

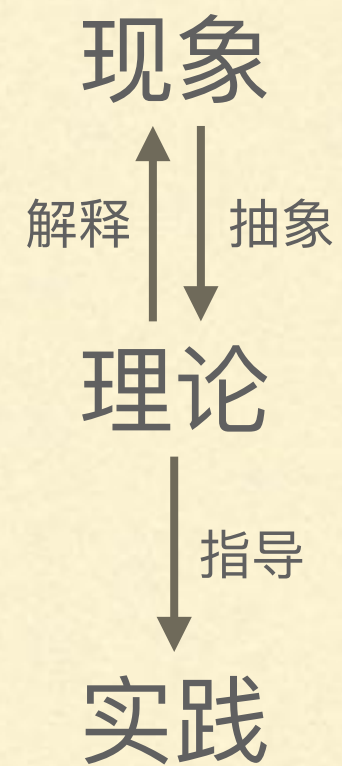


排队的艺术

@孟德

内容

- 到处都在排队
- 如何优雅排队
- 排队论的应用



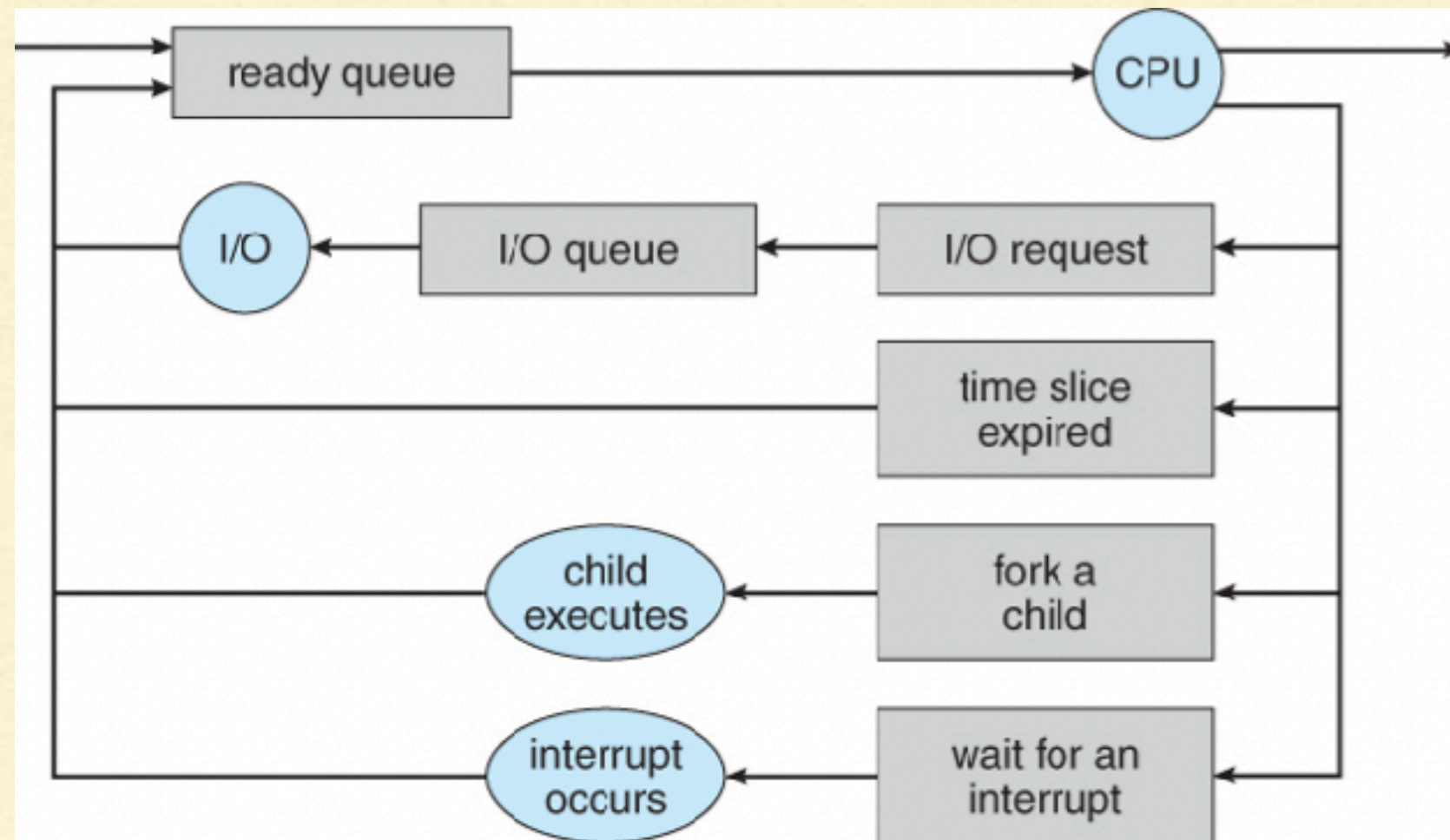
- 到处都在排队



- 到处都在排队

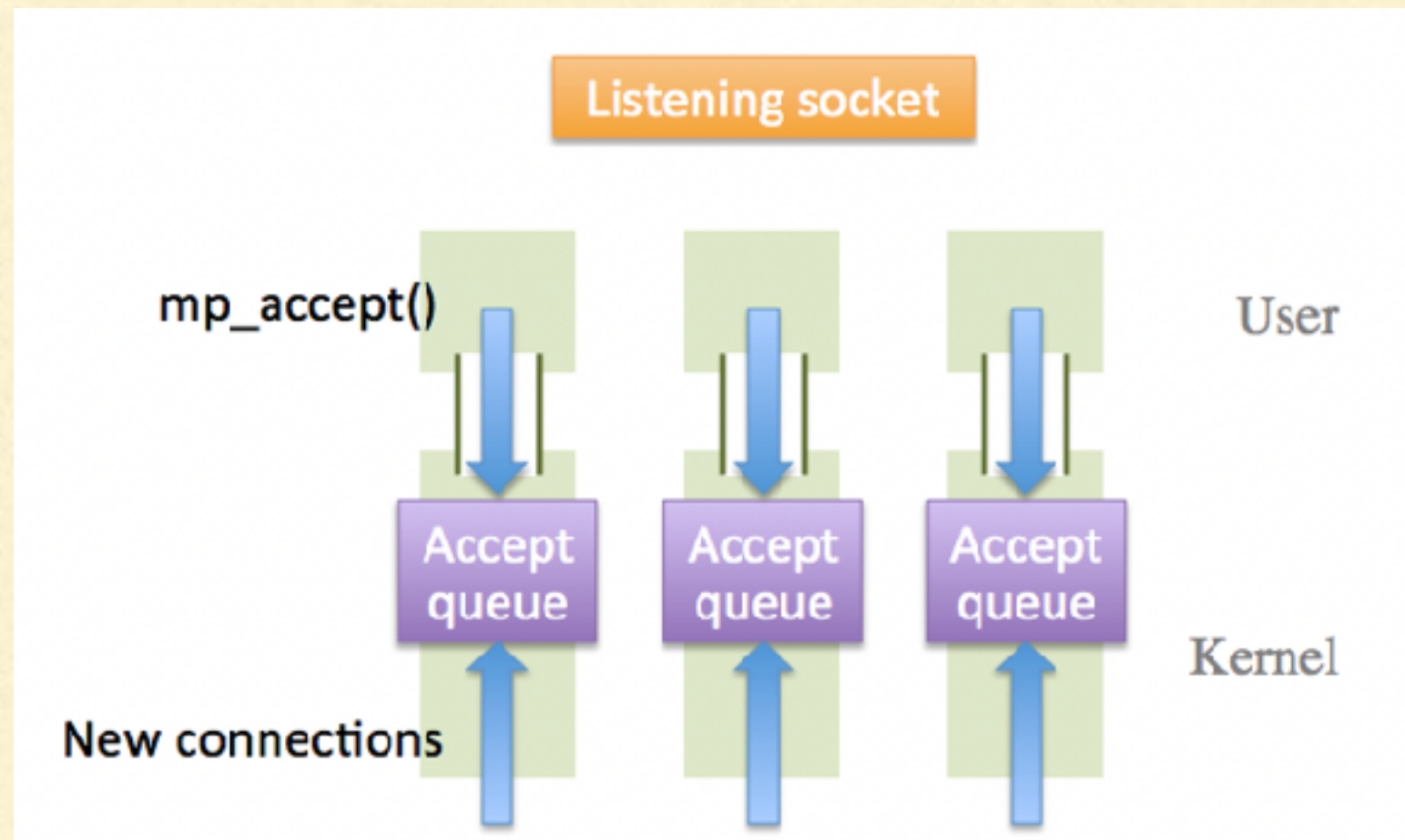


■ 到处都在排队



磁盘的util > 80%，有性能问题吗？

■ 到处都在排队



TCP建立连接涉及的两个队列:

- SYN_RECV状态的队列, `net.ipv4.tcp_max_syn_backlog`
- ESTABLISHED状态的队列, `min(net.core.somaxconn, backlog)`

当我们讨论排队时，我们都关心什么

- – 什么时候过去，排队的时间会比较短？
 - – 不排队的话，需要多少时间？
 - – 我前面一共还有几人？
 - – 排到我需要多少时间？
 - – 等我服务结束，又需要多少时间？
 - – 队伍长队达到什么程度，我就会放弃排队？
 - – 有多少个服务窗口？
-

当我们讨论排队时，我们都关心什么

- 想新开一家KFC，人流量是否够？需要多少收银人员？需要多少位置？
 - 银行营业厅内，先统一领号，再各窗口叫号，科学吗？
 - 应该部署几台服务器？异步线程池应该开多大？等待队列该怎么选？容量多少？
 - 如何解读命令iostat显示的磁盘性能指标？
-

