

JDK 7u40利器 —— Java Flight Recorder



GreenTea JUG

梅路晓 (云达)

同在阿里
修身养性
TongZaiAli



关于我

- 2011.7进入淘宝实习和工作至今，从事JVM相关工作
- GreenTea JUG创始人之一
- 对编程语言的实现很感兴趣，目前关注重点在JVM的实现，活跃于OpenJDK社区
- <http://weibo.com/u/1063244843>
- 来往账号：mly909



什么是Java Flight Recorder

- Tracer & Profiler
- 内置于JVM中
- 按需监控
- 事后分析
- 源于JRockit , Oracle JDK 7u40开始引入
- 商业工具 , OpenJDK中有基本埋点 : JEP 167



Tracer & Profiler

- 既能trace JVM事件，也能trace应用事件
 - ✓ GC
 - ✓ Synchronization
 - ✓ JIT
 - ✓ CPU usage
 - ✓ Exceptions
 - ✓ I/O
- 基于采样的profiler
 - ✓ 非常低的overhead
 - ✓ 准确的数据



内置于JVM中

- 核心代码位于JVM内部
- 性能损失低，一般2%-3%
- 便于和JVM各模块的交互
- 支持Java编写的模块
- 可以在运行时打开/关闭



按需监控

- 通过Java Mission Control启动
- 也可使用jcmd命令行工具
- 可以配置需要监控的事件类型
- 关闭监控时，没有任何额外开销



事后分析

- 记录在一个二进制文件中
- 通过Java Mission Control进行分析
- 记录足够多的信息以分析问题



Demo

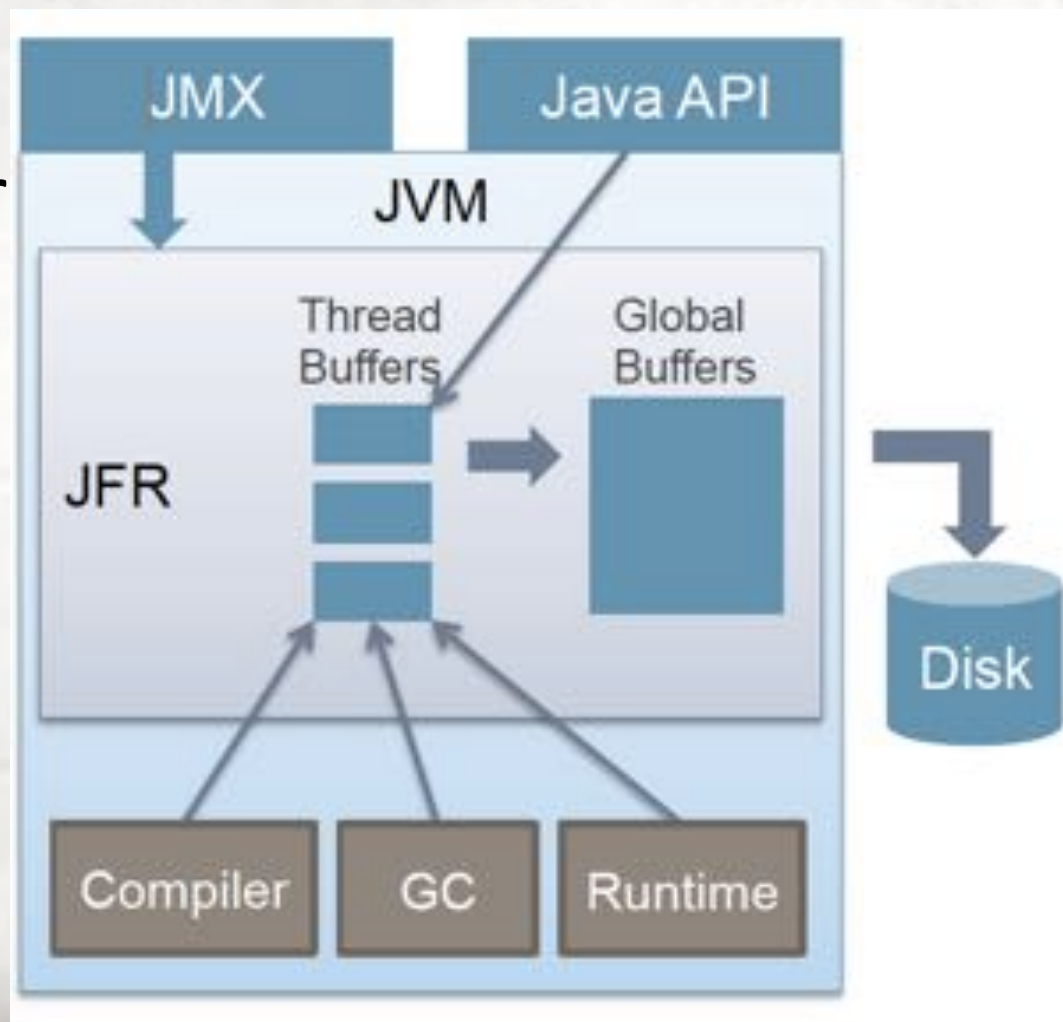


配置

- 启动参数:
-XX:+UnlockCommercialFeatures -XX:+FlightRecorder
- 开始记录
-XX:StartFlightRecording=filename=<path>,
duration=<time>
- jcmd
jcmd <pid> JFR.start filename=<path> duration=<time>

