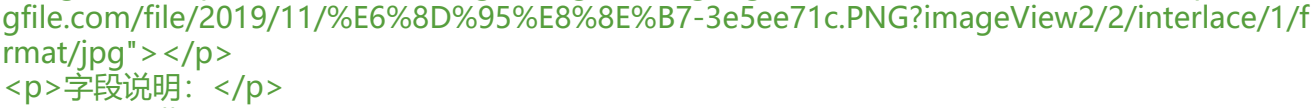
链滴

# Java 性能调优工具箱之操作系统的工具和分析

作者：AutisticV5

原文链接：https://ld246.com/article/1573353595802

来源网站：链滴

<ol>
<li>
CPU 使用率
通常 CPU 使用率可以分为两类：用户态时间和系统态时间。用户态时间是 CPU 执行应用代码所时间的百分比，而系统态时间则是 CPU 执行内核代码所占时间的百分比。系统态时间与应用无关，如应用执行 I/O 操作，系统就会执行内核代码从磁盘读取文件，或者将缓存数据发送到网络等。任何用底层系统资源的操作都会导致应用占用更多的系统态时间。
性能调优的目的是：在尽可能短的时间内让 CPU 使用率尽可能地高。
CPU 使用率是一段时间内的平均数——5 秒、30 秒，也可能只有 1 秒那么短。比如 10 分钟内个程序执行的 CPU 使用率为 50%。如果代码调优之后，CPU 使用率达到了 100%，说明程序的性能了倍，程序只需要执行 5 分钟就可以了。如果性能再翻倍，CPU 仍将是 100%，而执行完程序只要 2. 分钟。CPU 使用率表示程序以多高的效率使用 CPU，所以数字越大，性能越好。
如果再 Linxu 桌面系统上运行 vmstat 1，可以得到类似如下的几行信息（每隔 1 秒显示一行）<r>


字段说明：
procs(进程)
<table>
<thead>
<tr>
<th>字段名</th>
<th>字段名解释</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>运行队列中进程数量</td>
</tr>
<tr>
<td>b</td>
<td>等待 I/O 的进程数量</td>
</tr>
</tbody>
</table>
memory(内存)
<table>
<thead>
<tr>
<th>字段名</th>
<th>字段名解释</th>
</tr>
</thead>
<tbody>
<tr>
<td>swpd</td>
<td>使用虚拟内存大小(单位：B)</td>
</tr>
<tr>
<td>free</td>
<td>可用内存大小(单位：B)</td>
</tr>

<tr>
<td>buff</td>
<td>用作缓冲的内存大小(单位：B)</td>
</tr>
<tr>
<td>cache</td>
<td>用作缓存的内存大小(单位：B)</td>
</tr>
</tbody>
</table>
swap(交换内存)
<table>
<thead>
<tr>
<th>字段名</th>
<th>字段名解释</th>
</tr>
</thead>
<tbody>
<tr>
<td>si</td>
<td>每秒从交换区写到内存的大小</td>
</tr>
<tr>
<td>so</td>
<td>每秒写入交换区的内存大小</td>
</tr>
</tbody>
</table>
IO
<table>
<thead>
<tr>
<th>字段名</th>
<th>字段名解释</th>
</tr>
</thead>
<tbody>
<tr>
<td>bi</td>
<td>每秒读取的块数</td>
</tr>
<tr>
<td>bo</td>
<td>每秒写入的块数</td>
</tr>
</tbody>
</table>
system
<table>
<thead>
<tr>
<th>字段名</th>
<th>字段名解释</th>

</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>每秒中断数，包括时钟中断</td>
</tr>
<tr>
<td>cs</td>
<td>每秒上下文切换数</td>
</tr>
</tbody>
</table>
CPU(以百分比表示)
<table>
<thead>
<tr>
<th>字段名</th>
<th>字段名解释</th>
</tr>
</thead>
<tbody>
<tr>
<td>us</td>
<td>用户进程执行时间</td>
</tr>
<tr>
<td>sy</td>
<td>系统进程执行时间</td>
</tr>
<tr>
<td>id</td>
<td>空闲时间(包括 IO 等待时间)，中央处理器的空闲时间。以百分比表示</td>
</tr>
<tr>
<td>wa</td>
<td>等待 IO 时间</td>
</tr>
</tbody>
</table>
每秒内，CPU 被占用 300 毫秒(27% 的时间执行用户代码，3% 的时间执行系统代码)，相应地，PU 空闲 700 毫秒。CPU 空闲可能有一下原因：
<ul>
<li>
应用被同步原语阻塞，直至锁释放才能继续执行。
</li>
<li>
应用在等待某些东西，例如数据库调用所返回的相应。
</li>
<li>
应用的确是无所事事。
</li>
</ul>
前面两种情况通常都可用来识别某些问题。如果竞争降低，或优化数据库使之发送响应更快，程

运行都能变得更快，平均 CPU 使用率也会上升(当然得假设没有其他继续阻塞应用的问题)。
关于第三点，可能会有些困惑。如果应用有事情做(而不是等待锁或者其他资源而无所事事)，CPU 就会分配一些周期执行应用代码。这是一般性原则，而不只针对 Java。比如，包含无线循环的简单脚。这段脚本执行时，将消耗 100% 的 CPU。
ECHO OFF<br>
:BEGIN<br>
ECHO LOOPING<br>
GOTO BEGIN<br>
REM We never get here.....<br>
ECHO DONE
如果这段脚本没有消耗 100%CPU，那以为着操作系统还有些事可做——它可以答应一行 LOOPNG——却选择了空闲。这种情况下，空闲并没有什么好处，如果我们正在进行一些有用(耗时)的计，那么迫使 CPU 周期性空闲只会使我们得到响应的时间变得更长。
操作系统擅长为争用 CPU 周期的程序分配时间片，但新程序可用的 CPU 变少了，它也就运行得慢，所以基于这种经验，人们有时会认为，在其它程序可能需要 CPU 周期时预留一些空闲周期，没是个好主意，但操作系统无法猜到你接下来想做什么，所以(默认情况下)它会尽可能执行一切而不是让 CPU 空闲
</li>
<li>
CPU 运行队列
前面 vmstat 的输中，r 是所有正在运行或待运行的进程数。示例中至少有二个线程试图运行。行队列反应的是机器上所有东西的运行情况。如果试图运行的线程数超过了可用的 CPU，性能就会下。一般来说，Windows 的处理器队列长度最好为 0，小于或等于 Unix 系统 CPU 的数目。不过这也是硬性规定。有些系统或其他进程会周期性出现，在这瞬间数字会有提高，这对性能不会有实质性影。但是，如果想当长时间内运行队列很长，说明系统已经过载，这是你应该检查系统，减少机器正在理的工作量(将工作转移到其他机器或者优化代码)
</li>
</ol>
<h4 id="总结">总结</h4>
<ol>
<li>
检查应用性能时，首先应该审查 CPU 时间
</li>
<li>
优化代码的目的是提升而不是降低(在更短时间段内)CPU 使用率
</li>
<li>
在试图深入优化应用前，应该先弄清楚为何 CPU 使用率低
</li>
</ol>