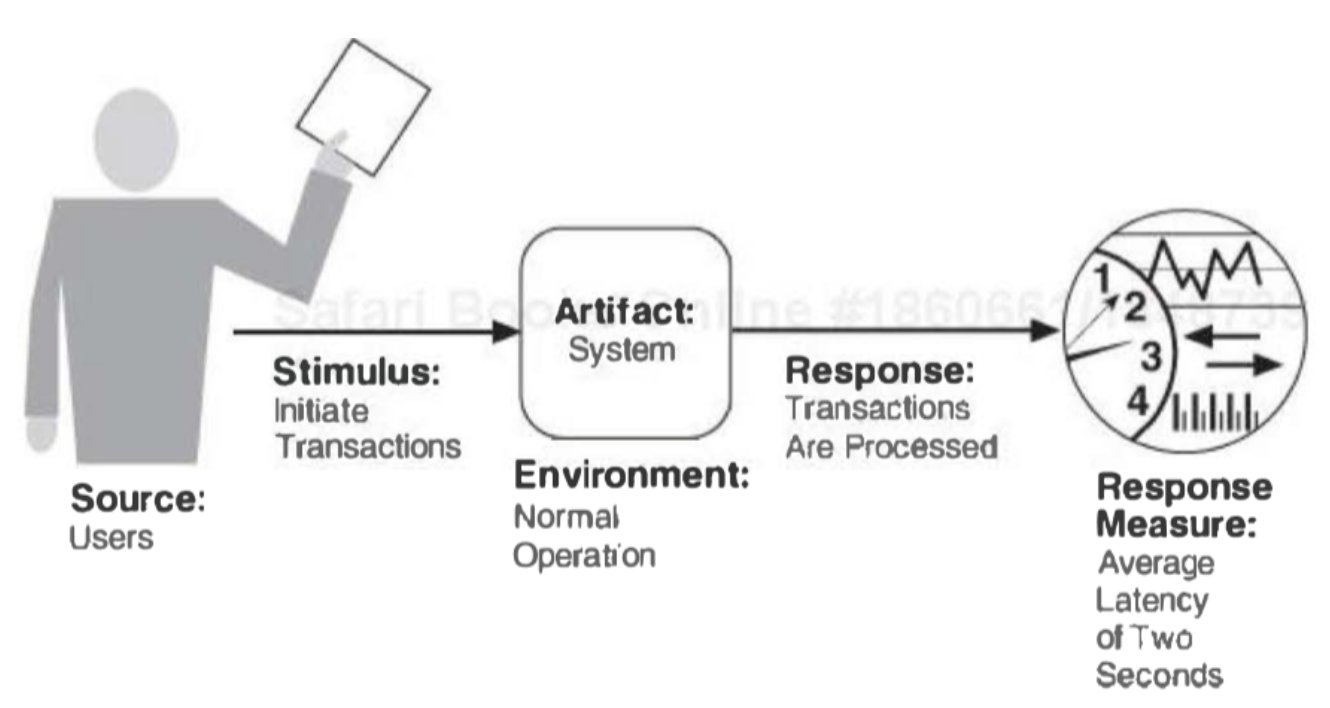
# 读书笔记：Software Architecture in Practice (1) Ch8. Performance

201900302022 计机19.1 姜九鸣

**软件性能(****Performance)是什么？**

一句话来说，就是它与时间和软件系统满足时序要求的能力有关。当事件发生时——中断、消息、来自用户或其他系统的请求，或标记时间流逝的时钟事件——系统或系统的某些元素必须及时响应它们。描述可能发生的事件（以及何时发生）以及系统或元素对这些事件的基于时间的响应是讨论性能的本质。



在软件工程的大部分历史中，性能一直是系统架构的驱动因素。因此，它经常损害所有其他品质的实现。随着硬件价格/性能比的持续下降和软件开发成本的不断上升，其他品质已成为性能的重要竞争对手。性能通常与可扩展性相关联，即增加系统的工作容量，同时仍能保持良好的性能。从技术上讲，可扩展性使您的系统易于以一种特殊的方式进行更改，一种可修改性也是如此。

**设计一个高性能软件，需要考虑其并发性（Concurrency）。**

* 并发是架构师必须理解的更重要的概念之一，也是计算机科学课程中教授最少的概念之一。并发性是指并行发生的操作。例如下面的代码：

 x := 1;  
 ×++;

* 如果多个线程并发地执行上述代码，就会出现并发的问题。如图。

**顺序执行和并发执行**

* 顺序执行：一个程序，其各个步骤之间的偏序关系是在程序设计时就确定了的。
* 并发执行：但是，如果这个程序有几个线程，而偏序关系又是跨线程的，那么谁也不能保证这个偏序关系会不会因为并发而被打乱。因为线程之间的调度顺序由操作系统的调度程序决定。
  + 虽然调度顺序由OS决定，但是我们可以控制它虽然被调度，但是不开始执行啊。
  + 因此，实际上，执行顺序仍由应用程序自行协商决定。

**临界资源**

* **临界资源是要求互斥访问的资源**。
* 例如打印机，注册表，文件。

**临界区**

* 程序的一个代码段。特点是，在该段内，该程序**访问了临界资源**。
* **为了保证**临界资源的互斥访问，要求对同一临界资源，同一时间**只能有一个**进程在其对应的临界区中。

