COMP9334项目，第1学期，2024:计算集群

截止日期:2024年4月19日星期五下午5:00

版本1.01

项目的更新，包括任何更正和澄清，将发布在课程网站上。请确保您定期查看课程网站以获取更新。

更改日志

· 版本1.01(2024年3月27日)。5.1.1节中两个概率密度函数的分母有一个错误。对于$g_0(t)$，应该t取+1缺失的η_0+1次幂。类似的错误出现在$g_1(t)$中，它应该是t的η1+1次方。

· 1.00版。2024年3月19日发布。

1 介绍及学习目标

在第4A周的讲座中，您已经了解到，到达间隔时间或服务时间的高度可变性会导致高响应时间。来自真实计算机集群的测量发现，这些集群中的服务时间具有非常高的可变性[1]。参考文献[1]也有一些建议来处理这个问题。一种建议是根据作业的服务时间要求将作业分开，让一组服务器处理服务时间短的作业，另一组服务器处理服务时间长的作业。这种安排与超市为购买不超过一定数量的商品的顾客提供快速收银台和其他对商品数量没有限制的收银台是一样的。你们已经在第4A周的复习题1中看到了这个理论的实际应用。我们也强烈推荐大家去阅读论文[1]。

在这个项目中，你将使用仿真来研究如何减少一个服务器场的响应时间，该服务器场使用不同的服务器来处理具有不同服务时间要求的作业。

在这个项目中，你将学习到:

1. 用离散事件模拟来模拟计算机系统
2. 用仿真来解决设计问题
3. 使用统计上合理的方法来分析模拟输出

我们在讲座中多次提到，仿真并不是简单地编写仿真程序。虽然让你的模拟代码正确是很重要的，但使用统计上合理的方法来分析模拟输出也很重要。在那里，这个项目大约一半的分数分配给了模拟程序，另一半分配给了统计分析;见7.2节。

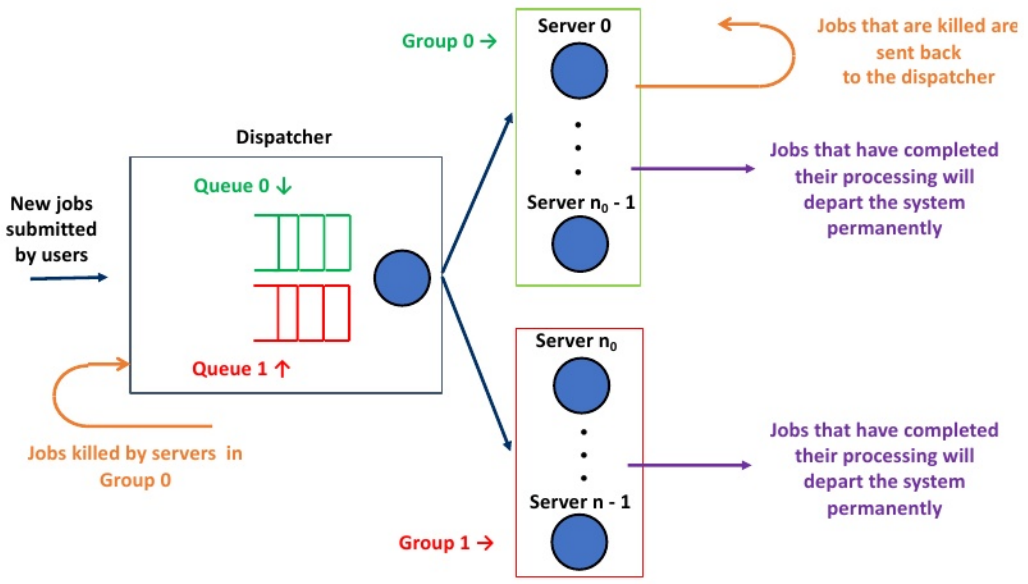


图1:这个项目的多服务器系统。

2 提供的支持和计算资源

如果你在做这个项目时遇到问题，可以在课程论坛上发表你的问题。我们强烈建议你这样做，因为提出问题并尝试回答问题是一种很好的学习方式。不要担心你的问题可能看起来很傻，其他学生很可能也有同样的问题!请注意，如果你的论坛帖子显示了你的部分解决方案或代码，你必须将该论坛帖子标记为私有。

另一种获得帮助的方法是参加咨询(参见课程网站的时间表部分，了解日期和时间)。

如果你需要计算资源来运行你的模拟程序，你可以在学院提供的VLAB远程计算设施上完成。关于VLAB的信息可以在这里找到:https: //taggi.cse.unsw.edu.au/ VLAB /

3 带有作业隔离的多服务器系统配置

您将在本项目中使用的多服务器系统的配置如图1所示。该系统由一个调度器和n台服务器组成，其中n≥2台。n个服务器被划分为2个不相交的组，称为组0和组1，每组中至少有一个服务器。组0和组1中的服务器个数分别为n_0和n_1，其中n_0、n_1≥1和n_0+n_1= n。

组0中的服务器用于处理短作业，这些作业的处理时间不超过T_{limit}的时间限制。组1中的服务器对服务时间没有任何限制。

调度程序有两个队列:队列0和队列1。队列i(其中i = 0,1)中的作业被分配给组i中的服务器，两个队列都有无限的排队空间。

当用户向这个多服务器系统提交作业时，用户需要指明该作业是打算发给组0中的服务器还是组1中的服务器。下面的一般处理步骤对于所有传入的作业都是通用的:

· 如果一个作业是为组i(其中i = 0,1)中的服务器准备的，该作业到达调度程序时，如果组i中有可用的服务器，则该作业将被发送到组i中的服务器，否则该作业将加入队列i。
· 当作业离开Group i中的服务器时，服务器将检查队列i的头部是否有作业，如果有，则该作业将被允许进入可用的服务器进行处理。

回想一下，Group 0中的服务器有一个服务时间限制。其目的是让用户对其提交的作业的服务时间需求进行估计。如果用户认为他们的作业应该能够在T_{limit}内完成，那么他们将其提交到第0组;否则，他们应该将其发送到组1。