优雅排队的艺术—QUEUEING THEORY

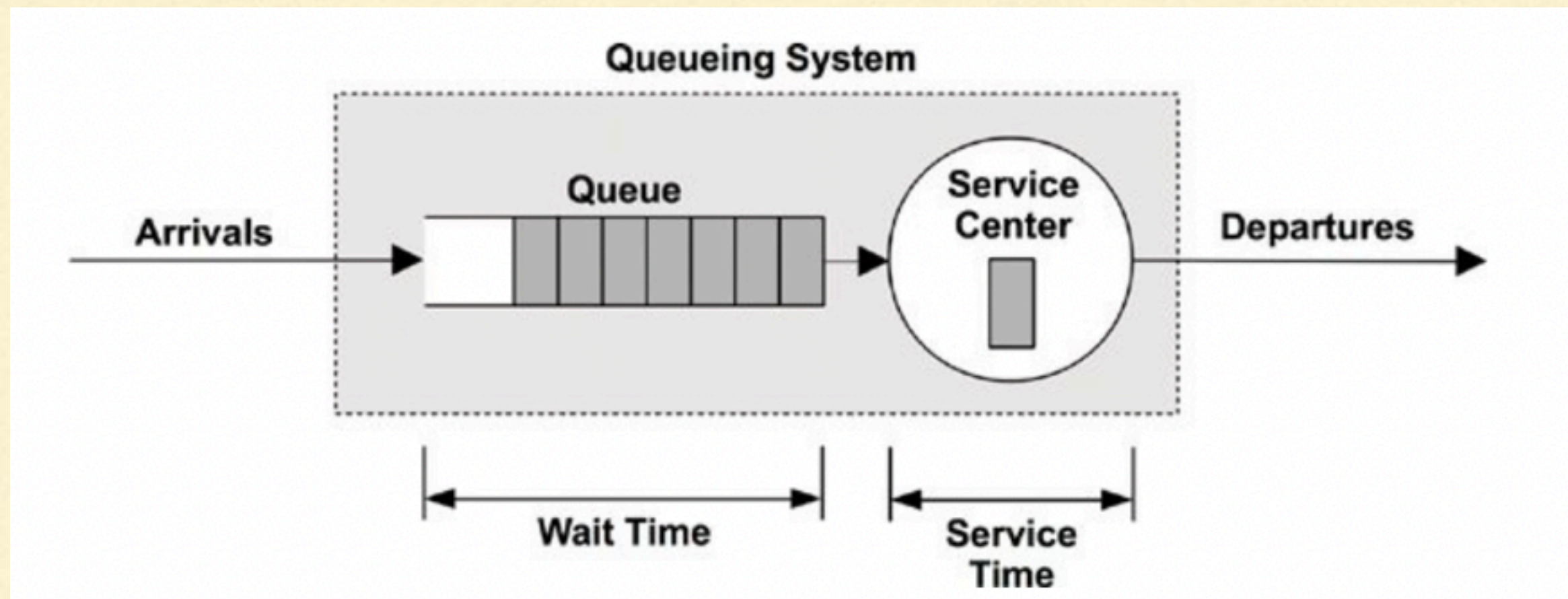
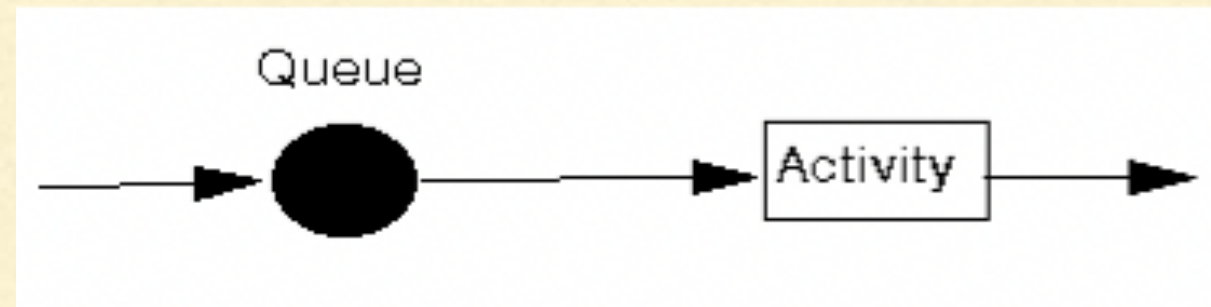
- 无论是现实中的排队，电话通讯领域的排队，还是计算机领域的排队，都能够抽象化为一个经典的理论——排队论。
 - 起源于20世纪初的电话通话。1909—1920年丹麦数学家、电气工程师 A.K.Erlang用概率论方法研究电话通话问题，从而开创了这门应用数学学科。
-

优雅排队的艺术—解决什么问题？

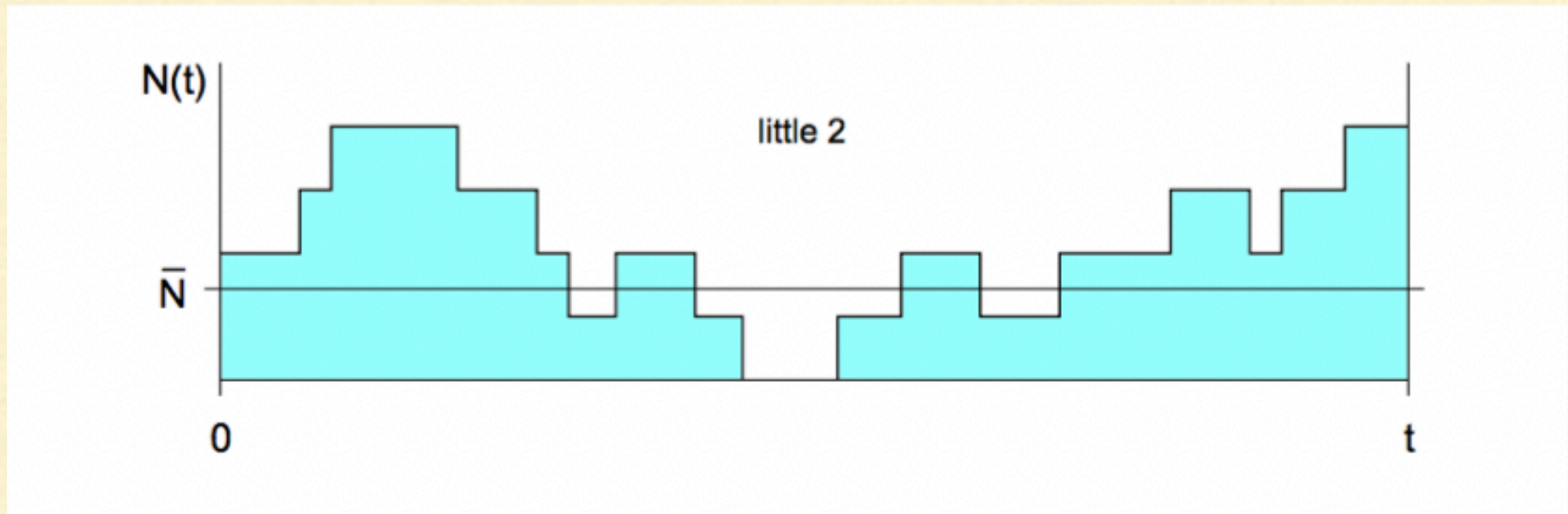
- 对用户来说: 响应时间 (满意度)
- 对服务提供者来说: 利用率 (成本)

在保障用户**满意度**的前提下，最大限度的**控制成本**， 充分挖掘系统的**潜力**。——**性能！**

优雅排队的艺术—抽象模型



优雅排队艺术—LITTLE'S LAW



- $N(t)$: t 时刻Queuing System内存在的请求数
- \bar{N} : Queuing System内在 t 时段的平均请求数
- T : 请求在Queuing System内的平均耗时
- λ : 单位时间进入Queuing System的请求数

