操作系统调研报告——CPU scheduling

CPU调度是多道程序操作系统的基础。通过在进程间切换CPU，操作系统可以使得计算机更加高效。

为了比较CPU调度算法，可以采用许多比较准则。比如CPU利用率，吞吐量，周转时间，等待时间，响应时间等，对于大多数算法而言，我们都想要得到最大化的CPU使用率和吞吐量，最小化的周转时间、等待时间和响应时间。但是对于用于不同目的的操作系统，希望得到的效果不同，所以优化的指标也不同，比如说桌面操作系统，研究人员就曾建议最小化响应时间的方差比最小化平均响应时间更为重要。有的操作系统则是要求能够节省更多的能源。为此，我在此次的操作系统技术调研中选择了CPU scheduling 的主题，去探索更多我们教材以外的CPU调度算法。

在教材中，我们学习到，最短作业优先调度算法有着最短的等待时间。但是，这样的算法是理论层面的。我们需要去预测进程下一次的CPU执行时间，对每一个进程保持一个历史记录，预测下一次的CPU执行时间为以前的指数平均。

1. **对多级反馈队列的改进**

通过课内的学习我们知道，MLFQ调度算法存在饥饿的问题。当长作业进入系统时，它将会被放置在优先级最低的就绪队列中，如果有源源不断的短作业进入，则会造成长作业长期得不到执行，产生饥饿。

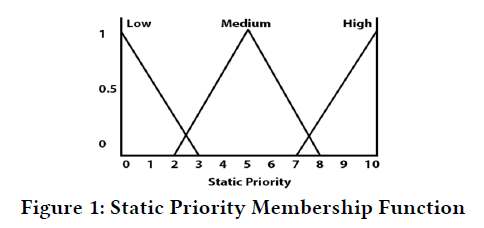
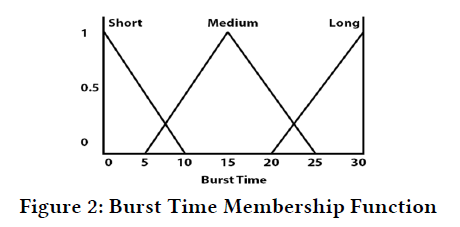
K. Hoganson,和J. Brown提出了在MLFQs在高负载下，通过将Q1中不需要的部分可用时间(否则会下降到Q2及以下)重新分配给最低优先级队列Qn。，以减轻这些进程的CPU时间的不足，尽量避免饥饿。通过仿真这样的调度方式，也发现可以安全地重定向一部分CPU时间，而不会影响队列1的性能。同时，他们还提出了另一种级别的CPU时间重定向方式，称为智能缓解来进一步增强，它将CPU时间重定向到Qn-1和Qn。结果表明，在不影响Qn性能的情况下，系统总体吞吐量得到了提高。

Kenneth E. Hoganson 则将多级反馈队列用在实时操作系统调度中，提出了一种新的方法和算法来解决多级反馈队列周期性地将时间从中间队列重定向到最低优先级队列的不足问题。根据MLFQ 的调度特点，位于高级队列的进程总是优先执行，因此，它能够满足实时的要求。同时，通过将CPU时间重定向，能够在系统处于重负载时能够有效地减少饥饿，同时又不会影响Q1中所需的快速响应时间。

1. **模糊逻辑在各种计算环境中的CPU调度中的使用：**
2. **优先权调度**

模糊推理在解决难以随机建模或分析的问题时具有一定的适用性。因此Bailey Granam 和 Hala E.lAarag 在2019年提出了在CPU调度中使用模糊逻辑。并得出通过仿真得出结论：fuzzy-priority 调度算法比轮转调度和优先级调度算法的平均等待时间低。

他们使用突发时间和CPU调度器可用的静态优先级来提供一个动态调整进程优先级的模糊推理系统。将模糊逻辑应用于优先级调度策略，以产生更有效率的优先级值。静态优先级选择“低”、“中”、“高”三个语言变量，突发时间选择“短”、“中”、“长”三个语言变量，如figure1、figure2所示。最后可以根据隶属函数执行规则，输出一个静态优先级作为成员函数决定的动态优先级值。



