

拉勾教育

— 互联网人实战大学 —

《31 讲带你搞懂 SkyWalking》

徐郡明 资深技术专家

— 拉勾教育出品 —

加餐1：DataCarrier 实现详解

DataCarrier

是一个轻量级的生产者-消费者模式的实现库

SkyWalking Agent 在收集到 Trace 数据之后

会先写入到 DataCarrier 中的缓存，然后由后台线程定时发送到后端的 OAP



Buffer 核心原理

拉勾教育

— 互联网人实战大学 —

DataCarrier 底层使用多个定长数组作为存储缓冲区

即 apm-datacarrier 模块中的 Buffer 类

其底层的 **buffer** 字段（Object[] 类型）是真正存储数据的地方



Buffer 核心原理

拉勾教育

— 互联网人实战大学 —

Buffer 可以指定下面三种写入策略，这些策略只在 Buffer 写满的情况下才生效：

- **BLOCKING 策略（默认）**

写入线程阻塞等待，直到 Buffer 有空闲空间为止

- **OVERRIDE 策略**

覆盖旧数据，会导致缓存在 Buffer 中的旧数据丢失

- **IF_POSSIBLE 策略**

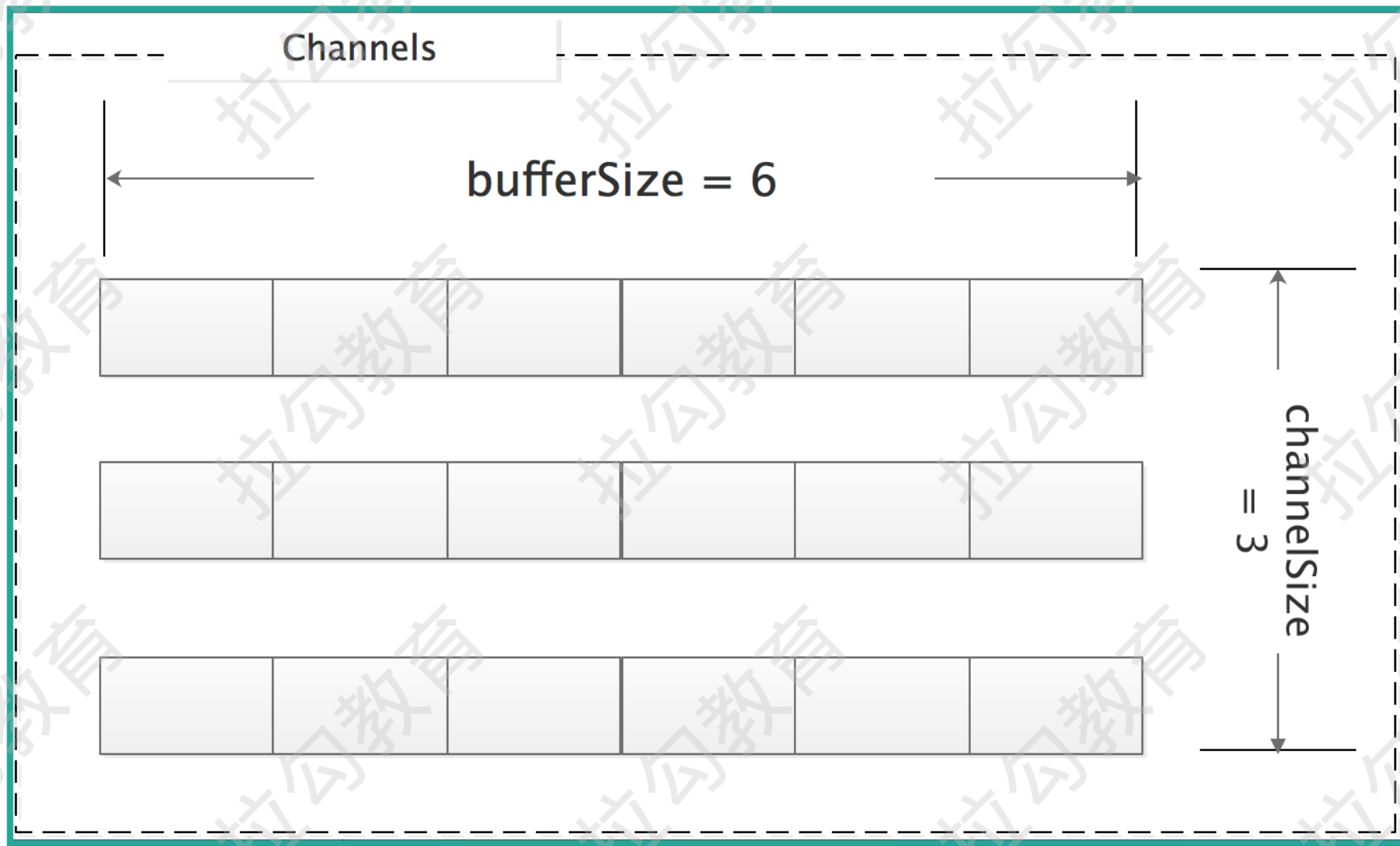