优雅排队的艺术—LITTLE'S LAW

Little's Theorem: $N = \lambda * T$

存量数量 = 到达速率 * 平均响应时间

- 平均响应时间越长，队列长度也越长
 - 单位时间到达的请求越多，队列长度越长
-

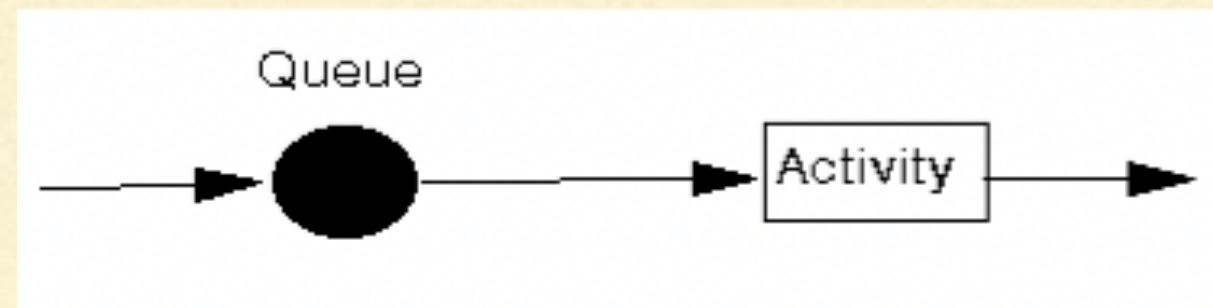
优雅排队的艺术—LITTLE'S LAW

例子：

一个小酒馆，客户平均访问频率为40人/小时；客户在小酒馆的平均消费时间是15分钟。那么，小酒馆的平均客户数量是多少？

解答： $40 \text{ 人/小时} * 0.25 \text{ 小时/人} = 10$

优雅排队艺术—行为规律



为什么会形成队列？

随机事件！

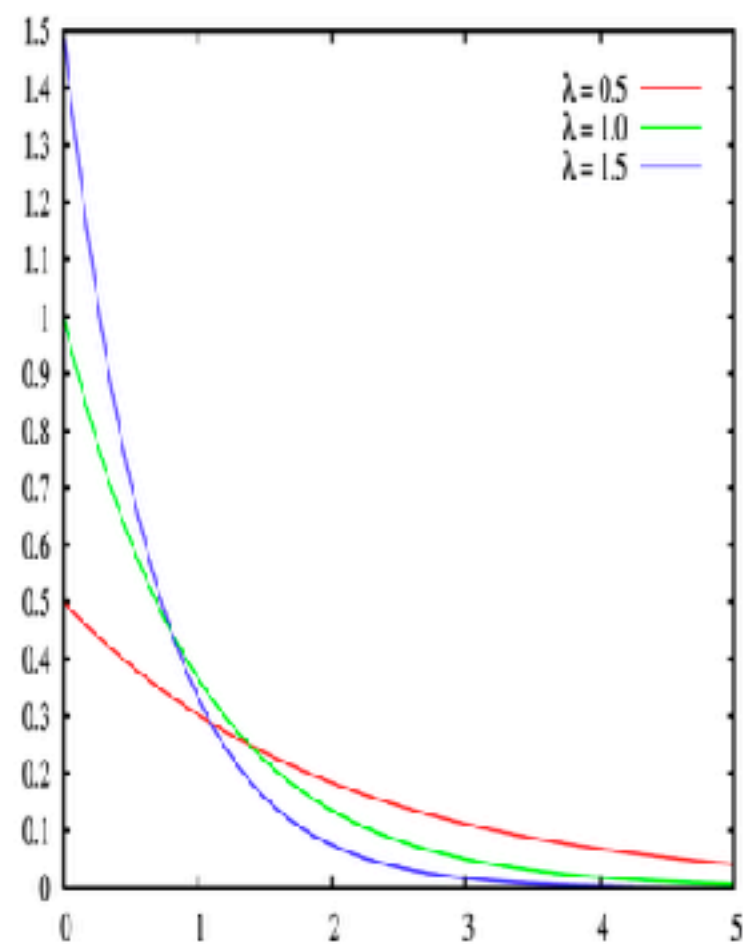
- 导致进队列的Arrival有什么规律？
- 导致离开系统的Service有什么规律？

优雅排队的艺术—行为规律

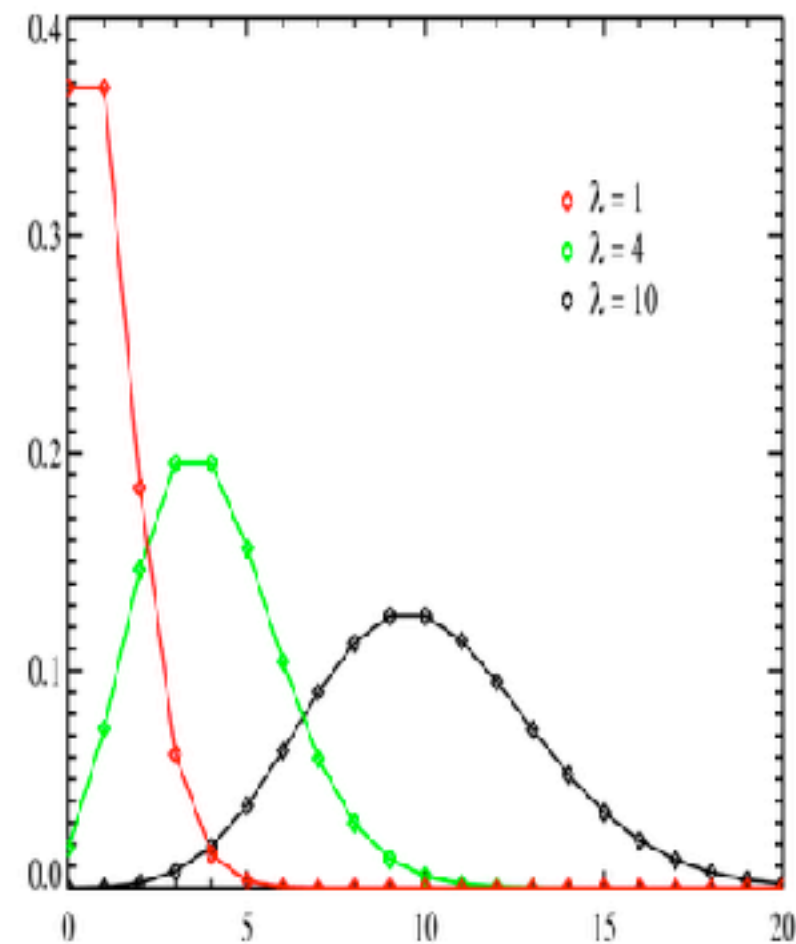
指数分布: 耗时与间隔, 如婴儿出生的时间间隔、来电的时间间隔

泊松分布: 事件发生, 如每小时出生3个婴儿、每10分钟接到1个电话

- 指数分布



泊松分布




优雅排队的艺术—行为规律

Arrival和Service均有可能遵循：

- M(Exponential/Markov):
 - 事件之间的时间间隔呈**指数分布**
 - 单位时间的事件次数呈**泊松分布**
 - D(Deterministic): 频率或间隔固定
 - G(General): 无规律
-

