# Concurrency Programming

[toc]

## 并发编程的挑战

```
并发编程的三个大挑战：
    1. 上下文切换
    2. 死锁
    3. 资源限制
```

### 查看Java程序运行的方法（jstack）

```
    jstack pid > ~/Documents/dump.txt
    //获取java程序运行过程中的日志

    vim ~/Documents/dump.txt
    //查看具体的运行参数，了解运行状态
```

### 上下文切换

```
并发能够提升CPU的使用效率，同时进行上下文切换有线程创建和上下文切换的成本。如果过于频繁，可能会导致性能下降。
```

#### 减少上下文切换的方法

1. 无锁的并发编程。
2. 使用CAS算法。
3. 减少使用线程。
4. 协程代替多线程（单线程里实现多任务的调度，在单线程李维持多个任务的切换）

### 死锁

#### 死锁的介绍

##### 死锁的条件

1. 互斥条件
2. 请求与保持
3. 不可剥夺条件
4. 循环等待

#### 预防死锁的方法

1. 一次性分配所有资源
2. 只要有一个资源无法获取，就不分配其他资源
3. 可以剥夺其他资源
4. 资源有序分配

#### 解决死锁的方法

1. 以确定的顺序获取锁
2. 超时放弃

### 死锁常见的预防方法

1. 避免一个线程同时获取多个锁
2. 避免一个锁内占用多个资源，尽量保证每个锁内之占用一个资源
3. 尝试使用定时锁，使用lock.tryLock(timeout)来替代内部锁机制
4. 对于数据库锁最好放在一个数据库连接内，否则会出解锁失败的情况

## 资源限制的挑战

```
硬件资源限制：网速、磁盘的读写速度、cpu的执行速度。
软件资源限制：数据库的连接数量、socket连接数。
```

- 问题：

卡在资源瓶颈，如果增加线程数，会造成上下文切换、资源调度的成本，降低性能。

- 解决方法：

1. 软件资源限制：通过池化的思想，创建连接池，减少创建连接和断开连接的成本。

2. 硬件资源限制：可以考虑使用集群并行执行程序。