如果无法写入则直接返回 false，由上层应用判断如何处理



Channels

拉勾教育

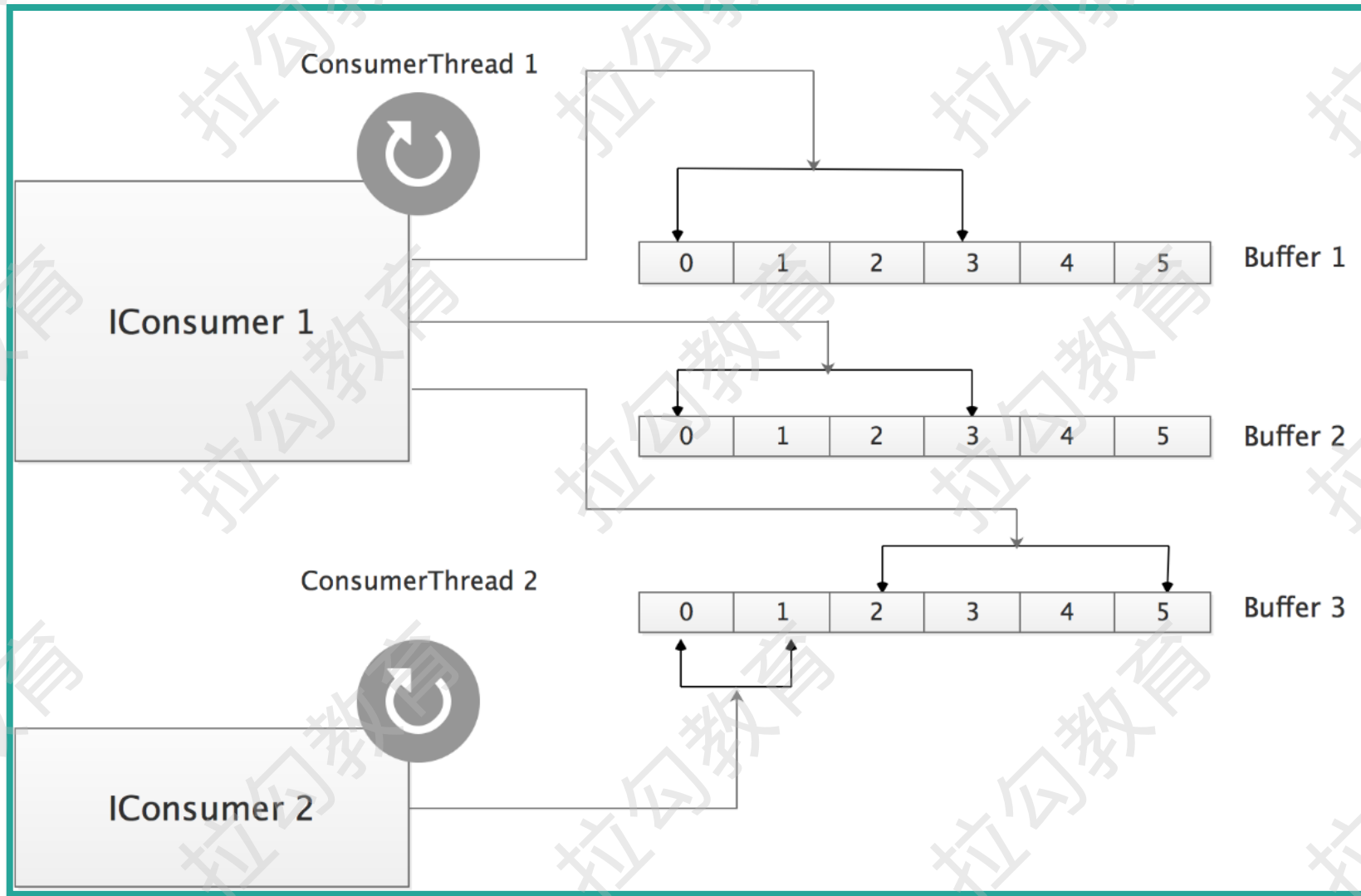
— 互联网人实战大学 —

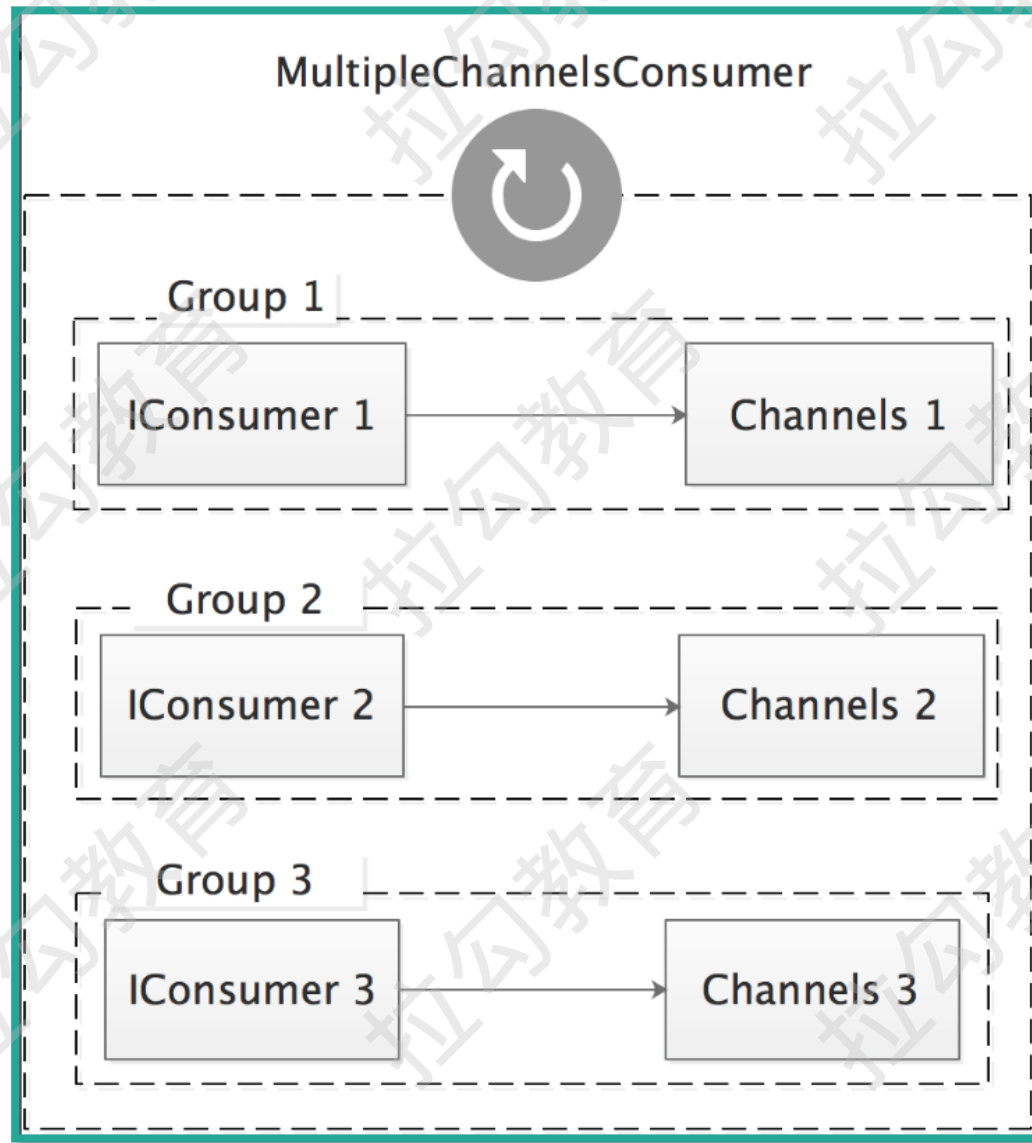


DataCarrier 消费者

拉勾教育

— 互联网人实战大学 —





ConsumerDriver 的核心逻辑是在其 begin() 方法中

会根据 Channels 中的 Buffer 数量以及 ConsumerThread 线程数进行分配：

- 如果 Buffer 个数较多，则一个 ConsumerThread 线程需要处理多个 Buffer
- 如果 ConsumerThread 线程数较多，则一个 Buffer 会被划分为多个区域
由不同的 ConsumerThread 线程进行消费
每个 ConsumerThread 线程负责消费一个 Buffer 的一个区域
- 如果两者数量正好相同，则是一对一的消费关系



DataCarrier

DataCarrier





是整个 DataCarrier 模块最顶层的门面类

其中整合 Channels、IDriver 并给 Producer 提供一个统一的入口

该构造方法中会接收 channelSize、bufferSize 两个参数初始化 Channels

默认使用 SimpleRollingPartitioner 分区选择器以及 BLOCKING 策略



- m  consume(Class<? extends IConsumer<T>>, int): DataCarrier
- m  consume(Class<? extends IConsumer<T>>, int, long): DataCarrier
- m  consume(IConsumer<T>, int): DataCarrier
- m  consume(IConsumer<T>, int, long): DataCarrier

m  consume(IConsumer<T>, int, long): DataCarrier

总结

拉勾教育

— 互联网人实战大学 —

- 介绍 DataCarrier 最底层的数据存储组件 Buffer 和 Channels 以及相关的填充策略
- 深入分析 DataCarrier 提供的消费者接口以及两种消费模型
- 介绍 IDriver 接口和 DataCarrier 门面类提供的 API 实现



Next: 第13讲 《收集、发送 Trace 核心原理，Agent 与 OAP 的大动脉》

拉勾教育

— 互联网人实战大学 —



关注拉勾「教育公众号」
获取更多课程信息