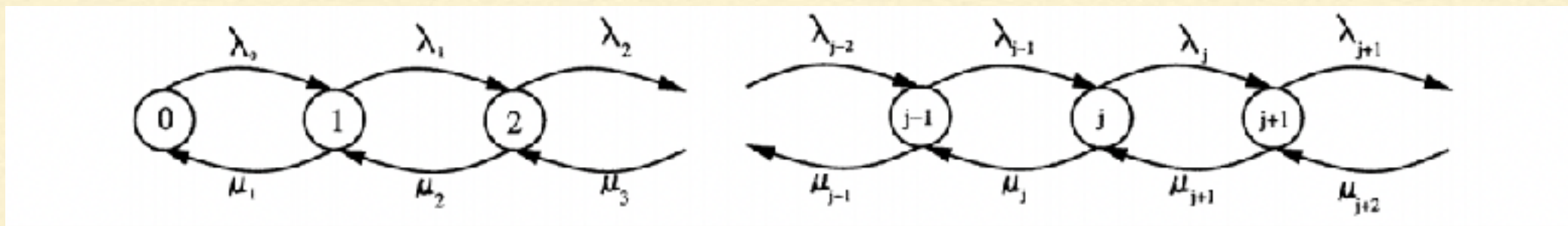
优雅排队的艺术—行为规律

两个关键行为Arrival和Service的规律 

系统的整体状态有什么规律 

优雅排队艺术——行为规律

用系统中存量请求的数量表示系统的状态 \mathbf{P} ，
每一个状态，只跟他前后两个状态有关。



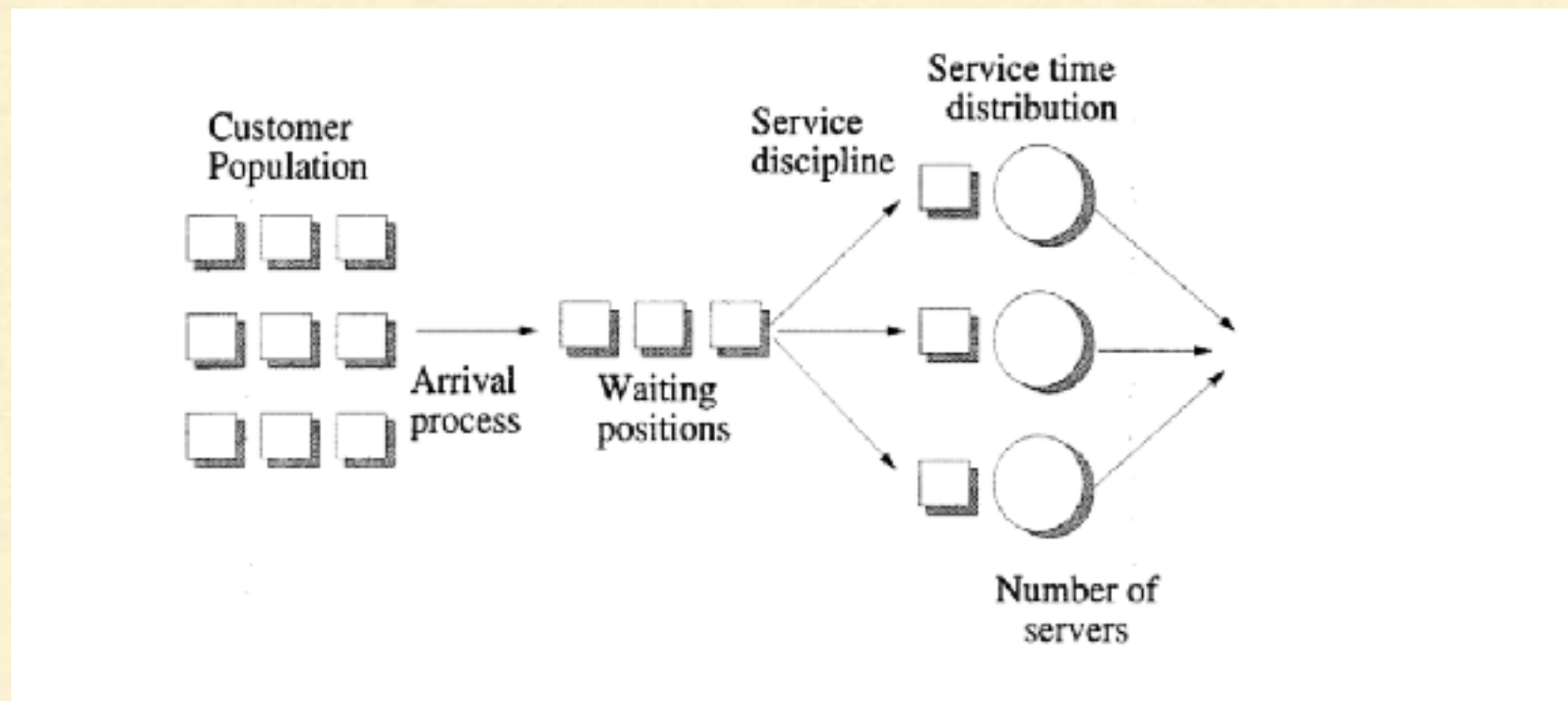
Markov链处于稳态时，节点的输入等于输出：

$$P_0 \lambda = P_1 \mu$$

$$P_1 (\lambda + \mu) = P_0 \lambda + P_2 \mu$$

$$P_j (\lambda + \mu) = P_{j-1} \lambda + P_{j+1} \mu$$

优雅排队的艺术—KENDALL NOTATION



Queuing System可描述为： $A/S/c$ ，称为Kendall Notation

A：请求到达过程，

S：服务过程，

c：表示服务承接者数量。

优雅排队的艺术—M/M/1模型

示例：M/M/1，最典型、最常见的模型

- 请求到达的间隔指数分布/单位时间到达数量泊松分布 (M)
- 服务时间指数分布 (M)
- 一个Server (1)