1. **公平共享调度**

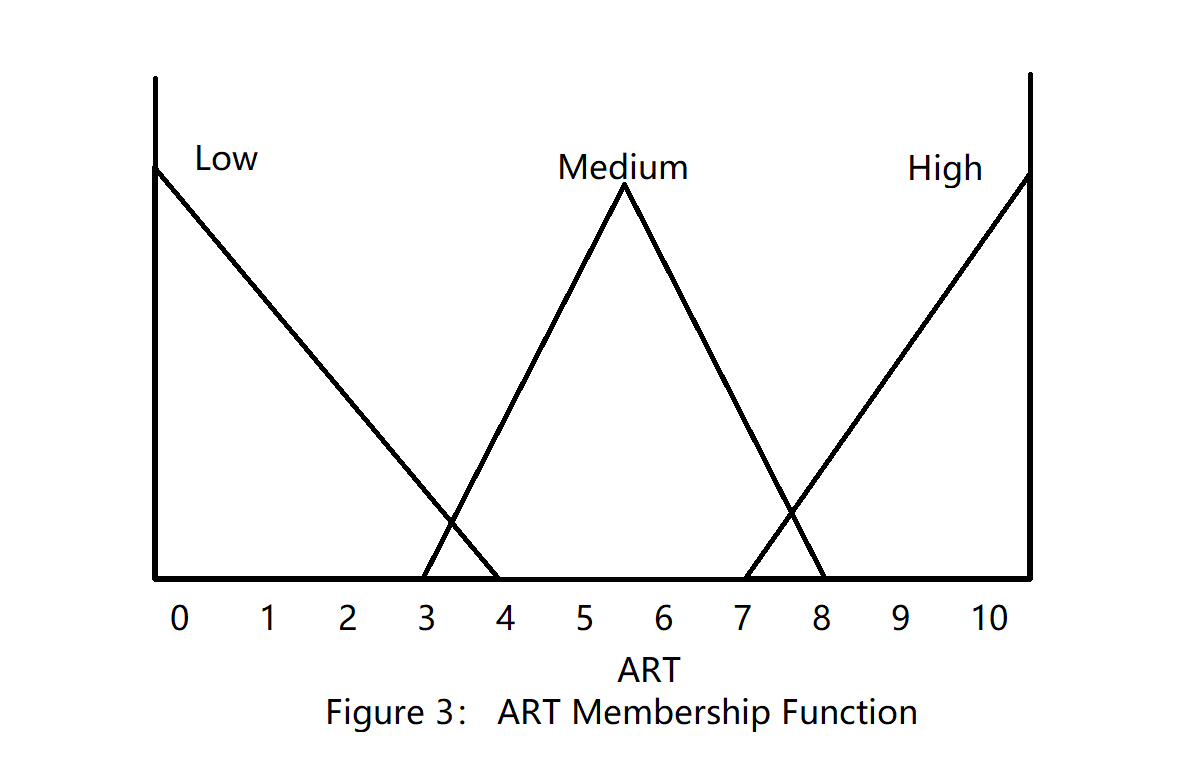
在公平共享调度中，每组进程的CPU时间份额是固定的，然后相应地将每组进程分配给CPU执行。Bashir Alam等人提出了一种FIS (Fuzzy Inference System)，动态地决定进程组的CPU份额。同时，FIS还用于在组内调度，以最小化开销。对于组内调度，公平共享调度算法采用RR算法调度。如果流程没有在给定时间内完成，则会进行上下文切换，将该进程放到就绪队列的末尾，同时将该进程的状态保存在堆栈中，将下一个执行的进程从堆栈中回复，并放入CPU执行。如果运行进程所需的时间比给定时间段稍微多一点点，也会抢占进程并发生上下文切换。这将导致该流程的等待时间更长，并且由于不必要的上下文切换而导致更多的开销。因此Bashir Alam等人提出了FIS来解决该问题，减少等待时间和不必要的上下文切换。

