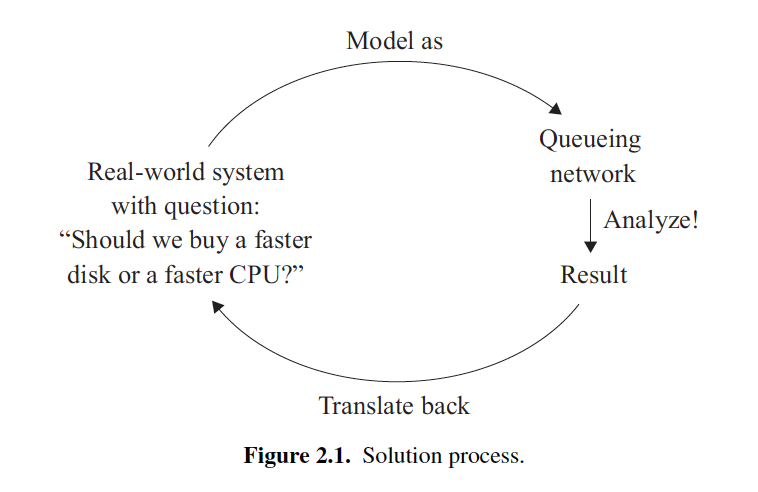
{{title}}

排队论术语

2.1 我们的目标

排队论是研究网络和系统中排队行为的理论。图 2.1 展示了解决问题的过程。在第 1 章中，我们讨论了排队论作为设计工具的威力示例。在本章中，我们从头开始，定义排队论中使用的术语。

2.2 单服务器网络

一个排队网络由多个服务器组成。

最简单的排队网络例子是单服务器网络，如图 2.2 所示。本节的讨论仅限于使用“先到先服务”（**First-Come-First-Served, FCFS**）顺序的单服务器网络。你可以将服务器想象为一块CPU。

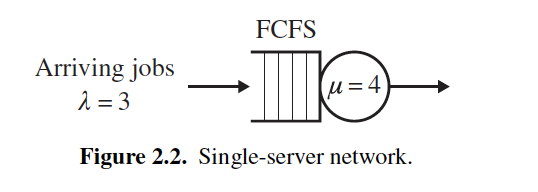


Fig. alt text

单服务器网络有几个相关参数：

* **服务顺序**：这是服务器为作业提供服务的顺序。除非另有说明，假设为“先到先服务”（**First-Come-First-Served, FCFS**）。
* **平均到达率**：这是作业到达服务器的平均速率，通常用 ( \lambda ) 表示（例如，( \lambda = 3 ) 个作业/秒）。
* **平均到达间隔时间**：这是连续作业到达之间的平均时间（例如，( 1/\lambda = 1/3 ) 秒）。
* **服务需求，大小**：作业的“大小”通常由随机变量 ( S ) 表示，这是作业在没有其他作业排队的情况下，在该服务器上运行所需的时间。在排队模型中，大小（也称为服务需求）通常与服务器相关联（例如，作业在该服务器上需要 5 秒）。
* **平均服务时间**：这是 ( S ) 的期望值，即作业在 CPU 上完成服务所需的平均时间，其中“服务”不包括排队时间。例如在图 2.2 中，( E[S] = 1/4 ) 秒。
* **平均服务速率**：这是作业完成服务的平均速率，通常用 ( \mu ) 表示（例如，( \mu = 4 ) 个作业/秒 = ( 1/E[S] )）。请注意，这种表达方式与我们日常讨论服务器的方式有所不同。例如，我们没有提到 CPU 的绝对速度，而是通过其处理的作业来定义其速度。

在日常对话中，我们可能会这样说：

* 作业的平均到达率是每秒 3 个作业。
* 作业有不同的服务需求，但每个作业平均需要 5,000 个周期。
* CPU 的速度是每秒 20,000 个周期。

这意味着每秒有平均 15,000 个周期的工作到达 CPU，而 CPU 每秒可以处理 20,000 个周期的工作。

在排队论的表达方式中，我们不会提到“周期”这个词。相反，我们会说：

* 作业的平均到达率是每秒 3 个作业。
* CPU 服务作业的平均速率是每秒 4 个作业。

这种表达方式省略了一些细节，从而使问题更容易思考。你应该能够在这两种表达方式之间自由转换。

在单服务器系统的上下文中，我们考虑以下常见的性能指标：

* **响应时间、周转时间、系统时间或停留时间（T）**：我们定义作业的响应时间为 ( T = t\_{\text{depart}} - t\_{\text{arrive}} )，其中 ( t\_{\text{depart}} ) 是作业离开系统的时间，( t\_{\text{arrive}} ) 是作业到达系统的时间。我们关心的是 ( E[T] )（平均响应时间），( \text{Var}(T) )（响应时间的方差）以及响应时间的尾部行为，即 ( P{T > t} ) 的概率。
* **等待时间或延迟（TQ）**：这是作业在队列中等待、未被服务的时间，也称为“排队时间”或“浪费时间”。请注意，( E[T] = E[TQ] + E[S] )。在**先到先服务（FCFS）**顺序中，等待时间可以定义为作业从到达系统到首次接受服务的时间。
* **系统中的作业数量（N）**：这包括在队列中的作业以及正在被服务的作业（如果有的话）。
* **队列中的作业数量（NQ）**：这仅表示等待中的作业数量（即在队列中排队的作业）。

