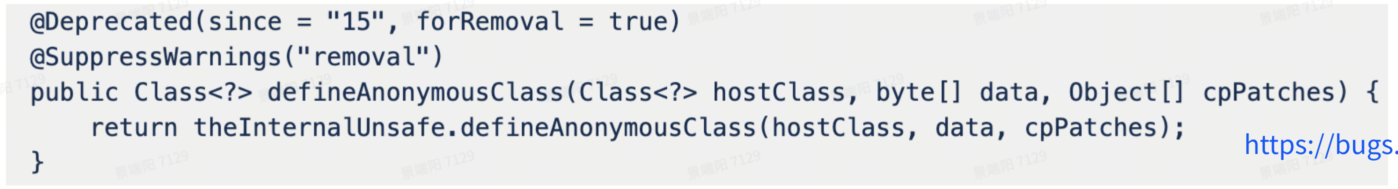
```java
public static void copyUSAsciiStrToBytes(String str, byte[] bytes) {
    if (isJavaVersion9Plus) {
        final byte[] chars = (byte[]) instance.getObject(str, stringValueFieldOffset);
        System.arraycopy(chars, 0, bytes, 0, str.length());
    } else {
        final char[] chars = (char[]) instance.getObject(str, stringValueFieldOffset);
        int i = 0;
        while (i < str.length()) {
            bytes[i] = (byte) chars[i++];
        }
    }
}
```

https://github.com/akka/akka/blob/v2.5.21/akka-actor/src/main/scala/akka/util/Unsafe.java

出于以下考虑CVM-8 没有支持CompactString

- 功能：CompactString将String的内部存储从char[]变为byte[]，会影响前向兼容性
- 性能：在一些benchmark上，CompactString对coder()的访问是性能瓶颈

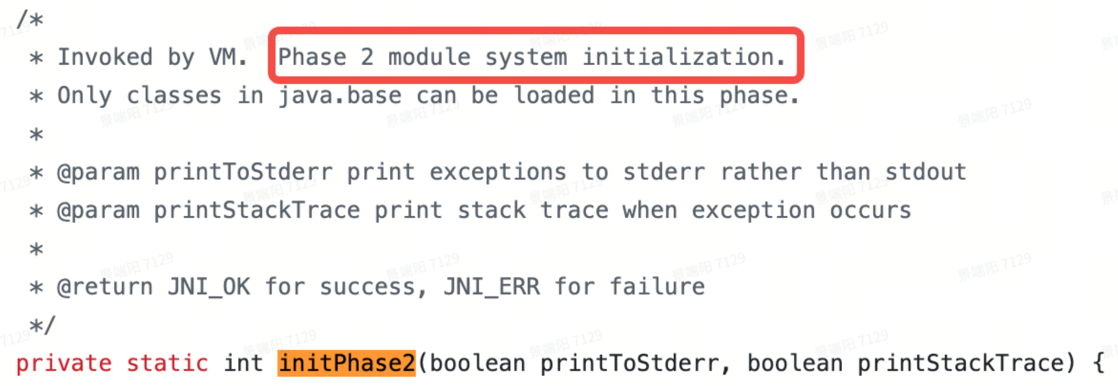
CVM实现 – Lambda & Module



```
@Deprecated(since = "15", forRemoval = true)
@SuppressWarnings("removal")
public Class<?> defineAnonymousClass(Class<?> hostClass, byte[] data, Object[] cpPatches) {
    return theInternalUnsafe.defineAnonymousClass(hostClass, data, cpPatches);
}
```

https://bugs.openjdk.org/browse/JDK-8266760

e.g. Lambda的实现依赖Unsafe_DefineAnonymousClass，需要适配



```
/*
 * Invoked by VM.  Phase 2 module system initialization.
 * Only classes in java.base can be loaded in this phase.
 *
 * @param printToStderr print exceptions to stderr rather than stdout
 * @param printStackTrace print stack trace when exception occurs
 *
 * @return JNI_OK for success, JNI_ERR for failure
 */
private static int initPhase2(boolean printToStderr, boolean printStackTrace) {
```

e.g. Module相关内容需要在17中移除

CVM实现 – SecurityManager

SecurityManager可以用于执行类似的代码，如果代码没有对应权限会抛出exception：

```java
String user = AccessController.doPrivileged(
    new PrivilegedAction<String>() {
    public String run() {
        return System.getProperty("user.name");
        }
    });
```