不幸的是，用户估计的服务时间并不总是正确的。有可能用户将无法在时限内完成的作业发送给组0。现在我们将解释多服务器系统将如何处理这样的作业。由于用户已经表示该作业将被分配给组0，因此该作业将按照前面解释的一般处理步骤进行处理。这意味着该作业将由组0中的服务器进行处理。在此作业被处理了T_{limit}一段时间后，服务器表示服务时间限制已满，并将终止该作业。服务器将把该作业发送给调度程序，并告诉它这是一个已杀死的作业。调度器将检查组1中的服务器是否可用。如果是，作业将被发送到可用的服务器;否则，它将加入队列1等待服务器可用。当第1组中的一个服务器可以处理这个作业时，它将从头处理这个作业，也就是说，第0组服务器之前的所有处理都将丢失。

如果一个作业在Group 0服务器上完成了它的处理，这意味着它的服务时间小于等于T_{limit}，那么这个作业就永久地离开了多服务器系统。类似地，在第1组服务器上完成处理的作业将永久离开系统。

我们在图1中的多服务器系统上做了以下假设。首先，调度程序对作业进行分类并将作业发送到可用的服务器所需的时间可以忽略不计。其次，服务器将已杀死的作业发送给调度程序所需的时间可以忽略不计。第三，服务器通知调度程序其可用性所需的时间可以忽略不计。作为这些假设的结果，它意味着:(1)如果到达调度程序的作业要立即发送到可用的服务器，那么它到达调度程序的时间与到达所选服务器的时间是相同的;(2)作业从调度器出发的时间与其到达所选服务器的时间相同;(3)已终止作业从服务器出发的时间与其到达调度程序的时间相同。最终，这些假设意味着系统的响应时间仅取决于队列和服务器。

现在，我们已经完成了图1中对系统操作的描述。我们将在第4节中提供一些数值示例来进一步解释其操作。

您将从第4节中的数值示例中看到，可以使用Group 0服务器n_0的数量来影响平均响应时间。因此，您在本项目中要考虑的一个设计问题是确定n_0的值以最小化平均响应时间。

备注1上述描述中的一些元素是现实的，但有些则不是。通常情况下，用户在向计算集群提交作业时需要指定一个walltime作为服务时间限制。如果服务器已经在作业上花费了指定的超时时间，那么服务器

将终止该作业。这些都是现实的。

被扼杀的工作的再循环通常不会完成。如果一个新作业已经被杀死，用户通常需要重新提交它。如果一个被杀死的作业被重新循环，那么它可能会被赋予一个较低的优先级，而不是像这里这样加入主队列。

一些编程技术(例如，检查点)允许一个被杀死的作业或崩溃的作业从最后保存的状态恢复，而不是从开始恢复。然而，这可能需要相当大的内存空间。

为了使这个项目更可行，我们简化了许多设置。例如，我们不使用较低的优先级来重新循环被杀的工作。

4 例子

现在我们将提供三个例子来说明您将在本项目中模拟的系统的操作。在所有这些例子中，我们都假设系统最初是空的。

4.1 Example 0: $n = 3$, $n_0 = 1$, $n_1 = 2$ and $T_{\text{limit}} = 3$

在这个例子中，我们假设场中有n = 3台服务器，其中1(=n_0))服务器在组0中，2(=n_1))服务器在组1中。第0组处理的时间限制是T_{limit}= 3。

表1显示了我们将在本例中使用的8个作业的属性。每个作业都有一个索引(从0到7)，对于每个作业，表1显示了它的到达时间、服务时间和用户指定的服务器组。例如，作业1在时间10到达，需要4个单位的服务时间，并且用户已经指示该作业需要转到第0组的服务器。由于此作业的服务时间要求超过了3的时间限制T_{limit}，因此此作业将在服务3个时间单位后被杀死，并在此之后被发送给调度器。

请注意，如果用户发送给Group 0服务器的作业的服务时间小于或等于所施加的服务时间限制T_{limit}，则该作业将被完成。因此，表1中的作业6将在Group 0服务器中完成，并且该作业不会被杀死。

Job index	Arrival time	Service time required	Server group indicated
0	2	5	1
1	10	4	0
2	11	9	0
3	12	2	0
4	14	8	1
5	15	5	0
6	19	3	0
7	20	6	1

表1:以0为例的作业。

注释2我们注意到，作业索引对于执行离散事件模拟是不必要的。我们包含了作业索引，以便在下面的描述中更容易引用作业。

图1中系统中的事件是
· 新作业到达调度器;而且,

· 工作从服务器上离开。

我们注意到，对于第1组服务器，一个离开的作业已经完成了它的服务。然而，对于第0组服务器，离开的作业可以是已杀死的作业，也可以是已完成的作业。请注意，我们没有将重新循环的已杀死作业到达调度器作为事件包括在内。这是因为重新循环的作业到达调度器的同时，该作业也离开了组0服务器。因此，模拟将一起处理这些事件:被杀死的作业的离开和调度器对它的处理。

我们将使用“纸上仿真”来说明图1中系统的仿真。你需要跟踪的数量包括:

· 下一个到达时间是下一个新工作(即不是已终止的工作)到达的时间

· 对于每个服务器，我们跟踪它的服务器状态，它可以是繁忙或空闲。

· 我们还会跟踪服务器上正在处理的作业的以下信息:

-下一次离开时间是作业将离开服务器的时间。如果服务器处于空闲状态，则下一次离开时间设置为∞。注意，每个服务器都有一个下一个出发时间。

-该作业到达系统的时间。这是计算作业永久离开系统时的响应时间所需要的。

· 队列0和队列1的内容。队列中的每个作业由一个2元组(到达时间，服务时间)标识。

还有其他额外的数量，你需要跟踪，他们将在后面提到。

“纸上模拟”如表2所示。最后一栏的注释解释了你需要为每个事件做哪些更新。回想一下，这个模拟中的两个事件类型是新作业到达调度器和离开服务器，我们将在表2的“事件类型”列(即第二列)中简单地将这两个事件称为arrival和departure。

