工作者机制（Worker Mechanism）

Vapula框架使用工作者机制优化任务的运行调度过程。

任务（Task）由用户通过运行时驱动（Driver）构造。

每个任务具有一个虚拟栈（Stack）。

任务和栈的信息依照库（Library）的指定方法（Method）的定义。

任务本身具有启动（Start）、暂停（Pause）、恢复（Resume）、停止（Stop）4个控制方法，用户可以调用这些控制方法操作任务。

用户负责任务对象的内存分配与释放。

工作者（Worker）是所有任务的调度中枢。

工作者构造后，本身不具备执行能力。

工作者具有与任务的控制方法同名的方法，但是参数不同。

任务的控制方法内部会完善地实现任务与工作者的交互过程。

工作者内部具有2个线程表，

线程表分为忙表（Busy）和闲表（Idle）。

工作者提供上线（Online）和下线（Offline）两种总控方法。

**上线**

工作者计算前线容量，并构造总数等于活动容量的线程，

将这些线程分别与每个逻辑CPU绑定，设置这些线程为活动线程。

**下线**

工作者执行内部统计，终止所有执行线程并销毁，然后清空任务队列。

**任务的控制方法**

**启动**

任务调用工作者的启动。

工作者检测任务是否已经启动，已经启动则返回。

如果未启动该任务，工作者会尝试从闲表获取一个空闲线程。

如果没有得到空闲线程，则创建临时线程。

安排该线程执行该任务，并将该线程置入忙表，置出闲表。

**停止**

停止方法具有时控参数。

时控为0，表示用户要求立即强行停止任务。

任务会调用工作者的停止。

工作者获取对应线程，检查是否是临时线程。

[如果是活动线程，工作者创建一个同属性的活动线程，置入闲表]

调用线程的终止并被等待，工作者将销毁当前线程。

设置上下文状态为空闲，返回为终止。

时控不为0，表示用户要求通过控制码通知任务主动停止。

任务会设置控制码，等待时间时，检测任务是否主动停止。

超时如果没有停止，则按照时控为0处理。

**暂停**

暂停方法具有时控参数。

时控为0，表示用户要求立即挂起任务。

任务会调用工作者的挂起。

工作者获取对应线程，检查是否是临时线程。

[如果是活动线程，工作者创建一个同属性的活动线程，置入闲表]

调用线程的挂起。

设置上下文状态为暂停。

时控不为0，表示用户要求通过控制码通知任务主动暂停。

任务会设置控制码，等待时间时，检测任务是否主动暂停（Pause）。

超时如果没有暂停，则按照时控为0处理。

**恢复**

任务调用工作者的恢复，将对应线程从可能的挂起中恢复。

如果没有挂起，工作者会返回false。

任务再设置控制码等待。

注意

当任务没有执行完成时，务必不要将该任务再次推入队列。

因为工作者机制会导致这个任务具有被多次并行调度的可能，

但是虚拟栈只有一个，并行运行一个任务会导致数据破坏。

无论用户使用什么操作系统，同时可以并行调度的实际任务数量是有限的，限制即CPU逻辑核心数量。无限制的多线程调度通常是由操作系统模拟的。

所以应当尽量保证一个软件系统并发调度等于逻辑核心数量或倍数的任务。

活动容量等于当前Vapula执行环境的逻辑CPU数量与调度负载系数的乘积。

调度负载系数是大于等于1的正整数。