CVM-8中绕过了Security规则的检查：

- 特性使用场景较少，JDK17 deprecated
- 更完善高效的security机制已普遍使用

JEP 411: Deprecate the Security Manager for Removal

Java 8实现：

```java
public final class AccessController {

    @CallerSensitive
    public static native <T> T doPrivileged(PrivilegedAction<T> action);
```

https://github.com/openjdk/jdk/blob/jdk8-b120/jdk/src/share/classes/java/security/AccessController.java

Java 17实现：

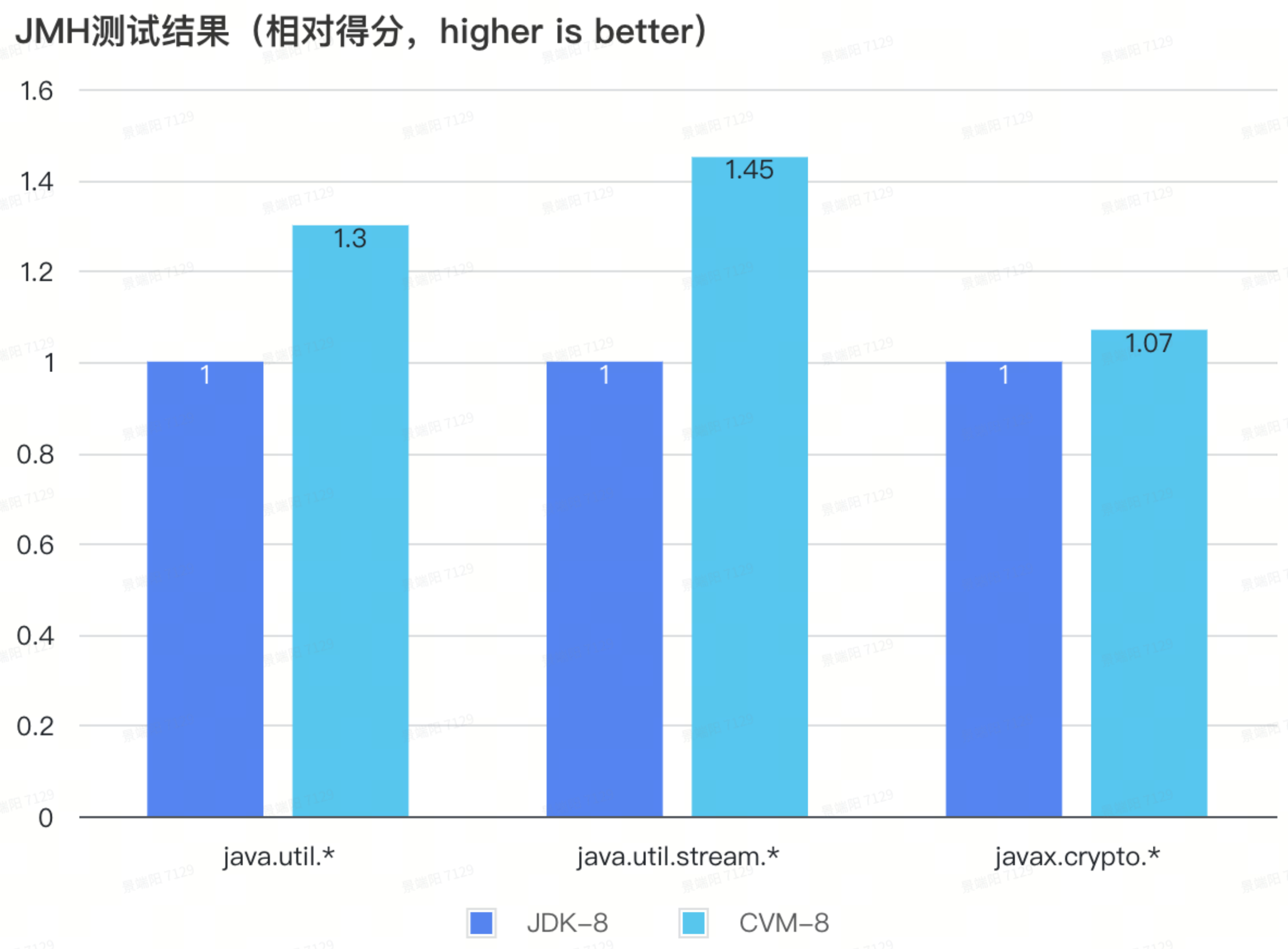
```java
public final class AccessController {

    @CallerSensitive
    public static <T> T doPrivileged(PrivilegedAction<T> action)
    {
        return executePrivileged(action, null, Reflection.getCallerClass());
    }
```

https://github.com/openjdk/jdk/blob/jdk-17%2B0/src/java.base/share/classes/java/security/AccessController.java

