实战Java高并发程序设计第1周

DATAGURU专业数据分析社区

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

内容提要

- 课程基础
- 为什么需要并行
- 几个重要的概念
- 2个重要的定理

课程基础

- 需要有Java的使用经验

- 为什么需要并行？
  - 业务要求
  - 性能

为什么需要并行

- 反对意见
  - Linus Torvalds ：忘掉那该死的并行吧！
  - 需要有多么奇葩的想象力才能想象出并行计算的用武之地？

为什么需要并行

- Linus Torvalds炮轰过的技术
  - GNU Emacs
  - GNOME
  - HFS+ (Mac OS 文件系统)
  - Java
    - “本质上我看到的只是 Java 引擎在走下坡路，因为它别无去处。 ” 1998年8月
    - “我不关心Java。多么可怕的语言。 “2011年11月
  - C++
    - “事实是，C++编译器不值得信任。整个C++异常处理从根本上是错误的。 “2004年1月19日 “
    - 尽管 C++ 可以用于原型或简单的 GUI 编程，但它不能使事情更简单。C 语言虽然并不精益于系统编程语言，但它积极鼓励你使用简单和直接的结构。 “2007年9月7日
    - “C ++ 是一个可怕的语言。” 2007年9月6日
  - XML
  - Solaris
  - MINIX

为什么需要并行

- Linus Torvalds ：并行计算只有在图像处理和服务端编程2个领域可以使用，并且它在这2个领域确实有着大量广泛的使用。但是在其它任何地方，并行计算毫无建树！

为什么需要并行

- 摩尔定律的失效
  - 预计18个月会将芯片的性能提高一倍
  - Intel CEO Barret单膝下跪对取消4GHz感到抱歉
    - 在2004年秋季，Intel宣布彻底取消4GHz计划
  - 虽然现在已经有了4GHZ的芯片，但频率极限已经逼近

10年过去了，我们还停留在4GHZ

为什么需要并行

- 顶级计算机科学家唐纳德·尔文·克努斯
  - **在我看来，这种现象(并发)或多或少是由于硬件设计者**
  - **已经无计可施了导致的，他们将摩尔定律失效的责任**
  - **推脱给软件开发者。**

为什么需要并行

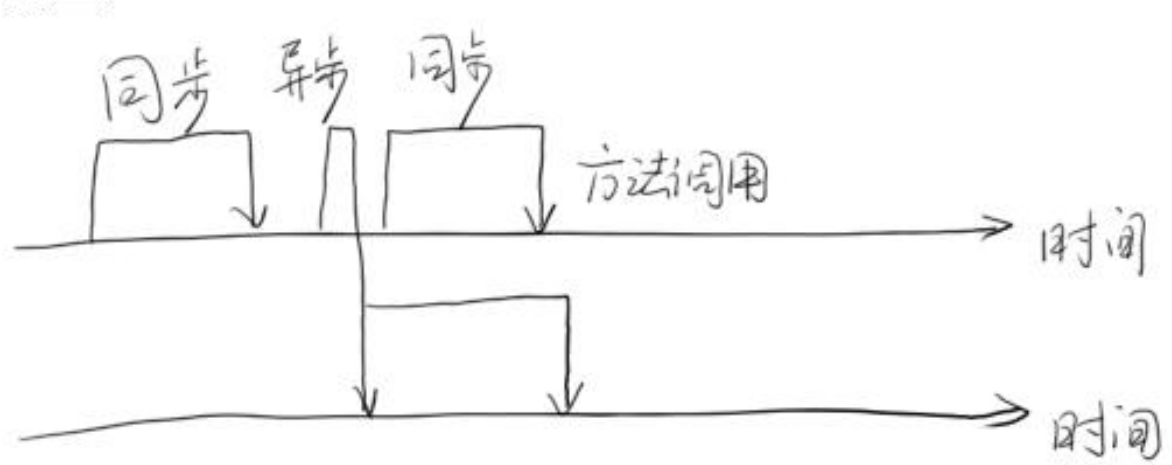
- 并行计算还出于业务模型的需要
  - 并不是为了提高系统性能，而是确实在业务上需要多个执行单元。
  - 比如HTTP服务器，为每一个Socket连接新建一个处理线程
  - 让不同线程承担不同的业务工作
  - 简化任务调度