我们可以立即对单服务器网络做出一些观察。首先注意到，随着 ( \lambda )（平均到达率）的增加，所有上述性能指标都会增加（变差）。同样，随着 ( \mu )（平均服务速率）的增加，所有上述性能指标都会减少（变好）。我们要求 ( \lambda \leq \mu )（假设始终 ( \lambda < \mu )）。

**问题**：如果 ( \lambda > \mu )，会发生什么？

**回答**：如果 ( \lambda > \mu )，队列长度会随着时间推移趋于无穷大。

**问题**：你能提供直观的解释吗？

**回答**：考虑一个较大的时间 ( t )。假设 ( N(t) ) 是时刻 ( t ) 系统中的作业数量，( A(t) )（分别是 ( D(t) )）表示到达的作业数（分别是完成的作业数），那么我们有： [ E[N(t)] = E[A(t)] - E[D(t)] \geq \lambda t - \mu t = t(\lambda - \mu) ] （不等式的来源是，由于服务器并非总是忙碌，因此到 ( t ) 时完成的作业数量的期望实际上小于 ( \mu t )）。现在注意，如果 ( \lambda > \mu )，则 ( t(\lambda - \mu) \to \infty )，随着 ( t \to \infty )，队列长度将趋于无穷大。

*译者注: 变差可能是渐进的, 比如CPU出现了竞争, 各种其他资源出现了竞争,比如网络出现了丢包和重传, 比如某些同步操作出现了自旋等待*

在本书中，我们假设 ( \lambda < \mu )，这是保持系统稳定性（防止队列无限增长）所必需的。对于 ( \lambda \geq \mu ) 的情况，我们在第 9 章中稍有讨论。

**问题**：在前述稳定性条件（( \lambda < \mu )）下，假设到达时间分布和服务时间分布是确定性的（即，它们都是常数）。那么 ( TQ ) 和 ( T ) 分别是什么？

**回答**：( TQ = 0 )，( T = S )。

因此，排队（等待）是由于服务时间和/或到达时间分布的可变性造成的。这里有一个例子说明可变性如何导致队列：我们将时间离散化。假设每个时间步，作业到达的概率为 ( p = 1/6 )，而每个时间步，作业完成的概率为 ( q = 1/3 )。在这种情况下，如果连续发生多次到达而没有完成作业，则队列有非零的概率暂时增加。

2.3 排队网络的分类

排队网络可以分为两类：**开放网络**和**闭合网络**。随机过程的书籍（如 [149, 150]）通常只讨论开放网络。相比之下，系统性能分析的书籍（如 [117, 125]）几乎专门讨论闭合网络。第 2.4 节介绍了开放网络，第 2.6 节介绍了闭合网络。

2.4 开放网络

开放排队网络具有外部到达和离开。本节展示了四个开放网络的示例。

**示例**：单服务器系统

这个例子已经在图 2.2 中展示过。

**示例**：具有概率路由的排队网络

这个例子展示在图 2.3 中。在这里，服务器 (i) 以到达率 (r\_i) 接收外部到达（“外部到达”）。服务器 (i) 还接收来自其他一些服务器的内部到达。一个在服务器 (i) 完成服务的数据包接着会以概率 (p\_{ij}) 被路由到服务器 (j)。我们甚至可以让概率取决于数据包的“类别”，因此并非所有数据包都必须遵循相同的路由方案。

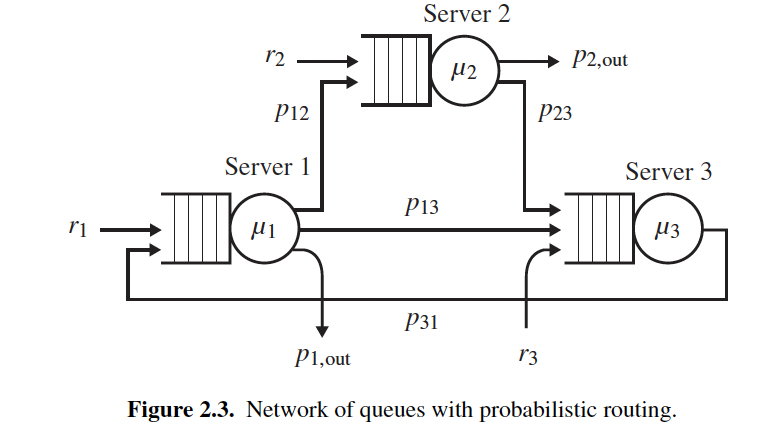


Fig. alt text

**应用**：例如，在对互联网中的数据包流进行建模时，可以让数据包的类别（以及因此它的路由）取决于其源 IP 和目标 IP 地址。在对延迟进行建模时，每条网络线路可以被替换为一个服务器，以用于模拟线路延迟。目标可能是预测特定路由上数据包的平均往返时间，假设有其他数据包同时存在。我们将在第 18 章中解决这个问题。

