一、实验简述

多队列仿真

背景描述：一个具有三个输入队列和一个服务器的传输系统的公平调度问题。

三个队列数据包到达服从泊松过程（Poisson process）。

服务时间服从指数分布（exponentially distributed）。

实验目标：探索一个能够为每个队列提供同等服务能力的调度方案。

获得平均队列长度、平均等待时间、每个队列的队列长度分布和等待时间分布。

二、实验内容

一、实验模型

以经典的 M/M/1 模型为基础，扩展成三队列的传输模型。

队列模型遵循以下特征：

1. 到达人数是泊松过程(Poisson process)
2. 服务时间是指数分布(exponentially distributed)
3. 只有一台服务器(server)
4. 队列长度无限制
5. 可加入队列的人数为无限

二、实现过程与算法分析

仿真过程利用 C++实现，利用面向对象的思想将队列、服务器抽象成类，并通过服务器利用特定的调度算法对三个队列的传输状态进行控制。

调度算法方面采用了加权平均队列(Weighted Fair Queue, WFQ)算法：

WFQ 算法通过参考流体调度模型的通用处理器共享(Generalized Processor Sharing, GPS) 算法来控制实际系统中分组的发送顺序。调度过程中系统维护队列的“虚拟时间”，基于该虚拟时间为到达分组计算其虚拟开始时间(Visual Start Time, VST)和虚拟完成时间(Visual Finish Time, VFT)，并以 VFT 作为分组调度的优先顺序。

由于 WFQ 算法和 GPS 算法都基于服务器匀速处理数据的假设，因此我们将服务器的处理速度设成匀速，相应的让数据包的大小服从指数分布，这样和之前的服务时间服从指数分布相同。

在每个数据包生成时，虚拟时间被标记为这个数据包及其离开的虚拟时间。如果第i个队列产生了一个数据包，其虚拟时间的计算公式如下:

$$VirStart = max(now(), LastVirFinish_i)$$

$$VirFinish = VirStart + \frac{packet.size}{R_i}$$

$$R_i = \frac{w_i}{\sum_j w_j} \times R$$

其中，$VirStart$ $VirFinish$分别表示这个数据包的虚拟开始和结束时间，$now()$表示当前时间，$LastVirFinish_i$ 表示第i个队列前一个数据包的虚拟结束时间，$packet.size$ 表示该包的大小，R_i 表示第i个队列所被分配的虚拟服务速率，w_i表示第i个队列的服务权重，R表示服务器的总服务速率。

WFQ 算法在调度过程中，从所有队列中选择$LastVirFinish$最小的一个进行服务，由于在计算虚拟结束时间时，是根据的权重算出来的虚拟服务速率，类似 GPS 算法，在决策的过程中将总服务速率进行了分割，选取虚拟结束时间最早的数据队列，但实际的服务过程中速率并没有被分割。

GPS 算法的公平性保障在于，对服务速率进行了分割，并且不考虑数据包大小的不同，这样便相当于几个子服务器分别服务一个队列，显然是最公平的调度算法。然而实际过程数据包并不能以流模型描述，因此，考虑了包的大小并同样以虚拟的子服务速率计算虚拟完成时间的 WFQ 算法在公平性上有一定的保障。

WFQ 算法的弊端之一是如果有n个数据队列，每次在选取调度的数据队列时，要通过$O(n)$的时间复杂度来选取出$LastVirFinish$最小的服务器，这一点在n很大时严重影响了算法性能。一个改进的做法是用最小堆来维护服务器的$LastVirFinish$值，每次选取只需要$O(1)$的时间复杂度，但更改的过程需要$O(logn)$的时间复杂度来更新最小堆，总的时间复杂度为$O(logn)$。

当然，由于本次实验只有三个队列，因此算法的$O(logn)$与$O(n)$的复杂度差别微乎其微，因此我们并没有对其进行堆的优化或者采用其他的诸如 WF2Q+、SCFQ 等算法。