Master clock	Event type	Next arrival time	Server 0 Group 0	Server 1 Group 1	Server 2 Group 1	Queue 0	Queue 1	Notes
0	–	2	Idle, ∞	Idle, ∞	Idle, ∞	–	–	We assume the servers are idle and queues are empty at the start of the simulation. The next departure times for all servers are ∞. The "–" indicates that the queues are empty.
2	Arrival	10	Idle, ∞	Busy, (2,7)	Idle, ∞	–	–	This event is the arrival of Job 0 for a Group 1 server. Since both Group 1 servers are idle before this arrival, the job can be sent to any one of the idle servers. We have chosen to send this job to Server 1. The job requires a service time of 5, so its completion time is 7. Note that the record of the job in the server is a 2-tuple consisting of (arrival time, scheduled departure time). Lastly, we need to update the arrival time of the next job, which is 10.
7	Departure	10	Idle, ∞	Idle, ∞	Idle, ∞	–	–	This event is the departure of a job from Server 1. Since Queue 1 is empty, Server 1 becomes idle.
10	Arrival	11	Busy (10,13, 4)	Idle, ∞	Idle, ∞	–	–	This event is the arrival of Job 1 for a Group 0 server. Since Server 0 is idle, the job can be sent to the idle server. This job requires a service time of 4 which exceeds the service time limit of 3 for Group 0 servers, so the simulation needs to schedule this job to depart Server 0 at time 13 because this is the time that this job will be killed by the server. We use the 3-tuple consisting of (arrival time, scheduled departure time, service time), which for this job is (10, 13, 4), to indicate that this job arrives at time 10, is scheduled to depart at time 13 and its service time requirement is 4 time units. We need to include the service time of the job because we will need it later when the job is re-circulated to a Group 1 server. Note that if you see a 3-tuple job in a Group 0 server, it means that the job will be killed and re-circulated to a Group 1 server. Lastly, we need to update the arrival time of the next job, which is 11.

11	Arrival	12	Busy (10,13, 4)	Idle, ∞	Idle, ∞	(11,9)	–	This event is the arrival of Job 2 for a Group 0 server. Since Server 0 is busy, this job will join Queue 0. The queue stores the 2-tuple (arrival time, service time) which is (11,9) for this job. We also need to update the arrival time of the next job, which is 12.
12	Arrival	14	Busy (10,13, 4)	Idle, ∞	Idle, ∞	(11,9), (12,2)	–	This event is the arrival of Job 3 for a Group 0 server. Since Server 0 is busy, this job will join Queue 0 with the job information (12,2). We also need to update the arrival time of the next job, which is 14.
13	Departure	14	Busy (11,16, 9)	Busy (10,17)	Idle, ∞	(12,2)	–	This event is the departure of a killed job from Server 0. This job will be re-circulated to the dispatcher. Since both Group 1 servers are idle, this job can go to any one of them. We have chosen to send it to Server 1. Since this job requires 4 time units of service, it is scheduled to depart Server 1 at time 17. The 2-tuple (10,17) indicates that this job arrives at 10 and will depart at time 17. Since this is a departure from a Group 0 server, we will also need to check Queue 0, which has 2 jobs. So the job at the head of the queue will advance to Server 0 which is becoming available. This job requires 9 units of service time which exceeds the service time limit. So, the job will be killed at time 13 + 3 = 16 time units.
14	Arrival	15	Busy (11,16, 9)	Busy (10,17)	Busy (14,22)	(12,2)	–	This event is the arrival of Job 4 for a Group 1 server. Since there is a Group 1 server available, this job goes to Server 2 directly. This job requires 8 units of service, so the job is scheduled to depart at time 22. We also need to update the arrival time of the next job, which is 15.
15	Arrival	19	Busy (11,16, 9)	Busy (10,17)	Busy (14,22)	(12,2) (15,5)	–	This event is the arrival of Job 5 for a Group 0 server. Since all Group 0 servers are busy, this job joins Queue 0. We also need to update the arrival time of the next job, which is 19.

16	Departure	19	Busy (12,18)	Busy (10,17)	Busy (14,22)	(15,5)	(11, 9)	This event is the departure of a killed job from Server 0. This job will be re-circulated to the dispatcher. Since both Group 1 servers are busy, this job will join Queue 1. The job at the head of Queue 0 will advance to Server 0. This job requires only 2 units of service which is within the limit. We use a 2-tuple to remember this job because the job is within the time limit so it will not be killed.
17	Departure	19	Busy (12,18)	Busy (11, 26)	Busy (14,22)	(15,5)	-	This event is the departure of a finished job at Server 1. Since there is a job in Queue 1, the job will move into Server 1.
18	Departure	19	Busy (15,21,5)	Busy (11, 26)	Busy (14,22)	-	-	This event is the departure of a finished job at Server 0. This job will depart from the system permanently. We can tell that because it is a 2-tuple in the server rather than a 3-tuple. Since there is a job in Queue 0, the job will move into Server 0.
19	Arrival	20	Busy (15,21,5)	Busy (11, 26)	Busy (14,22)	(19,3)	-	This event is the arrival of Job 6 for a Group 0 server. Since all Group 0 servers are busy, this job joins Queue 0. We also need to update the arrival time of the next job, which is 20.
20	Arrival	∞	Busy (15,21,5)	Busy (11, 26)	Busy (14,22)	(19,3)	(20, 6)	This event is the arrival of Job 7 for a Group 1 server. Since all Group 1 servers are busy, this job joins Queue 1. Since there are no more jobs arriving, we update the next arrival time to ∞
21	Departure	∞	Busy (19,24)	Busy (11, 26)	Busy (14,22)	-	(20,6), (15,5)	This event is the departure of a killed job from Server 0. This job will be re-circulated to the dispatcher. Since both Group 1 servers are busy, this job will join Queue 1. The job at the head of Queue 0 will advance to Server 0. This job requires only 3 units of service which is within the limit. We only need a 2-tuple to remember that this job arrives at time 19 and will depart at time 24.
22	Departure	∞	Busy (19,24)	Busy (11, 26)	Busy (20, 28)	-	(15,5)	This event is the departure of a finished job at Server 2. Since there is a job in Queue 1, the job will move into Server 2.
24	Departure	∞	Idle, ∞	Busy (11, 26)	Busy (20, 28)	-	(15,5)	This event is the departure of a finished job at Server 0. Since Queue 0 is empty, Server 0 is now idle.
26	Departure	∞	Idle, ∞	Busy (15, 31)	Busy (20, 28)	-	-	This event is the departure of a finished job at Server 1. The job at the head of Queue 1 advances to Server 1. The queue is now empty.

28	Departure	∞	Idle, ∞	Busy (15, 31)	Idle, ∞	-	-	This event is the departure of a finished job at Server 2. Server 2 is now idle as Queue 1 is empty.
31	Departure	∞	Idle, ∞	Idle, ∞	Idle, ∞	-	-	This event is the departure of a finished job at Server 1. Server 1 is now idle as Queue 1 is empty.

