

讲堂 > Linux性能优化实战 > 文章详情

13 | Linux 性能优化答疑（一）

2018-12-19 倪朋飞



13 | Linux 性能优化答疑（一）

朗读人：冯永吉 11'02" | 10.11M

你好，我是倪朋飞。

专栏更新至今，四大基础模块之一的 CPU 性能篇，我们就已经学完了。很开心过半数同学还没有掉队，仍然在学习、积极实践操作，并且热情地留下了大量的留言。

这些留言中，我非常高兴地看到，很多同学已经做到了活学活用，用学过的案例思路，分析出了线上应用的性能瓶颈，解决了实际工作中的性能问题。还有同学能够反复推敲思考，指出文章中某些不当或不严谨的叙述，我也十分感谢你，同时很乐意和你探讨。

此外，很多留言提出的问题也很有价值，大部分我都已经在 app 里回复，一些手机上不方便回复的或者很有价值的典型问题，我专门摘了出来，作为今天的答疑内容，集中回复。另一方面，也是为了保证所有人都能不漏掉任何一个重点。

今天是性能优化答疑的第一期。为了便于你学习理解，它们并不是严格按照文章顺序排列的。每个问题，我都附上了留言区提问的截屏。如果你需要回顾内容原文，可以扫描每个问题右下方的

二维码查看。

问题 1：性能工具版本太低，导致指标不全



威

写于 2018/11/23

为什么我的 `pidstat` 命令没有 `%wait` 列.....

引自：Linux性能优化实战

02 | 基础篇：到底应该怎么理解“平均负载”？



识别二维码打开原文
「极客时间」App

这是使用 CentOS 的同学普遍碰到的问题。在文章中，我的 `pidstat` 输出里有一个 `%wait` 指标，代表进程等待 CPU 的时间百分比，这是 `sysstat` 11.5.5 版本才引入的新指标，旧版本没有这一项。而 CentOS 软件库里的 `sysstat` 版本刚好比这个低，所以没有这项指标。

不过，你也不用担心。前面我就强调过，工具只是查找分析的手段，指标才是我们重点分析的对象。如果你的 `pidstat` 里没有显示，自然还有其他手段能找到这个指标。

比如说，在讲解系统原理和性能工具时，我一般会介绍一些 **proc 文件系统** 的知识，教你看懂 `proc` 文件系统提供的各项指标。之所以这么做，一方面，当然是为了让你更直观地理解系统的工作原理；另一方面，其实是想给你展示，性能工具上能看到的各项性能指标的原始数据来源。

这样，在实际生产环境中，即使你很可能需要运行老版本的操作系统，还没有权限安装新的软件包，你也可以查看 `proc` 文件系统，获取自己想要的指标。

但是，性能分析的学习，我还是建议你要用最新的性能工具来学。新工具有更全面的指标，让你更容易上手分析。这个绝对的优势，可以让你更直观地得到想要的数​​据，也不容易让你打退堂鼓。

当然，初学时，你最好试着去理解性能工具的原理，或者熟悉了使用方法后，再回过头重新学习原理。这样，即使是在无法安装新工具的环境中，你仍然可以从 `proc` 文件系统或者其他地方，获得同样的指标，进行有效的分析。

问题 2：使用 `stress` 命令，无法模拟 `iowait` 高的场景



张俊生

写于 2018/11/28

实测 `stress -i 1 --timeout 600`, 虽然提示
`dispatching hogs: 0 cpu, 1 io`, 通过
`mpstat - P ALL`, 反馈的仍旧是%sys 负载
高, 而不是%iowait???

引自: Linux性能优化实战

02 | 基础篇: 到底应该怎么理解“平均负载”?

识别二维码打开原文
「极客时间」App





白华

写于 2018/11/23

进行实验二 `stress -i 1 --timeout 600` 模拟 `sync`，平均负载确实上升了，但是在 `mpstst -P ALL 5 1` 查看是 `sys` 那一列接近 100% 而不是 `iowait`

引自：Linux性能优化实战

02 | 基础篇：到底应该怎么理解“平均负载”？

识别二维码打开原文
「极客时间」App



使用 `stress` 无法模拟 `iowait` 升高，但是却看到了 `sys` 升高。这是因为案例中的 `stress -i` 参数，它表示通过系统调用 `sync()` 来模拟 I/O 的问题，但这种方法实际上并不可靠。