JVM_doPrivileged在17中不存在，在Java层直接execute action

字节跳动

性能收益 – JMH

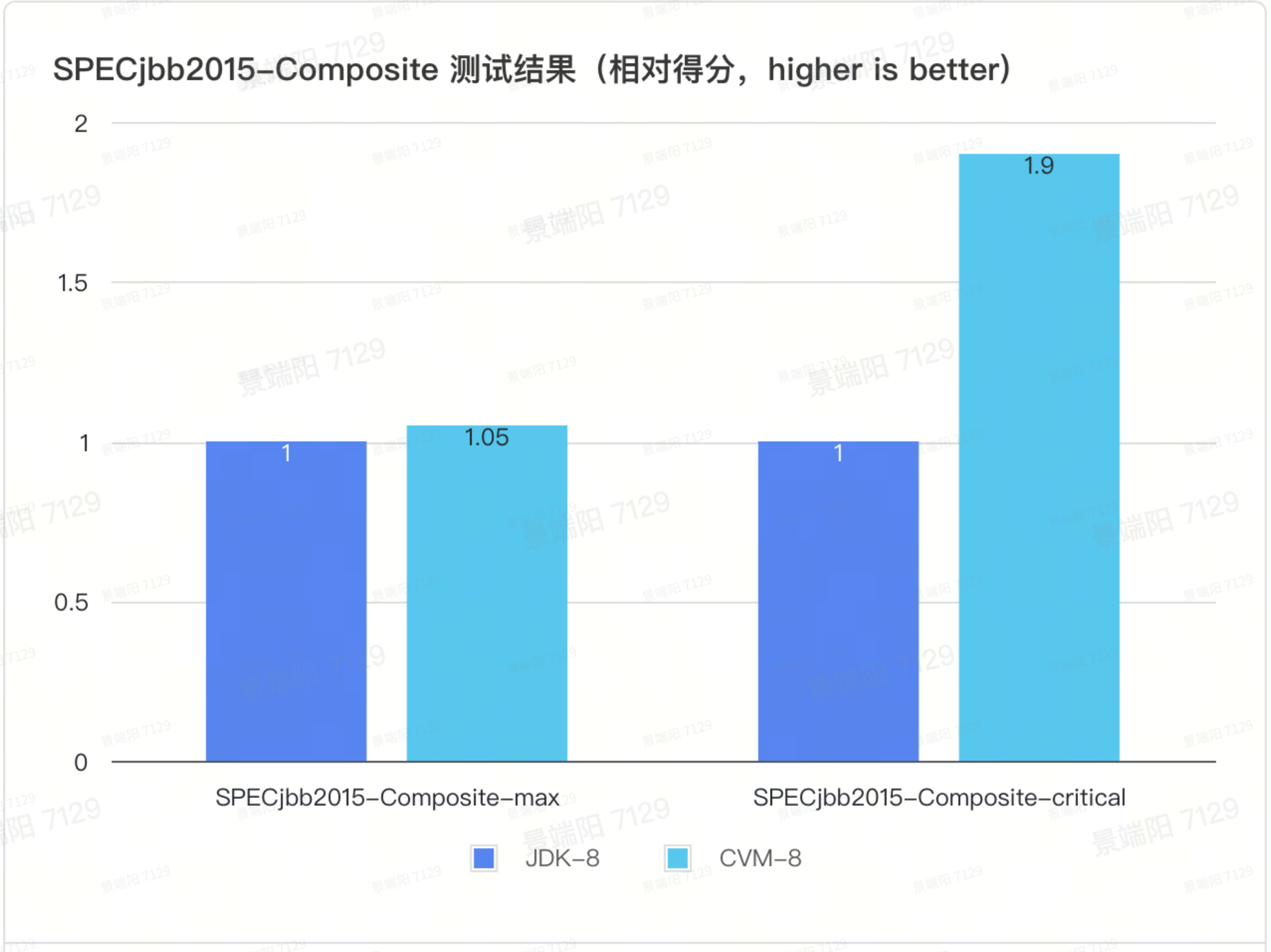


java.util相关API性能平均提升30%

java.util.stream相关API性能平均提升45%

javax.crypto相关API性能平均提升7%