改进后的算法分为两步，第一步是调用FIS来决定进程组的CPU份额。

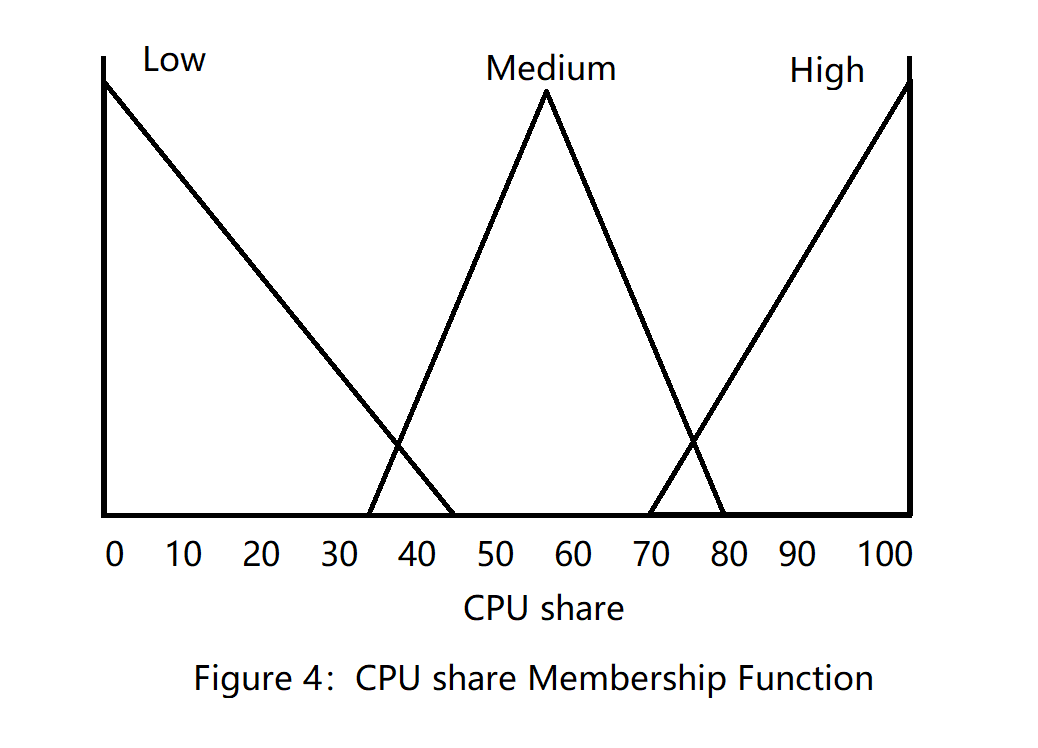
第二步则是在组内调度进程。

首先，第一步，调用FIS确定每个组的CPU时间份额。输入为每个组的进程数N和平均响应时间，输出为每个组的CPU时间份额。这些模糊变量的隶属度函数如下及图3、图4所示:

1. ART：Type- Triangular, Range:1-10, low-[0,2,4], medium-[3,5.5,8],High[7,8.5,10]



1. CPU Share: Type- Triangular, Range: 1-100, low-[0,20,45], medium-[35,55,80] High-[70,85,100]

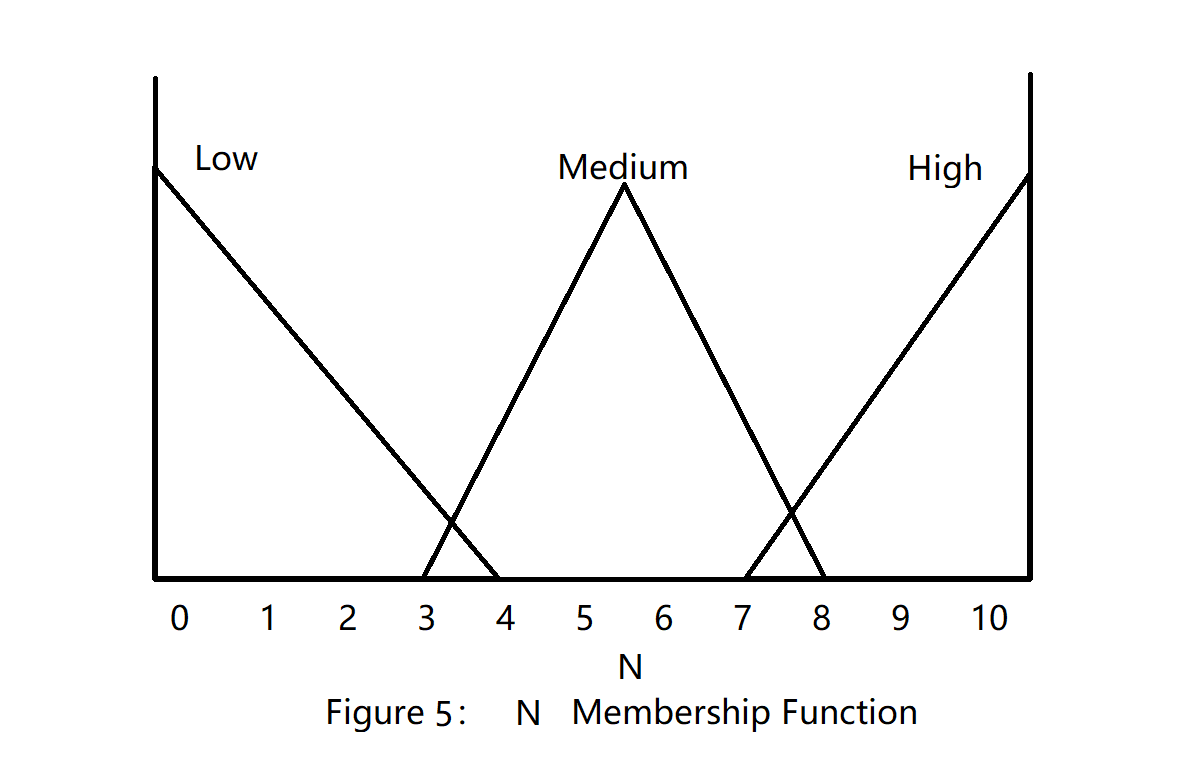


1. Rule base:

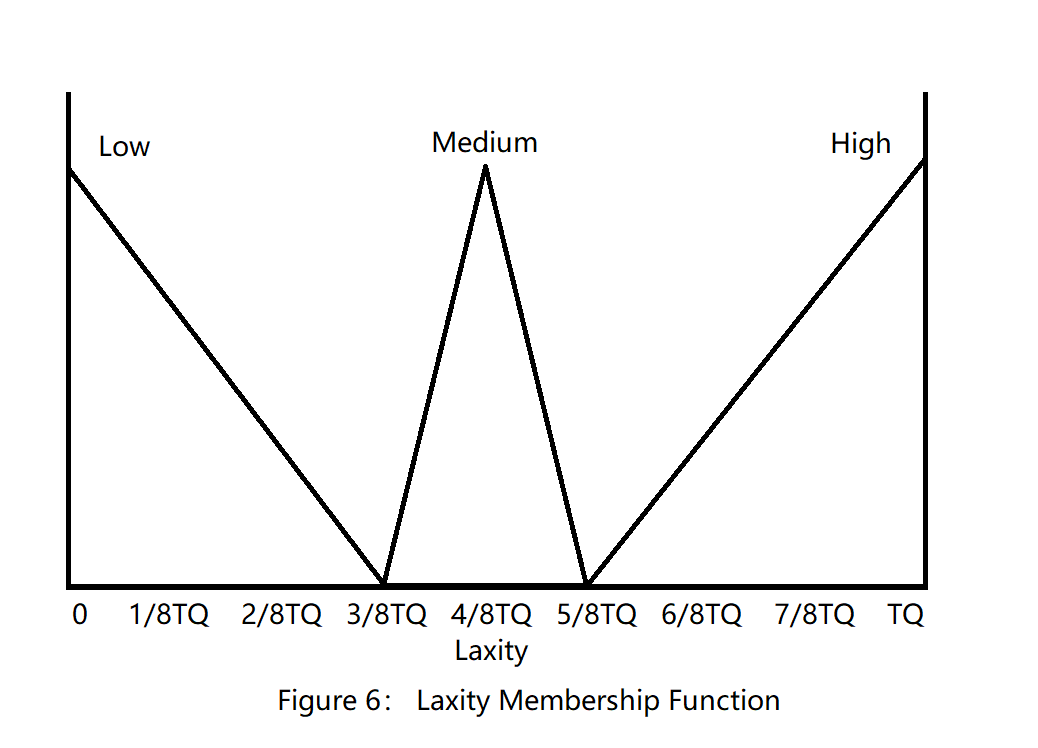
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Group A | | Group B | | CPU Share | |
| N | ART | N | ART | Group A | Group B |
| Low | Low | Low | Low | Medium | Medium |
| High | High | Low | Low | High | Low |
| High | Medium | Low | Medium | High | Medium |

