# 第三章 基于状态反馈负载均衡策略

本章首先介绍基于状态反馈的负载均衡策略，其中包括负载均衡依据、任务调度模型、任务优先级模型和任务分配模型。

## 3.1 基于状态反馈的负载均衡策略设计

### 3.1.1 负载均衡依据

以太坊的Geth客户端在Docker中的运行时，容器的工作状态可以表示为一个参数向量，如a = (a1, a2, …,an)。当任务分发到该容器执行时，容器的状态会改变，因此，通过参数向量的变化可以看出容器当前的服务能力和与工作状态。在Geth客户端运行的容器中，CPU和内存的占比受服务任务的影响较大，因此本文选取容器的这两个状态参数建立2维平面，然后将Geth服务节点按照对应参数映射到该平面。

假定Geth服务节点的数量为n，在任务分配调度时考虑CPU和内存的状态两个因素。CPU和内存在对Geth所服务的任务的重要性不同，因此假设它们在调度系统中的权重为w1和w2，显然w1+w2=1。则Geth服务节点在2维平面的点集为

 (3.1)

Geth服务容器在二维平面投影的重心位置反映了容器集群的整体负载，如果服务节点的位置均聚集在重心附近，则说明该系统的负载均衡性良好；否则，则说明负载均衡效果差，因此服务节点的分散投影说明了负载并不均衡。

综上所述，可以根据重心定义一个直观的负载均衡参数作为负载均衡依据，定义其为健康参数LBH。假设Geth服务节点的数量为n，其在2维状态平面投影的重心为G(X1, X2)，则各服务节点的投影到重心的举例平均值为

 (3.2)

由上式可知，当Geth服务节点中未执行任务的节点与因为执行任务导致节点满载的节点各占50%时，以太坊服务平台的整体负载均衡性最差。Geth服务节点的CPU和内存两参数在二维平面中的重心位置为，各服务节点到重心位置的距离的最大值为。考虑各服务节点到重心距离的平均值，作归一化处理后即为负载均衡健康参数LBH

 (3.3)

负载均衡健康参数是衡量以太坊服务平台中的Geth服务节点的负载均衡健康指标，通过该参数可以量化服务平台的负载均衡状况。理想情况下，当所有的服务节点的负载情况相同时，它们在二维平面的投影集中到重心位置，此时健康参数LBH = 0。因此，LBH的理论取值范围是[0, 1]，该健康参数越小，服务平台的负载均衡性越好。

### 3.1.2 任务调度模型

本文设计的调度模型首先假定任务为将m个任务调度分配到n各服务节点，其中n<m；基于私募股权交易的应用业务场景，调度任务可根据任务类型分类为读写两种任务，对于写任务，可进一步划分。由于读任务仅通过服务节点中的客户端调用相应的API，故资源消耗较少，而写任务需要等待数据打包、验证等操作，因此耗时更长，资源消耗也更多。假设任务集合为，其中为第i个子任务。任务由参数表示即

 (3.4)

在上述公式中，是任务完成所期望的花费，代表任务的大小，和分别代表完成任务所需的内存和CPU占用的需要。

m个容器资源可以表示成，表示第j个容器，其属性向量为

 (3.5)

公式3.5中，是容器处理任务的花销，是容器当前的资源能力，和是容器当前剩余内存和CPU的占比。

私募股权平台的任务在Geth服务节点上运行的时间可以表示为

 (3.6)

任务在服务节点上的期望完成时间为

 (3.7)

其中为任务开始时间，任务完成的总时间可以表示为

 (3.8)

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min(RT)；

约束条件为：

 (3.9)

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。

### 3.1.3 任务优先级模型

在私募股权业务场景中，根据任务对Geth服务容器的影响，将任务分为读写两大类。读任务仅读取信息，因此对容器的内存和CPU资源消耗较少。写任务需要Geth进行发起交易、打包数据到区块等操作，因此对内存和CPU资源的消耗较多。私募股权业务场景考虑到的业务操作及分类如下表3.1所示。

|  |  |  |
| --- | --- | --- |
| 任务分类 | 功能分类 | 详细功能 |
| 读任务 | 连接管理 | 获取某账户币值 |
| 获取当前主账户 |
| 获取本客户端所有账户 |
| 获取区块数量 |
| 账户管理 | 创建新账户 |
| 展示当前所有账户 |
| 解锁账户 |
| 写任务 | 交易管理 | 发送交易 |
| 获取交易哈希 |
| 通过交易哈希获取交易内容 |
| 合约发布 | 发布智能合约 |

任务优先级模型算法具体描述如下：

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3.4 任务流优先级图示

### 3.1.4 任务分配模型与算法描述

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3.5 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；