**示例**：具有非概率路由的排队网络

这个例子展示在图 2.4 中。在这里，所有作业遵循预定的路径：从 CPU 到磁盘 1，再到磁盘 2，接着返回磁盘 1，再到磁盘 2，最后到磁盘 1，作业完成后离开系统。

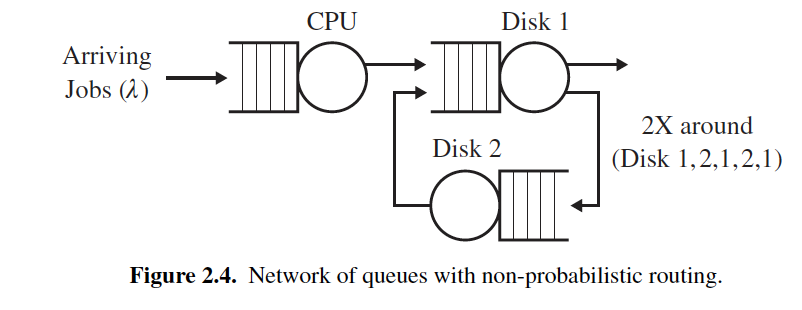


Fig. alt text

*译者注: 比如数据库的高可用, 复制同步*

**示例**：有限缓冲

图 2.5 展示了一个带有有限缓冲的单服务器网络示例。任何在到达时发现没有空位的作业将被丢弃。

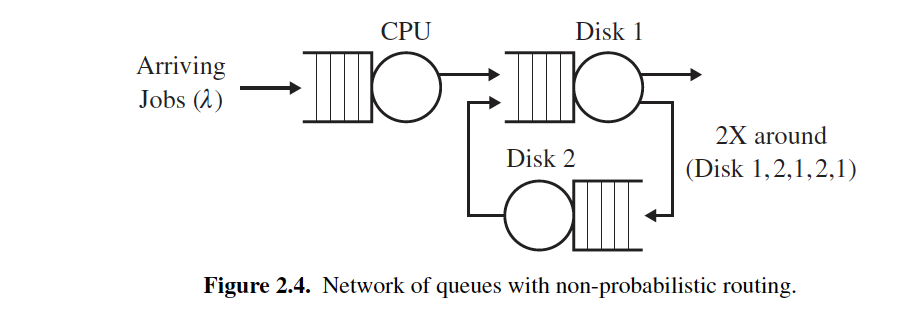


Fig. alt text

2.5 更多的性能指标：吞吐量和利用率

我们已经看到了四个性能指标：( E[N] )、( E[T] )、( E[NQ] ) 和 ( E[TQ] )。尽管这些指标是针对单服务器系统的，但它们也可以用于描述多服务器、多队列系统的性能。例如，( E[T] ) 表示作业在整个系统中所花费的平均时间，包括在各种队列中的时间以及在各个服务器上接受服务的时间，而 ( E[TQ] ) 则仅指作业在各个队列中“浪费”的平均等待时间。如果我们想单独指代系统中的第 (i) 个队列，通常会写作 ( E[N\_i] ) 来表示在服务器 (i) 处排队和正在服务的作业的期望数量，( E[T\_i] ) 表示作业在服务器 (i) 处排队和接受服务的期望时间。

现在我们引入两个新的性能指标：**吞吐量**和**利用率**。**吞吐量**可能是日常对话中最常用的性能指标。每个人都想要更高的吞吐量！让我们看看为什么。

**问题**：最大化吞吐量与最小化响应时间之间有何关系？例如，在图 2.6 中，哪个系统的吞吐量更高？

**回答**：我们很快就会知道。

让我们从定义**利用率**开始。

**设备利用率**（( \rho\_i )）是设备 ( i ) 忙碌的时间占总时间的比例。请注意，我们当前对利用率的定义仅适用于单个设备（服务器）。当设备已知时，我们简写为 ( \rho )（省略下标）。

假设我们观察设备 ( i ) 很长一段时间。设 ( \tau ) 表示观察期的长度，设 ( B ) 表示观察期间设备非空闲（忙碌）的总时间。则：

[ \rho\_i = \frac{B}{\tau} ]

**设备吞吐量**（( X\_i )）是设备 ( i ) 的作业完成率（例如，作业/秒）。系统的吞吐量（( X )）是系统内作业完成的速率。

设 ( C ) 表示在时间 ( \tau ) 内在设备 ( i ) 上完成的作业总数。则：

[ X\_i = \frac{C}{\tau} ]

那么，( X\_i ) 和 ( \rho\_i ) 之间的关系是什么呢？我们可以这样表示：

[ \frac{C}{\tau} = \frac{C}{B} \cdot \frac{B}{\tau} ]

**问题**：那么，( \frac{C}{B} ) 是什么？

**回答**：实际上，( \frac{B}{C} = E[S] )，所以 ( \frac{C}{B} = \frac{1}{E[S]} = \mu\_i )。

因此我们有：

[ X\_i = \mu\_i \cdot \rho\_i ]