**伯恩斯坦条件**

* 读写、写写操作不能并发访问同一资源，否则会出问题。

**Race condition**

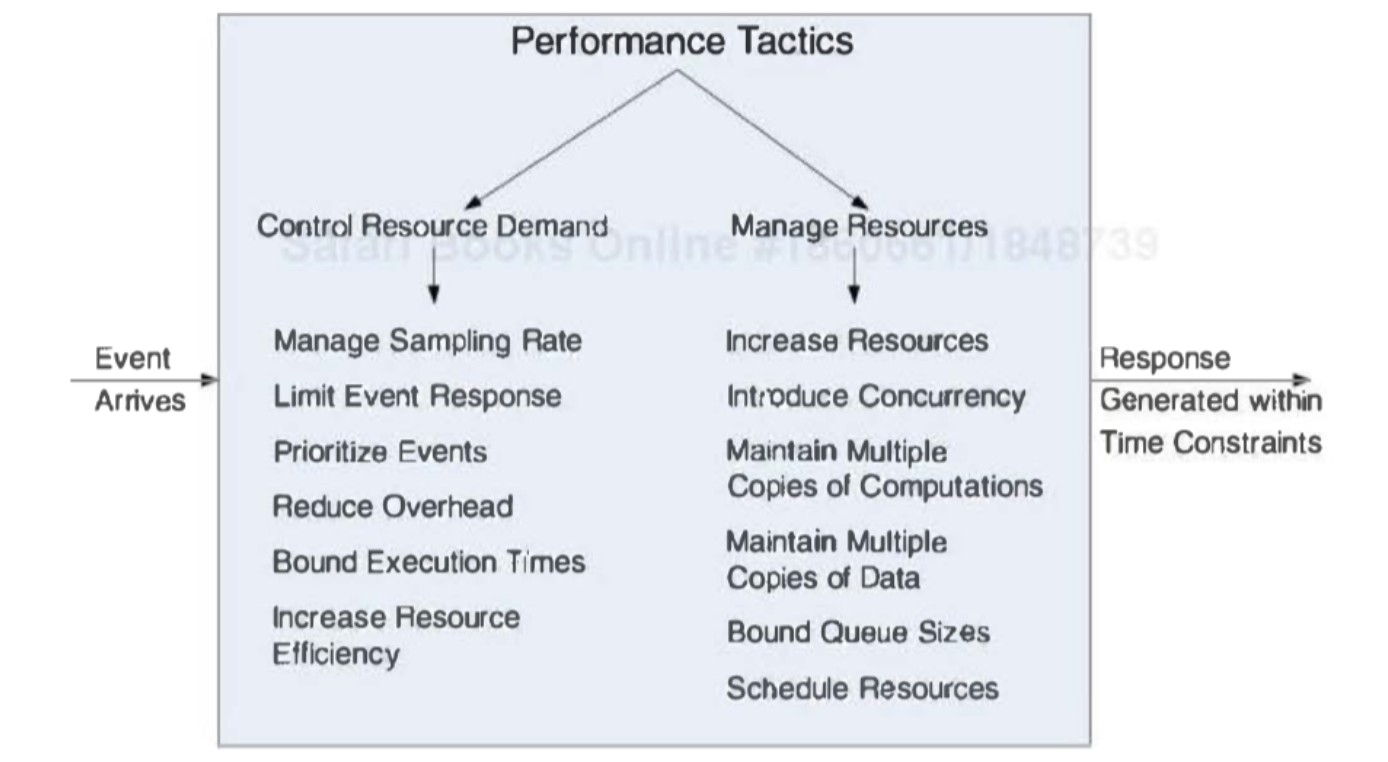
**Race condition**: The situation where several processes access，and manipulate shared data concurrently. The final value of the shared data depends upon which process finishes last.

* Race condition是一个状况，并不应该像书上那样翻译成“条件”。其同义词应该是situation才对。
* Race condition就是指多个进程并发**访问和操作**同一个共享数据的情况。该数据最终取值取决于**最后一个**完成访问和操作的进程。
* 如何解决Race condition？**——引入进程同步的概念。**

**原子操作和原语**

* 原子操作在核心态下实现，并常驻内存。
* 原语是指一段程序。这段程序实现的功能是一个原子操作。

**资源调度**



* 为了实现多线程并发执行一段代码，而且还不至于让彼此混乱，一定需要保证对临界资源的互斥访问。

**临界区算法需要满足的四个条件**

* 互斥（防止出错） Mutual Exclusion
  + 例如两边都绿灯
* 前进（防止饥饿） Progress
  + 如果临界区已经空闲，那么必须有进程能进来。
  + 即：有空让进
  + 例如红绿灯都是红的
* 有限等待（防止饥饿） Bounded Waiting
  + 在进程正在等待临界区空闲时，应该限制其让步的次数。这样防止它一直在让步，永远进不去。
  + 例如一边长期红，一边长期绿
* 让权等待（提升效率）
  + 当进程等待临界区空闲时，由于其一直死循环，而且什么也不做，只是占用CPU，显然是资源的浪费。
  + 如果等待的时候可以阻塞，就会进一步提升CPU效率。