Fiber 多条件唤醒的问题和解决方案

在使用 timed_yield()时，response 如期返回，receive fiber 会向完成队列投递一个 wakeup 消息。但如果在 fiber 被唤醒前，正好 timeout，timer 也向完成队列投递了一个 wakeup 消息。于是，完成队列中存在两个 wakeup 消息，指向同一个 fiber。第一个 wakeup 消息会得到正确的处理，唤醒 fiber。但第二个 wakeup 消息所对应的唤醒点已经不存在，结果就会唤醒这个 fiber 后续的 yield()，造成整个执行序列的混乱。

同样的问题也会发生在 timeout 触发，但 response 在 fiber 被唤醒前返回的情况下。

这个问题本质上是：在一个 yield()点上，对应着多个唤醒条件。这些唤醒条件中只需一个条件得到满足，便可以唤醒 fiber，而其他条件都将失效。

目前，有两种方案可以解决这个问题。

方案一：完成队列中一个 **fiber** 最多只能有一个 **wakeup** 消息

只要保证任何时候，完成队列中最多只有一个 wakeup 消息用于唤醒 fiber，便可以解决问题。

最简单的办法就是在处理一个 wakeup 消息之前，在完成队列中遍历，并删除其他唤醒同一个 fiber 的消息。但这种方法会导致性能严重下降，是不可接受的。

一种改进的方法，就是当 timeout 发生的时候，设置一个标志。在处理 response 触发的 wakeup 消息时，检测这个标志。如果标志被设置，再扫描完成队列，剔除 timeout 的 wakeup 消息。如果 timeout 先到，则反过来。

这种情况发生的几率较小，因而后一种方案对性能的影响有限。

方案二：采用 **FiberSerialNo** 匹配 **wakeup** 消息和 **yield()**

从另一个角度出发，可以提供一个更加完善的方案。multi-wakeup 实际上是多个唤醒条件对应一个唤醒点。如果 wakeup 消息可以检测出 fiber 是否还停留在对应的唤醒点，那么便可以将其唤醒。否则，说明 fiber 已经被唤醒，wakeup 消息可以被忽略。

最简单的实现，就是在 fiber 数据结构上增加一个 serial no.。fiber 每唤醒一次，就给这个 serial no 加一。这样，一个 fiber 上的每个唤醒点，都有各自的代码。在构造 wakeup 消息时，从 fiber 中取得这个参数，写入消息。表明这个 wakeup 消息所对应的唤醒点。

scheduler 在处理 wakeup 消息时，对比消息的 serial no 和 fiber 的 serial no。如果一致，便唤醒 fiber；否则，就抛弃消息。

此外，也可以在 post wakeup 消息时检测 serial no。如果不同，就不 post。但这种方案存在两个问题。首先，如果消息不 post，那么必须立刻回收它的内存。这样的话，造成跨线程的内存释放。其次，如果调用点的另一个 wakeup 消息已经在完成队列中等待执行。此时 fiber.serialno 的值同 wakeup.serialno 的相同，可以 post。于是，完成队列中依然出现了两个 wakeup 消息。只有将 fiber.serialno 锁住，才可能确保完成队列中只有一个 wakeup 消息。这不仅增加了复杂性，也对性能产生较大影响。

而且，在完成队列中同时出现一个 fiber 的多个 wakeup 消息的情况基本上是小概率事件，不会对系统性能产生很大影响。

此外，serial no 方案同时也解决了 yield()/wakeup 匹配的问题，消除了 wakeup 错乱的潜在问题。

综合起来，选择 serial no 方案，用 scheduler 检测 yield()/wakeup 匹配。