如何实现

- Event-based
- Thread-local buffer
- Global buffer
- 专有二进制文件





事件分类

- **Instant**
 - ✓ 只在某个时间点发生
 - ✓ Thread start
- **Duration**
 - ✓ 发生于一段持续时间内
 - ✓ GC
- **Requestable**
 - ✓ 以固定频率发生
 - ✓ 每秒统计CPU利用率



事件格式

- 事件记录
 - ✓ 事件ID
 - ✓ 线程ID
 - ✓ 事件起始时间
 - ✓ 事件结束时间
 - ✓ 栈记录ID
 - ✓ 事件具体信息

| |
|---------------|
| Event ID |
| Thread ID |
| Start |
| End |
| Stacktrace ID |
| Payload |



JVM中的事件定义

```
<event id="ThreadSleep"
      path="java/thread_sleep"
      label="Java Thread Sleep" ...>
  <value field="time"
        type="MILLIS"
        label="Sleep Time" />
</event>
```

- XML定义会被解释成C++类



事件的记录

```
JVM_Sleep(int millis) {  
    EventThreadSleep event;  
  
    ... // actual sleep happens here  
  
    event.set_time(millis);  
    event.commit();  
}
```

- Done! 数据已经记录到Flight Record中!

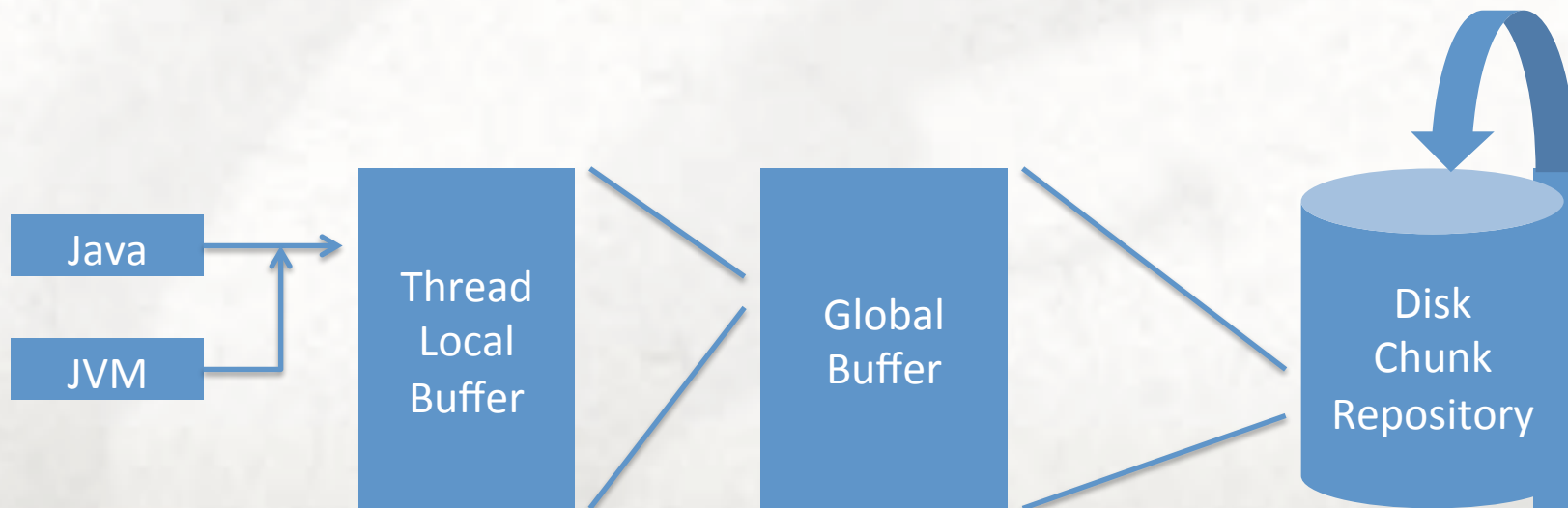
事件范例

Event Attributes

| Name | Value |
|--------------------------|--|
| Start Time | 2013-08-16 12:53:42.388 |
| End Time | 2013-08-16 12:53:42.416 |
| Duration | 28 ms 449 µs |
| Class Parked On | java.util.concurrent.locks.AbstractQueuedSynchronizer\$ConditionObject |
| Park Timeout | 0 s |
| Address of Object Parked | 0xE22D4DC8 |
| Event Thread | Thread-13 |
| | Unsafe.park(boolean, long) |
| | LockSupport.park(Object) line: 186 |
| | AbstractQueuedSynchronizer\$ConditionObject.await() line: 2043 |
| | LinkedBlockingQueue.take() line: 442 |
| | JDK15ConcurrentBlockingQueue.take() line: 89 |
| | PersistentStoreImpl.getOutstandingWork() line: 678 |
| | PersistentStoreImpl.synchronousFlush() line: 1078 |
| | PersistentStoreImpl.run() line: 1070 |
| | Thread.run() line: 724 |