RR调度算法的主要问题是，即使进程还需要少量的时间量来完成执行，但是它还是会在时间段用完时被抢占。如果给进程比时间段稍微多一点的时间，它们就能够在该轮中完成自己的执行。那么不必要的上下文切换开销和等待时间就会减少。任何进程所需的执行时间都不是很清楚，而模糊逻辑适用于模糊的事物。因此，模糊逻辑可以用来判断是否应该执行抢断。该模糊逻辑系统是一个双输入单输出的推理系统。松弛度和就绪进程的数量是输入，抢占状态是输出。这些模糊变量的隶属度函数如下所示。

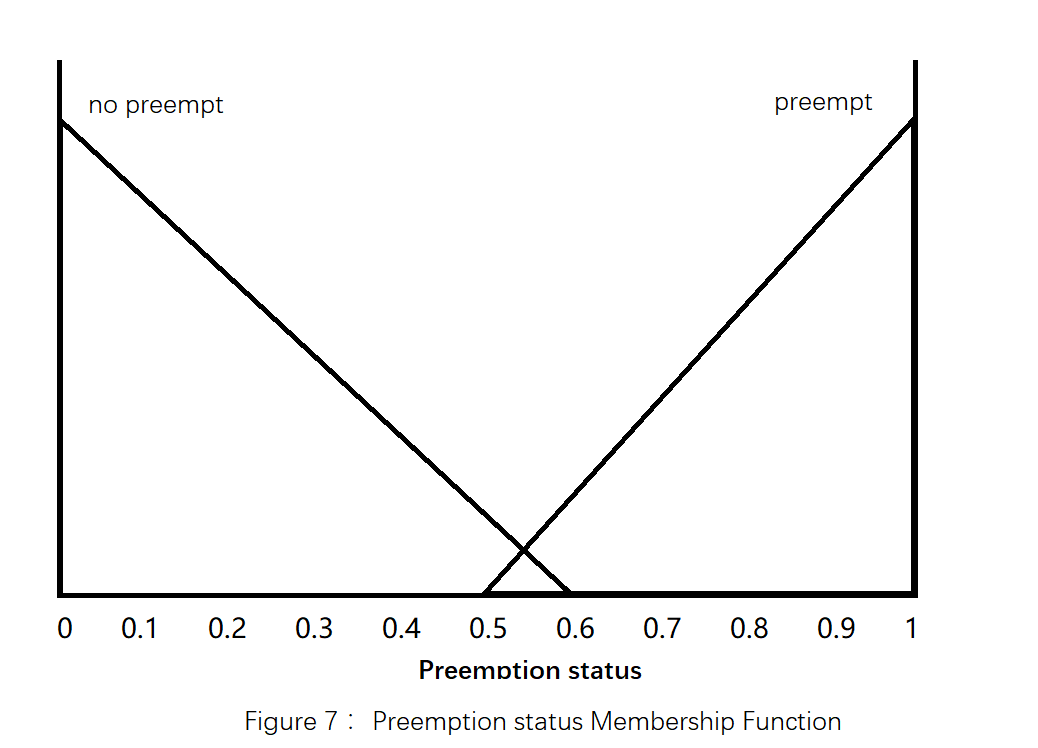
1. N: Type-Triangular, Range:1-10, low-[0,2,4], medium-[3,5.5,8] ,High[7,8.5,10]