因为 `sync()` 的本意是刷新内存缓冲区的数据到磁盘中，以确保同步。如果缓冲区内本来就没多少数据，那读写到磁盘中的数据也就不多，也就没法产生 I/O 压力。

这一点，在使用 SSD 磁盘的环境中尤为明显，很可能你的 `iowait` 总是 0，却单纯因为大量的系统调用，导致了系统 CPU 使用率 `sys` 升高。

这种情况，我在留言中也回复过，推荐使用 stress-ng 来代替 stress。担心你没有看到留言，所以这里我再强调一遍。

你可以运行下面的命令，来模拟 iowait 的问题。

```
1 # -i 的含义还是调用 sync，而-hdd 则表示读写临时文件
2 $ stress-ng -i 1 --hdd 1 --timeout 600
```

 复制代码

问题 3：无法模拟出 RES 中断的问题



许山山

写于 2018/11/28

老师我用的是 ubuntu 16.04 的 vps，单核、2G 内存，不论是否运行测试程序，RES 都是 0，in 没有明显变化，在 100~400 间波动，cs 从几百激增到十几万。我想问下问什么我这边没有出现明显的中断问题呢？而且我的测试程序也会很快的停止。是因为 vps 的配置太低了吗？

引自：Linux性能优化实战

04 | 基础篇：经常说的 CPU 上下文切换是什么意思？
(下)

识别二维码打开原文
「极客时间」 App



这个问题是说，即使运行了大量的线程，也无法模拟出重调度中断 RES 升高的问题。

其实我在 CPU 上下文切换的案例中已经提到，重调度中断是调度器用来分散任务到不同 CPU 的机制，也就是可以唤醒空闲状态的 CPU，来调度新任务运行，而这通常借助**处理器间中断**（Inter-Processor Interrupts, IPI）来实现。

所以，这个中断在单核（只有一个逻辑 CPU）的机器上当然就没有意义了，因为压根儿就不会发生重调度的情况。

不过，正如留言所说，上下文切换的问题依然存在，所以你会看到，cs（context switch）从几百增加到十几万，同时 sysbench 线程的自愿上下文切换和非自愿上下文切换也都会大幅上升，特别是非自愿上下文切换，会上升到十几万。根据非自愿上下文切换的含义，我们都知道，这是过多的线程在争抢 CPU。

其实这个结论也可以从另一个角度获得。比如，你可以在 pidstat 的选项中，加入 -u 和 -t 参数，输出线程的 CPU 使用情况，你会看到下面的界面：

复制代码

| | |
|---|--|
| 1 | \$ pidstat -u -t 1 |
| 2 | |
| 3 | 14:24:03 UID TGID TID %usr %system %guest %wait %CPU CPU Command |
| 4 | 14:24:04 0 - 2472 0.99 8.91 0.00 77.23 9.90 0 __sysbench |
| 5 | 14:24:04 0 - 2473 0.99 8.91 0.00 68.32 9.90 0 __sysbench |
| 6 | 14:24:04 0 - 2474 0.99 7.92 0.00 75.25 8.91 0 __sysbench |
| 7 | 14:24:04 0 - 2475 2.97 6.93 0.00 70.30 9.90 0 __sysbench |
| 8 | 14:24:04 0 - 2476 2.97 6.93 0.00 68.32 9.90 0 __sysbench |
| 9 | ... |

从这个 pidstat 的输出界面，你可以发现，每个 stress 线程的 %wait 高达 70%，而 CPU 使用率只有不到 10%。换句话说，stress 线程大部分时间都消耗在了等待 CPU 上，这也表明，确实是过多的线程在争抢 CPU。


在这里顺便提一下，留言中很常见的一个错误。有些同学会拿 pidstat 中的 %wait 跟 top 中的 iowait%（缩写为 wa）对比，其实这是没有意义的，因为它们是完全不相关的两个指标。

- pidstat 中，%wait 表示进程等待 CPU 的时间百分比。
- top 中，iowait% 则表示等待 I/O 的 CPU 时间百分比。

回忆一下我们学过的进程状态，你应该记得，等待 CPU 的进程已经在 CPU 的就绪队列中，处于运行状态；而等待 I/O 的进程则处于不可中断状态。


另外，不同版本的 sysbench 运行参数也不是完全一样的。比如，在案例 Ubuntu 18.04 中，运行 sysbench 的格式为：


```
1 $ sysbench --threads=10 --max-time=300 threads run
```

