# 系统性能问题的分析

随着用户对速度方面的用户体验的要求的提高，性能优化在系统开发中变得越来越重要。系统的速度的提升将带来好的用户体验，也将带来更高的用户粘性，用户忠诚度和用户转化率。性能优化不能盲目进行，需要在优化前找到重点，优化后监测和总结。本文只探讨用户提出系统性能问题之后如何进行分析，并不讨论具体的技术方面的优化办法。

近段时间，用户对我们的系统提出了几个性能问题。应该从如下方面进行分析。

弄清楚性能问题出现的时间和场景。

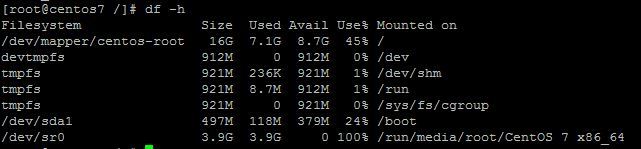
功能模块可能在第一次上线的时候就有性能问题，也有可能在经过多次修改后上线才出现性能问题。另外，性能问题的出现往往伴随着并发量的提升，数据量的增加等等场景。找到性能问题出现的时间和场景能让我们更块地定位问题地原因。

## 采集和分析性能问题出现时服务器的状态

系统出现性能问题时，服务器或者数据库往往处于不正常的状态。应该去重现该性能问题，并实时采集服务器或者数据库的各种状态。主要有网络通信，磁盘空间，内存使用，CPU使用等。

网络通信: 使用ping或者telnet指令判断网络是否正常。

磁盘空间: df -h命令来查看磁盘信息， -h 选项为根据大小适当显示。当Use%达到或者接近100%就应该引起注意了，这表明性能问题很可能是磁盘空间问题导致的。应该清理磁盘空间并且进一步监测磁盘空间使用率变化趋势。



内存使用：可以用Linux的free指令获取系统的内存剩余量，内存总量。也可以用java代码打印虚拟机堆的内存使用量。Runtime.maxMemory()这个方法返回的是java虚拟机（这个进程）能构从操作系统那里挖到的最大的内存。Runtime.totalMemory()这个方法返回的是java虚拟机现在已经从操作系统那里挖过来的内存大小，也就是java虚拟机这个进程当时所占用的所有内存。Runtime.freeMemory()指的是java虚拟机从操作系统那里挖过来的但还未使用的内存。此外，jdk自带的一些监控工具，例如jvisualvm，jmap也能查看内存使用状态。

CPU 运行等待的数量: 运行uptime 命令即可获得当前时间之前1 秒、5秒和15 秒的CPU运行等待数量的平均值。

磁盘IO 量：该指标体现的是磁盘的IO繁忙程度，可以通过vmstat指令获得。数据库的增删查改就是典型的IO操作。Web应用往往是IO密集型操作，大部分情况下，CPU在等待IO操作，IO的负载大小很大程度影响到整个系统的性能。

测量和分析时间主要花在哪个步骤

测量时间有多种方法。例如，可以运行程序，并用计时器记录运行的时间。也可以在程序中加上的时间戳。时间戳既可以加载前端脚本中，也可以加在在后端语言中。例如，为实现文件导入功能的java代码中的时间戳：

public void importFiles()

{

long start = System.currentTimeMillis();

//这里是一段耗时操作

long end = System.currentTimeMillis();

System.out.println(“importFiles() executed in ” + (end - start)/1000 +”s” );

}

用户提出的性能问题往往是业务层面的。从技术上分析实现该业务模块需要哪些步骤，并找到耗时最多的步骤。

|  |  |  |  |
| --- | --- | --- | --- |
| 时间(ms) | 文件 | 行数 | 时间(ms)/行数 |
| 350000 | customer.txt | 1361490 | 0.257 |
| 4240000 | user.txt | 2618800 | 1.619 |
| 280000 | order.txt | 177320 | 1.579 |
| 40000 | request.txt | 20280 | 1.972 |

做这些分析和测试的原因是为了找到那些最值得优化的步骤。根据阿姆达尔定律，优化系统中耗时比率在5%以下的模块，不可能给系统带来5%以上的性能提升。因此，不应该花时间去优化那些非关键的环节。例如，假设我们通过测试和分析得出如下四个模块的开销：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模块 | A | B | C | D |
| 耗时占比 | 60% | 10% | 2% | 28% |

如果将C的性能提升一倍，整个系统的性能也只提高了1%。因此，如果花在非关键环节的时间太多，既达不到目的，也浪费了人力成本。

## 给出具体的优化指标

应该给出一个清晰的优化指标。这个指标可以是响应时间的减小量，也可以是某个job运行时间的减小量，还可以是单位时间内系统完成的任务的增加量。

对于用户操作的响应时间，[Jakob Nielsen](https://www.nngroup.com/articles/author/jakob-nielsen/)曾在《Usability Engineering》中提到了三个重要的指标，用来描述响应的快慢: 0.1秒是让用户感觉到系统正在即时反应的极限，这意味着除了显示结果外，不需要特殊的反馈。1.0秒是用户的思想流保持不间断的极限，即使用户会注意到延迟。通常，在超过0.1秒但小于1.0秒的延迟期间不需要特殊反馈，但用户确实会失去直接操作数据的感觉。10秒是保持用户注意力集中在对话上的极限。对于较长的延迟，用户将希望在等待计算机完成时执行其他任务，因此应向他们提供反馈，指示计算机预期何时完成。如果响应时间可能是高度可变的，则延迟期间的反馈尤其重要，因为用户将不知道预期的结果。也就是说，如果延迟时间多于0.1秒，则有不平滑的感觉。如果多于1秒，则会感到软件速度缓慢。如果多余10秒，则会让用户沮丧。在设定优化指标时，可以参考这三个时间。

但时间的长短是一个客观的量，而对于性能的感受确是主观的。不同的系统功能，有着不同的应用场景，复杂度和用户期望。所以在给出优化指标的时候不能一概而论。这种情况下，可以用性能提高百分比来表示优化的指标。

## 从业务入手还是从技术入手

一个系统中，有些功能模块消耗较多资源，有些则消耗较少；有些用户用得多，有些用户用的少，甚至没有意义。在进行优化时应该首先思考并和业务人员沟通，系统中是否有一些没有意义但消耗资源多的功能模块，是否可以从系统中移除这些模块。如果能移除不必要的模块，既在短时间内完成了优化过程，也让整个系统变得更容易维护。