几个重要的概念

- 同步（synchronous）和异步（asynchronous）
- 并发（Concurrency）和并行（Parallelism）
- 临界区
- 阻塞（Blocking）和非阻塞（Non-Blocking）
- 锁（Deadlock）、饥饿（Starvation）和活锁（Livelock）
- 并行的级别

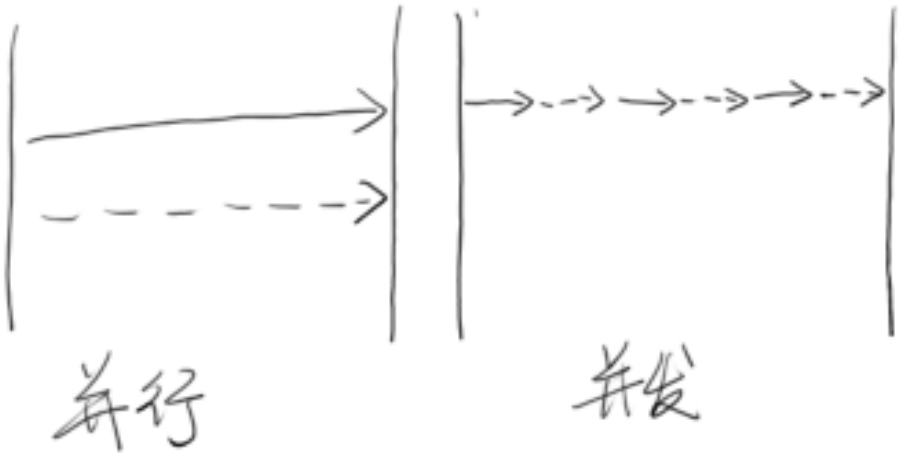
几个重要的概念

- 同步（synchronous）和异步（asynchronous）



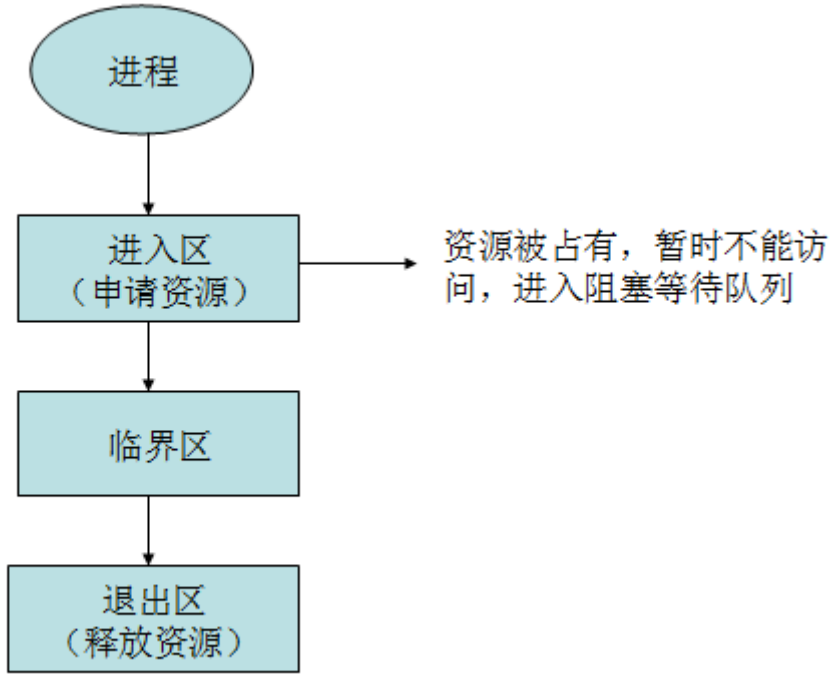
几个重要的概念

- 并发（Concurrency）和并行（Parallelism）



几个重要的概念

- 临界区
  - 临界区用来表示一种公共资源或者说是共享数据，可以被多个线程使用。但是每一次，只能有一个线程使用它，一旦临界区资源被占用，其他线程要想使用这个资源，就必须等待。