Kendall Notation的六元形式：A/S/c/K/N/D，额外包括：

- K: 队列最大长度，默认为无限
 - N: 请求总数量，默认为无限
 - D: 服务的调度策略，默认为先到先服务
-

优雅排队的艺术—M/M/1模型

- Server的利用率是多少？

$$U = \rho = \frac{\lambda}{\mu} (\lambda < \mu)$$

- 平均服务时间是多少？

$$S = \frac{1}{\mu}$$

- 服务队列长度是多少？

$$L_s = \lambda \cdot \frac{1}{\mu} = \frac{\lambda}{\mu}$$

- Server空闲概率是多少？

$$P_0 = 1 - U = 1 - \frac{\lambda}{\mu}$$

- 一个请求，需要等待的概率是多少？

$$P_L = 1 - P_0 = \frac{\lambda}{\mu}$$

已知：

- 单位时间到达的请求： λ
- 单位时间处理的请求： μ

- 总队列长度是多少？

$$L = \sum_{i=0}^{\infty} i P_i = \sum_{i=0}^{\infty} i (1 - \rho) \rho^i = \frac{\rho}{1 - \rho} = \frac{\lambda}{\mu - \lambda}$$

- 等待队列长度是多少？

$$L_q = \sum_{i=1}^{\infty} (i - 1) P_i = \frac{\rho^2}{1 - \rho}$$

- 平均响应时间是多少？

$$R = \frac{L}{\lambda} = \frac{1}{\mu} \frac{1}{1 - \rho} = \frac{1}{\mu - \lambda}$$

- 平均等待时间是多少？

$$W = \frac{L_q}{\lambda} = \frac{1}{\mu} \frac{\rho}{1 - \rho} = R - S$$

排队论的应用—M/M/1模型

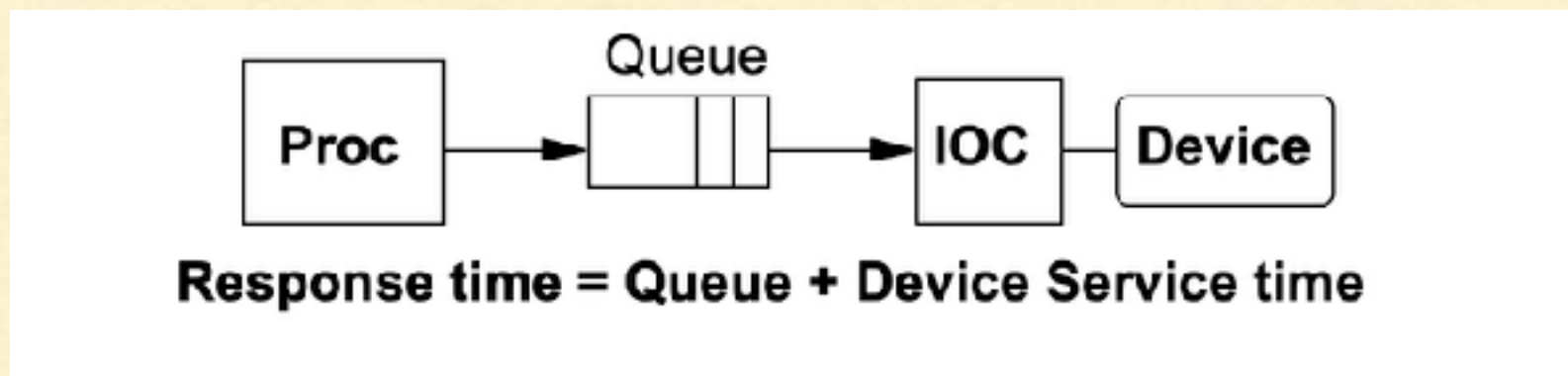
示例：M/M/1，最典型、最基础的模型

1.

```
// Creates an Executor that uses a single worker  
thread operating off an unbounded queue
```

```
final ExecutorService exService =  
Executors.newSingleThreadExecutor();
```

2. 磁盘IO



排队论的应用—M/M/1模型

计算器：

- [Queueing Theory Calculator](#)
- [Queuing Calculator](#)