性能收益 – SPECJbb

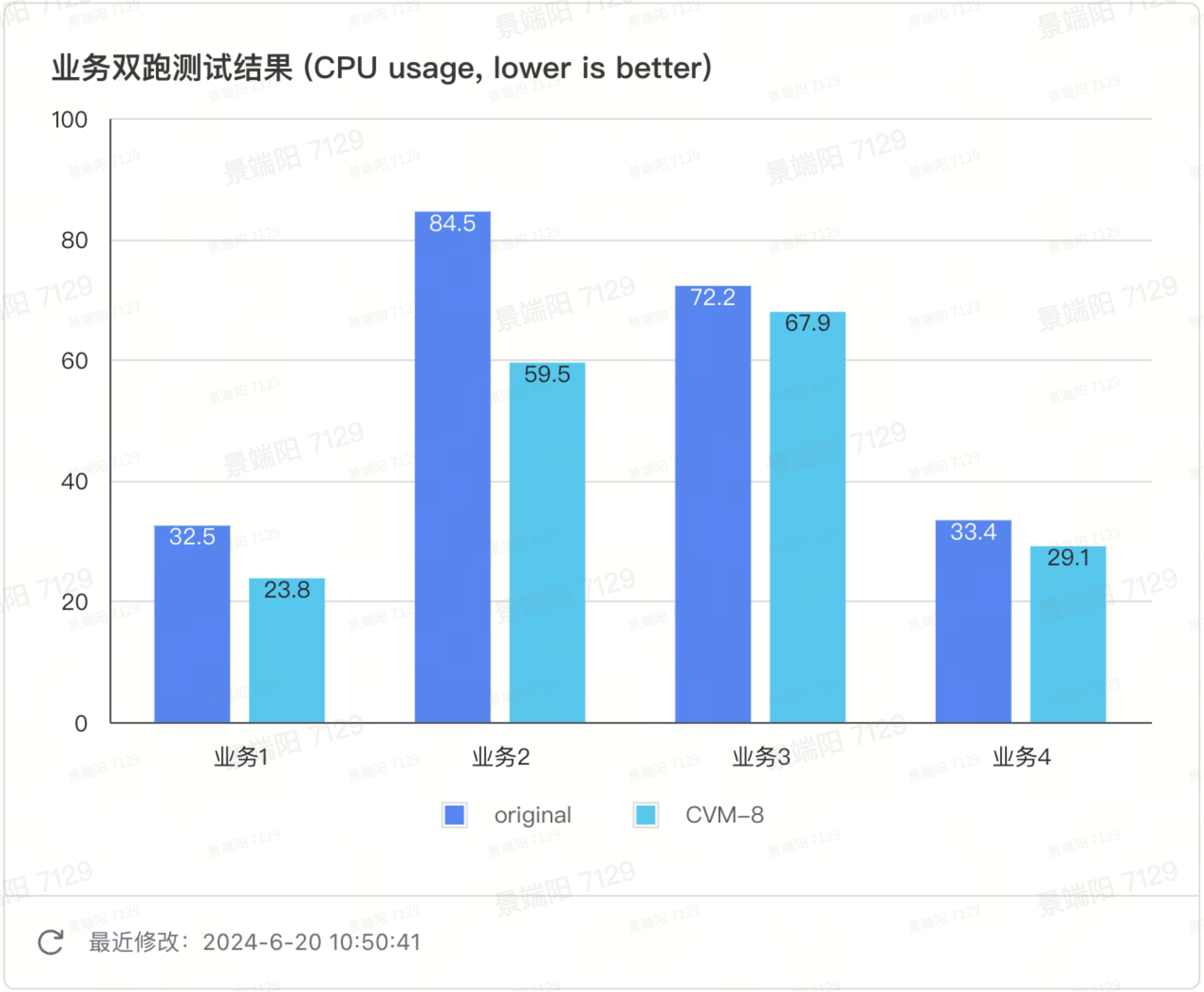


SPECJbb2015-Composite-max：侧重吞吐量，提升5%

SPECJbb2015-Composite-critical：侧重时延，提升90%

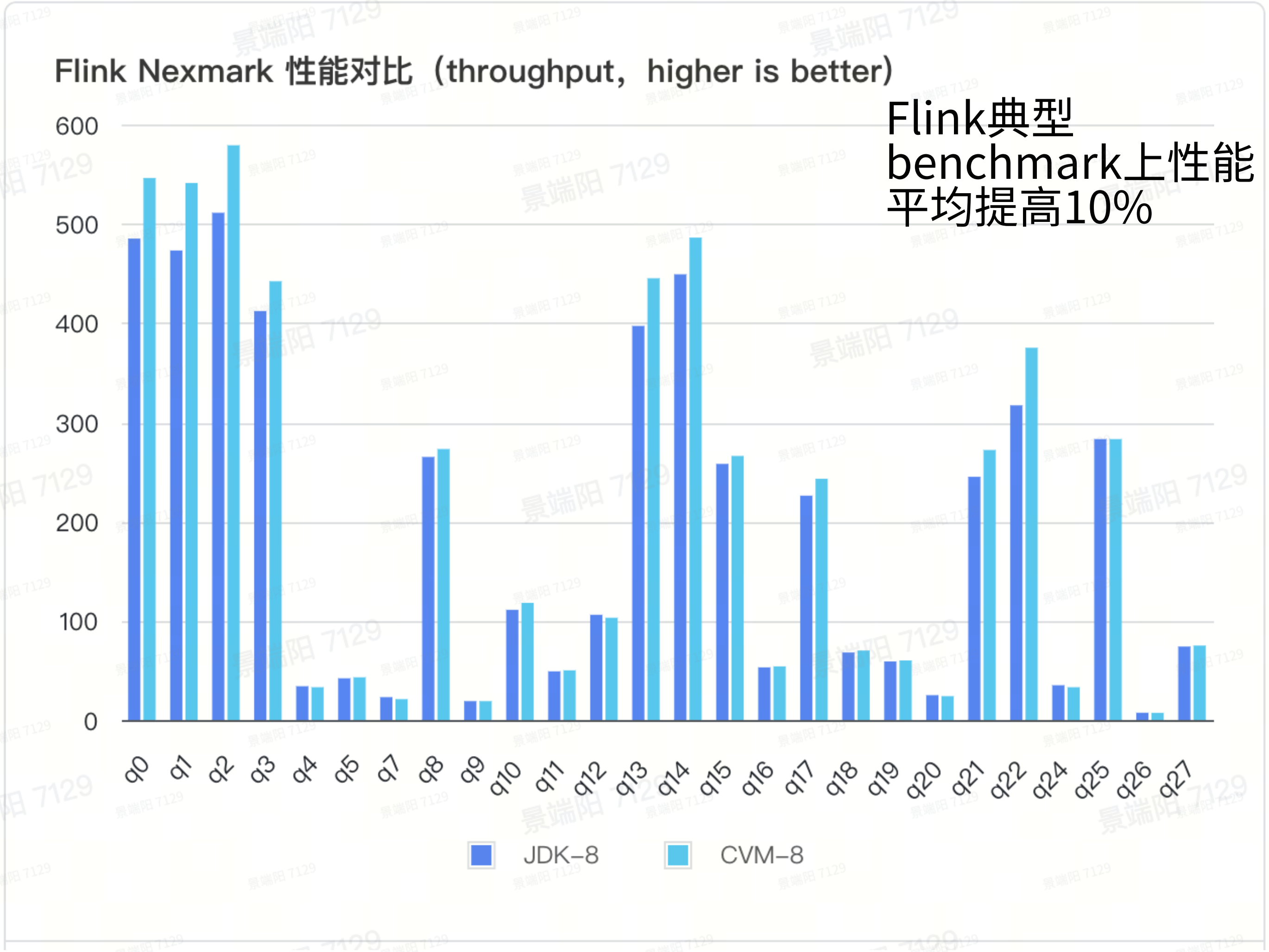