还有另一种通过条件概率推导这个表达式的方法：

[ X\_i = \text{设备 ( i ) 的平均完成速率} = E[\text{设备 ( i ) 的完成速率 | 设备 ( i ) 忙碌}] \cdot P{\text{设备 ( i ) 忙碌}} + E[\text{设备 ( i ) 的完成速率 | 设备 ( i ) 空闲}] \cdot P{\text{设备 ( i ) 空闲}} = \mu\_i \cdot P{\text{设备 ( i ) 忙碌}} + 0 = \mu\_i \cdot \rho\_i ]

或者，等效地表示为：

[ \rho\_i = X\_i \cdot E[S] ]

这种表述有一个名称：**利用率定律**（Utilization Law）。

*译者注: 但是这对于IO读写和AI不一定成立, 虽然也是单线程的, IO可以多个请求在缓冲区内合并成一个真实的物理请求, 只中断一次, 对于AI, 有个概念叫批处理, 数据库也有类似的概念, 这种合并处理会提高吞吐, 但是利用率可能还是不变*

**示例：单服务器网络：吞吐量是多少？**

在图 2.7 中，我们有一个单服务器系统。

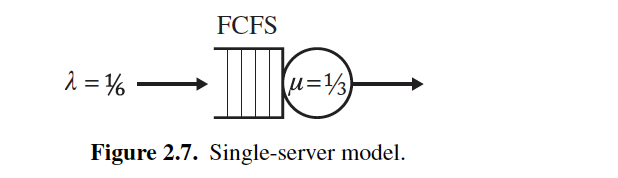


Fig. alt text

**问题**：( X ) 是多少？

**回答**：( X = \rho \cdot \mu )。那么 ( \rho ) 是什么呢？在第 6 章中，我们将证明 ( \rho = \frac{\lambda}{\mu} )。现在，这里有一个直观的但不严格的解释方式：

[ \rho = \text{服务器忙碌的时间比例} = \frac{\text{作业所需的平均服务时间}}{\text{到达间隔的平均时间}} = \frac{1/\mu}{1/\lambda} = \frac{\lambda}{\mu} ]

因此，吞吐量为：

[ X = \rho \cdot \mu = \frac{\lambda}{\mu} \cdot \mu = \lambda ]

由此可见，吞吐量与服务速率无关！

特别是，在图 2.6（再次展示在图 2.8 中）的示例中，两种系统的吞吐量都是 ( 1/6 ) 作业/秒。在处理器速度较快的情况下，响应时间会下降，队列长度也会减少，但吞吐量 ( X ) 并没有变化。因此，**较低的响应时间与较高的吞吐量无关**。

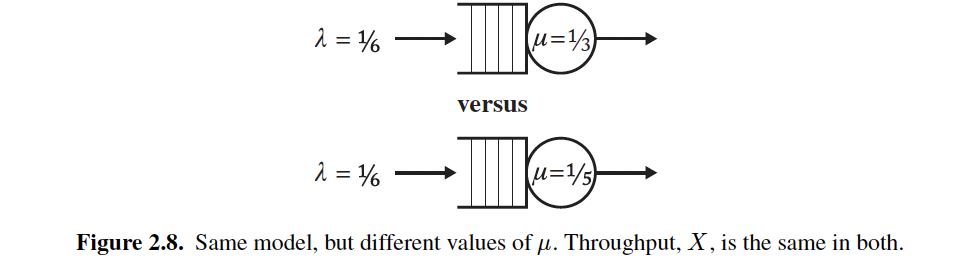


Fig. alt text

**问题**：解释为什么 ( X ) 没有变化。

**回答**：无论我们将 ( \mu ) 提高到多高，完成率依然受到到达率的限制：**“输入速率 = 输出速率”**。改变 ( \mu ) 只会影响最大可能的 ( X )，但不会改变实际的 ( X )。请注意，由于我们假设系统是稳定的，那么对于较大的时间 ( t )，在 ( t ) 期间的到达数大约等于在 ( t ) 期间的完成数。

**示例**：具有概率路由的排队网络：吞吐量是多少？

在图 2.3 中，( r\_i ) 表示到达服务器 ( i ) 的外部平均到达率，( \mu\_i ) 表示服务器 ( i ) 的平均服务速率。

**问题**：图 2.3 中的系统吞吐量 ( X ) 是多少？

**回答**：( X = \sum\_{i} r\_i )。

**问题**：服务器 ( i ) 的吞吐量 ( X\_i ) 是多少？

**回答**：设 ( \lambda\_i ) 表示进入服务器 ( i ) 的总到达率。那么 ( X\_i = \lambda\_i )。但是要获得 ( \lambda\_i )，我们需要解以下联立方程：

[ \lambda\_i = r\_i + \sum\_{j} \lambda\_j P\_{ji} ]