 复制代码

而在 Ubuntu 16.04 中，运行格式则为（感谢 Haku 留言分享的执行命令）：

```
1 $ sysbench --num-threads=10 --max-time=300 --test=threads run
```

 复制代码

问题 4：无法模拟出 I/O 性能瓶颈，以及 I/O 压力过大的问题



Brown 羊羊

写于 2018/12/05

没有模拟出来系统 I/O 瓶颈，可以帮忙看下吗：

容器运行起来后只发现一个 app 进程

```
[root@liyang2 ~]# ps aux|grep /app
```

```
root 23619 0.0 0.0 4368 380 pts/0 Ss+  
17:12 0:00 /app
```

```
root 23777 0.0 0.0 112648 952 pts/0  
S+ 17:12 0:00 grep --color=auto /app
```

CPU 情况 wa 也没有很高

```
%Cpu(s): 1.0 us, 1.5 sy, 0.0 ni, 94.0 id,  
3.3 wa, 0.0 hi, 0.2 si, 0.0 st
```

```
系统: redhat7 3.10.0-327.el7.x86_64  
x86_64 GNU/Linux
```

引自：Linux性能优化实战

07 | 案例篇：系统中出现大量不可中断进程和僵尸进程怎么办？（上）

识别二维码打开原文
「极客时间」App



这个问题可以看成是上一个问题的延伸，只是把 stress 命令换成了一个在容器中运行的 app 应用。

事实上，在 I/O 瓶颈案例中，除了上面这个模拟不成功的留言，还有更多留言的内容刚好相反，说的是案例 I/O 压力过大，导致自己的机器出各种问题，甚至连系统都没响应了。

之所以这样，其实还是因为每个人的机器配置不同，既包括了 CPU 和内存配置的不同，更是因为磁盘的巨大差异。比如，机械磁盘（HDD）、低端固态硬盘（SSD）与高端固态硬盘相比，性能差异可能达到数倍到数十倍。

其实，我自己所用的案例机器也只是低端的 SSD，比机械磁盘稍微好一些，但跟高端固态硬盘还是比不了的。所以，相同操作下，我的机器上刚好出现 I/O 瓶颈，但换成一台使用机械磁盘的机器，可能磁盘 I/O 就被压死了（表现为使用率长时间 100%），而换上好一些的 SSD 磁盘，可能又无法产生足够的 I/O 压力。

另外，由于我在案例中只查找了 /dev/xvd 和 /dev/sd 前缀的磁盘，而没有考虑到使用其他前缀磁盘（比如 /dev/nvme）的同学。如果你正好用的是其他前缀，你可能会碰到跟 Vicky 类似的问题，也就是 app 启动后又很快退出，变成 exited 状态。

Vicky🐣🐣🐣

写于 2018/12/11

老师，我按步骤执行后，ps里面显示exited了，也没有app进程，麻烦老师看一下

```
[root@VM_0_5_centos ~]# docker run --privileged --name=app -itd feisky/app:iowait
c6eaa823f3c7a8c4e29a8925fe424726840b1e2f2a568d22ba89966d8bdfb8fb
[root@VM_0_5_centos ~]# docker ps -a
```

| CONTAINER ID | IMAGE | COMMAND |
|---------------|-------------------|--------------|
| D | CREATED | STATUS |
| PORTS | NAMES | |
| c6eaa823f3c7 | feisky/app:iowait | "/app" |
| 3 seconds ago | Exited (1) | 1 second ago |
| go | app | |

引自：Linux性能优化实战

07 | 案例篇：系统中出现大量不可中断进程和僵尸进程怎么办？（上）

识别二维码打开原文
「极客时间」App



在这里，berryfl 同学提供了一个不错的建议：可以在案例中增加一个参数指定块设备，这样有需要的同学就不用自己编译和打包案例应用了。

berryfl

写于 2018/12/07

有些磁盘不是sd或者xvd前缀，比如AWS的新实例是以nvme作为前缀，建议留一个可选的命令行参数指定块设备，有需要的读者不用自己编译和打包

引自：Linux性能优化实战

08 | 案例篇：系统中出现大量不可中断进程和僵尸进程怎么办？（下）

识别二维码打开原文
「极客时间」App



所以，在最新的案例中，我为 app 应用增加了三个选项。

- -d 设置要读取的磁盘，默认前缀为 /dev/sd 或者 /dev/xvd 的磁盘。
- -s 设置每次读取的数据量大小，单位为字节，默认为 67108864（也就是 64MB）。

