Java 并发编程

为什么很重要?

并发编程可以充分利用计算机的资源，把计算机的性能发挥到最大，可以最大程度节约公司的成本，提高效率。

1、什么是高并发

并发 VS 并行的区别

> 并发 concurrency：多线程“同时”操作同一个资源，并不是真正的同时操作，而是交替操作，单核 CPU 的情况下，资源按时间段分配给多个线程。张三李四王五使用一口锅炒菜，交替

> 并行 parallelism：是真正的多个线程同时执行，多核 CPU，每个线程使用一个 CPU 资源来运行。张三李四王五使用三口锅炒菜，同时进行

并发编程描述的是一种使系统允许多个任务可以在重叠的时间段内执行的设计结构，不是指多个任务在同一时间段内执行，而是指系统具备处理多个任务在同一时间段内同时执行的能力。

高并发是指我们设计的程序，可以支持海量任务的执行在时间段上重叠的情况。

高并发的标准：

- QPS：每秒响应的 HTTP 请求数量，QPS 不是并发数。
- 吞吐量：单位时间内处理的请求数，由 QPS 和并发数来决定。
- 平均响应时间：系统对一个请求作出响应的评价时间。

QPS = 并发数 / 平均响应时间

- 并发用户数：同时承载正常使用系统的用户人数

互联网分布式架构设计，提高系统并发能力的方式：

- 垂直扩展
- 水平扩展

垂直扩展

提升单机处理能力

1、提升单机的硬件设备，增加 CPU 核数，升级网卡，硬盘扩容，升级内存。

2、提升单机的架构性能，使用 Cache 提高效率，使用异步请求来增加单服务吞吐量，NoSQL 提升数据库访问能力。

水平扩展

集群：一个厨师搞不定，多雇几个厨师一起炒菜，多个人干同一件事情。

分布式：给厨师雇两个助手，一个负责洗菜，一个负责切菜，厨师只负责炒菜，一件事情拆分成多个步骤，由不同的人去完成。

站点层扩展：Nginx 反向代理，一个 Tomcat 跑不动，那就 10 个 Tomcat 去跑。

服务层扩展：RPC 框架实现远程调用，Spring Boot/Spring Cloud，Dubbo，分布式架构，将业务逻辑拆分到不同的 RPC Client，各自完成对应的业务，如果某项业务并发量很大，增加新的 RPC Client，就能扩展服务层的性能，做到理论上的无限高并发。

数据层扩展：在数据量很大的情况下，将原来的一台数据库服务器，拆分成多台，以达到扩充系统性能的目的，主从复制，读写分离，分表分库。

2、JUC

JDK 提供的一个工具包，专门用来帮助开发者完成 Java 并发编程。

3、进程和线程

Java 默认的线程数 2 个

- mian 主线程
- GC 垃圾回收机制

Java 本身是无法开启线程的，Java 无法操作硬件，只能通过调用本地方法，C++ 编写的动态函数库。

> Java 中实现多线程有几种方式?

1、继承 Thread 类

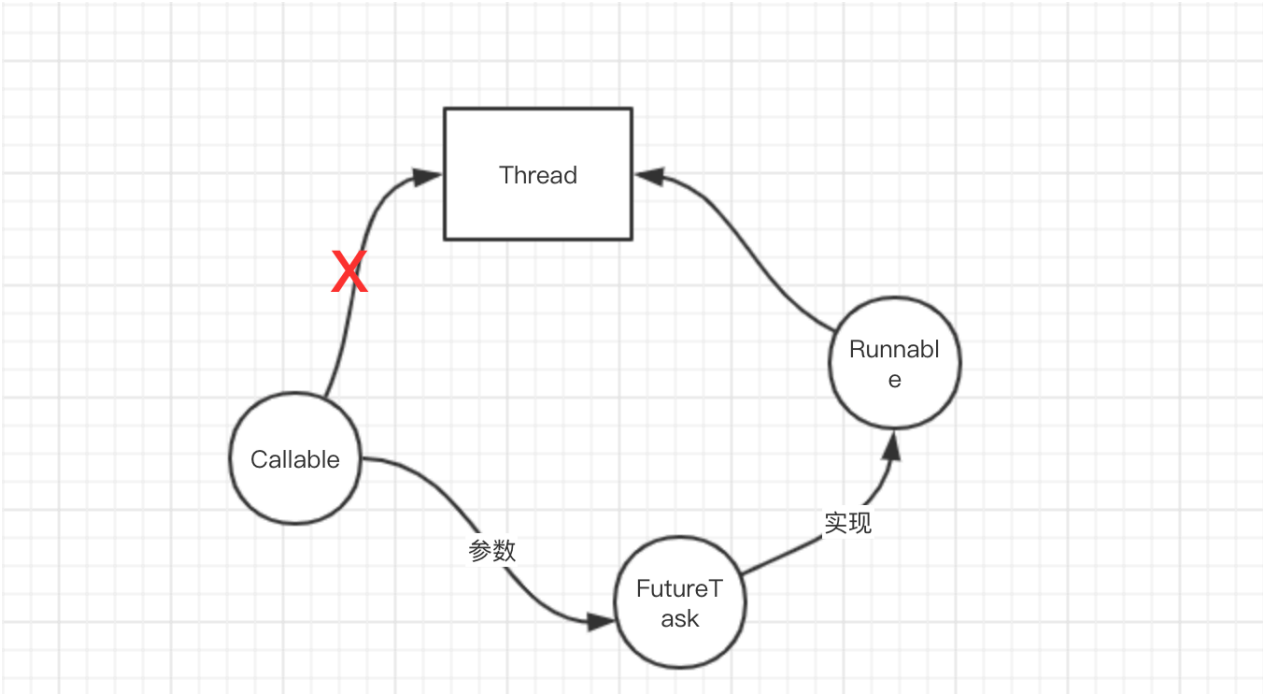
2、实现 Runnable 接口

3、实现 Callable 接口

Callable 和 Runnable 的区别在于 Runnable 的 run 方法没有返回值，Callable 的 call 方法有返回值。

```
package com.southwind.demo1;

import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.FutureTask;

public class Test {
    public static void main(String[] args) {
        MyCallable myCallable = new MyCallable();
        FutureTask<String> futureTask = new FutureTask(myCallable);
        Thread thread = new Thread(futureTask);
        thread.start();
```

```
        try {
            String value = futureTask.get();
            System.out.println(value);
        } catch (InterruptedException e) {
            e.printStackTrace();
        } catch (ExecutionException e) {
            e.printStackTrace();
        }
    }
}

class MyCallable implements Callable<String>{

    @Override
    public String call() throws Exception {
        System.out.println("callable");
        return "hello";
    }
}
```



4、sleep 和 wait

sleep 是让当前线程休眠，wait 是让访问当前对象的线程休眠。

sleep 不会释放锁，wait 会释放锁。

5、synchronized 锁定的是什么

1、synchronized 修饰非静态方法，锁定方法的调用者

2、synchronized 修饰静态方法，锁定的是类

3、synchronized 静态方法和实例方法同时存在，静态方法锁定的是类，实例方法锁定的是对象