**问题**：这些方程中的 ( r\_i ) 有什么约束？

**回答**：为了让网络达到**“平衡”（进入服务器的流量 = 离开服务器的流量**，我们必须有 ( \lambda\_i < \mu\_i ) 对于所有 ( i )，这对 ( r\_i ) 施加了约束（参见练习 2.1）。

**示例**：具有非概率路由的排队网络：吞吐量是多少？

**问题**：图 2.4 中的 ( X ) 是多少？

**回答**：( X = \lambda )。

**问题**：( X\_{\text{Disk1}} ) 和 ( X\_{\text{Disk2}} ) 是多少？

**回答**：( X\_{\text{Disk1}} = 3\lambda )，( X\_{\text{Disk2}} = 2\lambda )。

**示例**：有限缓冲：吞吐量是多少？

在图 2.5 中，外部到达率是 ( \lambda )，服务速率是 ( \mu )。

**问题**：( X ) 是多少？

**回答**：( X = \rho \mu )。但我们需要通过随机分析来确定 ( \rho )，因为它不再是简单的 ( \lambda / \mu )。请注意，( X < \lambda )，因为有一些到达作业会被丢弃。

2.6 闭合网络

闭合排队网络没有外部到达或离开。它们可以分为两类，如图 2.9 所示。

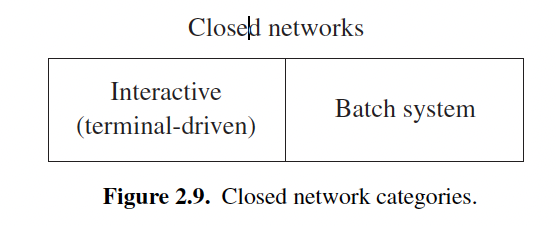


Fig. alt text

2.6.1 交互式（终端驱动）系统

图 2.10 展示了一个交互式（终端驱动）系统的示例。终端代表用户，每个用户发送一个作业到“中央子系统”，然后等待响应。中央子系统是一个排队网络。用户在其前一个作业返回之前无法提交下一个作业。因此，系统中的作业数量是固定的（等于终端的数量）。这个数量有时被称为负载或 **MPL**（多道程序级别），不要与设备利用率混淆。

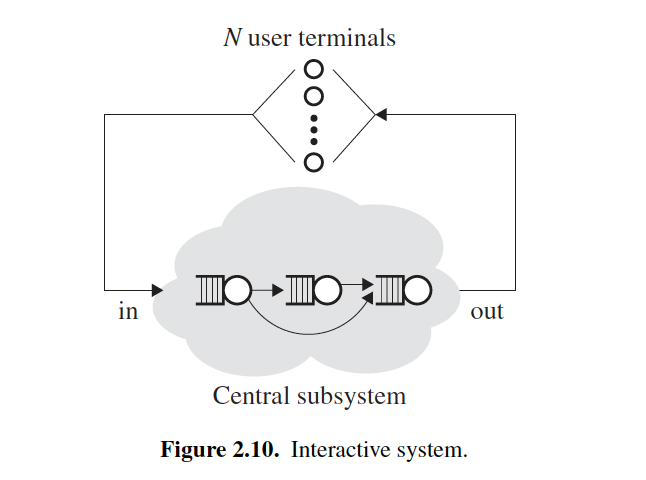


Fig. alt text

*译者注: 现代的终端会具有多进程和background进程的能力, 和这个假设不一致*

有一个思考时间 ( Z )，它是一个随机变量，表示每个终端在接收到一个作业结果后到发送下一个作业之间的时间。请注意，中央子系统中的作业数量最多等于终端数量，因为某些用户可能处于“思考”状态。

图 2.10 中所示的交互式系统的一个示例是数据输入应用程序。( N ) 个用户分别坐在终端前，填入屏幕上的条目。屏幕上有几个字段需要填写，完成后整个屏幕会提交到中央子系统进行相应处理和数据库更新。在上一个更新完成之前，无法填写新的屏幕。思考时间 ( Z ) 是用户键入数据的时间。

单个用户（终端）在“思考状态”和“提交状态”之间切换，如图 2.11 所示。

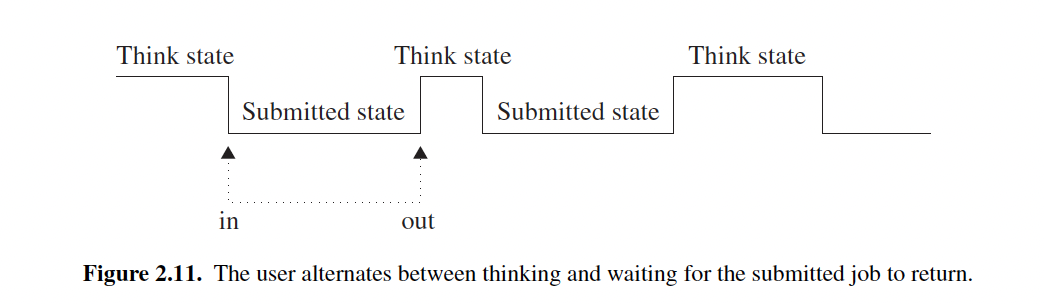


Fig. alt text

**问题**：你如何定义交互式系统的响应时间？

**回答**：响应时间是指作业在图 2.10 和图 2.11 中从“进入”到“离开”所花费的时间。我们用 ( \text{E[Response Time]} ) 或 ( \text{E[R]} ) 表示从“进入”到“离开”的平均时间，以区别于 ( \text{E[T]} )，其定义为：