三、仿真结果

1）各个队列相同权重

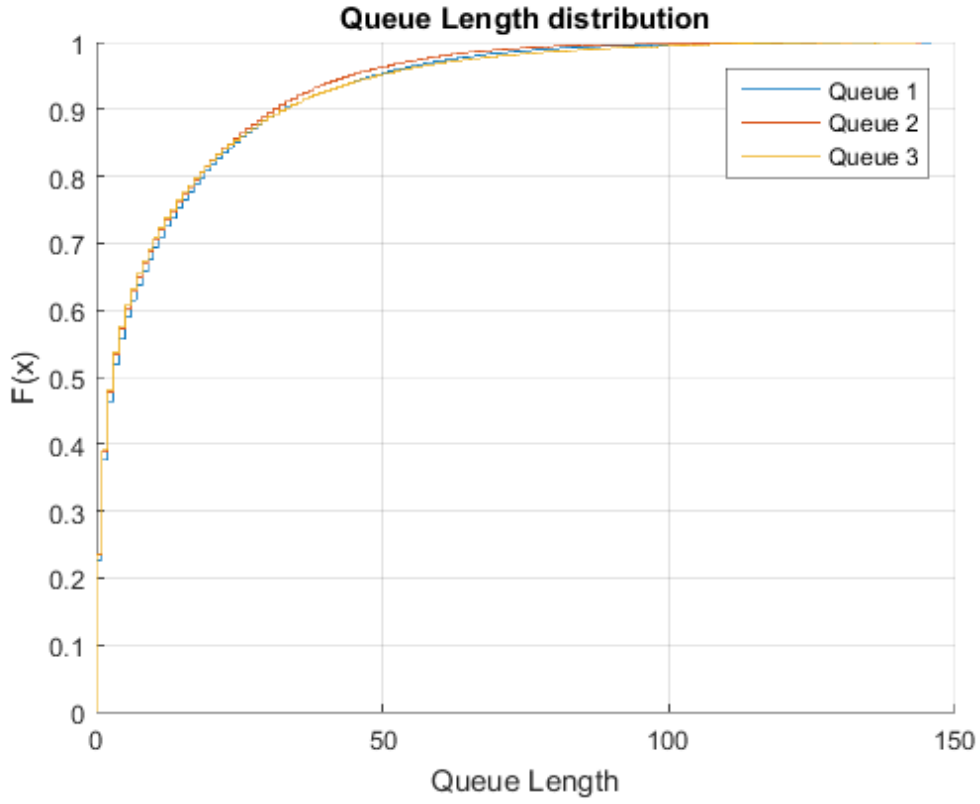
测试参数：

数据包到达过程频率（λ）：1

服务过程频率（λ）：3.1

在测试数据包数为三百万的情况下，三队列的队列及等待时间分布分别如下：

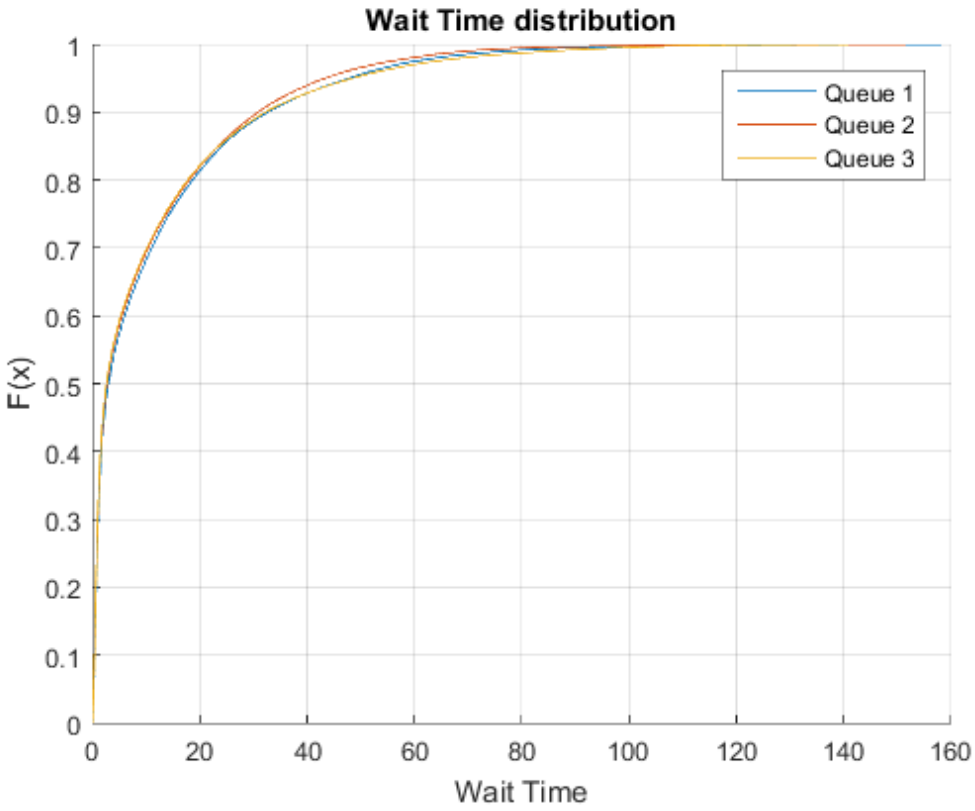
（1）三队列的队列长度分布



表格 1 队列长度统计参数

队列	最小长度	最大长度	平均长度	中位数	标准差
队列一	0	146	10.9744	3	17.2216
队列二	0	144	10.2708	3	15.9997
队列三	0	143	10.8683	3	17.9410

