### 多线程的优缺点

**什么是线程和进程？？**

进程是cpu资源分配的最小单位（是能拥有资源和独立运行的最小单位）

线程是cpu独立运行和独立调度的最小单位（线程是建立在进程的基础上的一次程序运行单位，一个进程中可以有多个线程）

**首先说下多线程出现的原因:**

为了解决负载均衡问题,充分利用CPU资源.

为了提高CPU的使用率,采用多线程的方式去同时完成几件事情而不互相干扰.为了处理大量的IO操作时或处理的情况需要花费大量的时间等等,比如:读写文件,视频图像的采集,处理,显示,保存等

**多线程的优点:**

1. 使用线程把耗费时间长的程序中的任务放到后台去处理，多个线程交替执行，减少或避免因程序阻塞或意外情况造成的响应过慢现象，降低了用户等待的概率。
2. 使多CPU系统更加有效。操作系统会保证当线程数不大于CPU数目时，不同的线程运行于不同的CPU上

3.线程间方便的通信机制，由于同一进程下的线程之间共享数据空间，所以一个线程的数据可以直接为其它线程所用，这不仅快捷，而且方便；

**多线程的缺点:**

1.如果有大量的线程,会影响性能,因为操作系统需要在它们之间切换.

2.更多的线程需要更多的内存空间

3.一个线程的崩溃可能影响到整个程序的稳定性

4.通常数据是在多个线程间共享的,需要复杂的同步机制，避免死锁等等。

5.大量的线程降低代码的可读性；

**多进程优点：**

每个进程互相独立，不影响主程序的稳定性，子进程崩溃没关系；

通过增加CPU，就可以容易扩充性能；

可以尽量减少线程加锁/解锁的影响，极大提高性能，就算是线程运行的模块算法效率低也没关系；

每个子进程都有2GB地址空间和相关资源，总体能够达到的性能上限非常大

**多进程缺点：**

多进程调度开销比较大，操作系统调度切换多个线程要比切换调度进程在速度上快的多。

逻辑控制复杂，需要和主程序交互；

线程之间由于共享进程内存空间，所以交换数据非常方便；进程间内存无法共享，通讯需要跨进程边界，比较麻烦。

在创建或撤消进程时，由于系统都要为之分配和回收资源，导致系统的开销明显大于创建或撤消线程时的开销。

**线程相对于进程的优点：**

1、开销小

2、资源共享性好。

**线程相对于进程的缺点：**

1、共享资源需要耗费一定的锁资源，同步相对复杂。

2、一个线程崩溃可能导致整个进程崩溃，这个当然是自己的应用程序有问题

**进程和线程的区别体现在以下几个方面：**

1.地址空间和其他资源（如打开文件）：进程间相互独立，同一进程的各线程间共享。某进程内的线程在其他进程内不可见。

2.通信：进程间通信IPC（管道，信号量，共享内存，消息队列），线程间可以直接独写进程数据段（如全局变量）来进程通信——需要进程同步和互斥手段的辅助，以保证数据的一致性。

3.调度和切换：线程上下文切换比进程上下文切换快得多。

**进程之间的通信**

管道：速度慢，容量有限，只有父子进程能通讯

FIFO：任何进程间都能通讯，但速度慢

消息队列：容量受到系统限制，注意第一次读的时候，要考虑上一次没有读完数据的问题

信号量：不能传递复杂消息，只能用来同步

共享内存区：能够很容易控制容量，速度快，但要保持同步，比如一个进程在写的时候，另一个进程要注意读写的问题，相当于线程中的线程安全，当然，共享内存区同样可以用作线程间通讯，不过没这个必要，线程间本来就已经共享了同一进程内的一块内存。

**浏览器每一个tab页面即为进程，页面间的通信即为进程的通信**

WebWorker，JS的多线程？

前文中有提到JS引擎是单线程的，而且JS执行时间过长会阻塞页面，那么JS就真的对cpu密集型计算无能为力么？

所以，后来HTML5中支持了Web Worker。

MDN的官方解释是：

Web Worker为Web内容在后台线程中运行脚本提供了一种简单的方法。线程可以执行任务而不干扰用户界面

一个worker是使用一个构造函数创建的一个对象(e.g. Worker()) 运行一个命名的JavaScript文件

这个文件包含将在工作线程中运行的代码; workers 运行在另一个全局上下文中,不同于当前的window

因此，使用 window快捷方式获取当前全局的范围 (而不是self) 在一个 Worker 内将返回错误

这样理解下：

创建Worker时，JS引擎向浏览器申请开一个子线程（子线程是浏览器开的，完全受主线程控制，而且不能操作DOM）

JS引擎线程与worker线程间通过特定的方式通信（postMessage API，需要通过序列化对象来与线程交互特定的数据）

所以，如果有非常耗时的工作，请单独开一个Worker线程，这样里面不管如何翻天覆地都不会影响JS引擎主线程，

只待计算出结果后，将结果通信给主线程即可，perfect!

而且注意下，JS引擎是单线程的，这一点的本质仍然未改变，Worker可以理解是浏览器给JS引擎开的外挂，专门用来解决那些大量计算问题。

其它，关于Worker的详解就不是本文的范畴了，因此不再赘述。

WebWorker与SharedWorker

既然都到了这里，就再提一下SharedWorker（避免后续将这两个概念搞混）

WebWorker只属于某个页面，不会和其他页面的Render进程（浏览器内核进程）共享

所以Chrome在Render进程中（每一个Tab页就是一个render进程）创建一个新的线程来运行Worker中的JavaScript程序。

SharedWorker是浏览器所有页面共享的，不能采用与Worker同样的方式实现，因为它不隶属于某个Render进程，可以为多个Render进程共享使用

所以Chrome浏览器为SharedWorker单独创建一个进程来运行JavaScript程序，在浏览器中每个相同的JavaScript只存在一个SharedWorker进程，不管它被创建多少次。

看到这里，应该就很容易明白了，本质上就是进程和线程的区别。

**SharedWorker由独立的进程管理，WebWorker只是属于render进程下的一个线程**

**浅谈两种方法实现浏览器内多个标签页之间的通信**

方法一：

localstorge在一个标签页里被添加、修改或删除时，都会触发一个storage事件，通过在另一个标签页里监听storage事件，即可得到localstorge存储的值，实现不同标签页之间的通信。

方法二：

使用cookie+setInterval，将要传递的信息存储在cookie中，每隔一定时间读取cookie信息，即可随时获取要传递的信息。