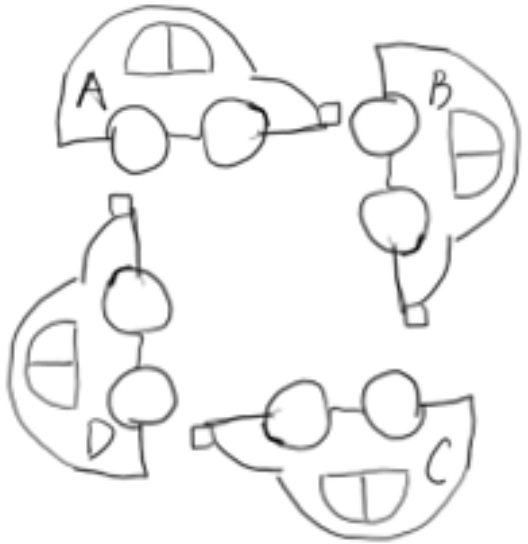


几个重要的概念

- 阻塞（Blocking）和非阻塞（Non-Blocking）
  - 阻塞和非阻塞通常用来形容多线程间的相互影响。比如一个线程占用了临界区资源，那么其它所有需要这个资源的线程就必须在这个临界区中进行等待，等待会导致线程挂起。这种情况就是阻塞。此时，如果占用资源的线程一直不愿意释放资源，那么其它所有阻塞在这个临界区上的线程都不能工作。
  - 非阻塞允许多个线程同时进入临界区

几个重要的概念

- 死锁（Deadlock）、饥饿（Starvation）和活锁（Livelock）



饥饿是指某一个或者多个线程因为种种原因无法获得所需要的资源，导致一直无法执行。

电梯遇人

几个重要的概念

- 并发级别
  - 阻塞
  - 无障碍
  - 无锁
  - 无等待

  无障碍、无锁、无等待：非阻塞

几个重要的概念

- 阻塞
  - 当一个线程进入临界区后，其他线程必须等待

几个重要的概念

- 无障碍（Obstruction-Free）
  - 无障碍是一种最弱的非阻塞调度
  - 自由出入临界区
  - 无竞争时，有限步内完成操作
  - 有竞争时，回滚数据

几个重要的概念

- 无锁（Lock-Free）
  - 是无障碍的
  - 保证有一个线程可以胜出

```
while (!atomicVar.compareAndSet(localVar, localVar+1))
{
      localVar = atomicVar.get();
}
```

几个重要的概念

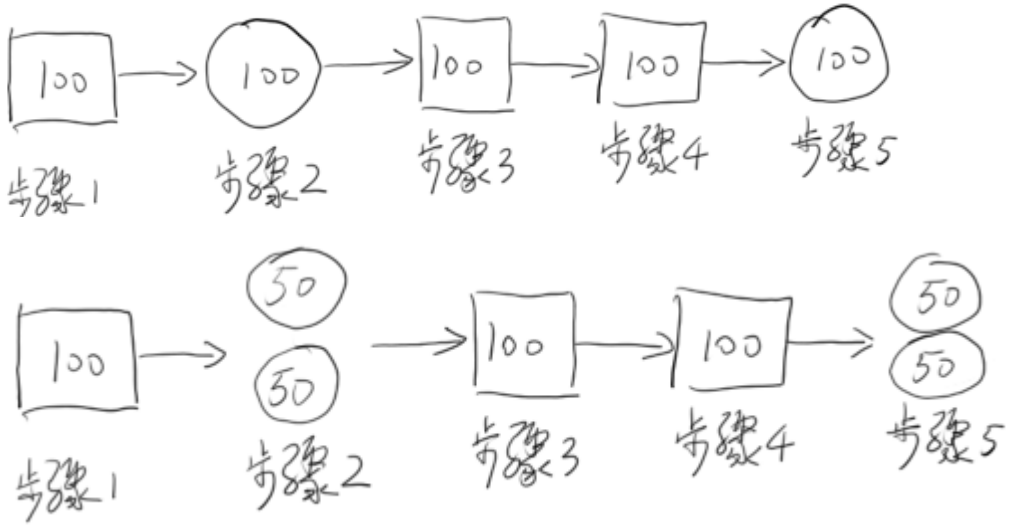
- 无等待（Wait-Free）
  - 无锁的
  - 要求所有的线程都必须在有限步内完成
  - 无饥饿的