（2）三队列的等待时间分布



表格 2 等待时间统计参数

队列	最小等待时间	最大等待时间	平均等待时间	中位数	标准差
队列一	0	158.2149	10.9529	2.9769	16.8792
队列二	0	151.6047	10.2457	2.6886	15.5947
队列三	0	140.7962	10.8269	2.6182	17.5700

2）三个队列不同权重

测试参数：

数据包到达过程频率（λ）：1

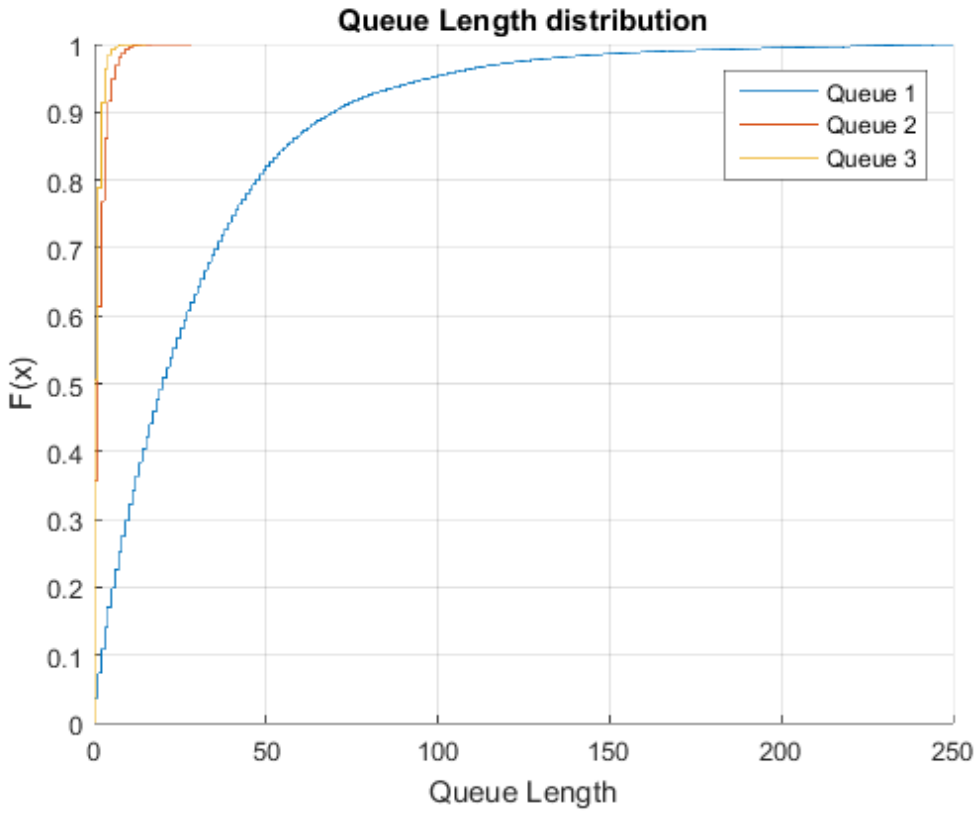
服务过程频率（λ）：3.1

队列一服务权重：1

队列二服务权重：2

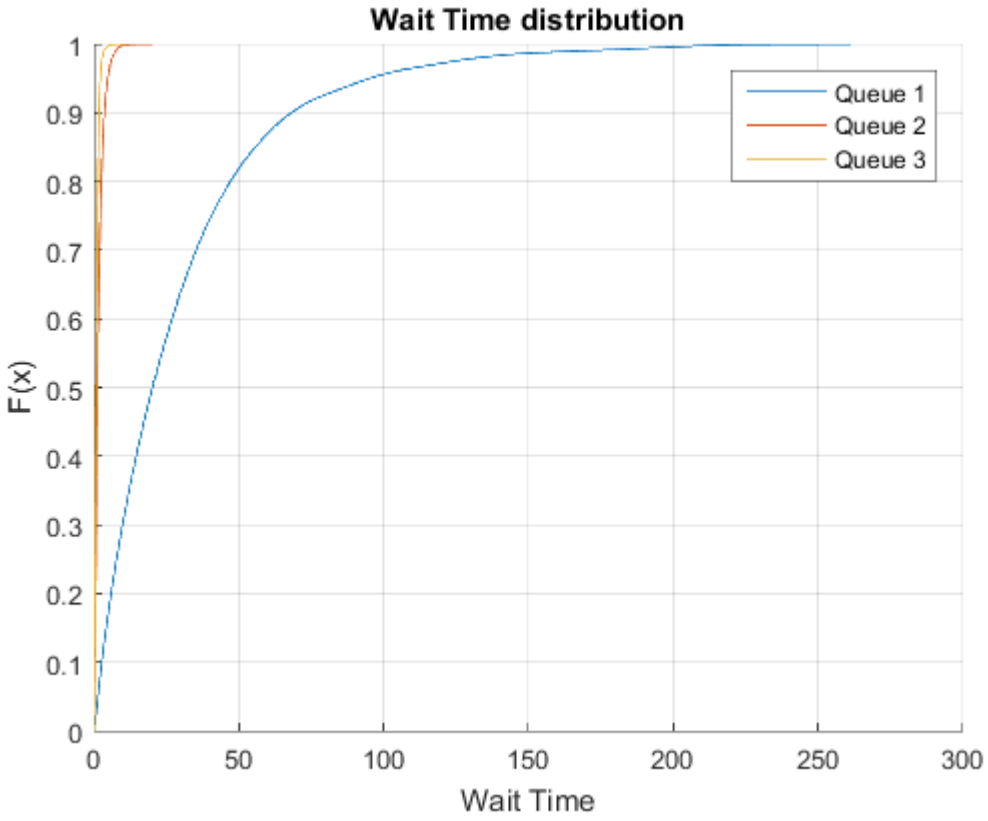
队列三服务权重：3

同样的数据量测试了三队列的队列分布及等待时间分布：



表格 3 加权队列长度统计参数

队列	最小长度	最大长度	平均长度	中位数	标准差
队列一	0	250	30.2853	20	33.6995
队列二	0	28	1.6164	1	1.9952
队列三	0	15	0.8561	0	1.1733



表格 4 加权等待时间统计参数

队列	最小等待时间	最大等待时间	平均等待时间	中位数	标准差
队列一	0	261.0357	30.1718	20.1949	32.9933
队列二	0	20.0131	1.6159	1.1392	1.5440
队列三	0	11.0315	0.8561	0.6838	0.7151

2. 结果评价