- -c 设置每个子进程读取的次数，默认为 20 次，也就是说，读取 20*64MB 数据后，子进程退出。

你可以点击 [Github](#) 查看它的源码，使用方法我写在了这里：

```
1 $ docker run --privileged --name=app -itd feisky/app:iowait /app -d /dev/sdb -s 67108864 -c 20
```

[复制代码](#)

案例运行后，你可以执行 `docker logs` 查看它的日志。正常情况下，你可以看到下面的输出：

```
1 $ docker logs app
2 Reading data from disk /dev/sdb with buffer size 67108864 and count 20
```

[复制代码](#)

问题 5：性能工具（如 `vmstat`）输出中，第一行数据跟其他行差别巨大

汤🐟🍲昱

写于 2018/12/10

```
procs -----memory----- ---swap-- -----io--
-- -system-- -----cpu-----
r b swpd free buff cache si so bi bo
in cs us sy id wa st
1 0 520 3831600 116 7871856 0 0 0
11 0 0 0 0 99 0 0
0 0 520 3831264 116 7871868 0 0 0
0 2285 2400 0 0 100 0 0
0 0 520 3830956 116 7871924 0 0 0
11 2591 2818 2 0 98 0 0
0 0 520 3830268 116 7871972 0 0 0
64 2440 2673 1 0 99 0 0
```

在使用vmstat 5 查看系统性能的时候，第一行cs，us小，之后数值都很大，这是为什么？

引自：Linux性能优化实战
04 | 基础篇：经常说的 CPU 上下文切换是什么意思？（下）

识别二维码打开原文
「极客时间」 App



这个问题主要是说，在执行 vmstat 时，第一行数据跟其他行相比较，数值相差特别大。我相信不少同学都注意到了这个现象，这里我简单解释一下。

首先还是要记住，我总强调的那句话，**在碰到直观上解释不了的现象时，要第一时间去查命令手册。**

比如，运行 man vmstat 命令，你可以在手册中发现下面这句话：

复制代码

```
1 The first report produced gives averages since the last reboot. Additional reports give informa
2 pling period of length delay. The process and memory reports are instantaneous in either case.
```

也就是说，第一行数据是系统启动以来的平均值，其他行才是你在运行 vmstat 命令时，设置的间隔时间的平均值。另外，进程和内存的报告内容都是即时数值。

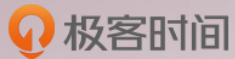
你看，这并不是什么不得了的故事，但如果我们不清楚这一点，很可能卡住我们的思维，阻止我们进一步的分析。这里我也不得不提一下，文档的重要作用。

授之以鱼，不如授之以渔。我们专栏的学习核心，一定是教会你**性能分析的原理和思路**，性能工具只是我们的路径和手段。所以，在提到各种性能工具时，我并没有详细解释每个工具的各种命令行选项的作用，一方面是因为你很容易通过文档查到这些，另一方面就是不同版本、不同系统中，个别选项的含义可能并不相同。

所以，不管因为哪个因素，自己 man 一下，一定是最快速并且最准确的方式。特别是，当你发现某些工具的输出不符合常识时，一定记住，**第一时间查文档弄明白。实在读不懂文档的话，再上网去搜，或者在专栏里向我提问。**

学习是一个“从薄到厚再变薄”的过程，我们从细节知识入手开始学习，积累到一定程度，需要整理成一个体系来记忆，这其中还要不断地对这个体系进行细节修补。有疑问、有反思才可以达到最佳的学习效果。