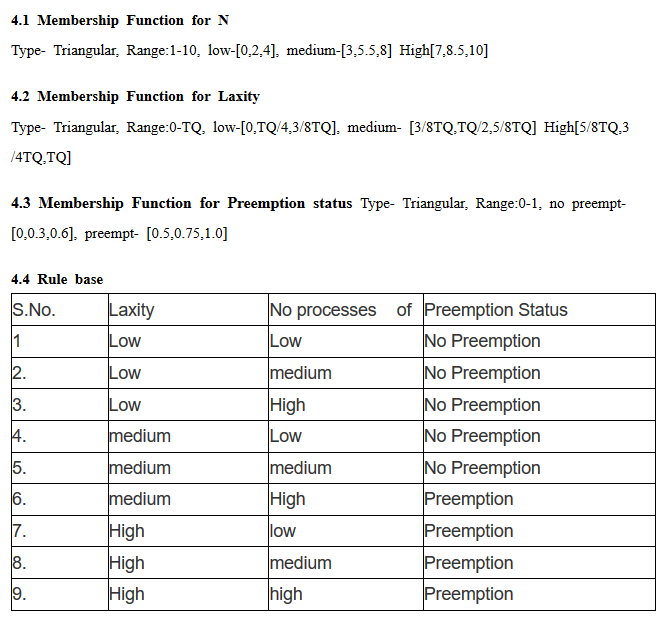
1. Laxity: Type-Triangular, Range:0-TQ, low-[0,TQ/4,3/8TQ], medium-[3/8TQ,TQ/2,5/8TQ] High[5/8TQ,3/4TQ,TQ]



1. Preemption status: Type-Triangular, Range:0-1, no preempt-[0,0.3,0.6], preempt-[0.5,0.75,1.0]



1. Rule base:



引入这样的逻辑推理系统后，RR调度算法流程变为了：

1. 选择就绪队列中的第一个进程p执行，并将其从就绪列表中删除。

2. 将时间量的值加载到间隔计时器中。

3. 在CPU上启动进程P的执行

4. 如果P启动I/O操作或完成其执行，请转到步骤1以调度另一个进程执行。

5. 当定时器中断发生时，不要抢占进程，而是启动另一个计数器来测量进程的松弛度。

i)。如果流程完成，则转到步骤1以调度另一个流程执行。

ii.) 将Laxity和N，即第4节中FIS输入的就绪进程数，取其输出作为抢占状态。

iii.) 如果抢占状态大于或等于0.5，则抢占进程并将其放在就绪队列的末尾，然后转到步骤1以调度另一个进程执行

else:

如果流程完成，转到步骤1以调度另一个流程执行。

参考文献：

【1】K.E. Hoganson, Multi-core Real-time Scheduling in Multilevel Feedback Queue with Starvation Mitigation (MLFQ-RT), In Proceedings of the ACMSE 2018 Conference, ACM, pp. 29, 2018.

【2】Bailey Granam and Hala ElAarag. 2019. Utilization of Fuzzy Logic in CPU Scheduling in Various Computing Environments. In Proceedings of the 2019 ACM Southeast Conference (ACM SE '19). ACM, New York, NY, USA, 222-225. DOI: https://doi.org/10.1145/3299815.3314463

【3】A. Silberschatz, P. B. Galvin, and G. Gagne, Operating System Concepts, 9th Edition. Hoboken, New Jersey: Wiley, 2018

【4】F. Howard, Fuzzy Logic, 2011, https://www.maths.tcd.ie/~fhoward/pdfs/fuzzy.pdf.