排队论的应用—M/M/1模型

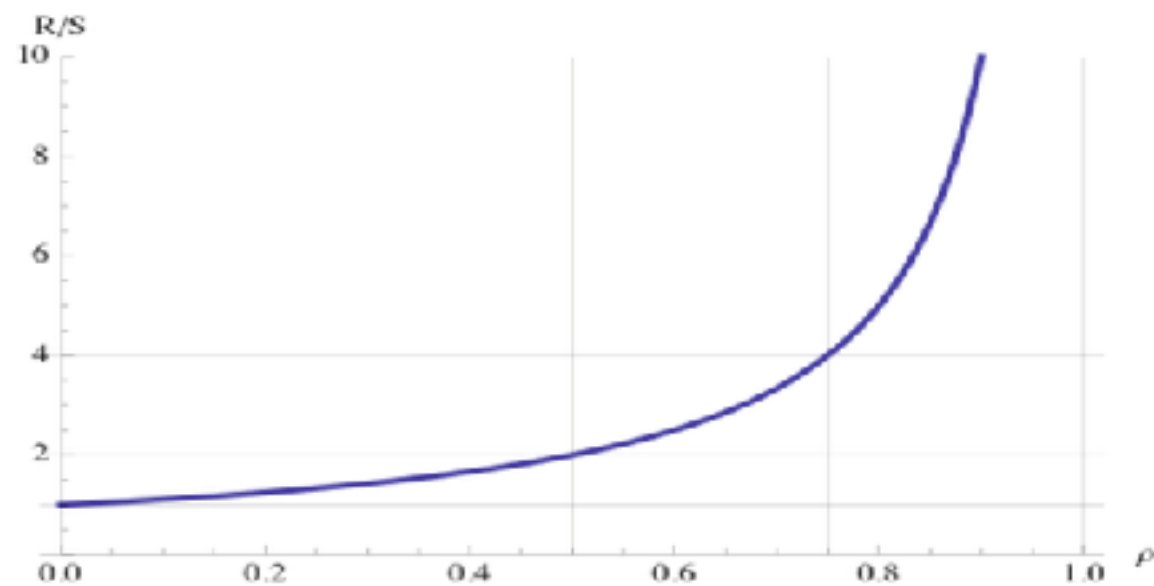
KFC窗口每小时有100人光顾，平均每人需要30秒完成购买并离开，问：

- 顾客需要在窗口前呆多久（排队+购买）？
 - 排队需要多久？
 - 窗口前会有多少顾客（排队+购买中）？
-

排队论的应用—M/M/1模型

利用率与响应时间之间的关系

$$R = \frac{1}{\mu} \frac{1}{1 - \rho} \quad (\text{其中: } \frac{1}{\mu} \text{ 为常数})$$



$$R = 1 / (\mu - \lambda)$$

I/O Requests/sec	Server Utilization	Mean response time (ms)
64	50%	15.6
72	56%	17.8
80	62%	20.7
88	69%	24.9
96	75%	31.1
104	81%	41.3
112	87%	61.7
120	94%	121.9
124	97%	237.8

排队论的应用—M/M/1/k模型

M/M/1/k模型

- 请求到达的间隔指数分布/单位时间到达数量泊松分布 (M)
 - 服务时间指数分布 (M)
 - 一个Server (1)
 - ~~无限等待队列长度~~ 等待队列长度为k, 超过则拒绝
 - 无限请求数量 (默认)
 - 先到先服务的调度策略 (默认)
-

排队论的应用—M/M/1/k模型

现实意义

- 大部分现实中的系统，都是有最大等待队列限制的；
- 哪怕是没有限制，用户发现队列过长时，也会主动放弃；

关键问题

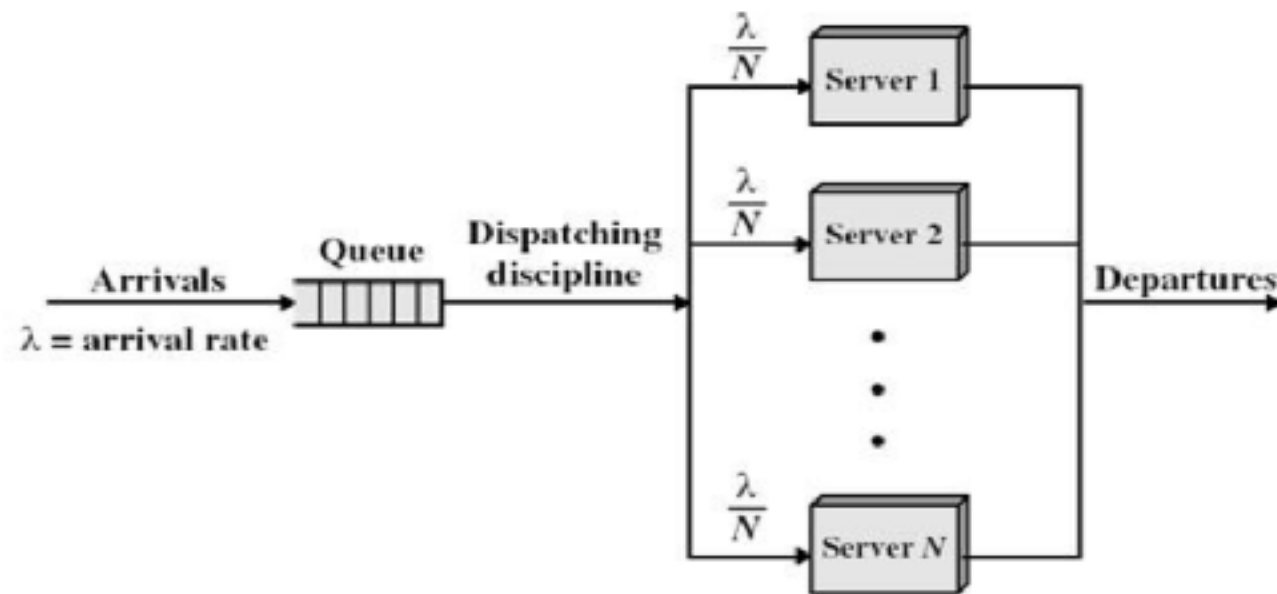
- 请求到达时，发现队列已满，而被丢弃的概率是多少？
- 要求丢弃概念小于一定值时，队列容量该如何设置？

对于长为n的队列，丢弃的概率P为：

$$\rho^n * (1-\rho) / (1-\rho^{(n+1)})$$

排队论的应用—M/M/m模型

m个处理能力相同的server
一个排队队列(无限)



