# 系统性能优化

### 王操 2020/4/25

### 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。

1. 能不能从业务入手

一个系统可能有很多功能，有些功能用户用的不多，但挺耗费性能。如果砍掉这些功能，或许大幅度提升性能。因此，从业务入手是一个好办法。

### 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。比如用StringBuilder替代String避免产生多余的对象。比如，MVC框架中，有些model在内存中重复，去掉重复的部分将节约很多空间。比如将一个大表拆分为一个个的小表，这样加载表的时候就只需加载小表。

以下是一些Java程序中最容易出现的问题：

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。
4. 并行

几年前，笔者认为并行跟并发是同一概念。后来听到一节课中说：并行是同一时刻多个事件一起运行，而并发是一个时间间隔内多个事件一起运行。有“并发量”之说，却无“并行量”之说。如何生动地理解并行？笔者一边烧开水，一边煮米饭，一边炒菜，这就是并行。著名的“煎饼问题”用到的就是一种并行的思维。

笔者认为，并行是一种很重要的优化思路。多件事情一起做，这里的“多件事情”的粒度控制很关键。它有可能是不同的计算，有可能是不同的数据库查询，也有可能指的是不同的运行的jar包。“多线程”和“多进程”就是因为事件粒度的大小不同而叫法不同。

1. **优化具体的耗时操作**

耗时操作包括数据库操作，复杂计算，网络传输等等。

数据库操作的优化牵涉到缓存的应用，数据库和表的设计，分库分表，SQL优化等。SQL优化是一门复杂的学问，笔者认为优化SQL的难度大于优化Java代码。首先SQL是对多张表的操作。对于多表数据的操作蕴含着丰富的数学知识。SQL中有些复杂的函数，不常用的话，大家容易陌生。另外，SQL中无法加入断点。很多情况下，对SQL的运行只能进行推理，而难以验证。比较有效的方法是将SQL划分为一个个的子SQL。关于SQL优化的教程有很多。如何有效地创建和使用索引，如何根据解释计划进行优化是SQL优化中重要的方面。

缓存是计算机和软件学科中非常体现智慧的应用。无论是在硬件还是在软件中，缓存得到了广泛的应用。缓存的原理是时间局部性和空间局部性。时间局部性是指，如果访问了某一个数据，则很有可能在附近的时间节点再次访问该数据。空间局部性是指，如果访问了某 一个数据，则很有可能也访问空间上附近的数据。

在系统中，Redis缓存能加快对热点数据的访问。在内存里存放一些数据，以在将来的时间点被系统使用也是一种缓存的方式。但数据不能被持久化。

数据库分库分表的设计和读写分离，跟分布式架构的原理类似。

如果有复杂的计算，选择好的数据结构和算法也很重要。这也牵涉到数据库和表结构的设计。异步是一种较好的处理方式，因为用户不用等待整个计算过程结束。但是有时候，用户需要看到一个计算过程结束的状态集合，程序就不能用异步方法了。

网络传输也常常是系统的性能瓶颈。笔者没有优化系统网络传输的经历，却在使用各种应用时经常有网速好网速坏的不同体验。几个月前，笔者在家办公老是连不上公司的内网。后来因为疫情，公司的远程系统加大了带宽，笔者终于可以连内网了。笔者在做区块链项目时曾有这样的场景：Linux shell中用git去clone Github的代码时老是time out，后来笔者clone阿里云码云仓库里面相同的代码终于成功。本地主机连接国内的网站在网络传输距离上较近，所以下载速度快不会timeout。这种镜像的原理有点类似于CDS（Content Delivery Service）。总之，对于网络传输的优化，笔者有这么几个认识。一是网络传输的协议栈，我们很难去更改。但我们可以选择不同的协议。二是考虑尽可能减小网络传输的数据量。三是加大带宽，四是减小网络传输的距离。五是选择传递的方式，如每次传递的量。需要根据具体的业务来选择不同的方案。

1. **改良架构**

目前流行的分布式架构通过将系统部署在不同的服务器上满足了高并发的要求。淘宝等电商通过分布式系统来支撑庞大的用户量。另一个比较火的技术是微服务架构。这种架构将系统分为不同的功能模块，不同的模块之间独立开发独立部署，实现了系统的解耦。但是笔者认为，并不是所有的系统都需要用分布式架构，也并不是所有的系统都适合使用微服务架构。但从性能上面讲，分布式的架构，本质上加大了数据传输的成本。这是与性能提升的要求相悖的。但是如果数据量足够大，没有分布式的支撑，服务器和数据库是会崩溃的。优秀的架构设计者，会根据业务要求去甄选合理的架构，以满足多方面的要求。

1. **提升硬件性能**

硬件工程师可以做出好的硬件给系统提升性能，而软件工程师则只能去选择好的硬件来为我所用。但是好硬件配置往往要更多的开销，所以需要在开销和性能之间进行权衡。另外，这种权衡不光是开销和性能上的。比如SSD，虽然性能很高，读写很快，但出于安全性的考虑，很多公司并没有使用SSD。

### 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。