有关并行的2个重要定律

- Amdahl定律（阿姆达尔定律）
- Gustafson定律（古斯塔夫森）

有关并行的2个重要定律

- Amdahl定律（阿姆达尔定律）
  - 定义了串行系统并行化后的加速比的计算公式和理论上限
  - **加速比定义：加速比=优化前系统耗时/优化后系统耗时**



加速比=优化前系统耗时/优化后系统耗时=500/400=1.25

$$T_n = T_1\left(F + \frac{1}{n}(1-F)\right)$$

串行比例　并行比例　代入

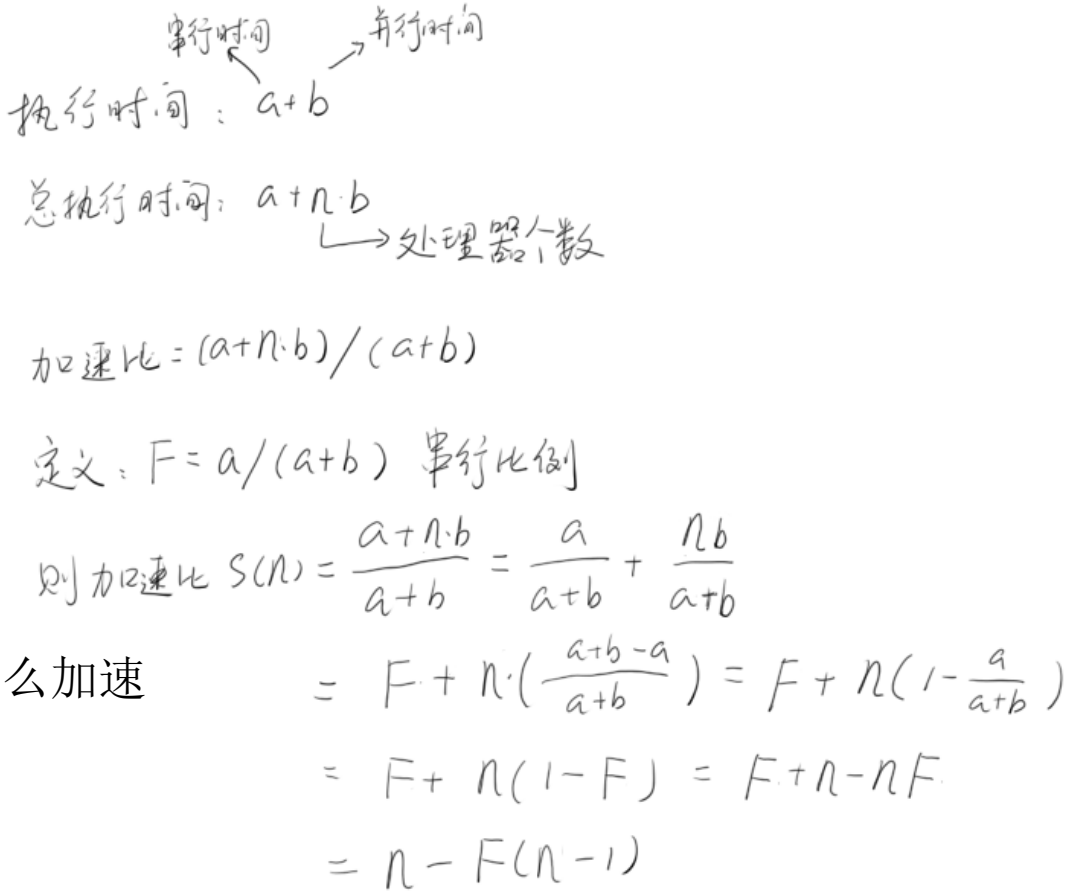
处理器个数

$$\text{加速比} = \frac{T_1 \rightarrow \text{优化前耗时}}{T_n \rightarrow \text{优化后耗时}} = \frac{T_1}{T_1\left(F + \frac{1}{n}(1-F)\right)} = \frac{1}{F + \frac{1}{n}(1-F)}$$

增加CPU处理器的数量并不一定能起到有效的作用
提高系统内可并行化的模块比重，合理增加并行处理器数量，才能以最小的投入，得到最大的加速比

有关并行的2个重要定律

- Gustafson定律（古斯塔夫森）
  - 说明处理器个数，串行比例和加速比之间的关系



只要有足够的并行化，那么加速比和CPU个数成正比

Thanks

FAQ时间