字节跳动

性能收益 – Flink



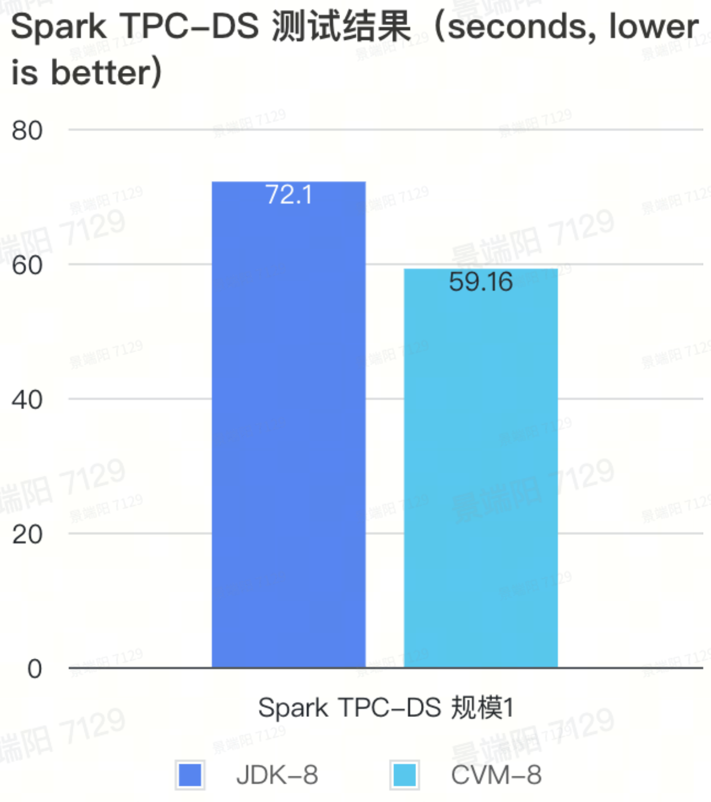
在多个业务上进行双跑测试，高峰期最多有28%的cpu资源节省，平均有近15%的CPU资源节省。

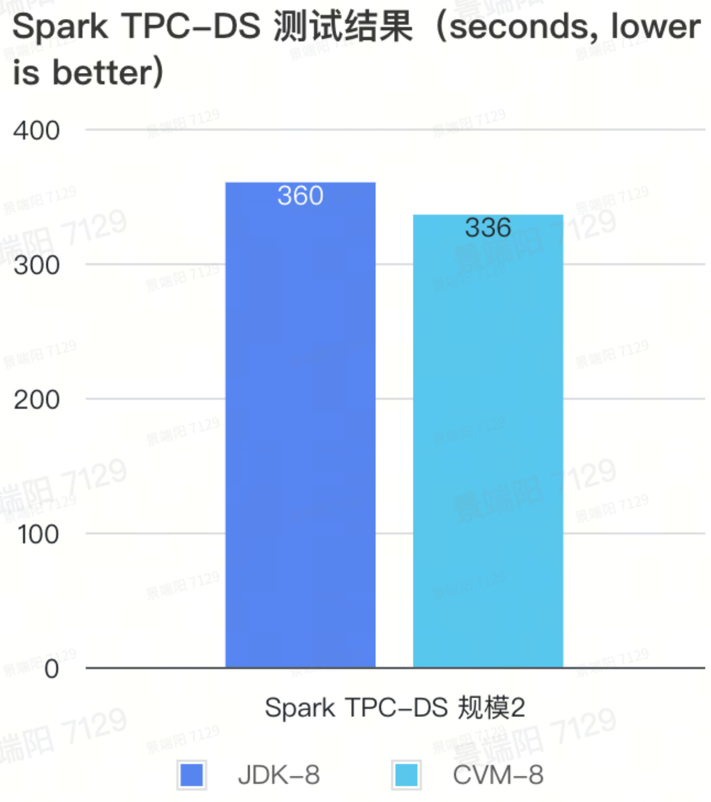
性能收益 – Flink nexmark



Flink典型benchmark上性能平均提高10%

性能收益 – Spark TPC-DS





Spark TPC-DS有10%左右提升

字节跳动

总结与展望

- CVM-8: JDK-8 + JVM-17
- CVM-11: JDK-11 + JVM-21 分代ZGC
- CVM-17: JDK-17 + JVM-25 TBD

字节跳动

THANKS

字节跳动