当各个队列权重相同时，观察三队列的队列长度分布和等待时间分布曲线，发现虽然队列长度的分布曲线由于取值较少且均为离散点而存在细微的锯齿状，但三条曲线相差不大，体现了 WFQ 算法优秀的公平性。

在对比实验中，改变了三个队列的服务权重，不出所料的队列长度分布和等待时间分布产生了明显的偏差，服务权重越大的队列侧面佐证了所用算法的公平性。

三、实验感想

在最一开始的 M/M/1 的 Debug 过程中，我们发现当数据包产生速率和服务速率相同时，队列长度并不能随测试仿真规模的增大而收敛到一个确定的值，因此我纠结了好久，看了半天代码也找不出来哪里出了问题。直到我找到了 M/M/1 的队列长度公式时，我发现包产生速率和服务速率相同时队列长度并不应该收敛到 0，而就应该随机的摆动，这个错误并不是代码中的错误，而是我想当然地得出了错误的结论，并没有认真的思考这个结论。在之后的测试过程中，M/M/1 的模拟得出的参数与公式算出的参数能够稳定到 0.01 的误差内，因此我才确定了代码能够在一定情况下正常运行。

这的确给了我挺大的启示，debug 过程中在找代码的错误之前要先分析错误产生的现象，并分析这个现象到底是否是正常的，确定了现象是错误的之后再去找代码中的 bug，不能盲目的认为自己的代码某处出了问题而是应该从现象入手分析导致错误结果的原因，从而能高效的找出出错位置。

数据处理过程，一开始每次都要将文件的数据导到 Matlab 中并手动输入函数进行画图，费时费力，之后写了一个脚本自动的加载数据进行画图分析数据给实验的数据处理工作增添了很大的帮助。