[ \text{E[T]} = \text{E[R]} + \text{E[Z]} ]

**重要提示**：尽管在开放系统中“响应时间”用随机变量 ( T ) 表示，但在闭合交互系统中，我们将 ( T ) 称为**系统时间**（或“系统中的时间”），并保留随机变量 ( R ) 表示响应时间。

**目标**：交互式系统的目标是找到一种方法，使尽可能多的用户能够同时进入系统并完成工作，同时保持 ( \text{E[R]} ) 足够低。请注意，交互式系统与开放系统非常不同，因为 ( N ) 的微小变化会对系统行为产生深远的影响。

系统设计人员通常会问以下问题：

* **给定原始系统，如何提高 ( N ) 的同时保持 ( \text{E[R]} ) 低于某个阈值？即，( \text{E[R]} ) 随 ( N ) 的增长如何变化？**
* **假设多道程序级别 ( N ) 固定。如果我们能够对中央子系统进行改动（即使某些设备变得更快），哪种改动能最有效地改善 ( \text{E[R]} )？**

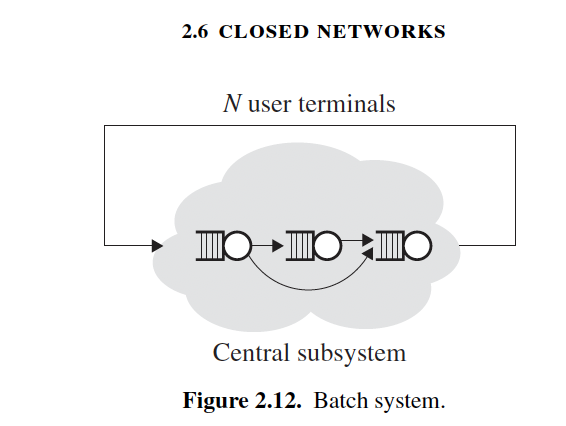
**问题**：假设我们正在对一个网站的性能进行建模。你会将网站建模为一个闭合交互系统还是开放系统？

**回答**：这个问题尚无定论。有两种类型的研究论文。一方面，当用户点击链接（提交作业）后，通常会等待结果再点击其他链接。因此，用户的行为似乎表明网站是一个闭合系统。另一方面，网站可能有大量用户，每个用户的使用时间非常短暂，因此网站在这方面可能更像一个开放系统。

Schroeder 等人 [165] 提出了“部分开放”系统的概念。这里用户像在开放系统中一样从外部到达，但每次到达时会向系统发出 ( k ) 个请求，每个请求只能在前一个请求完成后才能发出（如同闭合系统）。

2.6.2 批处理系统

图 2.12 展示了一个批处理系统的示例。批处理系统看起来像是一个思考时间为零的交互式系统。然而，对于批处理系统，目标有所不同。批处理系统中，通常在夜间运行许多作业。一个作业完成后，另一个作业立即开始。因此中央子系统中始终有 ( N ) 个作业。多道程序级别（MPL）通常是预先确定并固定的。例如，MPL 可能是能够装入内存的作业数量。

*译者注: 这对银行和电信运营商太常见了*

**目标**：对于批处理系统，目标是获得高吞吐量，以便在夜间尽可能多地完成作业。

系统设计人员通常会问的问题是：“如何改进中央子系统以最大化吞吐量？”

请注意，通常我们受某个固定的最大多道程序级别（MPL）的限制（因为内存容量有限，或出于其他原因，只有这么多作业能被容纳）。因此，我们唯一可以增加吞吐量的方法是通过改变中央子系统的路由或加速某些设备来实现。需要注意的是，在批处理系统中，我们不关心响应时间，因为作业是在夜间运行的。

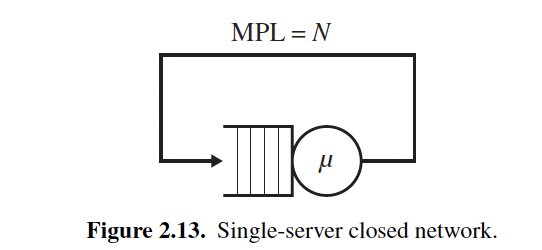
**问题**：在封闭系统中，( X ) 代表什么？

**回答**：( X ) 是每秒完成的作业数。请注意，对于批处理系统，“进入”的作业数等于“离开”的作业数。

2.6.3 封闭系统中的吞吐量

让我们来看一些示例。

**示例**：单服务器

图 2.13 展示了一个包含单服务器的封闭网络。

**问题**：图 2.13 中的吞吐量 ( X ) 是多少？

**回答**：( X = \mu )。

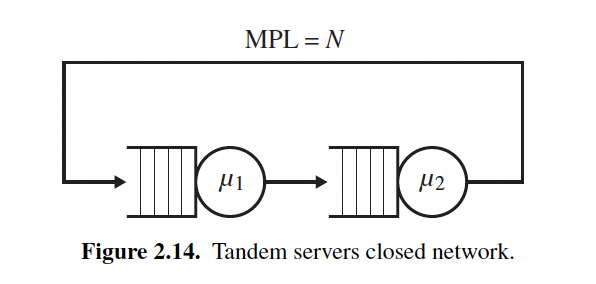
请注意，这与开放网络中的情况非常不同，在开放网络中，吞吐量与服务速率无关！

**问题**：图 2.13 中的平均响应时间 ( \text{E[R]} ) 是多少？

**回答**：对于封闭的批处理系统，( \text{E[R]} = \text{E[T]} )，也就是响应时间和系统内时间是相同的。对于图 2.13，( \text{E[T]} = \frac{N}{\mu} )，因为每个“到达”的作业需要等待前面的 ( N-1 ) 个作业完成，然后再运行。