排队论的应用—M/M/m模型

使用率 $\rho = \lambda / m\mu$ 响应时间 $R = 1/\mu * 1/(1-\rho^m)$

现实意义：

- 从安全和性能出发，一般会有多个服务节点

关键问题：

- 要多少个服务节点，才能不至于因等待太久而放弃？
-

排队论的应用—M/M/m vs M/M/1



排队论的应用一对比总结

场景：

Kafka可视为队列，某topic的消费者每秒种只能处理1个消息，无法满足业务需求。

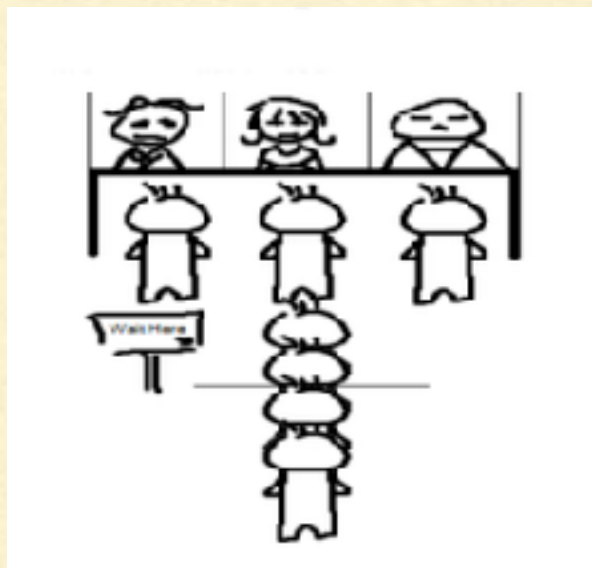
问题：

以下三种应对方案，哪个更“经济”？

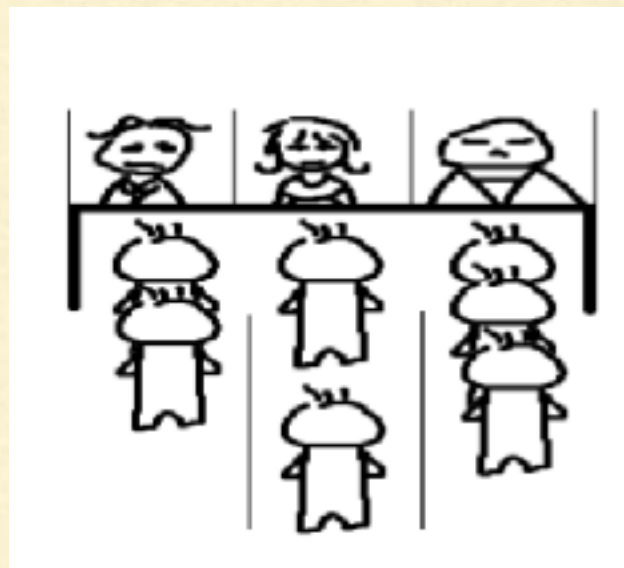
- 优化处理逻辑，使速度提升一倍
 - 添加一个相同的消费者(添加了partition)
 - 拆分成两个topic，带两相同消费者
-

排队论的应用—对比总结

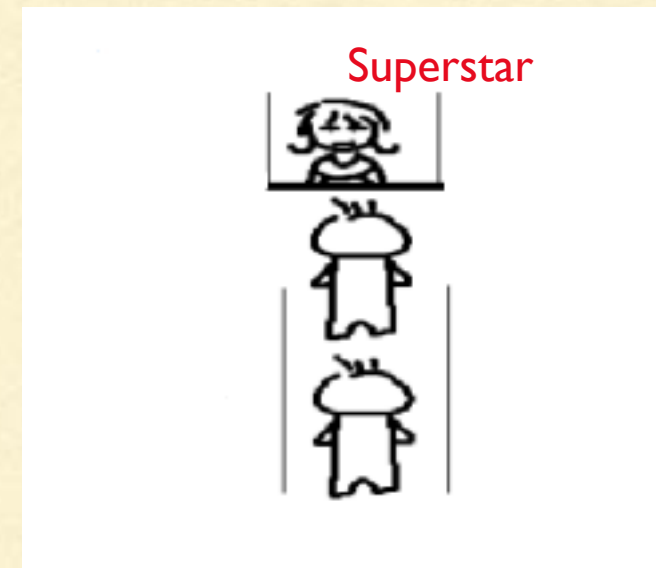
- $M/M/m$: m 个Server, 1个队列, 平均处理速度 μ , 总速度 $m\mu$
- $m M/M/1$: m 个Server, m 个队列, 平均处理速度 μ , 总速度 $m\mu$
- $M/M/1$: 1个Server, 1个队列, 平均处理速度 $m\mu$, 总速度 $m\mu$



优衣库



KFC



?

排队论的应用一对比总结

- M/M/m: m个Server, 1个队列, 平均处理速度 μ , 总速度 $m\mu$
$$R = 1/\mu * 1 / (1-\rho^m)$$
 - m M/M/1: m个Server, m个队列, 平均处理速度 μ , 总速度 $m\mu$
$$R = 1/\mu * 1 / (1-\rho)$$
 - M/M/1: 1个Server, 1个队列, 平均处理速度 $m\mu$, 总速度 $m\mu$
$$R = 1/m\mu * 1 / (1-\rho)$$
-

排队论的应用一对比总结

结论：

- $M/M/1(m\mu) > M/M/m > m M/M/1$

启发：

- 一个效率10的程序员 > 10个效率1的程序员
 - 软硬件的性能，先追求单机极致，再是多机
-

参考

- <https://cs.gmu.edu/~menasce/cs700/files/PerformanceModeling1.pdf>
 - <http://bnrg.eecs.berkeley.edu/~randy/Courses/CS252.S96/Lecture21.pdf>
 - <https://www.win.tue.nl/~iadan/queueing.pdf>
 - <https://goldenratiphi.wordpress.com/2013/05/06/comparison-between-single-and-multiple-queues/>
 - <https://www.csus.edu/indiv/b/blakeh/mgmt/documents/opm101supplc.pdf>
 - <http://williams.comp.ncat.edu/comp755/Q.pdf>
 - <http://web.engr.illinois.edu/~dmnicol/ece541/slides/queueing.pdf>
 - <http://www.supositorio.com/rcalc/rcalclite.htm>
 - <http://people.brunel.ac.uk/~mastjjb/jeb/or/queue.html>
 - <http://www.netlab.tkk.fi/opetus/s383143/kalvot/english.shtm>
 - <http://ocw.nctu.edu.tw/upload/classbfs121001554684839.pdf>
 - <http://www.ruanyifeng.com/blog/2015/06/poisson-distribution.html>
 - <http://web2.uwindsor.ca/math/hlynka/qsoft.html>
 - <https://www.stat.auckland.ac.nz/~stats255/qsim/qsim.html>
 - <https://pan.baidu.com/share/link?uk=4265849107&shareid=4118923932>
-