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1个任务队列和2个线程列表，

线程列表区分为前线（Head）和后备（Back）。

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，并构造总数等于前线容量的线程，

将这些线程分别与每个逻辑CPU绑定，设置这些线程为前线线程。

**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。

任务的执行分为两种，一种是即时执行，一种是排队执行。

即时执行

工作者创建一个后备线程，为该线程分配一个指定任务。

排队执行

工作者将任务推入任务队列，等待线程申请分配任务。

**任务的控制方法**

**启动**

启动具有时控参数，

时控为0，表示即时执行任务。

任务会调用工作者的启动（即时）。

（工作者检测是否队列中存在该任务，存在则移除。然后创建后备线程，安排任务运行。）

时控不为0，表示等待排队执行任务。

任务会调用工作者的启动（排队）。

等待时间时检测任务启动，

超时没有启动则调用工作者的启动（即时）。

**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。

（工作者停止会终止并等待成功，会Block当前线程）

然后设置任务栈为空闲。

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。

（工作者会终止线程，设置线程为濒死，防止线程在终止过程中执行完成并调度一个有效任务；当线程执行自身任务后，检测到自己是濒死的，会关闭线程句柄。如果线程原先是前线，则工作者会提升一个后备线程或创建前线线程。）

**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。

（工作者会挂起线程。如果线程是前线线程，工作者将线程降级为后备线程，并提升或创建前线线程。）

任务会设置状态为冻结（Freeze）。

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。

所以应当保证一个软件系统只并行调度等于逻辑核心数量的任务。

排队执行默认即采用

前线容量等于当前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。