Table 2: "On paper simulation" illustrating the event updates of the system.

上面的描述没有解释如果到达事件和离开事件同时发生会发生什么。我们将不做具体说明。如果我们要求您在跟踪驱动模式下进行模拟，我们将确保不会发生此类情况。如果到达间时间和服务时间是随机生成的，那么这种情况发生的几率实际上为零，所以你不必担心。

表3总结了本例中工作的到达、离开、工作分类和响应时间。在表中，我们将工作分为3类:

- 在限定时间内完成(即未杀死)的第0组作业。从现在开始，我们将这些作业称为已完成的第0组作业。这些作业被标记为0。
- 再循环的第0组作业。它们被标记为0。
- 用户为组1指定的作业。它们被标记为1。

在表3中，我们包括了已完成的第0组作业和第1组作业的响应时间。完成的第0组作业的平均响应时间为$_{112}$ = 5.5，第1组作业的平均响应时间为$_{213}$ = 7。

稍后，您将处理一个设计问题，以减少已完成的第0组作业和第1组作业的平均响应时间的加权和。在这里，我们故意忽略了重新循环的作业，因为我们不会试图减少它们的响应时间。原因是我们不想鼓励用户对其工作的服务时间要求给出较差的估计。

Job	Arrival time	Departure time	Job classification	Response time	
				Group 0 within limit	Group 1
0	2	7	1		5
1	10	17	r0		
2	11	26	r0		
3	12	18	0	6	
4	14	22	1		8
5	15	31	r0		
6	19	24	0	5	
7	20	28	1		8

表3:示例0中工作的到达和离开时间。

4.2 例1:n = 4, n0 = 2, n1 = 2, Tlimit = 3.5

对于这个例子，我们假设系统有n = 4台服务器。0组和1组各有2台服务器，即n_0= n_1= 2。组0服务器的服务时限为T_{limit}= 3.5。

表4显示了将到达该系统的作业的属性。表5总结了模拟的结果。完成的第0组作业的平均响应时间为$\overline{23.94}$ = 5.975
第一组作业的平均响应时间为$\overline{36.85}$ = 7.36。

Job index	Arrival time	Service time required	Server group indicated
0	2.1	5.2	1
1	3.4	4.1	1
2	4.1	3.1	0
3	4.4	3.9	0
4	4.5	3.4	0
5	4.7	4.4	1
6	5.5	4.7	1
7	5.9	4.1	0
8	6.0	2.5	0
9	6.5	8.6	1
10	7.6	4.1	0
11	8.1	2.6	0

表4:示例1中的作业。

Job	Arrival time	Departure time	Job classification	Response time	
				Group 0 within limit	Group 1
0	2.1	7.3	1		5.2
1	3.4	7.5	1		4.1
2	4.1	7.2	0	3.1	
3	4.4	16.1	r0		
4	4.5	10.6	0	6.1	
5	4.7	11.7	1		7.0
6	5.5	12.2	1		6.7
7	5.9	20.2	r0		
8	6.0	13.1	0	7.1	
9	6.5	20.3	1		13.8
10	7.6	24.3	r0		
11	8.1	15.7	0	7.6	

表5:例1中工作的到达和离开时间。

4.3 Example 2: $n = 4$, $n_0 = 1$, $n_1 = 3$ and $T_{\text{limit}} = 3.5$

本例与例1相同，除了$n_0 = 1$。表6总结了模拟的结果。完成的第0组作业的平均响应时间为$\overline{44.94} = 11.225$，平均
第一组作业的平均响应时间为$\overline{29.85} = 5.96$。这并不奇怪，平均响应时间
已完成的第0组作业的百分比上升，而第1组作业的百分比下降。这是因为在本例中，组0中的服务器较少。

Job	Arrival time	Departure time	Job classification	Response time	
				Group 0 within limit	Group 1
0	2.1	7.3	1		5.2
1	3.4	7.5	1		4.1
2	4.1	7.2	0	3.1	
3	4.4	14.6	r0		
4	4.5	14.1	0	9.6	
5	4.7	9.1	1		4.4
6	5.5	12.0	1		6.5
7	5.9	21.7	r0		
8	6.0	20.1	0	14.1	
9	6.5	16.1	1		9.6
10	7.6	27.7	r0		
11	8.1	26.2	0	18.1	

表6:例2中作业的到达和离开时间。

5 项目描述

本项目主要由两部分组成。第一部分是为图1中的系统开发仿真程序。该系统已经在第3节中进行了描述，并在第4节中进行了说明。在第二部分中，你将使用你开发的仿真程序来解决一个设计问题。

5.1 模拟程序

你必须用以下语言中的一种(或组合)来编写模拟程序:Python 3(注意:仅限版本3)、C、c++或Java。所有这些语言在CSE系统中都可以使用。

我们将在CSE系统上测试您的程序，因此您提交的程序必须能够在CSE计算机上运行。请注意，由于版本和/或操作系统的差异，在您自己的计算机上运行的代码可能无法在CSE系统上运行。确保您的代码在CSE系统上工作是您的责任。

请注意，我们的描述使用了以下变量名:

1. 字符串类型的可变模式。这个变量是用来控制你的程序是否会使用随机生成的到达时间和服务时间来运行模拟;还是以跟踪驱动模式运行。参数模式可以取的值要么是随机的，要么是跟踪的。
2. 一个time_end变量，如果主时钟超过这个值，它将停止模拟。这个变量只有在mode是随机的时候才有意义。这个变量是一个正浮点数。

注意，你的模拟程序必须是一个允许使用不同参数值的通用程序。当我们测试你的程序时，我们会改变参数值。您可以假设我们将只使用有效的输入进行测试。

对于模拟，你总是可以假设系统最初是空的。

提示:不要为随机模式和跟踪模式分别编写两个程序，因为它们有很多共同之处。在适当的地方使用一些if-else语句，你就需要在一个程序中同时使用这两种模式。

5.1.1 随机模式

当你的模拟在随机模式下工作时，它将以以下方式生成间隔到达时间和作业的工作量。

1. 我们使用{a_1，a_2，…，a_k，…,……}表示作业到达调度器的间隔到达时间。这些间隔到达时间具有以下属性:
   (a)每个ak是两个随机数a1k与a2k的乘积，亦即ak = a1ka2k∀k = 1,2，…(b)序列a_{1k}呈指数分布，平均到达率为λrequests/s。(c)序列a_{2k}均匀分布在区间[a_{2l}，a_{2u}]。