请注意，( X ) 和 ( \text{E[R]} ) 之间是反比关系！

**示例**：串联服务器

现在考虑一个更复杂的封闭网络的示例，如图 2.14 所示。

**问题**：吞吐量是多少？

**回答**：我们想说 ( X = \min(\mu\_1, \mu\_2) )……

**问题**：为什么这个答案不一定正确？

**回答**：如果我们知道较慢的服务器总是忙碌的，那么这个答案是正确的，但这并不一定是事实。想象一下，当 ( N = 1 ) 时，显然较慢的服务器并不总是忙碌的。

**问题**：好吧，但是当 ( N = 2 ) 时会发生什么？现在看来，慢速服务器总是至少有一个作业在处理，不是吗？

**回答**：不，较慢的服务器仍然不总是忙碌的。我们这里忽略了一个事实，即有时慢速服务器实际上比快速服务器处理得更快——因为这些服务速率只是平均值！那么，我们是否真的需要考虑作业大小分布才能得到确切答案？作业大小分布真的会对答案产生很大影响吗？

我们很快就会回答这些问题……现在，让我们总结开放网络和封闭网络的行为差异，以及为什么我们需要考虑两者。

2.7 封闭网络与开放网络的区别

**开放系统**：

* 吞吐量 ( X ) 与 ( \mu\_i ) 无关。
* 增加 ( \mu\_i ) 倍并不会影响 ( X )。
* 吞吐量和响应时间没有关联。

**封闭系统**：

* 吞吐量 ( X ) 取决于 ( \mu\_i )。
* 如果我们将所有 ( \mu\_i ) 翻倍，同时保持 ( N ) 不变，那么 ( X ) 会发生变化。
* 实际上，我们将在第 6 章中看到，对于封闭系统，**更高的吞吐量 ⇐⇒ 更低的平均响应时间**。

2.7.1 一个关于建模的问题

最后一个问题：几年前，我接到了 IBM 的一些人的电话。他们试图将他们的刀片服务器建模为一个单服务器队列。他们知道服务器的到达率 ( \lambda )（每秒作业数），但他们想知道如何获得 ( \text{E[S]} )，即平均作业大小。

**问题**：在实践中，如何为单服务器系统获取 ( \text{E[S]} )？

**回答**：乍一看，你可能会推理出，因为 ( \text{E[S]} ) 是单个作业在系统中运行所需的平均时间，所以你只需向系统发送一个作业并测量它的响应时间，重复这个实验一百次来获得一个平均值。从理论上讲，这很有道理，但在实践中效果不好，因为缓存条件和其他因素在只有单个作业的情况下与系统运行一段时间后的情况非常不同。

一个更好的方法是记住 ( \text{E[S]} = \frac{1}{\mu} )，因此只需考虑服务器的服务速率（每秒作业数）。要获得 ( \mu )，假设是一个开放系统，我们可以让 ( \lambda ) 越来越高，直到完成速率达到一个稳定值，该值就是服务速率 ( \mu )。

一个更好的主意是将服务器置于一个没有思考时间的闭合系统中。这样，服务器总是有作业要处理。如果我们测量服务器的完成速率（每秒完成的作业数），那么这将给我们服务器的 ( \mu )。然后，( \text{E[S]} ) 就是 ( \mu ) 的倒数。

2.8 相关阅读

理解闭合排队网络特别有帮助的书籍有 Lazowska（第 58-59 页）[117] 和 Menascé（第 84-87 页）[125]。这两本书都非常出色。

令人惊讶的是，关于闭合系统与开放系统的比较，文献中知之甚少。例如，考虑一个负载为 ( \rho ) 的闭合交互式单服务器系统，和一个相应的负载为 ( \rho ) 的开放系统。它们在平均响应时间方面如何比较？服务时间的可变性如何影响闭合系统与开放系统？这些问题和其他许多问题在 [186] 和 [24] 中讨论过，以及在习题 7.2、7.5、13.7 和 13.8 中。另一个问题是，服务器的调度策略（服务顺序）如何影响闭合系统与开放系统。这一问题直到 2006 年才真正讨论过（见 [165]）。关于开放与闭合话题的更近期讨论，推荐阅读 Y.C. Tay 的书籍 [173]。

