协同分配

1. 计算任务：M个相同计算任务(M>N)，组成队列
2. 算力资源：cpu核数
3. 系统结构：1 服务器——N个客户端：每个客户端对应一个容器，有不同大小的算力资源配置
4. 两层算法：
    1. 预分配算法：根据容器设备和任务配置情况，给出最合适当前任务分发的设备
    2. 动态分配算法：根据设备资源使用情况，动态调整容器资源配置
5. 系统设计
    1. 1服务器（**server.py**）——N个计算节点（N可配置）
    2. 计算任务(**cnn.py**)：cifar10 CNN图像识别，可配置其运行时间，从而控制任务规模
    3. 节点资源可用算力v：当前节点可用算力每隔T时间根据正态分布N(m,A)在[cpu-1, cpu+1]中变化（m,A,T可配置）
    4. 算法设置：
        1. 静态调度Static Assign（**task_schedule.py**）：系统将计算任务平均分给N个计算节点，在每个计算节点预分配固定算力用于执行任务，计算节点执行完毕后返回。
            1. **base_assign**：将任务按照顺序平均分配给各个节点
            2. **random_assign**：将任务随机分配给各个节点
            3. **min_min**：将当前任务优先分配给之前任务执行总时间最少（最早完成之前任务）的节点
        2. 动态调度：系统将计算任务平均分给N个计算节点，每个计算节点根据当前可用算力动态调整使用算力（考虑算力调动态分配的过程损耗），计算节点执行完毕后返回。
            1. **cpu_change**：模拟cpu资源变化的情况，并当没有动态调度时，只能减少cpu数量，无法扩张
            2. **cpu_change_ca**：模拟cpu资源变化的情况，并当没有动态调度时能够根据给定大小伸缩容器cpu数量
        3. 协同分配：系统根据当前算网状态，依次将任务按最优适配分配给对应计算节点，每个计算节点根据当前可用算力动态调整使用算力（考虑算力调动态分配的过程损耗），计算节点执行完毕后返回。
    5. 性能分析——评价指标
        1. 任务完成数：一定时间段内执行计算任务的总数量
        2. 算网资源利用率：指全网可用算力资源的使用效率
        3. 应用执行延迟（效率）：指M个计算任务的完成延迟 平均?

待解决

1. 当各节点在不同设备上时，docker daemon控制容器配置
2. 动态分配算法研究
3. 性能分析实现
4. gpu算力动态分配及k8s合理性