### 当遇到系统性能问题时，可以从如下方面进行思考

1. 性能问题是怎么触发的。

首先要分析性能问题出现的时间及情况，问题一直存在或者最近的某一个时间点才出现。

如果问题最近才出现，则应该搞清楚当时用户执行了什么操作，或者系统进行了哪些变更，或者是不是有数据量的突然增大。

1. 系统的性能瓶颈在哪里。

将该功能的实现划分为一个个小步骤。找到耗时最多和异常的步骤。着力分析这些可以的地方。这就缩小了优化的范围，有利于优化的顺利开展。

1. 找到优化的目标。

对优化的结果应该有一个目标：比如缩短响应时间，缩短Job运行时间等等。记录当前运行时间，优化后如果时间缩短了就达到了优化的目标。在某些情况下，用户会有特殊要求，比如要求响应时间在优化后不多于多少秒，或者要求Job在某一时刻之前运行完成。这些优化的目标会让我们更快找到相应的优化办法。

### 具体优化方法

1. **Java代码优化**

Java代码优化的核心是：保留有用的操作和数据，去掉没用的操作和数据。传说中，后羿射掉了九个多余的太阳。而在程序里，所有没有必要的操作和数据都是那些多余的太阳。比如某一变量，如果不一定被使用，就不要初始化。只有该变量一定被后面的程序所使用时，才有必要初始化。比如，MVC框架中，有些

Java程序中容易出现一下问题：

1. 循环太多，包括for循环和while循环。听起来道理很简单，但是这种问题却经常出现。因为很多时候，程序员为了代码复用，会放弃代码的性能。这种做法提高了开发效率，但牺牲了性能。
2. 耗时操作太多，例如查询数据库的次数太多。更有甚者，在循环里面使用进行耗时操作。这会让我们的系统变得非常用户不友好。笔者在优化系统时，曾经把SQL的log全部打印下来，然后对log进行分析，发现很多雷同的SQL。这些SQL往往结构相同，只是参数不同。如果仔细分析每个SQL的功能，我们可以找到合并SQL的方法。
3. 没有使用懒加载。同事曾经做过这样优化：将一些耗时的变量初始化操作从构造方法里面移除，加到set方法里面。这个优化让系统的性能提升了一倍。其原理是：如果耗时操作放在构造方法里，每次类初始化的时候，所有的操作都会执行。如果放在set方法里面，耗时操作只有在被需要的时候经过set方法执行。这就实现了懒加载。在一些框架中，也可以配置懒加载操作。

### 性能优化的原则

1. **不影响系统本来的功能是前提**

笔者曾经为了优化一个功能模块而打算减少不将某些看似不太重要的数据持久化到数据库。后来这种做法遭到了老板的反对。因为这些数据虽然不是最关键的，但可以用来让用户检查历史记录。这个案例说明性能优化应该以不影响系统本来的功能是前提。

1. **尽量使用简单有效的方法**

提高用户体验可以从多种角度出发，有各种不同的办法。找到简单有效的办法是优化的最佳实践。纵观互联网技术发展史，那些复杂的技术往往被简单易懂的技术所替代。用简单的方法做好事情，既省人力，又让系统容易维护。笔者曾经优化一个Java系统。系统打包成Jar包，运行在服务器上。而Jar包的运行需要6个小时。笔者当时为了减少运行时间相处了很多种方式，例如多线程，优化SQL，JDBC Batch。最后提交的办法是准备三个相同的jar包，给他们加上不同的参数。让三个jar同时运行，但处理不同的业务内容。这种方法让整个运行时间减小为3小时。