Schedule Lab 实验报告

前置要求

首先是理解Schedule policy.h里各个变量代表的是什么，和数据的特征以简化思路和代码量

1.ktimer和arrivaltime&ddl&time是两个时间体系，不能合并

2.不支持预约操作，即arrivaltime!=time的情况

3.任务是交互性质的，工作时长是未知的，但同一个时间的任务一般只有一个

4.IO后一定有CPU，每个任务一定是CPU打头

5.kIorequest,kIoend都是建立在已有的任务上，前者是CPU任务做完后请求IO，后者是IO任务做完后请求CPU

6.一般的调度算法里，ktimer是没有用的，无需理会，~~我也暂时无法理会~~

7.kTaskArrival是每一个任务的第一次出现，kTaskFinish是每个任务最后一次出现

尝试借鉴高响应比优先调度算法HRRN

第一步本来想写FIFO的，发现这玩意根本过不了

就直接采用对任务列表排序的思路，建立一个CPU任务列表，一个IO任务列表

一个IO正在执行时是不能更换的，一个CPU执行时是可以被替换的

高优先级标记为1，低优先级标记为0

排序依据是 (优先级-2) * (到达时间-截止时间)

由于两个都是负数，优先级越高的排在越前面，留存时间越少的排在越前面

每当有新事件出现后都如此做，每次cpu和io都取各自列表里第一个即可

Scores: 87

尝试优化排序依据

1.(优先级-2) * (当前时刻-截止时间)

Scores:63

原因：由于不知道任务运行所需时长和任务当前处于时长的哪一个时刻，该方法反而下降

2.(优先级-x) * (到达时间-截止时间) , $2 \leq x \leq 100$

Scores:87

3.(优先级-x) * (到达时间-截止时间) *(1+当前任务执行cpu次数) * (1+当前任务执行io次数*2)

Scores:68

不知道原因，明明执行io次数越多可以认为该任务在恶意占用io资源，但是反而反馈情况更差了

4.对于已经超时的任务，将权值设为inf，因为这部分任务分已经得不到了，不能再浪费其余的任务

Scores:89

添加新的思路

使用多级反馈队列调度算法

从第一个队列到最后一个队列，优先权从高往低，时间片长度从成倍递增

1.使用两个队列

对于cpucnt较多的任务放进优先级较低的队列里，每次都优先从优先级较高的队列里做任务。队列内部的排序方式采用以上述最优的方式

仍然表现不好

Scores:89

2.使用三个队列

仍然表现不好

3.观察得分发现测试点三表现很差，观察数据性质是随机出现，时间都很宽裕

于是采用rand的一个队列的方式来决策

仍然表现不好

4.最终放弃了