根据小论文描述负载均衡策略的实现细节

3-4

以太坊的Geth客户端在Docker中的运行时，其内存占用率和CPU占用率会随着操作改变，同时影响了该服务节点的服务能力。本文设计合适的负载均衡调度策略来使得服务能力最大化。将运行中的容器的工作状态表示为一个参数向量，经实验验证，运行Geth客户端的Docker容器的内存占用率和CPU占用率对其服务能力影响较大，因此考虑将这两个参数作为建立容器状态空间依据，基于此，可以在一个二维平面映射服务节点。在本文服务系统中，服务节点的状态大致可以分为两类，一类上述所选资源消耗度较小，其在二维平面的投影点距离原点较近，而第二类上述所选资源消耗较多，因此其在二维平面的投影距离原点较远。假定服务节点总数为n，在任务分配的过程中根据服务节点的状态合理考虑分配模型。首先，计算n个服务节点在未执行任务的初始状态和执行任务后服务能力较差的状态。首先，假定二维状态平面内的服务节点投影点集合为



其中，x1和x2是服务节点的内存占用率和CPU占用率的归一化指标。

读取n个服务节点分别在两种状态下的位置信息，并根据该信息求取到原点的距离，分别根据其距离求取平均值如下公式所示：



 (1)



 (2)

根据以上公式计算所致，状态平面中服务节点的分布情况大致如图所示：



图5 状态平面中的服务节点分布

计算以上平均值并以之做状态分界线：

 (3)

为评估以太坊服务系统的整体服务状态，定义健康参数作为负载均衡的健康参考。首先，假定n个服务节点，在二维状态平面中的服务节点的分布情况是有n1个节点在上述分界线以内，n2个节点分布于分界线以外，然后定义健康参数为：

 (4)

其中n1+n2=n，因此简化该式为：

 (5)

根据以上定义可知，当健康参数大于等于0时，服务节点的整体服务能力较强，且该指数越大说明服务能力越强。反之，如果该指数小于0，则说明服务能力较差。

本文将n个独立的任务分配到m台容器节点上，m<n；根据本文业务场景，我们将n个任务分为读任务与写任务，其中写任务又包括发布合约的任务、调用合约的任务、和操作账户的任务。经过实验验证，在n个任务集合中写任务不会占用节点较多资源，读任务会占用容器节点资源。假设任务集合为，为第i个子任务。任务由参数表示即

 (6)

公式中，是任务所需的内存大小，是任务处理时需要的IO内存大小。

m个容器资源可以表示成，表示第i个容器，其属性向量为

 (7)

公式中，是容器剩余可用内存，是容器现有可用IO内存。

本调度模型的目标是让系统承受尽量大的并发，函数与约束条件为

max(n)；

约束条件为：

 (8)

即仅在约束条件满足的时候才能将任务分配给相应的节点。

在区块链环境下，任务可以分为读任务RT(Read Task)和写任务，读任务不消耗容器资源因此优先使用第二区域的容器，在第二区域容器均无空闲时选择第三区域的容器。对于写任务，可以分为发布合约任务DT(Deploy Task)，发布合约任务较少，操作账户任务MAT(Manage Accounts Task)次之，调用合约任务MCT(Manage Contract Task)较多。本文根据任务出现的频率对其先后执行顺序，优先执行处理出现次数较少的任务。

任务优先级模型具体描述如下：

读任务与写任务分开执行；

读写任务可能在第二区域冲突，此时因为读任务耗时短且不消耗系统资源，优先执行；

读任务按时间先后顺序执行；

写任务分为发布合约任务，操作账户任务和调用合约任务，其在实际业务场景中出现频次依次增加，同时重要性会减小，因此优先级为降序排列；

具体任务优先级图示如下：



图6 任务流优先级图示

任务的优先级决定了任务的调度顺序，在本文中，我们根据容器的健康参数将容器分为三个区域，由此，我们可以将读任务交给第二区域的容器处理，写任务交给第一区域的容器处理。第三区域的容器，进行定时回收重启处理。在各个区域内部，本文结合区块链场景下的任务的性质以及资源的使用和分配情况，构建动态优先级。

因为容器的状态影响其执行写操作，因此我们将容器按状态分为两类，具体映射到状态平面空间为I区、II区。以公式(3)得到的数值为分界线。

我们将容器状态平面分为两个部分，I区代表安全区，II区代表非安全读区。

图7 容器状态平面分区示意图

任务分配模型描述：

对于写任务，优先分配到I区，当I区没有节点时分配到II区；

对于读任务，优先分配到II区，当II区没有节点时分配到I区；

本文设计的面向区块链微服务化场景下的任务调度算法描述如下：

将任务根据任务的读写性质进行分类并且按时间排序；

将任务依次动态的分配到适合的区域中的容器中；

更新容器的状态信息，动态将任务分配到其中；

将第三分区中的容器依次进行重启操作，回收资源；

检查健康状态参数，若健康状态参数为负数，则暂停如任务分配，等待知道健康状态为正数则继续分配任务；

返回流程开始下一个任务的调度；