最后，欢迎继续在留言区写下你的疑问，我会持续不断地解答。我的目的仍然不变，希望可以和你一起，把文章的知识变成你的能力，我们不仅仅在实战中演练，也要在交流中进步。



Linux 性能优化实战

10 分钟帮你找到系统瓶颈

倪朋飞

微软资深工程师
Kubernetes 项目维护者



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

©版权归极客邦科技所有，未经许可不得转载

上一篇 12 | 套路篇：CPU 性能优化的几个思路

写留言

精选留言



ninuxer

6

打卡day14

之前一直理解有误，感谢指出！

pidstat 中，%wait 表示进程等待 CPU 的时间百分比。此时进程是运行状态。

top 中，iowait% 则表示等待 I/O 的 CPU 时间百分比。此时进程处于不可中断睡眠态。

等待 CPU 的进程已经在 CPU 的就绪队列中，处于运行状态；而等待 I/O 的进程则处于不可中断状态。

2018-12-19

作者回复



2018-12-19



郭江伟

1

课程很系统，把自己以前的知识都串起来了，后续争取每个案例自己都做一次，并且融合自己的经验改进下案例

2018-12-19

作者回复

欢迎分享你的改进经验😊

2018-12-19



念你如昔

👍 1

非常非常感谢，这钱花的值，之前没有对这些东西形成体系，老是感觉有力使不上感觉，自从看了老师的文档，终于飘了，都想跳槽了?!。

2018-12-19

作者回复



2018-12-19



我来也

👍 1

[D13打卡]

多谢老师提出来, pidstat 和 top 中的 %wait 含义并不一样.
之前只知道top是io的wait, 而新接触的pidstat的倒没有细想过.
确实是应该多man一下,看下命令文档.
刚开始要把工具用起来,之后再查看命令的详细文档.

2018-12-19

作者回复

嗯嗯，虽然专栏里也有不少使用案例，但并能包括所有细节的知识，这都需要查文档

2018-12-19



MoFanDon

👍 1

做了几年运维一直想要掌握，却了解的很零散。这段时间的课程让我学习很多，感谢老师。

2018-12-19



dexter

👍 0

```
[root@localhost ~]# docker run --name phpfpn -itd --network container:nginx feisk y/php-fpm
```

flag provided but not defined: --network

See 'docker run --help'.

```
[root@localhost ~]# docker version
```

Client version: 1.7.1

Client API version: 1.19

Go version (client): go1.4.2

Git commit (client): 786b29d/1.7.1

OS/Arch (client): linux/amd64

Server version: 1.7.1

Server API version: 1.19

Go version (server): go1.4.2
Git commit (server): 786b29d/1.7.1
OS/Arch (server): linux/amd64
centos6.10 安装发现没有--network参数

2018-12-20



虎虎♡

👍 0

虽然不讲各个工具的各项参数，我是很赞同的。但是，像从来没有接触过性能工具的我，根本没有意识到pidstat 和 top的wait是不同的。

是不是pidstat所有输出，都是以进程时间为基础，而top中都是以cpu时间为基础呢？

希望作者多提点一下我们新手哈。

2018-12-20



奋斗的菜鸟
打卡

👍 0

2018-12-20



湖湘志
D13

👍 0

2018-12-20



风飘，吾独思
打卡

👍 0

2018-12-19



0.0
execsnoop无法安装

👍 0

2018-12-19



腾达

👍 0

perf report 显示的swapper [kernel.kallsyms] 这类信息要排除吗？自己查了一下，是内核的符号，在排障的时候要忽略不看吗？

2018-12-19



W.T

👍 0

老师，等待IO的不可中断进程是否一直占用CPU？

2018-12-19



清晓

👍 0

老师你好，最近遇到一个问题：目录下的jar包，都安装好了，怎么用命令查看用到哪些应用或软件。所有应用/软件都会显示版本号，路径，配置，等，是在一个页面显示的。

只是知道进到jar包的目录下，然后执行命令，就会出现一个页面，包含用到，比如Redis，zookeeper等的路径，版本号，等一些信息。

急需老师帮忙，这命令是什么？

望老师回复，万分感谢！

2018-12-19



饼子
继续学习

👍 0

2018-12-19



大兵

👍 0

@walker 你的阿里云服务是否用了Nginx做了反向代理？这是由于nginx代理使用了短链接的方式和后端交互的原因，使得nginx和后端的 ESTABLISHED变得很少而TIME_WAIT很多. 解决方案可参考 (<http://www.phpfans.net/article/htmls/201010/MzEyMTU0.html>) 这篇文章.

2018-12-19



walker

👍 0

在实际中遇到一个问题，阿里云的服务器，访问比较少TIME_WAIT已经达到阿里云设置的上限。但是负载，内存，io都不高，ESTABLISHED也只有300。si从0.3到3，这个应该从哪些方面入手去定位根源呢

2018-12-19

作者回复

网络，网络也会讲的

2018-12-19



小文
打卡

👍 0

2018-12-19



Geek_101627
打卡

👍 0

2018-12-19



万万
打个卡

👍 0

2018-12-19