   注:生成间隔到达时间的最简单方法是将给定速率的指数分布随机数与给定范围内的均匀分布随机数相乘。在这种情况下，使用逆变换方法会更加困难，尽管它是可行的。

2. 作业的工作负载由两个属性表征:作业要发送到的服务器组(即组0或组1)和作业的服务时间。

(a)确定将作业发送到哪个服务器组的第一步。这个决定是由一个参数$p_0 \in (0,1)$做出的:

- Prob[用户为Group 0服务器指定的作业]= p_0
- Prob[用户为第1组服务器指定的作业]= $1-p_0$

例如，如果p_0为0.8，则为Group 0服务器指示作业的概率为0.8，而为Group 1服务器指示作业的概率为0.2。每个作业的服务器组都是独立生成的。

(b)一旦为一个作业生成了服务器组，下一步就是生成它的

服务时间。需要使用的服务时间分布取决于服务器组。

i.如果指定某个作业转到组0服务器，则该作业的服务时间为概率密度函数PDF$g_0(t)$:

$$g_0(t) = \begin{cases} 0 & \text{for } 0 \le t \le \alpha_0 \\ \dfrac{\eta_0}{\gamma_0\, t^{\eta_0+1}} & \text{for } \alpha_0 < t < \beta_0 \\ 0 & \text{for } t \ge \beta_0 \end{cases} \quad (1)$$

在哪里

$$\gamma_0 = \alpha_0^{-\eta_0} - \beta_0^{-\eta_0}$$

注意，这个概率密度函数有3个参数:α_0，β_0和η_0。可以假设$\beta 0 > \alpha 0 > 0$，$\eta 0 > 1$。

2。如果一个作业被指示转到第1组服务器，则其服务时间具有PDF:

$$g_1(t) = \begin{cases} 0 & \text{for } 0 \le t \le \alpha_1 \\ \dfrac{\eta_1}{\gamma_1\, t^{\eta_1+1}} & \text{for } \alpha_1 < t \end{cases} \quad (2)$$

在哪里

$$\gamma_1 = \alpha_1^{-\eta_1}$$

注意，这个概率密度函数有2个参数:α_1和η_1。可以假设$\alpha_1 > 0$，$\eta_1 > 1$。

5.1.2 示踪模式

当您的模拟在跟踪模式下工作时，它将从两个单独的ASCII文件中读取间隔到达时间列表、服务时间列表和服务器组列表。我们将在6.1.3节和6.1.4节中解释这些文件的格式。

跟踪模式的一个重要要求是，你的程序需要进行模拟，直到所有的作业都离开系统。您可以参考表2来进行说明。

5.2 确定使加权平均响应时间最小化的n_0的值

在编写了模拟程序之后，下一步是使用模拟程序来确定使加权平均响应时间最小化的Group 0服务器n_0的数量。

对于这个设计问题，你将假设以下参数值:

· 服务器总数:n = 10
· 第0组服务器的服务时间限制T_{limit}为3.3。
· 对于间隔到达时间:λ= 3.1，$a_{2\ell}$= 0.85，a_{2u}= 1.21
· 为第0组服务器指示作业的概率p_0为0.74。
· 为组0指示的作业的服务时间:α_0= 0.5，β_0= 5.7，η_0= 1.9。
· 为第1组指示的作业的服务时间:α_1= 2.7，η_1= 2.5。设计问题的目的是最小化加权响应时间:

$$w_0T_0 + w_1T_1 \tag{3}$$

其中T_0是完成的第0组作业的平均响应时间，T_1是第1组作业的平均响应时间。对于这个设计问题，权重w_0和w_1的值是固定的，它们分别由0.83和0.059给出。举个例子，如果T_0= 1.86,T_1= 56.7，则加权平均响应时间为0.83 ×1.86 + 0.059 ×56.7。备注3解释了选择这些权重的基本原理。

设计问题的目的是找到n_0的值，以最小化这个加权响应时间。注意，我们假设每组中至少有一个服务器，因此1≤n_0≤n−1。

在解决这个设计问题时，您需要确保使用统计上合理的方法来比较系统。你需要考虑模拟控制，比如模拟的长度、重复的次数、瞬态移除等等。你需要在你的报告中证明你如何确定n_0的值。

备注3对于上述参数，未重循环的作业中，时限内第0组作业占73.65%，第1组作业占26.35%。时限内的0组作业的平均服务时间为0.887,1组作业的平均服务时间为4.5。权重w_0和w1分别从$_{0.7365\ 0.887}$和$_{0.26354.5}$计算。所以权重考虑到了
一类工作出现的频率。我们还使用逆服务时间作为权重，这样我们就不会给第一类工作太多的优势，因为它们需要大量的服务时间。

6 测试您的模拟程序

为了让我们测试你的模拟程序的正确性，我们将使用一些测试用例来运行你的程序。本节的目的是描述预期的输入/输出文件格式以及如何执行测试。

每个测试由4个配置文件指定。我们将从0开始索引测试。如果使用12个测试，则测试的索引为0,1,2，....11。配置文件的名称为:
· 对于Test 0，配置文件为mode_0.txt、para_0.txt、interarrival_0.txt和service_0.txt。这些文件的命名类似于索引1,2,3，..9。
· 对于Test 10，配置文件为mode_10.txt、para_10.txt、interarrival_10.txt和service_10.txt。如果测试索引是2位数字，则这些文件的命名类似。

我们将使用通用名称模式*.txt, para *.txt等来引用这些文件。- 我们将在6.1节中描述配置文件的格式

每个测试应该产生2个输出文件，其格式将在6.2节中描述。我们将在6.3节和6.5节中解释如何进行测试。

6.1 配置文件格式

注意，Test 0与4.1节中讨论的示例0相同。我们将使用该测试来说明文件格式。

6.1.1 模式* . txt

这个文件是用来指示模拟应该在随机模式还是跟踪模式下运行。该文件包含一个字符串，可以是随机模式，也可以是跟踪模式。

6.1.2 对位* . txt

如果仿真模式为trace，则此文件有三行。第一行的值为n(=服务器总数)，第二行的值为n_0(= Group 0服务器数量)，第三行的值为T_{limit}。如果测试为4.1节中的例0，则该文件的内容为:

3

1

3

这些值在样例文件para_0.txt中。

如果模拟模式是随机的，那么该文件有四行。前三行含义与上文相同。最后一行包含time_end的值，它是模拟的结束时间。示例文件para_4.txt的内容如下所示，其中最后一行表示模拟应该运行到200。

5

2

3.1

200

你可以假设我们只会给你有效的值。你可以期望n是一个大于2的正整数，$n_0 \geqslant 1$并且$T_{limit} > 0$。对于time_end，它是一个严格的正整数或浮点数。

6.1.3 interarrival * . txt

interarrival *.txt文件的内容取决于测试的模式。如果mode为trace，则文件interarrival *.txt包含作业的间隔到达时间，其中一个间隔到达时间占用一行。您可以假设到达间隔时间列表始终为正。例如4.1节中的0，到达时间为[2,10,11,12,14,15,19,20]，这意味着到达间隔时间为[2,8,1,1,2,1,4,1]。对于本例，到达间隔时间将由一个文件指定(参见示例文件interarrival 0.txt)，其内容为:

2.0000

8.0000

1.0000

1.0000

2.0000

1.0000

4.0000

1.0000

如果模式为随机，则文件interarrival *.txt在一行中包含三个数字。这三个数字分别对应参数入、$a_{2\ell}$和a_{2u}。举个例子，interarrival 4.txt的内容为:

0.9 0.91 - 1.27

在本例中，λ、$a_{2\ell}$和a_{2u}的值分别为0.9、0.91和1.27。可以假设这些参数值都是正的。

6.1.4 服务* . txt

对于跟踪模式，文件service *.txt包含每个作业的服务时间和该作业的目标服务器组。举例来说，第4.1节中例0的服务时间和服务器组将由一个文件指定(参见示例文件服务0.txt)，其内容为:

5.0000 - 1
4.0000 0
9.0000 0
2.0000 0
8.0000 - 1
5.0000 0
3.0000 0
6.0000 - 1

注意，每行有2个条目，它们对应于服务时间(第一个条目)和服务器组(第二个条目)。例如，第一个作业的服务时间为5，表示为第1组服务器。您会发现service 0.txt的内容与表1中的信息是一一对应的。可以假设第一个条目是正浮点数，每行的第二个条目要么是0，要么是1。

对于随机模式，文件服务*.txt包含三行。例如，service 4.txt的内容为:

0.7
1.2 3.6 2.1
2.8 - 4.1

第一行的数字是p_0。第二行的三个数字分别是α_0、β_0和η_0。最后，第三行的两个数字分别是α_1和η_1。可以假设所有这些值都是有效的。

您可以假设我们为跟踪模式提供的数据在以下方面是一致的:到达间次数和服务时间的行数相等。

6.2 输出文件格式

为了测试你的模拟程序，每次测试我们需要两个输出文件。一个文件包含两个平均响应时间。另一个文件包含到达时间、离开时间和工作分类信息，类似于表3中的第2-4列。

对于随机模式，应该使用那些在time_end之前已经永久离开系统的作业来计算平均响应时间。换句话说，对于那些在time_end时仍在队列中或正在服务器中处理的作业，在计算平均响应时间时不包括这些作业。

请注意，在将结果写入输出文件之前，您不必考虑对平均响应进行瞬态删除。但是，在进行设计时应该考虑瞬态移除。

应该将两个平均响应时间写入文件名为mrt_*.txt的文件中。例如第4.1节中的0，该文件的预期内容是:

5.5000 7.0000
其中两个数字分别对应于已完成的第0类和第1类作业的平均响应时间。

另一个文件dep_*.txt包含作业的出发类型和分类。对于第4.2节中的例1，该文件的预期内容是:

4.1000 7.2000 0
2.1000 7.3000 1
3.4000 7.5000 1
4. 5000 10. 6000 0 4.
7000 11.7000 1
5.5000 12.2000 1
6. 0000 13. 1000 0 8.
1000 15.7000 0 4.4000
16.1000 0
5.9000 20.2000 r0
6.5000 20.3000 1
7.6000 24.3000 r0

注意文件的要求如下:

1. 每行包含3个条目。
2. 对于每一行，第一个条目是作业到达系统的时间(即作为新作业)，第二个条目是其永久离开系统的时间，第三个条目是作业的分类，与表3中第4列的分类方式相同。一个作业可能的分类为0、0和1。您应该能够将上述文件的内容与第4.2节中的示例1相协调。
3. 作业必须按照升序完成时间排序。
4. 如果模拟处于跟踪模式，我们期望在处理完所有作业后模拟结束。因此，dep_*.txt中的行数应该等于作业的数量。
5. 如果模拟处于随机模式，则该文件应包含截至time_end已完成的所有作业。

mrt_*.txt和dep_*.txt中的所有平均响应时间、到达时间和完成时间都应该打印为精确到小数点后4位的浮点数。请注意，你的模拟应该以完全浮点精度执行，你应该只在写输出文件时进行四舍五入。

6.3 测试框架

当你提交你的项目时，你必须包含一个名为run_test.sh的Linux bash shell脚本，这样我们才能在CSE系统上运行你的程序。这个shell脚本是必需的，因为您可以使用自己选择的计算机语言。

让我们首先回忆一下，每个测试都由四个配置文件指定，并且应该产生两个输出文件。例如，测试号0由配置文件mode_0.txt、interarrival_0.txt、service_0.txt和para_0.txt指定;而测试号0预计会产生

输出文件mrt_0.txt和dep_0.txt。

在进行测试时，我们将使用以下目录结构。

包含run test.sh的目录 -
├─配置/
└─输出/

我们将把所有测试的所有配置文件放在config/子目录下。你应该把所有的输出文件写到output/子目录下。

要运行test number 0，我们使用shell命令:

/ run_test.sh 0。

预期的行为是，你的模拟程序将从config/中读取测试号0的配置文件，执行模拟并在output/中创建输出文件。

类似地，要运行测试号1，我们使用shell命令:
/ run_test.sh 1。

这意味着shell脚本run_test.sh有一个输入参数，该参数是要使用的测试号。

让我们暂时假设您使用Python(版本3)来编写模拟程序，并将模拟程序称为main.py。如果文件main.py和run_test.sh在同一个目录下，那么run_test.sh可以是下面的一行shell脚本:

Python3 main.py $1

shell脚本会将测试号(在输入参数$1中)传递给模拟程序main.py。这也意味着你的模拟程序应该接受一个输入参数，即测试号。

为了避免您不熟悉shell脚本，我们提供了两个示例文件:run_test.sh和main.py，以说明shell脚本和Python(版本3)文件之间的交互。你需要确保run_test.sh是可执行文件。(你可以通过“chmod u+x run_test.sh”命令使shell脚本run_test.sh exe-cutable。)如果运行。/run_test.sh 2命令，它会在output/目录下生成一个名为dummy_2.txt的文件。您还可以尝试使用样例shell脚本的其他输入参数。您可以使用这些示例文件来帮助您开发代码。

如果你使用C、c++或Java，那么你的run_test.sh应该先编译源代码，然后再运行可执行文件。当然，你应该把测试号作为输入传递给可执行文件。

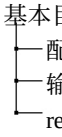
你可以把你的代码放在包含run_test.sh的同一目录中，或者放在它下面的子目录中。例如，你可以像下面这样为你的代码创建一个子目录src/:

包含run test.sh的目录 -
├─配置/
├─输出/
└─src /

6.4 示例文件

你应该从课程网站上的项目页面下载sample project files.zip文件。zip归档文件的目录结构如下:

基本目录包含cf输出ref.py，运行test.sh和main.py



zip-archive的详细信息如下:

· 子目录config/包含可用于测试的配置文件。

-文件mode_1.txt, mode_1.txt，…，和mode_7.txt。请注意，测试0-3用于跟踪模式，而测试4-6用于随机模式。

-文件para_*.txt, interarrival_*.txt和service_*.txt为*从0到6，作为模拟的输入。

-注意，测试0-2与第4节中的示例0-2相同。

· 子目录“/”为空。你的模拟程序应该把输出文件放在这个子目录中。

子目录ref/包含预期的模拟结果。

—“mrt_*_ref.txt”和“dep_*_ref.txt”作为输出的参考文件。对于测试0-3，您应该能够在mrt_*_ref.txt和dep_*_ref.txt中重现结果。但是，由于测试4-6处于随机模式，因此您将无法在输出文件中重现结果。他们已经提供，以便您可以检查文件的预期格式。

· Python文件cf_output_with_ref.py说明我们将如何将您的输出与参考输出进行比较。该文件接受一个输入参数，即测试号。例如，如果你想检查测试0的模拟输出，你可以使用:

Python3 cf_output_with_ref.py 0

注意以下事项:

文件cf_output_with_ref.py使用前面所示的目录结构。

-对于跟踪模式，我们将检查您的平均响应时间，出发时间和分类。请注意，我们不是在寻找一个完全匹配，而是你的结果是否在一个有效的公差范围内。跟踪模式的容差是10^{-3}，这对于小数点后4位的数字是相当慷慨的。

-对于随机模式，我们将只检查平均响应时间。从样例文件中可以看出，我们检查平均响应时间是否在一个区间内。我们通过以下方法获得这个间隔:(i)我们首先对系统进行多次模拟;(ii)然后我们使用模拟结果来估计最大和最小平均响应时间;(iii)我们使用估计的最大值和最小值来形成一个区间;(iv)为了提供一些由于随机性而产生的容忍度，我们进一步扩大这个区间。

-请注意，我们使用一个非常慷慨的容差，所以如果你的平均响应时间没有通过测试，那么很可能你的模拟程序是不正确的。

· 在6.3节中提到的run_test.sh和main.py文件。

6.5 在CSE系统上进行自己的测试

注意6.3节中提到的关于目录结构的假设是很重要的。你必须确保你的shell脚本和程序文件是按照这个假设编写的。

由于我们将在CSE系统上测试您的作品，我们强烈建议您在提交之前在CSE系统上进行以下测试。

· 在您的CSE帐户中创建一个新文件夹并cd到该文件夹。我们将把这个目录称为基目录。

将shell脚本run_test.sh和程序文件复制到基本目录

复制config和ref目录，以及它们的内容到基本目录
-创建一个空目录输出

· 确保你的shell脚本是可执行的，使用命令“chmod u+x run_test.sh”

· 为每个测试一个接一个地运行shell脚本。确保每次运行都在输出目录中为该测试生成相应的输出文件。
· 将cf_output_with_ref.py复制到基本目录。运行它，将你的输出与参考输出进行比较。

这些步骤与我们将用于测试的步骤相同。重要的是要知道，我们将在运行代码之前创建一个空的输出/目录。这意味着你的代码不需要创建输出/目录。

提交门户将尝试对你提交的文件运行编号为0的测试，参见章节7.3。

6.6 入门和基础代码

对于这个项目，我们不要求你从头开始编写代码。您可以使用以下方式来构建您的项目:(i)来自COMP9334的示例代码;或(ii)公共领域的代码，只要它满足以下要求。

如果你打算使用Python 3来编写模拟代码，最好的方法是使用第4B周修正问题的解决方案提供的M/M/ M模拟代码，并从那里进行修改。跟踪驱动仿真的示例代码随第4B周的讲座一起提供。

在公共领域，Python 3、C、c++和Java中也有很多离散事件模拟代码。只要满足以下要求，你就可以使用公共领域的代码作为项目工作的基础:

1. 代码有一个明确可识别的作者
2. 代码有一个日期，在这个项目文档发布的日期之前。
3. 您向我们提供源代码的URL。
4. 你要清楚地说明你对原始代码所做的改变，以使其适应这个项目的规范。

如果您在项目中使用了任何公共领域代码，您的项目报告必须包含满足上述四个要求的信息。

如果你想使用某个公共领域的源代码，但你不确定它是否符合我们的要求，你可以在论坛上用私信咨询讲师。

如果您的项目工作基于COMP9334样例代码，那么您的报告必须声明已经使用了COMP9334样例代码，并提供满足上述需求4的信息。

7 项目需求

这是一个单独的项目。你需要独立完成这个项目。

7.1 提交需求

您的提交应包括以下内容:

1. 书面报告

(a)只需要电子版。
(b)必须是acrobatpdf格式。
(c)必须命名为“report.pdf”。

(d)报告必须包括第6.6节要求的信息。

2. 程序源代码:

(a)用于做模拟
(b) shell脚本run_test.sh，参见6.3节。

3. 任何支持材料，例如由模拟创建的日志，您为处理数据而编写的脚本等。

评估将基于您的提交和在CSE系统上运行您的代码。提交正确版本的代码并确保它在CSE系统上运行是很重要的。

重要的是，你要写一份清晰而切中要害的报告。你需要意识到你是在给读者(报告的目标读者)写报告，而不是为你自己写报告。你的报告将主要根据你所做的工作的质量来评估。你不需要在报告中包含任何背景资料。你只需要谈谈你是如何做这项工作的，我们在7.2节中提供了一套评估标准来帮助你写报告。为了让你展示这些标准，你的报告应该参考你的程序、脚本和其他材料，以便我们了解它们。

7.2 评估标准

我们将根据以下标准评估您的项目质量:

1. 你的模拟代码的正确性。为此，我们将:
(a)使用测试用例测试你的代码

(b)在你的报告中寻找证据，证明你已经验证了到达间概率分布的正确性，将作业发送到组0或组1的概率，以及服务时间分布。你可以在提交的报告中包括适当的支持材料来证明这一点。

(c)在你的报告中寻找证据，证明你已经验证了模拟代码的正确性。虽然我们已经给了你们测试用例，但我们从来没有声称这些测试用例足以验证你们的模拟输出的正确性。

你可以通过论证我们提供的测试用例是足够的以及为什么是足够的来满足这个评估标准。的选择。您可以派生测试用例来测试您的代码，并解释新测试用例的基本原理。您可以在提交的文件中包含适当的支持材料来证明这一点。

2. 你需要证明你的结果是可重复的。你应该在报告中提供这方面的证据。

3. 对于确定n_0的合适值以最小化加权平均响应时间的部分，我们将在您的报告中寻找以下内容:

(a)使用统计上合理的方法分析模拟结果的证据

(b)关于你们如何选择模拟和数据处理参数的解释，例如模拟的长度、重复的次数、瞬态结束等。

上述标记标准密切遵循我们在离散事件模拟讲座中所提倡的信息。你需要确保你的仿真代码是正确的，同时你需要考虑仿真参数的选择，并使用统计合理的方法来比较系统。如果你想做好这个项目，你必须确保你涵盖了以上所有方面。大约一半的项目分数落在上面的第1点和第2点，大约一半的项目分数落在上面的第3点。

7.3 如何提交

你应该把你的报告、shell脚本、程序和支持材料“压缩”到一个名为project.zip的文件中。提交系统只会接受这个文件名。如果在压缩时需要存储目录，则需要使用-r开关来保留相对路径。

你应该通过课程网站提交你的作品。你提交的作品大小不能超过20MBytes。

您可以在截止日期前多次提交。较晚的提交会覆盖较早的提交，所以请确保提交正确的文件。我们只会标记你最后提交的文件。不要等到最后一刻才提交，因为可能有技术或通信错误，你没有时间纠正。

当您提交文件时，提交门户将解压缩您的project.zip并运行测试脚本。该脚本将搜索你的run_test.sh(使用shell命令find .sh)。-name test.sh)，如果找到唯一的run_test.sh，则执行样例test 0。如果测试脚本说它找不到你的run_test.sh或者它找到了多个名为run_test.sh的文件，那么你应该重新提交，并且你应该确保在你的zip存档中只有一个run_test.sh文件。你可以在你准备好模拟部分之后，在你尝试设计之前做这个测试。因为之后的提交会覆盖之前的提交，所以你可以更早地完成这个测试。

8 进一步的项目条件

1. 这个项目的总分是30分。
2. 提交截止日期为2024年4月19日星期五下午5点。在截止日期之后提交的作品将被处以每天5%的罚款。罚款适用于如果提交不迟，您将收到的分数。2024年4月24日(星期三)下午5点之前，将不接受逾期提交的作品。

3. 如果你使用计算机程序来完成你作业的任何部分，你必须提交程序，否则你将失去该部分的分数。这一要求既适用于计算机模拟程序，也适用于计算机统计分析程序。

4. 项目附加条件:

· 本项目不允许联合作业。

-这是一个独立的项目。如第6.6节所述，您必须标识您所使用的代码的源代码，无论它是来自COMP9334还是公共领域。

-不要向COMP9344的教学人员以外的任何人寻求帮助。

不要在课程论坛上发布你的项目工作或代码。
-提交的项目通常会被自动或手动检查其他人写的工作。

基本原理:该项目旨在培养解决问题所需的个人技能。使用别人写的或抄袭的作品/代码会阻碍你学习这些技能。其他CSE课程侧重于团队工作所需的技能。

· 本项目不允许使用人工智能生成工具，如ChatGPT。理由:我们已经允许您使用公共领域代码作为开发项目的基础，因此您没有必要使用ChatGPT。我们对ChatGPT的测试发现，它不能为我们提供一段完整的运行代码来模拟M/M/1队列。

· 不允许分享，发布或分发您的项目工作。

-不要向COMP9334的教学人员以外的任何人提供或展示你的项目作品。例如，不要把你的作品发给朋友。

不要通过互联网发布项目代码。例如，不要将你的项目放在公共GitHub存储库中。

理由:通过发布或分享你的作品，你可以促进其他学生使用你的作品。如果其他学生发现你的专题作品并将其部分或全部作为自己的作品提交，你可能会被卷入学术诚信调查。

· 在完成COMP9334后，不允许分享，发布或分发您的项目工作。

-例如，在COMP9334提供结束后，不要将您的项目放在公共GitHub存储库中。

基本原理:COMP9334可以重用涵盖类似概念和内容的项目主题。如果未来学期的学生发现你的项目工作，并将其部分或全部作为自己的工作提交，你可能会被卷入学术诚信调查。

参考文献。

[1] Mor Harchol-Balter and Ziv Scully. The Most Common Queueing Theory Questions Asked by Computer Systems Practitioners.First International Workshop on Teaching Performance Analysis of ComputerSystems (TeaPACS 2021) In conjunction with the IFIP Performance 2021 Conference. Milan, Italy, Nov 2021. DOI 10.1145/3543146.3543148

[2] Mor Harchol-Balter. Performance Modeling and Design of Computer Systems. Cambridge University Press (2013).