|  |  |
| --- | --- |
| Bug ID（缺陷标识符） | HBASE-16429 |
| Bug Report Page（缺陷报告页面） | https://issues.apache.org/jira/browse/HBASE-16429 |
| Bug Type（缺陷类别） | 本地通信死锁 |
| Bug Description（缺陷描述） | 对于HBASE-16429，先介绍一下相关背景。在HBase的RegionServer中存在一个环形缓冲区ringbuffer，普通的消息处理线程在处理消息时会通过FSHLog不断地向ringbuffer添加元素，如果ringbuffer已满的话，就调用LockSupport.parkNanos()进行等待，直到收到ringbuffer中有空余位置的信号时才继续执行；RingBufferEventHandler线程是ringbuffer的唯一消费者线程，它不断地读取ringbuffer中的元素并根据元素的信息进行相应处理，读取元素之后它会通知相应因ringbuffer已满而正在等待的线程继续执行，不过RingBufferEventHandler线程需要先等待一个SafePointReleaseLatch门闩锁对象latch的计数器变为0后才能进行对ringbuffer的元素读取操作；RollWriter线程rollWriter在启动后会先向ringbuffer添加一个元素，然后调用latch的countDown()方法使得计数器减1至0。  上述就是HBASE-16429的发生背景，而其发生过程如图1所示：（1）在初始时latch的计数器被设置为1；（2）多个普通消息处理线程不断向ringbuffer添加元素导致ringbuffer填满；（3）此时rollWriter启动并向ringbuffer添加元素，则它将进入等待状态（注意此时rollWriter还没有成功向ringbuffer添加元素，更没有将latch的计算器减为0）；（3）而ringbuffer的唯一消费者线程RingBufferEventHandler在试图获取ringbuffer中元素时，需要先等待latch上计数器变成0，但能够将其变为0的rollWriter正等待RingBufferEventHandler从ringbuffer消费元素。这样就发生了由本地线程rollWriter与RingBufferEventHandler、代表ringbuffer有空余位置的消息和门闩锁latch的计数器变成0的信号构成的本地通信死锁。 |
| Bug Diagram（缺陷示意图） | 图1 HBASE-16429本地通信死锁发生过程示意图  Fig. 1 A diagram to show how the local communication deadlock HBASE-16429 happens |