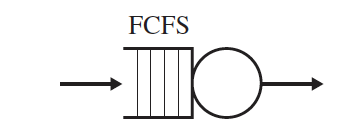
在本章中，我们多次提到，确保到达率小于服务速率（( \lambda < \mu )）是系统稳定的必要条件。这一条件对于我们在本书中讨论的网络来说也是充分的。然而，对于更复杂的排队网络，这通常不是稳定性的充分条件。要了解原因，推荐阅读 Maury Bramson 的论文（见 [29]）。

2.9 练习题

2.1 最大外部到达率

对于图 2.3 中给出的具有概率路由的排队网络，假设每个服务器的平均服务速率为 10 个作业/秒；即 ( \mu\_i = 10 )，对于所有 ( i )。假设 ( r\_2 = r\_3 = 1 )，假设 ( p\_{12} = p\_{2,\text{out}} = 0.8 )，( p\_{23} = p\_{13} = 0.2 )，( p\_{1,\text{out}} = 0 )，且 ( p\_{31} = 1 )。要保持系统稳定，( r\_1 ) 的最大允许值是多少？

2.2 延迟（Slowdown）

(a) 作业以先到先服务（FCFS）顺序到达服务器：

到达率为 ( \lambda = 12 ) 作业/秒。作业大小（服务时间）是独立同分布的随机变量 ( S )，其中

[ S = \begin{cases} 1 & \text{概率为} \ 3/4 \ 2 & \text{否则} \end{cases} ]

你测得的平均响应时间为 ( E[T] = \frac{29}{12} )。基于此信息，计算平均延迟 ( E[\text{Slowdown}] )，其中作业 ( j ) 的延迟定义为 ( \text{Slowdown}(j) = \frac{T(j)}{S(j)} )，( T(j) ) 是作业 ( j ) 的响应时间，( S(j) ) 是作业 ( j ) 的大小。

(b) 如果第 (a) 部分中的服务顺序为最短作业优先（SJF），是否可以使用相同的技术来计算平均延迟？

2.3 调度顺序

(a) 对于单服务器 CPU，作业按某种过程到达，设 SRPT 表示总是为当前最短剩余处理时间的作业提供服务的抢占调度策略（假设我们知道这些信息）。有人声称，对于任何到达序列，包括每个作业的到达时间和大小，SRPT 调度最小化了该到达序列的平均响应时间。证明或反驳这一说法。

(b) 作业的延迟定义为作业的响应时间除以其服务需求。(i) 许多人认为平均延迟比平均响应时间更重要的性能指标。你为什么认为会这样？(ii) 直觉上 SRPT 调度策略应该最小化平均延迟。证明或反驳这一假设。

译者的思考: 开放系统和封闭系统的区别

**开放系统**和**封闭系统**的主要区别在于作业的到达和离开方式，以及系统内作业数量的动态性。以下是两者的关键区别：

1. **作业到达和离开方式**

* **开放系统**：作业从外部源随机到达，处理完成后离开系统。系统中作业的数量随着时间变化，受外部到达率和离开率的影响。
  + **例子**：一个网站的访问者随机访问页面，点击链接后向服务器发送请求，服务器处理完请求后，用户离开网站。
  + **关键特性**：外部到达率和离开率决定了系统的负载。
* **封闭系统**：系统中作业的数量是固定的，作业在完成后不会离开系统，而是重新生成并继续在系统中运行。没有外部到达和离开，系统中的作业数量始终固定。
  + **例子**：一个数据录入系统，其中 ( N ) 个用户每次只能提交一个作业，等处理完后才能提交下一个作业。用户在思考状态和提交状态之间切换。
  + **关键特性**：系统内作业数量始终不变，负载取决于多道程序级别（MPL）和思考时间。

2. **作业数量**

* **开放系统**：作业的数量是动态变化的，随外部到达和离开而波动。系统负载主要由外部到达率（( \lambda )）决定。
* **封闭系统**：系统中作业的数量是固定的，始终等于预先设定的负载（例如用户或任务数量）。作业数量不会随时间变化。

3. **思考时间**

* **开放系统**：没有明确的思考时间概念，因为作业的到达是随机的，可能有时有作业到达，有时没有。
* **封闭系统**：有明确的思考时间（( Z )），表示用户在一个作业完成后，在提交下一个作业之前的时间。这段时间使得系统中的负载不是完全连续的。

4. **稳定性和响应时间**

* **开放系统**：系统需要保持到达率 ( \lambda ) 小于服务率 ( \mu ) 才能保持稳定（即，避免队列无限增长）。响应时间取决于作业的排队和服务时间。
* **封闭系统**：系统内没有外部到达，响应时间是系统中作业的处理时间和思考时间的总和。负载的变化（例如用户数量的增加）会对系统性能产生更大的影响。

5. **应用场景**

* **开放系统**：适用于具有随机到达的场景，如网页请求处理、电话呼叫中心等，系统外部的用户随时可以加入或离开。
* **封闭系统**：适用于用户或任务数量固定的场景，如多用户终端驱动系统、批处理系统，用户在完成一个任务后会继续提交下一个任务。

总结：

* **开放系统**：外部作业随机到达并完成后离开，作业数量随时间变化。
* **封闭系统**：系统中作业数量固定，作业循环在系统内执行，没有外部到达和离开。

[上一章](01Motivating.md)下一章