## 项目概要

市面上存在一种非易失存储器（NVM），它具有非易失、低能耗等优良特性，读写速度不对称，读远快于写，但写入次数有限（需要像SSD一样考虑磨损均衡），这种存储器当电流关掉后,所存储的数据不会消失。我们设想在传统云计算服务器PM中部分内存由NVM代替，以充分发挥其优良特性（在实际计算的过程中大多情况都是数据一次性写入，按需读取，这也十分契合NVM本身特性）。

基于假设，我们构建实验测试环境，在一台PM上有m个CPU slots，且NVM上也被划分为n个slots（存储空间有限）。一批任务Task随着时间的推移依次进入等待处理。但考虑CPU的负载，基于某种策略的情况下，一些任务可能在其本身需要完成的截止时间之前无法完成，这时就把此任务拒绝，相反则接受处理。最终用总的效益U来衡量此PM在指定时间内的运算效率。此外，在给定时间结束之后，还可以用NVM命中率Hr，NVM写入次数S，任务接受率Ar来评估NVM以及配合调度策略的执行效率。

接下来，我们细化其运转流程，并在关键数据上给与量化。首先，我们先指定Task中包含的属性以及含义：

Task(taskID,arriveTime,deadline,requiredData,exeTime,startTime,endTime,type,status,pu)

taskID：int，标识任务的唯一ID属性。

arriveTime：int，任务的到达时间。

deadline：int，任务的最晚完成的截止时间。

requiredData：int，所需要数据的唯一标识ID。

exeTime：int，需要的执行时间。

startTime：int，开始执行的时间。

endTime：int，执行结束的时间。

type：boolean，表示交互任务还是常规任务，交互任务为1，常规为0。

status：int，表示任务的状态，执行为1，等待为0，结束为-1。

pu：int，表示单位时间的收益。

下面指定Job的属性及其含义：

Job(JobID,tasks,arriveTime,deadline,endTime)

jobID：int，标识job的唯一属性。

tasks:List<Task>，表示job中包含的所有task。

deadline：int，job总的截止时间。

endTime：int，执行结束时间。

下面指定Data中包含的属性及其含义：

Data(dataID,hotness,isInteractive,status)

dataID：int，标识数据的唯一属性。

hotness：int，表示数据的热度（阶段不访问递减）。

备注：hotness = H1 + H2\*iLable 如果数据被使用，则增加热度，参数待定，iLable表示被交互式任务所调用。

hotness-=1 热度值随时间的推移递减

当hotness<CH的时候，迁移数据至Disk，（CH为全局变量，待输入）

isInteractive：boolean，表示数据是否会被交互式任务访问到，会访问到为1，不会为0。

status：boolean，表示数据是否在NVM之中，在为1，不在为0。

type：int,表示数据的自定义类型，分为一二三四五，5种类型，依次由整形数据表示。

备注 ：1-5类表示在NVM中的数据

第一类表示正在被调用的数据和在队列中交互式任务所需要的数据。

第二类表示在队列中的其他任务所需要的数据。

第三类表示之前交互式任务用过的数据。

第四类表示队列其他job需要的数据需要的数据。

第五类表示其他数据。

以下为在Disk中的数据

第一类表示在队列中交互式任务所需要的数据。

第二类表示在队列中的其他任务所需要的数据。

第三类表示之前交互式任务用过的数据。

第四类表示队列其他job需要的数据需要的数据。

第五类表示其他数据。

对于PM的状态的描述分为两个部分，CPU状态描述和NVM状态描述。先介绍CPU状态的描述：

CPUSlot(slotID,time,taskID)

slotID：int，表示CPU中某一slot的唯一标识（共有m个slots）。

time：int，表示某一时间点的时间属性。

taskID：int，表示某一时间点此CPU中此slot中正在执行的任务ID。

对于NVM状态描述如下：

NVMSlot(slotID,time,dataID)

slotID：int，表示NVM中某一slot的唯一标识（共有n个slots，但空间大小有限）。

time：int，表示某一时间点的时间属性。

dataID：int，表示某一时间点NVM中此slot中存储的数据ID。

在介绍完关键数据属性及其意义之后，接下来我们考虑任务完成后的效益（utility）。首先，我们接收的任务分为两种类型，交互式任务以及普通任务。交互式任务要求短时间内必须给出执行结果，超出，效益直接为负（如图2所示）。而普通任务对执行时间的要求就没有那么苛刻，在我们需要前给出结果即可（图3所示）。因此考虑两种不同的情况，效益函数设计如下：



图2 交互式任务效益函数图 图2 普通任务效益函数图

根据函数关系图，得到函数关系如下：

u = w0 , x<=α

-a(x-α) , x>α 表示交互式任务效益函数。

u1 = w1-△T\*[x/T] 其中△T=w1-w2 表示普通任务效益函数。

再来考虑任务的接受与拒绝策略，在若干个CPU的slot之中可能同时并行执行着很多不同的任务。在每个任务达到之时会有自己的任务结束时间（deadline），去除掉任务本身的执行时间（exeTime），不难算出任务最晚的调度时间（latSchTime）。等于结束时间减去执行时间。这个时间对于决定任务的取舍至关重要，如果在这个最晚调度时间之前有空余的slot可利用，则接受任务，否则拒绝。



当上面公式左边计算的结果小于latSchTime的时候，就接受此任务，相反则拒绝。（其中P表示数据在NVM之中的期望）。

文件存储与迁移策略：

在上面的Data定义中，把两个位置的文件分为了五类，表示存储时候的优先级。数据本身存储在Disk之中，在开始的时候，数据按照类别的大小从小到大依次迁移在NVM之中，存满为止。运行过程中，当Disk中存在一二类数据的时候，立马迁移至NVM之中，依次替换从大到小类别的数据，最多替换至第三类别，不可替换一二类别。当Disk中存在第三类别的数据的时候，与NVM之中第三类数据比较hotness值大小，哪个大，哪一个存储在NVM之中。

文件迁移（数据初始化在Disk中的类型都是第三类，初始化迁移至NVM时候，都为第五类）

文件迁移发生在：

1. 有任务进入等待队列的时候（hotness会改变，拒绝时候也会改变）。
   1. 如果需要的数据在Disk，且为交互式数据，迁移至NVM,3-1
   2. 如果需要的数据在Disk，且为普通数据，迁移至NVM，3-1
   3. 如果需要的数据在NVM，且为交互式数据，1
   4. 如果需要的数据在NVM，且为普通数据，2
2. 有任务进入运行队列的时候。

需要的所有的数据类型都变为1，且迁移至NVM

1. 有任务完成的时候。
   1. 如果是交互式任务需要的数据，变为3
   2. 如果是普通任务需要的数据，变为5

备注： 在迁移至NVM的时候，需要考虑NVM是否已经存满，如果存满，把NVM中的数据类型从大到小依次迁移至Disk