**新调度算法思想**

**【数据结构】**



**【新调度算法流程图】**



**【新进程调度流程图】**



**【关键数据】**

1.常量REFRESH：定值，用来对页表进行大更新；

2.全局变量T：初始值为0，每进行完一次进程调度时，T++；

3.左侧hash表：发生缺页中断后，进程号按照模n运算，以链表形式插入到相应位置；

4.初始TTL：页的生存周期；

5.TTL：TTL = 初始TTL + 当前时刻T + 缓冲变量值；

6.缓冲变量值：初始值为0，用来在大更新时完成对TTL的赋值。

**【关键算法思想】**

1.hash表和链表

对于其他页表，在查询是否命中时都是对表进行遍历，因此使用hash表的初心是为了提高查找效率，而为了减少冲突甚至是避免冲突的发生，又在此基础之上引入了链表。

2.大更新

在进程调度之前，会判断T mod REFRESH 是否0，若为真，则对页表进行大更新，因为对于这种数据结构，每次页的命中和调入，都是相互独立的，单独通过命中页和调入页无法对整个页表中的过期信息进行删减，于是引入大更新这个概念。

3.缓冲变量值

当未发生大更新，某页已超过生存周期时，在一次进程调度中又命中该页，缓冲变量值+1。

引入缓冲变量值这个概念，有以下优点：

1）、当出现上述情况时，有可能是该页恰好在这一时期频繁使用，在下一次大更新时，即使其生存周期结束，仍将其保留，TTL在初始TTL的基础上加上缓冲变量的大小。

例如当初始TTL为3，REFRESH为10，当大更新开始前，某一页在生存周期外被命中两次，缓冲变量=2，那么在大更新时，TTL=5。

2）、通过这种TTL分配方式，可以使不同页的生存周期不同，一定程度上实现了灵活分配。

4.对于TTL的赋值

TTL = 初始TTL + 当前时刻T + 缓冲变量值

因为T的值在每次进程调度后都会增加，所以TTL也要随着增加，保证TTL和T的差等于初始TTL的值，在大更新后，TTL = TTL - T，同时T的值会清零，保证不出现较大的数值。

5.算法优点

1）、每个页之间互相独立，对单个页的修改不会影响其他页的改变，通过命中页和调入页无法对整个页表中的过期信息进行删减。这就使得一些过期的页仍可保留一段时间，当这些过期页被访问时，不用再执行缺页中断，大大减少了处理时间。

2）、针对过期页，设置了缓冲变量这个概念，缓冲变量可以使得访问频度不一样的页，其生存周期也不同，对于访问频度高的页，TTL也高，反之亦然。这就相当于边调度，边预测，在调度进程的同时，根据每个进程的访问次数，预测并使其被分配的资源更灵活。

3）、使用hash表+链表结构，不用去遍历所有页表项就可以完成页调入的操作。设FIFO算法页表的大小为N，在完成m次进程调用时，其访问次数为m\*N，对于新算法，在完成m次进程调用时，最坏情况下，其访问次数为m+（T-初始TTL）/2 +T，两个式子同时减m，FIFO的访问次数（N-1）\*m，新算法（T-初始TTL）/2 +T，由此可见，新算法相比于FIFO，受m的影响特别小，主要和其本身的性质有关。

6.算法缺点

页表大小虽然受T的限制，但相对于其他调度算法的页表来说，占用的存储空间更大。最坏情况下，某一时刻页表内数据量为（T-初始TTL）/2 +T。

提纲：

先说数据结构，hash+链表

再说进程调度流程（去掉对于缓冲变量的计算部分）

再说调度算法流程（去掉对于缓冲变量的计算部分）

解释缓冲变量

最后说优点：缓冲表；减少缺页中断；动态分配资源；访问次数