Buffers

- Ring buffer
 - ✓ 可以设计成无锁的
- 尽可能减少竞争

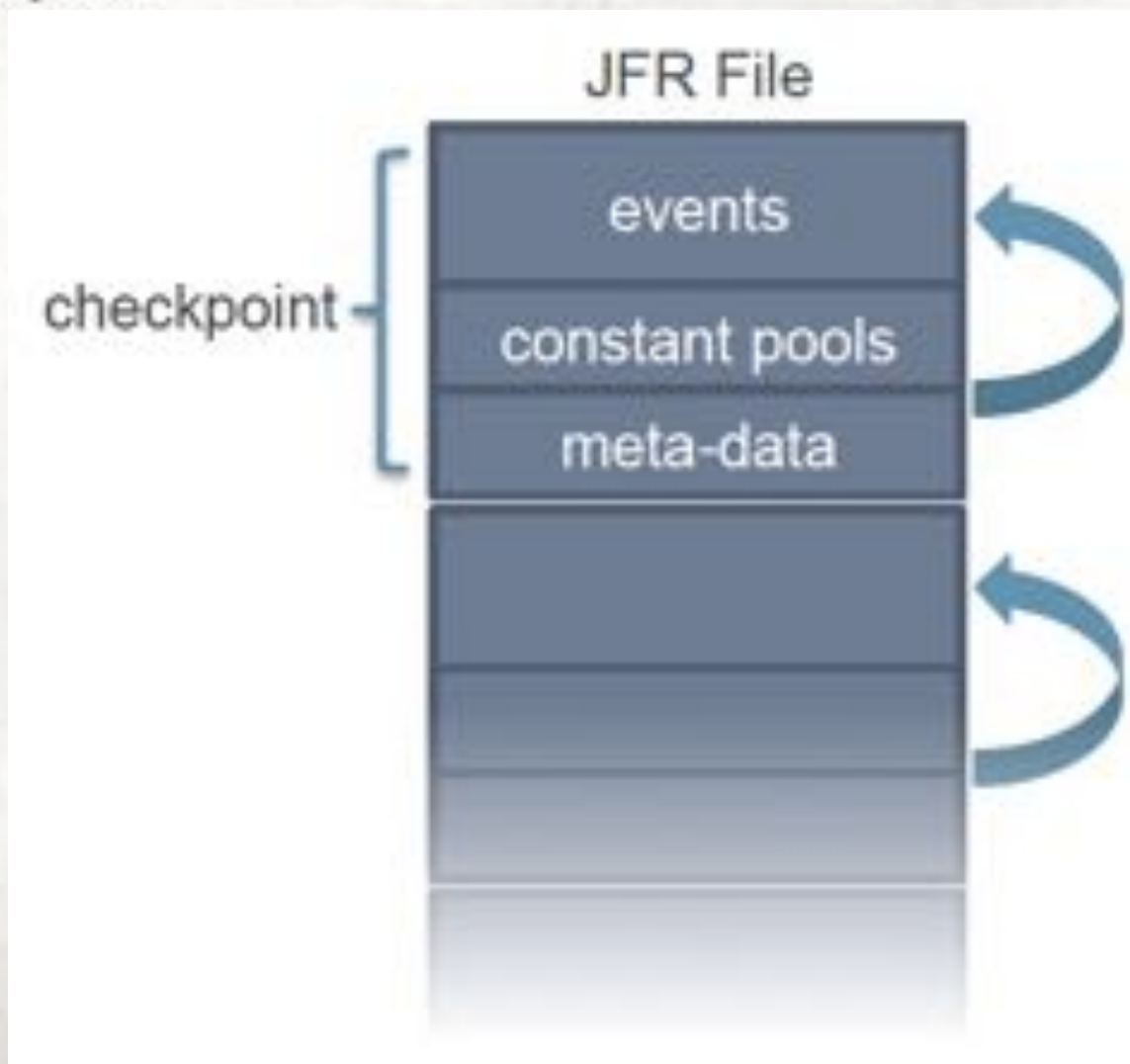




Checkpoints机制

- 有规律的创建Checkpoint
- 每个Checkpoint包含解析自上一次Checkpoint以后事件的信息的所需信息
- checkpoint =
 - events
 - + constant pools
 - + event meta-data

Checkpoint机制





参考资料

- JRockit Flight Recorder介绍：
http://docs.oracle.com/cd/E15289_01/doc.40/e15070/introduction.htm
- Java Flight Recorder Behind the Scenes：
https://oracleus.activeevents.com/2013/connect/sessionDetail.ww?SESSION_ID=5091
- Oracle Java Mission Control: Java Flight Recorder Deep Dive：
https://oracleus.activeevents.com/2013/connect/sessionDetail.ww?SESSION_ID=5181

Q & A



同在阿里
修身养性
TongZaiAli